



**A NOVEL DUAL QUATERNION BASED NEWTON-EULER INVERSE
DYNAMICS ALGORITHM**

CRISTIANA MIRANDA DE FARIAS

**DISSERTAÇÃO DE MESTRADO EM ENGENHARIA ELÉTRICA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA**

**FACULDADE DE TECNOLOGIA
UNIVERSIDADE DE BRASÍLIA**

UNIVERSIDADE DE BRASÍLIA
FACULDADE DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

**A NOVEL DUAL QUATERNION BASED NEWTON-EULER INVERSE
DYNAMICS ALGORITHM**

**UM NOVO MÉTODO PARA O ALGORÍTMO DE NEWTON-EULER
PARA DINÂMICA INVERSA BEASEADO EM QUATERNIOS DUAIS.**

CRISTIANA MIRANDA DE FARIAS

ORIENTADOR: JOÃO YOSHIYUKI ISHIHARA

DISSERTAÇÃO DE MESTRADO EM
ENGENHARIA ELÉTRICA

PUBLICAÇÃO: PPGENE.DM-721/19

BRASÍLIA/DF: JULHO - 2019

UNIVERSIDADE DE BRASÍLIA
FACULDADE DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

A NOVEL DUAL QUATERNION BASED NEWTON-EULER INVERSE
DYNAMICS ALGORITHM


CRISTIANA MIRANDA DE FARIAS

DISSERTAÇÃO DE MESTRADO SUBMETIDA AO DEPARTAMENTO DE ENGENHARIA
ELÉTRICA DA FACULDADE DE TECNOLOGIA DA UNIVERSIDADE DE BRASÍLIA, COMO
PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE.

APROVADA POR:



JOÃO YOSHIYUKI ISHIHARA, Dr., ENE/UNB
(ORIENTADOR)



MARIANA COSTA BERNARDES MATIAS, Dra., ENE/UNB
(EXAMINADORA INTERNA)



HENRIQUE MARRA TAIRA MENEGAZ, Dr., FGA/UNB
(EXAMINADOR INTERNO)

Brasília, 02 de julho de 2019.

FICHA CATALOGRÁFICA

MIRANDA DE FARIAS, CRISTIANA

A Novel Dual Quaternion based Newton-Euler Inverse Dynamics Algorithm [Distrito Federal] 2019.
xi+100 p., 210 x 297 mm (ENE/FT/UnB, Mestre em Engenharia Elétrica, Engenharia Elétrica, 2019).

DISSERTAÇÃO DE MESTRADO – Universidade de Brasília, Faculdade de Tecnologia.

Departamento de Engenharia Elétrica

- | | |
|-------------------------|---|
| 1. Dual Quaternion | 2. Newton Euler Inverse Dynamic Algorithm |
| 3. Robotic Manipulation | 4. Dynamic Modeling |
| I. ENE/FT/UnB | II. Título (série) |

REFERÊNCIA BIBLIOGRÁFICA

CRISTIANA, M (2019). A Novel Dual Quaternion based Newton-Euler Inverse Dynamics Algorithm, DISSERTAÇÃO DE MESTRADO, Publicação PPGENE.DM-721/19, Departamento de Engenharia Elétrica, Universidade de Brasília, Brasília, DF, xi+100.

CESSÃO DE DIREITOS

AUTOR: Cristiana Miranda de Farias

TÍTULO: A Novel Dual Quaternion based Newton-Euler Inverse Dynamics Algorithm.

GRAU: Mestre ANO: 2019

É concedida à Universidade de Brasília permissão para reproduzir cópias desta dissertação de mestrado e para emprestar ou vender tais cópias somente para propósitos acadêmicos e científicos. O autor reserva outros direitos de publicação e nenhuma parte dessa dissertação de mestrado pode ser reproduzida sem autorização por escrito do autor.

Cristiana Miranda de Farias

Departamento de Eng. Elétrica (ENE) - FT

Universidade de Brasília (UnB)

Campus Darcy Ribeiro

CEP 70919-970 - Brasília - DF - Brasil

ACKNOWLEDGMENTS

The experience of developing a Master thesis has been unique in my life. Between the many courses I have attended, there were also countless hours of studying, thinking and overcoming many difficulties. And it is in these most challenging moments that am most grateful for the many people around me, giving me support and many moments of fun.

First I would like to express my gratitude to my advisor Prof. João Y. Ishihara, who was always very supportive and always provided me with excellent advice and great insights on the topics I was studying. I would also like to extend my gratitude to all my labmates and friends from LARA, thank you for the companionship, laughs and cake. I love being a part of this team and I am very grateful for all the support you have all given me through the years. Also, I'd like to thank all the people at BioRob. The months I spent interning in Switzerland were of incredible growth and learning.

I would also like to thank Luis for all the help and support you have given me, even though you were so distant.

Moreover, I would like to give a special thanks for all my dear lifelong friends. Girls, thanks for cheering me on and for giving me so much support, especially when I was stressed and cranky. I would also like to thank the boys from my years doing mechatronics at UnB, thank you for always being there.

Finally, I could not forget to thank my wonderful family and my incredible boyfriend for always standing beside me on all my endeavours, you are amazing.

RESUMO

Título: Um novo método para o algoritmo de Newton-Euler para dinâmica inversa baseado em quatérnios duais.

Autor: Cristiana Miranda de Farias

Orientador: João Yoshiyuki Ishihara

Neste trabalho, o conhecido algoritmo recursivo de dinâmica inversa de Newton-Euler para manipuladores seriais é explorado e reformulado no contexto da álgebra dos Quatérnios Duais — como o dqRNEA (dual quaternion based Recursive Newton-Euler inverse dynamics). Aqui nós estruturamos a descrição da cinemática para fase direta a partir da álgebra de screws e deslocamentos de linha ao invés da já bem estabelecida parametrização de Denavit-Hartenberg. Dessa maneira, garantimos melhor eficiência e modelos dinâmicos mais simples. Além disso, a iteração inversa do algoritmo usa os valores calculados anteriormente para estimar os torques no espaço da junta.

Também apresentamos uma solução fechada para o dqRNEA, ou seja, reformulamos o problema recursivo como uma equação fechada baseada em matrizes de quatérnios duais. A fim de formalizar tal equação, definimos algumas das propriedades algébricas para vetores de quatérnios duais e matrizes de quatérnios duais. Além disso, com uma formulação fechada do dqRNEA, também é possível formalizar um esquema de controle de dinâmica inversa para o sistema. Ou seja, um método de linearização por feedback para controlar os torques do manipulador serial no espaço das juntas.

Finalmente, este trabalho também aborda a questão da validação do algoritmo, tanto em termos de eficiência quanto em termos de funcionalidade. Com o objetivo de validar eficiência, é feita uma análise de custo computacional das principais operações de Quatérnios Duais e do dqRNEA como um todo. Estes resultados são comparado com outros da literatura. Para melhorar ainda mais o custo, também foi reformulada a operação adjunta, de modo que ela requeira menos operações em sua execução. Para validar as funcionalidades do controlador, foi criado um pacote MATLAB para executar o algoritmo dqRNEA, tanto da formulação recursiva quanto na fechada, e testamos ambos junto a outros métodos na literatura e com o controle de dinâmica inversa.

ABSTRACT

Title: A Novel Dual Quaternion based Newton-Euler Inverse Dynamics Algorithm

Author: Cristiana Miranda de Farias

Supervisor: João Yoshiyuki Ishihara

In this manuscript, the well known recursive Newton- Euler inverse dynamics algorithm for serial manipulators is explored and reformulated into the context of the algebra of Dual Quaternions—as the dqRNEA (dual quaternion based Recursive Newton-Euler inverse dynamics). Here we structure the forward kinematic description with screws and line displacements rather than the well established Denavit-Hartenberg parameters, thus accounting better efficiency, compactness and simpler dynamical models. Furthermore, the backwards iteration uses the previously calculated values for estimating the joint space torques.

We also present a closed solution for the dqRNEA, that is, we reformulate the recursive problem as a closed equation based on dual quaternion-matrices. In order to accomplish such an endeavor, we formalize some of the algebra for dual quaternion-vectors and dual quaternion-matrices. Moreover, with a closed formulation of the dqRNEA we are also capable of creating a dual quaternion based formulation for the computed torque control, a feedback linearization method for controlling a serial manipulator's torques in the joint space.

In addition, this manuscript also covers the matter of validating the algorithm, both in terms of efficiency and in terms of functionality. Aiming to accomplish the former a computational cost analysis of the main Dual Quaternions operations and of the Newton-Euler inverse dynamics algorithm as a whole is made and compared with other results in the literature. To further improve on cost, we also reformulate the adjoint operation to run with fewer operations. To validate the functionality of our controller we create a MATLAB package to run the dqRNEA algorithm in both the recursive and closed formulation, and we test it both against other methods in the literature and with the computed torque control.

CONTENTS

1	INTRODUCTION	1
1.1	STATE OF THE ART	3
1.2	OBJECTIVES.....	4
1.3	CONTRIBUTIONS	5
1.4	ORGANIZATION OF THIS MANUSCRIPT.....	5
2	MATHEMATIC FUNDAMENTALS	7
2.1	DUAL QUATERNION REPRESENTATION	7
2.1.1	QUATERNIONS	8
2.1.1.1	ISOMORPHISM TO \mathbb{R}^4 AND \mathbb{R}^3	12
2.1.2	DUAL NUMBERS	13
2.1.3	DUAL QUATERNIONS	13
2.1.3.1	ISOMORPHISM TO \mathbb{R}^8 AND \mathbb{R}^6	18
2.1.4	LINEAR TRANSFORMATIONS	19
2.1.5	DUAL QUATERNION VECTORS AND MATRICES.....	21
2.1.6	ADVANTAGES OF DUAL QUATERNIONS	25
2.2	DUAL QUATERNIONS GEOMETRIC REPRESENTATION.....	25
2.2.1	POINTS IN THREE-DIMENSIONAL SPACE	25
2.2.2	PLUCKER COORDINATES: LINES IN THREE-DIMENSIONAL SPACE.....	30
2.2.3	THE EXPONENTIAL: CURVES IN THREE-DIMENSIONAL SPACE	32
3	ROBOTIC FUNDAMENTALS.....	33
3.1	RIGID BODY KINEMATICS	33
3.1.1	SCREW THEORY	35
3.1.2	QUATERNION BASED REPRESENTATION OF RIGID BODY ROTATION	38
3.1.3	DUAL QUATERNION BASED REPRESENTATION OF GENERAL RIGID MOTION	39
3.2	RIGID BODY DYNAMICS	41
3.2.1	DYNAMICAL EQUATIONS OF MOTION.....	41
3.2.2	INERTIA TENSOR.....	43
3.2.3	RIGID BODY DYNAMICS IN DUAL QUATERNION ALGEBRA	44
3.3	MANIPULATOR KINEMATICS	46
3.3.1	FORWARD POSITION KINEMATICS MODELING	48
3.3.1.1	DENAVIT-HARTENBERG PARAMETERS.....	48
3.3.1.2	PRODUCT OF EXPONENTIALS	49
3.3.1.3	COMPARISON.....	50
4	DUAL QUATERNION RECURSIVE NEWTON EULER ALGORITHM (DQRNEA)	52
4.1	PRELIMINARIES	52
4.1.1	INVERSE DYNAMICS.....	53
4.1.1.1	LAGRANGE-EULER FORMULATION	53
4.1.1.2	RECURSIVE NEWTON-EULER FORMULATION	54
4.1.1.3	COMPARISON	55

4.2	ALGORITHM DEVELOPMENT	55
4.2.1	FORWARD RECURSION	56
4.2.1.1	ALGORITHM INITIALIZATION	56
4.2.1.2	RECURSIVE POSITION AND TWIST	57
4.2.1.3	RECURSIVE ACCELERATION	57
4.2.2	BACKWARD RECURSION	58
4.2.3	ALGORITHM SUMMARY	60
4.3	DQRNEA IN CLOSED FORM	60
4.3.1	DUAL QUATERNIONIC-MATRIX FORMULATION	60
4.3.2	CLOSED FORM EQUATION	68
4.4	DQ BASED INVERSE DYNAMICS CONTROLLER	70
4.4.1	COMPUTED TORQUE CONTROL	70
5	COMPUTATIONAL COST ANALYSIS	72
5.1	METHODOLOGY FOR COMPUTATIONAL COST ANALYSIS	72
5.2	COMPUTATIONAL COST OF DQ-BASED OPERATIONS	73
5.2.1	COSTS FOR QUATERNION OPERATIONS	73
5.2.2	COSTS FOR DUAL QUATERNION OPERATIONS	73
5.2.3	OPTIMIZING THE ADJOINT MAPPING	74
5.2.3.1	COMPUTATIONAL COSTS FOR THE NEW ADJOINT REPRESENTATION	77
5.3	COMPUTATIONAL COST OF DQRNEA	79
5.4	COMPUTATIONAL COSTS OF HOMOGENEOUS TRANSFORMATION MATRICES	82
5.4.1	COST OF HTM OPERATIONS	82
5.4.2	COST OF HTM ALGORITHM	83
5.5	COMPUTATIONAL COST COMPARISON	85
5.5.1	RESULTS IN THE LITERATURE	85
5.5.2	ALGORITHMS COMPARISON	86
6	SIMULATION RESULTS	91
6.1	METHODOLOGY	91
6.1.1	MATLAB PACKAGE	91
6.1.2	VALIDATION PACKAGES	91
6.1.3	COMPUTATIONAL SETUP	93
6.2	TWO LINK PLANAR ARM: MODELING AND CONTROLLING A SIMPLE ROBOT	93
6.2.1	ROBOT MODEL	93
6.2.2	NUMERICAL RESULTS	95
6.2.3	CONTROL ALGORITHM	96
6.3	KUKA ROBOT: THE DQRNEA BASED CONTROLLER	96
6.3.1	ROBOT MODEL	96
6.3.2	NUMERICAL RESULTS	101
6.3.3	MATLAB CONTROLLER	101
7	CONCLUSION	106
7.1	FUTURE WORK	106
A	INVERSE DYNAMIC ALGORITHM WITH HOMOGENEOUS MATRICES	108

A.1 HTM 108

 A.1.1 POSITIONS, TWISTS, FORCES AND WRENCHES AS VECTORS 108

 A.1.2 ROTATION MATRIX 110

 A.1.3 HOMOGENEOUS TRANSFORMATION MATRICES 111

A.2 THE NEWTON-EULER ALGORITHM AS PRESENTED IN THE LITERATURE..... 111

B PUBLICATIONS 113

BIBLIOGRAPHY 114

List of Figures

2.1	Representation of a point in Imaginary Plane represented in different frames	26
2.2	Representation of a point in three-dimensional space	28
2.3	Representation of a Plücker Line in three-dimensional space.	31
3.1	Generic illustration of a manipulator to exemplify the base, links and end-effector being connected by different joints. In this illustration the robot is located besides the world frame.	34
3.2	Generic illustration of a manipulator with one of its links—that is one of the rigid bodies from its structure—being highlighted.	35
3.3	Representation of a rigid displacement: Object rotates with angle θ around line with plucker coordinates l and then moves distance d parallel to the screw axis. <i>Source [1]</i>	36
3.4	System of forces and moments acting on a body. In this illustration we have 3.4a being the representation of some of the forces and moments acting on the link (the red straight arrows represent the forces being transmitted on each joint and f_3 is the resulting gravity force the curved arrows are the moments acting at the joints). In 3.40 we have a representation of the resulting body wrench on the body (the straight arrow represents the resulting force f_b and the curved arrow represents the resulting moment m_b).	42
3.5	In 3.5a we have an illustration of the Forward kinematics problem, in which we aim to find end-effector pose given the joints. In 3.5b the inverse problem is illustrated, in which we want to calculate the joints based on the end-effector’s pose.	47
3.6	Representation of the product of exponentials formula. Here we show the screw transformation for each of the links. In <i>A</i> we have the robot at its home position, with all $\theta_i = 0$, from <i>B-D</i> we are moving one link at a time until the final robot configuration is reached in <i>D</i>	49
4.1	Illustration of the steps taken to compute the RNEA in a three link manipulator.	54
4.2	Block diagram for computed torque control, following equations (4.77)-(4.82).....	71
5.1	Costs for Forward Kinematics computation using different representations. In blue we have HTM and in red we have DQ representation. With the dotted line is the cost for D-H based parametrization and the straight line is the screw theory parametrization. The value of n ranges from 1-42.	87
5.2	Total costs for RNEA of both HTM and DQ. In the plots, the HTM is represented in blue and the DQ is represented in Red. Also n ranges from 1-42.	89
5.3	Total costs the different versions of the algorithm presented in Table 5.4. with n ranging from 1-42.	90
6.1	Schematic Upper View of a two link planar arm in which both links measure 1m and the center of mass is located exactly at the middle point of the link.	93
6.2	Matlab 3D Robot Plot for the two-link planar arm.	94
6.3	Torque plot for each of the joints in different versions of the algorithm. The straight lines are the results for the matrix-quaternion formulation, the dash-dotted lines refers to the closed equation and the dotted line are the torques for the recursive algorithm. Finally the dashed line are the resulting torques for the recursive Newton-Euler algorithm from Peter Corke’s Toolbox. .	95

6.4	Controller Design on Simulink—From the Peter Corke Toolbox, with the addition of our dual quaternion based recursive newton euler function.	97
6.5	Robot plot at different configurations. Here we merged the images for the plot at the initial joint configuration θ_0 and the final joint configuration θ_{end}	98
6.6	Plot for the Error Progression for the experiments in Figure 6.5	98
6.7	Model of the robot, from left to right we have the DH model of the KUKA LWR4, taken from [2], followed by the joint position and orientation and finally the center of mass fro each link—with positions c_1 to c_7 shown in Table	99
6.8	Torque plot for each of the joints in different versions of the algorithm. The straight lines are the results for the matrix-quaternion formulation, the dash-dotted lines refers to the closed equation and the dotted line are the torques for the recursive algorithm. Finally the dashed line are the resulting torques for the recursive Newton-Euler algorithm from Peter Corke’s Toolbox. .	102
6.9	Simulink Computed Torque Control Diagram for KUKA LWR4. This is based on Peter Corke robotic toolbox with the addition of our dual quaternion based recursive Newton Euler algorithm.	103
6.10	Robot plot at different configurations. Here we merged the images for the plot at the initial joint configuration θ_0 and the final joint configuration θ_{end}	104
6.11	Plot for the error progression for the experiments in Figure 6.10.....	105

List of Tables

4.1	List of main operations in Dual Quaternionic-Matrix formulation	68
5.1	Cost requirements of quaternion algebra operations	73
5.2	Cost requirements for dual quaternion operations	74
5.3	Comparison between different manners of performing the adjoint mapping	80
5.4	Cost of operations for UDQ.....	82
5.5	Cost requirements for vector operations	83
5.6	Cost requirements for HTM operations	83
5.7	Cost of operations for HTM	85
6.1	Function description for dqRNEA MatLab Package. Variables from Subsection 4.3.2.....	92
6.2	DH parameters for the KUKA robot, based on [2]	99
6.3	PoE formulation for the KUKA robot–center of mass position with regards to equivalent joint, as in Figure 6.7.....	101

SYMBOLS LIST

Basic Symbols

$a, b, c...$	Real Scalars are represented by lower case plain letters
$\mathbf{a}, \mathbf{b}, \mathbf{c}...$	Real Vectors are represented by lower case bold letters
$\mathbf{a}, \mathbf{b}, \mathbf{c}...$	Quaternions and Pure Quaternions are represented by lower case bold letters
$\underline{a}, \underline{b}, \underline{c}...$	Dual Numbers are represented by lower case underlined letter
$\underline{\mathbf{a}}, \underline{\mathbf{b}}, \underline{\mathbf{c}}...$	Dual Quaternions and Pure Dual Quaternions are represented by lower case bold underlined letters
$\underline{\mathbf{A}}, \underline{\mathbf{B}}, \underline{\mathbf{C}}...$	Dual Quaternion Vectors and Pure Dual Quaternions are represented by upper case bold underlined letters
$\underline{\mathbf{A}}, \underline{\mathbf{B}}, \underline{\mathbf{C}}...$	Dual Quaternion Matrices and Pure Dual Quaternions are represented by upper case bold underlined letters

Symbols Related to Sets and Groups

\mathbb{R}	Real Number Set
\mathbb{R}^n	Real vector with n entries
$\mathbb{R}^{n \times m}$	Real n by m Matrix
\mathbb{C}	Complex number Set
\mathbb{H}	Quaternion Set
\mathbb{H}_0	Pure Quaternion Set
\mathcal{S}^3	Unit Quaternion Set
$\text{Spin}(3)$	Unit Quaternion Group
\mathbb{D}	Dual Numbers Set
$\underline{\mathbb{H}}$	Dual Quaternions Set
$\underline{\mathbb{H}}_0$	Pure Dual Quaternion Set
$\underline{\mathcal{S}}$	Unit Dual Quaternion Set
$\text{Spin}(3) \times$	Unit Dual Quaternion Group
\mathbb{R}^3	
$\underline{\mathbb{H}}^n$	Dual Quaternion Vector with n elements
$\underline{\mathbb{H}}^{n \times m}$	Dual Quaternion n by m Matrix
$\underline{\mathbb{H}}_0^n$	Pure Dual Quaternion Vector with n elements
$\underline{\mathbb{H}}_0^{n \times m}$	Pure Dual Quaternion n by m Matrix

Symbols Related to Quaternions, Dual Numbers and Dual Quaternions

$\hat{i}, \hat{j}, \hat{k}$	Imaginary Units
ε	Dual Unit
\mathbf{a}^* and $\underline{\mathbf{a}}^*$	Conjugate of a quaternion \mathbf{a} and a dual quaternions $\underline{\mathbf{a}}$, respectively
\mathbf{a}^{-1} and $\underline{\mathbf{a}}^{-1}$	Inverse of a quaternion \mathbf{a} and a dual quaternions $\underline{\mathbf{a}}$, respectively
$\underline{\mathbf{A}}^T$	Transpose of a dual quaternion Vector or Matrix $\underline{\mathbf{A}}$
$\underline{\mathbf{A}}^*$	Conjugate of a dual quaternion Vector or Matrix $\underline{\mathbf{a}}$
$\text{Re}(\mathbf{q})$ $\text{Im}(\mathbf{q})$	Real and Imaginary Parts of the quaternion \mathbf{q}
$\text{Re}(\underline{\mathbf{q}})$ $\text{Im}(\underline{\mathbf{q}})$	Real and Imaginary Parts of the dual quaternion $\underline{\mathbf{q}}$
$\mathcal{P}(\underline{\mathbf{q}}), \mathcal{D}(\underline{\mathbf{q}})$	Primary and Dual Parts of the dual quaternion $\underline{\mathbf{q}}$
$\text{vec}_3\mathbf{a}$ and $\text{vec}_4\mathbf{a}$	Transformation from a pure quaternion or a quaternion to a vector in \mathbb{R}^3 or \mathbb{R}^4 , respectively
$\text{vec}_6\underline{\mathbf{a}}$ and $\text{vec}_8\underline{\mathbf{a}}$	Transformation from a pure dual quat. or dual quat. to a vector in \mathbb{R}^6 or \mathbb{R}^8 , respectively
vec	Inverse of the vector Map
$\ \mathbf{a}\ $ and $\ \underline{\mathbf{a}}\ $	Quaternion and dual quaternion norm
$\overset{+}{\mathbf{h}}(\bullet)$ and $\overset{-}{\mathbf{h}}(\bullet)$	Quaternion Hamiltonians
$\overset{+}{\underline{\mathbf{h}}}(\bullet)$ and $\overset{-}{\underline{\mathbf{h}}}(\bullet)$	Dual Quaternion Hamiltonians
$\text{Ad}(\mathbf{a})$ and $\text{Ad}(\underline{\mathbf{a}})$	Quaternion and Dual Quaternion Adjoint
$\mathcal{T}_4(\bullet)$ and $\mathcal{T}_8(\bullet)$	Quaternion Linear Transformation and Dual Quaternion Linear Transformation
$G(\underline{\mathbf{a}})$	Dual Inertia Transformation
$G(\underline{\mathbf{A}})$	Dual Quaternion Vector Inertia Transformation

Quaternion and Dual Quaternion Operators

$\mathbf{a} \times \mathbf{b}$	Is the cross product operator between two pure quaternions
$\underline{\mathbf{a}} \times \underline{\mathbf{b}}$	Is the cross product operator between two pure dual quaternions
$\mathbf{a} \cdot \mathbf{b}$	Is the inner product operator between two pure quaternions
$\underline{\mathbf{a}} \cdot \underline{\mathbf{b}}$	Is the inner product operator between two pure dual quaternions
$\underline{\mathbf{a}} \odot \underline{\mathbf{b}}$	Is the double geodesic product operator between two pure dual quaternions
$\underline{\mathbf{A}} \otimes \underline{\mathbf{B}}$	Is the element-wise cross product operator between two pure dual quaternions vectors
$\underline{\mathbf{A}} \odot \underline{\mathbf{B}}$	Is the element-wise inner product operator between two pure dual quaternions vectors
$\underline{\mathbf{A}} \otimes \underline{\mathbf{B}}$	Is the element-wise double-Geodesic product operator between two pure dual quaternions vectors

Important Variables

τ_i	Joint Torques for robot joint i
$\boldsymbol{\tau}$	Joint torque vector for robotic manipulator
$\underline{\theta}_i$ and θ_i	Double angle and angle for a robot joint i , respectively
$\boldsymbol{\theta}$	Joint Angle Vector for robotic manipulator
\mathcal{F}	Frame of reference
$\boldsymbol{\omega}_i$	Twist of rigid body i
\boldsymbol{W}	Twist Vector for robotic manipulator
$\boldsymbol{\omega}_i$	Angular Velocity of rigid body i
\boldsymbol{v}_i	Linear Velocity of rigid body i
$\underline{\boldsymbol{f}}_i$	Wrench of rigid body i
\boldsymbol{F}	Wrench Vector for robotic manipulator
\boldsymbol{f}_i	Force acting on rigid body i
\boldsymbol{m}_i	moment acting on rigid body i
$\underline{\boldsymbol{a}}_i$	Screw Axis of rigid body i

Other considerations

Throughout this manuscript we generally have

- \boldsymbol{x}_B^A being the DQ representation of frame B in terms of frame A ;
- \boldsymbol{x}_A^B as transformation from frame A to frame B ;
- $\underline{\boldsymbol{p}}_B^A$, for a $\underline{\boldsymbol{p}}$ representing position, velocity or acceleration being the coordinate representation in A frame of the position, velocity or acceleration of frame B .

1

INTRODUCTION

Throughout history, humans have always built machines to facilitate their work and make their labor more bearable. Indeed, as we have built the wheel to make transport easier we have also created automated assembly lines to spare workers from their monotonous (and many times dangerous) tasks. However, the story does not end there: although we do build machines to help us out in our daily lives, we have also built machines to go to places we could not go by ourselves. We have flown through the skies and we have dived to the depths of the ocean, we raced around the earth and conquered the moon. Everyday science and engineering evolve to allow us to go where we have never gone before and to do things that, once upon a time, were only science fiction.

It was indeed in fiction that human ingenuity has envisioned a perfect helper—the robot. This machine would never tire and would be able to perform incredible tasks with amazing precision for the benefit of humanity (when they were not rebelling against their creators, of course). The Czech author Karel Čapek is usually considered to have been the first person to use the word robot in its modern context: his 1921 play *Rossumovi Univerzální Roboti* (in English Rossum’s Universal Robots) starts out in a factory that creates artificial people—the robots—to work for the humans. The narrative of this play centers on the mistrust people at the time felt over the machine, and as such, was very popular with the public, running for 184 performances on Broadway and popularizing the term that is used to this day [3].

Although the modern meaning of word robot was popularized only in the 1920’s, the idea of substituting real individuals by machines has been around for a few millennia. In ancient Egypt priests would make animated statues to communicate the will of the gods, the Inuit people had legends of robot-like creatures created by sorcerers and Greek mythology has several tales of statues or metal creatures coming to life [3,4]. Of course, these ancient civilizations did not have the technology to make the robots of today, but there are also several records of working designs for automatons, the oldest one being a “the pigeon”, a bird-like contraption with a steam based propeller system, invented by Archytas of Tarentum (428–347 BC). Records of automata could also be found in ancient Alexandria and China, however, it was the Arabic civilization that have first created “programmable” robots, in the form of human figurines that could be adjusted to play different melodies [4,5].

Of course, as the centuries went by, different scientists and engineers have designed and imagined more complex machines operating in a number of different contexts (for a full overview of ancient robotic history see the works in [3,4,6]). Still, despite this long history we can trace for robotics, the modern design of these machines has only emerged on the 1950’s, with first modern robot being the Unimate, designed in 1954 by George Devol. The first Unimate, a manipulator capable of performing only one task and very difficult to program, was sold in 1961 to General Motors and in the following years automation has only grown. And so did research in robotics [6–8].

Despite its industrial roots, the robots of today have evolved—and is still evolving—to be fully integrated in our society. Indeed, the advances in technology and the maturity the research has achieved since the 1950’s has shifted the focus from industry to the human world, in which a new generation of robots is expected to safely operate in different segments of our society, providing aid for education [9–12], health-care [13–15], entertainment [5, 16] as well as a range of other services. More so, in the past few decades, a lot of effort has also been put to deploy robots to environments not suited to humans: we have robots exploring archaeological sites that are too fragile for humans to enter [17,18], we have robots diving to higher depths than a human could endure [19], we have robots exploring Mars and repairing space stations [20–23], drones are flying faster than

we could to deliver medication [24], researchers are developing robots to operate in ever more challenging rescue scenarios [25, 26] and these are only a few of the many applications being developed today.

In the context of the current research scenario, robots can already do amazing things and take us to explore many of the places we could not go ourselves. Yet, despite all the advancements, the field of robotics is still in its infancy, and there are many topics still to be better explored. Motivated by enormous potential of the area, this manuscript aims at exploring ways to enhance a robot’s capabilities of interacting with the environment.

Particularly, many of the applications being studied today demand for more adaptable solutions, working in unstructured environments while ensuring safety and being computationally efficient. Indeed, in the scenario of industrial robotics we many times encounter controlled environments, in which, although versatility might be welcomed, the nature of the tasks being performed require movements to be repeated many times under the same conditions with precision. The common, and simpler solution for this problem, was the implementation of position based controllers—if nothing were to change and conditions were known we could simply define a path for the robot to follow, and then control this path with as much precision as possible. Of course, a lot of innovation has come from those kinematic based controllers, as, after all, we are still studying more complex solutions in this field (cooperative robotics, soft robotics, etc). However, this solutions are still “human-unfriendly” and would not operate well at more unstructured environment, such as the world outside the factories or when we add more variables to the industrial process (such as a human collaborator). Indeed, its easy to imagine the very recurrent scenario of a robot manipulator set out to follow a path in order to grab an object, when suddenly something (or someone) blocks the path. If the robot considers only the kinematic variables, it will ignore the blockage and keep moving toward its target, thus causing damage or injury [27]. To deal with this kind of uncertainty it is important we also consider the forces of the system to ensure an efficient and safe response.

One way to guarantee the robustness and efficiency of a system is by choosing an adequate parametrization for it. Although there are many options in the literature, we believe the dual quaternions are particularly well suited for robotic applications, as they have strong algebraic properties and can be used to represent rigid motions, twists, wrenches and several geometrical primitives—e.g., Plücker lines, planes—in a very straightforward way [28].

In this manner, we set out to explore the usage of dual quaternions for describing the dynamics of a robotic manipulator and for controlling its forces in such a way that we could ensure safety for Human Robot Interaction (HRI) scenarios, as well as for the interaction of a robot with predominantly human environment. However, we notice the literature relating to a dual quaternion based dynamical model for a serial manipulator was quite lacking with works on the matter only surfacing a few months prior to this manuscript completion [29, 30] and still being very scarce.

Therefore, this manuscript presents a novel formulation of a dual quaternion based Recursive Newton-Euler Algorithm (dqRNEA)—as well its closed form equation. Here we add that we have not only chosen the dual quaternion representation, but we have also made some other important design choices to ensure that the algorithm is cost efficient and robust. We have chosen the Newton-Euler recursive formulation for the inverse dynamics rather than the Lagrange-Euler methods or Kane’s dynamical equations. Also, we have chosen to describe the robot forward kinematics by means of the Product of Exponential formulation rather than the more usual Denavit–Hartenberg parameters.

Finally, in this work we also aim at guaranteeing the validity and efficiency of our algorithms. In this manner, we have optimized some of the tools within the algebra of dual quaternions so that they would be faster and remain in the algebra—in particular we have optimized the adjoint operation, which was very inefficient

in its traditional form. More so, we have also made a very thorough cost analysis of all the equations in our algorithm and compared it to a similar formulation in another algebra. Lastly we have tested our dqRNEA formulation within a computed torque based controller in order to further validate the equations.

1.1 STATE OF THE ART

Rigid body motion and control have been extensively studied in the past decades in a number of disciplines, including, mechanical systems, robotic manipulation, satellites, etc. These were usually investigated and designed by exploiting the Homogeneous Transformation Matrices (HTM), however, recently, many new works have surfaced on the uses of dual quaternions (DQ) to describe such systems [31–35].

Particularly, regarding robotic manipulation, dual quaternion algebra proves itself to be advantageous compared to other representations as it allows us to represent the complete robot position and attitude—the pose—with a set of eight parameters. Moreover, it was argued in [36–39] that the unit dual quaternion (UDQ) in addition to being non-minimal and free of singularities, are also a more compact, efficient and less computationally demanding representation for rigid-body displacements compared to HTM [40–42]. Indeed, we can also list as further advantages of the dual quaternions the fact that they are very well suited to represent many geometric primitives, such as lines and planes, in a simple and intuitive way [28] and the fact that dual quaternions have been argued to be most efficient way to represent a screw motion [41]. The benefits of using dual quaternion algebra have been discussed also in many works with different applications comprising rigid body motion stabilization, tracking, multiple body coordination [35, 43], kinematic control of manipulators with single and multiple arms and human-robot interaction [37, 44–46], image based localization [47, 48], etc.

Although a lot has been published regarding the kinematic representation and control of robotic systems using dual quaternion algebra, there is still a lot to be done with respect to representing and controlling the dynamics of a robotic arm with DQ. Indeed, algorithms for rigid-body dynamics computation play a crucial role for simulation of motion, analysis of forces, torques and for the design of control techniques in robotics [36, 49, 50]. Rigid-body dynamics are at the core of many recent robotic applications in different fields, such as legged robot stabilization, forceful Human-Robot Interaction (HRI), computer animation and even biomechanics [51].

In Dooley and McCarthy’s work [34] the authors propose one of the earliest formulations for the dynamic modeling for an unconstrained rigid body and for a serial manipulator using dual quaternion algebra in Kane and Levinson’s formulation [52]. Although their work is pioneer, their equations are overly complicate and lack a clear and intuitive physical meaning. Other attempts to improve on [34] are presented on both the works of [53] and [35] in which both authors propose a decoupled formulation for both the rotational and translational dynamics using dual quaternions, and furthermore, they use this model in order to design a regulator for both the attitude and position of a body. Neither of this works exploit formulations for the dynamical problem that would couple together the translational and rotational forces acting on the body. Furthermore, a coupled formulation for the dynamic of an unconstrained rigid body can be found in [31, 32, 54] in which the authors exploit the manner in which dual quaternion twists and wrenches can couple together the linear and angular variables.

We may further extend the dynamics of a rigid body to the dynamics of a whole articulated robotic manipulator. In the literature, there are a few algorithms that are usually used to describe such systems [52, 55–57], with the most famous ones being the recursive Newton-Euler algorithm (RNEA) and the Lagrange-Euler (LE)

formulation. The LE equations and the RNEA are both used to describe the relations between the joint torques, forces at the end effector and kinematic variables, however, they differ in many other aspects. Concerning only computational complexity, the Newton–Euler formulation is considerably more efficient due to its inherently recursive formulation.

Regarding the dynamical representation of robotic manipulator in dual quaternion algebra, the literature is more lacking—especially when considering Newton-Euler’s algorithm. Recent works on the subject include [58], in which the authors exploit the principle of virtual work and UDQ to describe an efficient solution for the dynamics of a parallel manipulator. In [34, 59] both authors show an illustrative examples to how we can build the inverse dynamics of a manipulator, with [59] using the principle of virtual work and dual numbers and [34] the dual quaternion formulation and Kane’s equations.

The only other works to focus on the dynamical description of a serial manipulator through a dual quaternion based on the recursive Newton-Euler algorithm have been published recently by [29, 30] and by ourselves [60]. In [29, 30] the authors propose a formulation for describing the dynamics of a spacecraft-mounted robotic manipulator configured with different joint types. Although the equations in [29, 30] have some similarities to the ones proposed in this manuscript and in our work in [60] there are also many differences between the algorithms. In [29, 30] the authors propose framework, in which they take into account many different types of joints. However one big drawback in their formulation is that many equations presented in [29, 30] are in the Euclidean vector-matrix formulation, and therefore have to be mapped and back to dual quaternions, creating the need for transformations in and out of the algebra, which leads to more costly cumbersome and, although correct, unintuitive equations.

Opposed to [29, 30], our work [60] proposed a more intuitive solution for both the recursive and closed formulation of the Newton-Euler algorithm following the dual quaternion notation in the [37]. In its recursive formulation our dqRNEA clearly presents a forward and a backward iteration, and the in the closed formulation of the algorithm we identify the mass, Coriolis and gravity terms. More so, in addition, we also contribute with an optimization of the adjoint transformation allowing our algorithm to be more cost efficient.

