Multi-body modeling of robot dynamics and system identification during MPC

Thesis submitted for the degree of
Master of Science in Scientific Computing

Horea-Alexandru Cărămizaru

Supervised by:
Prof. Dr. Ekaterina Kostina
Prof. Dr. Dr. h. c. mult. Hans Georg Bock

Faculty of Mathematics and Computer Science
Ruprecht-Karls-Universität Heidelberg

Heidelberg, January 4, 2023
(*defended October 27, 2022*)

To my parents and my sisters, for their endless love and support.

To Eliza, for letting me have the chance to chase my dreams,
you have been my beacon of hope, my source of inspiration.

And to my true friends, may we all find a few!

Abstract

Due to external influences over parameters that characterize dynamical systems, an online parameter estimation must be added as part of model predictive control strategies. In this thesis, we show how continuous parameters estimation, using inverse dynamics, can be used for identifying the inertial parameters (mass, inertia, and center of mass) of multi-body systems as part of an adaptive control strategy. For this, a Featherstone spatial algebra equivalent model, based on screw theory was used. The system identification was done using a linear least squares approach using the Recursive Newton-Euler Algorithm as a way of implementing a generic solution. The process is for open-loop robots and is tested using an optimal control algorithm based on multiple shooting.

Contents

1 Introduction **5**
- 1.1 Motivation . . . 6
- 1.2 Organization . . . 7
- 1.3 Contributions . . . 7

2 Related work **9**
- 2.1 Automatic Differentiation . . . 9
- 2.2 Multi-body dynamics . . . 10
- 2.3 Optimal Control/MPC . . . 10
- 2.4 System identification . . . 11

3 Multi-body Screw Theory **13**
- 3.1 Rigid body kinematics . . . 13
  - 3.1.1 Rigid body Configuration . . . 13
  - 3.1.2 Rigid body Spatial Velocity (Twist) . . . 16
  - 3.1.3 Screw motion . . . 20
  - 3.1.4 Operator View of Rigid-Body Transformation . . . 22
  - 3.1.5 Exponential coordinate of Rigid Body Configuration . . . 28
  - 3.1.6 Instantaneous Velocity of Moving Frame . . . 30
  - 3.1.7 Kinematics of Open Chain, Product of Exponential . . . 32
  - 3.1.8 Velocity Kinematics . . . 36
  - 3.1.9 Spatial Acceleration . . . 37
  - 3.1.10 Plücker Coordinate System and Basis Vector . . . 38
- 3.2 Rigid Body dynamics . . . 42
  - 3.2.1 Spatial Force (Wrench) . . . 42
  - 3.2.2 Spatial Momentum . . . 45
  - 3.2.3 Newton-Euler Equation . . . 50
- 3.3 Multi-body dynamics of Open Chains . . . 52

3.3.1 Open-Chain Dynamics . . . 52
3.3.2 Inverse Dynamics . . . 53
3.3.3 Forward Dynamics . . . 56

4 System identification **58**
4.1 The analytical, algorithmically constructed, equation . . . 58
4.2 Parametric estimation: inertia matrix, mass, the center of mass . 62

5 Experimental setup **67**
5.1 Environment definition . . . 67
5.2 Sensitivity analysis . . . 68
5.3 Robot definition . . . 68
5.3.1 URDF file structure . . . 68
5.3.2 ODE parameter extraction . . . 71
5.4 Model Predictive Control . . . 73
5.4.1 MPC flow . . . 73
5.4.2 Optimal Control . . . 73
5.4.3 Configuration . . . 76

6 Tests **77**
6.1 ODE comparation . . . 77
6.1.1 Dynamics functionalities . . . 77
6.1.2 Time computation differences . . . 78
6.2 Derivatives - sensitivity analysis time computation . . . 82
6.3 System identification . . . 83
6.4 Videos . . . 83

7 Conclusions and future work **84**

References **86**

List of Figures

3.1 **Point** *vs.* **Vector** in different reference frames 14
3.2 Rigid body configuration 15
3.3 Rigid transformation 16
3.4 Linear velocity 17
3.5 Linear velocity generalization 18
3.6 Spatial Velocity in reference frame 19
3.7 Screw motion [31] 20
3.8 From Screw Motion to Twist 21
3.9 Rotation Operation via Differential Equation 23
3.10 Rigid-body Operation via Differential Equation 27
3.11 Instantaneous Velocity of Rotating Frame 30
3.12 Instantaneous Velocity of Moving Frame 32
3.13 Moving Frame 33
3.14 Forward Kinematics [31] 33
3.15 Product of exponentials [31] 34
3.16 Product of exponentials, $n = 2$ 35
3.17 Spatial Acceleration 37
3.18 Velocity in different frames 38
3.19 Rigid body 42
3.20 spatial force coordinate transformation 43
3.21 Joint Torque 44
3.22 Point-mass 45
3.23 Angular momentum about an arbitrary point 47
3.24 Spatial inertia mapping 48
3.25 Newton-Euler in inertial frame 51
3.26 Open Chain 53
3.27 Recursive Newton-Euler Algorithm 55

4.1 Open Chain 59

4.2 Open Chain-2 links 60
4.3 3 DOF robot - structural sparsity - τ 64
4.4 3 DOF robot 65
4.5 3 DOF robot - structural sparsity - $\mathcal{F}$ 66
4.6 Universal Joint [23] 66

5.1 Link [21] 69
5.2 Joint [22] 70

6.1 Franka Emika Panda 79
6.2 Double pendulum 79
6.3 Chain 79
6.4 $\tilde{C}$ evaluation 80
6.5 ID evaluation 80
6.6 *Inertia* evaluation 81
6.7 Derivative computation *vs.* Function computation 82
6.8 System identification 83

CHAPTER 1

Introduction

Eu nu strivesc corola de minuni a lumii
şi nu ucid
cu mintea tainele, ce le-ntâlnesc
în calea mea
în flori, în ochi, pe buze ori morminte.
...
Lucian Blaga [1]

The fascination of humankind with the way things move in nature, in general, and the animal movement, in particular, has its roots deep into the beginning of history. Over the years, we have tried to *mimic* it as much as possible hoping that, with every improvement, the end result to be closer to what we observe around us and thus the fine line, between knowing *why* things work the way they do and *how* they work, to become blurred up to the point where no one care to distinguish between them anymore. This endeavor has culminated nowadays with the birth of an interdisciplinary area that brings together *state-of-the-art* results, in domains, such as numerical optimization, model predictive control, mathematical modeling of complex dynamical systems (*e.g.* multi-rigid-body dynamics), robotics or machine learning.

[1] `https://blog.caramizaru.xyz/doku.php?id=blog:2023:03:17_eu_nu_strivesc_corola_de_minuni_a_lumii`

1.1 Motivation

It can be easily said, for a long time, the struggle was in finding better models to express the movement of systems of rigid bodies. This has put us on a path of new discoveries and nowadays, we have different, equivalent ways, to describe the dynamics, from **classical Newtonian mechanics** to more modern approaches like **Lagrange** and **Hamiltonian formalism**, each of them having its advantages and disadvantages.

Having a good mathematical model of a dynamical system can be helpful most of the time, since, empirically, one can state that a better model approximation can have a positive impact whenever a control strategy is in place, nonetheless, the control process methodology can have its own influence on what is and what is not a good model. In this regard, a good example can be the degrees of freedom as well as the state space representation of a dynamic system. Though it might seem that using a Lagrangian/Hamiltonian formalism will be more suitable for control, since it is using a **reduced** representation that can be directly used, when it comes to mechanical systems, this approach also comes with the disadvantage of redesigning the **ODE/DAE** whenever, during the process, discontinuities, in the model, are in place. This approach doesn't scale well as it requires human intervention for defining each phase manually. Also, we can observe that classical Newtonian mechanics comes with challenges as well, as the **redundant** form, which characterizes this approach, will fail to provide a right-hand side representation of the ODE required by most of the optimal control strategies. This limitation can be overcome by factorizing the system, using a numerical change of variable (*e.g.* factorization of the dynamical system using **SVD decomposition** in the direction of **actuated degrees of freedom**), this approach is heavily exposed to **numerical instability**.

One way to solve this problem is to try to come up with a different model that inherits the best characteristics of both worlds, reduced and redundant representation, for the dynamical systems.

Even though the mathematical modeling of dynamical systems, as well as the control strategy used, represents the backbone of any approach that hopes to deliver an automatic system, the final result is also strongly influenced by the quality of the parameters, the model is using. In general, the system identification applied *a priori* is not enough since the proprieties of the system are sensitive to external influences and changes over time. This results, again and again, in worse control prediction after a while.

The obvious solution to this problem is to extend the **model-prediction-control** approach by integrating, a **parametric estimation** phase as part of the loop. The result of this approach is the use of an updated model version during control prediction which is hoped to behave better.

1.2 Organization

We are starting with Chapter 2, Related work, where we draw the main lines of the current research done in the field of **automatic differentiation**, **optimal control**, **multi-body dynamics**, and **system identification** while, at the same time, we put into the context the results of this thesis. We then continue with Chapter 3, Multi-body Screw Theory, which introduces a **Screw Theory** formalism for mechanics and builds the prerequisites for the multi-body dynamics as well. In Chapter 4, System identification we show how the **Recursive Newton-Euler Algorithm** can be reinterpreted, in order to compute, in an algorithmic way, the equivalent analytical equation used for identifying the parameters that characterize each of the robot's links. Chapter 5, Experimental setup, describes the testing environment, the necessary tools used, the overall flow of the **MPC** and **system identification**, the sensor simulation and data acquisition, as well as the preprocessing steps done *a priori* that are required in the building process of the ODE that characterizes any multi-body system. Chapter 6, Tests puts the proposed solutions into context by comparing them with the current *state-of-the-art*. We conclude this thesis, with the last chapter, Chapter 7, Conclusions and future work.

1.3 Contributions

The main results of this thesis are as follows:

- Introducing a mathematical framework, a **Featherstone Algebra equivalent** formulation of multi-body dynamics, in a bottom-up manner, based on **Screw Theory** geometric perspective. The accent is put on geometric intuition first, even though, all the concepts are introduced in an algebraic, rigorous manner, that can be directly applicable afterward.
- Showing how, the introduced framework, can be used to overcome the necessity of manual define, **multi-phase system**, for multi-body systems.
- Showing the equivalence of the **Recursive Newton-Euler Algorithm** with an analytical equation, in reduced representation, that resembles the **Lagrangian mechanics formalism**, and how this approach can be used to come up with a **linear least square**, **system identification** approach, for the parameters that characterize a multi-body-system; an approach that, for this specialized ODE, is faster and offers better results than the more classical and more general **multiple shooting, nonlinear least squares**.
- Showing that the parameter identification is **path independent** and how **only 2 consecutive data acquisitions** can be sufficient.
- An implementation for a generic ODE generation of open-loop, multi-body dynamical systems, starting from **URDF** format representation; a

generic system identification of rigid bodies and a generic model predictive control, based on multiple shooting. The framework is available in Python and is built using **CasADi** framework for **sensitivity computation**.

CHAPTER 2

Related work

2.1 Automatic Differentiation

In the world of robotics, in the last 2 decades, we can observe a strong shift in adopting more complex tools, for modeling and optimization. This is necessary to address the increase in complexity, demanded by the current research objectives. In this regard, important work has been done in building new **algorithmic differentiation** tools moving from manual integration of automatic differentiation [6], to frameworks that offer full symbolical front-end used in writing differentiable algorithms [1]. Moreover, new **differentiable languages**, build with numerical computation in mind, have been proposed [7]. Due to the high demand for Python in the scientific community, different ways of making current Python code, differentiable, have been proposed (*e.g.* **JAX**). At the same time, from the **Machine Learning** community, we get to see new automatic differentiable tools as well, one of the most visible being **Pytorch** [36]. These tools have the advantages, among others, of having back-ends for CPU and GPU as well, but they are not specialized for optimal control. A current, state of the art, of **AD**, can be found in [32] and [5], also, foundation work in this field can be found in [20].

For this thesis, **CasADi** was used which is designed as an **optimization and optimal control mathematical modeling framework**. This project is language agnostic (the project is written in C++ and it offers **Python** and **Matlab** binding) and it can be used to generate C++ code once the problem is modeled using a height-level language. The design resembles a functional language and it offers the means to write vectorizable code in a natural way. At the same time, it offers access to the *state-of-the-art*, **Sundials suits**, **sensitivity computation**, and **nonlinear optimization tools**.

2.2 Multi-body dynamics

Throughout the years, related rigid-body simulation work could be found spread into multiple disciplines. We can observe work done in **computer graphics and animation**, particularly in **physics-based animation**, in **molecular dynamics** [39], in the **robotics** community as well as in the **optimization and control** community.

Given the specifics of each area, one can observe different developments of the current state of the art.

In computer graphics, we see overwhelmingly, solutions based on **redundant** systems [12]. This is due to the nature of the end goal, which in this case is the simulation itself for the entertainment industry (movie and game industry). In this regard, we can mention the work of Baraff, who has advocated the **physically plausible** approach for rigid body simulation [4].

On the opposite side, in the robotics and optimization community, solutions based on a **unified geometric representation** of the degrees of freedom are favored [35]. Among others, this approach simplifies the solver dramatically as you don't have to take care of rotation and translation separately as it is explained in [26]. In this regard, the foundation of **Lie groups/Lie algebra** for robotics used was synthesized in [34].

Another equivalent approach, used often in reduced multi-body dynamics simulation is using **Spatial Operator Algebra** [38]. This was popularized by **Featherstone** in [17] and it is the approach that most of the current frameworks have adopted.

Another important aspect of the multi-body frameworks in robotics as well as optimizations and control community is the way they provide access to **sensitivity computation**. Some, [18] [39], don't have internal sensitivity computation capability. Those who have, are either built on top of an **automatic differentiation** tool (like the implementation introduced by this thesis [10]), [24], or they provide an internal **analytical derivative** [11] [33]. In the past, solutions based on automatic differentiation resembled results equivalent to what analytical solutions could offer with the disadvantage that re-computation of sensitivity had to be done in case the model changed. Recent results [40] [41] have improved the sensitivity time computation up to a constant factor compared with automatic differentiation solutions.

2.3 Optimal Control/MPC

The application of **optimal control** for multi-body systems has seen practical results, at least since the late 80s, in computer animation, when Andrew Witkin and Michael Kass published [46], an approach of generating automatic, physical-based animation by characterizing the desired motion through **trajectory constraints**.

Later, **direct methods**, based on **multiple shooting** [8], in optimal control of multi-body systems were adopted as they are desirable when the model has

many degrees of freedom.

For systems that need to recompute the trajectory as often as possible, due to the change of model with time, the model predictive control needs to be computed as fast as possible. A real-time iteration scheme [14][15] it's one way to minimize the optimal control time computation for each iteration. An efficient implementation can be found in **MUSCOD-II** [30] [13].

Extensive work in trajectory optimization through non-smooth domains has been done in [16], which utilizes a **smooth contact** model thus no hybrid method is necessary. Another approach, that treats contact through inelastic impacts and Coulomb friction was proposed in [37].

Recently, multiple-shooting solutions based on inverse dynamics have been proposed [25]. These approaches have the potential of being more stable and faster than the solutions based on forward simulation since the linear time, forward dynamics tend to have a higher constant than the inverse dynamics based on the Recursive Newton-Euler Algorithm.

2.4 System identification

When it comes to **parametric estimation** in a multi-body system, the general approach, using forward simulation with multiple shooting, is not the best approach as this method requires the computation of the **mass inverse** which results in a loss of the particular **affine structure** in **lumped parameters**, the inverse dynamics have[43]. Though it might not seems that obvious at first, system identification in lumped parameters is not sub-optimal as not all the inertial parameters are necessary most of the time for a full characterization of the dynamics [43]. A generalization for full inertial parameter estimation was initially introduced in [3].

This approach doesn't come without its own limitations, like for example, the impossibility of computing the parameters for the first link and, sometimes even for the second one [43].

Another limitation that needs to be taken into consideration is the use of **acceleration**, which, most of the time is computed by using a numerical differentiation of the measured velocity. Solutions, that can avoid using **torque** and **acceleration** are based on the **total energy** of a system since it is also affine in the lumped parameters. This approach can also be adopted for the method proposed in this thesis [43].

Important results for parametric estimation can be found also in the **adaptive control** field where it has been shown that is not necessary to find a solution that **converges** to the real parameters in order to control a system. Results that contain guarantees for reaching objectives without converging to the true parameters can be found in [2].

When one has to take into consideration the **sensor measurements errors** as well, the result of the linear least squares system identification, though correct from a mathematical point of view, might not end up being **physically plausible** (*e.g.* inertia must be positive definite, mass must be positive, etc.)

A way to deal with this problem is to add supplementary **constraints for parameters** as part of the estimation for the result to become feasible [44] [45]. Another approach is to plan for the trajectory that minimizes the error induced by noise [19].

CHAPTER 3

Multi-body Screw Theory

3.1 Rigid body kinematics

3.1.1 Rigid body Configuration

Definition 3.1.1 (Free Vector) *A geometric quantity with length and direction.*

Remark 3.1.1 *Given a **reference frame** $\{A\}$, $\vec{v}$ can be moved to a position such that the base of the arrow is at the origin without changing the orientation. The vector $\vec{v}$ can be represented by its **coordinates** ${}^{A}v$ in the reference frame $\{A\}$*

Remark 3.1.2 *$\vec{v}$ denotes the **physical quantity** while ${}^{A}v$ denotes the coordinates with respect to $\{A\}$*

Definition 3.1.2 (Point representation) *A point p is represented by a vector with respect to the **origin** of a **reference frame**.*

Remark 3.1.3 *${}^{A}p$ denotes the **coordinates** of a point p w.r.t. frame $\{A\}$ Fig. 3.1*

Definition 3.1.3 (Cross product) ***Cross product** of $a \in \mathbb{R}^3$, $b \in \mathbb{R}^3$ is defined as: Eq. 3.1.*

$$a \times b = \begin{bmatrix} a_2 \cdot b_3 - a_3 \cdot b_2 \\ a_3 \cdot b_1 - a_1 \cdot b_3 \\ a_1 \cdot b_2 - a_2 \cdot b_1 \end{bmatrix} \tag{3.1}$$

with the following properties:

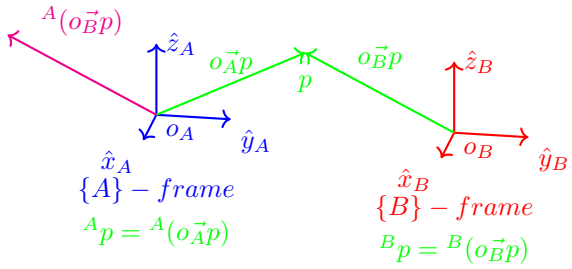


Figure 3.1: **Point** *vs.* **Vector** in different reference frames

- $||a \times b|| = ||a|| \cdot ||b|| \cdot \sin(\theta)$
- $a \times b = -b \times a$
- $a \times a = 0$

Definition 3.1.4 (Skew symmetric representation) ***Cross product*** *can be interpreted as a linear transformation,* $a \times b = [a] \cdot b$ *where:*

$$[a] \triangleq \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix} \tag{3.2}$$

with the following properties:

- $[a] = -[a]^T$
- $[a] \cdot [b] = [a \times b]$ ***(Jacobi's identity)***
- $a = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} \iff [a]$

Definition 3.1.5 (Reference frame) *3 coordinates, unit length, vectors* $\hat{x}, \hat{y}, \hat{z}$ *and fixed origin with the following properties:*

- $\hat{x}, \hat{y}, \hat{z}$ *mutually orthogonal vectors*
- $\hat{x} \times \hat{y} = \hat{z}$ ***right hand rule***

Definition 3.1.6 (Rotation matrix) *R is a* ***rotation matrix*** *if is a matrix that specifies the orientation of one frame, e.g.*$\{B\}$*-frame, relative to another frame, e.g.:*$\{A\}$*-frame: Eq. 3.3.*

$${}^{A}R_B = \begin{bmatrix} {}^{A}\hat{x}_B & {}^{A}\hat{y}_B & {}^{A}\hat{z}_B \end{bmatrix} \tag{3.3}$$

where R satisfies the ***Spatial Orthogonal Group*** *properties: Eq. 3.4.*

$$SO(3) = \{R \in \mathbb{R}^{3\times 3} : R^T \cdot R = I, \det(R) = 1\} \tag{3.4}$$

Remark 3.1.4 *Another interpretation of 3.1.6 is that ${}^{A}R_{B}$ represents the change of reference frame from $\{B\}$-frame to $\{A\}$-frame for a given vector $\vec{v}$: Eq. 3.5.*

$$ {}^{A}v = {}^{A}R_{B} \cdot {}^{B}v \tag{3.5}$$

Definition 3.1.7 (Rigid body transformation) *Given two coordinate frames $\{A\}$-frame and $\{B\}$-frame, the **configuration** of B relative to A is determined by: ${}^{A}R_{B}$, the rotation matrix from $\{B\}$-frame to $\{A\}$-frame and ${}^{A}o_{b}$, a point that represents the **origin** of $\{B\}$-frame expressed in $\{A\}$-frame: Fig. 3.2.*

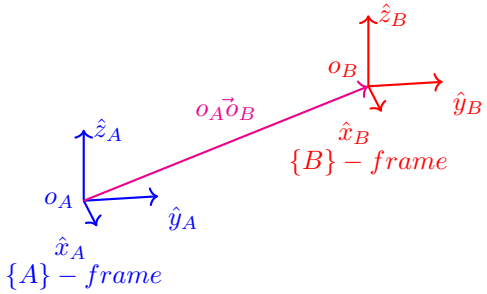


Figure 3.2: Rigid body configuration

Remark 3.1.5 (Rigid transformation of a vector) *Given a vector (**free vector**) $\vec{r}$, its coordinates ${}^{A}r$ and ${}^{B}r$ in $\{A\}$-frame and $\{B\}$-frame are related by the **liner transformation** Eq. 3.6.*

$$ {}^{A}r = {}^{A}R_{B} \cdot {}^{B}r \tag{3.6}$$

Remark 3.1.6 (Rigid body transformation of points) *Given a point p, its coordinates ${}^{A}p$ and ${}^{B}p$ are related by the **affine transformation** Eq. 3.7.*

$$ {}^{A}p = {}^{A}o_{B} + {}^{A}R_{B} \cdot {}^{A}p \tag{3.7}$$

***Proof**:*
From Fig. 3.3 we have the coordinate free equlity: Eq. 3.8

$$ o_{\vec{A}}p = o_A\vec{o}_B + o_{\vec{B}}p \tag{3.8}$$

And by using $\{A\}$-frame we get:

$$ {}^{A}(o_{\vec{A}}p) = {}^{A}p $$
$$ {}^{A}(o_A\vec{o}_B) = {}^{A}O_B $$
$$ {}^{A}(o_{\vec{B}}p) = {}^{A}R_{B} \cdot {}^{B}p $$

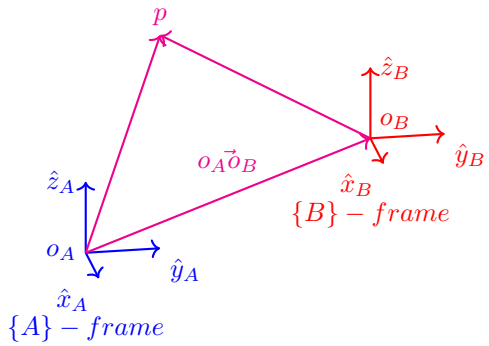


Figure 3.3: Rigid transformation

Definition 3.1.8 (Homogeneous Transformation) *For any affine transformation 3.7 there exist a corresponding* ***linear transformation*** *build using a* ***homogeneous transformation matrix*** ${}^{A}T_{B} \in \mathbb{R}^4$*: Eq. 3.9.*

$$ {}^{A}T_{B} = \begin{bmatrix} {}^{A}R_{B} & {}^{A}o_{B} \\ 0 & 1 \end{bmatrix} \tag{3.9} $$

Definition 3.1.9 (Homogeneous coordinates for points) *Given a point* $p \in \mathbb{R}^3$*, its* ***homogeneous coordinates*** *is given by Eq. 3.10.*

$$ \tilde{p} = \begin{bmatrix} p \\ 1 \end{bmatrix} \tag{3.10} $$

and it's corresponding linear transformations is: ${}^{A}\tilde{p} = {}^{A}T_{B} \cdot {}^{B}\tilde{p}$

Remark 3.1.7 (Homogeneous coordinates for vectors) *Given a vector* $v \in \mathbb{R}^3$*, its corresponding* ***homogeneous coordinates*** *are given by the Eq 3.11.*

$$ \tilde{v} = \begin{bmatrix} v \\ 0 \end{bmatrix} \tag{3.11} $$

since $\tilde{v} = \tilde{p}_1 - \tilde{p}_2$ *(where* $\tilde{p}_i \in \mathbb{R}^4$ *is a point). Hence,* $\tilde{v}_4 = 0$*.*

3.1.2 Rigid body Spatial Velocity (Twist)

Definition 3.1.10 (Linear velocity) *Given a* ***rigid body****, with* ***angular velocity*** ω ***(free vector)*** *and suppose the actual rotation axis passes through a point* p *for which the velocity is* v_p*, Fig. 3.4, one can compute the velocity for any* ***body-fixed point*** q *(a point that is rigidly attached to the body and moves with the body) as follows: Eq. 3.12.*

$$ v_q = v_p + \omega \times (\vec{pq}) \tag{3.12} $$

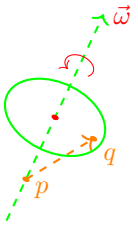


Figure 3.4: Linear velocity

Remark 3.1.8 *Given that a rigid body is characterized by an infinite number of points, each of them with different velocities, the Eq. 3.12 introduces a unique parameterization of the velocity of an arbitrary* ***body-fix point*** q *which depends only on* (ω, v_p, p).