1.2 OBJECTIVES

The main objectives of this manuscript is to develop a novel and cost efficient formulation for the Newton-Euler inverse dynamic algorithm based on the algebra of dual quaternions. With our dqRNEA we aim to answer whether the advantages we find when describing the kinematics of a robotic arm with dual quaternions would translate to a dynamical model.

With this context we aim to perform a cost analysis of the algorithm and compare it to results in the literature. More so, to fully validate our algorithm we also aim to integrate it with a computed torque control scheme and perform a few experiments.

Finally, we also aim to formulate our algorithm as a quaternion-matrix closed form equation, in which the terms related to mass, Coriolis forces and gravity are explicit.

1.3 CONTRIBUTIONS

The main contributions of this manuscript are:

- Development of a novel cost efficient dual quaternion based version of the Newton-Euler inverse dynamics algorithm (dqRNEA) based on the Product of Exponentials formulation for the forward kinematics. In addition to the traditional recursive method for the dqRNEA algorithm we also rearrange the equations in order to obtain the closed form inverse dynamic equations, from which the terms related to the mass, coriolis/centripetal and gravitational forces can be extracted.
- Development of an alternative version of the adjoint mapping within the algebra of dual quaternions. Our alternative representation stems from the fact that the traditional map is inefficient and, therefore, an optimization is necessary.
- We also provide, in this work, a cost analysis of the main operations within dual quaternion algebra, a cost analysis of the different methods for calculating the adjoint map and the cost analysis for the dqRNEA. Furthermore, we also analyze the costs of some of operations with Homogeneous Transformation Matrices and for a recursive Newton-Euler inverse dynamics algorithm with this parametrization. A comparison between these different methods is also provided.
- We created a MATLAB package with the tools to calculate the dqRNEA as well as the implementation of the dqRNEA itself (in recursive form, closed form, and matrix form).
- We added our dqRNEA function to a computed torque controller and tested for two different robot models. We also provide the description for these two models: the two-link planar arm and the Kuka robot.
- The mathematical definition for dual quaternion vectors and dual quaternions matrices, along with the main operations with these structures.

1.4 ORGANIZATION OF THIS MANUSCRIPT

This manuscript introduces, in Chapter 2 the main mathematic fundamentals used to construct the dqRNEA algorithm. There we show the basic formalism behind quaternion and dual quaternion algebra, as well as some more advanced proprieties, such as linear transformations within the dual quaternion group and the introduction of dual quaternion vectors and dual quaternion matrices. We also show, in this chapter, the geometric proprieties behind the dual quaternions.

In Chapter 3 we discuss the main topics on robotics that have served as a basis for this research. Mainly this chapter includes how to represent rigid body dynamics and kinematics with dual quaternions, as well as how to represent a the forward kinematics of a manipulator with dual quaternions—both with the Denavit–Hartenberg parameters and with the product of exponential formula.

Chapter 4 deals with the theory behind the inverse dynamics formulation and with the derivation of our dqRNEA algorithm. Furthermore, we also propose here the quaternionic-Matrix formulation of the Newton-Euler inverse dynamics, as well as the closed form of the algorithm. Finally, Chapter 4 also describes the computer torque control scheme for our algorithm.

Chapter 5 is mainly concerned with the cost analysis of the algorithm. It shows the costs for the main operations with quaternions and with dual quaternions as well as the dqRNEA algorithm. Moreover, in this chapter we also proposed our optimized version of the adjoint matrix and analyze its costs. Finally, Chapter 5 also compared the results of the dual quaternion based representation with the costs for a similar algorithm but with homogeneous transformation matrix (HTM) instead.

Chapter 6 shows the simulation results for our algorithm. Here we provide some remarks of how we implemented the algorithm as well as a description of the MATLAB functions we have used. More so, this chapter shows the results for the dqRNEA and for a dqRNEA based controller for two different robot models: a two-link planar arm and the Kuka LWR4.

Chapter 7 provides the concluding remarks as well as the direction for future works.

Finally, Appendix A shows some of the theory on the homogeneous transformation matrices and it presents a version of the recursive Newton-Euler algorithm that uses this parametrization. There we also provide a cost analysis for the algorithm.

2

MATHEMATIC FUNDAMENTALS

Throughout this manuscript we aim to introduce a novel description for the inverse dynamics of a serial robotic manipulator by taking advantage of the dual quaternion algebra. In order to start constructing our algorithm we must first introduce the main mathematical tools that we will be using for the rest of our work. In this manner this Chapter lays the theoretical foundation of our work by introducing both the quaternion and dual quaternion algebra, and, more so, by describing some of its main properties.

Thus, 2.1 starts describing the basic concepts behind the quaternions, dual numbers and dual quaternions and then further describes some more advanced properties and its advantages. Furthermore, in Section 2.2 we extend the geometric notion of the both the quaternions and dual quaternions in order to describe points, lines and curves in space.

2.1 DUAL QUATERNION REPRESENTATION

In a letter written by a famous 19th century mathematician to his son, it was said,

Every morning in the early part of October 1843, on my coming down to breakfast, your brother and yourself used to ask me: "Well, Papa, can you multiply triples?" Where to I was always obliged to reply, with a sad shake of the head, "No, I can only add and subtract them."

The mathematician in question was William Hamilton, who, for many years, sought a way to represent points in space by a new numeric system: the triplets. This system of hypercomplex numbers composed of one real and two imaginary components should represent tridimensional space similarly to how complex numbers represented points on a two dimensional plane. This endeavor, however, was impossible as multiplication properties would not hold in this systems and as was later stated by Georg Frobenius¹,

Proposition 2.1. *There exists no three-dimensional algebra over the field of real numbers that extends the complex numbers.*

In order to fulfill his goal, Hamilton insightfully added another dimension to his system, thus defining the Quaternions. Although at first his discovery was met with quite a lot of skepticism, as it was not usual at Hamilton's time to study a non-commutative algebra, the quaternions today are regarded as very powerful tools used in a range of disciplines, from engineering to computer science, and even in quantum physics! With some arguing that quaternions have "it opened the floodgates of modern abstract algebra" [62].

Although the quaternions are very useful to a number of applications—especially to represent rotations in space—they do have a few limitations. Mainly, quaternions do not couple together the rotation and translation of a body. A solution to this problem was introduced in 1873 by William Clifford who extended a structure known as biquaternions over to the dual numbers, and thus created the dual quaternions(DQ) [63]. This new structure would take advantage of the dual numbers in order to combine two quaternions into the eight parameter algebra, which can, in its unit form, represent solve the problem of representing both a translation and a

¹A full proof of Frobenius Theorem can be found at [61]

rotation in space.

Throughout the remainder of this manuscript we shall be using many concepts of both quaternion and dual quaternion algebra. In this manner, the remainder of this section will focus on formally defining the Quaternions, Dual Numbers and Dual Quaternions as well as discussing some its proprieties. Thus, Subsection 2.1.1 introduces the quaternion algebra, Subsection 2.1.2 the Dual numbers and Subsection 2.1.3 the dual quaternions. Moreover, we find it necessary to also discuss some other aspects of dual quaternion algebra not covered in Subsections 2.1.1 to 2.1.3, and in this manner we shall discuss how we may perform linear transformations with dual quaternions without leaving the group in Subsection 2.1.4 and in Subsection 2.1.5 we shall introduce dual quaternion vectors and matrices. Finally, in Subsection 2.1.6 we will be highlighting the main advantages of using dual quaternion algebra in robotics.

2.1.1 Quaternions

In this section we shall introduce the algebra of quaternions and some of their main proprieties and particularities. As discussed in this section's introduction, the set of quaternions was devised to be an extension of the algebra of complex numbers, but we can also see the complex numbers as a subset of the quaternions ($\mathbb{C} \subset \mathbb{H}$). This assertion leads to the intuitive notion that quaternions and complex numbers many times operate in a similar fashion, such is the case for the addition/subtraction and multiplication operations. In the following we formally define the first fundamental building block of this manuscript, the quaternions.

Definition 2.1. (Quaternions) [64] The quaternion algebra is composed by the set of linear combinations on elements $1, \hat{i}, \hat{j}$ and \hat{k}

$$\mathbb{H} \triangleq \left\{ \eta + \boldsymbol{\mu} : \boldsymbol{\mu} = \mu_1 \hat{i} + \mu_2 \hat{j} + \mu_3 \hat{k}, \eta, \mu_1, \mu_2, \mu_3 \in \mathbb{R} \right\}. \quad (2.1)$$

with the following proprieties for any quaternions $\mathbf{q} = \eta + \hat{i}\mu_1 + \hat{j}\mu_2 + \hat{k}\mu_3$ and $\mathbf{q}' = \eta' + \hat{i}\mu'_1 + \hat{j}\mu'_2 + \hat{k}\mu'_3$ and $\alpha \in \mathbb{R}$.

Propriety 2.1. (Quaternion Equality)

The quaternion equality is defined by the equality of each of its elements, that is

$$\mathbf{q} = \mathbf{q}' \iff \eta = \eta', \mu_1 = \mu'_1, \mu_2 = \mu'_2, \mu_3 = \mu'_3 \quad (2.2)$$

Propriety 2.2. (Quaternion Addition/Subtraction)

The quaternion addition operation $\mathbb{H} \times \mathbb{H} \mapsto \mathbb{H}$ is defined as

$$\mathbf{q} + \mathbf{q}' = (\eta + \eta') + \hat{i}(\mu_1 + \mu'_1) + \hat{j}(\mu_2 + \mu'_2) + \hat{k}(\mu_3 + \mu'_3) \quad (2.3)$$

Propriety 2.3. (Null Elements)

There exists a null element for addition/subtraction denoted as $\mathbf{0}$ such that $\mathbf{0} + \mathbf{q} = \mathbf{q} + \mathbf{0} = \mathbf{q}$.

Propriety 2.4. (Scalar Multiplication)

The multiplication of a real number by a quaternion $\mathbb{R} \times \mathbb{H} \mapsto \mathbb{H}$ is a commutative operation defined as,

$$\alpha \mathbf{q} = \mathbf{q} \alpha = \alpha \eta + \hat{i}(\alpha \mu_1) + \hat{j}(\alpha \mu_2) + \hat{k}(\alpha \mu_3). \quad (2.4)$$

Proprieties 2.1-2.4 yields that $(\mathbb{H}, +, \mathbb{R})$ is a real vector space, that is

$$\forall (\mathbf{q}, \mathbf{q}' \in \mathbb{H}) \quad \alpha \mathbf{q} + \beta \mathbf{q}' \in \mathbb{H} \text{ for any } \alpha, \beta \in \mathbb{R}.$$

Propriety 2.5. (Quaternion Multiplication)

The quaternion multiplication operation $\mathbb{H} \times \mathbb{H} \mapsto \mathbb{H}$ is

$$\begin{aligned} \mathbf{q} \mathbf{q}' = & (\eta \eta' - \mu_1 \mu'_1 - \mu_2 \mu'_2 - \mu_3 \mu'_3) + \hat{i}(\eta' \mu_1 + \mu_1 \eta' + \mu_2 \mu'_3 - \mu_3 \mu'_2) + \\ & \hat{j}(\eta \mu'_2 - \mu_1 \mu'_3 + \mu_2 \eta' + \mu_3 \mu'_1) + \hat{k}(\eta \mu'_3 + \mu_1 \mu'_2 - \mu_2 \mu'_1 + \mu_3 \eta'). \end{aligned} \quad (2.5)$$

Propriety 2.6. (Identity and Null Elements)

There exists a neutral element of a quaternion multiplication denoted as $\mathbf{1} \in \mathbb{H}$ such that $\mathbf{1} \mathbf{q} = \mathbf{q} \mathbf{1} = \mathbf{q}$.

Stemming from proprieties 2.1-2.6 we have that $(\mathbb{H}, +, \cdot)$ is an algebra.

Remark 2.1. Observing that for $1 \in \mathbb{R}$ the scalar multiplication yields $1 \mathbf{q} = \mathbf{q} 1 = \mathbf{q}$, it is natural to identify $\mathbf{1} = 1$.

Remark 2.2. From definition, it follows that the multiplication operation over \mathbb{H} is both associative and distributive, however it is non-commutative, that is $\mathbf{q} \mathbf{q}' \neq \mathbf{q}' \mathbf{q}$.

Remark 2.3. Also it follows that the quaternionic elements $\hat{i}, \hat{j}, \hat{k}$ satisfy

$$\hat{i} \hat{j} = \hat{k}, \hat{j} \hat{k} = \hat{i}, \text{ and } \hat{k} \hat{i} = \hat{j}, \text{ but } \hat{j} \hat{i} = -\hat{k}, \hat{k} \hat{j} = -\hat{i} \text{ and } \hat{i} \hat{k} = -\hat{j}.$$

In fact, these proprieties are used as an alternative way to define the multiplication itself.

Furthermore, quaternions may also be divided with respect to their real and imaginary parts [28], that is, given $\mathbf{q} = \eta + \mu_1 \hat{i} + \mu_2 \hat{j} + \mu_3 \hat{k}$, we have

$$\text{Re}(\mathbf{q}) \triangleq \eta, \quad (2.6)$$

$$\text{Im}(\mathbf{q}) \triangleq \mu_1 \hat{i} + \mu_2 \hat{j} + \mu_3 \hat{k}. \quad (2.7)$$

These operations allow us to rewrite the quaternion as $\mathbf{q} = \text{Re}(\mathbf{q}) + \text{Im}(\mathbf{q})$, and consequently, we can easily define the conjugate of a quaternion,

$$\mathbf{q}^* \triangleq \text{Re}(\mathbf{q}) - \text{Im}(\mathbf{q}). \quad (2.8)$$

One of the most important reasons for having the conjugate operation is to further define the quaternion square norm and the inverse operation.

Definition 2.2. (Square Norm) [28, 37, 45]

Let $\mathbf{q} \in \mathbb{H}$, then we may define the quaternion's norm as

$$\|\mathbf{q}\| \triangleq \sqrt{\mathbf{q}\mathbf{q}^*}. \quad (2.9)$$

Remark 2.4. We also highlight here that the norm of quaternion elements coincides with the Euclidean norm of a vector $\text{vec } \mathbf{q} \triangleq [\eta \ \mu_1 \ \mu_2 \ \mu_3]^T \in \mathbb{R}^4$, that is, $\|\mathbf{q}\| = \|\text{vec } \mathbf{q}\|$.

From the quaternion norm of a non zero quaternion at (2.9) we may easily derive the inverse operation, \mathbf{q}^{-1} . Starting from the definition that a quaternion multiplied by its inverse yields the unit element we have,

$$\begin{aligned} \mathbf{q}\mathbf{q}^{-1} &= 1 = \mathbf{q}^{-1}\mathbf{q}, \\ \mathbf{q}^*\mathbf{q}\mathbf{q}^{-1} &= \mathbf{q}^*. \end{aligned} \quad (2.10)$$

From (2.10) and (2.9) we may easily define the quaternion inverse, as is done in Definition 2.3.

Definition 2.3. (Inverse Operation) [28, 37, 45] Let $\mathbf{q} \in \mathbb{H}$, then the inverse operation may be defined as

$$\mathbf{q}^{-1} = \frac{\mathbf{q}^*}{\|\mathbf{q}\|^2}. \quad (2.11)$$

In the following, we will explore some subsets of the quaternions which are widely used and/or important to the proper comprehension of the remainder of this manuscript. Particularly, we will introduce the manifold that defines the group of unit quaternions, which can be used to define the rotation operation and the attitude representation in Subsection (2.2.1).

Complex Numbers

The complex numbers [65], although formulated before the quaternions, may be seen as one of its subsets. Indeed, if for a quaternion $\mathbf{q} = \eta + \hat{i}\mu_1 + \hat{j}\mu_2 + \hat{k}\mu_3$ we have μ_2 and μ_3 equal to zero, then $\mathbf{q}_{\hat{i}} = \eta + \hat{i}\mu_1$. Thus, we have that

$$\mathbb{C}_{\hat{i}} \triangleq \{\eta + \boldsymbol{\mu} : \boldsymbol{\mu} = \mu_1 \hat{i}, \eta, \mu_1 \in \mathbb{R}\}.$$

We highlight that, alternatively, we may have μ_1 and μ_3 equal to zero and $\mathbf{q}_{\hat{j}} = \eta + \hat{j}\mu_2$ or μ_1 and μ_2 equal to zero and $\mathbf{q}_{\hat{k}} = \eta + \hat{k}\mu_3$. The complex set for these cases would then be defined, respectively, as

$$\mathbb{C}_{\hat{j}} \triangleq \{\eta + \boldsymbol{\mu} : \boldsymbol{\mu} = \mu_2 \hat{j}, \eta, \mu_2 \in \mathbb{R}\}$$

or

$$\mathbb{C}_{\hat{k}} \triangleq \{\eta + \boldsymbol{\mu} : \boldsymbol{\mu} = \mu_3 \hat{k}, \eta, \mu_3 \in \mathbb{R}\}.$$

We will not detail the set of complex numbers here, however, its of easy verification that many of the proprieties for the quaternions hold for the complexes as well. We highlight here, however, that due to the lower number of dimensions \mathbb{C} becomes commutative under multiplication.

Pure Imaginary Quaternions

The Pure Imaginary Quaternions [28, 37, 66] (or pure quaternions) are the subset of quaternions in which the real part is null. That is,

$$\mathbb{H}_0 \triangleq \{\mathbf{q}_0 : \mathbf{q}_0 \in \mathbb{H}, \text{Re}(\mathbf{q}_0) = 0\}.$$

In terms of the conjugate operation, we have that pure quaternions can also be defined as the set of $\mathbf{q}_0 \in \mathbb{H}$ satisfying

$$\mathbf{q}_0^* = -\mathbf{q}_0. \quad (2.12)$$

The elements of this subset are isomorphic to the three-dimensional vectors² \mathbb{R}^3 , something that becomes particularly useful when representing dynamic and kinematic variables (translation, angular and linear velocities, accelerations, momentum and wrenches). Indeed the pure quaternions are very well suited to represent compactly an unified framework for robotics, and this will be further explored in Chapter 3. Here we will explore a little more some of the mathematical proprieties that relate \mathbb{H}_0 and \mathbb{R}^3 , mainly the inner and cross product as defined in [28, 66]

Definition 2.4. (Cross Product) [28, 37, 66] Given two pure quaternions $\mathbf{q} = \hat{i}\mu_1 + \hat{j}\mu_2 + \hat{k}\mu_3$ and $\mathbf{q}' = \hat{i}\mu'_1 + \hat{j}\mu'_2 + \hat{k}\mu'_3$ the cross product between two pure quaternions is

$$\mathbf{q} \times \mathbf{q}' \triangleq \frac{\mathbf{q}\mathbf{q}' - \mathbf{q}'\mathbf{q}}{2}. \quad (2.13)$$

Definition 2.5. (Inner Product) [28, 37, 66] Given two pure quaternions $\mathbf{q} = \hat{i}\mu_1 + \hat{j}\mu_2 + \hat{k}\mu_3$ and $\mathbf{q}' = \hat{i}\mu'_1 + \hat{j}\mu'_2 + \hat{k}\mu'_3$, the inner product is defined as

$$\mathbf{q} \cdot \mathbf{q}' \triangleq -\frac{\mathbf{q}\mathbf{q}' + \mathbf{q}'\mathbf{q}}{2} = \mu_1\mu'_1 + \mu_2\mu'_2 + \mu_3\mu'_3. \quad (2.14)$$

From definitions 2.5 and 2.4 we can redefine the multiplication of pure quaternions. That is, if $\mathbf{q}, \mathbf{q}' \in \mathbb{H}_0$, then from (2.5) we have

$$\mathbf{q}\mathbf{q}' = -\mathbf{q} \cdot \mathbf{q}' + \mathbf{q} \times \mathbf{q}'. \quad (2.15)$$

And, furthermore, if the quaternions are perpendicular to each other, than the inner product is zero and $\mathbf{q}\mathbf{q}' = -\mathbf{q}\mathbf{q}' = \mathbf{q} \times \mathbf{q}'$. This result is once more analogous to how complex numbers represent motions in the plane (see Section 2.2), thus furthering the notion that quaternions can represent a motion in space.

Unit Quaternions

The subset of unit quaternions [28, 37, 38, 66] is defined as

$$\mathcal{S}^3 \triangleq \{\mathbf{q} \in \mathbb{H} : \|\mathbf{q}\| = 1\}. \quad (2.16)$$

That is, the unit quaternion is the subset of quaternions corresponding to the three dimensional unit sphere in \mathbb{R}^4 with norm equal to one. This set, when equipped with multiplication, forms the Lie group Spin(3) [67], formally defined as follows.

²See Subsection 2.1.1.1

Definition 2.6. (Unit Quaternion Group)

The unit quaternion group, $\text{Spin}(3)$, is defined by the set \mathcal{S}^3 equipped with the quaternion multiplication. Given the quaternions $\mathbf{q}_1, \mathbf{q}_2$ and $\mathbf{q}_3 \in \text{Spin}(3)$ the multiplication satisfies the following axioms:

1. (Closure) The multiplication of any pair of unit quaternions in $\text{Spin}(3)$ results in a unit quaternion in the group, thus $\mathbf{q}_1 \mathbf{q}_2 \in \text{Spin}(3)$.
2. (Identity) $\exists \mathbf{1} \triangleq 1 \in \mathcal{S}^3$ such that $\mathbf{q}\mathbf{1} = \mathbf{1}\mathbf{q} = \mathbf{q}$;
3. (Inverse) The inverse operation exists and may be defined from (2.11) and (2.16). That is, considering the unit norm of this group of quaternions, (2.11) becomes

$$\mathbf{q}^{-1} = \mathbf{q}^*; \quad (2.17)$$

4. (Associative) $\mathbf{q}_1(\mathbf{q}_2\mathbf{q}_3) = (\mathbf{q}_1\mathbf{q}_2)\mathbf{q}_3$;

In particular $\text{Spin}(3)$ is extremely important when we are describing a rigid body's attitude, usually being comparable to the Euler Angles and Rotations Matrices in its purpose. The remainder of this manuscript will tackle related issues, such as the use of unit quaternions for frame transformation and point transformation (see subsection (2.2.1)) and rigid body motion (see (3.1.2)).

2.1.1.1 Isomorphism to \mathbb{R}^4 and \mathbb{R}^3

Another important quality of the quaternions is that they are isomorphic to \mathbb{R}^4 , or, in the case of pure quaternions, are isomorphic to \mathbb{R}^3 . What this means is that we are able to define an isomorphic map between the two groups—that is, we may perform a one-to-one transformation between those groups such that the operations of the original group will be respected [68].

This is an important definition, which allows some important results relating multiplication and commutation as will be further discussed in Subsection 2.1.4. However, still in regards to the isomorphism between quaternions and 4-dimensional vectors, we believe it is important to introduce here the *vector* operator: $\text{vec}_4 : \mathbb{H} \longrightarrow \mathbb{R}^4$. This is an one-by-one map between quaternions and vector, such that, for a quaternion $\mathbf{q} = \eta + \hat{i}\mu_1 + \hat{j}\mu_2 + \hat{k}\mu_3$, we have

$$\text{vec}_4 \mathbf{q} \triangleq \begin{bmatrix} \eta \\ \mu_1 \\ \mu_2 \\ \mu_3 \end{bmatrix}, \quad (2.18)$$

and similarly the inverse operation is defined as $\underline{\text{vec}}_4 : \mathbb{R}^4 \longrightarrow \mathbb{H}$ [37, 66].

Similarly, we may also consider the isomorphism between \mathbb{H}_0 and \mathbb{R}^3 , which in this case may be defined by means of the operator $\text{vec}_3 : \mathbb{H}_0 \longrightarrow \mathbb{R}^3$, such that for a pure quaternion $\mathbf{q}_0 = \hat{i}\mu_1 + \hat{j}\mu_2 + \hat{k}\mu_3$, we shall have

$$\text{vec}_3 \mathbf{q}_0 \triangleq \begin{bmatrix} \mu_1 \\ \mu_2 \\ \mu_3 \end{bmatrix}, \quad (2.19)$$

and its inverse map $\text{vec}_4 : \mathbb{R}^3 \rightarrow \mathbb{H}_0$. We note that the pure quaternions' isomorphism to \mathbb{R}^3 is particularly interesting, as both the cross and inner products with quaternions are analogous to their counterpart in \mathbb{R}^3 [66]. Moreover, as a consequence of this, we may use \mathbb{H}_0 to represent positions and translations in space also in a similar fashion to the real set.

2.1.2 Dual Numbers

The dual numbers are usually defined as motivators for the dual quaternions, however, they may be also seen as a its subset, in which the the imaginary part of the dual quaternion is equal to zero. Formally they may be defined as

$$\mathbb{D} \triangleq \{\underline{\mu} = \mu + \varepsilon\mu' : \mu, \mu' \in \mathbb{R}\}, \quad (2.20)$$

in which ε is the dual unit, and $\varepsilon^2 = 0$, $\varepsilon \neq 0$ [69].

As dual numbers share many proprieties with the dual quaternions, we shall not describe them in detail here. However, we find it important to show here the Taylor Expansion of a dual number.

Definition 2.7. (Dual Number Taylor Expansion) [1, 37] Given the dual number $\underline{\mu} = \mu + \varepsilon\mu' \in \mathbb{D}$, the differentiable function $f : \mathbb{D} \rightarrow \mathbb{D}$ can be defined by its Taylor Expansion at the point $\underline{\mu}_0 = \mu$. That is

$$\begin{aligned} f(\underline{\mu}) &= \sum_{n=0}^{\infty} \frac{f^{(n)}(\underline{\mu})}{n!} (\underline{\mu} - \underline{\mu}_0)^n \\ &= f(\underline{\mu}_0) + f'(\underline{\mu}_0) (\underline{\mu} - \underline{\mu}_0) + \frac{f''(\underline{\mu}_0)}{2} (\underline{\mu} - \underline{\mu}_0)^2 + \dots \end{aligned}$$

As $\varepsilon^2 = 0$, then the term $(\underline{\mu} - \underline{\mu}_0)^n = (\varepsilon\mu')^n$ is equal to zero for any $n \geq 2$. That results in

$$f(\underline{\mu}) = f(\underline{\mu}_0) + \varepsilon f'(\underline{\mu}_0) \mu' \quad (2.21)$$

Finally, we may also decompose the dual numbers in two different parts: the primary part and the dual part. Considering the dual number $\underline{\mu} = \mu + \varepsilon\mu'$ these parts can be retrieved by means of the primary part operator,

$$\mathcal{P}(\underline{\mu}) = \mu, \quad (2.22)$$

and the dual part operator

$$\mathcal{D}(\underline{\mu}) = \mu'. \quad (2.23)$$

2.1.3 Dual Quaternions

The dual quaternions were developed aiming a more compact and efficient representation for motions in space. In fact, the quaternions by themselves are very good to represent motions, however, they are limited to describing only a translation or a rotation at a time. Thus, we introduce the set of dual quaternions [28, 37, 66, 70, 71], which take advantage of the dual numbers to couple together two quaternions and represent the complete motion (or pose) of an object,

$$\mathbb{H} \triangleq \{\underline{q} = q_P + \varepsilon q_D \mid q_P, q_D \in \mathbb{H}\}. \quad (2.24)$$

As (2.24) makes clear, the dual quaternions expand the basis in \mathbb{H} to $\{1, \hat{i}, \hat{j}, \hat{k}, \varepsilon, \varepsilon\hat{i}, \varepsilon\hat{j}, \varepsilon\hat{k}\}$ and its dimensions to eight and also ε is called dual unit with $\varepsilon^2 = 0, \varepsilon \neq 0$. Indeed, similarly to how quaternions expand on some proprieties of complex numbers, the dual quaternions expands the quaternions.

Another propriety of dual quaternions is that its elements can be decomposed in primary and dual parts. That is, for a dual quaternion $\underline{q} = \underline{q}_P + \varepsilon \underline{q}_D$, we can define the primary part operator [28],

$$\mathcal{P}(\underline{q}) = \underline{q}_P \quad (2.25)$$

and the dual part operator

$$\mathcal{D}(\underline{q}) = \underline{q}_D. \quad (2.26)$$

From the operation defined in subsection 2.1.1 and by using (2.25) and (2.26) we may formally define the dual quaternion algebra.

Definition 2.8. (Dual Quaternion Algebra) [37, 63] The dual quaternion algebra is composed by the set

$$\mathbb{H} \triangleq \{\underline{q} = \underline{q}_P + \varepsilon \underline{q}_D \mid \underline{q}_P, \underline{q}_D \in \mathbb{H}\} \quad (2.27)$$

with the following properties for any \underline{q} and $\underline{q}' \in \mathbb{H}$ and $\alpha \in \mathbb{R}$.

Propriety 2.7. (Dual Quaternion Equality)

The dual quaternion equality is defined by the equality of each of its elements, that is

$$\underline{q} = \underline{q}' \iff \mathcal{P}(\underline{q}) = \mathcal{P}(\underline{q}'), \mathcal{D}(\underline{q}) = \mathcal{D}(\underline{q}') \quad (2.28)$$

Propriety 2.8. (Dual Quaternion Addition/Subtraction)

Dual quaternion addition/subtraction is defined as the operation $\pm : \mathbb{H} \times \mathbb{H} \mapsto \mathbb{H}$ is defined as

$$\underline{q} \pm \underline{q}' = (\mathcal{P}(\underline{q}) \pm \mathcal{P}(\underline{q}')) + \varepsilon (\mathcal{D}(\underline{q}) \pm \mathcal{D}(\underline{q}')). \quad (2.29)$$

Propriety 2.9. (Dual Quaternion Null Element)

There exists a null element for addition/subtraction denoted as $\underline{0}$ such that $\underline{0} + \underline{q} = \underline{q} + \underline{0} = \underline{q}$.

Propriety 2.10. (Scalar Multiplication)

The multiplication of a real number by a dual quaternion $\mathbb{R} \times \mathbb{H} \mapsto \mathbb{H}$ is a commutative operation defined as,

$$\alpha \underline{q} = \alpha \mathcal{P}(\underline{q}) + \varepsilon \mathcal{D}(\underline{q}) \quad (2.30)$$

Proprieties 2.7-2.10 yields that $(\mathbb{H}, +, \mathbb{R})$ is a real vector space, that is

$$\forall (\underline{q}, \underline{q}' \in \mathbb{H}) \quad \alpha \underline{q} + \beta \underline{q}' \in \mathbb{H} \text{ for any } \alpha, \beta \in \mathbb{R}.$$

Propriety 2.11. (Dual Quaternion Multiplication)

The dual quaternion multiplication operation $\mathbb{H} \times \mathbb{H} \mapsto \mathbb{H}$ takes into account the $\varepsilon^2 = 0$ propriety of the dual numbers, thus being

$$\underline{q}\underline{q}' = \mathcal{P}(\underline{q})\mathcal{P}(\underline{q}') + \varepsilon (\mathcal{P}(\underline{q})\mathcal{D}(\underline{q}') + \mathcal{D}(\underline{q})\mathcal{P}(\underline{q}')) \quad (2.31)$$

Propriety 2.12. (Identity and Null Elements)

There exists a neutral element of a quaternion multiplication denoted as $\underline{\mathbf{1}} \in \mathbb{H}$ such that $\underline{\mathbf{1}}\underline{\mathbf{q}} = \underline{\mathbf{q}}\underline{\mathbf{1}} = \underline{\mathbf{q}}$. Stemming from proprieties 2.7-2.12 we have that $(\mathbb{H}, +, \cdot)$ is an algebra.

Remark 2.5. For dual quaternions is still important to highlight that multiplication is both associative and distributive, but non-commutative, as is the case of the quaternions. However, we also highlight that in contrast to quaternion algebra, due to the presence of the dual number ε , the dual quaternion algebra is not a division algebra. Indeed, as will be discussed in Definition (2.10) there is a subset of non-null \mathbb{H} without a inverse definition.

We may also divide dual quaternions into their real and imaginary parts bay using the $Re(*)$ and $Im(*)$ operators. For a dual quaternion $\underline{\mathbf{q}} \in \mathbb{H}$, the real and imaginary parts are respectively given by

$$Re(\underline{\mathbf{q}}) \triangleq Re(\mathcal{P}(\underline{\mathbf{q}})) + \varepsilon Re(\mathcal{D}(\underline{\mathbf{q}})) \quad (2.32)$$

and

$$Im(\underline{\mathbf{q}}) \triangleq Im(\mathcal{P}(\underline{\mathbf{q}})) + \varepsilon Im(\mathcal{D}(\underline{\mathbf{q}})). \quad (2.33)$$

From (2.8), (2.32) and (2.33) we may also introduce the dual quaternion conjugate operation. Let $\underline{\mathbf{q}} = \mathcal{P}(\underline{\mathbf{q}}) + \varepsilon\mathcal{D}(\underline{\mathbf{q}})$ be a dual quaternion, then

$$\underline{\mathbf{q}}^* \triangleq \mathcal{P}(\underline{\mathbf{q}})^* + \varepsilon\mathcal{D}(\underline{\mathbf{q}})^* \triangleq Re(\underline{\mathbf{q}}) - Im(\underline{\mathbf{q}}). \quad (2.34)$$

As can be seen, the conjugate of a dual quaternion can be defined both as the conjugate of the quaternions representing the primary and dual parts, or as the real part minus the imaginary one. Furthermore, we can use (2.34) to define the norm and inverse operations.

Let $\underline{\mathbf{q}} \in \mathbb{H}$, then we may define the dual quaternion's square semi-norm as well as its inverse.

Definition 2.9. (Square Semi-norm) [66] The dual quaternion semi-norm can be defined as

$$\|\underline{\mathbf{q}}\| = \sqrt{\underline{\mathbf{q}}^*\underline{\mathbf{q}}} = \sqrt{\underline{\mathbf{q}}\underline{\mathbf{q}}^*}. \quad (2.35)$$

Definition 2.10. (Inverse Operation) [67] Similarly to definition (2.3), the dual quaternion's inverse may be defined as

$$\underline{\mathbf{q}}^{-1} = \frac{\underline{\mathbf{q}}^*}{\|\underline{\mathbf{q}}\|^2} \quad (2.36)$$

and only exists if $\mathcal{P}(\underline{\mathbf{q}}) \neq 0$ or, in other words, if $\|\underline{\mathbf{q}}\| \neq 0^a$.

^aThe algebra of dual quaternions is a eight dimensional associative and not commutative algebra over \mathbb{R}^8 , however, as there are divisors of zero, it does not form a division ring. The quaternions, on the other hand, are a division algebra over \mathbb{R}^4 and the octonions are a division algebra over \mathbb{R}^8 .

Remark 2.6. We notice here that this is not strictly a square norm, in which we would expect positive values for all non-zero elements. For the particular case of $\underline{\mathbf{q}} = \varepsilon\mathcal{D}(\underline{\mathbf{q}})$ the norm will be zero for any element. Thus we define this as the semi-norm rather than as a traditional norm.

Another important aspect of the dual quaternion semi-norm is that, differently from the quaternion norm, its result does not coincide with the euclidian norm, but rather it yields a dual number. This becomes clearer

once we rewrite (2.35) as

$$\begin{aligned}
\|\underline{\mathbf{q}}\| &= \sqrt{\underline{\mathbf{q}}^* \underline{\mathbf{q}}} \\
&= \sqrt{[\mathcal{P}(\underline{\mathbf{q}})^* \mathcal{P}(\underline{\mathbf{q}})] + \varepsilon [\mathcal{P}(\underline{\mathbf{q}})^* \mathcal{D}(\underline{\mathbf{q}}) + \mathcal{D}(\underline{\mathbf{q}})^* \mathcal{P}(\underline{\mathbf{q}})]} \\
&= \sqrt{\|\mathcal{P}(\underline{\mathbf{q}})\|^2 + \varepsilon 2 (\mathcal{P}(\underline{\mathbf{q}}) \cdot \mathcal{D}(\underline{\mathbf{q}}))} \\
&= \sqrt{\left(\|\mathcal{P}(\underline{\mathbf{q}})\| + \varepsilon \frac{1}{\|\mathcal{P}(\underline{\mathbf{q}})\|} (\mathcal{P}(\underline{\mathbf{q}}) \cdot \mathcal{D}(\underline{\mathbf{q}})) \right)^2} \\
&= \|\mathcal{P}(\underline{\mathbf{q}})\| + \varepsilon \frac{1}{\|\mathcal{P}(\underline{\mathbf{q}})\|} (\mathcal{P}(\underline{\mathbf{q}}) \cdot \mathcal{D}(\underline{\mathbf{q}})). \tag{2.37}
\end{aligned}$$

The dual quaternions also contains quite a few interesting subsets. In fact, the quaternion itself is a subset of dual quaternions ($\mathbb{R} \subset \mathbb{C} \subset \mathbb{H} \subset \underline{\mathbb{H}}$), as are the dual numbers ($\mathbb{D} \subset \underline{\mathbb{H}}$). However, we also have other subsets of interests, such as the dual numbers, pure dual quaternions and unit dual quaternions.

Pure Dual Quaternions

Much like the pure quaternions, the pure dual quaternions are the subset of quaternions with real part equal to zero [28, 37, 66]. Thus, we may define the subset as

$$\underline{\mathbb{H}}_0 \triangleq \{ \underline{\mathbf{q}}_0 = \mathbf{q}_P + \varepsilon \mathbf{q}_D \mid \mathbf{q}_P, \mathbf{q}_D \in \mathbb{H}_0 \}. \tag{2.38}$$

Moreover, as was the case for pure quaternions, the pure dual quaternions can be defined in terms of the conjugate operation. That is, the pure dual quaternions can be defined as the subset $\underline{\mathbf{q}}_0 \in \underline{\mathbb{H}}_0$ that satisfies

$$\underline{\mathbf{q}}_0^* = -\underline{\mathbf{q}}_0. \tag{2.39}$$

Similar to pure quaternions, the elements in $\underline{\mathbb{H}}_0$ can be deployed into the kinematics and dynamics analysis to compactly express the coupled angular and linear generalized rigid body twist, accelerations, momentum and wrenches [63]. In particular, the primary part of the dual quaternion will usually represent the angular part of the motion and the dual part will deal with the linear velocities, accelerations or forces. We can exploit $\underline{\mathbb{H}}_0$ to represent such physical qualities in a coupled manner due to the isomorphic relation they have with \mathbb{R}^6 ³, these equations of motion will be further explored in Chapter 3. Still, for the remainder of this section we shall explore some other mathematical proprieties of the pure dual quaternions, in particular, the cross and inner products, that, unlike the pure quaternions, do not coincide with their \mathbb{R}^6 counterparts.

Given two pure dual quaternions $\underline{\mathbf{q}} = \mathbf{q}_P + \varepsilon \mathbf{q}_D$ and $\underline{\mathbf{q}}' = \mathbf{q}'_P + \varepsilon \mathbf{q}'_D$

Definition 2.11. (Cross Product) [28] The cross product between two pure dual quaternions is defined in [37] as

$$\underline{\mathbf{q}} \times \underline{\mathbf{q}}' \triangleq \frac{\underline{\mathbf{q}} \underline{\mathbf{q}}' - \underline{\mathbf{q}}' \underline{\mathbf{q}}}{2}. \tag{2.40}$$

³See Subsection 2.1.3.1

Definition 2.12. (Inner Product) [28] The pure dual quaternion inner product is defined in [37] as

$$\underline{\mathbf{q}} \cdot \underline{\mathbf{q}}' \triangleq -\frac{\mathbf{q}\mathbf{q}' + \underline{\mathbf{q}}'\underline{\mathbf{q}}}{2}. \quad (2.41)$$

Remark 2.7. We highlight here that unlike the inner product of two vector in \mathbb{R}^6 and the inner product of two pure quaternions, the dual quaternion inner product does not result in a real scalar. It results in a dual number.

A dual number does not always provide us with results that posses clear physical meaning. In order to attain this we must define a new pure dual quaternion operation: the double geodesic product. This new product works by taking the combined real value of the inner product from the primary part and dual parts, as can be seen in the following Definition (2.13).

Definition 2.13. (Double Geodesic Product) Given two pure dual quaternions $\underline{\mathbf{q}} = \mathbf{q}_P + \varepsilon\mathbf{q}_D$ and $\underline{\mathbf{q}}' = \mathbf{q}'_P + \varepsilon\mathbf{q}'_D$ we define the double geodesic product as

$$\underline{\mathbf{q}} \odot \underline{\mathbf{q}}' \triangleq \mathbf{q}_P \cdot \mathbf{q}'_P + \mathbf{q}_D \cdot \mathbf{q}'_D. \quad (2.42)$$

Remark 2.8. .

Remark 2.9. This product is similar to the Euclidean norm of corresponding vectors and is similar to the double-geodesic metric provided in Bullo and Murray [72] for the rigid body transformations using the group of Euclidean displacements⁴.

Remark 2.10. We also remark here that, from (2.14), we have that the inner product between two quaternions is commutative. As a consequence of this, the individual terms of (2.42) are also commutative, and $\underline{\mathbf{q}} \odot \underline{\mathbf{q}}' = \underline{\mathbf{q}}' \odot \underline{\mathbf{q}}$

Unit Dual Quaternions

Once more, analogous to unit quaternions, the subset of unit dual quaternions is the one with unit semi-norm

$$\underline{\mathcal{S}} \triangleq \{ \underline{\mathbf{q}} \in \mathbb{H} : \|\underline{\mathbf{q}}\| = 1 \}. \quad (2.43)$$

We stress here that (2.37) implies that $\|\underline{\mathbf{q}}\| = 1$ if and only if $\|\mathcal{P}(\underline{\mathbf{q}})\| = 1$ and $\mathcal{P}(\underline{\mathbf{q}}) \cdot \mathcal{D}(\underline{\mathbf{q}}) = 0$ ⁵. Furthermore, under multiplication the set of unit dual quaternions forms the Lie group $\text{Spin}(3) \times \mathbb{R}^3$ [67].

⁴It is important to highlight that there is no well-defined Riemannian metric for the group of Euclidean transformations nor for the (pure) dual quaternions, but the double-geodesic approach used in [72] and herein ensures positiveness and equal actions in the attitude and translation geodesics. The study on metrics for Euclidean displacements and the correspondent topological obstruction lies out of the scope of the current manuscript, but readers are referred to excel the works [72, 73] for further details.

⁵One interesting consequence of the unit dual quaternion group is that it no longer has a semi-norm, but a regular norm. Indeed all elements have a clearly defined norm, which is $\|\underline{\mathbf{q}}\| = 1$.

Definition 2.14. (Unit Dual Quaternion Group) [66,67]

The unit dual quaternion group, $\text{Spin}(3) \times \mathbb{R}^3$, is defined if the quaternions \underline{q}_1 , \underline{q}_2 and $\underline{q}_3 \in \text{Spin}(3) \times \mathbb{R}^3$ have the following proprieties:

1. (Closure) The multiplication of any pair of unit quaternions in $\text{Spin}(3) \times \mathbb{R}^3$ results in a unit dual quaternion in the group, thus $\underline{q}_1 \underline{q}_2 \in \text{Spin}(3) \times \mathbb{R}^3$.
2. (Identity) For all unit dual quaternion in $\text{Spin}(3) \times \mathbb{R}^3$ the identity element is defined as $\underline{1} \triangleq 1$ and $\underline{q} \underline{1} = \underline{1} \underline{q} = \underline{q}$.
3. (Inverse) The inverse operation exists and may be defined from (2.36) and (2.43). That is, considering the unit semi-norm, (2.36) becomes

$$\underline{q}^{-1} = \underline{q}^*. \quad (2.44)$$

4. (Associative) The associative propriety holds $\underline{q}_1 (\underline{q}_2 \underline{q}_3) = (\underline{q}_1 \underline{q}_2) \underline{q}_3$

In the same manner the unit quaternions describe a body's attitude, the unit dual quaternions may describe the complete pose of an object, coupling translation and rotation operations. Indeed, much like the homogeneous matrices (see Appendix (A)), the unit dual quaternions represent a complete rigid body motion, having a double cover over the Special Euclidean Group SE(3).

The remainder of this manuscript will explore the use of dual quaternions—including unit dual quaternions and pure dual quaternions—to represent the kinematics and dynamics of a body, particularly the dynamics of a serial manipulator.

2.1.3.1 Isomorphism to \mathbb{R}^8 and \mathbb{R}^6

In Subsection 2.1.1.1 we have discussed how the quaternions and pure quaternions are isomorphic to \mathbb{R}^4 and \mathbb{R}^3 , and how we may exploit this characteristic in order to define an operator that can map the two groups. This notion can also be extended to the dual quaternions and the pure dual quaternions and, as was the case in Subsection 2.1.1.1, the results this mapping yields are also important for expanding some proprieties of the dual quaternions and will also be discussed in depth in section 2.1.4.

Here, however, we will present here the dual quaternion equivalent for equation (2.18), the *vector* operator: $\text{vec}_8 : \mathbb{H} \rightarrow \mathbb{R}^8$. This is a one-by-one map between dual quaternions and vector, such as, for a dual quaternion $\underline{q} = \mathcal{P}(\underline{q}) + \varepsilon \mathcal{D}(\underline{q})$, we have

$$\text{vec}_8 \underline{q} \triangleq \begin{bmatrix} \text{vec}_4(\mathcal{P}(\underline{q})) \\ \text{vec}_4(\mathcal{D}(\underline{q})) \end{bmatrix}, \quad (2.45)$$

and similarly the inverse operation is defined as $\text{vec}_8 : \mathbb{R}^8 \rightarrow \mathbb{H}$.

Regarding the pure dual quaternions we have the operator $\text{vec}_6 : \mathbb{H}_0 \rightarrow \mathbb{R}^6$ which can map a pure quaternion such as $\underline{q}_0 = \mathcal{P}(\underline{q}_0) + \varepsilon \mathcal{D}(\underline{q}_0)$ to \mathbb{R}^6 through the function

$$\text{vec}_6 \underline{q}_0 \triangleq \begin{bmatrix} \text{vec}_3(\mathcal{P}(\underline{q}_0)) \\ \text{vec}_3(\mathcal{D}(\underline{q}_0)) \end{bmatrix}, \quad (2.46)$$

with the inverse mapping being defined as $\underline{\text{vec}}_6 : \mathbb{R}^6 \longrightarrow \mathbb{H}_0$ [28, 37, 66].

2.1.4 Linear Transformations

Whenever studying quaternion or dual quaternion algebra, the multiplication is always a delicate matter to be addressed. Besides the issues that may arise with the non-commutativity of the multiplication, stated in Definitions (2.5) and (2.11), we have also a need to properly address how linear transformations operate in this algebra, specially those concerning the quaternion-matrix multiplication.

Stemming from (2.18) we have a simple manner to multiply a quaternion by a matrix. That is, by exploiting the isomorphism between \mathbb{H} and \mathbb{R}^4 we may perform operations such

$$\mathbf{q}' = \underline{\text{vec}}_4 (M \text{vec}_4(\mathbf{q})), \quad (2.47)$$

in which \mathbf{q} and $\mathbf{q}' \in \mathbb{H}$ and $M \in \mathbb{R}^{4 \times 4}$.

Similarly we may also exploit (2.45) and the isomorphism of \mathbb{H} to \mathbb{R}^8 in order to multiply dual quaternions by matrices. Thus, let $\underline{\mathbf{q}}$ and $\underline{\mathbf{q}}' \in \mathbb{H}$ and $N \in \mathbb{R}^{8 \times 8}$, then DQ by matrix multiplication is

$$\underline{\mathbf{q}}' = \underline{\text{vec}}_8 (N \text{vec}_8(\underline{\mathbf{q}})). \quad (2.48)$$

Furthermore, the transformations given in (2.47) and (2.48) can be also expressed solely in quaternion (or dual quaternion) form.

Thus (2.47) may be rewritten as the operator \mathcal{T}_4 mapping an operation such as (2.47) to a quaternion without the need to go to \mathbb{R}^4 and $\mathbb{R}^{4 \times 4}$, yielding a matrix linear transformation such as

$$\mathcal{T}_4 (M) \mathbf{q} = \mathbf{m}_1 q_1 + \mathbf{m}_2 q_2 + \mathbf{m}_3 q_3 + \mathbf{m}_4 q_4 \quad (2.49)$$

with $M = \begin{bmatrix} \mathbf{m}_1 & \mathbf{m}_2 & \mathbf{m}_3 & \mathbf{m}_4 \end{bmatrix}$ being a vector of quaternions $\mathbf{m}_i \in \mathbb{H}$ and $\mathbf{q} = q_1 + q_2 \hat{i} + q_3 \hat{j} + q_4 \hat{k} \in \mathbb{H}$. In the same manner,

$$\mathcal{T}_8 (\underline{N}) \underline{\mathbf{q}} = \underline{\mathbf{n}}_1 q_1 + \underline{\mathbf{n}}_2 q_2 + \underline{\mathbf{n}}_3 q_3 + \underline{\mathbf{n}}_4 q_4 + \underline{\mathbf{n}}_5 q_5 + \underline{\mathbf{n}}_6 q_6 + \underline{\mathbf{n}}_7 q_7 + \underline{\mathbf{n}}_8 q_8 \quad (2.50)$$

in which $\underline{N} = \begin{bmatrix} \underline{\mathbf{n}}_1 & \underline{\mathbf{n}}_2 & \underline{\mathbf{n}}_3 & \underline{\mathbf{n}}_4 & \underline{\mathbf{n}}_5 & \underline{\mathbf{n}}_6 & \underline{\mathbf{n}}_7 & \underline{\mathbf{n}}_8 \end{bmatrix}$ being a vector of dual quaternions $\underline{\mathbf{n}}_i \in \mathbb{H}$ and $\underline{\mathbf{q}} = q_1 + q_2 \hat{i} + q_3 \hat{j} + q_4 \hat{k} + \varepsilon (q_5 + q_6 \hat{i} + q_7 \hat{j} + q_8 \hat{k}) \in \mathbb{H}$.

As previously discussed it is well-known that \mathbb{H} is non-commutative, and the same is valid for the dual quaternions. Indeed, as discussed in Definitions 2.5 and 2.11, if $\underline{\mathbf{a}}$ and $\underline{\mathbf{b}}$ are dual quaternions, then $\underline{\mathbf{a}}\underline{\mathbf{b}} \neq \underline{\mathbf{b}}\underline{\mathbf{a}}$. However, we may deal with this lack of commutativity with the approach proposed in [28, 37]. That is, by taking the matrix algebra representation of quaternions or dual quaternions.

In [74], the quaternion commutativity is obtained by changes in the sign of the resulting matrix representation which is isomorphic to the quaternion group. The resulting orthogonal matrix is known as the Hamilton operator $\overset{\pm}{\mathbf{H}}(x) \in \mathbb{R}^{4 \times 4}$ of the quaternion element x [70, 75]. Thus, from (2.47), two quaternions \mathbf{a} and \mathbf{b} can commute as follows

$$\mathbf{y} = \mathbf{a}\mathbf{b} = \underline{\text{vec}}_4 \left(\overset{+}{\mathbf{H}}(\mathbf{a}) \text{vec}_4 \mathbf{b} \right) = \underline{\text{vec}}_4 \left(\overset{-}{\mathbf{H}}(\mathbf{b}) \text{vec}_4 \mathbf{a} \right), \quad (2.51)$$

in which the Hamilton operators for the right and left multiplications can be computed from the matrix representation of a quaternion such as $\mathbf{x} = x_1 + x_2\hat{i} + x_3\hat{j} + x_4\hat{k}$,

$$\overset{+}{\mathbf{H}}(\mathbf{x}) \triangleq \begin{bmatrix} x_1 & -x_2 & -x_3 & -x_4 \\ x_2 & x_1 & -x_4 & x_3 \\ x_3 & x_4 & x_1 & -x_2 \\ x_4 & -x_3 & x_2 & x_1 \end{bmatrix} \quad \text{and} \quad \overset{-}{\mathbf{H}}(\mathbf{x}) \triangleq \begin{bmatrix} x_1 & -x_2 & -x_3 & -x_4 \\ x_2 & x_1 & x_4 & -x_3 \\ x_3 & -x_4 & x_1 & x_2 \\ x_4 & x_3 & -x_2 & x_1 \end{bmatrix}. \quad (2.52)$$

To reduce the notation of (2.52) we may note that the columns of the Hamiltonians are the orthogonal vectors from \mathbf{x} , thus

$$\begin{aligned} \overset{+}{\mathbf{H}}(\mathbf{x}) &= [\text{vec}_4 \mathbf{x} \quad \text{vec}_4(\mathbf{x}\hat{i}) \quad \text{vec}_4(\mathbf{x}\hat{j}) \quad \text{vec}_4(\mathbf{x}\hat{k})]; \\ \overset{-}{\mathbf{H}}(\mathbf{x}) &= [\text{vec}_4 \mathbf{x} \quad \text{vec}_4(\hat{i}\mathbf{x}) \quad \text{vec}_4(\hat{j}\mathbf{x}) \quad \text{vec}_4(\hat{k}\mathbf{x})]. \end{aligned} \quad (2.53)$$

We note that we have only been exploiting a matrix formulation for allowing the commutative propriety, however, in this manuscript, we are more interested on the solutions that are defined within the quaternion group, without needing to perform mapping to \mathbb{R}^4 . One such solution can exploit (2.13) and extend it to \mathbb{H} , thus resulting in commutation for \mathbf{a} and $\mathbf{b} \in \mathbb{H}$ becoming

$$\mathbf{a}\mathbf{b} = \mathbf{b}\mathbf{a} - 2 \text{Im}(\mathbf{b}) \times \text{Im}(\mathbf{a}). \quad (2.54)$$

Indeed, this simplifies the equations and quickens the computation. Nonetheless, the solution (2.54) is still complex and we are more interested in easier, less computationally demanding and intuitive solutions. Thus, by exploiting (2.49), in Definition 2.15 we present a quaternion representation of (2.53).

Definition 2.15. (Quaternion Hamiltonian) Given two quaternions \mathbf{a} and \mathbf{b} , the transformation based on the quaternion multiplication (2.5) can commute within the group by exploiting the operators $\overset{+}{\mathbf{h}}: \mathbb{H} \times \mathbb{H} \longrightarrow \mathbb{H}$ and $\overset{-}{\mathbf{h}}: \mathbb{H} \times \mathbb{H} \longrightarrow \mathbb{H}$, such that

$$\begin{aligned} \mathbf{y} = \mathbf{a}\mathbf{b} &= \overset{+}{\mathbf{h}}(\mathbf{a})\mathbf{b} = \mathbf{a}b_0 + \mathbf{a}\hat{i}b_1 + \mathbf{a}\hat{j}b_2 + \mathbf{a}\hat{k}b_3, \\ &= \overset{-}{\mathbf{h}}(\mathbf{b})\mathbf{a} = \mathbf{b}a_0 + \hat{i}\mathbf{b}a_1 + \hat{j}\mathbf{b}a_2 + \hat{k}\mathbf{b}a_3. \end{aligned} \quad (2.55)$$

We highlight here that the operations in (2.55) are equivalent to the ones in (2.52).

The main advantage of the quaternion-Hamiltonian in (2.55) is that it is defined only as quaternion multiplication by scalars without the tedious and costly operation of mapping quaternions to \mathbb{R}^4 and $\mathbb{R}^{4 \times 4}$ and back to \mathbb{H} . Hence, all operations are defined within the group in a more compact and intuitive format.

Much like their quaternion counterpart, dual quaternions have a matrix algebra operator that can allow the commutation of elements. This can be obtained based on a $\mathbb{R}^{8 \times 8}$ multiplication by the vector representation of a dual quaternion element—and, then mapped back to \mathbb{H} —as in [28, 66]. Thus, for $\underline{\mathbf{a}}$ and $\underline{\mathbf{b}} \in \underline{\mathbb{H}}$, and from (2.31) we have

$$\underline{\mathbf{y}} = \underline{\mathbf{a}}\underline{\mathbf{b}} = \text{vec}_8 \left(\overset{+}{\mathbf{H}}(\underline{\mathbf{a}}) \text{vec}_8 \underline{\mathbf{b}} \right) = \text{vec}_8 \left(\overset{-}{\mathbf{H}}(\underline{\mathbf{b}}) \text{vec}_8 \underline{\mathbf{a}} \right) \quad (2.56)$$

in which, this time, the dual quaternion Hamiltonians is defined from (2.52) as and with $\underline{x} = \mathbf{x}_P + \varepsilon \mathbf{x}_D \in \mathbb{H}$

$$\underline{\mathbf{H}}^+(\mathbf{x}) = \begin{bmatrix} \underline{\mathbf{H}}^+(\mathbf{x}_P) & 0 \\ \underline{\mathbf{H}}^+(\mathbf{x}_D) & \underline{\mathbf{H}}^+(\mathbf{x}_P) \end{bmatrix} \text{ and } \underline{\mathbf{H}}^-(\mathbf{x}) = \begin{bmatrix} \underline{\mathbf{H}}^-(\mathbf{x}_P) & 0 \\ \underline{\mathbf{H}}^-(\mathbf{x}_D) & \underline{\mathbf{H}}^-(\mathbf{x}_P) \end{bmatrix} \quad (2.57)$$

In contrast, [70] introduces the concept of orthogonal dual matrix combining the Hamilton operator matrix for quaternions (in 2.52) with dual unit ε . Herein, to commute between dual quaternions we take a similar approach based on the direct product of dual numbers with quaternion Hamiltonian operators of Definition 2.15, thus we have Definition 2.16.

Definition 2.16. (Dual Quaternion Hamiltonian) Given two dual quaternions $\underline{\mathbf{a}}$ and $\underline{\mathbf{b}}$ the he transformation based on the operation $\underline{\mathbf{y}} = \underline{\mathbf{a}} \underline{\mathbf{b}}$ in (2.31) can commute within the group by exploiting the operators $\underline{\mathbf{h}}^+ : \mathbb{H} \times \mathbb{H} \rightarrow \mathbb{H}$ and $\underline{\mathbf{h}}^- : \mathbb{H} \times \mathbb{H} \rightarrow \mathbb{H}$, such that

$$\begin{aligned} \underline{\mathbf{y}} = \underline{\mathbf{h}}^+(\underline{\mathbf{a}}) \underline{\mathbf{b}} &= \underline{\mathbf{h}}^+(\mathbf{a}_P) \mathbf{b}_P + \varepsilon (\underline{\mathbf{h}}^+(\mathbf{a}_P) \mathbf{b}_D + \underline{\mathbf{h}}^+(\mathbf{a}_D) \mathbf{b}_P), \\ \underline{\mathbf{y}} = \underline{\mathbf{h}}^-(\underline{\mathbf{b}}) \underline{\mathbf{a}} &= \underline{\mathbf{h}}^-(\mathbf{b}_P) \mathbf{a}_P + \varepsilon (\underline{\mathbf{h}}^-(\mathbf{b}_P) \mathbf{a}_D + \underline{\mathbf{h}}^-(\mathbf{b}_D) \mathbf{a}_P), \end{aligned} \quad (2.58)$$

in which $\underline{\mathbf{a}} = \mathbf{a}_P + \varepsilon \mathbf{a}_D$ and $\underline{\mathbf{b}} = \mathbf{b}_P + \varepsilon \mathbf{b}_D$, and $\underline{\mathbf{h}}^+$ and $\underline{\mathbf{h}}^-$ yields an orthogonal dual matrix as described in [70].

Note the operators $\underline{\mathbf{h}}^+$ and $\underline{\mathbf{h}}^-$ are similar to the dual matrix described in [70], yet it maps to a dual quaternion element. In other words, similarly to quaternion Hamiltonian operation in (2.55), the operations described in (2.58) are exclusively defined within the algebra of dual quaternions with no need to external mapping.

2.1.5 Dual Quaternion Vectors and Matrices

Although not very usual in the literature, we believe it to be quite important to also describe how dual quaternion vectors and dual quaternion matrices work⁶, as well as their main proprieties. Indeed, although the formalism for dual quaternion matrices and vectors might be intuitive, we also list this section as one of our contributions to the literature, as we have not found any references formally describing this mathematical structures.

Following the traditional vector formalism, we may define a dual quaternion vector as

$$\mathbb{H}^n \triangleq \left\{ \left[\underline{\mathbf{q}}_1 \quad \underline{\mathbf{q}}_2 \quad \underline{\mathbf{q}}_3 \quad \dots \quad \underline{\mathbf{q}}_n \right]^T : \underline{\mathbf{q}}_i \in \mathbb{H}, i \in \{1, 2, \dots, n\} \right\}, \quad (2.59)$$

With one important propriety of the dual quaternion vectors being the transpose operation and the conjugate

⁶The definitions of vectors and matrices used throughout this paper are not strict, in the sense that vector elements are not defined over a field (note that quaternions, for instance, are division rings, not fields). Nonetheless, operations from vector space are valid for our vector definition as shown in the provided operations in the manuscript. The nomenclature is used for historical reasons and to easy analysis.

operation. Thus, given the dual quaternion vector $\underline{Q} = \begin{bmatrix} \underline{q}_1 \\ \underline{q}_2 \\ \underline{q}_3 \\ \vdots \\ \underline{q}_n \end{bmatrix} \in \mathbb{H}^n$, we may also define its transpose as

$$\underline{Q}^T \triangleq \left[\underline{q}_1 \quad \underline{q}_2 \quad \underline{q}_3 \quad \cdots \quad \underline{q}_n \right], \quad (2.60)$$

and its conjugate as

$$\underline{Q}^* = \begin{bmatrix} \underline{q}_1^* \\ \underline{q}_2^* \\ \underline{q}_3^* \\ \vdots \\ \underline{q}_n^* \end{bmatrix}. \quad (2.61)$$

Moreover, we may also define some proprieties of the dual quaternion vector, such as the addition in Propriety 2.13.

Propriety 2.13. (Dual Quaternion Vector Addition) Given two dual quaternions vectors such as $\underline{Q} = \begin{bmatrix} \underline{q}_1 & \underline{q}_2 & \underline{q}_3 & \cdots & \underline{q}_n \end{bmatrix} \in \mathbb{H}^n$ and $\underline{Q}' = \begin{bmatrix} \underline{q}'_1 & \underline{q}'_2 & \underline{q}'_3 & \cdots & \underline{q}'_n \end{bmatrix} \in \mathbb{H}^n$ the addition operation $(\mathbb{H}^n \times \mathbb{H}^n \rightarrow \mathbb{H}^n)$ can be defined as

$$\underline{Q} + \underline{Q}' := \left[\underline{q}_1 + \underline{q}'_1 \quad \underline{q}_2 + \underline{q}'_2 \quad \underline{q}_3 + \underline{q}'_3 \quad \cdots \quad \underline{q}_n + \underline{q}'_n \right], \quad (2.62)$$

in which the individual elements of the vector are calculated by the dual quaternion addition in (2.29).

Propriety 2.14. (Dual Quaternion Vector Multiplication) Given two dual quaternions vectors such as $\underline{Q} = \begin{bmatrix} \underline{q}_1 & \underline{q}_2 & \underline{q}_3 & \cdots & \underline{q}_n \end{bmatrix} \in \mathbb{H}^n$ and $\underline{Q}' = \begin{bmatrix} \underline{q}'_1 & \underline{q}'_2 & \underline{q}'_3 & \cdots & \underline{q}'_n \end{bmatrix} \in \mathbb{H}^n$ the multiplication operation $(\mathbb{H}^n \times (\mathbb{H}^n)^T \rightarrow \mathbb{H})$ can be defined as

$$\underline{Q} (\underline{Q}')^T = \underline{q}_1 \underline{q}'_1 + \underline{q}_2 \underline{q}'_2 + \underline{q}_3 \underline{q}'_3 + \cdots + \underline{q}_n \underline{q}'_n, \quad (2.63)$$

in which the product between two dual quaternions is defined in (2.31) and the addition is in (2.29).

Dual quaternion vectors can be very useful in some applications, as we will see in Chapter 4. However, we may also find it necessary, in many cases, to further expand these vectors to their matrix notation. Thus, similarly to (2.59) we may define the dual quaternion matrix as

$$\mathbb{H}^{n \times m} \triangleq \left\{ \begin{bmatrix} \underline{q}_{1,1} & \underline{q}_{1,2} & \cdots & \underline{q}_{1,m} \\ \underline{q}_{2,1} & \underline{q}_{2,2} & \cdots & \underline{q}_{2,m} \\ \vdots & \vdots & \ddots & \vdots \\ \underline{q}_{n,1} & \underline{q}_{n,2} & \cdots & \underline{q}_{n,m} \end{bmatrix} : \underline{q}_{i,j} \in \mathbb{H}, i \in \{1, 2, \dots, n\} \text{ and } j \in \{1, 2, \dots, m\} \right\}. \quad (2.64)$$

In this case, the elements may be combined similarly to how they are in real matrix notation to perform a number of operations.

Furthermore, in Proprieties 2.15 and 2.16 we show the addition and multiplication.

Given two dual quaternion matrices such as $\underline{Q} = \begin{bmatrix} \underline{q}_{1,1} & \underline{q}_{1,2} & \cdots & \underline{q}_{1,m} \\ \underline{q}_{2,1} & \underline{q}_{2,2} & \cdots & \underline{q}_{2,m} \\ \vdots & \vdots & \ddots & \vdots \\ \underline{q}_{n,1} & \underline{q}_{n,2} & \cdots & \underline{q}_{n,m} \end{bmatrix} \in \mathbb{H}^{n \times m}$ and $\underline{Q}' = \begin{bmatrix} \underline{q}'_{1,1} & \underline{q}'_{1,2} & \cdots & \underline{q}'_{1,m} \\ \underline{q}'_{2,1} & \underline{q}'_{2,2} & \cdots & \underline{q}'_{2,m} \\ \vdots & \vdots & \ddots & \vdots \\ \underline{q}'_{n,1} & \underline{q}'_{n,2} & \cdots & \underline{q}'_{n,m} \end{bmatrix} \in \mathbb{H}^{n \times m}$ we define the operations bellow.

Propriety 2.15. (Dual Quaternion Matrix Addition) The addition operation such that $\mathbb{H}^{n \times m} \times \mathbb{H}^{n \times m} \rightarrow \mathbb{H}^{n \times m}$ can be defined as

$$\underline{Q} + \underline{Q}' = \begin{bmatrix} \underline{q}_{1,1} + \underline{q}'_{1,1} & \underline{q}_{1,2} + \underline{q}'_{1,2} & \cdots & \underline{q}_{1,m} + \underline{q}'_{1,m} \\ \underline{q}_{2,1} + \underline{q}'_{2,1} & \underline{q}_{2,2} + \underline{q}'_{2,2} & \cdots & \underline{q}_{2,m} + \underline{q}'_{2,m} \\ \vdots & \vdots & \ddots & \vdots \\ \underline{q}_{n,1} + \underline{q}'_{n,1} & \underline{q}_{n,2} + \underline{q}'_{n,2} & \cdots & \underline{q}_{n,m} + \underline{q}'_{n,m} \end{bmatrix}, \quad (2.65)$$

in which the individual elements of the vector calculated through the dual quaternion addition in (2.29)

Propriety 2.16. (Dual Quaternion Matrix Multiplication) The multiplication operation, such that $\mathbb{H}^{n \times m} \times \mathbb{H}^{m \times n} \rightarrow \mathbb{H}^{n \times n}$, can be defined as

$$\underline{Q} (\underline{Q}')^T = \begin{bmatrix} \underline{q}_{1,1} \underline{q}'_{1,1} + \underline{q}_{1,2} \underline{q}'_{1,2} + \cdots + \underline{q}_{1,m} \underline{q}'_{1,m} & \cdots & \underline{q}_{1,1} \underline{q}'_{n,1} + \underline{q}_{1,2} \underline{q}'_{n,2} + \cdots + \underline{q}_{1,m} \underline{q}'_{n,m} \\ \underline{q}_{2,1} \underline{q}'_{1,1} + \underline{q}_{2,2} \underline{q}'_{1,2} + \cdots + \underline{q}_{2,m} \underline{q}'_{1,m} & \cdots & \underline{q}_{2,1} \underline{q}'_{n,1} + \underline{q}_{2,2} \underline{q}'_{n,2} + \cdots + \underline{q}_{2,m} \underline{q}'_{n,m} \\ \vdots & \ddots & \vdots \\ \underline{q}_{n,1} \underline{q}'_{1,1} + \underline{q}_{n,2} \underline{q}'_{1,2} + \cdots + \underline{q}_{n,m} \underline{q}'_{1,m} & \cdots & \underline{q}_{n,1} \underline{q}'_{n,1} + \underline{q}_{n,2} \underline{q}'_{n,2} + \cdots + \underline{q}_{n,m} \underline{q}'_{n,m} \end{bmatrix} \quad (2.66)$$

Remark 2.11. The dual quaternions vector multiplication defined in Propriety 2.14 is one particular case of dual quaternion matrix multiplication, in which we are multiplying $\mathbb{H}^{1 \times n} \times \mathbb{H}^{n \times 1} = \mathbb{H}^{1 \times 1}$.

As was the case for the vector, we have both the transposed and the conjugate operations. That is, given the

dual quaternion matrix $\underline{Q} = \begin{bmatrix} \underline{q}_{1,1} & \underline{q}_{1,2} & \cdots & \underline{q}_{1,m} \\ \underline{q}_{2,1} & \underline{q}_{2,2} & \cdots & \underline{q}_{2,m} \\ \vdots & \vdots & \ddots & \vdots \\ \underline{q}_{n,1} & \underline{q}_{n,2} & \cdots & \underline{q}_{n,m} \end{bmatrix} \in \mathbb{H}^{n \times m}$, we have the transpose operation being

the “swapping” of the matrix’s elements such as $\underline{q}_{i,j}$ in which $i \neq j$, resulting in

$$\underline{Q}^T = \begin{bmatrix} \underline{q}_{1,1} & \underline{q}_{2,1} & \cdots & \underline{q}_{n,1} \\ \underline{q}_{1,2} & \underline{q}_{2,2} & \cdots & \underline{q}_{n,2} \\ \vdots & \vdots & \ddots & \vdots \\ \underline{q}_{1,m} & \underline{q}_{2,m} & \cdots & \underline{q}_{n,m} \end{bmatrix}. \quad (2.67)$$

Regarding the conjugate, for the same matrix \underline{Q} , we have

$$\underline{Q}^* = \begin{bmatrix} \underline{q}_{1,1}^* & \underline{q}_{1,2}^* & \cdots & \underline{q}_{1,m}^* \\ \underline{q}_{2,1}^* & \underline{q}_{2,2}^* & \cdots & \underline{q}_{2,m}^* \\ \vdots & \vdots & \ddots & \vdots \\ \underline{q}_{n,1}^* & \underline{q}_{n,2}^* & \cdots & \underline{q}_{n,m}^* \end{bmatrix}, \quad (2.68)$$

with each $\underline{q}_{i,j}^*$ being the dual quaternion conjugate as in 2.34.

Finally, to close off this section, we must add that a subgroup for both the dual quaternion vector as the dual quaternion matrices can be the one in which the the individual elements of the vector (or matrix) is a pure quaternion. That is, the pure dual quaternion vector could be defined as

$$\mathbb{H}_0^n \triangleq \left\{ \left[\underline{q}_1 \quad \underline{q}_2 \quad \underline{q}_3 \quad \cdots \quad \underline{q}_n \right]^T : \underline{q}_i \in \mathbb{H}_0, i \in \{1, 2, \dots, n\} \right\}, \quad (2.69)$$

and the pure dual quaternion matrix would be

$$\mathbb{H}_0^{n \times m} \triangleq \left\{ \left[\begin{array}{cccc} \underline{q}_{1,1} & \underline{q}_{1,2} & \cdots & \underline{q}_{1,m} \\ \underline{q}_{2,1} & \underline{q}_{2,2} & \cdots & \underline{q}_{2,m} \\ \vdots & \vdots & \ddots & \vdots \\ \underline{q}_{n,1} & \underline{q}_{n,2} & \cdots & \underline{q}_{n,m} \end{array} \right] : \underline{q}_{i,j} \in \mathbb{H}_0, i \in \{1, 2, \dots, n\} \text{ and } j \in \{1, 2, \dots, m\} \right\}. \quad (2.70)$$

Regarding the pure dual quaternion vectors we can also define three very important operations: the element-wise cross product, the element-wise inner product and the element-wise double geodesic product. All this operations take advantage of their equivalents in \mathbb{H}_0 and perform these operations between every two pairs of pure dual quaternions in the vector. This can be better illustrated by Definitions 2.17 to 2.19.

Definition 2.17. (DQ-Vec Cross Product) Let $\underline{P}_1 = \left[\underline{p}_{1,1} \quad \underline{p}_{1,2} \quad \cdots \quad \underline{p}_{1,n} \right]^T$ and $\underline{P}_2 = \left[\underline{p}_{2,1} \quad \underline{p}_{2,2} \quad \cdots \quad \underline{p}_{2,n} \right]^T \in \mathbb{H}_0^n$, then the element wise cross product operation $\otimes : \mathbb{H}^n \times \mathbb{H}^n \rightarrow \mathbb{H}^n$ can be defined as the element wise cross product between the dual quaternions

$$\underline{P}_1 \otimes \underline{P}_2 = \left[\underline{p}_{1,1} \times \underline{p}_{2,1} \quad \underline{p}_{1,2} \times \underline{p}_{2,2} \quad \cdots \quad \underline{p}_{1,n} \times \underline{p}_{2,n} \right]^T. \quad (2.71)$$

Definition 2.18. (DQ-Vec Inner Product) Let $\underline{P}_1 = \left[\underline{p}_{1,1} \quad \underline{p}_{1,2} \quad \cdots \quad \underline{p}_{1,n} \right]^T$ and $\underline{P}_2 = \left[\underline{p}_{2,1} \quad \underline{p}_{2,2} \quad \cdots \quad \underline{p}_{2,n} \right]^T \in \mathbb{H}_0^n$, then the element wise inner product operation $\odot : \mathbb{H}^n \times \mathbb{H}^n \rightarrow \mathbb{H}^n$ can be defined as the element wise inner product between the dual quaternions

$$\underline{P}_1 \odot \underline{P}_2 = \left[\underline{p}_{1,1} \cdot \underline{p}_{2,1} \quad \underline{p}_{1,2} \cdot \underline{p}_{2,2} \quad \cdots \quad \underline{p}_{1,n} \cdot \underline{p}_{2,n} \right]^T. \quad (2.72)$$

Definition 2.19. (DQ-Vec Double Geodesic Product) Once more let $\underline{P}_1 = \left[\underline{p}_{1,1} \quad \underline{p}_{1,2} \quad \cdots \quad \underline{p}_{1,n} \right]^T$ and $\underline{P}_2 = \left[\underline{p}_{2,1} \quad \underline{p}_{2,2} \quad \cdots \quad \underline{p}_{2,n} \right]^T \in \mathbb{H}_0^n$, then the element-wise double geodesic product operation $\otimes : \mathbb{H}^n \times \mathbb{H}^n \rightarrow \mathbb{H}^n$ can be defined as

$$\underline{P}_1 \otimes \underline{P}_2 = \left[\underline{p}_{1,1} \odot \underline{p}_{2,1} \quad \underline{p}_{1,2} \odot \underline{p}_{2,2} \quad \cdots \quad \underline{p}_{1,n} \odot \underline{p}_{2,n} \right]^T. \quad (2.73)$$

2.1.6 Advantages of Dual Quaternions

The algebra of dual quaternions, presented in this chapter, has many advantages when compared to other representations. In particular, the DQ may represent complex problems in a compact and unified manner [71, 76]. Particularly, the dual quaternions compactness can be verified in the fact that with only eight parameters they are capable of representing the full pose of a body—in contrast to the homogeneous transformation matrices, which need 16 elements to represent a full pose. Moreover, the dual quaternions, opposed to the rotations matrices and the Euler angles, represent both translations and rotations simultaneously, which is by itself a great advantage for applications in control.