Remark 3.1.9 *The Eq. 3.12 is* ***coordinate free****.*

Definition 3.1.11 (Generalization of Definition: 3.1.10) *Given Fig. 3.5, and an arbitrary point* r *in space, where:*

- r *might not be on the rotation axis*
- r *may be a stationary point in space*
- v_r *be the* ***velocity of the body-fixed point that currently coincides with*** r

we still have Eq. 3.13.

$$v_q = v_r + \omega \times (\vec{rq}) \tag{3.13}$$

since:

$$\begin{aligned}
\vec{r}_r &= \vec{v}_p + \vec{\omega} \times (\vec{pr}) \\
\vec{v}_q &= \vec{v}_p + \vec{\omega} \times (\vec{pq}) \\
&= \vec{v}_r - \vec{\omega} \times (\vec{pr}) + \omega \times (\vec{pq}) \\
&= \vec{v}_r + \omega \times (\vec{pq} - \vec{pr}) \\
&= \vec{v}_r + \vec{\omega} \times (\vec{rq})
\end{aligned}$$

Remark 3.1.10 *Using Definition 3.1.11, the body can be regarded as* ***translating*** *with* ***linear velocity*** v_r *while* ***rotating*** *with* ***angular velocity*** ω *about an axis passing through* r.

Definition 3.1.12 (Spatial Velocity (Twist) - coordinates free) *We define the* ***Spatial velocity*** *as the* ***free vector*** $\mathcal{V}_r = (\omega, v_r) \in \mathbb{R}^6$ *where:*

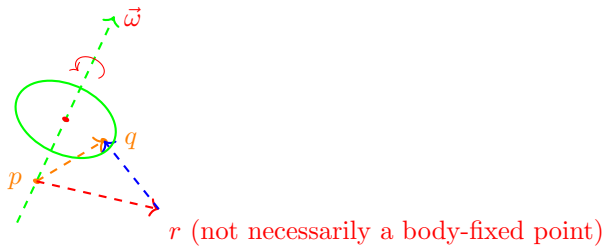


Figure 3.5: Linear velocity generalization

- ω *is the angular velocity*
- v_r *is the velocity of the body-fixed point that currently coincides with* r
- *for any other body-fixed point* q*, its velocity is:*

$$v_q = v_r + \omega \times \vec{rq}$$

Remark 3.1.11 ***Twist*** *is a* ***physical*** *quantity, just like* ***linear*** *or* ***angular velocity*** *and it can be represented in any frame for any chosen reference point* r*.*

Remark 3.1.12 *We can end up with the same conclusion for* ***twists*** $\mathcal{V}_r = (\omega, v_r)$ *as the one introduced in remark: 3.1.10.*

Remark 3.1.13 *For coordinates representation of* ***twists*** *reference a frame must be chosen.*

Definition 3.1.13 (Spatial Velocity reference frame representation) *Given a frame* $\{O\}$ *and a spatial velocity* $\mathcal{V} \in \mathbb{R}^6$*, w.l.o.g. one can choose o, the origin of* $\{O\}$*-frame as the reference point to represent the rigid body velocity Fig. 3.6. Then, the coordinates for* $\mathcal{V}$ *in* $\{O\}$*-frame is Eq. 3.14.*

$$^{O}\mathcal{V}_o = (^{O}\omega, ^{O}v_o) \tag{3.14}$$

If not specified otherwise, one should assume the ***origin of the frame*** *used as the* ***reference point*** *Eq. 3.15.*

$$^{O}\mathcal{V} = ^{O}\mathcal{V}_o \tag{3.15}$$

Definition 3.1.14 (Change Reference Frames for Twists) *Given a* ***twist*** $\mathcal{V}$*, let* $^{A}\mathcal{V}$ *and* $^{B}\mathcal{V}$ *be their coordinates in* $\{A\}$*-frame and* $\{B\}$*-frame Eq. 3.16.*

$$^{A}\mathcal{V} = \begin{bmatrix} ^{A}\omega \\ ^{A}v_A \end{bmatrix}, ^{B}\mathcal{V} = \begin{bmatrix} ^{B}\omega \\ ^{B}v_B \end{bmatrix} \tag{3.16}$$

They are connected by Eq:3.17

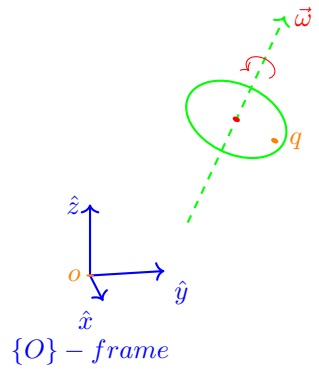


Figure 3.6: Spatial Velocity in reference frame

$$^{A}\mathcal{V} = {}^{A}X_{B} \cdot {}^{B}\mathcal{V} \tag{3.17}$$

Where $^{A}X_{B} \in \mathbb{R}^{6\times6}$ *is defined as Eq. 3.18.*

$$^{A}X_{B} = [Ad_{T}] \triangleq \begin{bmatrix} {}^{A}R_{B} & 0 \\ [{}^{A}o_{B}] \cdot {}^{A}R_{B} & {}^{A}R_{B} \end{bmatrix} \tag{3.18}$$

Where R *is the rotation matrix,* $[{}^{A}o_{B}]$ *is the skew-symmetric matrix representation of the origin of the* $\{B\}$*-frame expressed in frame* $\{A\}$ *and* $[Ad_{T}]$ *is the* ***Adjoint*** *operator for the configuration transformation* $T = (R, p)$ *Eq. 3.9.*

Proof:
$^{A}\omega$ *and* $^{B}\omega$ *are connected through Eq. 3.19 since* ω *is a* ***free vector****.*

$$^{A}\omega = {}^{A}R_{B} \cdot {}^{B}\omega \tag{3.19}$$

For $^{A}v_{A}$ *(* $^{A}v_{o_A}$*, the velocity of the body-fixed point that currently coincides with the origin of* $\{A\}$*-frame,* o_A*) and* $^{B}v_{B}$ *(* $^{B}v_{o_B}$*, Velocity of the body-fixed point that currently coincides with the origin of* $\{B\}$*-frame,* o_B*) are connected through Eq. 3.20.*

Coordinate free:

$$v_{A} = v_{B} + \omega \times (\vec{o_{B}o_{A}})$$

By using $\{A\}$*-frame we get:*

$$\begin{aligned} {}^{A}v_{A} &= {}^{A}v_{B} + {}^{A}\omega \times {}^{A}(\vec{o_{B}o_{A}}), \quad {}^{A}(-\vec{o_{A}o_{B}}) = -{}^{A}o_{B} \\ &= {}^{A}R_{B} \cdot {}^{B}v_{o_B} + ({}^{A}R_{B} \cdot {}^{B}\omega) \times (-{}^{A}o_{B}) \\ &= {}^{A}R_{B} \cdot {}^{B}v_{o_B} + ({}^{A}o_{B}) \times ({}^{A}R_{B} \cdot {}^{B}\omega) \\ &= {}^{A}R_{B} \cdot {}^{B}v_{o_B} + [{}^{A}o_{B}] \cdot ({}^{A}R_{B} \cdot {}^{B}\omega) \end{aligned} \tag{3.20}$$

By putting together Eq. 3.19 and Eq. 3.20 one ends up with Eq. 3.18.

Remark 3.1.14 *The change of coordinates frame for a twist depends only on the configuration of $\{B\}$-frame relative to $\{A\}$-frame.*

3.1.3 Screw motion

Definition 3.1.15 (Screw motion) *Is the process of rotation and translating along an axis of a rigid body, Fig. 3.7, and is represented by the* ***screw axis*** *$\{q, \hat{s}, h\}$ and the* ***rotation speed*** *$\dot{\theta}$ where:*

- *$\hat{s}$: unit vector in the direction of the rotation*
- *q: any point on the rotation axis*
- *h:* ***screw pitch*** *which defines the ratio of the linear velocity along the screw axis to the angular velocity about the screw axis*
- *$\dot{\theta}$: rotation speed*

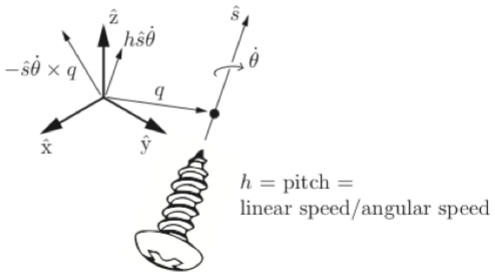


Figure 3.7: Screw motion [31]

Theorem 3.1.1 (Chasles' theorem) *Every rigid body motion can be realized by a* ***screw motion****.*

Definition 3.1.16 (From Screw Motion to Twist) *Consider a rigid body under a* ***screw motion*** *with screw axis $\{\hat{s}, h, q\}$ and (rotation) speed $\dot{\theta}$, Fig. 3.8. By fixing a reference frame $\{A\}$ with origin o_A we find the corresponding twist ${}^A\mathcal{V} = ({}^A\omega, {}^Av_{o_A})$ as follows:*

The ${}^A\omega$ can be directly computed using Eq. 3.21.

$$ {}^A\omega = {}^A\hat{s} \cdot \dot{\theta} \tag{3.21} $$

For ${}^Av_{o_A}$, by picking q as a reference point, we get the coordinate-free relation Eq. 3.22. The corresponding result in $\{A\}$-frame is Eq. 3.23.

$$ \begin{aligned} v_{o_A} &= v_q + \omega \times (\vec{qo_A}) \\ &= (h \cdot \dot{\theta}) \cdot \hat{s} + \omega \times (\vec{qo_A}) \end{aligned} \tag{3.22} $$

$$ \begin{aligned} {}^Av_{o_A} &= (h \cdot \dot{\theta}) \cdot {}^A\hat{s} + {}^A\omega \times (-\vec{{}^Aq}) \\ &= {}^A\hat{s} \cdot (h \cdot \dot{\theta}) + {}^Aq \times {}^A\omega \\ &= {}^A\hat{s} \cdot (h \cdot \dot{\theta}) + {}^Aq \times ({}^A\hat{s} \cdot \dot{\theta}) \end{aligned} \tag{3.23} $$

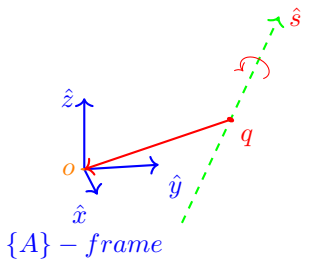


Figure 3.8: From Screw Motion to Twist

Remark 3.1.15 *The result holds as long as all the vectors and the twist are represented in the same reference frame.*

Definition 3.1.17 (From Twist to Screw Motion) *Given any twist $\mathcal{V} = (\omega, v)$ we compute the corresponding screw motion by finding $\{q, \hat{s}, h\}$ and θ as fallows:*

- *if $\omega = 0$, then it is a pure translation ($h = \infty$):*

$$\hat{s} = \frac{v}{||v||}, \;\; \dot{\theta}, \;\; h = \infty, \;\; q \text{ can be arbitrary selected}$$

$$\text{Then we get: } \mathcal{V} = \mathcal{S} \cdot \dot{\theta} = \begin{bmatrix} 0 \\ \frac{v}{||v||} \end{bmatrix} \cdot ||v||$$

- *if $\omega \neq 0$:*

$$\hat{s} = \frac{\omega}{||\omega||}, \;\; \dot{\theta} = ||\omega||, \;\; q = \frac{\omega \times v}{||\omega||^2}, \;\; h = \frac{\omega^T \cdot v}{||\omega||}$$

$$\text{If } v = 0 \text{ (no translation) then } q = 0 \text{ and } h = 0 \text{ we get: } \mathcal{V} = \mathcal{S} \cdot \dot{\theta} = \begin{bmatrix} \frac{\omega}{||\omega||} \\ 0 \end{bmatrix} \cdot ||\omega||$$

$$\text{Otherwise (general case): } \mathcal{V} = \mathcal{S} \cdot \dot{\theta} = \begin{bmatrix} \frac{\omega}{||\omega||} \\ \frac{v}{||\omega||} \end{bmatrix} \cdot ||\omega||$$

Remark 3.1.16 *The main goal of the **Screw motion** is to generalize twists Eq. 3.14 to a structure that resembles the angular velocity, ω operator. An angular velocity vector ω can be viewed as $\hat{\omega} \cdot \dot{\theta}$, where $\hat{\omega}$ is the unit rotation axis and $\dot{\theta}$ is the rate of rotation about that axis. Similarly, a twist (spatial velocity) $\mathcal{V}$ can be interpreted in terms of a **screw axis** $\hat{S} = \{s, \hat{h}, q\}$ and a velocity $\dot{\theta}$ about the screw axis. With a **slight abuse of notation**, we will often write its twist as Eq. 3.24. In this notation, we think of $\hat{S}$ as the twist associated with a unit speed motion along the screw axis $\{\hat{s}, h, q\}$.*

$$\mathcal{V} = \hat{S} \cdot \dot{\theta} \tag{3.24}$$

3.1.4 Operator View of Rigid-Body Transformation

Definition 3.1.18 (Rotation Operation via Differential Equation) *Given a point initially located at p_0 at time $t = 0$, rotating the point with unit angular velocity $\hat{\omega}$, for which the rotation axis is passing through the origin, one will end up with a motion described by Eq. 3.25 as a solution of a linear ODE. After $t = \theta$, the point has been rotated by θ degree $p(t = \theta) = e^{[\hat{\omega}] \cdot \theta} \cdot p_0$ Fig. 3.9.*

$$\begin{aligned} &\dot{p}(t) = \hat{\omega} \times p(t) = [\hat{\omega}] \cdot p(t), \quad \text{with: } p(0) = p_0 \\ &p(t) = e^{[\omega] \cdot t} \cdot p_0 \end{aligned} \tag{3.25}$$

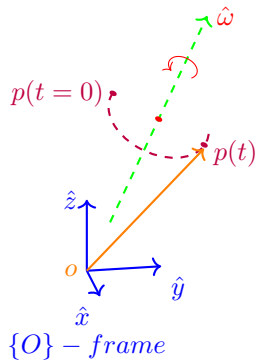


Figure 3.9: Rotation Operation via Differential Equation

Remark 3.1.17 *One can view Eq. 3.26 as a* ***rotation operator*** *that rotates a point about $\hat{\omega}$ through θ degree.*

$$Rot(\hat{\omega}, \theta) \triangleq e^{[\hat{\omega}]\cdot\theta} \tag{3.26}$$

Theorem 3.1.2 (Rotation matrix, Rotation Operator equivalence) *Every rotation matrix R can be written as $R = Rot(\hat{\omega}, \theta)$ (represents a rotation operator about ω by θ). Any matrix of the form $e^{[\hat{\omega}]\cdot\theta}$ is part of $SO(3)$.*

Proof:

Based on checking the Spatial Orthogonal Group, 3.4, properties:

$$\begin{aligned}\left(e^{[\hat{\omega}]\cdot\theta}\right)^T &= \left(I + [\hat{\omega}]\cdot\theta + \frac{[\hat{\omega}]^2\cdot\theta^2}{2!} + ...\right)^T \\ &= \sum_{j=0}^{\infty}\left(\frac{([\hat{\omega}]^T\cdot\theta)^j}{j!}\right) \\ &= e^{-[\hat{\omega}]\cdot\theta}\end{aligned} \tag{3.27}$$

$$e^A \cdot e^{-A} = I \tag{3.28}$$

From Eq. 3.27 and Eq. 3.28 we end up with the following result Eq. 3.29 (the transpose is the inverse).

$$\left(e^{[\hat{\omega}]\cdot\theta}\right)^T \cdot e^{[\hat{\omega}]\cdot\theta} = I \tag{3.29}$$

Also, since $e^{[\hat{\omega}]\cdot 0} = I \implies \det(I) = 1$ and because of the continuity of the function in terms of θ, the Eq. 3.30 is true for any ω.

$$\det\left(e^{[\hat{\omega}]\cdot\theta}\right) = 1 \tag{3.30}$$

Definition 3.1.19 ($so(3)$ set) *We define the set of so(3) matrices as 3.31*

$$so(3) \triangleq \{\mathcal{S} \in \mathbb{R}^{3\times 3} : \mathcal{S}^T = -\mathcal{S}\} \tag{3.31}$$

Remark 3.1.18 (The dual interpretation of rotation) *Given a rotation matrix, R, and the equality $q = R \cdot p$ one can interpret the result as:*

- *changing reference frame, where one has 2 frames, $\{A\}$, $\{B\}$, and one fizical point a. R represents the orientation of B relative to A while $p = {}^{B}a$ and $q = {}^{A}a$. Hence, $q = A \cdot p \iff {}^{A}a = {}^{A}R_B \cdot {}^{B}a$.*
- *applying the rotation operator, where one has 1 frame and 2 points: $a \overset{Rot()}{\rightarrow} a'$, $p = {}^{A}a$, $q = {}^{A}a'$.*

Remark 3.1.19 (Rotation Operator applied to Reference Frame) *One can extend Remark 3.1.18 to frames as well.*

- *Change of reference frame interpretation: Given a "frame object" R_A the representation of this "object frame" between 2 frames $\{B\}$ and $\{O\}$ is given by the equation: ${}^{O}R_A = {}^{O}R_B \cdot {}^{B}R_A$.*
- *Operator interpretation: Given 2 "frames objects" defined using the same reference frame one can relate them as follows: ${}^{O}R'_A = R \cdot {}^{O}R_A$ where R is the operator: $Rot(\hat{\omega}, \theta)$.*

Definition 3.1.20 (Rotation matrix properties) *Given a matrix $R \in SO(3)$, R has the following properties:*

1. $R^T \cdot R = I$
2. $R_1 \cdot R_2 \in SO(3)$ *if* $R_1, R_2 \in SO(3)$
3. $||R \cdot p - R \cdot q|| = ||p - q||$
4. $R \cdot (v \times w) = (R \cdot v) \times (R \cdot w)$
5. $R \cdot [w] \cdot R^T = [R \cdot w]$

Proof:

1. *Proven directly by using $SO(3)$* ***group*** *definition.*
2. *Directly proven by using the fact that $SO(3)$ group is closed under multiplication.*
3. *Rotation operation preserves the distance.*

$$||R \cdot p - R \cdot q|| = ||R \cdot (p - q)||^2 = (p - q)^T \cdot R^T \cdot R \cdot (p - q) \overset{by\ prop.\ 1}{=} ||p - q||$$

4. *Rotation operation preserves the orientation between 2 vectors after rotation.*

$$\begin{aligned}(R\cdot v)\times(R\cdot w) &= [R\cdot v]\cdot(R\cdot w)\\ &\overset{\text{by prop. 5}}{=} R\cdot[v]\cdot R^T\cdot(R\cdot w)\\ &= R\cdot[v]\cdot w\\ &= R\cdot(v\times w)\end{aligned}$$

5. *For $\forall u\in\mathbb{R}^n$ we have:*

$$\begin{aligned}[R\cdot w]\cdot u &= (R\cdot w)\times u\\ &= (R\cdot w)\times(R\times R^T\times u)\\ &= R\cdot(w\times R^T\cdot u)\\ &= R\cdot[w]\cdot R^T\cdot u\end{aligned}$$

Remark 3.1.20 *Given $\{A\}$-frame and $\{B\}$-frame, the numerical values of the operator $Rot(\hat{\omega},\theta)$ depend on both, the reference frame to represent $\hat{\omega}$ as well as the reference frame to represent the operator itself.*

Definition 3.1.21 (Rotation operator in Different Frames) *Consider a rotation axis $\hat{\omega}$, with $\{A\}$-frame coordinates, ${}^A\hat{\omega}$ and $\{B\}$-frame coordinates, ${}^B\hat{\omega}$ connected by the Eq. 3.32. Also, let ${}^BRot({}^B\hat{\omega},\theta)$ and ${}^ARot({}^A\hat{\omega},\theta)$ be the two rotation matrices representing the same rotation operator $Rot(\hat{\omega})$ in $\{A\}$-frame and respectively $\{B\}$-frame.*

$$ {}^A\hat{\omega} = {}^AR_B\cdot{}^B\hat{\omega} \tag{3.32}$$

Then we have the relation Eq. 3.33.

$$ {}^ARot({}^A\hat{\omega},\theta) = {}^AR_B\cdot{}^BRot({}^B\hat{\omega},\theta)\cdot{}^BR_A \tag{3.33}$$

Proof Approach 1:

We consider 2 points $p \overset{Rot(\hat{\omega},\theta)}{\rightarrow} p'$ (operator view, one reference frame, 2 points) then, by using $\{A\}$-frame we have:

$$\begin{aligned}{}^Ap' &= {}^ARot({}^A\hat{\omega},\theta)\cdot{}^Ap \quad (using\ \{A\}-frame)\\ {}^Bp' &= {}^BRot({}^B\hat{\omega},\theta)\cdot{}^Bp \quad (using\ \{B\}-frame)\\ {}^AR_B\cdot{}^Bp' &= {}^AR_B\cdot{}^BRot({}^B\hat{\omega},\theta)\cdot{}^Bp\\ {}^Ap' &= {}^AR_B\cdot{}^BRot({}^B\hat{\omega},\theta)\cdot{}^BR_A\cdot{}^Ap\\ \Longrightarrow\ {}^ARot({}^A\hat{\omega},\theta) &= {}^AR_B\cdot{}^BRot({}^B\hat{\omega},\theta)\cdot{}^BR_A\end{aligned}$$

Proof Approach 2:

By using the property 5 *(3.1.20) we have:*

$$\begin{aligned} Rot({}^A\hat{\omega},\theta) &= e^{[{}^A\hat{\omega}]\cdot\theta} \\ &= e^{[{}^AR_B\cdot{}^B\hat{\omega}]\cdot\theta} \\ &= e^{{}^AR_B\cdot[{}^B\hat{\omega}]\cdot{}^AR_B^T\cdot\theta} \qquad (3.34) \\ e^{P\cdot A\cdot P^{-1}} &= P\cdot e^A\cdot P^{-1} \qquad (3.35) \\ ({}^AR_B)^T &= ({}^AR_B)^{-1} \qquad (3.36) \end{aligned}$$

From: $(3.34, 3.36, 3.35)$ *:*

$$\implies Rot({}^A\hat{\omega},\theta) = {}^AR_B\cdot e^{[{}^B\hat{\omega}]\cdot\theta}\cdot{}^BR_A$$

Remark 3.1.21 (Rigid-body Operation via Differential Equation) *By extending the* ***Rotation Operator via Differential Equation*** *to rigid body transformations one can obtain a similar ODE characterization for* $T \in SE(3)$ *which leads to exponential coordinates.*

Definition 3.1.22 (Rigid-body Operation via Differential Equation) *Consider a point* p *that undergoes a* ***screw motion*** *with* ***screw axis*** $\mathcal{S}$ *and unit speed* $(\dot{\theta} = 1)$*. Let the corresponding twist be* $\mathcal{V} = \mathcal{S} = (\omega, v)$ *Fig. 3.10. The motion can be described by the following ODE:*

$$\dot{p(t)} = \omega \times p(t) + v \in \mathbb{R}^3 \implies \begin{bmatrix}\dot{p(t)}\\ 0\end{bmatrix} = \begin{bmatrix}[\omega] & v\\ 0 & 0\end{bmatrix}\cdot\begin{bmatrix}p(t)\\ 1\end{bmatrix} \in \mathbb{R}^4 \qquad (3.37)$$

$$\text{Where we define: } \tilde{p(t)} = \begin{bmatrix}p(t)\\ 1\end{bmatrix} \in \mathbb{R}^3 \qquad (3.38)$$

And it has the following solution in homogeneous coordinates Eq. 3.39

$$\begin{bmatrix}p(t)\\ 1\end{bmatrix} = e^{\left(\begin{bmatrix}[\omega] & v\\ 0 & 0\end{bmatrix}\cdot t\right)}\cdot\begin{bmatrix}p(0)\\ 1\end{bmatrix} \qquad (3.39)$$

Definition 3.1.23 (Matrix representation of a twist) *Given a twist* $\mathcal{V} = (\omega, v)$*, let* $[\mathcal{V}] \in \mathbb{R}^4$ *be its matrix representation Eq. 3.40*

$$[\mathcal{V}] = \begin{bmatrix}[\omega] & v\\ 0 & 0\end{bmatrix} \in \mathbb{R}^{4\times4} \qquad (3.40)$$

Remark 3.1.22 *The Def. 3.1.23 also applies to screw axis* $\mathcal{S}$ *and in this case, the solution of Eq. 3.39 becomes Eq. 3.41 where,* $[\mathcal{S}]$ *is of the form Eq. 3.40*

$$\tilde{p(t)} = e^{[\mathcal{S}]\cdot t}\cdot\tilde{p(0)} \qquad (3.41)$$

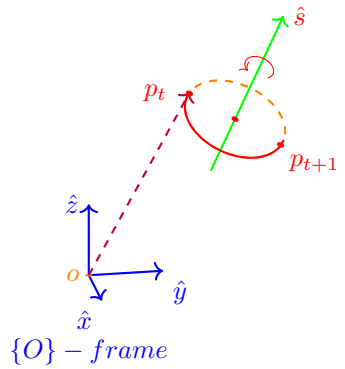


Figure 3.10: Rigid-body Operation via Differential Equation

Remark 3.1.23 *$e^{[\mathcal{S}]\cdot t} \in SE(3)$ is always a valid homogeneous transformation matrix.*

Remark 3.1.24 *Any $T \in SE(3)$ can be written as $T = e^{[\mathcal{S}]\cdot t}$, i.e., it can be viewed as an operator that moves a point/frame along the screw axis at unit speed for time t.*

Definition 3.1.24 ($se(3)$ set) *Similarly with $so(3)$ 3.1.19, we define $se(3)$ set as Eq. 3.42 where the matrix representation of $se(3)$ is Eq. 3.40.*

$$se(3) = \{([w], v) : [w] \in so(3), v \in \mathbb{R}^3\} \tag{3.42}$$

Remark 3.1.25 (Homogeneous Transformation as Rigid-body Operator) *The operator viewed of a rotation matrix, remark 3.1.18, can be extended to homogeneous transformations as well, by using the screw axes interpretation of twists. W.l.o.g., by considering the "unit velocity" $\mathcal{V} = \mathcal{S}$, the time t from Eq. 3.41 has a similar meaning, with the degrees, as it was for the rotation matrix.*

For any homogeneous transformation, T, one can find a screw axis, $\mathcal{S}$ s.t. $T = e^{[\mathcal{S}]\cdot\theta}$. This way, one can interpret the transformation T of $\tilde{p}' = T \cdot \tilde{p}$ as the "rotation" of p about the screw axis $\mathcal{S}$ by θ degrees. The same argument can be made for frames as well.