Another great advantage of dual quaternions is that they are also singularity free. That is, they can represent many geometric meaningful variables and the representation itself will not produce any singularity, such as the Gimbal lock, which is famous under the Euler angles. In terms of efficiency, we have many works providing an extensive analysis between Dual Quaternions and other algebras and coming to the conclusion that they are the best way to describe a screw in space [39–41, 77].

2.2 DUAL QUATERNIONS GEOMETRIC REPRESENTATION

Dual quaternions are particularly good for representing geometric variables in space. In particular, a good motivation for the use of quaternions and dual quaternions in the literature is their compact and singularity free representation for points in three-dimensional space—not only it is easy for us to represent a pose in space, but it is also fairly easy to represent translations and rotations within these algebras. Regarding only dual quaternions, they are an even richer tool for describing motions, as within this algebra we have the translation and rotation coupled together in one single singularity free structure. In this manner, Subsection 2.2.1 will explore the representation of a point in both 2D and 3D space.

Moreover, there are also other geometric structures we may wish to explore with the algebra of dual quaternions. In particular, although dual quaternions are very well suited to represent points in space, they are even better at representing another geometric variable in space: lines. Indeed, as [41] has argued, dual quaternions are the most compact and efficient manner to represent a line in space, providing us with one extra motivation for studying the DQ lines. Thus, Subsection 2.2.2 will better explore how we may represent lines with dual quaternions.

Finally, when dealing with serial robots we will also want to describe arc-like trajectories, which is done via the exponential map. This will be explored in Subsection 2.2.3.

2.2.1 Points in Three-dimensional Space

Whenever we start reading a book on robotics one of the first aspects detailed there will usually be the representation of a point in two or three-dimensional space. Indeed, to perform any task with a robot we must know, before all else, where said robot is with respect to a frame of reference – or at least we must know where the point located at its center of mass is.

To better illustrate the three-dimensional representation of a point in space, we first take a step back to the 2D case. Here we will look at the imaginary plane and a complex number $p = (p_x + \hat{i}p_y) \in \mathbb{C}$. Of course,

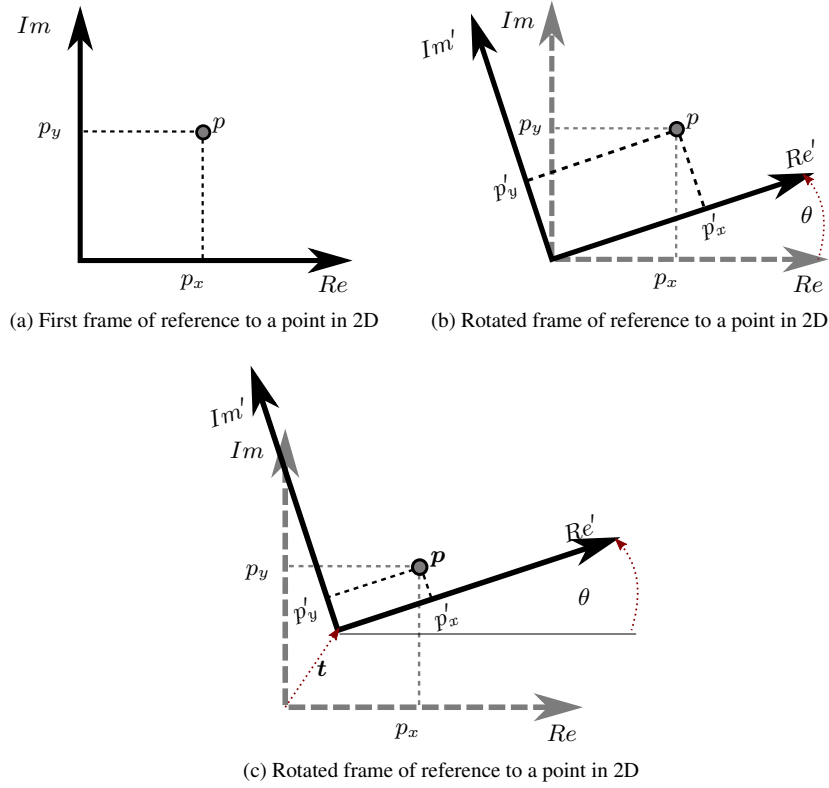


Figure 2.1: Representation of a point in Imaginary Plane represented in different frames

knowing that the orthogonal basis is $\{1, \hat{i}\}$ we can simply use the coordinates (p_x, p_y) and place the point p in the plane, as in Figure 2.1a. When we rotate this plane by an angle of θ , however, the point p will change its coordinates, as is seen in Figure 2.1b.

Let the frame in Figure 2.1a be \mathcal{F} and the rotated frame in Figure 2.1b be \mathcal{F}' . From simple trigonometry we can map the axis in \mathcal{F} to \mathcal{F}' . That is,

$$\begin{aligned} Re : 1 &\longrightarrow \cos(\theta) + \hat{i}\sin(\theta) \\ Im : \hat{i} &\longrightarrow -\sin(\theta) + \hat{i}\cos(\theta). \end{aligned}$$

Thus we have the point p in frame \mathcal{F}' is

$$\begin{aligned} \mathbf{p}' &= p_x \left(\cos(\theta) + \hat{i}\sin(\theta) \right) + p_y \left(-\sin(\theta) + \hat{i}\cos(\theta) \right), \\ &= (p_x \cos(\theta) - p_y \sin(\theta)) + \hat{i} (p_x \sin(\theta) + p_y \cos(\theta)) \\ &= \left(\cos(\theta) + \hat{i}\sin(\theta) \right) \left(p_x + \hat{i}p_y \right) \\ &= \left(\cos(\theta) + \hat{i}\sin(\theta) \right) \mathbf{p}. \end{aligned} \tag{2.74}$$

We note in this example that $\mathbf{p}' = \left(\cos(\theta) + \hat{i}\sin(\theta) \right) \mathbf{p}$, thus we define $\mathbf{r} \triangleq \left(\cos(\theta) + \hat{i}\sin(\theta) \right)$ a frame rotation around the origin of the complex plane and $\mathbf{p}' = \mathbf{r}\mathbf{p}$. Furthermore, if we want to also translate the frame, as in Figure 2.1c, we shall add the components of the translation to the rotated point, that is for

$\mathbf{t} = t_x + \hat{i}t_y$, $\mathbf{p}' = \mathbf{r}\mathbf{p} + \mathbf{t}$. Also, a number of successive frame rotations can be performed by multiplying the results, that is, if we rotate the frame by \mathbf{r}_1 and then \mathbf{r}_2 , the total rotation is $\mathbf{r}_{\text{Total}} = \mathbf{r}_1\mathbf{r}_2$ and $\mathbf{p}' = \mathbf{r}_{\text{Total}}\mathbf{p} = \mathbf{r}_1\mathbf{r}_2\mathbf{p}$ [28, 78].

Another interesting propriety in this scenario is the fact that rotations in the complex plane are made possible by a subgroup of complex numbers with norm equal to one, that is, $\|\mathbf{r}\| = \sqrt{\mathbf{r}^*\mathbf{r}} = 1$, and, similarly to what was shown for unit quaternions $\mathbf{r}^* = \mathbf{r}^{-1}$. Furthermore, a complex number's conjugate may represents a rotation on the opposite direction. That is, for the angle $-\theta$

$$\cos(-\theta) + \hat{i}\sin(-\theta) = \cos(\theta) - \hat{i}\sin(\theta) = \mathbf{r}^*.$$

This implies that if we have a transformation from frame \mathcal{F} to \mathcal{F}' such as (2.74), to go back to \mathcal{F} we must simply multiply for the conjugate,

$$\begin{aligned}\mathbf{p}' &= \mathbf{r}\mathbf{p} \\ \mathbf{r}^*\mathbf{p}' &= \mathbf{r}^*\mathbf{r}\mathbf{p} \\ \mathbf{r}^*\mathbf{p}' &= \mathbf{p}.\end{aligned}$$

Here we presented the transformation of a fixed point at one frame of reference to another frame of reference. However, this may be seen at another perspective, complex numbers can be used in the similarly to represent the position and orientation of a frame attached to a point in relation to a frame of reference.

As previously discussed, quaternions and complex numbers share many attributes, as the former is an expansion to represent motions in the three-dimensional space. We shall now take a similar approach to describe a point in space with quaternions. Thus, let us analyze the point $\mathbf{p} = \hat{i}p_x + \hat{j}p_y + \hat{k}p_z$ represented in Figure 2.2.

To represent the point at frame \mathcal{F}_0 , as in Figure 2.2a we shall use the notation \mathbf{p}^0 . We note that the point is represented by its position, thus, it is a pure quaternion, with its coordinates in each of the orthogonal axis $(\hat{i}, \hat{j}, \hat{k})$ and, as $Re(\mathbf{p}) = 0$, we omit the real axis.

Translation can be achieved by adding pure quaternions, that is, if $\mathbf{p}_t^0 = \hat{i}t_x + \hat{j}t_y + \hat{k}t_z$ is a pure quaternion representing a point's translation in \mathcal{F}_0 , then the translation operation can be given by

$$\mathbf{p}_{\text{new}}^0 = \mathbf{p}^0 + \mathbf{p}_t^0 \quad (2.75)$$

We also define here a quaternion rotation as follows.

Definition 2.20. (Quaternion Attitude) [28] A quaternion rotation of an angle θ around and axis $\mathbf{u} \in \mathbb{H}_0$ is defined as

$$\mathbf{r} \triangleq \cos\left(\frac{\theta}{2}\right) + \mathbf{u}\sin\left(\frac{\theta}{2}\right), \quad (2.76)$$

in which \mathbf{r} will always be a unit quaternion.

With quaternion rotations we may perform both frame and point transformation [79]. Also, this definition makes the reason behind why in (2.17) the unit quaternion conjugate represents the inverse of a rotation clearer. If we take a simple inverse of the rotation angle we have that $\cos(-\theta/2) = \cos(\theta/2)$ and $\sin(-\theta/2) = -\sin(\theta/2)$, which turns (2.76) in \mathbf{r}^* .

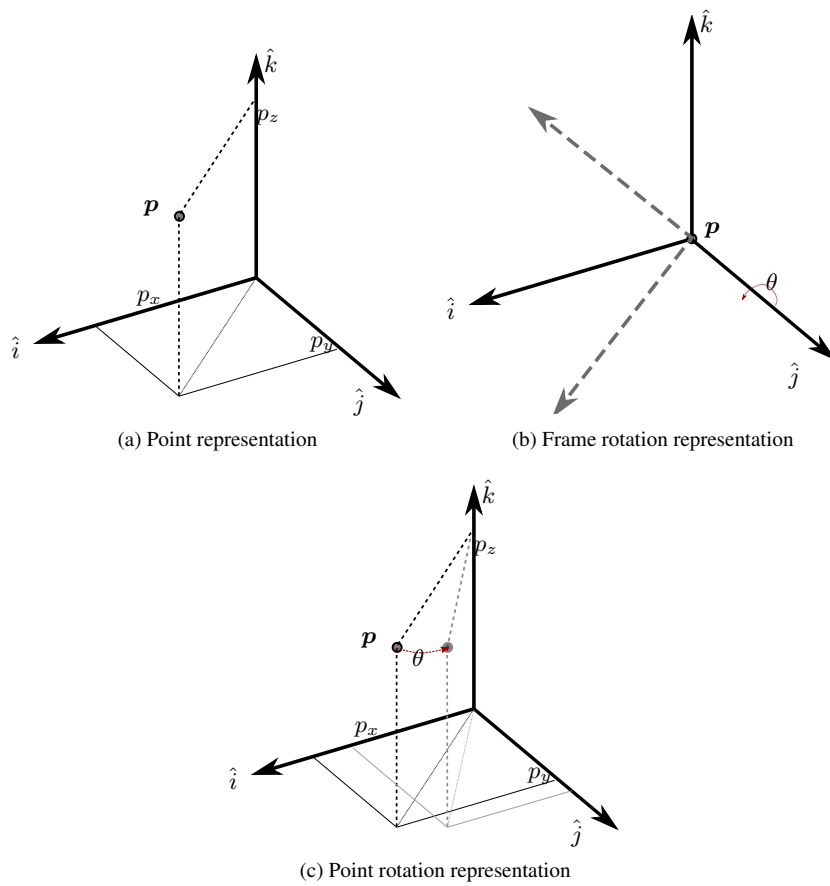


Figure 2.2: Representation of a point in three-dimensional space

Moreover, we may also define the transformation of a point between two frames. That is, considering $\mathbf{p}^0 = \hat{i}p_x^0 + \hat{j}p_y^0 + \hat{k}p_z^0$ to be a point with regards to the fixed frame \mathcal{F}_0 , $\mathbf{p}^1 = \hat{i}p_x^1 + \hat{j}p_y^1 + \hat{k}p_z^1$ to be the same point but in \mathcal{F}_1 and the unit rotation \mathbf{r}_1^0 to be the transformation from \mathcal{F}_0 to \mathcal{F}_1 , then transformation can be given by

$$\mathbf{p}^1 = \mathbf{r}_1^0 \mathbf{p}^0 (\mathbf{r}_1^0)^*, \quad (2.77)$$

that is, we have changed the frame of reference of \mathbf{p} while it remained fixed in space.

The operation in (2.77) is the adjoint transformation and it can be more succinctly expressed with the operator $Ad(\bullet)$, as per Definition 2.21. The frame transformation operation is also illustrated in figure 2.2b.

Definition 2.21. (Quaternion Adjoint Transformation) [28] The adjoint transformation [28] can be defined compactly expressed by the operator

$$Ad(\mathbf{r})\mathbf{p} \triangleq \mathbf{r}\mathbf{p}\mathbf{r}^*. \quad (2.78)$$

Here, once more \mathbf{r} is a unit quaternion and \mathbf{p} is a pure quaternion.

From Definition 2.21 we have a frame transformation such as in Figure 2.2b. However, we may also define a point rotation, such as in Figure 2.2c. In this case the transformation of point \mathbf{p}_0^0 to point \mathbf{p}_1^0 is defined as

$$\mathbf{p}_0^0 = (\mathbf{r}_1^0)^* \mathbf{p}_1^0 \mathbf{r}_1^0, \quad (2.79)$$

with \mathbf{r}_1^0 being the rotation between the two points.

Herein, we highlight that (2.77) and (2.79) are closely connected and, as argued in [79] are simply a matter of perspective. In other words, (2.77) can be interpreted as an observer fixed at point \mathbf{p} , watching its reference frame rotate at an angle of $-\theta$. Conversely, (2.79) can be interpreted as if the observer is fixed to the reference frame and watches as the point rotates $-\theta$. However, these two equations are interchangeable, this is evident for example when we invert (2.77) and obtain $\mathbf{p}^0 = (\mathbf{r}_1^0)^* \mathbf{p}^1 \mathbf{r}_1^0$. This time the observer fixed at \mathbf{p} sees the frame rotating in the opposite direction. A good thought exercise is to apply this same interpretation to (2.75) in order to differentiate point translation and frame translation.

As stated in Definition 2.6, the unit quaternion group has the closure property, this means that the product of two rotations is still a rotation, and from that we may perform sequences of rotations in objects, such as $\mathbf{r}_3^0 = \mathbf{r}_1^0 \mathbf{r}_2^1 \mathbf{r}_3^2$ and from (2.77) we can have

$$\mathbf{p}^1 = \mathbf{r}_3^0 \mathbf{p}^0 (\mathbf{r}_3^0)^* = (\mathbf{r}_1^0 \mathbf{r}_2^1 \mathbf{r}_3^2) \mathbf{p}^0 (\mathbf{r}_1^0 \mathbf{r}_2^1 \mathbf{r}_3^2)^*. \quad (2.80)$$

Finally, we may also represent a full rigid body motion with quaternions by making compositions of both (2.75) and (2.77).

To close this section we shall also present here how a point can be represented in three dimensions with dual quaternion algebra.

Definition 2.22. (Dual Quaternion Pose) [47, 80] We may represent an arbitrary displacement (or a pose) with dual quaternions by defining a rotation followed by a translation, or, in terms of a unit dual quaternion,

$$\underline{\mathbf{x}} \triangleq \mathbf{r} + \varepsilon \frac{1}{2} \mathbf{r} \mathbf{p}. \quad (2.81)$$

In this case, $\mathbf{r} \in \text{Spin}(3)$ is a unit quaternion representation of the point's attitude with respect to an inertial frame, whilst $\mathbf{p} \in \mathbb{H}_0$ is the position (or similarly the point's rotation and translation).

We remark here that from 2.81 we may intuitively extract the rotation of element of (2.81) with the operator (2.25),

$$\mathbf{r} = \mathcal{P}(\underline{\mathbf{x}}). \quad (2.82)$$

By taking advantage of (2.82) and the operator in (2.26), we may also define the translation element,

$$\begin{aligned} \mathcal{D}(\underline{\mathbf{x}}) &= (1/2) \mathbf{r} \mathbf{p} \\ 2\mathcal{D}(\underline{\mathbf{x}}) &= \mathbf{r} \mathbf{p} \\ 2\mathcal{P}(\underline{\mathbf{x}})^* \mathcal{D}(\underline{\mathbf{x}}) &= \mathbf{p}. \end{aligned} \quad (2.83)$$

We also highlight that (2.81) may also be rewritten as a translation followed by a rotation

$$\underline{\mathbf{x}} = \mathbf{r} + \varepsilon (1/2) \mathbf{p}' \mathbf{r}. \quad (2.84)$$

In this case, we may relate \mathbf{p} and \mathbf{p}' by a frame transformation, such as $\mathbf{p}' = \text{Ad}(\mathbf{r})\mathbf{p}$. This result is due to the fact that \mathbf{p} happens after the rotation, thus we must undo that rotation in order to get \mathbf{p}' .

Finally, due to Definition 2.14, we may also represent a sequence of motions with dual quaternions as sequence of multiplications. That is

$$\underline{\mathbf{x}}_n^0 = \underline{\mathbf{x}}_1^0 \underline{\mathbf{x}}_2^1 \dots \underline{\mathbf{x}}_n^{n-1}. \quad (2.85)$$

2.2.2 Plucker Coordinates: Lines in Three-dimensional Space

Besides representing points, dual quaternions are also particularly good to represent lines in space and their transformations. Herein, in this subsection, we will introduce the a formulation for Plücker's representation of a line based on dual quaternion lines, that is the dual quaternion Plücker coordinates. Figure 2.3 shows the aforementioned representation in three dimensional space and in Definition 2.23 we formally present its mathematical formulation.

Definition 2.23. (Plücker's DQ Coordinates for a Line) [28, 40] A Plücker's dual quaternion representation of a line is a pure dual quaternion defined as

$$\underline{\mathbf{l}} \triangleq \mathbf{l} + \varepsilon \mathbf{m}, \quad (2.86)$$

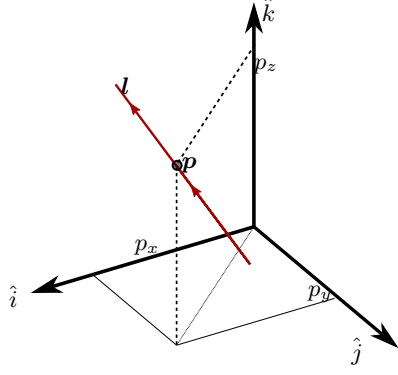


Figure 2.3: Representation of a Plücker Line in three-dimensional space.

in which $\underline{l} \in \mathbb{H}_0$ and $\|\underline{l}\| = 1$ is a unit vector representing the line direction, $\underline{p} \in \mathbb{H}_0$ is a point in the line and $\underline{m} \in \mathbb{H}_0$ is the line's moment with respect to the space frame origin, defined as

$$\underline{m} = \underline{p} \times \underline{l}. \quad (2.87)$$

Furthermore, the set of parameters \underline{l} and \underline{m} are collectively called Plücker's coordinates.

Although the literature has many references to Plücker's line representation with different algebras (as a pair of free vectors) [49, 81–83], it fits particularly well in the dual quaternion formalism as we can have the \underline{l} and \underline{m} coupled together in a unique element in the dual quaternion space.

Similarly to Definition 2.21 for quaternions, we may define the map of one frame to another within the algebra of dual quaternions. That is, the line represented by \underline{l}^0 in frame \mathcal{F}_0 may be expressed in frame \mathcal{F}_1 as

$$\underline{l}^1 = \underline{x} \underline{l}^0 \underline{x}^*. \quad (2.88)$$

Here $\underline{x} \in \text{Spin}(3) \times \mathbb{R}^3$ is the transformation from \mathcal{F}_0 to \mathcal{F}_1 .

Moreover, we may also find it useful to define frame transformations for dual quaternion lines as we have defined for quaternion points in Definition 2.21.

Definition 2.24. (Dual Quaternion Adjoint) [28, 37] The Adjoint representation of an unit dual quaternion \underline{x} is given by the operator $Ad(\underline{x})$ defined over \mathbb{H}_0 as

$$Ad(\underline{x}) \underline{p}^0 \triangleq \underline{x} \underline{p}^0 \underline{x}^* = \underline{p}^1. \quad (2.89)$$

Having in mind that the primal and dual parts of a pure dual quaternion have physical interpretation, it is convenient to describe the adjoint transformation in terms of these parts as in the following proposition.

Proposition 2.2. (Dual Quaternion Adjoint - Alternative definition) [60]^a The Adjoint operation associated to a unit dual quaternion \underline{x} for any pure dual quaternion $\underline{p} = \mathbf{p} + \varepsilon\mathbf{p}'$ can be defined as

$$Ad(\underline{x})\underline{p} \triangleq \mathbf{A}_P(\underline{x})\mathbf{p} + \varepsilon(\mathbf{A}_P(\underline{x})\mathbf{p}' + \mathbf{A}_D(\underline{x})\mathbf{p}), \quad (2.90)$$

in which $\mathbf{A}_P(\underline{x})$ and $\mathbf{A}_D(\underline{x})$ are given by

$$\mathbf{A}_P(\underline{x}) = \overset{+}{\mathbf{h}}(\mathbf{x})\bar{\mathbf{h}}(\mathbf{x}^*), \mathbf{A}_D(\underline{x}) = 2\overset{+}{\mathbf{h}}(\mathbf{x})\bar{\mathbf{h}}(\mathbf{x}^*). \quad (2.91)$$

^aA complete derivation for (2.90) is presented in Subsection 5.2.3

2.2.3 The Exponential: Curves in Three-dimensional Space

In this section we have explored the many different ways we can represent points, both in planes with complex numbers and in space with quaternions and dual quaternions. We have also described how to represent lines in space by means of Plücker coordinates and how to do frame transformations with them. Thus, we shall introduce here the dual quaternion exponential and how they can be attached to a fixed point in a line in order for it to rotate and perform a motion with an arc-like trajectory.

Much like with imaginary numbers, the dual quaternions exponential is suited to represent a curve in space. In Definition 2.25 we show the exponential map for a dual quaternion.

Definition 2.25. (Dual Quaternion Exponential) [84] For a pure dual quaternion \underline{q} , the exponential can be defined as

$$exp(\underline{q}) \triangleq \mathcal{P}(exp(\underline{q})) + \varepsilon\mathcal{D}(\underline{q})\mathcal{P}(exp(\underline{q})), \quad (2.92)$$

with

$$\mathcal{P}(exp(\underline{q})) \triangleq \begin{cases} \cos \|\mathcal{P}(\underline{q})\| + \frac{\sin \|\mathcal{P}(\underline{q})\|}{\|\mathcal{P}(\underline{q})\|} \mathcal{P}(\underline{q}) & \text{if } \|\mathcal{P}(\underline{q})\| \neq 0 \\ 1 & \text{o.w.} \end{cases} \quad (2.93)$$

3

ROBOTIC FUNDAMENTALS

According to the Robot Institute of America, in 1979, a robot could be defined as “A robot is a reprogrammable multifunctional manipulator designed to move materials, parts, tools or specialized devices through variable programmed motions for the performance of a variety of tasks”. That is, a robot was supposed to be a very adaptable tool to interact with the environment in order to perform tasks—and that obvious tool at the time was a robot arm. Indeed, one of the earliest examples of a modern robot was also a manipulator: the Unimate, invented in 1950 by George Devol [3,6]. Of course, the definition of what a robot is supposed to be has evolved through the years, and the robots have become much more sophisticated than Devol’s first prototype. Yet, anthropomorphic robotic arms remain to this day one of the main areas of interest among roboticists around the world—and, in particular, it is also the object of interest of this manuscript.

One of the kinds of anthropomorphic robotic arm that is widely available today are the serial manipulators. These robots may be represented by a sequence of rigid bodies (links) connected to each other by means of revolute, prismatic, cylindrical or spherical joints [5, 36, 49, 85]. In Figure 3.1 we see a representation of a manipulator with four joints and three links. In addition, this manipulator also has a base and a gripper—the end effector.

This Chapter will further explore the kinematics and dynamics of a manipulator such as in Figure 3.1. We first introduce the unconstrained rigid body kinematics with the algebra of dual quaternions and then rigid body dynamics. As each of the robot’s link represents a rigid body, it is important to define the physical equations that rule these bodies motion when unconstrained. Afterwards, Section 3.3 introduces the manipulator’s kinematics. In short, in this section we add the constraints imposed by the robot’s geometry in order to model the robot as a whole.

3.1 RIGID BODY KINEMATICS

A manipulator is made out of interconnected rigid bodies called links. As defined in [85, 86], a rigid body is a set of particles in which the relative distance of any two points is time invariant, regardless of the motion or forces exerted on the body. Mathematically, we may define a rigid body as in Definition 3.1 and also the rigid body’s motion, as in Definition 3.2.

Definition 3.1. (Rigid Body) [85] A rigid body is a collection of particles in which the position in time of any two given particles, such as $a(t)$ and $b(t)$, will always preserve the relation

$$\|a(t) - b(t)\| = \|a(0) - b(0)\| = K \quad \forall t. \quad (3.1)$$

In other words, at any given time the distances between particles a and b is a constant K . Thus, we consider that deformations on rigid bodies can be neglected.

We may extend Definition 3.1 to also describe the motions of rigid bodies. That is, we may create a map g between the current pose of the body’s particles to another pose, such as $g : \text{Spin}(3) \times \mathbb{R}^3 \rightarrow \text{Spin}(3) \times \mathbb{R}^3$.

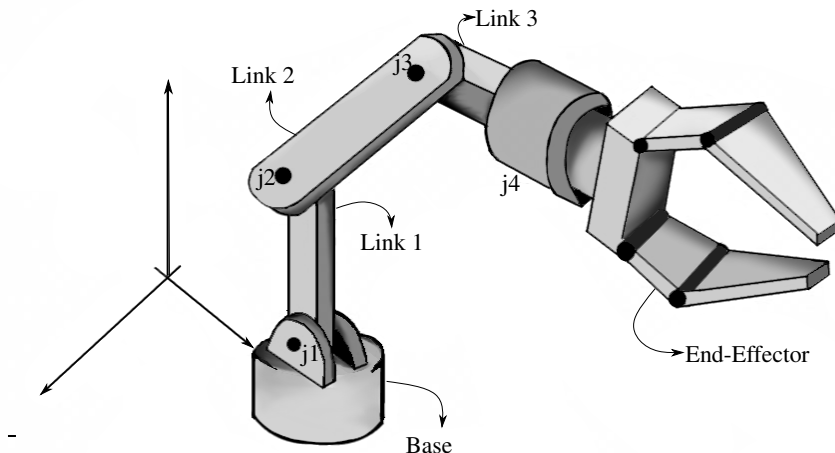


Figure 3.1: Generic illustration of a manipulator to exemplify the base, links and end-effector being connected by different joints. In this illustration the robot is located besides the world frame.

This map will be a rigid transformation if it satisfies the conditions on Definition 3.2.

Definition 3.2. (Rigid Transformation) [85] A map $g, g : \text{Spin}(3) \times \mathbb{R}^3 \rightarrow \text{Spin}(3) \times \mathbb{R}^3$ is a rigid transformation of a rigid body if the following conditions are satisfied:

1. The body's length is preserved during the motion — or (3.1) is satisfied.
2. The cross product is preserved, that is, given two points in the body, such as $\mathbf{a}, \mathbf{b} \in \text{Spin}(3) \times \mathbb{R}^3$, then

$$g(\mathbf{a} \times \mathbf{b}) = g(\mathbf{a}) \times g(\mathbf{b}). \quad (3.2)$$

Remark 3.1. The cross product condition exists to avoid internal reflection of the rigid body.

The description of a rigid body and its transformations in Definitions 3.1 and 3.2 have an important consequence for the study of rigid body kinematics: as all particles in the body preserve their distance we may choose only one point to describe its pose. Furthermore, another important theorem, this time regarding the rigid motion itself, is the Chasles Theorem, stated in Theorem (3.1)¹.

Theorem 3.1. *"The most general rigid body displacement can be produced by a translation along a line followed (or preceded) by a rotation about that line."*

Theorem 3.1 describes a motion reminiscent of the movement of a screw, and thus is referred to as a screw motion, which will be further discussed in Subsection 3.1.1.

In Section 2.2 we have started to address the issue of rigid body kinematics by describing points in space as well as their frame transformations. Thus, we shall further define the kinematics of rigid bodies in terms of translations, rotations as well as their derivatives. To ensure completeness as well as a better understanding, the Subsections 3.1.2 and 3.1.3 will tackle the kinematics of quaternions and dual quaternions.

¹The proof of Chasles Theorem is out of the scope of this manuscript, however it can be easily found in [87].

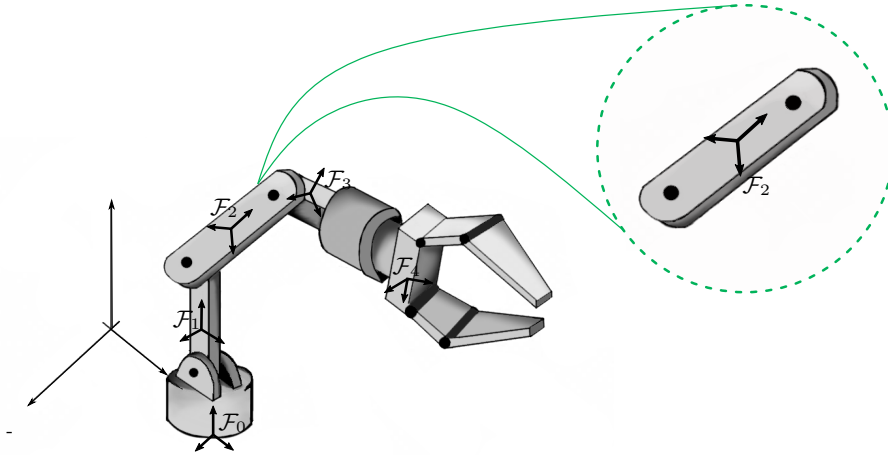


Figure 3.2: Generic illustration of a manipulator with one of its links—that is one of the rigid bodies from its structure—being highlighted.

Finally, to better illustrate the propositions presented in this section we have Figure 3.2, in which a frame is attached to each of the rigid bodies that composes the robot. We note that usually each rigid body can be completely described by the pose of its frame and the rigid transformations applied to it. There are, however, some extra constrains imposed by the joints in the manipulator, which will be discussed in Section 3.3. Here we shall present the unconstrained kinematic equations, for one single detached link, such as the one highlighted in Figure 3.2.

3.1.1 Screw Theory

Following Theorem 3.1, we have the definition of screw motions. As stated, rigid body displacements can be seen as a composition of a translation and a rotation around an axis—namely the screw axis.

The screw axis may be understood as a line composed of two concatenated three-dimensional vectors: the first part being a unit directional vector and the second (dual) vector representing the moment produced by the primary vector about a reference point in the line. Indeed, one manner to represent screws is through a combination of parameters such as

$$\mathcal{S} = \{p, l, h\}, \quad (3.3)$$

in which p is a point in the axis, l is the unit directional vector and h is the screw pitch—or ratio of the linear and angular parameters. In this representation the screw can also be represented by a combination of two \mathbb{R}^3 vectors such as $\omega, v \in \mathbb{R}^3$ resulting in

$$\mathcal{S} = \begin{bmatrix} \omega \\ v \end{bmatrix} \in \mathbb{R}^6. \quad (3.4)$$

Thus we may have two different scenarios: the first, and most general, motion has $\|\omega\| = 1$ and $v = -\omega \times q + h\omega$, with $h = 0$ in case of pure rotations; and the second represents only a translation, in which $\omega = 0$ and the pitch is infinite [49, 88].

Another way of representing a screw is through the use of dual quaternions' Plücker's representation of a

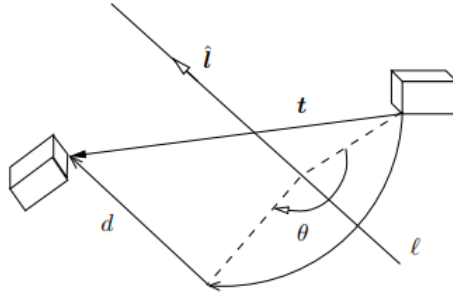


Figure 3.3: Representation of a rigid displacement: Object rotates with angle θ around line with plucker coordinates l and then moves distance d parallel to the screw axis. *Source [1].*

line, as presented in Definition 2.23. Indeed, we can intuitively see how the parameters in (3.3) and (3.4) are very similar to the Plücker coordinates in (2.86). In this case we have the screw axis being $\underline{l} = l + \varepsilon m$ with its unit direction given by l and moment m . Moreover, we may also wish to define θ as the motion's rotation angle and t as the translation with $d = t \cdot l$ being the distance of the translational motion in direction l . Also, comparable to (3.3), we can define $h = \frac{d}{\theta}$ as the pitch of line [40]. Considering these coordinates, Figure 3.3 illustrates the motion of a screw with Plücker's coordinates..

It is also important to derive the equation for the screw displacement, as presented in [1, 37, 43, 47, 80]. Thus, in order to connect the parameters θ and d , to the screw axis equation, let us consider the dual quaternion $\underline{x} = x + \varepsilon x'$ and connect it to the Plücker line's equation. In (2.81) we have the equation for a point's displacement (and also for a rigid body displacement, as will be discussed in Subsection 3.1.3) resulting in

$$\underline{x} = r + \varepsilon \frac{1}{2} r t^b = r + \varepsilon \frac{1}{2} t^o r. \quad (3.5)$$

in which t^b is the translation in the body frame and t^o is the translation relative to an inertial frame [80].