Definition 3.1.25 (Rigid-Body Operator in Different Frames) *Generalization of Definition 3.1.21 For a rigid-body operator T represented in another frame (e.g. $\{O\}$) than the object (e.g. $\{B\}$), one will have to transform the object to the same frame with the operator, apply the operator and then transform back in the initial frame Eq. 3.43.*

$$^{O}T \iff T_B^{-1} \cdot {}^{O}T \cdot {}^{O}T_B \tag{3.43}$$

Definition 3.1.26 (Rigid Operation applied on Screw Axis) *Considering an arbitrary screw axis $\mathcal{S}$ and suppose the axis has gone through a rigid transformation $T = (R, p)$ and the resulting new screw axis is $\mathcal{S}'$, then we have the following relation Eq. 3.44.*

$$\mathcal{S}' = [Ad_T] \cdot \mathcal{S} \tag{3.44}$$

Proof:

One can consider an arbitrarily frame $\{A\}$, rigidly attached to the screw axes, and the $\{B\}$-frame, obtained after applying the operator T.

The coordinates of $\mathcal{S}$ in $\{A\}$ are the same as the coordinates of $\mathcal{S}'$ in $\{B\}$ (i.e. ${}^A\mathcal{S} = {}^B\mathcal{S}'$).

Also, we know that for Eq. 3.45, $T = {}^AT_B$ when $\{A\}$-frame is used.

$$T_B = T \cdot T_A \tag{3.45}$$

By multiplying ${}^AX_B$ (applying the change of coordinates of twist) to Eq. 3.44 we get:

$$\begin{aligned} {}^AX_B \cdot {}^A\mathcal{S} &= {}^AX_B \cdot {}^B\mathcal{S}' = {}^A\mathcal{S}' \\ \implies {}^A\mathcal{S}' &= {}^AX_B \cdot {}^A\mathcal{S} \end{aligned} \tag{3.46}$$

Remark 3.1.26 *${}^AX_B$, from Eq. 3.46 can be interpreted in 2 equivalent ways:*

- *the transformation between frame $\{B\}$ and $\{A\}$*
- *the operator inside its corresponding transformation T*

These proprieties are inherited directly from the dual interpretation of homogeneous transformations to which it corresponds.

3.1.5 Exponential coordinate of Rigid Body Configuration

Definition 3.1.27 (Exponential Coordinate)
For any unit vector $[\hat{\omega}] \in so(3)$ and any $\theta \in \mathbb{R}$, we have $e^{[\hat{\omega}]\cdot\theta} \in SO(3)$.
For any $R \in SO(3)$, exists $\hat{\omega} \in \mathbb{R}^3$ with $||\hat{\omega}|| = 1$ and $\theta \in \mathbb{R}$ s.t. $R = e^{[\hat{\omega}]\cdot\theta}$.
*The **vector** $\hat{\omega} \cdot \theta$ is called the **exponential coordinate** for R.*
*The **exponential coordinates** are also called the **canonical coordinates** of the rotation group $SO(3)$.*

Definition 3.1.28 (Exponential Map)

$$e^{[\omega]\cdot\theta} = I + \theta \cdot [\omega] + \frac{\theta^2}{2!} \cdot [\omega]^2 + \frac{\theta^3}{3!} \cdot [\omega]^3 + ... \tag{3.47}$$

Definition 3.1.29 (Rodrigues' Formula) *Given any unit vector $[\hat{\omega}] \in so(3)$, we have Eq. 3.48*

$$e^{[\hat{\omega}]\cdot\theta} = I + [\hat{\omega}] \cdot sin(\theta) + [\hat{\omega}]^2 \cdot (1 - cost(\theta)) \tag{3.48}$$

Proof:

W.l.o.g. we consider Eq. 3.49. In case this is not true, we normalize it and rescale θ with its norm.

$$||\hat{\omega}|| = 1 \tag{3.49}$$

Also, the following equalities are true:

$$\begin{aligned}
[\hat{\omega}]^2 &= -[\hat{\omega}]^T \\
[\hat{\omega}]^3 &= -[\hat{\omega}] \quad (\textit{using Eq. } 3.49) \\
[\hat{\omega}]^4 &= [\hat{\omega}]^3 \cdot [\hat{\omega}] = -[\hat{\omega}]^2
\end{aligned}$$

Then we get:

$$\begin{aligned}
e^{[\hat{\omega}]\cdot\theta} &= I + \theta \cdot [\hat{\omega}] + \frac{\theta^2}{2!} \cdot [\hat{\omega}]^2 + \frac{\theta^3}{3!} \cdot [\hat{\omega}]^3 + \frac{\theta^4}{4!} \cdot [\hat{\omega}]^4 + ... \\
&= I + \underbrace{(\theta + \frac{\theta^3}{3!} + \frac{\theta^5}{5!} + ...)}_{sin(\theta)} \cdot [\hat{\omega}] + \underbrace{(\frac{\theta^2}{2} - \frac{\theta^4}{4!} + \frac{\theta^6}{6!} + ...)}_{1-cos(\theta)} \cdot [\hat{\omega}]^2
\end{aligned}$$

Definition 3.1.30 *Given $R \in SO(3)$ the the corresponding θ and ω are computed as follows Eq. 3.1.30*

- *If $R = I$, the $\theta = 0$ and $\hat{\omega}$ is undefined*
- *If Tr(R)=-1, then $\theta = \pi$ and set $\hat{\omega}$ equal to one of the following:*

$$\frac{1}{\sqrt{2 \cdot (1 + r_{33})}} \cdot \begin{bmatrix} r_{13} \\ r_{23} \\ 1 + r_{33} \end{bmatrix}, \frac{1}{\sqrt{2 \cdot (1 + r_{22})}} \cdot \begin{bmatrix} r_{12} \\ 1 + r_{22} \\ r_{32} \end{bmatrix}, \frac{1}{\sqrt{2 \cdot (1 + r_{11})}} \cdot \begin{bmatrix} 1 + r_{11} \\ r_{21} \\ r_{31} \end{bmatrix}$$

- *Otherwise, $\theta = cos^{-1}(\frac{1}{2} \cdot \ Tr\ (R) - 1) \in [0, \pi)$ and $[\hat{\omega}] = \frac{1}{2 \cdot sin(\theta)}) \cdot (R - R^T)$*

Definition 3.1.31 (Exponential Map of se(3)) *For any twist $\mathcal{V} = (\omega, v)$ and $\theta \in \mathbb{R}$, we have $e^{[\mathcal{V}]\cdot\theta} \in SE(3)$ computed as follows:*

- $(\omega = 0) : e^{[\mathcal{V}]\cdot\theta} = \begin{bmatrix} I & v \cdot \theta \\ 0 & 1 \end{bmatrix}$
- $(\omega \neq 0)$ *without loss of generality one can assume $||\omega|| = 1$ then:*

$$e^{[\mathcal{V}]} = \begin{bmatrix} e^{[\omega]\cdot\theta} & G(\theta) \cdot v \\ 0 & 1 \end{bmatrix}, \ G(\theta) = I \cdot \theta + (1 - cos(\theta)) \cdot [\omega] + (\theta - sin(\theta)) \cdot [\omega]^2$$

Definition 3.1.32 (Log of SE(2)) *Given any transformation $T = (R, p) \in SE(3)$, one can always find the twist $\mathcal{S} = (\omega, v)$ and a scalar θ s.t. Eq.3.50*

$$e^{[\mathcal{S}]\cdot\theta} = T = \begin{bmatrix} R & p \\ 0 & 1 \end{bmatrix} \tag{3.50}$$

which is computed as follows:

- *if $R = I$ then $\omega = 0,\ v = \frac{p}{||p||}$, and $\theta = ||p||$*
- *Otherwise, use matrix logarithm on $SO(3)$ to determine ω and θ from R. Then v is computed as $v = G^{-1}(\theta) \cdot p$, where Eq. 3.51*

$$G^{-1}(\theta) = \frac{1}{\theta} \cdot I - \frac{1}{2} \cdot [\omega] + (\frac{1}{\theta} - \frac{1}{2} \cdot cos(\frac{\theta}{2})) \cdot [\omega]^2 \tag{3.51}$$

3.1.6 Instantaneous Velocity of Moving Frame

Definition 3.1.33 (Instantaneous Velocity of Rotating Frame) *Given an $\{A\}$-frame with the orientation $R_A(t)$ and the velocity $\omega_A(t)$, both expressed in $\{O\}$-frame Fig. 3.11. The rate of change of R_A w.r.t. time is Eq. 3.52.*

$$\frac{d}{dt}R_A(t) = [\omega_A] \cdot R_A(t) \implies [\omega_A(t)] = \dot{R}_A(t) \cdot R_A^{-1}(t) \tag{3.52}$$

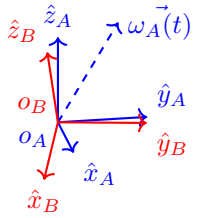


Figure 3.11: Instantaneous Velocity of Rotating Frame

Proof:

$$
\begin{aligned}
{}^{O}R_A(t) &= \begin{bmatrix} {}^{O}\hat{x}_A(t) & {}^{O}\hat{y}_A(t) & {}^{O}\hat{z}_A(t) \end{bmatrix} \\
\implies {}^{O}\dot{R}_A(t) &= \begin{bmatrix} {}^{O}\dot{\hat{x}}_A(t) & {}^{O}\dot{\hat{y}}_A(t) & {}^{O}\dot{\hat{z}}_A(t) \end{bmatrix}
\end{aligned}
$$

Since we know that:

$$
\begin{aligned}
\dot{\hat{x}}_A &= \omega_A \times \hat{x}_A = [\omega_A] \cdot \hat{x}_A \\
\dot{\hat{y}}_A &= \omega_A \times \hat{y}_A = [\omega_A] \cdot \hat{y}_A \\
\dot{\hat{z}}_A &= \omega_A \times \hat{z}_A = [\omega_A] \cdot \hat{z}_A
\end{aligned}
$$

$$
\implies \dot{R}_A = [\omega_A] \cdot R_A \implies [\omega_A] = \dot{R}_A \cdot R_A^{-1} \tag{3.53}
$$

$$
\textit{More precisely: } [{}^{O}\omega_A] = {}^{O}\dot{R}_A \cdot {}^{O}R_A^{-1} \tag{3.54}
$$

And since: ${}^{A}\omega_A = {}^{A}R_O \cdot {}^{O}\omega_A$

$$
\begin{aligned}
[{}^{A}\omega_A] = [{}^{A}R_O \cdot {}^{O}\omega_A] &\overset{3.1.20}{=} {}^{A}R_O \cdot [{}^{O}\omega_A] \cdot {}^{A}R_O^T \\
&= {}^{A}R_O \cdot ({}^{O}\dot{R}_A \cdot {}^{O}R_A^{-1}) \cdot {}^{O}R_A \\
&= {}^{A}R_O \cdot {}^{O}\dot{R}_A = {}^{O}R_A^{-1} \cdot {}^{O}\dot{R}_A \\
\implies [{}^{A}\omega_A] &= {}^{O}R_A^{-1} \cdot {}^{O}\dot{R}_A
\end{aligned} \tag{3.55}
$$

Clarification:

- ${}^{A}\omega_A$ *Is the velocity of* $\{A\}$*-frame w.r.t.* $\{A\}$*-frame.*
- ${}^{O}R_A$ *Is the orientation of frame* $A : R_A$*, expressed in* $\{O\}$*-frame.*

Definition 3.1.34 (Instantaneous Velocity of Moving Frames) *Given a moving frame* $\{A\}$ *with configuration* $T_A(t)$ *and velocity* $\mathcal{V} = (\omega, v)$*, both expressed in* $\{O\}$*-frame Fig. 3.12 then, the rate of change of* T_A *w.r.t. time is Eq. 3.56.*

$$
\frac{d}{dt}T_A(t) = [\mathcal{V}_A(t)] \cdot T_A(t) \implies [\mathcal{V}_A(t)] = \dot{T}_A(t) \cdot T_A^{-1}(t) \tag{3.56}
$$

Proof:
By using the direction vectors of the $\{A\}$*-frame* $\hat{x}, \hat{y}, \hat{z}$ *as well as the origin point* O_A *,all in homogeneous form, we build* $T_A = \begin{bmatrix} \tilde{x}_A & \tilde{y}_A & \tilde{z}_A & \tilde{O}_A \end{bmatrix}$ *Fig. 3.13.*

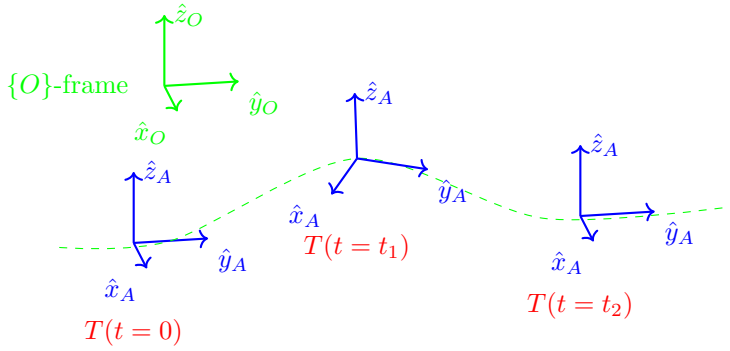


Figure 3.12: Instantaneous Velocity of Moving Frame

$$\hat{x}_A = O_A\vec{P}_x = P_x - O_A$$
$$\implies \dot{x}_A = \dot{P}_x - \dot{O}_A = (v_o + \omega \times O\vec{P}_x) - (\vec{v}_o + \omega \times O\vec{O}_A)$$
$$= \omega \times (O\vec{P}_x - O\vec{O}_A) = \omega \times \hat{x}_A \tag{3.57}$$

And since: $\dot{\hat{x}}_A = \begin{bmatrix}\dot{\hat{x}}_A \\ 0\end{bmatrix} \overset{3.57}{\implies} \dot{\hat{x}}_A = \begin{bmatrix}[\omega] & v_o \\ 0 & 0\end{bmatrix} \cdot \begin{bmatrix}\hat{x}_A \\ 0\end{bmatrix}$ (3.58)

Analogous, we get: $\dot{\hat{y}}_A = \begin{bmatrix}[\omega] & v_o \\ 0 & 0\end{bmatrix} \cdot \begin{bmatrix}\hat{y}_A \\ 0\end{bmatrix}$ *and* $\dot{\hat{z}}_A = \begin{bmatrix}[\omega] & v_o \\ 0 & 0\end{bmatrix} \cdot \begin{bmatrix}\hat{z}_A \\ 0\end{bmatrix}$ (3.59)

Also: $\dot{\hat{O}}_A = \begin{bmatrix}\dot{O}_A \\ 0\end{bmatrix} = \begin{bmatrix}v_o + \omega \times O_A \\ 0\end{bmatrix} = \begin{bmatrix}[\omega] & v_o \\ 0 & 0\end{bmatrix} \cdot \begin{bmatrix}O_A \\ 1\end{bmatrix}$ (3.60)

$$\overset{3.58}{\implies} \overset{3.59}{\implies} \overset{3.60}{\implies} \dot{T}_A = [\mathcal{V}_A] \cdot T_A \implies [\mathcal{V}_A] = \dot{T}_A \cdot T_A^{-1} \tag{3.61}$$
$$\implies [{}^A\mathcal{V}_A] = {}^OT_A^{-1} \cdot {}^O\dot{T}_A \tag{3.62}$$

The proof of Eq. 3.62 is similar to the one from Rotation Frames, Eq. 3.55.

Clarification:

- ${}^A\mathcal{V}_A$ *represents the velocity of frame* $\{A\}$ *relative to frame* $\{O\}$ *expressed in frame* $\{A\}$.

3.1.7 Kinematics of Open Chain, Product of Exponential

Definition 3.1.35 (Forward Kinematics) *Is the process of calculation of the configuration* $T = (R, p)$ *of the end-effector frame from joint variables:* $\theta = (\theta_1, ..., \theta_n)$ *Fig. 3.14.*

Definition 3.1.36 *Suppose we have a robot with n joints and n links. Each joint has* ***one degree of freedom*** *represented by joint variable* θ_i *for* $i = 1, ...n$ *where:*

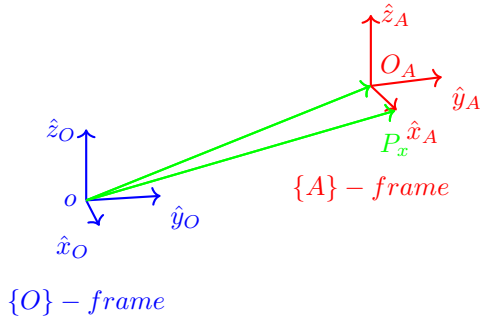


Figure 3.13: Moving Frame

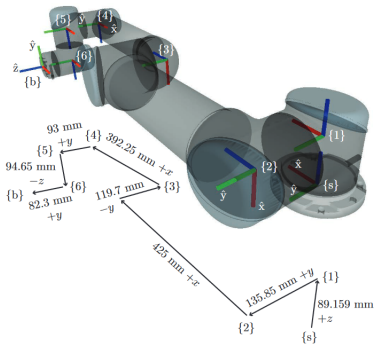


Figure 3.14: Forward Kinematics [31]

- θ_i*: joint angle (revolute joint) or joint displacement (prismatic joint)*
- $\{s\}$*-frame: fixed frame (also referred as frame* $\{0\}$*)*
- $\{i\}$*-frame: is attached to link* i*, for* $i = 1, ...n$
- $\{b\}$*-frame: frame attached at the* ***end-effector*** *(also referred* $\{n+1\}$*-frame)*
- ${}^i\mathcal{S}_i$*: screw axis of joint* i *expressed in* $\{i\}$*-frame (also referred as* $\mathcal{S}$ *expressed in local frame)*
- ${}^0\mathcal{S}_i$*: screw axis of joint* i *expressed in* ***fixed frame*** $\{0\}$ *(global frame /* $\{s\}$*)*
- $\theta_1 = 0, ..., \theta_n = 0$*:* ***Home position*** *is the configuration where all the joint angles are zero. Other fixed angles can be chosen as the* ***home position****.*
- ${}^0\bar{\mathcal{S}}_i = {}^0\mathcal{S}_i(0, ..., 0)$*: the screw axis of joint* i *expressed in frame* $\{0\}$*, when the robot is at the* ***home position****.*

*The **computation** of ${}^bT_s(\theta_1, ..., \theta_n)$ Fig. 3.15 is done as following:*

Step 1:** Configuration of **end-effector** when the robot is at **home position

$$M \triangleq {}^bT_s(0, ..., 0)$$

***Step 2:** Apply all screw motions:*

$$ {}^bT_s(\theta_1, ..., \theta_n) = e^{[{}^0\bar{\mathcal{S}}_1]\cdot\theta_1} \cdot e^{[{}^0\bar{\mathcal{S}}_2]\cdot\theta_2} \cdot ...e^{[{}^0\bar{\mathcal{S}}_n]\cdot\theta_n} \cdot M \tag{3.63}$$

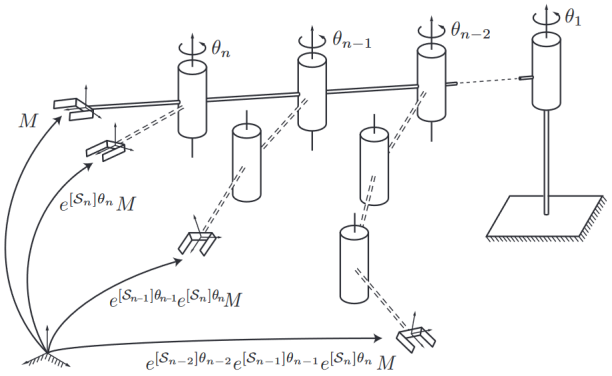


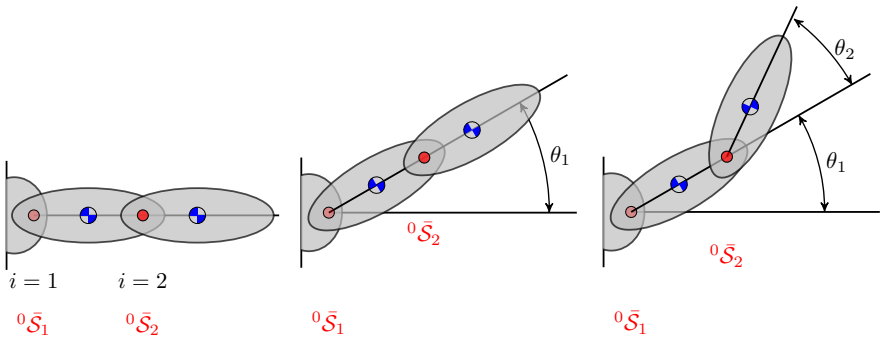
Figure 3.15: Product of exponentials [31]

Remark 3.1.27 *${}^i\mathcal{S}_i$ does not change when the robot moves (i.e. when θ changes), but ${}^0\mathcal{S}_i$ depends on $\theta_1, ..., \theta_i$.*

Remark 3.1.28 (Product of exponentials is order independent) *The product of exponentials was obtained in Eq. 3.63 by applying the screw motions along screw axes in this order: ${}^0\bar{\mathcal{S}}_n, {}^0\bar{\mathcal{S}}_{n-1}, ..., {}^0\bar{\mathcal{S}}_1$. The same result can be obtained in any different order as well.*

Proof:

For simplicity, we assume that $n = 2$ Fig. 3.16 . By applying the screw motion, Eq. 3.64, along ${}^0\bar{\mathcal{S}}_1$ first we get Eq. 3.65:



(a) Rest position (b) Motion along axes 1 (c) Motion along axes 2

Figure 3.16: Product of exponentials, $n = 2$

$$T = e^{[{}^0\bar{\mathcal{S}}_1]\cdot\theta_1} \tag{3.64}$$

$${}^sT_b(\theta, 0) = e^{[{}^0\bar{\mathcal{S}}_1]\cdot\theta_1} \cdot M \tag{3.65}$$

The screw axes for joint 2 *has been changed. The new axes is*

$${}^0\bar{\mathcal{S}}_2 \overset{T=e^{[{}^0\bar{\mathcal{S}}_1]\cdot\theta_1}}{\longrightarrow} {}^0\mathcal{S}_2 = [Ad_T] \cdot {}^0\bar{\mathcal{S}}_2 \tag{3.66}$$

$${}^0\mathcal{S}_2 = {}^0\mathcal{S}_2(\theta, 0) \neq {}^0\bar{\mathcal{S}}_2 \tag{3.67}$$

$$\implies {}^sT_b(\theta_1, \theta_2) = e^{[{}^0\mathcal{S}_2 \neq {}^0\bar{\mathcal{S}}_2]\cdot\theta_2} \cdot {}^sT_b(\theta_1, 0) \tag{3.68}$$

From property 5 in 3.1.20 *we can also show that:* (3.69)

$$\mathcal{S}' = [Ad_T] \cdot \mathcal{S} \iff [\mathcal{S}'] = T \cdot [\mathcal{S}] \cdot T^{-1} \tag{3.70}$$

$$\overset{3.70}{\implies} e^{[{}^0\mathcal{S}_2]\cdot\theta_2} = e^{[[Ad_T]\cdot{}^0\bar{\mathcal{S}}_2]\cdot\theta_2} = e^{T\cdot[{}^0\bar{\mathcal{S}}_2]\cdot T^{-1}\cdot\theta_2}$$

$$= T \cdot e^{{}^0\bar{\mathcal{S}}_2} \cdot T^{-1} \tag{3.71}$$

$$\implies {}^sT_b(\theta_1, \theta_2) = (T \cdot e^{[{}^0\bar{\mathcal{S}}_2]\cdot\theta_2} \cdot T^{-1}) \cdot (e^{[{}^0\bar{\mathcal{S}}_1]\cdot\theta_1}) \cdot M$$

$$= e^{[{}^0\bar{\mathcal{S}}_2]} \cdot e^{[{}^0\bar{\mathcal{S}}_1]\cdot\theta_1} \cdot M \tag{3.72}$$

3.1.8 Velocity Kinematics

Definition 3.1.37 (Velocity Kinematics) *Is the process of* ***end-effector*** *frame* $\{b\}$ *velocity computation relative to the joint velocities* $\dot{\theta}_1, ...\dot{\theta}_n$ *and it depends on how, the velocity of* $\{b\}$*-frame is represented:*

- *Twist representation* $\implies$ ***Geometric Jacobian***
- *Local representation of* $SE(3)$ $\implies$ ***Analytic Jacobian***

Definition 3.1.38 (Geometric Jacobian) *Given the* ***end-effector*** *twist* $\mathcal{V} = (\omega, v)$*, the connection with joint velocities* $\dot{\theta}_1, ...\dot{\theta}_n$ *is given by* $J(\theta)$*: Eq. 3.73.*

$$\mathcal{V} = J(\theta) \cdot \dot{\theta} = J_1(\theta) \cdot \dot{\theta}_1 + ... + J_n(\theta) \cdot \dot{\theta}_n \tag{3.73}$$

where the i*th column* $J_i(\theta)$ *is the* ***end-effector velocity*** *when the robot is rotating about* $\mathcal{S}_i$ *at unit speed* $\dot{\theta}_i = 1$ *while all the other joints do not move (i.e.* $\dot{\theta}_j = 0$ *for* $j \neq i$*).*

Remark 3.1.29 *The column vectors* J_i *of the* ***Geometric Jacobian*** *represents the* ***joint*** i*th* ***screw axis*** *Eq. 3.74 and it depends on* θ *as well as the reference frames.*

$$J_i(\theta) = \mathcal{S}_i(\theta) \tag{3.74}$$

Remark 3.1.30 *Each column of the* ***Geometric Jacobian*** *can be represented as:*

- ${}^iJ_i = {}^i\mathcal{S}_i, \quad i = 1, ...n$ *local coordinate, which makes the corresponding screw axes independent of* θ *with the disadvantage of having a Jacobian column matrix in different coordinate frames.*
- ${}^0J_i(\theta) = {}^0X_i(\theta) \cdot {}^i\mathcal{S}_i, \quad i = 1, ..n$ *in a selected,* ***fixed frame*** $\{0\}$

and can be computed, in a systematic way using the following algorithm: Alg. 1.