From (2.86) and (2.76) it is straightforward that the unit directional vector together with the angle θ relate in

$$x = r = \cos\left(\frac{\theta}{2}\right) + l \sin\left(\frac{\theta}{2}\right), \quad (3.6)$$

solving the problem for the primary part of the dual quaternion. Next, we have the quaternion t^o representing the translation in the inertial frame, and from (2.83) we infer that $t^o = 2x^* x'$ and, therefore, $d = 2(x^* x') \cdot l$. Finally, the moment m can be derived as in [43, 47, 80, 89], resulting in

$$m = \frac{1}{2} \left(t^o \times l + (t^o - dl) \cot\left(\frac{\theta}{2}\right) \right). \quad (3.7)$$

Multiplying both sides of (3.7) by $\sin\left(\frac{\theta}{2}\right)$ we have

$$\begin{aligned} m \sin\left(\frac{\theta}{2}\right) &= \frac{1}{2} \left(t^o \times l \sin\left(\frac{\theta}{2}\right) + (t^o - dl) \sin\left(\frac{\theta}{2}\right) \cot\left(\frac{\theta}{2}\right) \right) \\ m \sin\left(\frac{\theta}{2}\right) &= \frac{1}{2} \left(t^o \times l \sin\left(\frac{\theta}{2}\right) + (t^o - dl) \cos\left(\frac{\theta}{2}\right) \right) \\ m \sin\left(\frac{\theta}{2}\right) + \frac{d}{2} \cos\left(\frac{\theta}{2}\right) &= \frac{1}{2} \left(t^o \times l \sin\left(\frac{\theta}{2}\right) + t^o \cos\left(\frac{\theta}{2}\right) \right). \end{aligned} \quad (3.8)$$

From (2.76) we also have

$$\begin{aligned}\frac{1}{2}\mathbf{t}^o\mathbf{r} &= \frac{1}{2}\mathbf{t}^o\left(\cos\left(\frac{\theta}{2}\right) + \mathbf{l}\sin\left(\frac{\theta}{2}\right)\right) \\ \frac{1}{2}\mathbf{t}^o\mathbf{r} &= \frac{1}{2}\mathbf{t}^o\cos\left(\frac{\theta}{2}\right) + \frac{1}{2}\mathbf{t}^o\mathbf{l}\sin\left(\frac{\theta}{2}\right),\end{aligned}\quad (3.9)$$

and, as both \mathbf{l} and \mathbf{t} are pure quaternions, by using propriety (2.15) we may expand (3.9) such that

$$\begin{aligned}\frac{1}{2}\mathbf{t}^o\mathbf{r} &= \frac{1}{2}\mathbf{t}^o\cos\left(\frac{\theta}{2}\right) + \frac{1}{2}(-\mathbf{t}^o\cdot\mathbf{l} + \mathbf{t}^o \times \mathbf{l})\sin\left(\frac{\theta}{2}\right) \\ \frac{1}{2}\mathbf{t}^o\mathbf{r} &= \frac{1}{2}\mathbf{t}^o\cos\left(\frac{\theta}{2}\right) - \frac{1}{2}\mathbf{t}^o\cdot\mathbf{l}\sin\left(\frac{\theta}{2}\right) + \frac{1}{2}\mathbf{t}^o \times \mathbf{l}\sin\left(\frac{\theta}{2}\right) \\ \frac{1}{2}\mathbf{t}^o\mathbf{r} &= \frac{1}{2}\left(\mathbf{t}^o\cos\left(\frac{\theta}{2}\right) + \mathbf{t}^o \times \mathbf{l}\sin\left(\frac{\theta}{2}\right)\right) - \frac{1}{2}(\mathbf{t}^o\cdot\mathbf{l})\sin\left(\frac{\theta}{2}\right)\end{aligned}$$

As $d = \mathbf{t}^o \cdot \mathbf{l}$ and from (3.8)

$$\frac{1}{2}\mathbf{t}^o\mathbf{r} = \left(\mathbf{m}\sin\left(\frac{\theta}{2}\right) + \frac{d}{2}\cos\left(\frac{\theta}{2}\right)\right) - \frac{1}{2}(d)\sin\left(\frac{\theta}{2}\right). \quad (3.10)$$

We notice here that with the quaternion in the inertial frame. However, in the remainder of this manuscript we shall be dealing with the variables in the body frame. As shown in [80] and (3.5) the transformations to the inertial frame and the body frame are equivalent. Thus

$$\frac{1}{2}\mathbf{r}\mathbf{t}^b = \left(\mathbf{m}\sin\left(\frac{\theta}{2}\right) + \frac{d}{2}\cos\left(\frac{\theta}{2}\right)\right) - \frac{1}{2}(d)\sin\left(\frac{\theta}{2}\right). \quad (3.11)$$

Finally, we may reassemble the quaternion in (3.5) with (3.6) and (3.11). That is,

$$\begin{aligned}\underline{\mathbf{x}} &= \mathbf{r} + \varepsilon\frac{1}{2}\mathbf{r}\mathbf{t}^b \\ \underline{\mathbf{x}} &= \cos\left(\frac{\theta}{2}\right) + \mathbf{l}\sin\left(\frac{\theta}{2}\right) + \varepsilon\left(\mathbf{m}\sin\left(\frac{\theta}{2}\right) + \frac{d}{2}\cos\left(\frac{\theta}{2}\right) - \frac{1}{2}d\sin\left(\frac{\theta}{2}\right)\right) \\ \underline{\mathbf{x}} &= \left(\cos\left(\frac{\theta}{2}\right) - \varepsilon\frac{d}{2}\sin\left(\frac{\theta}{2}\right)\right) + \left(\mathbf{l}\sin\left(\frac{\theta}{2}\right) + \varepsilon\left(\mathbf{m}\sin\left(\frac{\theta}{2}\right) + \frac{d}{2}\cos\left(\frac{\theta}{2}\right)\right)\right).\end{aligned}\quad (3.12)$$

And to further simplify our equation, we may use (2.21) to obtain the sine and cosine functions of dual numbers. That is

$$\cos\left(\frac{\theta + \varepsilon d}{2}\right) = \cos\left(\frac{\theta}{2}\right) - \varepsilon\sin\left(\frac{\theta}{2}\right)\frac{d}{2} \quad (3.13)$$

and

$$\sin\left(\frac{\theta + \varepsilon d}{2}\right) = \sin\left(\frac{\theta}{2}\right) + \varepsilon\cos\left(\frac{\theta}{2}\right)\frac{d}{2}. \quad (3.14)$$

Then, we have that (3.12) becomes

$$\underline{\mathbf{x}} = \left(\cos\left(\frac{\theta + \varepsilon d}{2}\right)\right) + \sin\left(\frac{\theta + \varepsilon d}{2}\right)(\mathbf{l} + \varepsilon\mathbf{m}),$$

or, simply

$$\underline{\mathbf{x}} = \left(\cos\left(\frac{\theta}{2}\right)\right) + \sin\left(\frac{\theta}{2}\right)\underline{\mathbf{l}} \quad (3.15)$$

with

$$\underline{\theta} = \theta + \varepsilon d \quad (3.16)$$

being a dual angle.

Moreover, [90] states we may also use Taylor Expansion in order to obtain Euler's Identity for dual quaternions, resulting in

$$\exp\left(\frac{1}{2}\theta\mathbf{l}\right) \triangleq \cos\left(\frac{\theta}{2}\right) + \mathbf{l}\sin\left(\frac{\theta}{2}\right). \quad (3.17)$$

Thus, to summarize, we may introduce Definition (3.3) for the displacement of a screw.

Definition 3.3. (Screw Displacement) Given the screw axis, defined as the Plücker coordinates $\underline{\mathbf{l}} = \mathbf{l} + \varepsilon\mathbf{m}$ and the dual angle $\underline{\theta} = \theta + \varepsilon d$, with $\theta, d \in \mathbb{R}$, we can represent the line's position as the unit dual quaternion $\underline{\mathbf{x}} \in \text{Spin}(3) \times \mathbb{R}^3$ defined as

$$\underline{\mathbf{x}} = \exp\left(\frac{1}{2}\underline{\theta}\underline{\mathbf{l}}\right) = \cos\left(\frac{\theta}{2}\right) + \underline{\mathbf{l}}\sin\left(\frac{\theta}{2}\right). \quad (3.18)$$

To close this section, we should also introduce another concept that arise from the definition of a screw and screw displacement: the twists. A twist is the infinitesimal version of a screw motion, providing information for the instantaneous velocity of a body in terms of linear and angular components. Considering the twist represents the velocities around a screw axis, it can also be represented in terms of Plücker lines, that is

$$\underline{\boldsymbol{\omega}} = \boldsymbol{\omega} + \varepsilon\mathbf{v} \quad (3.19)$$

in which $\boldsymbol{\omega}$ is the body's angular velocity and \mathbf{v} is the body linear velocity [31, 37]².

3.1.2 Quaternion based representation of rigid body rotation

As discussed in subsection 2.2.1, quaternions can be used to represent both translation and rotations in space, as is shown in (2.75) and (2.79). Although translations and rotations can be used to fully describe an object, we are also interested in the differential kinematics, dealing with twists and velocities. Here we shall show, as in [28, 66], how to obtain the first derivative of a unit quaternion \mathbf{r} that represents the attitude of the body.

To start of, let us consider \mathbf{p}^0 as a rigidly fixed point in \mathcal{F}_0 such that $\dot{\mathbf{p}}^0 = 0$ and a frame transformation to the point \mathbf{p}^1 in the moving frame \mathcal{F}_1 such as in (2.77), that is

$$\mathbf{p}^1 = \mathbf{r}_1^0 \mathbf{p}^0 \mathbf{r}_0^1. \quad (3.20)$$

Then, by differentiating (3.20) we obtain

$$\begin{aligned} \dot{\mathbf{p}}^1 &= \dot{\mathbf{r}}_1^0 \mathbf{p}^0 \mathbf{r}_0^1 + \mathbf{r}_1^0 \dot{\mathbf{p}}^0 \mathbf{r}_0^1 + \mathbf{r}_1^0 \mathbf{p}^0 \dot{\mathbf{r}}_0^1 \\ \dot{\mathbf{p}}^1 &= \dot{\mathbf{r}}_1^0 \mathbf{p}^0 \mathbf{r}_0^1 + \mathbf{r}_1^0 \mathbf{p}^0 \dot{\mathbf{r}}_0^1. \end{aligned} \quad (3.21)$$

Furthermore, as $\mathbf{p}^0 = \mathbf{r}_0^1 \mathbf{p}^1 \mathbf{r}_1^0$, we may rewrite (3.21),

$$\begin{aligned} \dot{\mathbf{p}}^1 &= \dot{\mathbf{r}}_1^0 \mathbf{r}_0^1 \mathbf{p}^1 \mathbf{r}_1^0 \mathbf{r}_0^1 + \mathbf{r}_1^0 \mathbf{r}_0^1 \mathbf{p}^1 \mathbf{r}_1^0 \dot{\mathbf{r}}_0^1 \\ \dot{\mathbf{p}}^1 &= \dot{\mathbf{r}}_1^0 \mathbf{r}_0^1 \mathbf{p}^1 + \mathbf{p}^1 \mathbf{r}_1^0 \dot{\mathbf{r}}_0^1, \end{aligned} \quad (3.22)$$

²This manuscript is only dealing with the case where twists are described in terms of the body frame, and for simplicity we will omit the superscript indicating so. If we wish to change the reference frame a transformation is needed, this is also described in [31, 37].

and also create the variable $\alpha = r_1^0 \dot{r}_0^1$ to simplify (3.22). Here we note that $\alpha^* = (r_1^0 \dot{r}_0^1)^* = (\dot{r}_0^1)^* (r_1^0)^* = \dot{r}_1^0 r_0^1$ and, furthermore, that the pair $r^* r = 1$ and thus

$$\begin{aligned} (\dot{r}^* r) &= \dot{r}^* r + r^* \dot{r} = 0 \\ &= \text{Re}(\dot{r}^* r) + \text{Re}(r^* \dot{r}) = 0 \\ &= 2\text{Re}(\dot{r}^* r) = 0 \\ &= \text{Re}(\dot{r}^* r) = 0. \end{aligned} \tag{3.23}$$

As $\alpha \in \mathbb{H}_0$, this yields that

$$\begin{aligned} \dot{p}^1 &= \alpha^* p^1 + p^1 \alpha \\ \dot{p}^1 &= p^1 \alpha - \alpha p^1 \\ \dot{p}^1 &= 2(p^1 \times \alpha). \end{aligned} \tag{3.24}$$

But we know, from the discussions of Subsection 3.1.1 that the linear velocity should also be the cross product between the body's angular velocity and a point in the screw, that is $\dot{p}^1 = (p^1 \times \omega_{10}^1)$, with ω_{10}^1 being the angular velocity \mathcal{F}_0 in relation to \mathcal{F}_1 . Thus

$$\begin{aligned} (p^1 \times \omega_{10}^1) &= 2(p^1 \times \alpha) \\ \omega_{10}^1 &= 2\alpha = 2r_1^0 \dot{r}_0^1 \end{aligned} \tag{3.25}$$

which, in turn, yields the differential equation we have been looking for

$$\dot{r}_0^1 = \frac{1}{2} r_0^1 \omega_{10}^1. \tag{3.26}$$

This result can be summarized in Definition (3.4).

Definition 3.4. (Quaternion Attitude Kinematics) [28, 66] Given the unit quaternion r representing the rotation of a rigid body, we have that its first order derivative is

$$\dot{r} = \frac{1}{2} r \omega. \tag{3.27}$$

Here ω is a pure quaternion which represents the angular velocity in the body frame.

3.1.3 Dual quaternion based representation of general rigid motion

As was the case for the quaternions, we have presented how dual quaternions represent translations and rotations in space, as well as how they can represent screws. Here we shall proceed as in Subsection 3.1.2 and introduce the relation between a unit dual quaternion first order derivative and the body twist as in [28, 66].

To start, let us consider the derivative of the unit dual quaternion expressed in the body frame in (2.81)

$$\dot{\underline{x}} = \dot{r} + \varepsilon \frac{1}{2} (\dot{r} p + r \dot{p}). \tag{3.28}$$

We have from (3.27) that $\mathcal{P}(\dot{\underline{x}}) = \dot{r} = \frac{1}{2} r \omega$ in the body frame, and, moreover, that the derivative of the body's translation equals to the origin of the moving frame (body), or $\dot{p} = v$. From there we have

$$\mathcal{D}(\dot{\underline{x}}) = \frac{1}{2} \left(\frac{1}{2} r \omega p + r v \right) = \frac{1}{2} r \left(\frac{1}{2} \omega p + v \right), \tag{3.29}$$

and by reassembling the dual quaternion

$$\begin{aligned}\dot{\underline{x}} &= \frac{1}{2}\underline{r}\underline{\omega} + \varepsilon\frac{1}{2}\underline{r}\left(\frac{1}{2}\underline{\omega}\underline{p} + \underline{v}\right) \\ \dot{\underline{x}} &= \frac{1}{2}\underline{r}\left(\underline{\omega} + \varepsilon\left(\frac{1}{2}\underline{\omega}\underline{p} + \underline{v}\right)\right).\end{aligned}\quad (3.30)$$

Now we shall hypothesize that $\dot{\underline{x}} = \frac{1}{2}\underline{x}\underline{\omega}$ [31], with $\underline{\omega}^b$ being the body twist. Then we have, from (2.81), (2.25) and (2.26), that

$$\begin{aligned}\dot{\underline{x}} &= \frac{1}{2}\underline{x}[\mathcal{P}(\underline{\omega}) + \varepsilon\mathcal{D}(\underline{\omega})] \\ \dot{\underline{x}} &= \frac{1}{2}\left[\underline{r} + \varepsilon\frac{1}{2}\underline{r}\underline{p}\right][\mathcal{P}(\underline{\omega}) + \varepsilon\mathcal{D}(\underline{\omega})] \\ \dot{\underline{x}} &= \frac{1}{2}\underline{r}\left[\mathcal{P}(\underline{\omega}) + \varepsilon\left(\mathcal{D}(\underline{\omega}) + \frac{1}{2}\underline{p}\mathcal{P}(\underline{\omega})\right)\right].\end{aligned}\quad (3.31)$$

Furthermore, comparing (3.30) to (3.31) we have

$$\mathcal{P}(\underline{\omega}) + \varepsilon\left(\mathcal{D}(\underline{\omega}) + \frac{1}{2}\underline{p}\mathcal{P}(\underline{\omega})\right) = \underline{\omega} + \varepsilon\left(\frac{1}{2}\underline{\omega}\underline{p} + \underline{v}\right)$$

thus

$$\mathcal{P}(\underline{\omega}) = \underline{\omega}$$

and from (2.13),

$$\begin{aligned}\mathcal{D}(\underline{\omega}) + \frac{1}{2}\underline{p}\underline{\omega} &= \frac{1}{2}\underline{\omega}\underline{p} + \underline{v} \\ \mathcal{D}(\underline{\omega}) &= \frac{1}{2}(\underline{\omega}\underline{p} - \underline{p}\underline{\omega}) + \underline{v} \\ \mathcal{D}(\underline{\omega}) &= (\underline{\omega} \times \underline{p}) + \underline{v}\end{aligned}$$

resulting in

$$\underline{\omega} = \underline{\omega} + \varepsilon((\underline{\omega} \times \underline{p}) + \underline{v}). \quad (3.32)$$

However, as we are defining the twist in the body frame, we may rewrite (3.32) as

$$\underline{\omega}^b = \underline{\omega}^b + \varepsilon\underline{v}^b. \quad (3.33)$$

With $\underline{v}^b = (\underline{\omega} \times \underline{p}) + \underline{v}$ and $\underline{\omega}^b = \underline{\omega}$. We also note that (3.33) is equal to the body twist defined in (3.19). More so, we also have our hypothesis confirmed, resulting in the rigid body kinematics equation, which is succinctly defined in (3.5).

Definition 3.5. (Dual Quaternion Kinematics) Given the unit dual quaternion \underline{x} we have that its first order derivative is

$$\dot{\underline{x}} = \frac{1}{2}\underline{x}\underline{\omega}^b. \quad (3.34)$$

With $\underline{\omega}^b = \underline{\omega}^b + \varepsilon\underline{v}^b$, $\underline{v}^b = (\underline{\omega} \times \underline{p}) + \underline{v}$ and $\underline{\omega}^b = \underline{\omega}$ and being the twist in the body frame.

3.2 RIGID BODY DYNAMICS

In this section we shall deal with the dynamics of an unconstrained rigid body, as presented in the formulation proposed in [31]. Moreover, we shall start by introducing, in Subsection (3.2.1), Newton's equations of motion using dual quaternion algebra as well as Euler's equation for rotating a rigid body and a formulation for wrenches, furthermore, in subsection 3.2.2, we will discuss some aspects of the inertia tensor. Finally, in Subsection 3.2.3 we shall introduce the formulation for rigid body dynamics used throughout the rest of this manuscript.

3.2.1 Dynamical Equations of Motion

In this manuscript, we introduce a novel algorithm to calculate the inverse dynamics of a manipulator. Indeed, so far we have been introducing the mathematical concepts behind dual quaternion algebra as well as the very well established kinematic equations for a rigid body. However, the literature is still lagging behind on many topics regarding general concepts of dynamics with dual quaternions, which motivate us to take our time to systematically present even classic Newtonian dynamics here.

To better illustrate those classical concepts, let us look at Figure 3.4a. We note that a system of forces and torques are acting on the body, each may be represented by a pure quaternion such as $\mathbf{f}_i = f_i^x \hat{i} + f_i^y \hat{j} + f_i^z \hat{k}$ and $\mathbf{m}_i = m_i^x \hat{i} + m_i^y \hat{j} + m_i^z \hat{k}$ respectively. Moreover, we may also represent this system of forces as one single force acting on the body's center of mass [49], that is, for n forces

$$\mathbf{f}^b \triangleq \sum_{i=1}^n \mathbf{f}_i \quad (3.35)$$

and for m torques

$$\mathbf{m}^b \triangleq \sum_{i=1}^m \mathbf{m}_i. \quad (3.36)$$

One important definition we must also take into account when describing a rigid body dynamic is the center of mass itself. This point is defined as the place where, given a position \mathbf{r}_i for the point of mass m_i , the equation

$$\sum_{i=1} m_i \mathbf{r}_i \triangleq 0 \quad (3.37)$$

is satisfied³.

Another important equations in this section are the quaternion version of the classic Newton law of motion and the Euler's equation for a rotating rigid body, which will be presented in Definitions 3.6 and 3.7 respectively.

³In this manuscript, we shall define the body frame to be located in the center of mass, unless stated otherwise. If any other point is chosen, we must transform it to the center of mass before any dynamical equation presented here can be used.

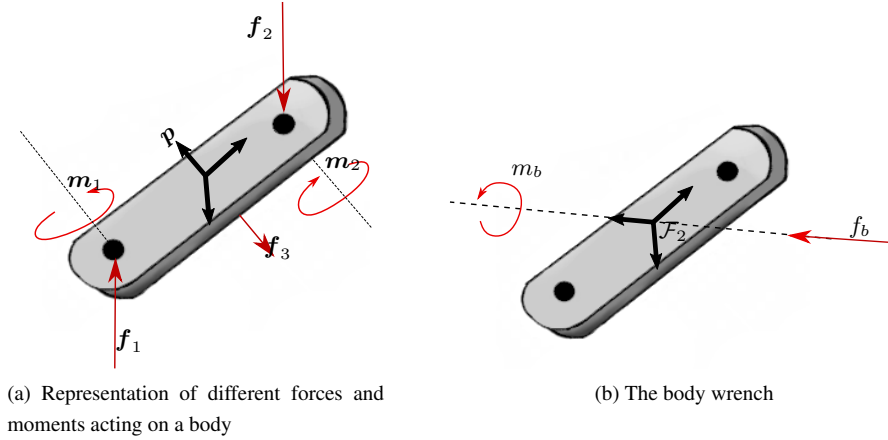


Figure 3.4: System of forces and moments acting on a body. In this illustration we have 3.4a being the representation of some of the forces and moments acting on the link (the red straight arrows represent the forces being transmitted on each joint and f_3 is the resulting gravity force the curved arrows are the moments acting at the joints). In 3.4b we have a representation of the resulting body wrench on the body (the straight arrow represents the resulting force f_b and the curved arrow represents the resulting moment m_b).

Definition 3.6. (Newton Law of Motion) [49] The classic law of motion defined by Newton can be written as

$$\mathbf{f} \triangleq m\ddot{\mathbf{p}}, \quad (3.38)$$

in which $\mathbf{f} \in \mathbb{H}_0$ is a pure quaternion representing the force acting at the center of mass, $\ddot{\mathbf{p}} \in \mathbb{H}_0$ is the second derivative of the center of mass' position (its acceleration) and $m \in \mathbb{R}^+$ is a positive scalar denoting the body's mass [33, 34].

Definition 3.7. (Euler's Equation of Rotational Motion) [33] The torque in Euler's equations for the rotational dynamics of a body in quaternion algebra is

$$\mathbf{m} \triangleq \mathcal{T}(\mathbf{I})\dot{\boldsymbol{\omega}} + \boldsymbol{\omega} \times (\mathcal{T}(\mathbf{I})\boldsymbol{\omega}), \quad (3.39)$$

in which $\mathbf{I} \in \mathbb{R}^{3 \times 3}$ is the body's inertia, $\boldsymbol{\omega} \in \mathbb{H}_0$ is the angular velocity and $\dot{\boldsymbol{\omega}} \in \mathbb{H}_0$ its first order derivative.

Remark 3.2. We note from Definition 3.7 that, although \mathbf{I} is sometimes defined as a matrix, it can be substituted by other representations, as will be discussed in more length in Subsection 3.2.2.

Remark 3.3. It is also important to stress that (3.38) and (3.39) are generic equations expressed in the body frame, but that can be easily transformed to another frame by means of an adjoint transformation as (2.78).

Finally, we shall also discuss here the definition of a wrench applied to a screw axis as illustrated in Figure 3.4b. Thus we introduce Poincot's theorem, stated bellow [83].

Theorem 3.2. "Every system of forces is equivalent to a single force, plus a couple with a moment parallel to that force."

As we can see, Theorem 3.2 is analogous to Theorem 3.1, only in terms of dynamic variables. Indeed, due to their similarity both theorems can be written in terms of screw theory, which in the case proposed by

Chasles resulted in the definition of the body twist and in Poinot's case will result in the definition of a wrench. Therefore, we may define a wrench as the pair of force and torque applied on a screw axis, which in terms of Plücker coordinates is

$$\underline{\mathbf{f}} = \mathbf{f} + \varepsilon \mathbf{m}. \quad (3.40)$$

In (3.40) \mathbf{f} represent the force and \mathbf{m} the moment applied about the body's center of mass [31, 37]⁴. As previously stated wrenches, as twists, are made out of a linear component and an angular component acting out along a screw axis, only in this case, this components are a force [as in (3.38)] and a moment [as in (3.39)] respectively.

3.2.2 Inertia Tensor

One important model in rotational dynamics is the distribution of mass in a body, given by the inertia tensor (some times also referred to as rotational inertia matrix). In its most basic form, the rotational inertia is a function of a point's position and its mass, which can be generally described by the six parameters

$$\mathbf{I} = \begin{bmatrix} \mathcal{I}_{xx} & \mathcal{I}_{xy} & \mathcal{I}_{xz} \\ \mathcal{I}_{yx} & \mathcal{I}_{yy} & \mathcal{I}_{yz} \\ \mathcal{I}_{zx} & \mathcal{I}_{zy} & \mathcal{I}_{zz} \end{bmatrix}. \quad (3.41)$$

The matrix (3.41) can be calculated through many different strategies and the literature also deals extensively with identification algorithms in order to estimate its parameters for more complicated structures [54, 91, 92]. The identification problem is a complex and important one, which should be explored in more depth in future works. In this manuscript, however, we will assume that the parameters of (3.41) are known. Furthermore, it is still important to explore other aspects pertaining the inertia tensor.

The inertia tensor, much like the mass, is a value resisting the motion of a body —or ensuring its inertia. In this manner, the inertia tensor is defined as symmetric and positive definite. Moreover, another important observation regarding \mathbf{I} is that it is calculated by taking in consideration the positions of the particles in the body, and therefore it is not constant if represented at the inertial frame. However, if we represent \mathbf{I}^b in a frame rigidly attached to a rigid body, the positions of the particles will be constants, and consequently so will the inertia tensor [49, 93].

Some authors, in order to write more concise and efficient equations, may also define a spatial inertia matrix by creating an augmented matrix $\mathcal{G} \in \mathbb{R}^{6 \times 6}$ that encloses both (3.41) and the mass. That is,

$$\mathbf{G} = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & m\mathbf{I}_3 \end{bmatrix} \quad (3.42)$$

with \mathbf{I}_3 being the 3×3 identity matrix and m the mass. Although this approach suffices for applications in \mathbb{R}^3 , the dimensions in (3.42) are incompatible with a dual quaternion by matrix multiplication as defined in Subsection 2.1.4, thus the need to adapt this solution arises. In [31] a more suitable matrix is formulated by relying on a manifold mapping from $\mathbb{H}^0 \otimes \mathbb{D}$ to \mathbb{R}^8 and its inverse mapping. In other words, they introduce the

⁴As was the case for the twist, this manuscript is only dealing with the case where wrenches are described in terms of the body frame. If we wish to change the reference frame a transformation is needed; this is also described in [31, 37].

Dual Inertia Matrix (a real 8×8 symmetric matrix) as

$$\mathcal{G}_b = \begin{bmatrix} 1 & \mathbf{0}_{1 \times 3} & 0 & \mathbf{0}_{1 \times 3} \\ \mathbf{0}_{3 \times 1} & m\mathbf{I}_3 & \mathbf{0}_{3 \times 1} & \mathbf{0}_{3 \times 3} \\ 0 & \mathbf{0}_{1 \times 3} & 1 & \mathbf{0}_{1 \times 3} \\ \mathbf{0}_{3 \times 1} & \mathbf{0}_{3 \times 3} & \mathbf{0}_{3 \times 1} & \mathbf{I}^b \end{bmatrix}. \quad (3.43)$$

Although (3.43) provides us with a great solution to write down the dynamics of a rigid body in terms of dual quaternions, there are still some issues with it. In [31] the use of (3.43) also creates the need for a complex and counter-intuitive “switch operator” to properly describe the rigid body dynamics. Thus, in our previous work [60] we have improved parts of the formulation in [31] by substituting (3.43) by our *Dual Quaternion Inertia Transformation* (see Definition 3.8), and, therefore, we were able to write the dynamical equations only in terms of multiplications and additions, as will be further discussed in Subsection (3.2.3).

Finally, we introduce in Definition 3.8 our alternative to (3.42) and (3.43).

Definition 3.8. (Dual Quaternion Inertia Transformation) The *Dual Quaternion Inertia Transformation* is defined by the operator $G : \mathbb{H}_0 \rightarrow \mathbb{H}_0$, with

$$\begin{aligned} G(\underline{\omega}^b) &\triangleq m\mathcal{D}(\underline{\omega}^b) + \varepsilon \left(\mathbf{I}_x^b \omega_x + \mathbf{I}_y^b \omega_y + \mathbf{I}_z^b \omega_z \right), \\ G(\underline{\dot{\omega}}^b) &\triangleq m\mathcal{D}(\underline{\dot{\omega}}^b) + \varepsilon \left(\mathbf{I}_x^b \dot{\omega}_x + \mathbf{I}_y^b \dot{\omega}_y + \mathbf{I}_z^b \dot{\omega}_z \right), \end{aligned} \quad (3.44)$$

in which, $\underline{\omega}^b$ and $\underline{\dot{\omega}}^b$ are the body generalized twist and accelerations, with respect to the center of mass. Also the inertia tensor of the rigid body characterized by $\mathbf{I}_x^b = I_{xx}\hat{i} + I_{yx}\hat{j} + I_{zx}\hat{k}$, $\mathbf{I}_y^b = I_{xy}\hat{i} + I_{yy}\hat{j} + I_{zy}\hat{k}$, and $\mathbf{I}_z^b = I_{xz}\hat{i} + I_{yz}\hat{j} + I_{zz}\hat{k}$, in the body frame.

Remark 3.4. We notice that the angular component of $\underline{\omega}^b$ and $\underline{\dot{\omega}}^b$ are given by $\mathcal{P}(\underline{\omega}^b) = \omega_x\hat{i} + \omega_y\hat{j} + \omega_z\hat{k}$ and $\mathcal{P}(\underline{\dot{\omega}}^b) = \dot{\omega}_x\hat{i} + \dot{\omega}_y\hat{j} + \dot{\omega}_z\hat{k}$ respectively. We also note that they can be viewed as pure quaternion representations of column vectors of (3.43).

3.2.3 Rigid Body Dynamics in Dual Quaternion Algebra

In this Subsection, we take advantage of the formulation proposed by [31] for the dynamics of an unconstrained rigid body and, together with the classical Newton-Euler equations in (3.38) and (3.39), the formulation for wrenches in (3.40) and the Dual Quaternion Inertia Transformation in Definition 3.8, we shall introduce the dynamic equations that will be used on the remainder of this manuscript.

Thus, let us consider a rigid body such as the one in Figure 3.4b, with its reference frame located at the object’s center of mass. This body is made out of a finite number of particles with positions \mathbf{p}_i and mass m_i , also the body is moving with a body twist of $\underline{\omega}^b = \omega^b + \varepsilon v^b$. From Definition 3.5 we can infer that

$$\dot{\mathbf{p}}_i = \mathbf{v}_i = \omega^b \times \mathbf{p}_i + v^b$$

and, furthermore,

$$\begin{aligned}
\ddot{\mathbf{p}}_i &= \dot{\boldsymbol{\omega}}^b \times \mathbf{p}_i + \boldsymbol{\omega}^b \times \dot{\mathbf{p}}_i + \dot{\mathbf{v}}^b \\
&= \dot{\boldsymbol{\omega}}^b \times \mathbf{p}_i + \boldsymbol{\omega}^b \times (\boldsymbol{\omega}^b \times \mathbf{p}_i + \mathbf{v}^b) + \dot{\mathbf{v}}^b \\
&= \dot{\boldsymbol{\omega}}^b \times \mathbf{p}_i + \boldsymbol{\omega}^b \times \boldsymbol{\omega}^b \times \mathbf{p}_i + \boldsymbol{\omega}^b \times \mathbf{v}^b + \dot{\mathbf{v}}^b.
\end{aligned} \tag{3.45}$$

By incorporating (3.45) in (3.38) we have the force acting on each particle,

$$\mathbf{f}_i = m_i \left(\dot{\boldsymbol{\omega}}^b \times \mathbf{p}_i + \boldsymbol{\omega}^b \times \boldsymbol{\omega}^b \times \mathbf{p}_i + \boldsymbol{\omega}^b \times \mathbf{v}^b + \dot{\mathbf{v}}^b \right). \tag{3.46}$$

Moreover, stemming from the fact that the sum of forces in a body can be substituted by a single force acting along the wrench [Theorem 3.2 and (3.35)], we have

$$\mathbf{f}^b = \sum_{i=1}^n \mathbf{f}_i = \sum_{i=1}^n m_i \left(\dot{\boldsymbol{\omega}}^b \times \mathbf{p}_i + \boldsymbol{\omega}^b \times \boldsymbol{\omega}^b \times \mathbf{p}_i + \boldsymbol{\omega}^b \times \mathbf{v}^b + \dot{\mathbf{v}}^b \right) \tag{3.47}$$

Also, by direct verification, we see that (2.13) is endowed with the propriety that $\mathbf{a} \times \mathbf{b} = -\mathbf{b} \times \mathbf{a}$. By taking advantage of this, we may rewrite (3.47) as

$$\begin{aligned}
\mathbf{f}^b &= \sum_{i=1}^n m_i \left(-\mathbf{p}_i \times \dot{\boldsymbol{\omega}}^b - \mathbf{p}_i \times \boldsymbol{\omega}^b \times \boldsymbol{\omega}^b + \boldsymbol{\omega}^b \times \mathbf{v}^b + \dot{\mathbf{v}}^b \right) \\
&= \sum_{i=1}^n m_i \left(\boldsymbol{\omega}^b \times \mathbf{v}^b + \dot{\mathbf{v}}^b \right) - \sum_{i=1}^n m_i \mathbf{p}_i \times \left(\dot{\boldsymbol{\omega}}^b + \boldsymbol{\omega}^b \times \boldsymbol{\omega}^b \right).
\end{aligned} \tag{3.48}$$

From (3.37) we also know that the sum of the products $m_i \mathbf{p}_i$ for all particles should be equal to zero in the center of mass, resulting in $\sum_{i=1}^n m_i \mathbf{p}_i \times \left(\dot{\boldsymbol{\omega}}^b + \boldsymbol{\omega}^b \times \boldsymbol{\omega}^b \right) = 0$ and

$$\mathbf{f}^b = \sum_{i=1}^n m_i \left(\boldsymbol{\omega}^b \times \mathbf{v}^b + \dot{\mathbf{v}}^b \right) = m \left(\boldsymbol{\omega}^b \times \mathbf{v}^b + \dot{\mathbf{v}}^b \right). \tag{3.49}$$

Considering the version of Newton's equation in (3.49) and Euler's equation in (3.39), also in the body frame [i.e. $\mathbf{m}^b = \mathbf{I}^b \dot{\boldsymbol{\omega}}^b + \boldsymbol{\omega}^b \times \left(\mathbf{I}^b \boldsymbol{\omega}^b \right)$] we may write the body wrench

$$\begin{aligned}
\underline{\mathbf{f}}^b &= \mathbf{f}^b + \varepsilon \mathbf{m}^b \\
\underline{\mathbf{f}}^b &= m \left(\boldsymbol{\omega}^b \times \mathbf{v}^b + \dot{\mathbf{v}}^b \right) + \varepsilon \left(\mathbf{I}^b \dot{\boldsymbol{\omega}}^b + \boldsymbol{\omega}^b \times \left(\mathbf{I}^b \boldsymbol{\omega}^b \right) \right)
\end{aligned} \tag{3.50}$$

in which, for now, we assume \mathbf{I} is a compatible inertia tensor. Furthermore, we can make use of two facts to manipulate (3.50):

Fact 3.1. *As follows directly from (2.13) $\mathbf{v}^b \times m \mathbf{v}^b = 0$*

Fact 3.2. *For any $\underline{\mathbf{q}} \in \mathbb{H}$ we have*

$$\underline{\mathbf{q}} = \mathbf{q}_P + \varepsilon (\mathbf{q}_D + \varepsilon \mathbf{q}'_D) = \mathbf{q}_P + \varepsilon \mathbf{q}_D + \varepsilon^2 \mathbf{q}'_D = \mathbf{q}_P + \varepsilon \mathbf{q}_D.$$

With Facts 3.1 and 3.2, (3.50) can become

$$\underline{\mathbf{f}}^b = m \left(\boldsymbol{\omega}^b \times \mathbf{v}^b + \dot{\mathbf{v}}^b \right) + \varepsilon \left(\mathbf{I}^b \dot{\boldsymbol{\omega}}^b + \boldsymbol{\omega}^b \times \left(\mathbf{I}^b \boldsymbol{\omega}^b \right) + \mathbf{v}^b \times m \mathbf{v}^b + \varepsilon \mathbf{v}^b \times \left(\mathbf{I}^b \boldsymbol{\omega}^b \right) \right)$$

Rearranging (3.50) we have

$$\begin{aligned}\underline{\mathbf{f}}^b &= \boldsymbol{\omega}^b \times \left(m\mathbf{v}^b + \varepsilon \left(\mathbf{I}^b \boldsymbol{\omega}^b \right) \right) + \left(m\mathbf{v}^b + \varepsilon \left(\mathbf{I}^b \dot{\boldsymbol{\omega}}^b \right) \right) + \varepsilon \left(\mathbf{v}^b \times \left(m\mathbf{v}^b + \varepsilon \left(\mathbf{I}^b \dot{\boldsymbol{\omega}}^b \right) \right) \right) \\ \underline{\mathbf{f}}^b &= (\boldsymbol{\omega}^b + \varepsilon \mathbf{v}^b) \times \left(m\mathbf{v}^b + \varepsilon \left(\mathbf{I}^b \boldsymbol{\omega}^b \right) \right) + \left(m\mathbf{v}^b + \varepsilon \left(\mathbf{I}^b \dot{\boldsymbol{\omega}}^b \right) \right).\end{aligned}\quad (3.51)$$

Moreover, as discussed in Subsection 3.2.2, there are different ways to represent the spatial inertia and not all of them are well posed for the dual quaternion problem. In [31], the matrix (3.43) requires us to switch the primary and dual parts of $\boldsymbol{\omega}^b$ and $\dot{\boldsymbol{\omega}}^b$ when performing multiplication, resulting in complicated equations which are also a costly (See Chapter 5 for a discussion on computational cost). Since the linear and angular elements of the generalized twist and body acceleration are described by pure quaternions of the form $\underline{\boldsymbol{\omega}}^b = \boldsymbol{\omega}^b + \varepsilon \mathbf{v}^b$, we can take advantage of (3.19) to rewrite (3.51) in terms of the elements of \mathbf{I}^b ,

$$\underline{\mathbf{f}}^b = \underline{\boldsymbol{\omega}}^b \times \left(m\mathcal{D}(\underline{\boldsymbol{\omega}}^b) + \varepsilon \left(\mathbf{I}_x^b \omega_x + \mathbf{I}_y^b \omega_y + \mathbf{I}_z^b \omega_z \right) \right) + \left(m\mathcal{D}(\underline{\dot{\boldsymbol{\omega}}^b}) + \varepsilon \left(\mathbf{I}_x^b \dot{\omega}_x + \mathbf{I}_y^b \dot{\omega}_y + \mathbf{I}_z^b \dot{\omega}_z \right) \right). \quad (3.52)$$

As the terms $m\mathcal{D}(\underline{\boldsymbol{\omega}}^b) + \varepsilon \left(\mathbf{I}_x^b \omega_x + \mathbf{I}_y^b \omega_y + \mathbf{I}_z^b \omega_z \right)$ and $m\mathcal{D}(\underline{\dot{\boldsymbol{\omega}}^b}) + \varepsilon \left(\mathbf{I}_x^b \dot{\omega}_x + \mathbf{I}_y^b \dot{\omega}_y + \mathbf{I}_z^b \dot{\omega}_z \right)$ in (3.52) are the definition of the Dual Quaternion Inertia Transformation in (3.44), (3.52) becomes

$$\underline{\mathbf{f}}^b = \underline{\boldsymbol{\omega}}^b \times G(\underline{\boldsymbol{\omega}}^b) + G(\underline{\dot{\boldsymbol{\omega}}^b}). \quad (3.53)$$

Thus, to summarize, we introduce Definition for the dynamics of a rigid body in terms of twists and wrenches.

Definition 3.9. (Rigid Body Dynamics) Given the twist of an object in the body frame, $\underline{\boldsymbol{\omega}}^b$, and the Dual Quaternion Inertia Transformation in (3.44) we may obtain the object's wrench $\underline{\mathbf{f}}^b$, also in the body frame from

$$\underline{\mathbf{f}}^b = \underline{\boldsymbol{\omega}}^b \times G(\underline{\boldsymbol{\omega}}^b) + G(\underline{\dot{\boldsymbol{\omega}}^b}), \quad (3.54)$$

3.3 MANIPULATOR KINEMATICS

The kinematics of a manipulator is concerned with modeling the relationships between the robot's joints and links. Thus, to address this problem we shall return to the manipulator description stated in the beginning of this chapter, and to Figure 3.1.

A manipulator—particularly a serial manipulator—is made out of a series of rigid bodies known as links which are connected to each other by a set of joints. Indeed, we have addressed extensively the free movement of one link (or rigid body) throughout Sections 3.1 and 3.2, yet we are still to explore the connections between a series of links as well as the role of the joints.

To start understanding the full mechanism of a robotic arm we must first look at the joint which is attached to a motor and therefore can provide movement to its adjacent links. Joints are mainly prismatic, revolute or cylindrical, something that can be better understood when looking at the dual angle such as (3.16) of a link performing a screw motion:

1. Revolute joints will only provide single axis rotation to the object they are attached to, thus in (3.16) $d = 0$ and the motion will be governed by $\underline{\theta} = \theta$;

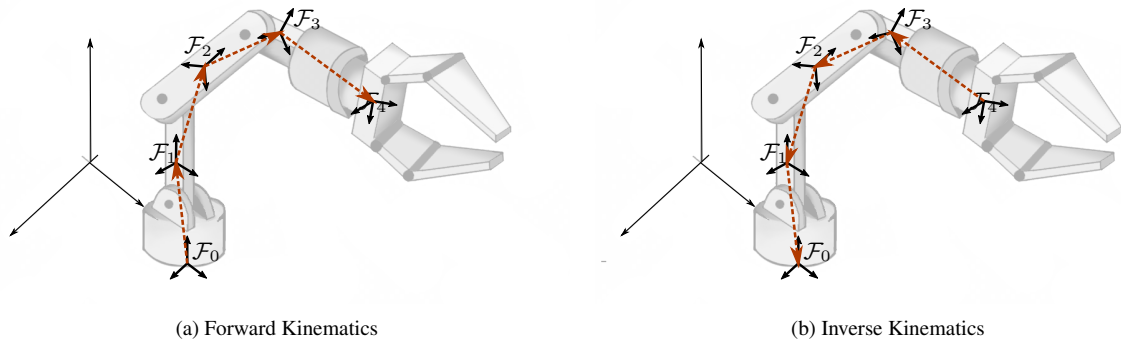


Figure 3.5: In 3.5a we have an illustration of the Forward kinematics problem, in which we aim to find end-effector pose given the joints. In 3.5b the inverse problem is illustrated, in which we want to calculate the joints based on the end-effector's pose.

2. Prismatic Joints will provide a sliding motion in one axis, thus in (3.16) $\theta = 0$ and the motion will be governed by $\underline{\theta} = \varepsilon d$;
3. Cylindrical joints will provide combination of the sliding and rotational motion of the previous joints , resulting in $\underline{\theta} = \theta + \varepsilon d$.

One important fact about the joints is that they not only provide movement, but also create constrains to the body, something that is usually accounted when we refer to the “Degrees of Freedom” (DoF) of a robot. One revolute or prismatic joints only allow for movements in one axis, thus they provide one DoF to the body, cylindrical joints, on the other hand, yields two DoFs. To reach all points in space a mechanism would usually need 6-DoFs (to allow both translation and rotation in the three orthogonal vectors), which is usually achieved by connecting the joints and the links: as 6-DoF robots are needed, we usually expect robots to have six revolute or prismatic joints. Robots with less than six joints are considered under-actuated and robots with more than six degrees of freedom are over-actuated [85].

Following the discussion on the joints, we must also tackle the relationship between them and the links in the whole chain of the robot. This problem is usually posed in terms of the Forward Kinematics (FKM) or the Inverse Kinematics (IK) models. The forward kinematics problem, illustrated in Figure 3.5a, aims at determining the end-effector configuration based on a function of the joints, this will be detailed in Subsection 3.3.1. Moreover, the inverse kinematic problem (Figure 3.5b) is, as the name suggests, the inverse problem: we aim to find the joint variable in terms of the end-effector position. Indeed, while won't be dealing with the inverse problem in this manuscript, we would like to highlight that the IK is more difficult than the direct problem, usually having multiple solutions.

Additionally, in Subsection 3.3.1, we shall be discussing the FKM problem in dual quaternion algebra as well as two different ways of parametrization for the equations.

3.3.1 Forward Position Kinematics Modeling

The forward position kinematics, as illustrated in Figure 3.5a, is a mapping of the joint configuration of an n -joint robot to its end-effector pose, such that, we have the function $\underline{f}_{\text{FKM}} : \mathbb{D}^n \rightarrow \text{Spin}(3) \times \mathbb{R}^3$,

$$\underline{\mathbf{x}}_{\text{EF}} = \underline{f}_{\text{FKM}}(\underline{\boldsymbol{\theta}}). \quad (3.55)$$

Here $\underline{\mathbf{x}}_{\text{EF}} \in \text{Spin}(3) \times \mathbb{R}^3$ is the end-effector pose and $\underline{\boldsymbol{\theta}} \in \mathbb{D}^n$ is the joint configuration vector defined as $\underline{\boldsymbol{\theta}} = \begin{bmatrix} \theta_1 & \theta_2 & \theta_3 & \dots & \theta_n \end{bmatrix}^T$. In order to define the function (3.55) we should consider each intermediate link's pose relative to the previous one, and through a number of successive rigid transformations up to the end of the chain we may obtain the end-effector pose itself—note that a sequence of rigid motions can be represented by a sequence of unit dual quaternion multiplications [28, 66].

Therefore, let $\underline{\mathbf{x}}_{i+1}^i$ be a transformation from link i to link $i + 1$, if we consider an n -joint serial chain we may obtain the end-effector's pose as

$$\underline{\mathbf{x}}_{\text{EF}} = \underline{\mathbf{x}}_1^0 \underline{\mathbf{x}}_2^1 \dots \underline{\mathbf{x}}_{n+1}^n. \quad (3.56)$$

We highlight here that $\underline{\mathbf{x}}_{i+1}^i$ is a function of θ_i , such as $\underline{\mathbf{x}}_{i+1}^i = \underline{f}_{\text{FKM},i}(\theta_i)$, which in turn results in (3.56) being a function of $\underline{\boldsymbol{\theta}} = \begin{bmatrix} \theta_1 & \theta_2 & \theta_3 & \dots & \theta_n \end{bmatrix}$ as described in (3.55).

Finally, in order to properly compute (3.55) we must define a convention to describe the pose of each link relative to the last one. Indeed the literature provides us with quite a few solutions for the parametrization of $\underline{\mathbf{x}}_{i+1}^i$, the most typical one being the Denavit-Hartenberg (D-H) parameters. However, in this manuscript we have chosen to use an alternative, less “famous” solution: the Product of Exponentials (PoE). In the next few subsections we shall detail a bit more of both the D-H parameters and the Product of exponentials, and furthermore, we shall also compare both of them in order to make our case for why the PoE formulation is more suited for this work.

3.3.1.1 Denavit-Hartenberg parameters

The basic idea behind the D-H parameters is to derive the forward kinematics by attaching frames to each of the robot's links and then calculating its displacement through a set of four parameters, as exemplified in [36, 37].

The first step to calculate (3.56) through this method is allocating the frames in the robot, something that cannot be done arbitrarily in this case. The D-H convention has a set of rules to systematically assign frames to links:

Rule 1: First we establish a base coordinate system, here we will use $\mathcal{F}_0 = \{\hat{x}_0, \hat{y}_0, \hat{z}_0\}$ coinciding with the robot's base.

Rule 2: Then we establish the joint axis. The \hat{z}_i axis of \mathcal{F}_i is aligned with the $i + 1$ joint axis, the \hat{z}_{i+1} axis of the frame \mathcal{F}_{i+1} coincides with the axis of joint $i + 2$ and so forth. The direction of a positive rotation around each \hat{z} axis is defined by the right hand rule.

Rule 3: The next step is defining origin O_i of each of the frames. Thus, we find the line segment that orthogonally intersects both \hat{z}_i and \hat{z}_{i+1} and place frame O_{i+1} there (alternatively we may place O_{i+1} at the intersection of the common normal between \hat{z}_i and \hat{z}_{i+1} and the \hat{z}_i axis).

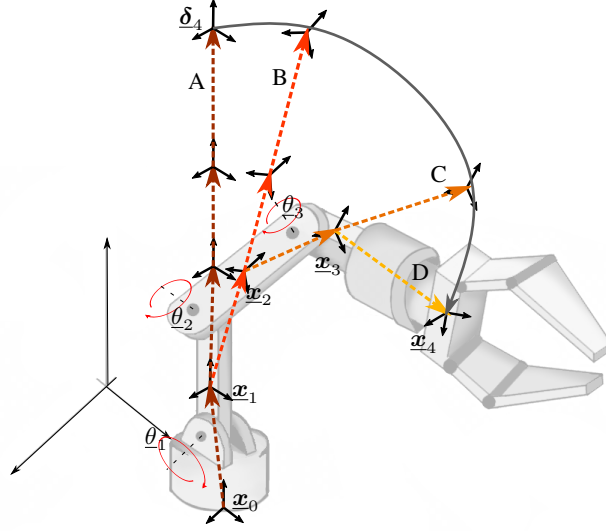


Figure 3.6: Representation of the product of exponentials formula. Here we show the screw transformation for each of the links. In A we have the robot at its home position, with all $\theta_i = 0$, from B-D we are moving one link at a time until the final robot configuration is reached in D.

Rule 4: Next we must place \hat{x}_{i+1} in the direction of the common normal between \hat{z}_i and \hat{z}_{i+1} .

Rule 5: Finally we determine \hat{y}_i at the direction satisfying the right hand rule, that is $\hat{x}_i \times \hat{y}_i = \hat{z}_i$.

Once the frames are assigned to all links we will be able to calculate the four D-H parameters and thus the transformation between the consecutive links. Hence, the transformation \underline{x}_{DH} is defined as

$$\underline{x}_{DH} = \mathbf{r}_{\hat{z},\theta} \underline{p}_{\hat{z},d} \underline{p}_{\hat{x},a} \mathbf{r}_{\hat{x},\alpha}, \quad (3.57)$$

with the parameters being

$$\begin{cases} \mathbf{r}_{\hat{z},\theta} = \cos \frac{\theta}{2} + \sin \frac{\theta}{2} \hat{k} & \text{The quaternion rotation of } \theta \text{ along the } \hat{z} \text{ axis;} \\ \underline{p}_{\hat{z},d} = 1 + \varepsilon \left(\frac{1}{2} d \right) \hat{k} & \text{The pure translation } d \text{ along the } \hat{z} \text{ axis;} \\ \underline{p}_{\hat{x},a} = 1 + \varepsilon \left(\frac{1}{2} a \right) \hat{i} & \text{The pure translation } a \text{ along the } \hat{x} \text{ axis;} \\ \mathbf{r}_{\hat{x},\alpha} = \cos \frac{\alpha}{2} + \sin \frac{\alpha}{2} \hat{i} & \text{The quaternion rotation of } \alpha \text{ along the } \hat{x} \text{ axis.} \end{cases}$$

Finally, by substituting each \underline{x}_{i+1}^i in (3.56) by transformation (3.57) we may obtain the end-effector's pose (as well as the pose of all frames placed throughout the manipulator).

3.3.1.2 Product of Exponentials

One alternative to the Denavit-Hartenberg parameters is the product of exponentials formula (PoE) [40,49]. In this formulation we take key concepts of screw theory in order to model manipulator: in other words, here we regard each joint as applying a screw motion to all the outward links. Moreover, to better contextualize the PoE let us look at Figure (3.6) throughout the remainder of this section.

We start out the process of calculation the PoE based transformation by placing the robot at the home configuration, that is, the set up $\underline{\theta}^h$ in which all joints are at their initial position—this is represented by A in

Figure 3.6. At the home configuration, we also define $\underline{\delta}_{i-1}^i$ as the dual quaternion representing the configuration of \mathcal{F}_{i-1} expressed in the frame \mathcal{F}_i , where \mathcal{F}_0 is the base frame coinciding with the robot's base and \mathcal{F}_{n+1} is the reference frame attached to the end-effector. We note here that the PoE does not impose any constraints in regards to the positioning of the intermediate frames, however, as we are dealing with dynamic equations in this manuscript, it becomes interesting to allocate the frames \mathcal{F}_1 to \mathcal{F}_n at the center of mass of each link. Furthermore, we can represent the transformation from one frame to another as $\underline{\delta}_i^{i-1} = (\underline{\delta}_{i-1}^0)^* \underline{\delta}_i^0$ and $(\underline{\delta}_i^{i-1})^* = \underline{\delta}_{i-1}^i = (\underline{\delta}_i^0)^* \underline{\delta}_{i-1}^0$.

Hence, the rigid transformation from robot's base to its end-effector, $\underline{\delta}_{EF}$, in the home configuration can be given by

$$\underline{\delta}_{EF} = \underline{\delta}_1^0 \underline{\delta}_2^1 \dots \underline{\delta}_{n+1}^n. \quad (3.58)$$

As all angles in (3.58) are in their initial state the transformation becomes very simple. However, we aim to obtain a more general model of the robot when the joints are actuating. To this end we must return to the concept outlined in the beginning of this subsection: that the PoE regards each joint as applying a screw motion over the links. Thus, we proceed by taking the concepts of Subsection 3.1.1.

Still at the home position, we define the screw axis of the links as the Plucker line $\underline{s}_i = \omega_i + \varepsilon v_i$, with reference to the base frame. In this axis ω_i and v_i are the angular and linear velocity, respectively, and, if joint i is a revolute joint, p_i is a point located on the screw axis such that $v_i = p_i \times \omega_i$. As this representation is in frame \mathcal{F}_0 , so we also need to transform it to each link's own frame of reference, thus we use the adjoint expression in (2.89) to make this change,

$$\underline{a}_i = Ad(\underline{\delta}_0^i) \underline{s}_i. \quad (3.59)$$

Finally, with all screw axis in place we only need to move them over the dual angle displacement $\Delta\theta_i = \theta_i(t) - \theta_i^h$. Thus, from (3.17) we have the pose transformation of the link i expressed in frame \mathcal{F}_{i-1} ,

$$\underline{x}_i^{i-1}(\theta_i(t)) = \underline{\delta}_{i-1}^i \exp\left(\underline{a}_i \frac{\Delta\theta_i}{2}\right). \quad (3.60)$$

As we can see in Figure 3.6 we may displace one joint at a time, through a screw, motion until we get to the final pose. Indeed this process is an illustration of (3.56) in which each displacement is given by (3.60). Thus, it becomes clear the end-pose stems from (3.56).

3.3.1.3 Comparison

In [28,40,94], we see different recursive techniques being presented to solve the forward kinematics model (FKM) of a serial manipulator based on dual quaternions. The DH parameters are indeed the most popular way to describe the transformations between one joint and another, as we do not need to be overly concerned with frame placements and the robot model is very easily obtained. However, many applications demand the PoE solution, and in particular, robotic systems described with dual quaternions benefit from it.

In [40], the authors makes a compelling case for the screw theory alternative. As it is argued, this approach based on line transformations [95] is considerably more efficient than the other techniques such as the computation of the FKM based on DH approach. More so, the screw representation is much more intuitive and the position of the frames can be chosen without the many restrictions that DH parameters impose, thus each link's frame of reference can be at the center of mass of the body, which simplifies the dynamical equations.

Of course, both representations have advantages and disadvantages. However, aiming to explore cost efficient solutions, we believe the screw based approaches are more suited to the work being explored in the scope of this manuscript.

4

DUAL QUATERNION RECURSIVE NEWTON EULER ALGORITHM (DQRNEA)

In this chapter we shall continue our investigation over the equations that govern the motions of a robot and, particularly, how they may be written in the dual quaternion formulation. Indeed, as was shown in Chapter 3, a lot has been published regarding the kinematics representation of open chains in dual quaternion algebra and on the dynamics of rigid bodies. Nonetheless, the literature is still quite lacking in terms of the representation of manipulators with the algebra of dual quaternions.

In this context, we aim at introducing, to the best of the author's knowledge, a novel, computationally efficient algorithm for modeling the dynamics of a serial chain based on dual quaternion algebra: the Dual Quaternion Recursive Newton Euler Algorithm (dqRNEA). In Section 4.1, we shall first explore the preliminary concepts relating to the dynamics of serial chains, introducing the two main approaches to solving this problem. In Section 4.2, we shall outline the algorithm we have developed and show its derivation stemming from the concepts presented in Chapter 3. As will be discussed, the dqRNEA is a recursive algorithm, thus consisting of many different steps, however, in Section 4.3 we show that with the aid of an alternative formulation for the adjoint map we are able to rewrite the dqRNEA in a single closed equation. Finally, in order to better validate our models we propose a simple dqRNEA based torque controller, as will be discussed in Section 4.4.

4.1 PRELIMINARIES

Due to the unstructured nature of the world, the study of forces that govern the motion of a robotic manipulator are paramount in reliable and robust applications. Indeed, to deal with kinematics alone implies that models should reflect faithfully all facets of an ever-changing and many times unknown environment, which is mostly unfeasible. In this sort of scenario we recur to the derivation of the dynamic equations of motion, which are extremely important to most areas of robotic manipulation, including force control, mechanical design and implementation of simulation models [5, 27].

In its most general form, the dynamic model can be expressed as a set of nonlinear, second-order, ordinary differential equations which are depended on the inertial and kinematic properties seen, respectively, in Sections 3.2 and 3.3 of this manuscript. Furthermore, also as discussed in Section 3.3, was the distinction between the forward kinematics problem and the inverse kinematics. As is the case in kinematics, the dynamical analysis of a serial manipulator includes the distinction between direct (or forward) dynamics (FDY) and inverse dynamics (IDY) [49]. The forward problem takes the form

$$\ddot{\theta} = f_{\text{FDY}}(\tau, \theta, \dot{\theta}), \quad (4.1)$$

in which we input the torque (τ), velocity ($\dot{\theta}$) and configuration (θ) of the robot's joints in function $f_{\text{FDY}}(\bullet)$, and obtain the robot's acceleration ($\ddot{\theta}$). The inverse problem is of the form

$$\tau = f_{\text{IDY}}(\theta, \dot{\theta}, \ddot{\theta}). \quad (4.2)$$

in which the inputs comprise of the joint configuration, velocity and acceleration and $f_{\text{IDY}}(\bullet)$ results in the joint's torques.

The inverse dynamic function $f_{\text{IDY}}(\bullet)$ can be written in terms of the mass function $M(\theta)$, which is a positive definite invertible matrix, the combined Coriolis and centrifugal terms $c(\theta, \dot{\theta})$ and a gravity function $g(\theta)$ [49, 85], becoming

$$\tau = f_{\text{IDY}}(\theta, \dot{\theta}, \ddot{\theta}) = \underline{\mathbf{M}}(\theta)\ddot{\theta} + \underline{\mathbf{c}}(\theta, \dot{\theta}) + \underline{\mathbf{g}}(\theta). \quad (4.3)$$

Furthermore, as (4.1) and (4.2) are inversions of each other, we may rewrite the direct dynamics function as an inversion of (4.3),

$$\ddot{\theta} = f_{\text{FDY}}(\tau, \theta, \dot{\theta}) = \underline{\mathbf{M}}^{-1}(\theta) \left[\tau + \underline{\mathbf{c}}(\theta, \dot{\theta}) + \underline{\mathbf{g}}(\theta) \right]. \quad (4.4)$$

In this section we will better explore the basic concepts of both the inverse and direct dynamics. As is the focus of this manuscript we will particularly explore the inverse problem and some of the different methods for solving it: namely the Lagrange-Euler formulation is presented in Subsection 4.1.1.1 and the Recursive Newton-Euler formulation is introduced in Subsection 4.1.1.2. We also take our time to compare both methods in Subsection 4.1.1.3.

4.1.1 Inverse Dynamics

The inverse dynamics problem can be solved throughout many different methods, such as Kane’s method, the principal of virtual work, the Lagrange-Euler (LE) formulation, the Newton-Euler Inverse dynamic algorithm (RNEA), etc. Still, although many options do exist, the two most common solutions for this problem are either the LE formulation and the RNEA. Here we shall present the basic concepts behind both formulations, as well as comparison between them.

4.1.1.1 Lagrange-Euler Formulation

One of the methods for computing the inverse dynamics of a manipulator are the Lagrange-Euler equations, which describe a robot in terms of the the work and energy stored in that system.

In order to determine the LE equations, one must first calculate the Lagrangian, defined as the difference between the kinetic ($\mathcal{K}(\theta, \dot{\theta})$) and potential ($\mathcal{P}(\theta)$) energies,

$$\mathcal{L}(\theta, \dot{\theta}) = \mathcal{K}(\theta, \dot{\theta}) - \mathcal{P}(\theta) \quad (4.5)$$

and, from (4.5), solve the equation of motion—or the Euler-Lagrange equations—expressed as

$$f = \frac{d}{dt} \frac{\partial \mathcal{L}}{\partial \dot{\theta}} - \frac{\partial \mathcal{L}}{\partial \theta}. \quad (4.6)$$

This manuscript will not delve into the details regarding this solution, however, we remark here that although this formalism seems to be simple and elegant, it does become cumbersome and inefficient as the degrees of freedom of the robot start to increase [49].

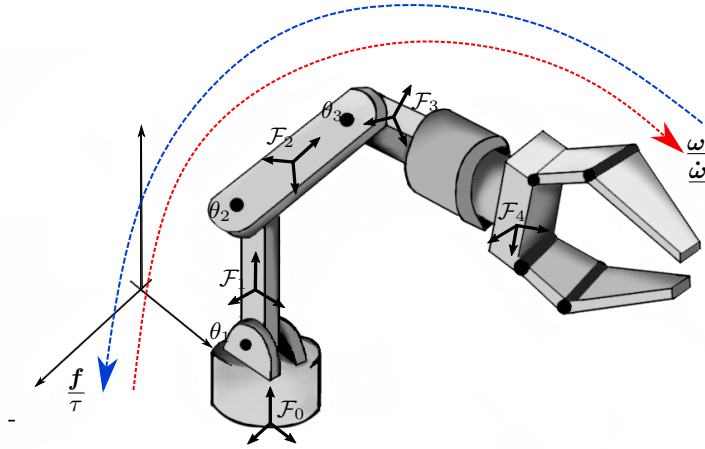


Figure 4.1: Illustration of the steps taken to compute the RNEA in a three link manipulator.

4.1.1.2 Recursive Newton-Euler Formulation

The main method for computing the inverse dynamic equations in this manuscript is the recursive Newton-Euler algorithm. This method should lead to the same torque as the Lagrangian method presented above, however by taking a completely different approach: in the case of the RNEA, we exploit the balance of forces in each of the robot's links, taken one at a time.

To better explain the algorithm, let us look once again at our three-link revolute robot illustration in Figure 4.1. In the case of the RNEA we shall analyze one link at a time, writing down the equations that describe its angular and linear motion. Thus let us start looking at the base of our manipulator—the frame \mathcal{F}_0 . If the base is fixed (and let us consider it is), then the initial velocity in the base is zero as there are no joints allowing it any degree of freedom. In this case, we will have the gravity acting on it, so we have that the acceleration is equal to it.

We then move to our first link, in the frame \mathcal{F}_1 . This link's velocities are given by the sum of the twist provoked by its joint (θ_1) and the velocities inherited by the base, which for a fixed base are zero. Thus the first link of our fixed based robot will have the twist being equal to $\underline{\omega}_1 = \underline{a}_1 \Delta \dot{\theta}_1(t)$ (here \underline{a} is the link's screw and $\underline{\omega}_1$ its twist. To see this equation in context refer to Subsection 4.2.1).

Next, to the second link, in frame \mathcal{F}_2 . This will follow the same idea as the first one: the twist will be a sum of the velocities generated by joint θ_2 in the screw axis of link 2 and the velocities inherited from the previous links—or more specifically, the velocity of the previous link, which should already account for the velocity of the ones prior to it. One important remark here is that the inherited twist should be added in the frame of the link we are considering, thus requiring a frame transformation. In this case we will therefore have $\underline{\omega}_2 = \underline{a}_2 \Delta \dot{\theta}_2(t) + \underline{\omega}_1^2$. The last link will also follow the same procedure resulting in $\underline{\omega}_3 = \underline{a}_3 \Delta \dot{\theta}_3(t) + \underline{\omega}_2^3$.

The process of going from the base to the top of the manipulator recursively accounting for the twists (as well as their first derivatives) is the **forward recursion** of the RNEA. The next step would be going back, now from the tip of the manipulator, where an external force might be acting on the end-effector, back to the base, this time calculating the forces on the links. This second part is the **Backward Recursion** of the algorithm.

For the manipulator in Figure 4.1 let us start from the End-Effector, in \mathcal{F}_4 where we consider initialize the forces with the external forces acting on this point. Thus, these external wrenches will be equal \underline{f}_4 .

Going down to link 3, back in \mathcal{F}_3 , we now have to consider the balance of the forces, that is, according to Newton’s law, the resulting forces in a body should be the sum of all forces acting on it. Thus, considering a reduced system in which only the forces in the joints are acting in the system we will have, for link 3, that the resulting force in the link is the force acting on joint 3 minus the force acting on the end effector (expressed in the frame \mathcal{F}_3) $\underline{f}_{R,3} = \underline{f}_3 - \underline{f}_4^3$. Here we highlight that the resulting wrench equation is given by (3.54).

The process for link 3 will be repeated for links 2 and 1, respectively $\underline{f}_{R,2} = \underline{f}_2 - \underline{f}_3^2$, $\underline{f}_{R,1} = \underline{f}_1 - \underline{f}_2^1$. We also add that with each force calculated we can also obtain the torques and all the equations here shall also be explored in more detail in Section 4.2.

4.1.1.3 Comparison

One important question we should try to answer in regards to the two formulations presented above is which one should we use, and how do they compare. Indeed, it becomes clear from the descriptions above the two methods have widely varying philosophies: while the Lagrange takes the manipulator as a whole and calculate the forces and torques through the energy of the system, the RNEA uses the mechanical constraints and the transmission of forces and velocities through the links in order to balance the equations and describe the motion.

Historically, both formulations evolved in parallel [93] each with its advantages and each providing different insights to the mechanics of the manipulator. Due to some of those insights we have chosen the Newton-Euler equations rather than the LE equations: we believe the dual quaternion formulation fits better with the former, especially as we aim to obtain a cost efficient representation of the manipulator. Indeed, as discussed in the previous chapters, dual quaternions are particularly good to represent geometric variables and they are the most efficient manner to represent a screw, and the RNEA exploits a lot of those characteristics.

4.2 ALGORITHM DEVELOPMENT

In this section we introduce our formulation for a dual quaternion based recursive Newton-Euler inverse dynamics algorithm (dqRNEA), as was proposed in our recent paper [60]. Our aim with this formulation is to build upon the many works that detail the kinematic models and controllers within the dual quaternion framework [45, 46, 96–101] and expand their possibilities to also encompass the dynamics of the manipulator, opening the dual quaternion framework up for a number of more complex and robust possibilities. Of course, as our aim is to use the dual quaternion representation for dynamic modeling, we also aim at ensuring our algorithm retains the dual quaternion’s advantages for the kinematics representation—particularly its cost effectiveness when compared to other algebras, the lack of representation singularities and its capabilities to succinctly represent rigid body pose, velocities, accelerations, momentum and wrenches.

In order to ensure a better understanding of the dqRNEA, we have outlined and discussed, throughout the manuscript, the many building blocks for our formulation. Chapter 2 introduces the mathematical concepts behind dual quaternions, and moreover, it outlines the advantages of this framework. Particularly it delves into the use of dual quaternions for the motions of points and lines in space. Chapter 3 expands the concepts point and line representation to rigid body kinematics, and in Section 3.3 we make our case for using the product of exponentials formulation rather than the DH parameters when representing an open chain manipulator. We

reiterate here that the screw description is particularly useful in the task we aim to accomplish, as it gives us the ability to place the link's frames at the center of mass location, thus making it possible to use the rigid body dynamic equations presented in Section 3.2 without the need to translate the frame (which consequently makes the algorithm equations simpler). Finally, in Section 4.1 he have also presented two of the main forms of computing the inverse dynamics, and also we have made our case in Subsection 4.1.1.3 as to why we have chosen to explore the Newton-Euler formulation in this manuscript.

As was introduced in Subsection 4.1.1.2 the Recursive Newton-Euler inverse dynamics algorithm is composed of two steps the forwards iteration and backwards iteration, which will be derived in Subsections 4.2.1 and 4.2.2 respectively. Furthermore, in Subsection 4.2.3 we summarize the whole dqRNEA algorithm.

Finally, to aid our description of this algorithm throughout this section we have Figure 4.1 showing the forward and backward iterations. Furthermore, we would also like to highlight that throughout the reminder of this manuscript we shall consider manipulators with revolute joints only, thus simplifying the dual angle $\underline{\theta} = \theta + \varepsilon 0$. However we point out that the extension for manipulators with prismatic or cylindrical joints (in which $d \neq 0$) should be simple and intuitive.

4.2.1 Forward Recursion

The forward iteration of the dqRNEA aims at calculating recursively the positions, velocities and accelerations the links are subjected to, starting at the robot's base, and going up to its end-effector. We take inspiration from the different recursive techniques for calculating the FKM explored in Section 3.3. Moreover, we take a systematic approach in presenting this recursion, dividing it in algorithm initialization, recursive position, recursive twist and recursive acceleration.

4.2.1.1 Algorithm Initialization

In this section, we use the PoE formulation to describe the configuration of each of the manipulator's links, and, therefore, we take into account the forward kinematics process described in subsection 3.3.1.2. To use the product of exponentials formulation, we must first place the robot in its home configuration with all joints at their initial position, that is, for an n -joint robot we will have the initial position vector $\theta^h \in \mathbb{R}^n$.

We then place frame \mathcal{F}_0 in the robot base, frames \mathcal{F}_1 to \mathcal{F}_n at each of link's center of mass and \mathcal{F}_{n+1} at the end-effector's center of mass. Next, as in Subsection 3.3.1.2, we have the transformation from frame \mathcal{F}_{i-1} to frame \mathcal{F}_i defined as $\underline{\delta}_i^{i-1}$ and its inverse transformation being $(\underline{\delta}_i^{i-1})^* = \underline{\delta}_{i-1}^i$.

Finally, before moving to the next steps we must also initialize some of the variables used throughout the algorithm:

1. The base velocity will be initialized as zero: $\underline{\omega}_0 = 0$;
2. The base acceleration should be equal to the gravity: $\underline{\dot{\omega}}_0 = -\varepsilon g$
3. The wrench applied at the end effector $\underline{f}_{n+1} = \underline{f}_{EF} = \underline{f}_{EF} + \varepsilon m_{EF}$.

This is perhaps the most important step of the dqRNEA algorithm; so, all variables should be initialized with great care. A poorly modeled robot will certainly hinder the results of this algorithm, producing erroneous

results. In Chapter 6 models of a few robots will be presented and we will discuss some project considerations regarding the computational implementation of this algorithm.

4.2.1.2 Recursive Position and Twist

The recursive calculation for the position is done via the PoE formula of Subsection 3.3.1.2. As the procedure for obtaining the link's positions has already been thoroughly explained, we shall only summarize it here.

First we define the screw of link i in frame \mathcal{F}_0 as $\underline{s}_i = \underline{\omega}_i + \varepsilon \underline{v}_i$, with $\underline{\omega}_i$ and \underline{v}_i being respectively its angular and linear velocities. Then, as in (3.59), we apply an adjoint transformation to represent the screw in \mathcal{F}_i , that is $\underline{a}_i = Ad(\underline{\delta}_0^i) \underline{s}_i$. From that we have the exponential formula of (3.60),

$$\underline{x}_i^{i-1}(\theta_i(t)) = \underline{\delta}_{i-1}^i \exp\left(\underline{a}_i \frac{\Delta\theta_i}{2}\right).$$

Finally, we recall that the forward kinematics of a serial chain is given by the multiplication of the link's transformations, given by (3.56): $\underline{x}_{EF} = \underline{x}_1^0 \underline{x}_2^1 \dots \underline{x}_n^{n-1}$.

Regarding the recursive calculation of the twist, we have that for serial chains the twist of a particular link is the sum of the twist at previous links (propagated from base to end-effector) added to the twist generated by the velocities of that link's joint ($\underline{a}_i \Delta\dot{\theta}_i(t)$). Thus, for a twist $\underline{\omega}_i$ at link i , we have

$$\underline{\omega}_i = \underline{a}_i \Delta\dot{\theta}_i(t) + Ad(\underline{x}_i^{i-1}) \underline{\omega}_{i-1}, \quad (4.7)$$

in which we must include the transformation $Ad(\underline{x}_i^{i-1}) \underline{\omega}_{i-1}$ in order to ensure $\underline{\omega}_{i-1}$ is represented at \mathcal{F}_i .

4.2.1.3 Recursive Acceleration

The links accelerations may also be calculated through recursive iterations, propagating from the base to the end-effector. In particular, its formulation can be obtained by finding the derivative of (4.7). That is

$$\underline{\dot{\omega}}_i = \frac{d}{dt} \left(\underline{a}_i \Delta\dot{\theta}_i(t) + Ad(\underline{x}_i^{i-1}) \underline{\omega}_{i-1} \right). \quad (4.8)$$

To solve (4.8) we must take into account that \underline{a}_i and $\underline{\delta}_i^{i-1}$ are constants, and, as per (2.89), we may rewrite the adjoint term as

$$Ad(\underline{x}_i^{i-1}) \underline{\omega}_{i-1} = \underline{x}_i^{i-1} \underline{\omega}_{i-1} \underline{x}_{i-1}^i \quad (4.9)$$

thus

$$\begin{aligned} \underline{\dot{\omega}}_i &= \underline{a}_i \Delta\ddot{\theta}_i(t) + \frac{d}{dt} (\underline{x}_i^{i-1} \underline{\omega}_{i-1} \underline{x}_{i-1}^i) \\ &= \underline{a}_i \Delta\ddot{\theta}_i(t) + \underline{x}_i^{i-1} \underline{\dot{\omega}}_{i-1} \underline{x}_{i-1}^i + \underline{\dot{x}}_i^{i-1} \underline{\omega}_{i-1} \underline{x}_{i-1}^i + \underline{x}_i^{i-1} \underline{\omega}_{i-1} \underline{\dot{x}}_{i-1}^i. \end{aligned} \quad (4.10)$$

In order to simplify (4.10) we can look at the first derivative of (3.60),

$$\begin{aligned} \underline{\dot{x}}_{i-1}^i &= \frac{d}{dt} \left(\underline{\delta}_i^{i-1} \exp\left(\underline{a}_i \frac{\Delta\theta_i}{2}\right) \right) \\ &= \underline{\delta}_i^{i-1} \exp\left(\underline{a}_i \frac{\Delta\theta_i}{2}\right) \frac{d}{dt} \left(\underline{a}_i \frac{\Delta\theta_i}{2} \right) \end{aligned}$$

and by going back to (3.60) itself,

$$\underline{\dot{\mathbf{x}}}_{i-1}^i = \frac{1}{2} \underline{\mathbf{x}}_{i-1}^i \underline{\mathbf{a}}_i \Delta \dot{\theta}_i(t). \quad (4.11)$$

In order to solve for $\underline{\dot{\mathbf{x}}}_i^{i-1}$ we can recall the inverse propriety of an unit dual quaternion in (2.44). As the inverse changes the UDQ's direction, $\underline{\dot{\mathbf{x}}}_i^{i-1}$ can be rewritten as $(\underline{\dot{\mathbf{x}}}_{i-1}^i)^* = \underline{\dot{\mathbf{x}}}_i^{i-1}$, and consequently

$$\begin{aligned} \underline{\dot{\mathbf{x}}}_i^{i-1} &= \left(\frac{1}{2} \underline{\mathbf{x}}_{i-1}^i \underline{\mathbf{a}}_i \Delta \dot{\theta}_i(t) \right)^* \\ &= \frac{1}{2} \Delta \dot{\theta}_i(t) \underline{\mathbf{a}}_i^* \underline{\mathbf{x}}_i^{i-1}. \end{aligned} \quad (4.12)$$

With the conjugate propriety for the set of pure dual quaternion in (2.38) we have that $\underline{\mathbf{a}}_i^* = -\underline{\mathbf{a}}_i$, thus (4.12) becomes

$$\underline{\dot{\mathbf{x}}}_i^{i-1} = -\frac{1}{2} \Delta \dot{\theta}_i(t) \underline{\mathbf{a}}_i \underline{\mathbf{x}}_i^{i-1}. \quad (4.13)$$

Also, from (2.89) the twist's derivative frame transformation is

$$Ad(\underline{\mathbf{x}}_i^{i-1}) \underline{\dot{\omega}}_{i-1} = \underline{\mathbf{x}}_i^{i-1} \underline{\dot{\omega}}_{i-1} \underline{\mathbf{x}}_i^i. \quad (4.14)$$

Finally may combine (4.11), (4.13) and (4.14) with (4.10),

$$\begin{aligned} \underline{\dot{\omega}}_i &= \underline{\mathbf{a}}_i \Delta \ddot{\theta}_i(t) + Ad(\underline{\mathbf{x}}_i^{i-1}) \underline{\dot{\omega}}_{i-1} + \left(-\frac{1}{2} \Delta \dot{\theta}_i(t) \underline{\mathbf{a}}_i \underline{\mathbf{x}}_i^{i-1} \right) \underline{\omega}_{i-1} \underline{\mathbf{x}}_i^{i-1} + \underline{\mathbf{x}}_i^{i-1} \underline{\omega}_{i-1} \left(\frac{1}{2} \underline{\mathbf{x}}_{i-1}^i \underline{\mathbf{a}}_i \Delta \dot{\theta}_i(t) \right) \\ &= \underline{\mathbf{a}}_i \Delta \ddot{\theta}_i(t) + Ad(\underline{\mathbf{x}}_i^{i-1}) \underline{\dot{\omega}}_{i-1} + \left(\underline{\mathbf{x}}_i^{i-1} \underline{\omega}_{i-1} \underline{\mathbf{x}}_i^{i-1} \underline{\mathbf{a}}_i - \underline{\mathbf{a}}_i \underline{\mathbf{x}}_i^{i-1} \underline{\omega}_{i-1} \underline{\mathbf{x}}_i^{i-1} \right) \frac{\Delta \dot{\theta}_i(t)}{2}. \end{aligned} \quad (4.15)$$

By further analyzing the last term in (4.15) we notice that it fits Definition (2.11) for the dual quaternion cross product, that is

$$\left(\underline{\mathbf{x}}_i^{i-1} \underline{\omega}_{i-1} \underline{\mathbf{x}}_i^{i-1} \underline{\mathbf{a}}_i - \underline{\mathbf{a}}_i \underline{\mathbf{x}}_i^{i-1} \underline{\omega}_{i-1} \underline{\mathbf{x}}_i^{i-1} \right) = 2 \left(\underline{\mathbf{x}}_i^{i-1} \underline{\omega}_{i-1} \underline{\mathbf{x}}_i^{i-1} \right) \times \underline{\mathbf{a}}_i. \quad (4.16)$$

Now, combining 4.15 and (4.16), we have

$$\underline{\dot{\omega}}_i = \underline{\mathbf{a}}_i \Delta \ddot{\theta}_i(t) + Ad(\underline{\mathbf{x}}_i^{i-1}) \underline{\dot{\omega}}_{i-1} + \left(\underline{\mathbf{x}}_i^{i-1} \underline{\omega}_{i-1} \underline{\mathbf{x}}_i^{i-1} \right) \times \underline{\mathbf{a}}_i \Delta \dot{\theta}_i(t). \quad (4.17)$$

And finally, from (4.7), we obtain a final expression for the recursive forward model for the accelerations derived solely from screw theory based on dual quaternion algebra,

$$\underline{\dot{\omega}}_i = \underline{\mathbf{a}}_i \Delta \ddot{\theta}_i(t) + Ad(\underline{\mathbf{x}}_i^{i-1}) \underline{\dot{\omega}}_{i-1} + (\underline{\omega}_i \times \underline{\mathbf{a}}_i) \Delta \dot{\theta}_i(t). \quad (4.18)$$

4.2.2 Backward Recursion

In the backwards recursion of the dqRNEA, given the dual quaternion wrenches acting at the end-effector— $\underline{\mathbf{f}}_{\text{EF}}$, which is initialized in the forward iteration's first step— and the resulting velocities and accelerations of the link's center of mass from the forward recursion, we may compute the required torques to be applied at the joints to obtain the prescribed motion. In this case, we start our analysis at the end-effector and go back down to the robot's base (see Figure 4.1).