Algorithm 1 Geometric Jacobian Algorithm

Require: ${}^0\bar{\mathcal{S}}_1, ..., {}^0\bar{\mathcal{S}}_n, \theta_1, ..., \theta_n$
Ensure: ${}^0J(\theta)$
1: ${}^0\mathcal{S}_1(\theta) = {}^0\bar{\mathcal{S}}_1$ // $= {}^0\mathcal{S}_1(0)$ (independent of θ)
2: **for** i=2,n **do**
3: $\quad \hat{T}(\theta_1, ..., \theta_{i-1}) = e^{[{}^0\bar{\mathcal{S}}_1]\cdot\theta_1} \cdot ... \cdot e^{[{}^0\bar{\mathcal{S}}_{i-1}]\cdot\theta_{i-1}}$
4: $\quad {}^0J_i(\theta) = [Ad_{\hat{T}(\theta_1,...,\theta_{i-1})}] \cdot {}^0\bar{\mathcal{S}}_i$ // $= {}^0\mathcal{S}_i$
5: **end for**

3.1.9 Spatial Acceleration

Definition 3.1.39 (Spatial Acceleration) *Given Fig. 3.17, a rigid body with spatial velocity* $\mathcal{V} = (\omega, v_o)$*, its spatial acceleration is* $\mathcal{A} = \lim_{\delta \to 0} \frac{\mathcal{V}(t+\delta t) - \mathcal{V}(\delta t)}{\delta} = \begin{bmatrix} \dot{\omega} \\ \dot{v}_o \end{bmatrix}$ *where:*

- v_o *is the velocity of the body-fixed particle coinciding which the frame origin* o *at the current time* t
- $\dot{\omega}$ *is an angular acceleration of the body*
- $\dot{v}_o$ *is the acceleration of* ***no*** *body-fixed point but in fact,* $\dot{v}_o$ *gives the rate of change in* ***stream velocity of body-fixed particles passing through point*** o

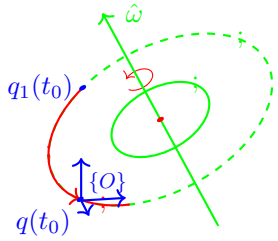


Figure 3.17: Spatial Acceleration

Remark 3.1.31 *Why* $\dot{v}_o$ ***is the acceleration of no body-fixed point****? Suppose* $q(t_0)$ *is the body fixed particle that coincides with point* o *at time* t_0 *then* $v_o(t_0) = \dot{q}(t_0)$*. However,* $\dot{v}_o(t_0) \neq \ddot{q}(t_0)$*, where* $\ddot{q}(t)$ *is the conventional acceleration of the body-fixed point* q*.*

$$\dot{v}_o(t_0) \triangleq \lim_{\delta \to 0} \frac{v_o(t_0 + \delta t) - v_o(\delta t_0)}{\delta} \neq \lim_{\delta \to 0} \frac{\dot{q}(t_0 + \delta) - \dot{q}(t_0)}{\delta} = \ddot{q}(t_0) \tag{3.75}$$

$$since:$$

$$at\ time\ t = t_0,\ \ q(t_0) = 0,\ \ v_o(t_0) = \dot{q}(t_0) \tag{3.76}$$

$$and\ at\ time\ t = t_0 + \delta,\ \ q_1(t_0 + \delta) = 0,\ \ v_o(t_0 + \delta) = \dot{q}_1(t_0 + \delta) \neq \dot{q}(t_0 + \delta) \tag{3.77}$$

Definition 3.1.40 (Computation of $\ddot{q}(t)$**)** *For all* t *holds:*

$$\dot{q}_(t) = v_o(t) + \omega(t) \times q(t) \tag{3.78}$$

$$then:$$

$$\ddot{q}(t) = \dot{v}_o(t) + \dot{\omega}(t) \times q(t) + \omega(t) \times \dot{q}(t) \tag{3.79}$$

Remark 3.1.32 *If $q(t)$ is the body fixed particle that coincides with o at time t ($q(t) = \vec{0}$), then we have Eq. 3.80.*

$$\ddot{q}(t) = \dot{v}_o(t) + \omega(t) \times \dot{q}(t) \tag{3.80}$$

3.1.10 Plücker Coordinate System and Basis Vector

Definition 3.1.41 (Derivatives) *Let $r \in \mathbb{R}^3$ be a free vector with the coordinates defined in $\{O\}$-frame respectively $\{B\}$-frame (both with the same fixed origin):*

$$^{O}r = \begin{bmatrix} ^{O}r_x \\ ^{O}r_y \\ ^{O}r_z \end{bmatrix} \in \mathbb{R}^3 \iff r = \begin{bmatrix} \hat{x}_O & \hat{y}_O & \hat{z}_O \end{bmatrix} \cdot {}^{O}r \tag{3.81}$$

$$^{B}r = \begin{bmatrix} ^{B}r_x \\ ^{B}r_y \\ ^{B}r_z \end{bmatrix} \in \mathbb{R}^3 \iff r = \begin{bmatrix} \hat{x}_B & \hat{y}_B & \hat{z}_B \end{bmatrix} \cdot {}^{B}r \tag{3.82}$$

$$\overset{\text{Expressing Eq. 3.82 in } \{O\}\text{-frame}}{\Longrightarrow} \quad {}^{O}r = \begin{bmatrix} ^{O}\hat{x}_B & ^{O}\hat{y}_B & ^{O}\hat{z}_B \end{bmatrix} \cdot {}^{B}r \tag{3.83}$$

where $\{O\}$-frame is an inertial frame Fig. 3.18 ***and where the derivatives are defined implicitly w.r.t. the inertial $\{O\}$-frame.***

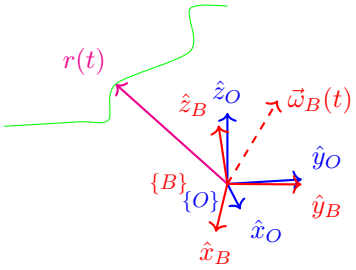


Figure 3.18: Velocity in different frames

$$\overset{\text{From Eq. 3.81}}{\Longrightarrow} \dot{r} = \begin{bmatrix}\hat{x}_O & \hat{y}_O & \hat{z}_O\end{bmatrix} \cdot \underbrace{\frac{d}{dt}({}^O r)}_{\triangleq {}^O\dot{r} \text{ (apparent derivative)}} \tag{3.84}$$

$$\overset{\text{using } \{O\}-frame}{\Longrightarrow} {}^O(\dot{r}) = \underbrace{\begin{bmatrix}{}^O\hat{x}_O & {}^O\hat{y}_O & {}^O\hat{z}_O\end{bmatrix}}_{I_{3\times 3}} \cdot \frac{d}{dt}({}^O r) = \frac{d}{dt}({}^O r) \tag{3.85}$$

$$\overset{\text{From Eq.3.82}}{\Longrightarrow} \dot{r} = \begin{bmatrix}\dot{\hat{x}}_B & \dot{\hat{y}}_B & \dot{\hat{z}}_B\end{bmatrix} \cdot {}^B r + \begin{bmatrix}\hat{x}_B & \hat{y}_B & \hat{z}_B\end{bmatrix} \cdot \frac{d}{dt}({}^B r) \tag{3.86}$$

$$= \omega_B \times \begin{bmatrix}\hat{x}_B & \hat{y}_B & \hat{z}_B\end{bmatrix} \cdot {}^B r + \begin{bmatrix}\hat{x}_B & \hat{y}_B & \hat{z}_B\end{bmatrix} \cdot \frac{d}{dt}({}^B r) \tag{3.87}$$

$$\overset{\text{using } \{B\}\text{-frame}}{\Longrightarrow} {}^B(\dot{r}) = \underbrace{{}^B\omega_B \times {}^B r}_{\substack{\text{account for coordinates} \\ \text{frame axes is moving}}} + \underbrace{\frac{d}{dt}({}^B r)}_{\text{apparent derivative}} \tag{3.88}$$

Definition 3.1.42 (Plücker Coordinates) *Given $\{B\}$-frame we define $\{e_{B_1}, e_{B_2}, ...e_{B_6}\}$ a 6-dimensional **motion basis vectors** that can be used to define coordinate free twists as follows Eq. 3.89.*

$$\mathcal{V}_{body} = \alpha_1 \cdot e_{B_1} + ... + \alpha_6 \cdot e_{B_6} \tag{3.89}$$

where:

$$\begin{bmatrix}\alpha_1 & \alpha_2 & \alpha_3 & \alpha_4 & \alpha_5 & \alpha_6\end{bmatrix}^T = {}^B\mathcal{V}_{body} \tag{3.90}$$

$$\begin{bmatrix}1 & 0 & 0 & 0 & 0 & 0\end{bmatrix}^T \text{ means rotation about } \hat{x}_B \text{ at unit speed}, \mathcal{V} = e_{B_1} \tag{3.91}$$

$$\begin{bmatrix}0 & 1 & 0 & 0 & 0 & 0\end{bmatrix}^T \text{ means rotation about } \hat{y}_B \text{ at unit speed}, \mathcal{V} = e_{B_2} \tag{3.92}$$

$$\begin{bmatrix}0 & 0 & 1 & 0 & 0 & 0\end{bmatrix}^T \text{ means rotation about } \hat{z}_B \text{ at unit speed}, \mathcal{V} = e_{B_3} \tag{3.93}$$

$$\begin{bmatrix}0 & 0 & 0 & 1 & 0 & 0\end{bmatrix}^T \text{ means motion along } \hat{x}_B \text{ at unit speed}, \mathcal{V} = e_{B_4} \tag{3.94}$$

$$\begin{bmatrix}0 & 0 & 0 & 0 & 1 & 0\end{bmatrix}^T \text{ means motion along } \hat{y}_B \text{ at unit speed}, \mathcal{V} = e_{B_5} \tag{3.95}$$

$$\begin{bmatrix}0 & 0 & 0 & 0 & 0 & 1\end{bmatrix}^T \text{ means motion along } \hat{z}_B \text{ at unit speed}, \mathcal{V} = e_{B_6} \tag{3.96}$$

*We call $\alpha_1, \alpha_2, ..., \alpha_6$ **Plücker Coordinates** that corresponds to **motion basis vector** $e_{B_1}, e_{B_2}, ...e_{B_6}$.*

Remark 3.1.33 *By using $\{O\}$-frame to represents a twist we get Eq. 3.97*

$${}^O\mathcal{V}_{body} = \alpha_1 \cdot {}^O e_{B_1} + ... + \alpha_6 \cdot {}^O e_{B_6} = \begin{bmatrix}{}^O e_{B_1} & ... & {}^O e_{B_6}\end{bmatrix} \cdot {}^O\mathcal{V}_{body} \tag{3.97}$$

where:

$$\begin{bmatrix}{}^O e_{B_1} & ... & {}^O e_{B_6}\end{bmatrix} = {}^O X_B = \begin{bmatrix}Ad_{{}^O T_B}\end{bmatrix} \tag{3.98}$$

Definition 3.1.43 (Spatial Acceleration Computation) *Given a body twist, the corresponding spatial acceleration is Eq. 3.99.*

$$\mathcal{A}_{body} = \frac{d}{dt}(\mathcal{V}_{body}) = \begin{bmatrix} \dot{e}_{B_1} & ... & \dot{e}_{B_6} \end{bmatrix} \cdot {}^{B}\mathcal{V}_{body} + \begin{bmatrix} e_{B_1} & ... & e_{B_6} \end{bmatrix} \cdot \frac{d}{dt}({}^{B}\mathcal{V}_{body}) \tag{3.99}$$

If $\{B\}$-frame does not change then we get Eq. 3.100.

$$\dot{\mathcal{V}}_{body} = \begin{bmatrix} e_{B_1} & ... & e_{B_6} \end{bmatrix} \cdot {}^{B}\overset{\circ}{\mathcal{V}}_{body} \tag{3.100}$$

otherwise $\begin{bmatrix} \dot{e}_{B_1} & ... & \dot{e}_{B_6} \end{bmatrix}$ need to be computed.

Remark 3.1.34 *Considering the transformation ${}^{O}T_B = (R, p)$, the corresponding derivative $\begin{bmatrix} \dot{e}_{B_1} & ... & \dot{e}_{B_6} \end{bmatrix}$, using $\{O\}$-frame, when $\{B\}$-frame has instantaneous velocity $\mathcal{V}_B = \begin{bmatrix} \omega \\ v \end{bmatrix}$ is:*

$$\begin{aligned} \begin{bmatrix} {}^{O}\dot{e}_{B_1} & ... & {}^{O}\dot{e}_{B_6} \end{bmatrix} = {}^{O}\dot{X}_B &= \frac{d}{dt}[Ad_{{}^{O}T_B}] \\ &= \frac{d}{dt}\left(\begin{bmatrix} R & 0 \\ [p] \cdot R & R \end{bmatrix} \right) = \begin{bmatrix} \dot{R} & 0 \\ ([p] \cdot R)' & \dot{R} \end{bmatrix} \\ &= \begin{bmatrix} [\omega] & 0 \\ [v] & [\omega] \end{bmatrix} \cdot {}^{O}X_B \end{aligned} \tag{3.101}$$

Proof based on the following identities:

$$\begin{aligned} \dot{R} &= \omega \times R \\ \dot{p} &= v + \omega \times p \\ [R \cdot \omega] &= R \cdot [\omega] \cdot R^T \\ [w_1 \times w_2] &= [w_1] \cdot [w_2] - [w_2] \cdot [w_1] \end{aligned}$$

Remark 3.1.35 *We define $[\mathcal{V}_B \times]$ as the linear operator that corresponds to cross product induced by the twist $\mathcal{V}_B$ Eq. 3.102*

$$[\mathcal{V}_B \times] \overset{\Delta}{=} \begin{bmatrix} [\omega] & 0 \\ [v] & [\omega] \end{bmatrix} \tag{3.102}$$

s.t.

$${}^{O}\dot{X}_B = [\mathcal{V}_B \times] \cdot {}^{O}X_B \tag{3.103}$$

which is similar in construction with angular velocity ω in ***Instantaneous Velocity of Rotating Frames*** *3.1.34 in Eq. 3.52*

$$\frac{d}{dt}R_A(t) = [\omega_A] \cdot R_A(t)$$

Remark 3.1.36 *Equation 3.103 is* ***coordinate independent****.*

Definition 3.1.44 (Spatial Cross Product) *Given two spatial velocities, $\mathcal{V}_1$ and $\mathcal{V}_2$, their spatial cross product, corresponding to linear operator Eq. 3.102 from Eq. 3.103, is Eq. 3.104*

$$\mathcal{V}_1 \times \mathcal{V}_2 = [\mathcal{V}_1\times] \cdot \mathcal{V}_2 = \begin{bmatrix}\omega_1\\ v_1\end{bmatrix} \times \begin{bmatrix}\omega_2\\ v_2\end{bmatrix} \triangleq \begin{bmatrix}\omega_1 \times \omega_2\\ \omega_1 \times v_2 + v_1 \times \omega_2\end{bmatrix} \in \mathbb{R}^6 \tag{3.104}$$

Definition 3.1.45 (Spatial Cross Product Properties) *Assuming $\{A\}$-frame is moving w.r.t. $\{O\}$-frame with velocity $\mathcal{V}_A$ then we have the following properties:*

$$^O\dot{X}_A = [^O\mathcal{V}_A\times] \cdot {}^O X_A \tag{3.105}$$

$$[X \cdot \mathcal{V}\times] = X \cdot [\mathcal{V}\times] \cdot X^T \tag{3.106}$$

Definition 3.1.46 (Spatial Acceleration with Moving Reference Frame) *Considering a body with velocity $\mathcal{V}_{body}$ (w.r.t. inertial frame), and $^O\mathcal{V}_{body}$ and $^B\mathcal{V}_{body}$ be its Plücker coordinates w.r.t. $\{O\}$ and $\{B\}$ then the Eq. 3.107 is the spatial acceleration in $\{B\}$-frame and Eq. 3.108 is in $\{O\}$-frame.*

$$^B\mathcal{A}_{body} = \frac{d}{dt}\left(^B\mathcal{V}_{body}\right) + {}^B\mathcal{V}_B \times {}^B\mathcal{V}_{body} \tag{3.107}$$

$$^O\mathcal{A} = {}^O X_B \cdot {}^B\mathcal{A} \tag{3.108}$$

Proof

Eq 3.107 it's a direct result of applying Eq. 3.99 and Eq. 3.103.

For Eq. 3.108 we have:

$$\begin{aligned}
^O\mathcal{A}_{body} &= \frac{d}{dt}(^O\mathcal{V}_{body}) = \frac{d}{dt}(^O X_B \cdot {}^B\mathcal{V}_{body}) \\
&= {}^O\dot{X}_B \cdot {}^B\mathcal{V}_{body} + {}^O X_B \cdot {}^B\mathring{\mathcal{V}}_{body} \\
&= [^O\mathcal{V}_B\times] \cdot {}^O X_B \cdot {}^B\mathcal{V}_{body} + {}^O X_B \cdot {}^B\dot{\mathcal{V}}_{body} \\
&= {}^O X_B \cdot \{ \underbrace{{}^B X_O \cdot [^O\mathcal{V}_B] \cdot {}^O X_B}_{Eq.3.106 [{}^B X_O \cdot {}^O\mathcal{V}_B\times] = [{}^B\mathcal{V}_B\times]} \cdot {}^B\mathcal{V}_{body} + {}^B\mathring{\mathcal{V}}_{body}\} \\
&= {}^O X_B \cdot \{\underbrace{{}^B\mathring{\mathcal{V}}_{body} + {}^B\mathcal{V}_B \times {}^B\mathcal{V}_{body}}_{Eq.3.107 {}^B\mathcal{A}_{body}}\}
\end{aligned}$$

3.2 Rigid Body dynamics

3.2.1 Spatial Force (Wrench)

Definition 3.2.1 (Spatial Force) *Given a rigid body with many forces acting on it, and given an arbitrary point O in space Fig. 3.19, the net effect of these forces can be expressed as a force f, acting along a line passing through O Eq. 3.109, and torque n_O about point O Eq. 3.110*

$$f = \sum_i f_i \tag{3.109}$$

$$\tau_O = n_O = \sum_i (\vec{OP_i}) \times f_i \tag{3.110}$$

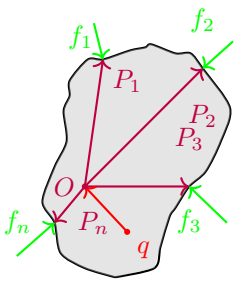


Figure 3.19: Rigid body

*We define as the **Spatial Force (Wrench)** the $6D$ vector of the form Eq. 3.111*

$$\mathcal{F} = \begin{bmatrix} n_O \\ f \end{bmatrix} \tag{3.111}$$

Remark 3.2.1 (Changing the torque reference point) *Given the torque around reference point n_O, we can computer the torque around another point n_q as follows Eq. 3.112.*

$$\begin{aligned} n_q &= \sum_i (q\vec{P_i}) \times f_i = n_O + \sum_i (q\vec{P_i} - O\vec{P_i}) \times f_i \\ &= n_O + q\vec{O} \times \sum_i = n_O + q\vec{O} \times f \\ &= n_0 + f \times \vec{Oq} \end{aligned} \tag{3.112}$$

Definition 3.2.2 (Spatial Force in Plücker Coordinate Systems) *Given a frame $\{A\}$, the Plücker coordinate of a spacial force $\mathcal{F}$ is given by Eq. 3.113*

$$ {}^{A}\mathcal{F} = \begin{bmatrix} {}^{A}n_{O_A} \\ {}^{A}f \end{bmatrix} \tag{3.113} $$

For the transformation ${}^{A}T_B = ({}^{A}R_B, {}^{A}p_B)$, Fig. 3.20 the corresponding ***spatial force*** *coordinate transformation is Eq. 3.114.*

$$ {}^{A}\mathcal{F} = {}^{A}X_B^* \cdot {}^{B}\mathcal{F} \tag{3.114} $$

where

$$ {}^{A}X_B^* = {}^{B}X_A^T \tag{3.115} $$

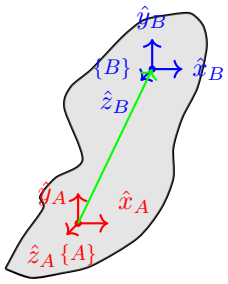


Figure 3.20: spatial force coordinate transformation

Proof Eq. 3.114:

$$ \textit{Force: } {}^{A}f = {}^{A}R_B \cdot {}^{B}f \textit{ (free vector)} \tag{3.116} $$

$$ \textit{Torque: } {}^{A}n_{O_A} = {}^{A}n_{O_B} + {}^{A}(O_A\vec{O}_B) \times {}^{A}f $$

$$ = {}^{A}R_B \cdot {}^{B}n_{O_A} + {}^{A}R_B \cdot (-{}^{B}p_A \times {}^{B}f) \tag{3.117} $$

$$ = {}^{A}R_B \cdot {}^{B}n_{O_A} - {}^{A}R_B \cdot [{}^{B}p_A] \cdot {}^{B}f \tag{3.118} $$

$$ \overset{Eq.3.116 3.118}{\Longrightarrow} \begin{bmatrix} {}^{A}n_{O_A} \\ {}^{A}f \end{bmatrix} = \underbrace{\begin{bmatrix} {}^{A}R_B & -{}^{A}R_B \cdot [{}^{B}p_A] \\ 0 & {}^{A}R_B \end{bmatrix}}_{\triangleq {}^{A}X_B^*} \cdot \begin{bmatrix} {}^{B}n_{O_B} \\ {}^{B}f \end{bmatrix} \tag{3.119} $$

Definition 3.2.3 (Power: Wrench-Twist pair) *Suppose a rigid body has a twist* ${}^{A}\mathcal{V} = ({}^{A}\omega, {}^{A}v_{O_A})$ *and a wrench* ${}^{A}\mathcal{F} = ({}^{A}n_{O_A}, {}^{A}f)$ *acting on the body. Then the power is Eq. 3.120*

$$P = ({}^{A}\mathcal{V})^T \cdot {}^{A}\mathcal{F} \tag{3.120}$$

$$= \underbrace{({}^{A}\omega)^T \cdot {}^{A}n_{O_A}}_{rotationalPower} + \underbrace{{}^{A}v_{O_A}^T \cdot {}^{A}f}_{linearpower} \tag{3.121}$$

Remark 3.2.2 *Power is a coordinate-free quantity.*

Definition 3.2.4 (Joint Torque) *Consider a ling attached to a* 1*-degree of freedom joint (revolute or prismatic). Let* $\hat{\mathcal{S}}$ *be the screw axis of the joint. The velocity of the link induced by the point motion is given by* $\mathcal{V} = \hat{\mathcal{S}} \cdot \dot{\theta}$ *and* $\mathcal{F}$ *is the wrench provided by the joint Fig. 3.21. We can define the power produced by the joint as Eq. 3.122.*

$$P = \mathcal{V}^T \cdot \mathcal{F} = \underbrace{\hat{\mathcal{S}}^T \cdot \mathcal{F}}_{\triangleq \tau \ (scalar)} \cdot \dot{\theta} \triangleq \tau \cdot \dot{\theta} \tag{3.122}$$

where :

$$\tau = \hat{\mathcal{S}}^T \cdot \mathcal{F} = \mathcal{F}^T \cdot \hat{\mathcal{S}} \tag{3.123}$$

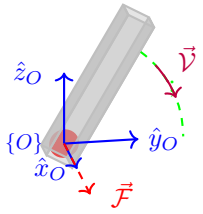


Figure 3.21: Joint Torque

We refer to τ *as* ***joint torque*** *or* ***generalized force*** *and it represents the projection of the wrench onto the screw axis, i.e. the effective part of the wrench.*