To calculate the resulting rigid body dynamics for each individual body, stemming from each link's velocities and accelerations, we may recall Newton's and Euler's laws discussed in Section 3.2, as well as the

resulting dual quaternion rigid body dynamical equation in (3.54), which is inspired by [31] work. Thus, for each individual link i we have that the resulting wrench is

$$\underline{\mathbf{f}}_{R,i} = \underline{\boldsymbol{\omega}}_i \times (G_i(\underline{\boldsymbol{\omega}}_i)) + G_i(\underline{\dot{\boldsymbol{\omega}}}_i), \quad (4.19)$$

in which $G_i(\bullet)$ is the Dual Quaternion Inertia Transformation, defined in (3.44).

Due to the balance of forces in the body, the resulting forces of a rigid body are equal to the sum of the forces being applied in that body (in the case of a serial manipulator we will assume the the only forces being applied on the link stems from attached joints). Thus, the resulting wrench $\underline{\mathbf{f}}_R$ must also be given by the addition of the wrenches at the two adjacent joints (all expressed in the current link's frame), that is,

$$\underline{\mathbf{f}}_{R,i} = \underline{\mathbf{f}}_i - Ad(\underline{\mathbf{x}}_i^{i+1}) \underline{\mathbf{f}}_{i+1}. \quad (4.20)$$

Here we highlight that in the first iteration of the backward recursion, and considering an n joint manipulator, we will have $\underline{\mathbf{f}}_{R,n} = \underline{\mathbf{f}}_n + Ad(\underline{\mathbf{x}}_n^{n+1}) \underline{\mathbf{f}}_{n+1}$. The term $\underline{\mathbf{f}}_{n+1}$ is the force at the end effector: $\underline{\mathbf{f}}_{n+1} = \underline{\mathbf{f}}_{\text{EF}}$.

In the next step we combine (4.19) and (4.20) in order to obtain the full dynamic equation for the required wrench at link i . That is,

$$\underline{\mathbf{f}}_i = \underline{\boldsymbol{\omega}}_i \times (G_i(\underline{\boldsymbol{\omega}}_i)) + G_i(\underline{\dot{\boldsymbol{\omega}}}_i) + Ad(\underline{\mathbf{x}}_i^{i+1}) \underline{\mathbf{f}}_{i+1}. \quad (4.21)$$

Finally, stemming from (4.21) we may take advantage of the double geodesic product in (2.42) in order to calculate the torques on each of the joints,

$$\tau_i = \underline{\mathbf{f}}_i \odot [\mathcal{D}(\underline{\mathbf{a}}_i) + \varepsilon \mathcal{P}(\underline{\mathbf{a}}_i)]. \quad (4.22)$$

We remark here that the double geodesic product is used rather than the usual product because we are dealing with a robot with only revolute joints, and we have simplified $\Delta\theta = \theta \in \mathbb{R}$, therefore, it does not make sense for the torque to be a dual number, such as would be the case if the inner product was used. Moreover, we also highlight here the need to swap the primary and secondary parts of the $\underline{\mathbf{a}}_i$, that is due to the fact that $\underline{\mathbf{f}}_i = \mathbf{f}_i + \varepsilon \mathbf{m}_i$ and $\underline{\mathbf{a}}_i$ is related to the screw $\underline{\mathbf{s}}_i = \boldsymbol{\omega}_i + \varepsilon \mathbf{v}_i$. Thus, by making this change we will be pairing up the linear and angular terms.

4.2.3 Algorithm Summary

Here we present a summary of the dual quaternion based recursive Newton Euler algorithm for an n -DoF manipulator in Algorithm 4.1.

Algorithm 4.1 Dual-quaternion based recursive Newton-Euler Inverse Dynamics for n DoF Manipulator - Basic Formulation

Initialization:

At the home config., set the frames \mathcal{F}_0 to the base, \mathcal{F}_1 to \mathcal{F}_n to the n -links' center of mass, and \mathcal{F}_{n+1} to the end-effector.

Set $\underline{\delta}_{i-1}^i$, \underline{a}_i as the pose of \mathcal{F}_{i-1} and the screw axis of joint i expressed in \mathcal{F}_i , with null vel. $\underline{\omega}_0=0$ and gravity accel. $\underline{\dot{\omega}}_0 = -\varepsilon \mathbf{g}$ at base and end-pose wrench $\underline{f}_{EF} = \underline{f}_{EF} + \varepsilon \mathbf{m}_{EF}$.

Forward iteration

$$\begin{aligned}\underline{x}_i^{i-1}(\theta_i(t)) &= \underline{\delta}_{i-1}^i \exp\left(\underline{a}_i \frac{\Delta\theta_i}{2}\right) \\ \underline{\omega}_i &= \underline{a}_i \dot{\theta}_i(t) + Ad(\underline{x}_i^{i-1})\underline{\omega}_{i-1} \\ \underline{\dot{\omega}}_i &= \underline{a}_i \ddot{\theta}_i(t) + Ad(\underline{x}_i^{i-1})\underline{\dot{\omega}}_{i-1} + (\underline{\omega}_i \times \underline{a}_i) \dot{\theta}_i(t)\end{aligned}$$

Backward iteration

$$\begin{aligned}\underline{f}_{R,i} &= \underline{\omega}_i \times (G_i(\underline{\omega}_i)) + G_i(\underline{\dot{\omega}}_i) \\ \underline{f}_i &= \underline{f}_{R,i} + Ad(\underline{x}_i^{i+1})\underline{f}_{i+1} \\ \tau_i &= \underline{f}_i \odot \underline{a}_i\end{aligned}$$

4.3 DQRNEA IN CLOSED FORM

When dealing some applications it becomes interesting to have the recursive Newton-Euler algorithm rewritten in a closed form, without the many recursive steps being taken. We therefore will present in Subsection 4.3.1 a dual quaternionic-matrix formulation for the equations in dqRNEA. This formulation, which was inspired from [49], works by joining dual quaternions together in vectorial form. Subsequently, in Subsection 4.3.2 we use the quaternionic-matrix equations in order to obtain an expression similar to (4.3), which is the actual closed formulation for the algorithm.

4.3.1 Dual Quaternionic-Matrix Formulation

In order to describe in Algorithm ?? in closed form we must first define a vectorial formulation for some of the variables we will use to construct our equation, for such we will be using the notation and operations presented in Subsection 2.1.5. Thus, we define the joint vector, torque vector, resulting force vector, force vector and twist vector, respectively defined as

$$\underline{\theta} = \begin{bmatrix} \theta_1(t) & \theta_2(t) & \dots & \theta_n(t) \end{bmatrix}^T \in \mathbb{R}^n, \quad (4.23)$$

$$\underline{\tau} = \begin{bmatrix} \tau_1 & \tau_2 & \dots & \tau_n \end{bmatrix}^T \in \mathbb{R}^n, \quad (4.24)$$

$$\underline{F}_R = \begin{bmatrix} \underline{f}_{R,1} & \underline{f}_{R,2} & \dots & \underline{f}_{R,n} \end{bmatrix}^T \in \mathbb{H}_0^n, \quad (4.25)$$

$$\underline{\mathbf{F}} = \begin{bmatrix} \underline{\mathbf{f}}_1 & \underline{\mathbf{f}}_2 & \dots & \underline{\mathbf{f}}_n \end{bmatrix}^T \in \mathbb{H}_0^n, \quad (4.26)$$

$$\underline{\mathbf{W}} = \begin{bmatrix} \underline{\boldsymbol{\omega}}_1 & \underline{\boldsymbol{\omega}}_2 & \dots & \underline{\boldsymbol{\omega}}_n \end{bmatrix}^T \in \mathbb{H}_0^n \quad (4.27)$$

and (4.23) and (4.27) derivatives are

$$\dot{\boldsymbol{\theta}} = \begin{bmatrix} \dot{\theta}_1(t) & \dot{\theta}_2(t) & \dots & \dot{\theta}_n(t) \end{bmatrix}^T \in \mathbb{R}^n, \quad (4.28)$$

$$\ddot{\boldsymbol{\theta}} = \begin{bmatrix} \ddot{\theta}_1(t) & \ddot{\theta}_2(t) & \dots & \ddot{\theta}_n(t) \end{bmatrix}^T \in \mathbb{R}^n, \quad (4.29)$$

$$\dot{\underline{\mathbf{W}}} = \begin{bmatrix} \dot{\underline{\boldsymbol{\omega}}}_1 & \dot{\underline{\boldsymbol{\omega}}}_2 & \dots & \dot{\underline{\boldsymbol{\omega}}}_n \end{bmatrix}^T \in \mathbb{H}_0^n. \quad (4.30)$$

We shall then look back at equation (4.7), and try to construct $\underline{\mathbf{W}}$ line by line. That is, let us look at the individual entries in $\underline{\mathbf{W}}$

$$\begin{aligned} \underline{\boldsymbol{\omega}}_1 &= \underline{\mathbf{a}}_1 \dot{\theta}_1(t) + Ad(\underline{\mathbf{x}}_1^0) \underline{\boldsymbol{\omega}}_0, \\ \underline{\boldsymbol{\omega}}_2 &= \underline{\mathbf{a}}_2 \dot{\theta}_2(t) + Ad(\underline{\mathbf{x}}_2^1) \underline{\boldsymbol{\omega}}_1, \\ \underline{\boldsymbol{\omega}}_3 &= \underline{\mathbf{a}}_3 \dot{\theta}_3(t) + Ad(\underline{\mathbf{x}}_3^2) \underline{\boldsymbol{\omega}}_2, \\ &\vdots \\ \underline{\boldsymbol{\omega}}_n &= \underline{\mathbf{a}}_n \dot{\theta}_n(t) + Ad(\underline{\mathbf{x}}_n^{n-1}) \underline{\boldsymbol{\omega}}_{n-1}. \end{aligned} \quad (4.31)$$

We then rewrite (4.31) in its vectorial form,

$$\begin{aligned} \underline{\mathbf{W}} = \begin{bmatrix} \underline{\boldsymbol{\omega}}_1 \\ \underline{\boldsymbol{\omega}}_2 \\ \underline{\boldsymbol{\omega}}_3 \\ \vdots \\ \underline{\boldsymbol{\omega}}_n \end{bmatrix} &= \begin{bmatrix} \underline{\mathbf{a}}_1 \dot{\theta}_1(t) + Ad(\underline{\mathbf{x}}_1^0) \underline{\boldsymbol{\omega}}_0 \\ \underline{\mathbf{a}}_2 \dot{\theta}_2(t) + Ad(\underline{\mathbf{x}}_2^1) \underline{\boldsymbol{\omega}}_1 \\ \underline{\mathbf{a}}_3 \dot{\theta}_3(t) + Ad(\underline{\mathbf{x}}_3^2) \underline{\boldsymbol{\omega}}_2 \\ \vdots \\ \underline{\mathbf{a}}_n \dot{\theta}_n(t) + Ad(\underline{\mathbf{x}}_n^{n-1}) \underline{\boldsymbol{\omega}}_{n-1} \end{bmatrix} \\ &= \begin{bmatrix} \underline{\mathbf{a}}_1 \dot{\theta}_1(t) \\ \underline{\mathbf{a}}_2 \dot{\theta}_2(t) \\ \underline{\mathbf{a}}_3 \dot{\theta}_3(t) \\ \vdots \\ \underline{\mathbf{a}}_n \dot{\theta}_n(t) \end{bmatrix} + \begin{bmatrix} Ad(\underline{\mathbf{x}}_1^0) \underline{\boldsymbol{\omega}}_0 \\ Ad(\underline{\mathbf{x}}_2^1) \underline{\boldsymbol{\omega}}_1 \\ Ad(\underline{\mathbf{x}}_3^2) \underline{\boldsymbol{\omega}}_2 \\ \vdots \\ Ad(\underline{\mathbf{x}}_n^{n-1}) \underline{\boldsymbol{\omega}}_{n-1} \end{bmatrix}. \end{aligned} \quad (4.32)$$

And as we have already defined $\dot{\boldsymbol{\theta}}$ in (4.28) and $\underline{\mathbf{W}}$ itself, we may rewrite (4.32) in terms of $\dot{\boldsymbol{\theta}}, \underline{\mathbf{W}}$. However, we must first address the issue of the adjoint map, defined in (2.89) as $Ad(\underline{\mathbf{x}}) \underline{\mathbf{p}} = \underline{\mathbf{x}} \underline{\mathbf{p}} \underline{\mathbf{x}}^*$. In order to isolate $\underline{\mathbf{p}}$ we must commute the terms $\underline{\mathbf{p}}$ and $\underline{\mathbf{x}}^*$, which can be done through the operation defined in (2.58), yielding the alternative version of (2.89)

$$A_H(\underline{\mathbf{x}}) \underline{\mathbf{p}} = \overset{+}{\underline{\mathbf{h}}}(\underline{\mathbf{x}}) \bar{\underline{\mathbf{h}}}(\underline{\mathbf{x}}^*) \underline{\mathbf{p}}. \quad (4.33)$$

in which $A_H(\underline{\mathbf{x}}) = \overset{+}{\underline{\mathbf{h}}}(\underline{\mathbf{x}}) \bar{\underline{\mathbf{h}}}(\underline{\mathbf{x}}^*)$.

With (4.33) we can then rewrite (4.32) as

$$\underline{W} = \begin{bmatrix} \underline{\omega}_1 \\ \underline{\omega}_2 \\ \underline{\omega}_3 \\ \vdots \\ \underline{\omega}_n \end{bmatrix} = \begin{bmatrix} \underline{a}_1 & 0 & 0 & \dots & 0 \\ 0 & \underline{a}_2 & 0 & \dots & 0 \\ 0 & 0 & \underline{a}_3 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \underline{a}_n \end{bmatrix} \underline{\theta} + \begin{bmatrix} 0 & 0 & 0 & \dots & 0 \\ A_H(\underline{x}_2^1) & 0 & 0 & \dots & 0 \\ 0 & A_H(\underline{x}_3^2) & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & A_H(\underline{x}_n^{n-1}) & 0 \end{bmatrix} \underline{W} + \begin{bmatrix} Ad(\underline{x}_1^0)\underline{\omega}_0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}. \quad (4.34)$$

Furthermore, let us define the matrices,

$$\underline{A} = \begin{bmatrix} \underline{a}_1 & 0 & 0 & \dots & 0 \\ 0 & \underline{a}_2 & 0 & \dots & 0 \\ 0 & 0 & \underline{a}_3 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \underline{a}_n \end{bmatrix}, \quad (4.35)$$

$$\underline{Q} = \begin{bmatrix} 0 & 0 & 0 & \dots & 0 \\ A_H(\underline{x}_2^1) & 0 & 0 & \dots & 0 \\ & A_H(\underline{x}_3^2) & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & A_H(\underline{x}_n^{n-1}) & 0 \end{bmatrix}, \quad (4.36)$$

and the base vector, containing the initialization values,

$$\underline{W}_{\text{BASE}} = \begin{bmatrix} Ad(\underline{x}_1^0)\underline{\omega}_0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad (4.37)$$

such that we may rewrite (4.34) as

$$\underline{W} = \underline{A}\underline{\theta} + \underline{Q}\underline{W} + \underline{W}_{\text{BASE}}. \quad (4.38)$$

Moreover, from (4.36) we can define

$$\underline{L}_{\text{tmp}} = \underline{I}_n - \underline{Q}, \quad (4.39)$$

in which \underline{I}_n is the identity $n \times n$ dual quaternion matrix. We also notice that

$$\underline{L}_{\text{tmp}} = \begin{bmatrix} \underline{1} & 0 & 0 & \dots & 0 \\ 0 & \underline{1} & 0 & 0 & 0 \\ 0 & 0 & \underline{1} & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \underline{1} \end{bmatrix} - \begin{bmatrix} 0 & 0 & 0 & \dots & 0 & 0 \\ A_H(\underline{x}_2^1) & 0 & 0 & \dots & 0 & 0 \\ 0 & A_H(\underline{x}_3^2) & 0 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & A_H(\underline{x}_n^{n-1}) & 0 \end{bmatrix}, \quad (4.40)$$

$$= \begin{bmatrix} \underline{1} & 0 & 0 & \dots & 0 & 0 \\ -A_H(\underline{x}_2^1) & \underline{1} & 0 & \dots & 0 & 0 \\ 0 & -A_H(\underline{x}_3^2) & \underline{1} & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & -A_H(\underline{x}_n^{n-1}) & \underline{1} \end{bmatrix} \quad (4.41)$$

and considering that there exists an inverse such that $\underline{\mathbf{L}}_{\text{tmp}}\underline{\mathbf{L}}_{\text{tmp}}^{-1} = \underline{\mathbf{I}}_n$, such that

$$\underline{\mathbf{L}}_{\text{tmp}}^{-1} = \begin{bmatrix} \underline{\mathbf{1}} & 0 & 0 & \dots & 0 & 0 \\ A_{\text{H}}(\underline{\mathbf{x}}_2^1) & \underline{\mathbf{1}} & 0 & \dots & 0 & 0 \\ A_{\text{H}}(\underline{\mathbf{x}}_3^1) & A_{\text{H}}(\underline{\mathbf{x}}_3^2) & \underline{\mathbf{1}} & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ A_{\text{H}}(\underline{\mathbf{x}}_n^1) & A_{\text{H}}(\underline{\mathbf{x}}_n^2) & A_{\text{H}}(\underline{\mathbf{x}}_n^3) & \dots & A_{\text{H}}(\underline{\mathbf{x}}_n^{n-1}) & \underline{\mathbf{1}} \end{bmatrix}. \quad (4.42)$$

Therefore, letting $\underline{\mathbf{L}}_{\text{tmp}} = \underline{\mathbf{L}}^{-1}$ we may rearrange (4.38) such that

$$\begin{aligned} \underline{\mathbf{W}} &= \underline{\mathbf{A}}\dot{\underline{\boldsymbol{\theta}}} + \underline{\mathbf{Q}}\underline{\mathbf{W}} + \underline{\mathbf{W}}_{\text{BASE}} \\ \underline{\mathbf{W}} - \underline{\mathbf{Q}}\underline{\mathbf{W}} &= \underline{\mathbf{A}}\dot{\underline{\boldsymbol{\theta}}} + \underline{\mathbf{W}}_{\text{BASE}} \\ \underline{\mathbf{W}}(\underline{\mathbf{I}}_n - \underline{\mathbf{Q}}) &= \underline{\mathbf{A}}\dot{\underline{\boldsymbol{\theta}}} + \underline{\mathbf{W}}_{\text{BASE}} \\ \underline{\mathbf{W}}\underline{\mathbf{L}}^{-1} &= \underline{\mathbf{A}}\dot{\underline{\boldsymbol{\theta}}} + \underline{\mathbf{W}}_{\text{BASE}} \\ \underline{\mathbf{W}} &= \underline{\mathbf{L}}\left(\underline{\mathbf{A}}\dot{\underline{\boldsymbol{\theta}}} + \underline{\mathbf{W}}_{\text{BASE}}\right). \end{aligned} \quad (4.43)$$

We may now employ the same procedure in order to obtain $\underline{\dot{\mathbf{W}}}$. Thus, starting from (4.18), we have

$$\begin{aligned} \underline{\dot{\boldsymbol{\omega}}}_1 &= \underline{\mathbf{a}}_1\ddot{\theta}_1(t) + Ad(\underline{\mathbf{x}}_1^0)\underline{\dot{\boldsymbol{\omega}}}_0 + (\underline{\boldsymbol{\omega}}_1 \times \underline{\mathbf{a}}_1)\dot{\theta}_1(t) \\ \underline{\dot{\boldsymbol{\omega}}}_2 &= \underline{\mathbf{a}}_2\ddot{\theta}_2(t) + Ad(\underline{\mathbf{x}}_2^1)\underline{\dot{\boldsymbol{\omega}}}_1 + (\underline{\boldsymbol{\omega}}_2 \times \underline{\mathbf{a}}_2)\dot{\theta}_2(t) \\ \underline{\dot{\boldsymbol{\omega}}}_3 &= \underline{\mathbf{a}}_3\ddot{\theta}_3(t) + Ad(\underline{\mathbf{x}}_3^2)\underline{\dot{\boldsymbol{\omega}}}_2 + (\underline{\boldsymbol{\omega}}_3 \times \underline{\mathbf{a}}_3)\dot{\theta}_3(t) \\ &\vdots \\ \underline{\dot{\boldsymbol{\omega}}}_n &= \underline{\mathbf{a}}_n\ddot{\theta}_n(t) + Ad(\underline{\mathbf{x}}_n^{n-1})\underline{\dot{\boldsymbol{\omega}}}_{n-1} + (\underline{\boldsymbol{\omega}}_n \times \underline{\mathbf{a}}_n)\dot{\theta}_n(t) \end{aligned} \quad (4.44)$$

which in vectorial form becomes

$$\begin{aligned} \underline{\dot{\mathbf{W}}} &= \begin{bmatrix} \underline{\dot{\boldsymbol{\omega}}}_1 \\ \underline{\dot{\boldsymbol{\omega}}}_2 \\ \underline{\dot{\boldsymbol{\omega}}}_3 \\ \vdots \\ \underline{\dot{\boldsymbol{\omega}}}_n \end{bmatrix} = \begin{bmatrix} \underline{\mathbf{a}}_1\ddot{\theta}_1(t) + Ad(\underline{\mathbf{x}}_1^0)\underline{\dot{\boldsymbol{\omega}}}_0 + (\underline{\boldsymbol{\omega}}_1 \times \underline{\mathbf{a}}_1)\dot{\theta}_1(t) \\ \underline{\mathbf{a}}_2\ddot{\theta}_2(t) + Ad(\underline{\mathbf{x}}_2^1)\underline{\dot{\boldsymbol{\omega}}}_1 + (\underline{\boldsymbol{\omega}}_2 \times \underline{\mathbf{a}}_2)\dot{\theta}_2(t) \\ \underline{\mathbf{a}}_3\ddot{\theta}_3(t) + Ad(\underline{\mathbf{x}}_3^2)\underline{\dot{\boldsymbol{\omega}}}_2 + (\underline{\boldsymbol{\omega}}_3 \times \underline{\mathbf{a}}_3)\dot{\theta}_3(t) \\ \vdots \\ \underline{\mathbf{a}}_n\ddot{\theta}_n(t) + Ad(\underline{\mathbf{x}}_n^{n-1})\underline{\dot{\boldsymbol{\omega}}}_{n-1} + (\underline{\boldsymbol{\omega}}_n \times \underline{\mathbf{a}}_n)\dot{\theta}_n(t) \end{bmatrix} \\ &= \begin{bmatrix} \underline{\mathbf{a}}_1\ddot{\theta}_1(t) \\ \underline{\mathbf{a}}_2\ddot{\theta}_2(t) \\ \underline{\mathbf{a}}_3\ddot{\theta}_3(t) \\ \vdots \\ \underline{\mathbf{a}}_n\ddot{\theta}_n(t) \end{bmatrix} + \begin{bmatrix} Ad(\underline{\mathbf{x}}_1^0)\underline{\dot{\boldsymbol{\omega}}}_0 \\ Ad(\underline{\mathbf{x}}_2^1)\underline{\dot{\boldsymbol{\omega}}}_1 \\ Ad(\underline{\mathbf{x}}_3^2)\underline{\dot{\boldsymbol{\omega}}}_2 \\ \vdots \\ Ad(\underline{\mathbf{x}}_n^{n-1})\underline{\dot{\boldsymbol{\omega}}}_{n-1} \end{bmatrix} + \begin{bmatrix} (\underline{\boldsymbol{\omega}}_1 \times \underline{\mathbf{a}}_1)\dot{\theta}_1(t) \\ (\underline{\boldsymbol{\omega}}_2 \times \underline{\mathbf{a}}_2)\dot{\theta}_2(t) \\ (\underline{\boldsymbol{\omega}}_3 \times \underline{\mathbf{a}}_3)\dot{\theta}_3(t) \\ \vdots \\ (\underline{\boldsymbol{\omega}}_n \times \underline{\mathbf{a}}_n)\dot{\theta}_n(t) \end{bmatrix}. \end{aligned} \quad (4.45)$$

By taking advantage of the already defined matrices in (4.35) and (4.36), as well as $\dot{\underline{\boldsymbol{\theta}}}$, $\ddot{\underline{\boldsymbol{\theta}}}$ and $\underline{\dot{\mathbf{W}}}$ itself, defined respectively in (4.28), (4.29) and (4.30), we can rewrite (4.45) as

$$\underline{\dot{\mathbf{W}}} = \underline{\mathbf{A}}\ddot{\underline{\boldsymbol{\theta}}} + \underline{\mathbf{Q}}\underline{\dot{\mathbf{W}}} + \begin{bmatrix} (\underline{\boldsymbol{\omega}}_1 \times \underline{\mathbf{a}}_1) & 0 & 0 & \dots & 0 \\ 0 & (\underline{\boldsymbol{\omega}}_2 \times \underline{\mathbf{a}}_2) & 0 & 0 & 0 \\ 0 & 0 & (\underline{\boldsymbol{\omega}}_3 \times \underline{\mathbf{a}}_3) & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & (\underline{\boldsymbol{\omega}}_n \times \underline{\mathbf{a}}_n) \end{bmatrix} \dot{\underline{\boldsymbol{\theta}}} + \begin{bmatrix} Ad(\underline{\mathbf{x}}_1^0)\underline{\dot{\boldsymbol{\omega}}}_0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}. \quad (4.46)$$

Now let

$$\underline{\mathbf{C}} = \begin{bmatrix} (\underline{\omega}_1 \times \underline{\mathbf{a}}_1) & 0 & 0 & \dots & 0 \\ 0 & (\underline{\omega}_2 \times \underline{\mathbf{a}}_2) & 0 & 0 & 0 \\ 0 & 0 & (\underline{\omega}_3 \times \underline{\mathbf{a}}_3) & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & (\underline{\omega}_n \times \underline{\mathbf{a}}_n) \end{bmatrix}, \quad (4.47)$$

be the matrix for the cross product terms, and

$$\underline{\dot{\mathbf{W}}}_{\text{BASE}} = \begin{bmatrix} Ad(\underline{\mathbf{x}}_1^0) \underline{\dot{\omega}}_0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad (4.48)$$

be the vector with initial accelerations. Then, (4.46) becomes

$$\underline{\dot{\mathbf{W}}} = \underline{\mathbf{A}} \underline{\ddot{\theta}} + \underline{\mathbf{Q}} \underline{\dot{\mathbf{W}}} + \underline{\mathbf{C}} \underline{\dot{\theta}} + \underline{\dot{\mathbf{W}}}_{\text{BASE}}. \quad (4.49)$$

Moreover, if we take a similar approach to what was done in (4.43), we have

$$\begin{aligned} \underline{\dot{\mathbf{W}}} &= \underline{\mathbf{A}} \underline{\ddot{\theta}} + \underline{\mathbf{Q}} \underline{\dot{\mathbf{W}}} + \underline{\mathbf{C}} \underline{\dot{\theta}} + \underline{\dot{\mathbf{W}}}_{\text{BASE}} \\ \underline{\dot{\mathbf{W}}} - \underline{\mathbf{Q}} \underline{\dot{\mathbf{W}}} &= \underline{\mathbf{A}} \underline{\ddot{\theta}} + \underline{\mathbf{C}} \underline{\dot{\theta}} + \underline{\dot{\mathbf{W}}}_{\text{BASE}} \\ \underline{\dot{\mathbf{W}}} (\underline{\mathbf{I}}_n - \underline{\mathbf{Q}}) &= \underline{\mathbf{A}} \underline{\ddot{\theta}} + \underline{\mathbf{C}} \underline{\dot{\theta}} + \underline{\dot{\mathbf{W}}}_{\text{BASE}} \\ \underline{\dot{\mathbf{W}}} \underline{\mathbf{L}}^{-1} &= \underline{\mathbf{A}} \underline{\ddot{\theta}} + \underline{\mathbf{C}} \underline{\dot{\theta}} + \underline{\dot{\mathbf{W}}}_{\text{BASE}} \\ \underline{\dot{\mathbf{W}}} &= \underline{\mathbf{L}} \left(\underline{\mathbf{A}} \underline{\ddot{\theta}} + \underline{\mathbf{C}} \underline{\dot{\theta}} + \underline{\dot{\mathbf{W}}}_{\text{BASE}} \right) \end{aligned} \quad (4.50)$$

Before proceeding to find a similar formulation for the equations in the backwards iteration let us define one new operations for the $\underline{\mathbb{H}}_0^n$: the element-wise dual quaternion inertia transformation.

Definition 4.1. (DQ-Vec dual quaternion inertia transformation) The dual quaternion inertia operation is defined, for dual quaternion vectors, as the operator $\underline{\mathbf{G}} : \underline{\mathbb{H}}_0^n \rightarrow \underline{\mathbb{H}}_0^n$ such that, given $\underline{\mathbf{P}} = \begin{bmatrix} \underline{\mathbf{p}}_1 & \underline{\mathbf{p}}_2 & \dots & \underline{\mathbf{p}}_n \end{bmatrix}^T \in \underline{\mathbb{H}}_0^n$,

$$\underline{\mathbf{G}}(\underline{\mathbf{P}}) = \begin{bmatrix} \underline{\mathbf{G}}(\underline{\mathbf{p}}_1) & \underline{\mathbf{G}}(\underline{\mathbf{p}}_2) & \dots & \underline{\mathbf{G}}(\underline{\mathbf{p}}_n) \end{bmatrix}^T. \quad (4.51)$$

In which $\underline{\mathbf{G}}(\bullet)$ is the Dual Quaternion Inertia Transformation from Definition 3.8.

Remark 4.1. The dual quaternion inertia transformation can also be defined for matrices in a similar manner to what was done in (4.51). In this case our operator shall be defined as $\underline{\mathbf{G}} : \underline{\mathbb{H}}_0^{n \times m} \rightarrow \underline{\mathbb{H}}_0^{n \times m}$ and we apply $\underline{\mathbf{G}}(\bullet)$ in every element of the matrix.

Remark 4.2. We may also define some proprieties for the operator for the DQ-vec dual quaternion inertia transformation in (4.51). Thus, given the dual quaternion-vectors $\underline{\mathbf{P}}_1 = \begin{bmatrix} \underline{\mathbf{p}}_{1,1} & \underline{\mathbf{p}}_{1,2} & \dots & \underline{\mathbf{p}}_{1,n} \end{bmatrix}^T$ and $\underline{\mathbf{P}}_2 = \begin{bmatrix} \underline{\mathbf{p}}_{2,1} & \underline{\mathbf{p}}_{2,2} & \dots & \underline{\mathbf{p}}_{2,n} \end{bmatrix}^T \in \underline{\mathbb{H}}_0^n$ and the constant $\underline{\mathbf{K}} \in \mathbb{R}^n$ we have

$$1. \underline{\mathbf{G}}(\underline{\mathbf{P}}_1 + \underline{\mathbf{P}}_2) = \underline{\mathbf{G}}(\underline{\mathbf{P}}_1) + \underline{\mathbf{G}}(\underline{\mathbf{P}}_2);$$

$$2. \mathbf{G}(\underline{\mathbf{P}}_1 \mathbf{K}) = \mathbf{G}(\underline{\mathbf{P}}_1) \mathbf{K}.$$

Both proprieties can be proved by inspection.