3.2.2 Spatial Momentum

	Linear	Rotational
Velocity:	$v = \dot{r}$	$\omega = \hat{\omega} \cdot \dot{\theta}, \;\; v = \omega \times r$
Acceleration:	$a = \ddot{r}$	$\alpha = \dot{\omega}$
Force / Torque	$f = m \cdot a$	$n = r \times f$
Momentum	$L = m \cdot v$	$\phi = r \times L$
		$= r \times (m \cdot v)$
		$= r \times (m \cdot \omega \times r)$
		$= m \cdot (r \times \omega \times r)$
		$= m \cdot r \times (-r) \times \omega$
		$= m \cdot [r] \cdot [-r] \cdot \omega$
Inertia Matrix $\bar{I} \in \mathbb{R}^{3\times3}$		$\bar{I} = m \cdot [r] \cdot [-r]$

Table 3.1: Point-mass kinematic / dynamic properties Fig. 3.22

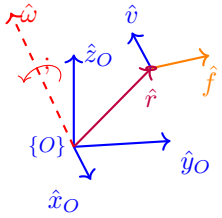


Figure 3.22: Point-mass

Definition 3.2.5 (Rotational Inertia) *Based on table 3.1, we define the* ***Rotational Inertia*** *of a rigid body as Eq. 3.124 where ρ is the density function.*

$$\bar{I} \triangleq \int_V \rho \cdot [r] \cdot [-r]^T dv \tag{3.124}$$

Remark 3.2.3 *$\bar{I}$ depends on the coordinate system.*

Remark 3.2.4 *$\bar{I}$ is a constant matrix if the origin of the reference frame coincides with the center of mass (CoM)*

Definition 3.2.6 (Center of Mass) *A point C is defined as the center of mass (CoM) as Eq. 3.125*

$$CoM \triangleq \frac{1}{m} \sum_i m_i \cdot r_i \tag{3.125}$$

Remark 3.2.5 *If C is the **center of mass**, then we have the following properties:*

$$\frac{1}{m} \cdot \sum_i m_i \cdot (\vec{Cr_i}) = 0 \tag{3.126}$$

$$\sum_i m_i \cdot (\vec{Cr_i}) = 0 \implies \sum_i m_i \cdot [\vec{Cr_i}] = 0 \tag{3.127}$$

Proof:
For Eq. 3.126:

$$\overset{From Eq. 3.125}{\implies} \frac{1}{m} \cdot \sum_i m_i \cdot r_i - C = 0 \implies \sum_i m_i \cdot r_i - m \cdot C = 0$$
$$\implies \sum_i m_i \cdot (r_i - C) = 0 \implies \sum_i m_i \cdot (Cr_i) = 0$$

For Eq. 3.127:

$$\overset{From Eq. 3.126}{\implies} [\sum m_i \cdot (Cr_i)] = [0] \implies \sum [m_i \cdot (Cr_i)] = [0] \implies \sum m_i \cdot [(Cr_i)] = [0]$$

Definition 3.2.7 (Momentum) *Given a rigid body with spatial velocity $\mathcal{V}_C = (\omega, v_C)$ expressed at the center of mass C, the **linear momentum** is defined as Eq. 3.128 and the **angular momentum** about center of mass is defined as Eq. 3.129*

$$L = m \cdot v_C \tag{3.128}$$

$$\phi = \bar{I}_C \cdot \omega \tag{3.129}$$

Proof:
Eq. 3.128:

$$L = \sum_i m_i \cdot v_i = \sum_i m_i \cdot (v_C + \omega \times \vec{Cr_i}) = (\sum_i m_i) \cdot v_i + \underbrace{\sum_i m_i \cdot (-\vec{Cr_i})}_{\overset{Eq. 3.126}{=} 0} \times \omega$$

Eq. 3.129:

$$\phi_C = \sum_i \vec{Cr_i} \times L = \sum_i \vec{Cr_i} \times (m_i \cdot v_i) = \sum_i \vec{Cr_i} \times (m_i \cdot v_C + m_i \cdot \omega \times \vec{Cr_i})$$
$$= \underbrace{\sum_i \vec{Cr_i} \times (m_i \cdot v_C)}_{\overset{Eq. 3.126}{=} 0} + \sum_i m_i \cdot \vec{Cr_i} \times \omega \times \vec{Cr_i} = \sum_i m_i \cdot \vec{Cr_i} \times \omega \times \vec{Cr_i} = \bar{I}_C \cdot \omega$$

Remark 3.2.6 *Given an arbitrary point O, the angular momentum about it relative to the center of mass C is Eq. 3.130 Fig. 3.23.*

$$\phi_O = \phi_C + \vec{OC} \times L \tag{3.130}$$

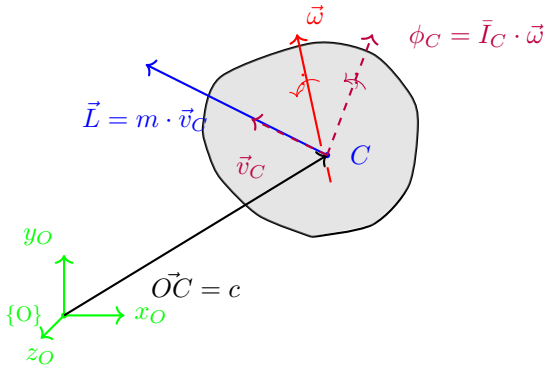


Figure 3.23: Angular momentum about an arbitrary point

Definition 3.2.8 (Spatial Momentum) *Given the **angular momentum** ϕ and the **linear momentum** of a rigid body, we define the corresponding **Spatial Momentum** h as Eq. 3.131.*

$$h = \begin{bmatrix} \phi \\ L \end{bmatrix} \in \mathbb{R}^6 \tag{3.131}$$

Definition 3.2.9 (Change Reference Frame for Spatial Momentum) *Under transformation ${}^A T_C = ({}^A RC, {}^A p_C)$ the corresponding **spatial momentum transformation** is Eq. 3.132 which is the same operator required by **spatial forces**.*

$${}^A h = {}^A X_C^* \cdot {}^C h \tag{3.132}$$

Proof:

$${}^A L = {}^A R_C \cdot {}^C L \quad \textit{(free vector)} \tag{3.133}$$

$$\overset{Eq.3.130}{\Longrightarrow} {}^A \phi_{O_A} = {}^A R_C \cdot {}^C \phi_{O_C} + {}^A R_C \cdot (-{}^C p_A \times {}^C L) \tag{3.134}$$

$$\overset{\textit{Using: Eq.3.133 and Eq.3.134}}{\Longrightarrow} \begin{bmatrix} {}^A \phi_{O_A} \\ {}^A L \end{bmatrix} = \underbrace{\begin{bmatrix} {}^A R_C & {}^A R_C \cdot [-{}^C p_A] \\ 0 & {}^A R_C \end{bmatrix}}_{{}^A X_C^*} \cdot \begin{bmatrix} {}^C \phi_{O_C} \\ {}^C L \end{bmatrix} \tag{3.135}$$

Definition 3.2.10 (Spatial Inertia) *Spatial inertia of a rigid body defines the linear mapping Fig. 3.24 between spatial velocity (twist) and spatial momentum Eq. 3.136, connecting* ***Moving Space****:* $\mathcal{M}$ *and* ***Force Space****:* $\mathcal{F}$ *and it has the following form Eq. 3.137.*

$$h = \mathcal{I} \cdot \mathcal{V} \tag{3.136}$$

$${}^{C}\mathcal{I} \overset{By\ Eq.3.128\ and\ Eq.3.129}{=} \begin{bmatrix} {}^{C}\bar{I}_{C} & 0 \\ 0 & m \cdot I_3 \end{bmatrix} \tag{3.137}$$

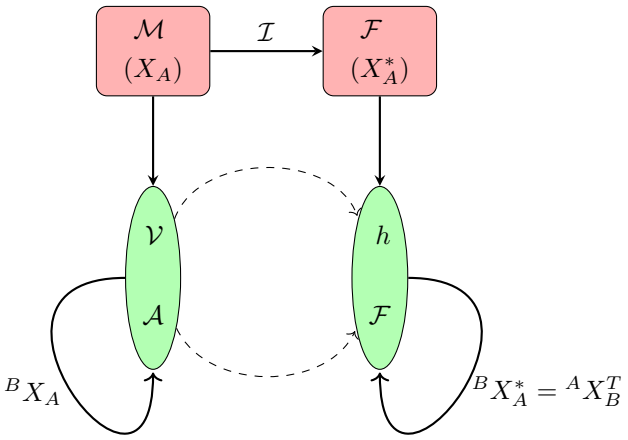


Figure 3.24: Spatial inertia mapping

Definition 3.2.11 (Changing Reference Frame for Spatial Inertia) *Given to 2 frames* $\{C\}$ *respectively,* $\{A\}$ *the spatial inertia representation in one frame w.r.t. the other is given by Eq. 3.138.*

$${}^{A}\mathcal{I} = {}^{A}X_C^* \cdot {}^{C}\mathcal{I} \cdot {}^{C}X_A \tag{3.138}$$

Proof:

$$\begin{aligned} {}^{A}h &= {}^{A}\mathcal{I} \cdot {}^{A}\mathcal{V} \\ &= {}^{A}X_C^* \cdot {}^{C}h \\ &= {}^{A}X_C^* \cdot {}^{C}\mathcal{I} \cdot {}^{C}\mathcal{V} \\ &= \underbrace{{}^{A}X_C^* \cdot {}^{C}\mathcal{I} \cdot {}^{C}X_A}_{={}^{A}\mathcal{I}} \cdot {}^{A}\mathcal{V} \end{aligned} \tag{3.139}$$

Eq. 3.139 can be interpreted using spatial inertia mapping Fig. 3.24. Spatial momentum in $\{A\}$-frame is the application of spatial velocity in $\{A\}$-frame transformed into $\{C\}$-frame, computing the spatial momentum in $\{C\}$-frame and in the end, going back to $\{A\}$-frame.

Remark 3.2.7 *When $\{A\}$-frame has the same orientation as the orientation of the center of mass frame (${}^{A}R_{C} = I_3$), we have Eq. 3.140. The change of reference frame of spatial inertia representation has an explicit form Eq. 3.141*

$$ {}^{A}X_{C} = \begin{bmatrix} I_3 & 0 \\ [{}^{A}p_{C} & I_3] \end{bmatrix} \tag{3.140} $$

$$ {}^{A}\mathcal{I} = \begin{bmatrix} {}^{C}\bar{I} + m \cdot [{}^{A}p_{C}] \cdot [{}^{A}p_{C}]^{T} & m \cdot [{}^{A}p_{C}] \\ m \cdot [{}^{A}p_{C}]^{T} & m \cdot I_3 \end{bmatrix} \tag{3.141} $$

Definition 3.2.12 (Spatial Force differentiation) *Given a **spatial force vector** $\mathcal{F}$, expressed using **spatial force bases vectors** $\begin{bmatrix} e^{*}_{B_1} & e^{*}_{B_2} & ... & e^{*}_{B_6} \end{bmatrix} = X^{*}_{B}$, its corresponding differential $\dot{\mathcal{F}}$ is Eq. 3.142*

$$ \dot{\mathcal{F}} = \dot{X}^{*}_{B} \cdot {}^{B}\mathcal{F} + X^{*}_{B} \cdot \underbrace{({}^{B}\mathcal{F})'}_{{}^{B}\mathring{\mathcal{F}} \text{ :apparent derivative of spatial force}} \tag{3.142} $$

Remark 3.2.8 *The differential of **spatial force bases vectors** $\dot{e}^{*}_{B_i}$ is Eq. 3.143 where $\times^{*}$ operator is the **cross product** between current **spatial velocity** and applied **spatial force** Eq. 3.144. By defining Eq. 3.145 we can rewrite $\dot{X}^{*}_{B}$ as Eq. 3.146 which resembles in form with the **differential of motion basis** 3.103.*

$$ \dot{e}^{*}_{B_i} = \mathcal{V}_{B} \times^{*} \cdot e^{*}_{B_i} \tag{3.143} $$

$$ \mathcal{V} \times^{*} \mathcal{F} \triangleq \begin{bmatrix} \omega \times n_{O} + v_{O} \times f \\ \omega \times f \end{bmatrix} \tag{3.144} $$

$$ [\mathcal{V}\times^{*}] = \begin{bmatrix} [\omega] & [v_o] \\ 0 & [\omega] \end{bmatrix} \tag{3.145} $$

$$ \dot{X}^{*}_{B} = [\mathcal{V}\times^{*}_{B}] \cdot X^{*}_{B} \tag{3.146} $$

Proof:

The approach is similar to the proof sketched in 3.1.34 or it can be deduced using physics.

Remark 3.2.9 *Spatial force cross-product has the following property:*

$$ [\mathcal{V}\times^{*}] = -[\mathcal{V}\times]^{T} \tag{3.147} $$

Remark 3.2.10 *By using Definition 3.2.12 and Remark 3.2.8 one can compute the differentiation of spatial force as Eq. 3.2.8*

$$\dot{\mathcal{F}} = X_B^* \cdot {}^B\overset{\circ}{\mathcal{F}} + [\mathcal{V}_B \times^*] \cdot X_B^* \cdot {}^B\mathcal{F} \tag{3.148}$$

Remark 3.2.11 *Assuming $\{A\}$-frame is moving with velocity ${}^A\mathcal{V}_A$ then the derivative of $\mathcal{F}$ in $\{A\}$-frame is Eq. 3.149 and the derivative of h in $\{A\}$ is 3.150.*

$${}^A \underbrace{\left(\frac{d}{dt}\mathcal{F}\right)}_{coordinate\ free} = \underbrace{\frac{d}{dt}({}^A\mathcal{F})}_{{}^A X_A^* \cdot {}^A\overset{\circ}{\mathcal{F}}} + \underbrace{{}^A\mathcal{V}_A \times^* {}^A\mathcal{F}}_{[{}^A\mathcal{V}_A \times^*] \cdot {}^A X_A^* \cdot {}^A\mathcal{F}} \tag{3.149}$$

$${}^A\left(\frac{d}{dt}h\right) = \underbrace{\frac{d}{dt}({}^A h)}_{{}^A X_A \cdot \frac{d}{dt}({}^A h)} + \underbrace{\underbrace{{}^A\mathcal{V}_A \times^* {}^A h}_{[{}^A\mathcal{V}_A \times^*] \cdot {}^A X_A^*}}_{({}^A X_A^*)'} \cdot {}^A h \tag{3.150}$$

3.2.3 Newton-Euler Equation

Definition 3.2.13 (Newton-Euler equation) *By adopting the spatial vectors, the Newton-Euler equation has the same form in any frame Eq. 3.151, on the other hand, most of the other kinds of derivations end up with a more complex form.*

$$\mathcal{F} = \frac{d}{dt}h = \mathcal{I} \cdot \mathcal{A}_{body} + \mathcal{V}_{body} \times^* \mathcal{I} \cdot \mathcal{V}_{body} \tag{3.151}$$

Choosing an arbitrary reference frame $\{B\}$ we get Eq. 3.152

$${}^B\mathcal{F} = {}^B\mathcal{I} \cdot {}^B\mathcal{A}_{body} + {}^B\mathcal{V}_{body} \times^* {}^B\mathcal{I} \cdot {}^B\mathcal{V}_{body} \tag{3.152}$$

The Newton-Euler Eq. 3.151 remains the same even when the chosen frame of reference is away from the center of mass.

Proof: Eq. 3.151

$$\begin{aligned} {}^B\mathcal{F} = {}^B\left(\frac{d}{dt}h\right) = {}^B\left(\frac{d}{dt}(\mathcal{I} \cdot \mathcal{V})\right) = {}^B(\mathcal{I} \cdot \mathcal{A} + \dot{\mathcal{I}} \cdot \mathcal{V}) \\ \overset{\substack{This\ equality \\ must\ be \\ proven}}{=} {}^B\left(\underbrace{\mathcal{I} \cdot \mathcal{A}}_{\substack{due\ to\ change \\ in\ velocity\ \mathcal{V}}} + \underbrace{\mathcal{V} \times^* \mathcal{I} \cdot \mathcal{V}}_{\substack{accounts\ for\ the\ fact \\ that\ inertia\ is\ moving}}\right) \end{aligned} \tag{3.153}$$

We introduce an inertial frame $\{O\}$, w.r.t. which the computation will be done and we consider $\{B\}$ asa frame attached to the body which imply that $\mathcal{V}_{body} = \mathcal{V}_B$ and ${}^B\mathcal{I}$ is constant Fig. 3.25.

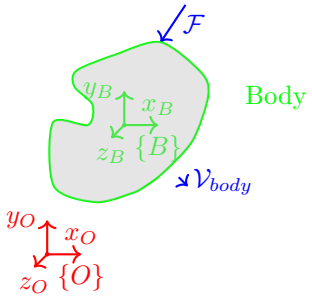


Figure 3.25: Newton-Euler in inertial frame

$$
\begin{aligned}
&\textit{then}: \\
\frac{d}{dt}({}^{O}h) &= \frac{d}{dt}({}^{O}\mathcal{I} \cdot {}^{O}\mathcal{V}) \qquad (3.154)\\
&= {}^{O}\dot{\mathcal{I}} \cdot {}^{O}\mathcal{V} + {}^{O}\mathcal{I} \cdot {}^{O}\mathcal{A} \\
&\overset{\substack{\textit{since } {}^{O}\dot{\mathcal{I}} \textit{ changes} \\ \textit{w.r.t. time,} \\ \textit{we switch from} \\ \{O\} \textit{ to } \{B\}}}{\Longrightarrow} \\
&= \frac{d}{dt}(\underbrace{{}^{O}X_B^* \cdot {}^{B}\mathcal{I} \cdot {}^{B}X_O}_{Eq.3.138}) \cdot {}^{O}\mathcal{V} + {}^{O}\mathcal{I} \cdot {}^{O}\mathcal{A} \\
&= {}^{O}\dot{X}_B^* \cdot {}^{B}\mathcal{I} \cdot {}^{B}X_O \cdot {}^{O}\mathcal{V} \\
&\quad + {}^{O}X_B^* \cdot {}^{B}\mathcal{I} \cdot \underbrace{{}^{B}\dot{X}_O}_{\substack{Remark \\ 3.2.12}} \cdot {}^{O}\mathcal{V} + {}^{O}\mathcal{I} \cdot {}^{O}\mathcal{A} \\
&= [{}^{O}\mathcal{V}_B \times^*] \cdot \underbrace{{}^{O}X_B^* \cdot {}^{B}\mathcal{I} \cdot {}^{B}X_O}_{={}^{O}\mathcal{I}} \cdot {}^{O}\mathcal{V} \\
&\quad - {}^{O}X_B^* \cdot {}^{B}\mathcal{I} \cdot {}^{B}X_O \cdot \underbrace{[{}^{O}\mathcal{V}_B \times] \cdot \overbrace{{}^{O}\mathcal{V}}^{{}^{O}\mathcal{V}={}^{O}\mathcal{V}_B}}_{=0} + {}^{O}\mathcal{I} \cdot {}^{O}\mathcal{A} \\
&= [{}^{O}\mathcal{V}_B \times^*] \cdot {}^{O}\mathcal{I} \cdot {}^{O}\mathcal{V} + {}^{O}\mathcal{I} \cdot {}^{O}\mathcal{A} \\
&= {}^{O}\mathcal{I} \cdot {}^{O}\mathcal{A} + {}^{O}\mathcal{V}_B \times^* {}^{O}\mathcal{I} \cdot {}^{O}\mathcal{V} \qquad (3.155)
\end{aligned}
$$

Remark 3.2.12 (Note) *It can't be computed directly so instead, we rely on the implicit, product rule differentiation Eq. 3.156, to reduce the result to ${}^{O}\dot{X}_{B}$ Eq. 3.103.*

$$\implies ({}^{O}X_{B} \cdot {}^{B}X_{O}) = I \tag{3.156}$$

$$\implies {}^{O}\dot{X}_{B} \cdot {}^{B}X_{O} + {}^{O}X_{B} \cdot {}^{B}\dot{X}_{O} = 0$$

$$\implies {}^{B}\dot{X}_{O} = -{}^{B}X_{O} \cdot {}^{O}\dot{X}_{B} \cdot {}^{B}X_{O}$$

$$= -{}^{B}X_{O} \cdot \underbrace{[{}^{O}\mathcal{V}_{B}\times] \cdot {}^{O}X_{B}}_{Eq.\ 3.103} \cdot {}^{B}X_{O}$$

$$= -{}^{B}X_{O} \cdot [{}^{O}\mathcal{V}_{B}\times] \tag{3.157}$$

3.3 Multi-body dynamics of Open Chains

3.3.1 Open-Chain Dynamics

Definition 3.3.1 (Open Chain) *Consist of multiple rigid links (bodies) connected through joints (revolute/prismatic) Fig. 3.26. The corresponding equations of motion are a set of 2^{nd}-order differential equations Eq. 3.158*

$$\tau = M(\theta) \cdot \ddot{\theta} + \tilde{c}(\theta, \dot{\theta}) \tag{3.158}$$

$$where:$$

$$\tilde{c}(\theta, \dot{\theta}) = c(\theta, \dot{\theta}) + \tau_{g}(\theta) + J^{T} \cdot \mathcal{F}_{ext} \tag{3.159}$$

$\theta \in \mathbb{R}^{n}$: *vector of joint variables*

$\tau \in \mathbb{R}^{n}$: *vector of joint forces / torques*

$M(\theta) \in \mathbb{R}^{n \times n}$: *mass matrix*

$\tilde{c}(\theta, \dot{\theta}) \in \mathbb{R}^{n}$: (*forces that lump together centripetal, Coriolis, gravity, friction terms, and torques induced by external forces*)

Definition 3.3.2 (Forward Dynamics) *Is the process of determining the acceleration $\ddot{\theta}$ given the state $(\theta, \dot{\theta})$ as well as the joint forces/torques Eq. 3.160.*

$$\ddot{\theta} = FD(\tau, \theta, \dot{\theta}, \mathcal{F}_{ext}) \tag{3.160}$$

Definition 3.3.3 (Inverse Dynamics) *Is the process of finding torques/forces given the state $(\theta, \dot{\theta})$ and desired acceleration $\ddot{\theta}$ Eq. 3.161.*

$$\tau = ID(\theta, \dot{\theta}, \ddot{\theta}, \mathcal{F}_{ext}) \tag{3.161}$$

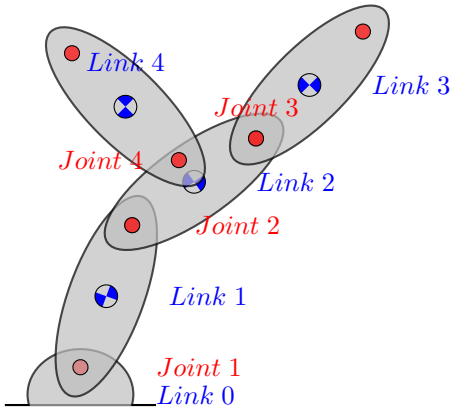


Figure 3.26: Open Chain

Definition 3.3.4 (Lagrangian *versus* Newton-Euler Methods) *There are two ways to derive the equation of motion for an open-chain robot, each of them with advantages and disadvantages:*

Lagrangian Formulation	***Newton-Euler Formulation***
-Energy based method	*-Balance of forces/torques*
-Dynamic equation in closed form	*-Dynamic equations in numeric/recursive form*
-Often used for study of dynamic properties and analysis of control methods	*-Often used for numerical solution of forward / inverse dynamics*

Remark 3.3.1 (Mixed approach) *Apart from the Lagrangian formulation, which can be interpreted as a top-down, reduced formulation, starting from the entire system at once, and the Newtonian, where we build the equation button-up, in a redundant way, but using an iterative approach, one can choose a* 3^{rd} *way, a mixed approach, in which we still use the Newtonian way, but we choose to characterize the degree of freedom of each rigid body in a reduced way. This leads to a result where we get to use the best of both worlds, as the computation process remains iterative while the result is computed in a reduced manner. In a nutshell, this is what we obtain by using Featherstone Algebra.*

3.3.2 Inverse Dynamics

Remark 3.3.2 (Chain notations) *Given an Open Chain with N links, we define:*

- *Parent of body i:* $p(i)$
- *Children of body i:* $c(i)$

- *Joint i connects $p(i)$ to i*
- *Frame $\{i\}$ is attached to body i at the corresponding joint*
- *$\mathcal{S}_i$ is the spatial velocity of joint i*
- *$\mathcal{V}_i$ and $\mathcal{A}_i$ are the spatial velocity and spatial acceleration of body i*
- *$\mathcal{F}_i$ is the wrench onto body i from body $p(i)$*

Note: *By default, all vectors $(\mathcal{S}_i, \mathcal{V}_i, \mathcal{F}_i)$ are expressed in local frame $\{i\}$.*

Definition 3.3.5 (Velocity and Acceleration propagation - *Forward Pass*) *Given the **joint velocity** $\dot{\theta}$ and the **joint acceleration** $\ddot{\theta}$ for an open chain we compute the body spatial velocity $\mathcal{V}_i$ and the spatial acceleration $\mathcal{A}_i$ as follows: Eq. 3.162 and Eq. 3.163.*

$$ {}^{i}\mathcal{V}_i = \left({}^{i}X_{p(i)}\right) \cdot {}^{p(i)}\mathcal{V}_{p(i)} + {}^{i}\mathcal{S}_i \cdot \dot{\theta}_i \tag{3.162} $$

$$ {}^{i}\mathcal{A}_i = \left({}^{i}X_{p(i)}\right) \cdot {}^{p(i)}\mathcal{A}_{p(i)} + {}^{i}\mathcal{V}_i \times {}^{i}\mathcal{S}_i \cdot \dot{\theta}_i + {}^{i}\mathcal{S}_i \cdot \ddot{\theta}_i \tag{3.163} $$

Proof:

$$
\begin{aligned}
\overset{3.162}{\Longrightarrow}\ \mathcal{V}_1 &= \mathcal{S}_1 \cdot \dot{\theta}_1 \\
\mathcal{V}_i &= \sum_{j=1}^{i} \mathcal{V}_{j/(j-1)} = \sum_{j=1}^{(i-1)} \mathcal{V}_{j/(j-1)} + \mathcal{V}_{i/(i-1)} = \mathcal{V}_{(i-1)} + \mathcal{V}_{i/(i-1)} \\
&= \mathcal{V}_{(i-1)} + \mathcal{S}_i \cdot \dot{\theta}_i \ \textit{(coordinate free)} \\
\overset{3.163}{\Longrightarrow}\ \mathcal{A}_i &= \dot{\mathcal{V}}_i = \dot{\mathcal{V}}_{(i-1)} + \dot{\mathcal{V}}_{i/(i-1)} = \mathcal{A}_{(i-1)} + \mathcal{A}_{i/(i-1)} \\
{}^{i}\mathcal{A}_i &= {}^{i}X_{(i-1)} \cdot {}^{(i-1)}\mathcal{A}_{(i-1)} + {}^{i}\left(\frac{d}{dt}(\underbrace{{}^{i}\mathcal{S}_2 \cdot \dot{\theta}}_{{}^{i}\mathcal{V}_{i/(i+1)}}) \right) \\
&= {}^{i}X_{(i-1)} \cdot {}^{(i-1)}\mathcal{A}_{(i-1)} + \underbrace{\frac{d}{dt}\left({}^{i}\mathcal{S}_i \cdot \dot{\theta}_i\right)}_{\textit{apparent derivative}} + \underbrace{{}^{i}\mathcal{V}_i \times {}^{i}\mathcal{S}_i \cdot \dot{\theta}_i}_{\textit{derivative of the frame}} \\
&= {}^{i}X_{(i-1)} \cdot {}^{(i-1)}\mathcal{A}_{(i-1)} + {}^{i}\mathcal{S}_i \cdot \ddot{\theta} + {}^{i}\mathcal{V}_i \times {}^{i}\mathcal{S}_i \cdot \dot{\theta}_2
\end{aligned} \tag{3.164}
$$

Definition 3.3.6 (Force Propagation - *Backward Pass*) *Given body spatial velocity $\mathcal{V}_i$ and spatial acceleration $\mathcal{A}_i$ for an open chain with N links, one must compute the joint wrench $\mathcal{F}_i$ and the corresponding torque $\tau_i = \mathcal{S}_i^T \cdot \mathcal{F}_i$ as follows Eq. 3.165 and Eq. 3.166*

$$ \mathcal{F}_i = \mathcal{I}_i \cdot \mathcal{A}_i + \mathcal{V}_i \times^{*} \mathcal{I}_i \cdot \mathcal{V}_i + \sum_{j \in c(i)} \mathcal{F}_j \tag{3.165} $$

$$ \tau_i = \mathcal{S}_i^T \cdot \mathcal{F}_i \tag{3.166} $$

Proof: Eq. 3.165 is the direct application of the second law of Newtonian mechanics.

Remark 3.3.3 *By putting together the previous 2 steps,* ***Forward Pass*** *and* ***Backward Pass****, we end up with the solution for the* ***Inverse Dynamics*** *problem in the form of the* ***Recursive Newton-Euler Algorithm*** *Fig. 3.27 that can be resumed as Alg. 2 where the gravity is incorporated into the virtual body 0 as a way of integrating gravity in a generically. Another option is to consider $\mathcal{A}_0 = 0$ and to add ${}^i\mathcal{I}_i \cdot {}^iX_0 \cdot {}^0\mathcal{A}_g$ to line 6.*

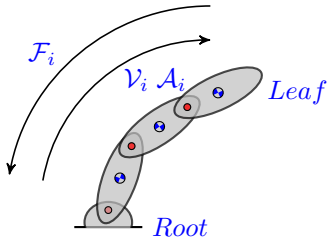


Figure 3.27: Recursive Newton-Euler Algorithm

Remark 3.3.4 *The time complexity of the Recursive Newton-Euler Algorithm is $O(N)$ in the number of joints.*

Remark 3.3.5 *As an example, how this is applied for Fig. 3.26, we have:*

Body 3:

$$\begin{aligned}
\mathcal{F}_3 + \mathcal{F}_{g_3} &= \mathcal{I}_3 \cdot \mathcal{A}_3 + \mathcal{V}_3 \times^* \mathcal{I}_3 \cdot \mathcal{V}_3 - \mathcal{F}_3^{ext} \\
\implies \mathcal{F}_3 &= \mathcal{I}_3 \cdot \mathcal{A}_3 + \mathcal{V}_3 \times^* \mathcal{I}_3 \cdot \mathcal{V}_3 - \mathcal{F}_{g_3} - \mathcal{F}_3^{ext} \\
\tau_3 &= \mathcal{S}_3^T \cdot \mathcal{F}_3 \\
(\textit{Note: } \mathcal{F}_{g_3} &= \mathcal{I}_3 \cdot {}^3\mathcal{A}_g = \mathcal{I}_3 \cdot {}^3X_0 \cdot {}^0\mathcal{A}_g)
\end{aligned}$$

Body 2:

$$\mathcal{F}_2 = \underbrace{\mathcal{I}_2 \cdot \mathcal{A}_2 + \mathcal{V}_2 \times^* \mathcal{I}_2 \cdot \mathcal{V}_2}_{\textit{used to produce motion}} + \underbrace{\overbrace{\mathcal{F}_3 + \mathcal{F}_4}^{\textit{to the environment}} - \overbrace{\mathcal{F}_{g_2} - \mathcal{F}_2^{ext}}^{\textit{by the environment}}}_{\textit{used to counteract forces}}$$

$$\tau_2 = \mathcal{S}_2^T \cdot \mathcal{F}_2$$

Remark 3.3.6 *Since the computation of τ can be seen as the projection of Spatial Force to feasible space (the action that happens only in the direction where the screw can do work) and since the screws are vector spaces (since screws are characterized by Lie algebras which are vector spaces as well) we can conclude that the projection can be computed using the dot product. This process*

has an unexpected result since it allows external forces, which can happen due to interaction with the environment, to be treated in e general way, by projecting them as well on the feasible space without being worried about possible changes in the dynamics of the system (no need for a multi-phase system).

Algorithm 2 Recursive Newton-Euler Algorithm

Require: $\theta, \dot{\theta}, \ddot{\theta}, \mathcal{F}_{ext}, Model$
Ensure: $\tau, \mathcal{F}$
1: $\mathcal{V}_0 \leftarrow 0$
2: $\mathcal{A}_0 \leftarrow -A_g$
3: **for** i=1,N **do**
4: $\quad {}^i\mathcal{V}_i \leftarrow {}^iX_{p(i)} \cdot {}^{p(i)}\mathcal{V}_{p(i)} + {}^i\mathcal{S}_i \cdot \dot{\theta}_i$
5: $\quad {}^i\mathcal{A}_i \leftarrow {}^iX_{p(i)} \cdot {}^{p(i)}\mathcal{A}_{p(i)} + {}^i\mathcal{S}_i \cdot \ddot{\theta}_i + {}^i\mathcal{V}_i \times {}^i\mathcal{S}_i \cdot \dot{\theta}_i$
6: $\quad {}^i\mathcal{F}_i \leftarrow {}^i\mathcal{I}_i \cdot {}^i\mathcal{A}_i + {}^i\mathcal{V}_i \times^* {}^i\mathcal{I}_i \cdot {}^i\mathcal{V}_i$
7: **end for**
8: **for** i=N,1 **do**
9: $\quad \tau_i \leftarrow {}^i\mathcal{S}_i^T \cdot {}^i\mathcal{F}_i$
10: $\quad {}^i\mathcal{F}_{p(i)} = {}^i\mathcal{F}_{p(i)} + {}^{p(i)}X_i^* \cdot {}^i\mathcal{F}_i$
11: **end for**

3.3.3 Forward Dynamics

Remark 3.3.7 *One way to compute the Forward Dynamics equation specific for a dynamical system is by evaluating, multiple times, the corresponding Inverse Dynamics, Eq. 3.160, using the RNEA algorithm by selecting the input parameters in Eq. 3.167 s.t. the resulting τ to correspond either to M or $\tilde{C}$ parameter.*

$$\begin{aligned}\tau &= M(\theta)\cdot\ddot{\theta} + C(\theta,\dot{\theta})\cdot\dot{\theta} + \tau_g + J^T(\theta)\cdot\mathcal{F}_{ext} \\ &= RNEA(\theta,\dot{\theta},\ddot{\theta},\mathcal{F}_{ext}) \\ &= M(\theta)\cdot\ddot{\theta} + \tilde{C}(\theta,\dot{\theta})\end{aligned} \tag{3.167}$$

Remark 3.3.8 *Since the Inverse Dynamics, computed with $RNEA(\theta,\dot{\theta},\ddot{\theta},\mathcal{F}_{ext})$ algorithm, can work directly with a given URDF model (a file format that defines the kinematic tree, joint model and the dynamic parameters of each link) in an algorithmic way, an explicit formula for $M(\theta)$ and $\tilde{C}(\theta,\dot{\theta})$ it is not required in the construction of the Forward Dynamics.*

Definition 3.3.7 (Forward Dynamics) - *the computational steps*

1. *Calculate $\tilde{C}(\theta,\dot{\theta})$*
2. *Calculate the Mass matrix $M(\theta)$*

3. Solve for $\ddot{\theta}$ the equation $M(\theta) \cdot \ddot{\theta} = \tau - \tilde{C}(\theta, \dot{\theta})$

Remark 3.3.9 (Calculation of $\tilde{C}$)

$$\begin{aligned} \tau = \tilde{C}(\theta, \dot{\theta}) \text{ by setting } \ddot{\theta} = 0 \text{ in Eq. 3.167} \\ \tilde{C}(\theta, \dot{\theta}) = RNEA(\theta, \dot{\theta}, \ddot{\theta} = 0, \mathcal{F}_{ext}) \end{aligned}$$

Remark 3.3.10 (Calculation of $M(\theta)$)

$$\begin{aligned} \tau = M(\theta) \cdot \ddot{\theta} \text{ by setting } g = 0, \dot{\theta} = 0, \mathcal{F}_{ext} = 0 \text{ in Eq. 3.167} \\ \implies \tilde{C}(\theta, \dot{\theta}) = 0 \\ \implies M_{:,j}(\theta) = RNEA(\theta, \dot{\theta} = 0, \ddot{\theta} = \ddot{\theta}_j^0, \mathcal{F}_{ext} = 0) \\ where: \end{aligned}$$

$$\ddot{\theta}_j^0 = \begin{bmatrix} \vdots \\ 1 \\ \vdots \end{bmatrix} \text{ where: } j^{th} \text{ element is } 1$$

Remark 3.3.11 *By assuming $(\theta, \dot{\theta}), \tau, M(\theta), \tilde{C}(\theta, \dot{\theta})$ as known, we can immediately compute:*

$$\begin{aligned} \ddot{\theta} &= M^{-1} \cdot (\tau - \tilde{C}) \\ &= FD(\theta, \dot{\theta}, \tau, \mathcal{F}_{ext}) \end{aligned} \tag{3.168}$$

This provides a 2^{nd}-order differential equation, Eq. 3.168, in $\mathbb{R}^N$, computed in an algorithmic way, that can be simulated in the joint trajectory over any time period under a given initial conditions $(\theta, \dot{\theta})$ which can be rewritten as a system of first-order differential equations in $\mathbb{R}^{2N}$ as follows Eq. 3.169.

$$\begin{aligned} X_1 &= \theta \in \mathbb{R}^N \\ X_2 &= \dot{\theta} \in \mathbb{R}^N \\ \dot{X} &= \begin{bmatrix} \dot{X}_1 \\ \dot{X}_2 \end{bmatrix} = \begin{bmatrix} X_2 \\ M^{-1}(X) \cdot (\tau - \tilde{C}(X_1, X_2) \end{bmatrix} \end{aligned} \tag{3.169}$$

Remark 3.3.12 *The computational time of the Forward Dynamics equation is a function of the RNEA algorithm. For $\tilde{C}$ we have $O(N)$ as it required only one evaluation of RNEA. For $M(\theta)$ we have $O(N^2)$ as we have to evaluate the RNEA for each column of the mass matrix.*
The overall time complexity is given by the computation of the inverse of the matrix $M^{-1}(\theta)$: $O(N^3)$.

CHAPTER 4

System identification

4.1 The analytical, algorithmically constructed, equation

Remark 4.1.1 *Given an open chain with N links, we denote J_i as the Jacobian of link / body i, where:*

$$\mathcal{V}_i = J_i \cdot \dot{\theta} = \begin{bmatrix} J_{i,1} & ... & J_{i,N} \end{bmatrix} \cdot \begin{bmatrix} \dot{\theta}_1 \\ \vdots \\ \dot{\theta}_N \end{bmatrix}$$

$$where:$$

$$J_i = \begin{bmatrix} \delta_{i,1} \cdot \mathcal{S}_1 & ... & \delta_{i,N} \cdot \mathcal{S}_N \end{bmatrix}$$

$$\delta_{i,j} = \begin{cases} 1, & \text{if joint } j \text{ supports body } i \\ 0, & \text{otherwise} \end{cases}$$

Example:

Given Fig. 4.1 we have:

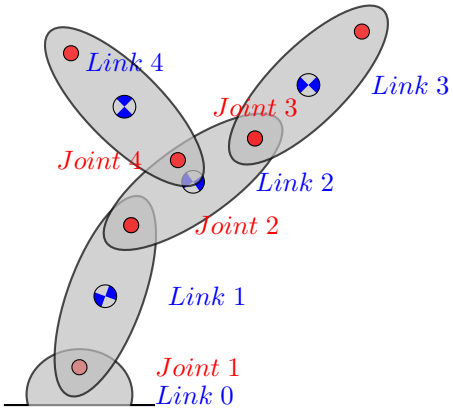


Figure 4.1: Open Chain

$$\mathcal{V}_{i=1} = J_{i=1} \cdot \dot{\theta} = \begin{bmatrix} \delta_{i=1,1} \cdot \mathcal{S}_1 & \delta_{i=1,2} \cdot \mathcal{S}_2 & \delta_{i=1,3} \cdot \mathcal{S}_3 & \delta_{i=1,2} \cdot \mathcal{S}_2 \end{bmatrix} \cdot \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \dot{\theta}_3 \\ \dot{\theta}_4 \end{bmatrix}$$

$$= \begin{bmatrix} \mathcal{S}_1 & 0 & 0 & 0 \end{bmatrix} \cdot \dot{\theta} \qquad \textit{(coordinate-free)}$$

$$\mathcal{V}_{i=2} = J_{i=2} \cdot \dot{\theta} = \begin{bmatrix} \mathcal{S}_1 & \mathcal{S}_2 & 0 & 0 \end{bmatrix} \cdot \dot{\theta}$$

$$\overset{\{2\}\textit{-frame}}{\Longrightarrow} {}^2\mathcal{V}_2 = \underbrace{\begin{bmatrix} {}^2X_1 \cdot {}^1\mathcal{S}_1 & {}^2\mathcal{S}_2 & 0 & 0 \end{bmatrix}}_{{}^2J_2} \cdot \dot{\theta}$$

$$\overset{For\ {}^4\mathcal{V}_4\ {}^4J_4}{\Longrightarrow} = \begin{bmatrix} {}^4X_1 \cdot {}^1\mathcal{S}_1 & {}^4X_2 \cdot {}^2\mathcal{S}_2 & 0 & {}^4\mathcal{S}_4 \end{bmatrix}$$

Remark 4.1.2 *By evaluating the Inverse Dynamics using the RNEA algorithm, one will end up with a closed-form expression that characterizes the entire iterative approach, which can then be factorized. The end result structure of the dynamics provides further properties which can be used for system identification.*

Remark 4.1.3 (Dynamics Equation - Structure) *In the process of building an analytical form of the dynamics, we will first analyze a 2−link, open-loop multibody system, Fig. 4.2. In the end, we will generalize for any number of links.*

After the Forward pass, the result of Backward pass is:

$$\mathcal{F}_2 = (\mathcal{I}_2 \cdot \mathcal{A}_2 + \mathcal{V}_2 \times^* \cdot \mathcal{I}_2 \cdot \mathcal{V}_2) - \mathcal{F}_2^{ext}$$

$$\begin{aligned}\mathcal{F}_1 &= \mathcal{I}_1 \cdot \mathcal{A}_1 + \mathcal{V}_1 \times^* \mathcal{I}_1 \cdot \mathcal{V}_1 + {}^1X_2^* \cdot \mathcal{F}_2 \\ &= \underbrace{\mathcal{I}_1 \cdot \mathcal{A}_1 + \mathcal{V}_1 \times^* \mathcal{I}_1 \cdot \mathcal{V}_1}_{(1)} + {}^2X_1^T \cdot \underbrace{(\mathcal{I}_2 \cdot \mathcal{A}_2 + \mathcal{V}_2 \times^* \cdot \mathcal{I}_2 \cdot \mathcal{V}_2)}_{(2)} - {}^2X_1^T \cdot \underbrace{\mathcal{F}_2^{ext}}_{(3)}\end{aligned}$$

$$\tau_2 = {}^2\mathcal{S}_2^T \cdot {}^2\mathcal{F}_2 = {}^2\mathcal{S}_2^T \cdot ({}^2\mathcal{I}_2 \cdot {}^2\mathcal{A}_2 + {}^2\mathcal{V}_2 \times^* \cdot {}^2\mathcal{I}_2 \cdot {}^2\mathcal{V}_2) - {}^2\mathcal{S}_2^T \cdot {}^2\mathcal{F}_2^{ext}$$

$$\tau_1 = {}^1\mathcal{S}_1^T \cdot {}^1\mathcal{F}_1 = \underbrace{{}^1\mathcal{S}_1^T \cdot \overbrace{(...)}^{(1)}}_{(A)} + \underbrace{({}^2X_1 \cdot {}^1\mathcal{S}_1)^T \cdot \overbrace{(...)}^{(2)}}_{(B)} - \underbrace{({}^2X_1 \cdot {}^1\mathcal{S}_1)^T \cdot \overbrace{(...)}^{(3)}}_{(C)}$$

where :

(A): Torque at joint 1 due to motion of body 1

(B): Torque at joint 1 due to motion of body 2

(C): Torque at joint 1 due to external force, $\mathcal{F}_2^{ext}$ applied to body 2

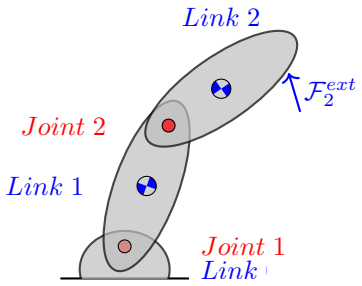


Figure 4.2: Open Chain-2 links

By rewriting the equations in vectorial form one gets:

$$\tau = \begin{bmatrix}\tau_1\\ \tau_2\end{bmatrix} = \begin{bmatrix}\mathcal{S}_1^T \cdot (\mathcal{I}_1 \cdot \mathcal{A}_1 + ...) + ({}^2X_1 \cdot \mathcal{S}_1)^T \cdot (\mathcal{I}_2 \cdot \mathcal{A}_2 + ...) + ({}^2X_1 \cdot \mathcal{S}_1)^T \cdot (-\mathcal{F}_2^{ext})\\ 0 \cdot (\mathcal{I}_1 \cdot \mathcal{A}_1 + ...) + (\quad\quad \mathcal{S}_1)^T \cdot (\mathcal{I}_2 \cdot \mathcal{A}_2 + ...) + (\quad\quad \mathcal{S}_1)^T \cdot (-\mathcal{F}_2^{ext})\end{bmatrix} \tag{4.1}$$

$$= \underbrace{\begin{bmatrix}\mathcal{S}_1^T\\ 0\end{bmatrix}}_{{}^1J_1^T} \cdot (\mathcal{I}_1 \cdot \mathcal{A}_1 + ...) + \underbrace{\begin{bmatrix}({}^2X_1 \cdot \mathcal{S}_1)^T\\ \mathcal{S}_2^T\end{bmatrix}}_{{}^2J_2^T} \cdot (\mathcal{I}_2 \cdot \mathcal{A}_2 + ...) + \underbrace{\begin{bmatrix}({}^2X_1 \cdot \mathcal{S}_1)^T\\ \mathcal{S}_2^T\end{bmatrix}}_{{}^2J_2^2} \cdot (-\mathcal{F}_2^{ext}) \tag{4.2}$$

Where:

$$\underbrace{\begin{bmatrix}\mathcal{S}_1 & 0\end{bmatrix}}_{{}^1J_1} \qquad\qquad \underbrace{\begin{bmatrix}{}^2X_1 \cdot \mathcal{S}_1 & \mathcal{S}_2\end{bmatrix}}_{{}^2J_2} \tag{4.3}$$

Eq. 4.2: Each line i represents the sum of all the projections of the spatial forces over joint i

Eq. 4.2: Each column i represents body's i spatial force projection over all joints

Remark 4.1.4 *From: Eq. 4.2, Eq. 4.3 we can conclude that the torque vector τ is the product between the Jacobian transpose and the force (induced by each body or by external forces, e.g. gravity) and it can be interpreted as the motion induced by the force.*

$$\tau = J^T \cdot \mathcal{F}$$

Remark 4.1.5 (Generalization) *Given N links the general vectorial form is:*

$$\tau = \sum_{i=1}^{N} \Big\{ J_i^T \cdot \underbrace{(\mathcal{I}_i \cdot \mathcal{A}_i + \mathcal{V}_i \times^* \mathcal{I}_i \cdot \mathcal{V}_i)}_{\substack{\textit{The (internal) force}\\ \textit{induced by the body } i}} + J_i^T \cdot (-\mathcal{F}^{ext}) \Big\} \tag{4.4}$$

Remark 4.1.6 *By applying the following change of variables to Eq. 4.4 (computation done for internal forces):*

$$\mathcal{V} = J_i \cdot \dot{\theta} \tag{4.5}$$

$$\mathcal{A}_i = \dot{\mathcal{V}}_i = J_i \cdot \ddot{\theta} + \dot{J}_i \cdot \dot{\theta} + \mathcal{V}_i \times J_i \cdot \dot{\theta} \tag{4.6}$$

- *Where J_i represents the Jacobian of the body i as introduced in Remark: 4.1.1*
- *We rewrite the torque, τ, as a function of θ, $\dot{\theta}$ and $\ddot{\theta}$*

$$\implies \tau = \sum_{i=1}^{N} J_i^T \cdot \mathcal{I}_i \cdot J_i \cdot \ddot{\theta} + J_i^T \cdot \mathcal{I}_i \cdot \dot{J}_i \cdot \dot{\theta} + J_i^T \cdot \mathcal{I}_i \cdot \mathcal{V}_i \times J_i \cdot \dot{\theta} + J_i^T \cdot \mathcal{V}_i \times^* \mathcal{I}_i \cdot \overbrace{\mathcal{V}_i}^{J_i \cdot \dot{\theta}}$$

$$= \underbrace{\left(\sum_{i=1}^{N} J_i^T \cdot \mathcal{I}_i \cdot J_i \right)}_{M(\theta)} \cdot \ddot{\theta} + \sum_{i=1}^{N} \underbrace{J_i^T \cdot (\mathcal{I}_i \cdot \dot{J}_i + \mathcal{I}_i \cdot \mathcal{V}_i \times J_i + \mathcal{V}_i \times^* \mathcal{I}_i \cdot J_i)}_{C(\theta, \dot{\theta})} \cdot \dot{\theta} \tag{4.7}$$

$$= ID(\theta, \dot{\theta}, \ddot{\theta}, \mathcal{F}_{ext}, Model)$$

Remark 4.1.7 *By using a **reduced representation** of the degrees of freedom of each rigid body, constructed using the Lie Algebras, in a Newton-Euler Algorithm for Inverse Dynamics, one ends up with an equation,Eq. 4.7, that resembles the energy-based Euler-Lagrange **reduced representation equation**.*

Remark 4.1.8 *Eq. 4.7 represents the 3^{rd} way of building the DAE/ODE for multibody dynamics, other than the Newton-Euler and Lagrange/Hamiltonian way.*

Remark 4.1.9 *For adding external forces in Eq. 4.7, one must add the term $J^T(\theta) \cdot \mathcal{F}^{ext}$. The gravity term, as an example of external forces, can be factorized and added as well:*

$$\sum_{i=1}^{N} J_i^T \cdot {}^i\mathcal{I}_i \cdot {}^iX_0 \cdot (-{}^0\mathcal{A}_g) \tag{4.8}$$

4.2 Parametric estimation: inertia matrix, mass, the center of mass

Remark 4.2.1 *By knowing $\tau, J_i, \theta, \dot{\theta}, \ddot{\theta}, \dot{J}_i, \mathcal{V}_i\times, \mathcal{V}_i\times^*$ we observe that the Eq. 4.7 is **linear** in spatial inertia, $\mathcal{I}_i$. This introduces the options for a **Linear Least Square approach** for parametric estimation for the **inertia** and **mass** for each of the rigid bodies as the spatial inertia, $\mathcal{I}_i$ is a function of both of them.*

Remark 4.2.2 *Each Spatial Inertial of each body is represented relative to the center of the mass, Eq. 4.9. Since Eq. 4.7 is just another computational process of Inverse Dynamics, the Spatial Inertia of each body will end up being represented in joint frames as the equation it's written as a function of* $(\theta, \dot{\theta}, \ddot{\theta})$*, hence, the* ${}^{i}\mathcal{I}_i$*, after translation, will look like Eq. 4.10.*

$$ {}^{C}\mathcal{I}_i = \begin{bmatrix} {}^{C}\bar{I}_i & 0 \\ 0 & m \cdot I_3 \end{bmatrix} \tag{4.9} $$

$$ {}^{i}\mathcal{I}_i = \begin{bmatrix} {}^{C}\bar{I}_i + m \cdot [{}^{i}p_{C_i}] \cdot [{}^{i}p_{C_i}]^T & m \cdot [{}^{i}p_{C_i}] \\ m \cdot [{}^{i}p_{C_i}]^T & m \cdot I_3 \end{bmatrix} \tag{4.10} $$

Remark 4.2.3 *Since the Eq. 4.10 of all the bodies, represents Eq. 4.10 translated, it also contains, the parameters,* ${}^{i}p_{C_i}$*, that characterize the center of the mass expressed in a joint frame i. Hence, Eq. 4.7, expressed in joint space, it's linear in the* ***inertia matrix, mass, and center of the mass*** *of all the bodies.*

Definition 4.2.1 (Sensor Input) *The measurements from sensors are:* $\tau, \theta, \dot{\theta}, \ddot{\theta}$*.*

Remark 4.2.4 (Standard form of the problem) *By computing* $J_i, \dot{J}_i, \mathcal{V}_i\times, \mathcal{V}_i\times^*$*, one can either choose to use directly the Eq. 4.7 or to put the problem in a general form Eq.4.11 and to solve for* π *Eq. 4.12.*

$$ A \cdot \pi + B = \tau \tag{4.11} $$

$$ \pi_i = [m, h_x, h_y, h_z, I_{xx}, I_{xy}, I_{xz}, I_{yy}, I_{yz}, I_{zz}]^T \in \mathbb{R}^{10} \tag{4.12} $$
$$ \textit{where: } h = [h_x, h_y, h_z]^T = m \cdot CoM $$

Remark 4.2.5 (Linear Least Square formulation) *Since we always have an initial guess, either from the factory (using the CAD model) or from a previous prediction, one can choose to leverage this prior knowledge of the parameters by modeling the solution as an optimization problem Eq.4.13. The result is an algorithm that will continuously improve the estimation of the parameters after each measurement.*

$$ \min_{\pi} \sum_{i=1}^{measurements} ||A^i \cdot \pi + B^i - \tau^i||^2 \tag{4.13} $$

Remark 4.2.6 (A better approach) *The approach introduced in Remark: 4.2.4 has 2 disadvantages:*

- *one must recompute A and B every time when the structure of the equation changes*
- *one must compute $\dot{J}_i$*

Instead of coming up with the previous closed form for the linear least square, one can observe that Eq. 4.7 is equivalent to the Inverse Dynamics which can be computed iteratively using a modified RNEA algorithm where the model of the robot (that contains all the parameters that need to be estimated) is also part of the input parameters Eq. 4.14.

For this, we leverage CasADi's functionality of generating a mathematical expression that is equivalent to running the algorithm itself.

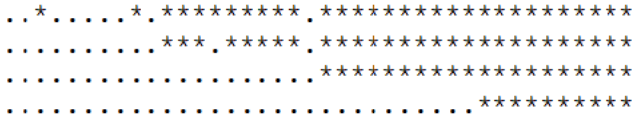
$$\tau = RNEA(\theta = Value_1, \dot{\theta} = Value_2, \ddot{\theta} = Value_3, \mathcal{F}_{ext}, Model(\pi = Symbolic)) \tag{4.14}$$

By evaluating the algorithm with constant values for $\theta, \dot{\theta}, \ddot{\theta}, \tau, \mathcal{F}^{ext}$ for each instance of sensors measurements, as well as with the model as a function of symbolic parameters representation of π, one gets, as output, an expression linear in π. This can also be seen as a change of variable on the expression level, where we change all the numerical values for the parameters we want to identify with variables. As the last step, the extraction of A and B is done also, automatically using CasADi's functionality of extracting the linear coefficients from expressions Eq. 4.15.

$$\min_{\pi} \sum_{i=1}^{measurements} ||RNEA(\theta_i, \dot{\theta}_i, \ddot{\theta}_i, \mathcal{F}_i^{ext}) - \tau^i||^2 \tag{4.15}$$

Remark 4.2.7 *By checking the structural sparsity of matrix A, Fig. 4.3 - corresponding to Fig. 4.4, one can observe that, independently of the sensor readings, the parameters for the first 2 links are not identifiable.*

Figure 4.3: 3 DOF robot - structural sparsity - τ



By changing the torque sensor to a sensor that can measure spatial forces (e.g. forces and torques) the only unidentifiable link is the first one Fig. 4.5. Also, we get 6 times more rows.

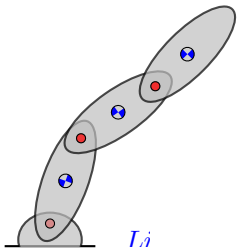


Figure 4.4: 3 DOF robot

Remark 4.2.8 *For a better understanding of why the first link is not identifiable and all the other are one must take a close look at what is happening at the level of each joint during data acquisition and system identification.*

$$\underbrace{\begin{bmatrix}\mathcal{S}_1^T\\0\end{bmatrix}\cdot(\mathcal{I}_1\cdot\mathcal{A}_1+\ldots)}_{(1)}+\underbrace{\begin{bmatrix}({}^2X_1\cdot\mathcal{S}_1)^T\\\mathcal{S}_2^T\end{bmatrix}\cdot(\mathcal{I}_2\cdot\mathcal{A}_2+\ldots)}_{(2)}+\ldots$$

In (2) *one can observe that each column of* $\mathcal{I}_2$ *gets projected on the joint* $\mathcal{S}_1$ *for each data acquisition. As the system changes with time, the configuration of the system changes as well. Since the adjoint matrix,* ${}^2X_1$ *is a function of the configuration it also changes as well over time. This results in projecting each column of the spatial inertia,* $\mathcal{I}_2$*, on the different representation of joint* $\mathcal{S}_1$*. This way, during different measurements, different parameters get to be excited once in a while and therefore all of them become identifiable.*

This is not the case for the first joint (1)*, which is fixed as it doesn't have a parent and therefore it doesn't have an adjoint matrix* X *as well.*

Remark 4.2.9 (Parametric estimation of Link 1)
There are 2 ways possible to identify the parameters for link 1:

- *Forcing Joint 1 to be an Universal Joint, Fig. 4.6, e.g. it has 2 DOF. This is equivalent to adding another joint as a parent for joint* 1 *where the distance between them is* 0*.*
- *After identifying parameters for the last* $n-1$ *links, apply Nonlinear Least Squares parametric estimation using multiple shooting to determine the parameters for* ***Link 1****.*

Remark 4.2.10 *The Nonlinear Least Squares (using multiple-shooting), even though, is a more general approach is significantly more computationally intensive than the Linear Least Squares approach for an ODE of a multi-body system. Moreover, it's less precise.*

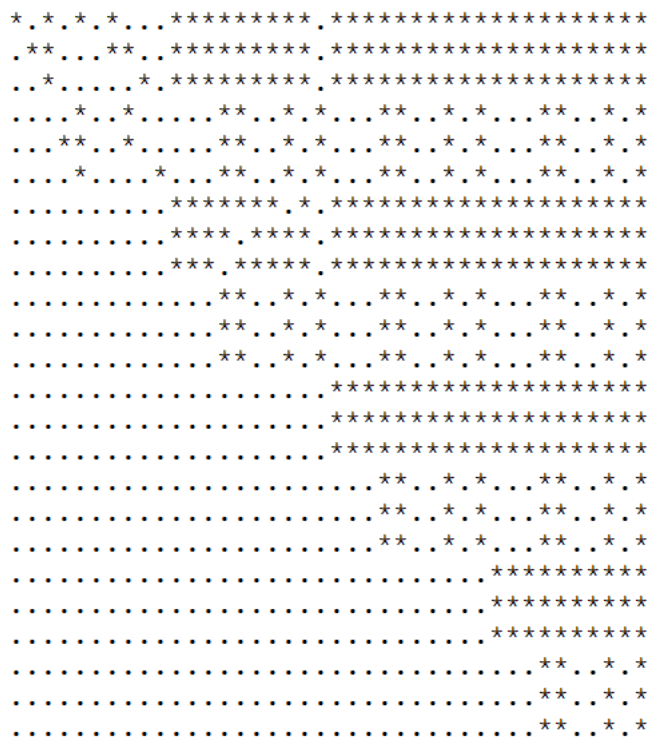


Figure 4.5: 3 DOF robot - structural sparsity - $\mathcal{F}$

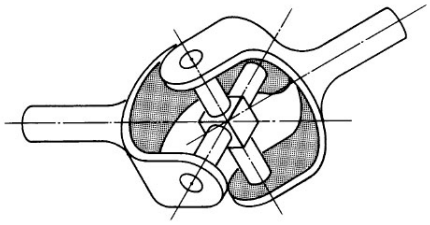


Figure 4.6: Universal Joint [23]

CHAPTER 5

Experimental setup

For the validation of the proposed solution, particular decisions, with respect to the testing process, were taken. These have an impact on the following:

- environment setup
- sensitivity analysis computation
- robot definition
- model predictive control and objective function definition

The open-source implementation of the project is available at [10].

5.1 Environment definition

The process is done in a contained, simulated environment. This is necessary due to the nature of the problem, as we want to be able to test the solution in full generality. At the same time, we want to contain the engineering complexity required by not having to work with a real robot. This way we get to ignore problems such as time latency in data acquisition, clock synchronization between multiple sensors, and changes in the properties of sensors and control systems due to external factors over a long period of time, etc. As a rule of thumb, all of these problems required supplementary attention and involve embedded programming (system identification and control applied on a mechatronic level in a hierarchical way) using a real-time operating system. The result is an intermediate interface that abstractisezes the low-level complexity of each of the sub-components of the robot and tries to contain the stochasticity of each of them. These subtasks can easily grow to become subjects on their own, hence, the simulation environment was more suitable.

5.2 Sensitivity analysis

The development framework selected provides mathematical modeling tools as well as optimization tools and offers easy access to sensitivity analyses. In this regard, *CasADi* was chosen, an open-source software tool for **numerical optimization** in general and **optimal control** (i.e. optimization involving differential equations) in particular. The main scope of **CasADi** is **automatic differentiation**. Besides that, it also offers support for external **ODE/DAE integrators**, **sensitivity analysis**, as well as nonlinear optimization tools. All these functionalities are accessible using a height-level API, which is implemented using a functional design. This symbolical framework makes the modeling problems more accessible by letting the developer focus more on the problem and less on the low-level code functionalities.

5.3 Robot definition

The testing process is applicable to any open-loop robot. This is achieved by building the robot's ODE, previously introduced in chapter 2, in a generic way starting from the corresponding URDF multi-body file which contains all the necessary information about each rigid body as well as the tree structure.

5.3.1 URDF file structure

This **URDF** file format is an XML-type file format that defines the geometry of a robot in a tree structure. Since it can describe only rigid body robots the file is a description for *links* and *joints* Code 5.1.

```
<robot name="name">
  <link>  </link>
  <link>  </link>
  <link>  </link>

  <joint>  </joint>
  <joint>  </joint>
</robot>
```

Listing 5.1: URDF example

A link, Fig. 5.1, represents the properties of a rigid body, like inertia, collision properties as well as visual features Code 5.2.

The visual tag, as its name suggests is used for visual purposes and specifies the shape of the object (e.g.:mesh, box, cylinder, sphere). The reference frame used by the visual element is relative to the reference frame of the link and it is maintained by *origin*.

Most of the time, the collision geometry tag is the same as visual geometry, or its geometry is a little bit bigger. The frame of reference of the collision element is maintained into *origin* element and it is relative to the reference frame of the link.

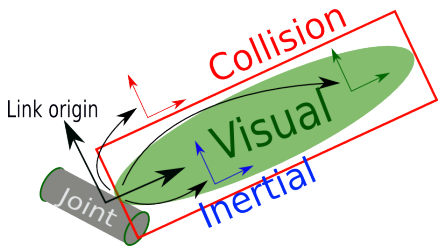


Figure 5.1: Link [21]

The inertial tag is used for building the physical model (the robot's ODE). The reference frame's origin is maintained in the field *origin* and is the pose (e.g. position and orientation) of the inertial reference frame, expressed relative to the link reference frame. The *origin* is always in the center of gravity of the rigid body but the axes of the inertial reference frame can have arbitrary orientation.

```
<link name="link_name">

  <visual>
      <origin xyz="0 0 2" rpy="1.2 0 0" />
      <geometry>
        <cylinder length="0.2" radius="0.1"/>
      </geometry>
  </visual>

  <collision>
      <origin xyz="0 0 2" rpy="1.2 0 0" />
      <geometry>
          <cylinder length="0.22" radius="0.11"/>
      </geometry>
  </collision>

  <inertial>
      <origin xyz="0 0 2" rpy="1.2 0 0" />
      <mass value="5"/>
      <inertia
          ixx="1.0" ixy="0.0" ixz="0.0"
          iyy="1.0" iyz="0.0"
          izz="1.0"/>
  </inertial>
</link>
```

Listing 5.2: link example

Joints, Fig. 5.2, are used to describe the relative motion between two consecutive links. A joint can be of the following types: prismatic, revolute, continuous, planar, fixed and floating. Code 5.3

The *origin* tag represents the transformation from the parent link to the child link. The location of the joint is at the origin of the child link. Hence, the *origin* represents the *pose* of the child frame *w.r.t.* the parent frame.

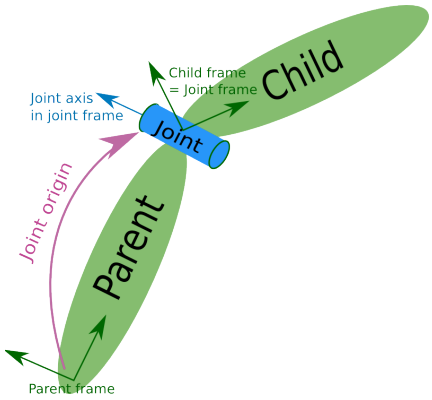


Figure 5.2: Joint [22]

The *axis* tag is defined in the joint frame and it can represent one of the following: the axis of translation for prismatic joints, the axis of rotation for revolute joints, or the surface normal for planar joints. The floating and the Fixed joint does not require a specific axis, hence they don't use the field.

Other properties, a joint can have, are represented by the degree of freedom limits as well as dynamical limits. Limits are in meters for prismatic joints, are omitted for continuous and fixed joints, and are in radians for revolute joints.

```
<joint name="name" type="revolute">
  <parent link="link_a"/>
  <child link="link_b"/>

  <origin xyz="0 0 0" rpy="0 0 0"/>
  <axis xyz="1 0 0"/>

</joint>
```

Listing 5.3: joint example

As an example, this is how a robot in an *URDF* XML format, formed by 2 links connected by a joint looks like Code 5.4.

```
<?xml version="1.0"?>
<robot name="simple_robot">
  <link name="link_A">
  </link>
  <link name="link_B">
  </link>

  <joint name="link_A_to_link_B" type="fixed">
    <parent link="link_A"/>
    <child link="link_B"/>
    <origin xyz="1 -0.124 0.42"/>
  </joint>

</robot>
```

Listing 5.4: robot example

5.3.2 ODE parameter extraction

An extraction of the parameters for a rigid body chain from a **URDF** file is done as explained by Alg. 3 where the following operations were used:

1. $getChain(RobotInfo, Root, EndEffector)$ - returns a chain of links and joins that exist between 2 joints.
2. $StartFromFirstNonFixedJoint(chain)$ - clear all the links and joints *s.t.* the *chain* to start with a non fixed joint.
3. $Joints(RobotInfo)$ - returns a list of joints.
4. $type(joint)$ - returns the type of a joint (*e.g.* revolute, fixed, etc)
5. $Frame(origin\_rpy(joint), origin\_xyz(joint))$ - return a $4X4$ homogeneous transformation for an roll pitch yaw Euler orientation and x, y, z displacement.
6. $se\_TO\_SE(Twist(joint)$ - return the exponential map of $se(3)$ as a $4X4$ homogeneous transformation.
7. ${}^{A}X_{B}\_Build((frame \cdot transform)^{-1})$ - build the *Adjoint* transformation matrix as a $6X6$ matrix.
8. $No\_fixed\_joint(Current\_{}^{i}X_{p(i)})$ - checks whether the expression contains any symbols. In case no symbols are found, the joint has no degree of freedom.
9. ${}^{i}S_i \leftarrow {}^{i}S_i + [Twist(joint)]$ - append a new *twist* object to *list* ${}^{i}S_i$
10. $Links(RobotInfo)$ - returns a list of links.
11. $Inertia\_Build\_From(link)$ - it builds a $6X6$ spatial inertia matrix.
12. $Is\_eye(Prev\_{}^{i}x_{p(i)}\_path)$ - it checks for identity matrix.

This process can be easily adapted for a URDF tree structure. The result of applying this algorithm is the following 4 lists:

1. ${}^{i}X_{p(i)}$: a list of $transformations$ between 2 consecutive *screws* (*e.g.* from the parent of screw i to screw i)
2. ${}^{i}S_i$: a list of screws where each screw is expressed in the local frame of reference (*e.g.* screw i in frame i)
3. ${}^{i}I_i$: a list of the spatial inertia of bodies expressed in local frame (*e.g.* spatial inertia of body i in frame i)
4. ${}^{i}X_0$: a list of the transformations from the global to local frame of each screw

Once these transformations are computed, they are used as input for **Inverse Dynamics** as well as **Forward Dynamics**.

Algorithm 3 *Model Parameter* extraction from *URDF* file

Require: $Root, EndEffector, q, RobotInfo$
Ensure: ${}^{i}X_{p_{(i)}}, {}^{i}S_i, {}^{i}I_i, {}^{i}X_0$

1: ${}^{i}X_{p_{(i)}} \leftarrow [], {}^{i}S_i \leftarrow [], {}^{i}I_i \leftarrow [], {}^{i}X_0 \leftarrow [I_6]$
2: $chain \leftarrow getChain(RobotInfo, Root, EndEffector)$
3: $chain \leftarrow StartFromFirstNonFixedJoint(chain)$
4: $Current\_{}^{i}X_{p_{(i)}} \leftarrow I_6$
5: $Prev\_{}^{i}X_{p_{(i)}}\_path \leftarrow I_6$
6: $q_{index} \leftarrow -1$
7: **while** $element \in chain$ **do**
8: **if** $element \in Joints(RobotInfo)$ **then**
9: $joint \leftarrow Joints(RobotInfo)[element]$
10: **if** $type(joint)$ is 'fixed' **then**
11: $q_{current} \leftarrow 0$
12: **else**
13: $q_{index} \leftarrow q_{index} + 1$
14: $q_{current} \leftarrow q[q_{index}]$
15: **end if**
16: $frame \leftarrow Frame(origin\_rpy(joint), origin\_xyz(joint))$
17: $transform \leftarrow se\_TO\_SE(Twist(joint), q_{current})$
18: $Current\_{}^{i}X_{p(i)} \leftarrow {}^{A}X_B\_Build((frame \cdot transform)^{-1})$
19: **if** $No\_fixed\_joint(Current\_{}^{i}X_{p(i)})$ **then**
20: ${}^{i}S_i \leftarrow {}^{i}S_i + [Twist(joint)]$
21: ${}^{i}X_{p(i)} \leftarrow {}^{i}X_{p(i)} + [Current\_{}^{i}X_{p(i)} \cdot Prev\_{}^{i}x_{p(i)}\_path]$
22: ${}^{i}X_0 \leftarrow {}^{i}X_0 + [Current\_{}^{i}X_{p(i)} \cdot Prev\_{}^{i}x_{p(i)}\_path \cdot {}^{i}X_0[last]]$
23: $Prev\_{}^{i}x_{p(i)}\_path \leftarrow I_6$
24: **else**
25: $Prev\_{}^{i}x_{p(i)}\_path \leftarrow Current\_{}^{i}X_{p(i)} \cdot Prev\_{}^{i}x_{p(i)}\_path$
26: **end if**
27: **end if**
28: **if** $element \in Links(RobotInfo)$ **then**
29: $link \leftarrow Links(RobotInfo)[element]$
30: $Current\_Link\_Inertia \leftarrow Inertia\_Build\_From(link)$
31: $frame \leftarrow Frame(Extract\_origin\_From\_inertial(link))$
32: ${}^{CoM}X_A \leftarrow {}^{A}X_B\_Build(frame^{-1})$
33: ${}^{CoM}X_A \leftarrow {}^{CoM}X_A \cdot Prev\_{}^{i}x_{p(i)}\_path$
34: ${}^{A}X^{*}_{CoM} = ({}^{CoM}X_A)^T$
35: $Current\_Link\_Inertia \leftarrow {}^{A}X^{*}_{CoM} \cdot Current\_Link\_Inertia \cdot {}^{CoM}X_A$
36: **if** $Is\_eye(Prev\_{}^{i}x_{p(i)}\_path)$ **then**
37: ${}^{i}I_i \leftarrow {}^{i}I_i + [Current\_Link\_Inertia]$
38: **else**
39: ${}^{i}I_i[last] \leftarrow {}^{i}I_i[last] + Current\_Link\_Inertia$
40: **end if**
41: **end if**
42: **end while**
43: ${}^{i}X_0 \leftarrow {}^{i}X_0[1 : last]$

5.4 Model Predictive Control

The model predictive control accepts generic ODEs (*e.g.* different robots) as well as generic objective functions. For this, a custom implementation of multiple shooting was used.

5.4.1 MPC flow

The overall flow of the MPC can be summarized by the following algorithm Alg. 4. The process can be interpreted as follows: Given a robot in an initial state, the objective function, the ground truth parameters, and the initial guess of the robot parameters, the algorithm tries to find, at each iteration, the next control action, that puts the robot closer to the objective by computing an optimal control problem. A second process, that re-estimates the parameters based on the new measurements, is taken place in parallel. These 2 complement each other. At the same time, changes in the model are simulated by adding noise.

At each iteration, readings for $q, \dot{q}, \ddot{q}$ are used. Given the nature of the implementation, the sensors are simulated as well. The measurements for $(q, \dot{q})$ can be directly taken from the robot simulation. On the other hand, acceleration and spatial force require more attention. The sensor that returns $\ddot{q}$ is simulated using the Forward Dynamics algorithm applied in the context of the robot (using the ground truth parameters that characterize the ODE of the robot and the current state of the robot). The spatial force is simulated using the Recursive Newton-Euler Algorithm applied in the context of the robot. Nowadays, one can find inertial sensors with different properties. During this thesis, we have considered idealized sensors (no error in measurements) which is far from what we get in a real-case scenario. For a better perspective of the current state-of-the-art, work in force/torque sensors, one can check the following article [9].

During testing, no error in sensor measurements was taken into consideration.

In addition, by adding extra constraints, at the optimal control level, one can overcome the limitation of the $URDF$ format (being defined only for open-loop robots). This way, the current implementation can be used for closed-loop robots as well.

5.4.2 Optimal Control

Every time when $ExtractNextControlAction(...)$ doesn't return an object an optimal control sub-task starts. The formulation of the OCP, using multiple shooting, looks as follows:

Algorithm 4 System Identification during Nonlinear Model Predictive Control

Require: $CurrentState, Cfg, GroundTruthParam, CurrentPredictedParam$
Ensure: $None$

```
1:
2: while (True) do
3:
4:     if TimeExpiredForCurrentParameters then
5:         UPDATE(GroundTruthParam)                    ▷ Add random noise
6:         Intg ← FwdDynamicsIntegrator(GroundTruthParam)
7:         RESET(TimeExpiredForCurrentParameters)
8:     end if
9:
10:    q ← UPDATE(CurrentState)                         ▷ "Sensor" reading
11:    q̇ ← UPDATE(CurrentState)                         ▷ "Sensor" reading
12:
13:    while True do
14:        τ ← ExtractNextControlAction(OCP)            ▷ Next control action
15:        if τ then
16:            break
17:        else
18:            Cfg[S_0] ← (q, q̇)      ▷ Update the Configuration with new state
19:            OCP ← UPDATE(Cfg, CurrentPredictedParam)
20:                                   ▷ Compute Optimal Control Problem
21:        end if
22:    end while
23:
24:    q̈ ← FwdDynamics(q, q̇, τ, GroundTruthParam)      ▷ "Sensor" reading
25:
26:    F ← RNEA(q, q̇, q̈, GroundTruthParam, f_ext = 0)  ▷ "Sensor" reading
27:
28:    CurrentPredictedParam ← LLS_SysIden_NoSensorError(q, q̇, q̈, F, CurrentPredictedParam)
29:
30:    CurrentState ← Intg(Current_State, τ, dt)  ▷ Update the "real" robot
31: end while
```

$$\begin{aligned} \min_{x(\cdot),u(\cdot)} \quad & \Phi(x(t_f)) \\ \text{s.t.} \quad & \dot{x}(t) = f(x(t), u(t), p), \\ & x(t_0) = x_0 \\ & x^{lo} \leq x(t) \leq x^{up}, \\ & u^{lo} \leq u(t) \leq u^{up} \qquad \forall t \in [t_0, t_f] \end{aligned} \tag{5.1}$$

To solve Eq. 5.1 numerically, we are using a discretized version by introducing the following multiple shooting variables: $s_0, \cdots, s_N$; $q_0, \cdots, q_N$ for Eq. 5.2.

$$\begin{aligned} \min_{x(\cdot),u(\cdot)} \quad & \Phi(S_N) \\ \text{s.t.} \quad & s_{i+1} = x(t_{i+1}; t_i, s_i, q_i, p) \quad i = 0, ..., N-1 \\ & s_0 = x_0 \\ & x^{lo} \leq s_i \leq x^{up}, \qquad\qquad i = 0, ..., N \\ & u^{lo} \leq q_i \leq u^{up} \qquad\qquad i = 0, ..., N \end{aligned} \tag{5.2}$$

where $x(t; t_0, s, q, p)$ is the solution of initial value problem Eq. 5.3.

$$\begin{cases} \dot{x}(t) & = f(x(t), q, p) \\ x(t_0) & = s \end{cases} \tag{5.3}$$

Next, we define the primal variables as $w = (s, q)$ and we introduce the following functions for equality and inequality constraints:

$$a(w) = \begin{bmatrix} x_0 - s_0 \\ x(t_1; t_0, s_0, q_0, p) - s_1 \\ \vdots \\ x(t_N; t_{N-1}, s_{N-1}, q_{N-1}, p) - s_N \end{bmatrix} \tag{5.4}$$

$$b(w) = \begin{bmatrix} x^{lo} - s \\ s - x^{up} \\ q^{lo} - q \\ q - q^{up} \end{bmatrix} \tag{5.5}$$

Based on Eq. 5.4 and Eq. 5.5 one can write the OCP in a more compact form:

$$\begin{aligned} \min_{w} \quad & \Phi(w) \\ \text{s.t.} \quad & a(w) = 0 \\ & b(w) \leq 0 \end{aligned}$$

The end formulation is then used as input for a nonlinear optimizer.

5.4.3 Configuration

```
# timestep
dt = 1/100
# gravitational acceleration
g = [0, 0, -9.8]

Cfg = {

  # initial guess for the final time
    'tf': dt,
    # lower bound for the final time
  'tf_lb': dt,
    # upper bound for the final time
  'tf_ub': 3*dt,
    #Number of shooting nodes, between 2 and 10
  'NumberShootingNodes': 2,
    # initial state
  'S0': vertcat(q, q_dot),
    # a mask that characterizes which components of q and q_dot are
     taken into consideration for the initial state
  'S0_mask': DM.ones(vertcat(q, q_dot).numel()).elements(),
    'SnName': ["Sn_"+str(idx) for idx in range(vertcat(q, q_dot).
    numel())],
  # the final state
    'Xf': vertcat(q, q_dot),
  # a mask that characterizes which components of q and q_dot are
    taken into consideration for the final state
    'Xf_mask': vertcat(DM.ones(q.numel()), DM.ones(q_dot.numel())),
  # initial guess for control parameters
    'w': tau.elements(),
    'wName': ["tau_"+str(idx) for idx in range(tau.numel())],
  # lower bound of the control parameters
    'lbw': [-100 for idx in range(tau.numel())],
  # upper bound of the control parameters
    'ubw': [100 for idx in range(tau.numel())],
  # extra trajectory constraints w.r.t. "q"
    'q': [],
    'qName': [],
    'lbq': [],
    'ubq': [],
  # number of iterations done by the nonlinear optimizer
    'max_iter': 10
}
```

Listing 5.5: Configuration

CHAPTER 6

Tests

6.1 ODE comparation

The proposed solutions are tested relative to the current *state-of-the-art* similar implementation: **u2c** [24], **RBDL** [18], and **Bullet** [12]. These libraries have different goals, hence, not all the functionalities can be found available. The tests are done using Python bindings offered by each of them.

The most important difference is given by the symbolic front-end of **CasADi** [1] which introduces some extra overhead. On the other hand, it also provides free sensitivity analysis, making it more suitable for optimization. This computation can only be archived by using the finite difference method for Bullet and **RBDL**.

6.1.1 Dynamics functionalities

Of the 3 compared libraries, **u2c** is the closest one being also implemented with **CasADi**. The main difference between the two of them is given by the implementation of inverse dynamics functionality which is more suitable for system identification as it has input the spatial inertia as a symbolic parameter. The current proposed solution uses the **ID** as the main bone of the library. This makes it slower given that the computation of inertia requires more time. This decision is expected to impact the performance of the forward dynamics as we will see.

Of the chosen libraries, **Bullet** is the only one based on a redundant approach, hence, no forward dynamics is offered.

RBDL is also a well-established robotics library that also makes use of Featherstone algebra implementation.

An overview of the functionalities offered by the 4 implementations can be seen in table 6.1

Functionality	u2c	RBDL	Bullet	mine
$I.D.$	Yes	Yes	Yes	Yes
$F.D.(ABA)$	Yes	Yes	No	No
$F.D.(CRBA)$	Yes	Yes	Yes	Yes
M	Yes	Yes	Yes	Yes
$\tilde{C}$	Yes	Yes	Yes	Yes
$URDF$	Yes	Yes	No	Yes
$Sensitivity$	Yes	No	No	Yes
$Closed\ loop$	No	Yes	Yes	No
$Open\ loop$	Yes	Yes	Yes	Yes
$Constrains$	No	Yes	Yes	No

Table 6.1: Library functionality provided

6.1.2 Time computation differences

Since in this thesis, we are doing parametric estimation as well as optimal control, the time evaluation of the dynamics expressed in **CasADi** has a big impact on the end results.

For this, 2 types of tests are done. First, we check the time performance between the libraries for 3 robots: **double pendulum** (2 DOF), **Franka Emika Panda** (7 DOF), and **open chain** (14 DOF). A second test is done to observe the implication of increasing the number of degrees of freedom.

The tests are done under Linux, using a 2.4GHz Intel I7 CPU in Python 3.10. No C++ code generation was used for **CasADi**. Also, no vectorization was used as part of the dynamics evaluation. To avoid possible external influences on the results, all the experiments were done 100 times each.

Impact of Number of Input Variables

From Fig. 6.1, Fig. 6.2, and Fig. 6.3 it can be observed that the proposed implementation is slower. This is to be expected due to the symbolical expression used in maintaining the dynamics. The difference relative to **u2c** is due to the algorithm used. (**ABA** in **u2c** vs. **CRBA** for the proposed implementation). It can also be observed that the computation of $\tilde{C} = C + G$ (Coriolis and Gravity forces) is lower than the computation of inverse dynamics. This is happening because, even though, the computation of $\tilde{C}$ is done using the same Recursive Newton-Euler Algorithm, the evaluation of this algorithm for $\tilde{C}$ has fewer input symbolical parameters, thus, the computation graph is smaller.

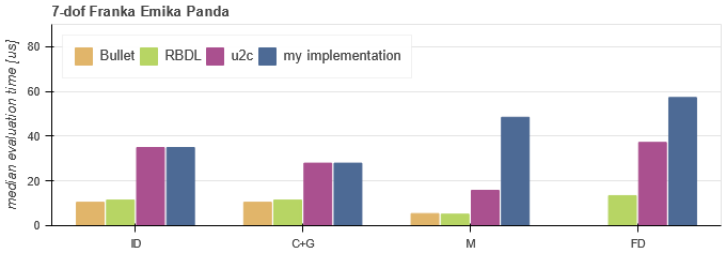


Figure 6.1: Franka Emika Panda

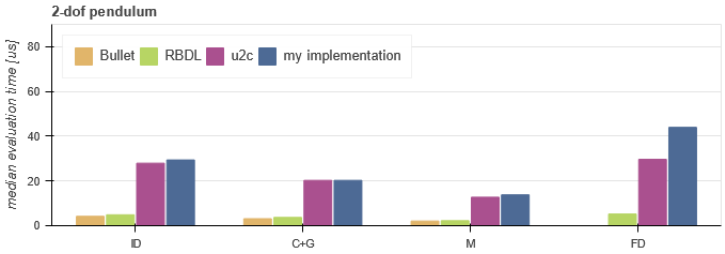


Figure 6.2: Double pendulum

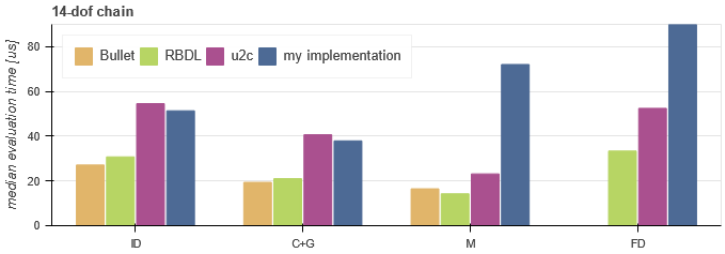


Figure 6.3: Chain

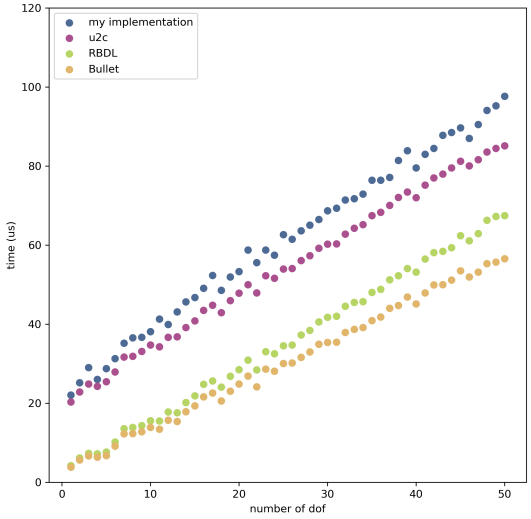


Figure 6.4: $\tilde{C}$ evaluation

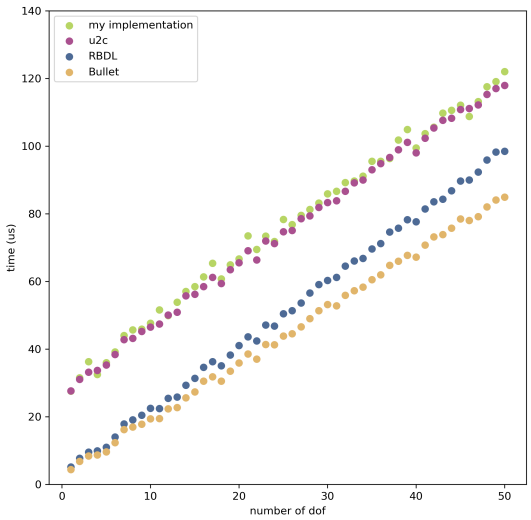


Figure 6.5: ID evaluation

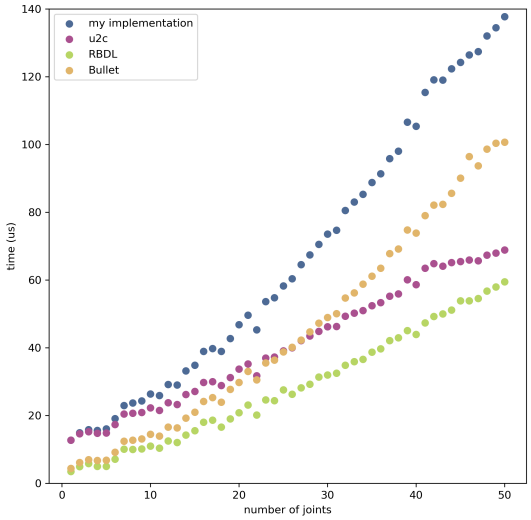


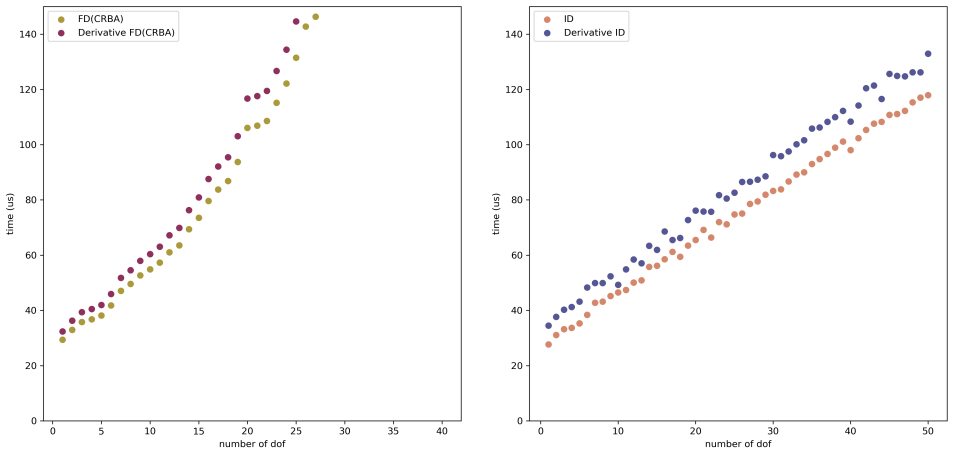
Figure 6.6: *Inertia* evaluation

Impact of Number of degrees of freedom

The main observation from Fig. 6.4, Fig. 6.5 and Fig. 6.6 is that the overhead introduced by using **CasADi** doesn't seem to increase as a function of robot's complexity. Even though direct numerical implementations are faster, the increase in performance it's not significant if one is willing to factor in the time required by the sensitivity computation.

6.2 Derivatives - sensitivity analysis time computation

From 6.7, one can observe that the complexity of the robot doesn't change with the increase of the robot's complexity (degrees of freedom), nor with the increase of the number of symbolical parameters of the function. For this test, the full jacobian was computed.

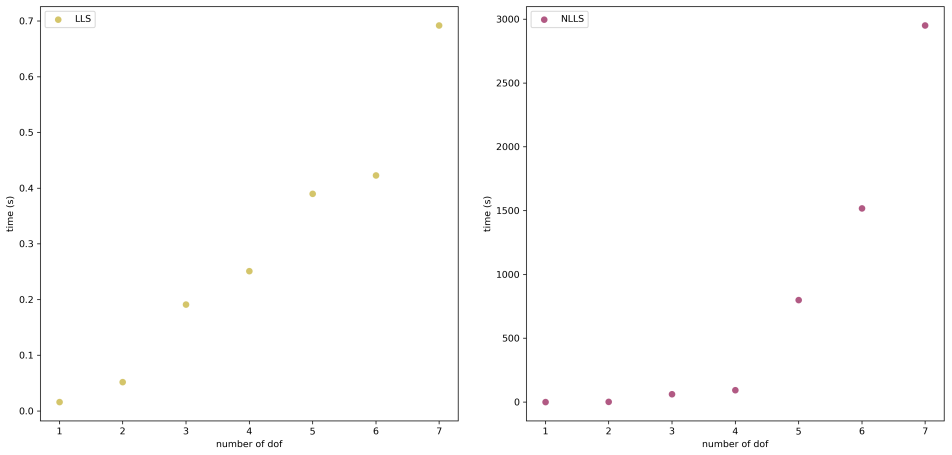


(a) *FD vs. total derivative* (b) *ID vs. total derivative*

Figure 6.7: Derivative computation *vs.* Function computation

6.3 System identification

As we can observe from 6.8, the inverse dynamics, linear least square tends to have a linear increase in computation whereas the multiple shooting, nonlinear last square has an exponential increase.



(a) *Linear least squares* (b) *Nonlinear Least Squares*

Figure 6.8: System identification

6.4 Videos

For a more direct interpretation of the final results, a visual inspection of some simulations as well as for model predictive control, while continuous parametric estimation is applied, can be accessed here [1]. As part of this list, the **Franka Emika Panda** robot was used.

[1] `https://www.youtube.com/playlist?list=PLrLtvnPdx3nES55ZrEqJPL42UNDTrMlkL`

CHAPTER 7

Conclusions and future work

The first thing one gets to observe is the engineering complexity, a project involving adapting control, requires. Without proper tools for automatic differentiation, that provide a focus on mathematical modeling, the entire process would have been significantly longer as well as heavily error-prone.

Another thing one gets to observe is the different types of challenges different sub-problems have.

The process of building the *ODE* requires more attention on the mathematical modeling side as well as building the necessary parameters from standard *URDF* robot format, the process is deterministic in nature.

On the other hand, the system identification process, besides requiring the proof of the equivalence of the *Recursive Newton-Euler Algorithm* for a generic implementation, also involves intermediate data analysis and physical interpretation of the results. This is necessary to establish the limits of the current approach, when it comes to what is not identifiable, as well as to motivate why this approach is trajectory independent and thus, does not require an optimal design method as well.

Last but not least, the model predictive control is very sensitive *w.r.t.* different parameters, like for example the frequency of data equation or the number of shooting nodes, etc. These problems were not addressed in this thesis (though, the current implementation, was influenced by them). A rigorous approach to this matter will involve a more extensive analysis using specific tools from non-linear dynamics, like for example *Lyapunov analysis*, [42] as well as other concepts from adaptive control [2].

It goes without saying, the proposed multi-body framework, by comparison with the current *state-of-the-art* implementation, is not faster. Nor it tries to be. The main goal of this thesis was to provide a proof of concept for parametric estimation problems in the context of multi-body dynamics, thus, the list of possible improvements is big, among others we mention:

- Adding constraints for self-collision avoidance ([27])
- Implementing optimal control/MPC on top of *Inverse Dymaics* [25]
- Implementation of high level *API* for *CasADi's Conic solver*, required for physically plausible system identification [44] [45])
- Applying the *Multiple shooting, non-linear least squares* after *Linear least squares*
- Moving away from *URDF* robot topology limitation
- Adding friction
- Integrate motion planning [28] [29]
- Test using a *real* robot

Bibliography

[1] Joel AE Andersson, Joris Gillis, Greg Horn, James B Rawlings, and Moritz Diehl. Casadi: a software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation*, 11(1):1–36, 2019.

[2] Karl J Åström and Björn Wittenmark. *Adaptive control*. Courier Corporation, 2013.

[3] Christopher G Atkeson, Chae H An, and John M Hollerbach. Estimation of inertial parameters of manipulator loads and links. *The International Journal of Robotics Research*, 5(3):101–119, 1986.

[4] David Baraff. *Dynamic simulation of nonpenetrating rigid bodies*. Cornell University, 1992.

[5] Atilim Gunes Baydin, Barak A Pearlmutter, Alexey Andreyevich Radul, and Jeffrey Mark Siskind. Automatic differentiation in machine learning: a survey. *Journal of Marchine Learning Research*, 18:1–43, 2018.

[6] Bradley M Bell. Cppad: a package for c++ algorithmic differentiation. *Computational Infrastructure for Operations Research*, 57(10), 2012.

[7] Jeff Bezanson, Alan Edelman, Stefan Karpinski, and Viral B Shah. Julia: A fresh approach to numerical computing. *SIAM review*, 59(1):65–98, 2017.

[8] Hans Georg Bock and Karl-Josef Plitt. A multiple shooting algorithm for direct solution of optimal control problems. *IFAC Proceedings Volumes*, 17(2):1603–1608, 1984.

[9] Max Yiye Cao, Stephen Laws, and Ferdinando Rodriguez y Baena. Six-axis force/torque sensors for robotics applications: A review. *IEEE Sensors Journal*, 2021.

[10] Horea-Alexandru Caramizaru. Multi-body modeling of robot dynamics and system identification during MPC, Research Software. `https://github.com/nashmit/thesis-scientific-computing`, 12 2022.

[11] Justin Carpentier, Guilhem Saurel, Gabriele Buondonno, Joseph Mirabel, Florent Lamiraux, Olivier Stasse, and Nicolas Mansard. The pinocchio c++ library: A fast and flexible implementation of rigid body dynamics algorithms and their analytical derivatives. In *2019 IEEE/SICE International Symposium on System Integration (SII)*, pages 614–619. IEEE, 2019.

[12] Erwin Coumans. Bullet physics simulation. In *ACM SIGGRAPH 2015 Courses*, page 1. 2015.

[13] M. Diehl, D.B. Leineweber, and A.A.S. Schäfer. MUSCOD-II Users' Manual. Technical Report 2001-25, 2001.

[14] Moritz Diehl. *Real-time optimization for large scale nonlinear processes*. PhD thesis, 2001.

[15] Moritz Diehl, Rolf Findeisen, Stefan Schwarzkopf, Ilknur Uslu, Frank Allgöwer, Hans Georg Bock, Ernst-Dieter Gilles, and Johannes P Schlöder. An efficient algorithm for nonlinear model predictive control of large-scale systems part i: Description of the method (ein effizienter algorithmus für die nichtlineare prädiktive regelung großer systeme teil i: Methodenbeschreibung). 2002.

[16] Tom Erez and Emanuel Todorov. Trajectory optimization for domains with contacts using inverse dynamics. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4914–4919. IEEE, 2012.

[17] Roy Featherstone. *Rigid body dynamics algorithms*. Springer, 2014.

[18] Martin L Felis. Rbdl: an efficient rigid-body dynamics library using recursive algorithms. *Autonomous Robots*, 41(2):495–511, 2017.

[19] M Gautier and W Khalil. Exciting trajectories for robot inertial parameters identification. *IFAC Proceedings Volumes*, 25(15):585–590, 1992.

[20] Andreas Griewank and Andrea Walther. *Evaluating derivatives: principles and techniques of algorithmic differentiation*. SIAM, 2008.

[21] https://abedgnu.github.io. Unified robot description format urdf. `https://abedgnu.github.io/Notes-ROS/_images/link.png`, 2022. [Online; accessed September, 2022].

[22] https://abedgnu.github.io. Unified robot description format urdf. `https://abedgnu.github.io/Notes-ROS/_images/joint.png`, 2022. [Online; accessed September, 2022].

[23] https://www.engineersedge.com. Truck and car universal joint design and engineering equation. `https://www.engineersedge.com/power_transmission/images/universal-joint.jpg`, 2022. [Online; accessed September, 2022].

[24] Lill Maria Gjerde Johannessen, Mathias Hauan Arbo, and Jan Tommy Gravdahl. Robot dynamics with urdf & casadi. In *2019 7th International Conference on Control, Mechatronics and Automation (ICCMA)*, pages 1–6. IEEE, 2019.

[25] Sotaro Katayama and Toshiyuki Ohtsuka. Efficient solution method based on inverse dynamics for optimal control problems of rigid body systems. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2070–2076. IEEE, 2021.

[26] Junggon Kim. Lie group formulation of articulated rigid body dynamics. Technical report, Technical report, Carnegie Mellon University, 2012.

[27] K Knese. Realizing online (self-) collision avoidance based on inequality constraints with hierarchical inverse kinematics. *Master's thesis, Technical University of Munich (July 2014)*, 2014.

[28] Steven M La Valle. Motion planning. *IEEE Robotics & Automation Magazine*, 18(2):108–118, 2011.

[29] Jean-Claude Latombe. *Robot motion planning*, volume 124. Springer Science & Business Media, 2012.

[30] D.B. Leineweber. The theory of MUSCOD in a nutshell. Technical Report 96-19, 1996.

[31] Kevin M Lynch and Frank C Park. *Modern robotics*. Cambridge University Press, 2017.

[32] Charles C Margossian. A review of automatic differentiation and its efficient implementation. *Wiley interdisciplinary reviews: data mining and knowledge discovery*, 9(4):e1305, 2019.

[33] Carlos Mastalli, Rohan Budhiraja, Wolfgang Merkt, Guilhem Saurel, Bilal Hammoud, Maximilien Naveau, Justin Carpentier, Ludovic Righetti, Sethu Vijayakumar, and Nicolas Mansard. Crocoddyl: An efficient and versatile framework for multi-contact optimal control. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2536–2542. IEEE, 2020.

[34] Richard M Murray, Zexiang Li, and S Shankar Sastry. *A mathematical introduction to robotic manipulation*. CRC press, 2017.

[35] Frank C Park, James E Bobrow, and Scott R Ploen. A lie group formulation of robot dynamics. *The International journal of robotics research*, 14(6):609–618, 1995.

[36] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

[37] Michael Posa, Cecilia Cantu, and Russ Tedrake. A direct method for trajectory optimization of rigid bodies through contact. *The International Journal of Robotics Research*, 33(1):69–81, 2014.

[38] Guillermo Rodriguez, Abhinandan Jain, and K Kreutz-Delgado. Spatial operator algebra for multibody system dynamics. *Journal of the Astronautical Sciences*, 40(1):27–50, 1992.

[39] Michael A Sherman, Ajay Seth, and Scott L Delp. Simbody: multibody dynamics for biomedical research. *Procedia Iutam*, 2:241–261, 2011.

[40] Shubham Singh, Ryan P Russell, and Patrick M Wensing. Analytical second-order partial derivatives of rigid-body inverse dynamics. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 11781–11788. IEEE, 2022.

[41] Shubham Singh, Ryan P Russell, and Patrick M Wensing. Efficient analytical derivatives of rigid-body dynamics using spatial vector algebra. *IEEE Robotics and Automation Letters*, 7(2):1776–1783, 2022.

[42] Jean-Jacques E Slotine, Weiping Li, et al. *Applied nonlinear control*, volume 199. Prentice hall Englewood Cliffs, NJ, 1991.

[43] Russ Tedrake. Underactuated robotics, 2022. *URL http://underactuated. mit. edu*.

[44] Silvio Traversaro, Stanislas Brossette, Adrien Escande, and Francesco Nori. Identification of fully physical consistent inertial parameters using optimization on manifolds. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5446–5451. IEEE, 2016.

[45] Patrick M Wensing, Sangbae Kim, and Jean-Jacques E Slotine. Linear matrix inequalities for physically consistent inertial parameter identification: A statistical perspective on the mass distribution. *IEEE Robotics and Automation Letters*, 3(1):60–67, 2017.

[46] Andrew Witkin and Michael Kass. Spacetime constraints. *ACM Siggraph Computer Graphics*, 22(4):159–168, 1988.

Erklärung:

Ich versichere, dass ich diese Arbeit selbstständig verfasst habe und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Heidelberg, den (Datum) .....

Declaration:

I hereby confirm that I wrote this work independently and did not use any sources other than those indicated.

Heidelberg, (Date) .....