With Definitions 2.17, 2.19 and 4.1 we may proceed to analyze (4.19) as was done with the other variables in the dqRNEA. In this instance we shall have

$$\begin{aligned} \underline{\mathbf{f}}_{R,1} &= \underline{\boldsymbol{\omega}}_1 \times (G_1(\underline{\boldsymbol{\omega}}_1)) + G_1(\underline{\dot{\boldsymbol{\omega}}}_1) \\ \underline{\mathbf{f}}_{R,2} &= \underline{\boldsymbol{\omega}}_2 \times (G_2(\underline{\boldsymbol{\omega}}_2)) + G_2(\underline{\dot{\boldsymbol{\omega}}}_2) \\ \underline{\mathbf{f}}_{R,3} &= \underline{\boldsymbol{\omega}}_3 \times (G_3(\underline{\boldsymbol{\omega}}_3)) + G_3(\underline{\dot{\boldsymbol{\omega}}}_3) \\ &\vdots \\ \underline{\mathbf{f}}_{R,n} &= \underline{\boldsymbol{\omega}}_n \times (G_n(\underline{\boldsymbol{\omega}}_n)) + G_n(\underline{\dot{\boldsymbol{\omega}}}_n), \end{aligned} \quad (4.52)$$

which, in vectorial form and from (4.25) becomes,

$$\begin{aligned} \underline{\mathbf{F}}_R = \begin{bmatrix} \underline{\mathbf{f}}_{R,1} \\ \underline{\mathbf{f}}_{R,2} \\ \underline{\mathbf{f}}_{R,3} \\ \vdots \\ \underline{\mathbf{f}}_{R,n} \end{bmatrix} &= \begin{bmatrix} \underline{\boldsymbol{\omega}}_1 \times (G_1(\underline{\boldsymbol{\omega}}_1)) + G_1(\underline{\dot{\boldsymbol{\omega}}}_1) \\ \underline{\boldsymbol{\omega}}_2 \times (G_2(\underline{\boldsymbol{\omega}}_2)) + G_2(\underline{\dot{\boldsymbol{\omega}}}_2) \\ \underline{\boldsymbol{\omega}}_3 \times (G_3(\underline{\boldsymbol{\omega}}_3)) + G_3(\underline{\dot{\boldsymbol{\omega}}}_3) \\ \vdots \\ \underline{\boldsymbol{\omega}}_n \times (G_n(\underline{\boldsymbol{\omega}}_n)) + G_n(\underline{\dot{\boldsymbol{\omega}}}_n) \end{bmatrix} \\ &= \begin{bmatrix} \underline{\boldsymbol{\omega}}_1 \times (G_1(\underline{\boldsymbol{\omega}}_1)) \\ \underline{\boldsymbol{\omega}}_2 \times (G_2(\underline{\boldsymbol{\omega}}_2)) \\ \underline{\boldsymbol{\omega}}_3 \times (G_3(\underline{\boldsymbol{\omega}}_3)) \\ \vdots \\ \underline{\boldsymbol{\omega}}_n \times (G_n(\underline{\boldsymbol{\omega}}_n)) \end{bmatrix} + \begin{bmatrix} G_1(\underline{\dot{\boldsymbol{\omega}}}_1) \\ G_2(\underline{\dot{\boldsymbol{\omega}}}_2) \\ G_3(\underline{\dot{\boldsymbol{\omega}}}_3) \\ \vdots \\ G_n(\underline{\dot{\boldsymbol{\omega}}}_n) \end{bmatrix}. \end{aligned} \quad (4.53)$$

From our definition for the DQ-Vec Cross product in (2.71) and DQ-Vec dual quaternion inertia transformation in (4.51), we may rewrite (4.53) as

$$\underline{\mathbf{F}}_R = \underline{\mathbf{W}} \otimes \mathbf{G}(\underline{\mathbf{W}}) + \mathbf{G}(\underline{\dot{\mathbf{W}}}). \quad (4.54)$$

The next equation is (4.20), with individual entries being

$$\begin{aligned} \underline{\mathbf{f}}_1 &= \underline{\mathbf{f}}_{R,1} + Ad(\underline{\mathbf{x}}_1^2) \underline{\mathbf{f}}_2 \\ \underline{\mathbf{f}}_2 &= \underline{\mathbf{f}}_{R,2} + Ad(\underline{\mathbf{x}}_2^3) \underline{\mathbf{f}}_3 \\ \underline{\mathbf{f}}_3 &= \underline{\mathbf{f}}_{R,3} + Ad(\underline{\mathbf{x}}_3^4) \underline{\mathbf{f}}_4 \\ &\vdots \\ \underline{\mathbf{f}}_{n-1} &= \underline{\mathbf{f}}_{R,n-1} + Ad(\underline{\mathbf{x}}_{n-1}^n) \underline{\mathbf{f}}_n \\ \underline{\mathbf{f}}_n &= \underline{\mathbf{f}}_{R,n} + Ad(\underline{\mathbf{x}}_n^{n+1}) \underline{\mathbf{f}}_{n+1}. \end{aligned} \quad (4.55)$$

From (4.55) we can write the vectorial formulation

$$\begin{aligned}
\underline{\mathbf{F}} &= \begin{bmatrix} \underline{\mathbf{f}}_1 \\ \underline{\mathbf{f}}_2 \\ \underline{\mathbf{f}}_3 \\ \vdots \\ \underline{\mathbf{f}}_{n-1} \\ \underline{\mathbf{f}}_n \end{bmatrix} = \begin{bmatrix} \underline{\mathbf{f}}_{R,1} + Ad(\underline{\mathbf{x}}_1^2)\underline{\mathbf{f}}_2 \\ \underline{\mathbf{f}}_{R,2} + Ad(\underline{\mathbf{x}}_2^3)\underline{\mathbf{f}}_3 \\ \underline{\mathbf{f}}_{R,3} + Ad(\underline{\mathbf{x}}_3^4)\underline{\mathbf{f}}_4 \\ \vdots \\ \underline{\mathbf{f}}_{R,n-1} + Ad(\underline{\mathbf{x}}_{n-1}^n)\underline{\mathbf{f}}_n \\ \underline{\mathbf{f}}_{R,n} + Ad(\underline{\mathbf{x}}_n^{n+1})\underline{\mathbf{f}}_{n+1} \end{bmatrix} \\
&= \begin{bmatrix} \underline{\mathbf{f}}_{R,1} \\ \underline{\mathbf{f}}_{R,2} \\ \underline{\mathbf{f}}_{R,3} \\ \vdots \\ \underline{\mathbf{f}}_{R,n-1} \\ \underline{\mathbf{f}}_{R,n} \end{bmatrix} + \begin{bmatrix} Ad(\underline{\mathbf{x}}_1^2)\underline{\mathbf{f}}_2 \\ Ad(\underline{\mathbf{x}}_2^3)\underline{\mathbf{f}}_3 \\ Ad(\underline{\mathbf{x}}_3^4)\underline{\mathbf{f}}_4 \\ \vdots \\ Ad(\underline{\mathbf{x}}_{n-1}^n)\underline{\mathbf{f}}_n \\ Ad(\underline{\mathbf{x}}_n^{n+1})\underline{\mathbf{f}}_{n+1} \end{bmatrix}, \tag{4.56}
\end{aligned}$$

in which we may define force at the end effector as

$$\underline{\mathbf{F}}_{\text{EF}} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ Ad(\underline{\mathbf{x}}_n^{n+1})\underline{\mathbf{f}}_{n+1} \end{bmatrix}. \tag{4.57}$$

With (4.57), the adjoint definition in (4.33) and the vectors $\underline{\mathbf{F}}_R$ and $\underline{\mathbf{F}}$ itself we can rewrite (4.56) as

$$\underline{\mathbf{F}} = \underline{\mathbf{F}}_R + \begin{bmatrix} 0 & A_H(\underline{\mathbf{x}}_1^2) & 0 & 0 & \dots & 0 \\ 0 & 0 & A_H(\underline{\mathbf{x}}_2^3) & 0 & \dots & 0 \\ 0 & 0 & 0 & A_H(\underline{\mathbf{x}}_3^4) & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & 0 \\ 0 & 0 & 0 & 0 & \dots & A_H(\underline{\mathbf{x}}_{n-1}^n) \\ 0 & 0 & 0 & 0 & \dots & 0 \end{bmatrix} \underline{\mathbf{F}} + \underline{\mathbf{F}}_{\text{EF}}. \tag{4.58}$$

More so, from (4.36), we also highlight that

$$\begin{aligned}
(\underline{Q}^T)^* &= \begin{bmatrix} 0 & (A_H(\underline{\mathbf{x}}_2^1))^* & 0 & 0 & \dots & 0 \\ 0 & 0 & (A_H(\underline{\mathbf{x}}_3^2))^* & 0 & \dots & 0 \\ 0 & 0 & 0 & (A_H(\underline{\mathbf{x}}_4^3))^* & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & 0 \\ 0 & 0 & 0 & 0 & \dots & (A_H(\underline{\mathbf{x}}_n^{n-1}))^* \\ 0 & 0 & 0 & 0 & \dots & 0 \end{bmatrix} \\
&= \begin{bmatrix} 0 & A_H(\underline{\mathbf{x}}_1^2) & 0 & 0 & \dots & 0 \\ 0 & 0 & A_H(\underline{\mathbf{x}}_4^3) & 0 & \dots & 0 \\ 0 & 0 & 0 & A_H(\underline{\mathbf{x}}_3^4) & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & 0 \\ 0 & 0 & 0 & 0 & \dots & A_H(\underline{\mathbf{x}}_{n-1}^n) \\ 0 & 0 & 0 & 0 & \dots & 0 \end{bmatrix}, \tag{4.59}
\end{aligned}$$

and thus,

$$\underline{\mathbf{F}} = \underline{\mathbf{F}}_R + (\underline{Q}^T)^* \underline{\mathbf{F}} + \underline{\mathbf{F}}_{\text{EF}}. \tag{4.60}$$

Finally, by taking an approach similar to (4.43) and (4.50), we have

$$\begin{aligned}
\underline{\mathbf{F}} &= \underline{\mathbf{F}}_R + (\underline{Q}^T)^* \underline{\mathbf{F}} + \underline{\mathbf{F}}_{\text{EF}} \\
\underline{\mathbf{F}} - (\underline{Q}^T)^* \underline{\mathbf{F}} &= \underline{\mathbf{F}}_R + \underline{\mathbf{F}}_{\text{EF}} \\
\underline{\mathbf{F}} (\underline{\mathbf{I}}_n - (\underline{Q}^T)^*) &= \underline{\mathbf{F}}_R + \underline{\mathbf{F}}_{\text{EF}} \\
\underline{\mathbf{F}} ((\underline{\mathbf{L}}^{-1})^T)^* &= \underline{\mathbf{F}}_R + \underline{\mathbf{F}}_{\text{EF}} \\
\underline{\mathbf{F}} &= (\underline{\mathbf{L}}^T)^* (\underline{\mathbf{F}}_R + \underline{\mathbf{F}}_{\text{EF}}). \tag{4.61}
\end{aligned}$$

Lastly, we must calculate the torque, as in (4.20), thus checking the individual entries of the matrix

$$\begin{aligned}
\tau_1 &= \underline{\mathbf{f}}_1 \odot [\mathcal{D}(\underline{\mathbf{a}}_1) + \varepsilon \mathcal{P}(\underline{\mathbf{a}}_1)] \\
\tau_2 &= \underline{\mathbf{f}}_2 \odot [\mathcal{D}(\underline{\mathbf{a}}_2) + \varepsilon \mathcal{P}(\underline{\mathbf{a}}_2)] \\
\tau_3 &= \underline{\mathbf{f}}_3 \odot [\mathcal{D}(\underline{\mathbf{a}}_3) + \varepsilon \mathcal{P}(\underline{\mathbf{a}}_3)] \\
&\vdots \\
\tau_n &= \underline{\mathbf{f}}_n \odot [\mathcal{D}(\underline{\mathbf{a}}_n) + \varepsilon \mathcal{P}(\underline{\mathbf{a}}_n)]. \tag{4.62}
\end{aligned}$$

Equation	
1	$\underline{W} = \underline{L} \left(\underline{A}\dot{\theta} + \underline{W}_{\text{BASE}} \right)$
2	$\underline{\dot{W}} = \underline{L} \left(\underline{A}\ddot{\theta} + \underline{C}\dot{\theta} + \underline{\dot{W}}_{\text{BASE}} \right)$
3	$\underline{F}_R = \underline{W} \otimes \underline{G}(\underline{W}) + \underline{G}(\underline{\dot{W}})$
4	$\underline{F} = \underline{L}^* (\underline{F}_R + \underline{F}_{\text{BASE}})$
5	$\underline{\tau} = \underline{F} \otimes \underline{A}_S$

Table 4.1: List of main operations in Dual Quaternionic-Matrix formulation

To write (4.62) in the vectorial form we must take into account the product defined in (2.73) resulting in

$$\begin{aligned}
\underline{\tau} = \begin{bmatrix} \tau_1 \\ \tau_2 \\ \tau_3 \\ \vdots \\ \tau_n \end{bmatrix} &= \begin{bmatrix} \underline{f}_1 \odot [\mathcal{D}(\underline{a}_1) + \varepsilon\mathcal{P}(\underline{a}_1)] \\ \underline{f}_2 \odot [\mathcal{D}(\underline{a}_2) + \varepsilon\mathcal{P}(\underline{a}_2)] \\ \underline{f}_3 \odot [\mathcal{D}(\underline{a}_3) + \varepsilon\mathcal{P}(\underline{a}_3)] \\ \vdots \\ \underline{f}_n \odot [\mathcal{D}(\underline{a}_n) + \varepsilon\mathcal{P}(\underline{a}_n)] \end{bmatrix} \\
&= \begin{bmatrix} \underline{f}_1 \\ \underline{f}_2 \\ \underline{f}_3 \\ \vdots \\ \underline{f}_n \end{bmatrix} \otimes \begin{bmatrix} \mathcal{D}(\underline{a}_1) + \varepsilon\mathcal{P}(\underline{a}_1) \\ \mathcal{D}(\underline{a}_2) + \varepsilon\mathcal{P}(\underline{a}_2) \\ \mathcal{D}(\underline{a}_3) + \varepsilon\mathcal{P}(\underline{a}_3) \\ \vdots \\ \mathcal{D}(\underline{a}_n) + \varepsilon\mathcal{P}(\underline{a}_n) \end{bmatrix}. \tag{4.63}
\end{aligned}$$

Finally we may define

$$\underline{A}_S = \begin{bmatrix} \mathcal{D}(\underline{a}_1) + \varepsilon\mathcal{P}(\underline{a}_1) \\ \mathcal{D}(\underline{a}_2) + \varepsilon\mathcal{P}(\underline{a}_2) \\ \mathcal{D}(\underline{a}_3) + \varepsilon\mathcal{P}(\underline{a}_3) \\ \vdots \\ \mathcal{D}(\underline{a}_n) + \varepsilon\mathcal{P}(\underline{a}_n) \end{bmatrix} \tag{4.64}$$

and together with the vectors $\underline{\tau}$ and \underline{F} we have,

$$\underline{\tau} = \underline{F} \otimes \underline{A}_S, \tag{4.65}$$

thus finishing the set of equations that form the dual quaternionic-matrix formulation.

To close this section, we present, in Table 4.1, a summary of the main equations used in this section to form this non-recursive formulation.

4.3.2 Closed Form Equation

While writing the closed form equation for the dqRNEA our aim is to obtain a result similar to (4.3). Thus, we shall use the expressions in Table 4.1.

Starting with (4.65), we must notice that

$$\underline{\tau} = \underline{F} \otimes \underline{A}_S = \underline{A}_S \otimes \underline{F}. \tag{4.66}$$

Indeed, as the double geodesic product is commutative, and because the operation defined in (2.73) is done element-wise throughout the vectors, then it stands that the vectors may commute as well. Furthermore, we substitute (4.61) in the right-hand side of (4.66), thus

$$\begin{aligned}\tau &= \underline{\mathbf{A}}_S \otimes [\underline{\mathbf{L}}^* (\underline{\mathbf{F}}_R + \underline{\mathbf{F}}_{\text{EF}})] \\ \tau &= \underline{\mathbf{A}}_S \otimes [\underline{\mathbf{L}}^* \underline{\mathbf{F}}_R + \underline{\mathbf{L}}^* \underline{\mathbf{F}}_{\text{EF}}] \\ \tau &= \underline{\mathbf{A}}_S \otimes [\underline{\mathbf{L}}^* \underline{\mathbf{F}}_R] + \underline{\mathbf{A}}_S \otimes [\underline{\mathbf{L}}^* \underline{\mathbf{F}}_{\text{EF}}].\end{aligned}\quad (4.67)$$

Next we combine (4.67) with (4.54) and rearrange the terms

$$\begin{aligned}\tau &= \underline{\mathbf{A}}_S \otimes \left[\underline{\mathbf{L}}^* \left(\underline{\mathbf{W}} \otimes \underline{\mathbf{G}}(\underline{\mathbf{W}}) + \underline{\mathbf{G}}(\underline{\dot{\mathbf{W}}}) \right) \right] + \underline{\mathbf{A}}_S \otimes [\underline{\mathbf{L}}^* \underline{\mathbf{F}}_{\text{EF}}] \\ \tau &= \underline{\mathbf{A}}_S \otimes \left[\underline{\mathbf{L}}^* (\underline{\mathbf{W}} \otimes \underline{\mathbf{G}}(\underline{\mathbf{W}})) + \underline{\mathbf{L}}^* \underline{\mathbf{G}}(\underline{\dot{\mathbf{W}}}) \right] + \underline{\mathbf{A}}_S \otimes [\underline{\mathbf{L}}^* \underline{\mathbf{F}}_{\text{EF}}] \\ \tau &= \underline{\mathbf{A}}_S \otimes [\underline{\mathbf{L}}^* (\underline{\mathbf{W}} \otimes \underline{\mathbf{G}}(\underline{\mathbf{W}}))] + \underline{\mathbf{A}}_S \otimes \left[\underline{\mathbf{L}}^* \underline{\mathbf{G}}(\underline{\dot{\mathbf{W}}}) \right] + \underline{\mathbf{A}}_S \otimes [\underline{\mathbf{L}}^* \underline{\mathbf{F}}_{\text{EF}}].\end{aligned}\quad (4.68)$$

Finally, we substitute (4.43) and (4.50) in (4.68) and, once more, rearrange the terms,

$$\begin{aligned}\tau &= \underline{\mathbf{A}}_S \otimes \left[\underline{\mathbf{L}}^* \left(\underline{\mathbf{W}} \otimes \underline{\mathbf{G}} \left(\underline{\mathbf{L}} \left(\underline{\mathbf{A}}\dot{\boldsymbol{\theta}} + \underline{\mathbf{W}}_{\text{BASE}} \right) \right) \right) \right] + \underline{\mathbf{A}}_S \otimes \left[\underline{\mathbf{L}}^* \underline{\mathbf{G}} \left(\underline{\mathbf{L}} \left(\underline{\mathbf{A}}\ddot{\boldsymbol{\theta}} + \underline{\mathbf{C}}\dot{\boldsymbol{\theta}} + \underline{\dot{\mathbf{W}}}_{\text{BASE}} \right) \right) \right] + \underline{\mathbf{A}}_S \otimes [\underline{\mathbf{L}}^* \underline{\mathbf{F}}_{\text{EF}}] \\ &= \underline{\mathbf{A}}_S \otimes \left[\underline{\mathbf{L}}^* \left(\underline{\mathbf{W}} \otimes \underline{\mathbf{G}} \left(\underline{\mathbf{L}}\underline{\mathbf{A}}\dot{\boldsymbol{\theta}} + \underline{\mathbf{L}}\underline{\mathbf{W}}_{\text{BASE}} \right) \right) \right] + \underline{\mathbf{A}}_S \otimes \left[\underline{\mathbf{L}}^* \underline{\mathbf{G}} \left(\underline{\mathbf{L}}\underline{\mathbf{A}}\ddot{\boldsymbol{\theta}} + \underline{\mathbf{L}}\underline{\mathbf{C}}\dot{\boldsymbol{\theta}} + \underline{\mathbf{L}}\underline{\dot{\mathbf{W}}}_{\text{BASE}} \right) \right] + \underline{\mathbf{A}}_S \otimes [\underline{\mathbf{L}}^* \underline{\mathbf{F}}_{\text{EF}}].\end{aligned}\quad (4.69)$$

Here we use the proprieties highlighted in Remark 4.2 in order to break down the terms in $\underline{\mathbf{G}}(\bullet)$. Thus

$$\begin{aligned}\tau &= \underline{\mathbf{A}}_S \otimes \left[\underline{\mathbf{L}}^* \left(\underline{\mathbf{W}} \otimes \underline{\mathbf{G}} \left(\underline{\mathbf{L}}\underline{\mathbf{A}}\dot{\boldsymbol{\theta}} \right) + \underline{\mathbf{W}} \otimes \underline{\mathbf{G}} \left(\underline{\mathbf{L}}\underline{\mathbf{W}}_{\text{BASE}} \right) \right) \right] \\ &\quad + \underline{\mathbf{A}}_S \otimes \left[\underline{\mathbf{L}}^* \underline{\mathbf{G}} \left(\underline{\mathbf{L}}\underline{\mathbf{A}}\ddot{\boldsymbol{\theta}} \right) + \underline{\mathbf{L}}^* \underline{\mathbf{G}} \left(\underline{\mathbf{L}}\underline{\mathbf{C}}\dot{\boldsymbol{\theta}} \right) + \underline{\mathbf{L}}^* \underline{\mathbf{G}} \left(\underline{\mathbf{L}}\underline{\dot{\mathbf{W}}}_{\text{BASE}} \right) \right] + \underline{\mathbf{A}}_S \otimes [\underline{\mathbf{L}}^* \underline{\mathbf{F}}_{\text{EF}}] \\ &= \underline{\mathbf{A}}_S \otimes \left[\underline{\mathbf{L}}^* \underline{\mathbf{W}} \otimes \left(\underline{\mathbf{G}} \left(\underline{\mathbf{L}}\underline{\mathbf{A}} \right) \dot{\boldsymbol{\theta}} \right) \right] + \underline{\mathbf{A}}_S \otimes [\underline{\mathbf{L}}^* \underline{\mathbf{W}} \otimes \underline{\mathbf{G}} \left(\underline{\mathbf{L}}\underline{\mathbf{W}}_{\text{BASE}} \right)] \\ &\quad + \underline{\mathbf{A}}_S \otimes [\underline{\mathbf{L}}^* \underline{\mathbf{G}} \left(\underline{\mathbf{L}}\underline{\mathbf{A}} \right)] \ddot{\boldsymbol{\theta}} + \underline{\mathbf{A}}_S \otimes [\underline{\mathbf{L}}^* \underline{\mathbf{G}} \left(\underline{\mathbf{L}}\underline{\mathbf{C}} \right)] \dot{\boldsymbol{\theta}} + \underline{\mathbf{A}}_S \otimes \left[\underline{\mathbf{L}}^* \underline{\mathbf{G}} \left(\underline{\mathbf{L}}\underline{\dot{\mathbf{W}}}_{\text{BASE}} \right) \right] + [\underline{\mathbf{A}}_S \otimes \underline{\mathbf{L}}^*] \underline{\mathbf{F}}_{\text{EF}}.\end{aligned}\quad (4.70)$$

The expression obtained in (4.70) is the closed form version of the dqRNEA, however, in order to ensure a more intuitive representation, we may look again at (4.3). Finally, matching the two equations we have

$$\tau = \underline{\mathbf{M}}(\boldsymbol{\theta})\ddot{\boldsymbol{\theta}} + \underline{\mathbf{c}}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}) + \underline{\mathbf{g}}(\boldsymbol{\theta}) + \underline{\mathbf{J}}^T(\boldsymbol{\theta})\underline{\mathbf{F}}_{\text{EF}},\quad (4.71)$$

in which

$$\underline{\mathbf{M}}(\boldsymbol{\theta}) = \underline{\mathbf{A}}_S \otimes [\underline{\mathbf{L}}^* \underline{\mathbf{G}} \left(\underline{\mathbf{L}}\underline{\mathbf{A}} \right)];\quad (4.72)$$

$$\underline{\mathbf{c}}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}) = \underline{\mathbf{A}}_S \otimes \left[\underline{\mathbf{L}}^* \underline{\mathbf{W}} \otimes \left(\underline{\mathbf{G}} \left(\underline{\mathbf{L}}\underline{\mathbf{A}} \right) \dot{\boldsymbol{\theta}} \right) \right] + \underline{\mathbf{A}}_S \otimes [\underline{\mathbf{L}}^* \underline{\mathbf{W}} \otimes \underline{\mathbf{G}} \left(\underline{\mathbf{L}}\underline{\mathbf{W}}_{\text{BASE}} \right)] + \underline{\mathbf{A}}_S \otimes [\underline{\mathbf{L}}^* \underline{\mathbf{G}} \left(\underline{\mathbf{L}}\underline{\mathbf{C}} \right)] \dot{\boldsymbol{\theta}};\quad (4.73)$$

$$\underline{\mathbf{g}}(\boldsymbol{\theta}) = \underline{\mathbf{A}}_S \otimes \left[\underline{\mathbf{L}}^* \underline{\mathbf{G}} \left(\underline{\mathbf{L}}\underline{\dot{\mathbf{W}}}_{\text{BASE}} \right) \right];\quad (4.74)$$

$$\underline{\mathbf{J}}^T(\boldsymbol{\theta}) = \underline{\mathbf{A}}_S \otimes \underline{\mathbf{L}}^*.\quad (4.75)$$

We add here that our formulation for the dqRNEA is one attempt to close a gap in the literature, both with the recursive formulation and the closed one. Indeed, there not many works tackling the use of dual quaternions for describing the RNEA for serial manipulators (or any manipulator for that matter). One exception however are the recent works in [29,30], which describe the quaternion-based inverse dynamics of a spacecraft-mounted

robotic manipulator configured with different joint types. We reiterate here, however, that although some of the equations in [29,30] are similar to our own, their approach is vastly different. While they propose a framework for all types of joint, their approach has many equations presented the Euclidean vector-matrix formulation. This is solution, although valid, is more costly and many times also unintuitive and cumbersome, as there is the need for mapping the dual quaternions to vectors and matrices.

Remark 4.3. The closed formulation presented in (4.71) is exactly the same the general formulation presented in (4.3). We note however, that the individual terms ((4.72) to (4.75)) are not the same as they would have been in other algebras.

4.4 DQ BASED INVERSE DYNAMICS CONTROLLER

One of the many applications for the models derived throughout this chapter is on the control of robotic mechanisms. Indeed, dynamical controllers are very important and can be deployed in a number of applications [102–107] in order to enhance a robot’s abilities to reliably interact with its environment.

In this context, and in order to further validate our models, we propose to close up this chapter with by designing of a computed torque controller based on dual quaternions¹. Although conceptually simple and well known, this MIMO controller can be quite powerful to fully compensate the non-linearities of presented in the dynamics of a robot motion [49,93]. Thus, in Subsection 4.4.1 we present, as a contribution, the formulation for a PD computed torque controller based on the equations of Subsection 4.3.2.

4.4.1 Computed Torque Control

The computed-torque control (CTC) is a well known technique to control the motion of a manipulator while accounting for the robot’s dynamics model [108]. As a special case of a feedback stabilization scheme for non linear systems the CTC produces a very intuitive and easy to understand linear system. Provided that the robot model is well known, we may use this approach to control our manipulator very reliably, however, more complex approaches should be taken once we deal with real world systems in which many parameters are not completely known. Although in this manuscript we are only presenting the basic formulation for the CTC, we highlight there is a great variety of torque control based approaches that can better account for uncertainties in the models being used [5, 49, 93, 108, 109].

To design our controller we take (4.71) as a starting point, however, in order to simplify our problem we consider that no forces are acting on the end-effector—thus $\underline{F}_{\text{BASE}} = 0$, turning our model into

$$\tau = \underline{M}(\theta)\ddot{\theta} + \underline{c}(\theta, \dot{\theta}) + \underline{g}(\theta). \quad (4.76)$$

The intuition for this controller is that we want to find a feedback law such that

$$\tau_{\text{input}} = f(\theta, \dot{\theta}, \ddot{\theta})$$

results in a linear system when substituted in our original system (4.76). Usually, for a multi-variable non-linear system τ_{input} would be incredibly difficult to find, however, for our particular system, in which we consider $\underline{M}(\theta)$, $\underline{c}(\theta, \dot{\theta})$ and $\underline{g}(\theta)$ to be completely known, the problem becomes quite simple [93].

¹Also known as inverse dynamic control

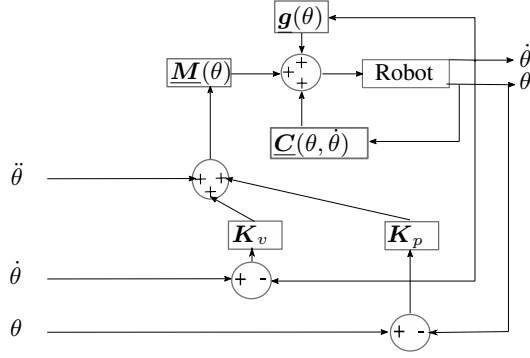


Figure 4.2: Block diagram for computed torque control, following equations (4.77)-(4.82)

Let us choose a control law such

$$\tau_{\text{input}} = \underline{M}(\theta)\mathbf{a}_c + \underline{c}(\theta, \dot{\theta}) + \underline{g}(\theta), \quad (4.77)$$

in which \mathbf{a}_c is an input we are still to choose. Furthermore, since $\underline{M}(\theta)$ is invertible, then we may combine (4.76) and (4.77) to obtain

$$\ddot{\theta} = \mathbf{a}_c. \quad (4.78)$$

The system in (4.78) is known as a double integrator system and it stands out as a linear and decoupled system, enabling us to use each individual element of \mathbf{a}_c to control each output.

We now design the input \mathbf{a}_c with a control law of our choice, in this case we will choose a PD controller scheme to drive the input as a function of the position and velocity, that is,

$$\mathbf{a}_c = -\mathbf{K}_P\dot{\theta} - \mathbf{K}_D\ddot{\theta} + \mathbf{r}(t) \quad (4.79)$$

with \mathbf{K}_P and \mathbf{K}_D being respectively the proportional and derivative gains and $\mathbf{r}(t)$ is the desired feed-forward trajectory, which we may define as

$$\mathbf{r}(t) = \ddot{\theta}_{\text{desired}} + \mathbf{K}_P\dot{\theta}_{\text{desired}} + \mathbf{K}_D\theta_{\text{desired}}. \quad (4.80)$$

Substituting (4.80) in (4.79) we have

$$\mathbf{a}_c = \mathbf{K}_P(\theta_{\text{desired}} - \theta) + \mathbf{K}_D(\dot{\theta}_{\text{desired}} - \dot{\theta}) + \ddot{\theta}_{\text{desired}}, \quad (4.81)$$

in which we may define $(\theta_{\text{desired}} - \theta) = e$ and $(\dot{\theta}_{\text{desired}} - \dot{\theta}) = \dot{e}$ as the error functions, thus creating our control law,

$$\mathbf{a}_c = \mathbf{K}_Pe + \mathbf{K}_D\dot{e} + \ddot{\theta}_{\text{desired}}. \quad (4.82)$$

As stated this controller is quite simple and yet very powerful, making it ideal for validating our models. In Figure 4.2 we show the block diagram for this particular control-scheme.

5

COMPUTATIONAL COST ANALYSIS

One of the main goals of the work being described in the manuscript was to introduce a cost efficient, dual quaternion based inverse dynamics algorithm. Indeed, throughout Chapter 4 we have shown both the recursive and closed versions of the dqRNEA and we have shown how some of the main design choices for this algorithm were taken in order to guarantee better computational cost.

In this chapter, we shall take a better look at the computational costs involved in our algorithm and compare it with some results in the literature. Thus, in Section 5.1 we present the methodology we will be using for calculating the costs throughout this chapter. In Section 5.2 we show the individual costs for each dual quaternion operation, and more so, we show how to optimize the adjoint operation so that it will produce a lower computational cost. In Section 5.3 we show the costs for the dqRNEA algorithm, in Section 5.4 we show the costs for a similar algorithm, but described in HTM and, finally, in Section 5.5 we compare those costs with similar algorithms in the literature.

5.1 METHODOLOGY FOR COMPUTATIONAL COST ANALYSIS

In this section we detail the methodology used for evaluating our algorithm performance and computational costs associated with the rigid body kinematics and dynamics. The methodology we will present is not a novelty and has been done in the works of [37, 39–41, 77], and here, as in our previous work [60], we add as a contribution the analysis of individual costs for operations with dual quaternions, as well as the costs for our dqRNEA algorithm.

The computational cost is, therefore, computed in terms of the function

$$\text{cost}_{\text{Total}}(\text{Operation}) = ([\text{storage}] f, [\text{n}^\circ \text{Mult.}] \times, [\text{n}^\circ \text{Add.}] +). \quad (5.1)$$

In other words, we calculate the costs in terms of the number of elementary operations (multiplications/division and addition/subtraction of floating units), and we also add the storage costs, as fetching an operand from the memory greatly affects the computational time for the algorithm [110]. In order to illustrate how (5.1) can be used, let us look at the quaternion addition in (2.3). In this case we have two quaternions: $\mathbf{q} = \eta + \hat{i}\mu_1 + \hat{j}\mu_2 + \hat{k}\mu_3$, consisting of four real numbers, and $\mathbf{q}' = \eta' + \hat{i}\mu'_1 + \hat{j}\mu'_2 + \hat{k}\mu'_3$, consisting of another four real units, thus making necessary eight floating points of storage. Moreover, the quaternion addition is defined as

$$\mathbf{q} + \mathbf{q}' = (\eta + \eta') + \hat{i}(\mu_1 + \mu'_1) + \hat{j}(\mu_2 + \mu'_2) + \hat{k}(\mu_3 + \mu'_3),$$

thus having four additions and zero multiplications between the real numbers. This analysis results on

$$\text{cost}_{\text{Total}}(\text{Quat. Addition}) = (8f, 0 \times, 4+). \quad (5.2)$$

Another illustrative example would be the sum of n quaternions. In this case each quaternion would have a storage cost of $4f$ and we would have the four additions of (5.2) being repeated $(n - 1)$ times. Thus, the total cost would be

$$\text{cost}_{\text{Total}}\left(\sum_{i=1}^n \mathbf{q}_i\right) = \begin{pmatrix} n \\ 1 \\ n-1 \end{pmatrix} (4f, 0 \times, 4+). \quad (5.3)$$

We highlight that this analysis does not take computational costs into account, as it is dependent on software and hardware specifications.

5.2 COMPUTATIONAL COST OF DQ-BASED OPERATIONS

In this Section we use the methodology presented in Section 5.1 to calculate the computational costs for the main quaternion (Subsection 5.2.1) and dual quaternion (Subsection 5.2.2) operations found in Chapter (2). Furthermore, we acknowledge the DQ based operation for the adjoint mapping is not as optimized as some representations for the HTM based adjoint, thus, stemming from our work in [60] we propose in Subsection 5.2.3 an optimized version of (2.89).

5.2.1 Costs for Quaternion Operations

Here, Table 5.1 lists the storage and computational costs related to quaternion algebra.

Table 5.1: Cost requirements of quaternion algebra operations

Operation	Storage	\times/\div	$+/-$
Addition ($\mathbf{x}_1 + \mathbf{x}_2$) (2.3)	$8f$	$0\times$	$4+$
Multiplication by Scalar ($\alpha\mathbf{x}$) (2.4)	$5f$	$4\times$	$0+$
Quat. Multiplication $\mathbf{x}_1\mathbf{x}_2$ (2.5)	$8f$	$16\times$	$12+$
Quat. Cross Product (2.13)	$6f$	$9\times$	$5+$
Quat. Inner Product (2.14)	$6f$	$3\times$	$2+$
Matrix Linear Transf. \mathcal{T}_4 (2.49)	$20f$	$16\times$	$12+$
Quat. Hamilton Operator $\overset{+}{\mathbf{h}}(\bullet)$ or $\overset{-}{\mathbf{h}}(\bullet)$ (2.55)	$8f$	$16\times$	$12+$
Quat. Adjoint Transformation (2.78)	$7f$	$20\times$	$28+$

* where $\alpha \in \mathbb{R}$ and $\mathbf{x}_1, \mathbf{x}_2 \in \mathbb{H}$

Note that, although we reckon the same computational complexity could be directly extracted from basic operations, to explicitly state the compound operations is valid for future reference and computational cost analysis within the quaternion algebra.

5.2.2 Costs for Dual Quaternion Operations

In Table 5.2 we show a detailed list of dual quaternion operations with respective storage and elementary operation costs, following the methodology described in 5.1.

The costs of operations for dual quaternions are generally lower than for other algebras. In particular, the fact that the general dual quaternion needs only eight parameters is in itself an advantage, as a [110] argues the cost for memory access exceeds the cost for basic arithmetic operations. Moreover, although there are many other interesting proprieties that come with using a dual quaternion algebra for kinematic and dynamic

Table 5.2: Cost requirements for dual quaternion operations

Operation	Storage	\times/\div	$+/-$
Addition ($\underline{\mathbf{x}}_1 + \underline{\mathbf{x}}_2$)	$16f$	$0\times$	$8+$
Multiplication by Scalar ($\alpha\underline{\mathbf{x}}_1$)	$9f$	$8\times$	$0+$
Dual Quat. Multiplication ($\underline{\mathbf{x}}_1\underline{\mathbf{x}}_2$)	$16f$	$48\times$	$40+$
Cross Product (2.40)	$12f$	$18\times$	$12+$
Inner Product (2.41)	$12f$	$6\times$	$5+$
Double Geodesic Product (4.20)	$12f$	$6\times$	$5+$
Dual Quat Linear Transformation	$72f$	$64\times$	$56+$
Dual Quat. Hamilton Operator (2.58)	$28f$	$48\times$	$48+$
Adjoint Transformation (2.89)	$14f$	$84\times$	$70+$

* where $\alpha \in \mathbb{R}$ and $\underline{\mathbf{x}}_1, \underline{\mathbf{x}}_2 \in \mathbb{H}$

algorithms, we still aim to create a cost efficient representation. In this manner, we may notice from Tables 5.1 and 5.2 that the Adjoint transformation is generally quite costly in its traditional form. Thus, in the next section we present our optimization for this operation, and analyze its subsequent costs.

5.2.3 Optimizing the Adjoint Mapping

The usual adjoint mapping for dual quaternions, as defined in (2.89), can present quite a few issues for the cost efficient dqRNEA algorithm we aimed at designing. As we previously stated, many of our design choices were made to optimize the algorithm in terms of computational cost. More so, as discussed in Subsection 4.3.1, sometimes, in order to isolate mapped out variable in the equation we might also need to commute the elements on (2.89). Thus, taking these issues into account we proposed the alternative Definition 2.2 for the dual quaternion adjoint [60].

In order to derivate the alternative expressions (2.90) and (2.91) we should use as a starting point the traditional adjoint map of (2.89),

$$Ad(\underline{\mathbf{x}})\underline{\mathbf{p}} = \underline{\mathbf{x}}\underline{\mathbf{p}}\underline{\mathbf{x}}^*, \quad (5.4)$$

in which we have that $\underline{\mathbf{x}} = \mathbf{x}_P + \varepsilon\mathbf{x}_D$ and $\underline{\mathbf{p}} = \mathbf{p}_P + \varepsilon\mathbf{p}_D$. As stated, one of our aims of this alternative representation is to isolate the variable we are mapping out, which in (5.4) is $\underline{\mathbf{p}}$. Therefore, we use here the dual quaternion hamiltonians defined in (2.58), such that

$$Ad(\underline{\mathbf{x}})\underline{\mathbf{p}} = \overset{+}{\underline{\mathbf{h}}}(\underline{\mathbf{x}}) \overset{-}{\underline{\mathbf{h}}}(\underline{\mathbf{x}}^*) \underline{\mathbf{p}}. \quad (5.5)$$

Next, by expanding $\overset{-}{\underline{\mathbf{h}}}(\underline{\mathbf{x}}^*) = \overset{-}{\underline{\mathbf{h}}}(\mathbf{x}_P^*) + \varepsilon\overset{-}{\underline{\mathbf{h}}}(\mathbf{x}_D^*)$ and $\underline{\mathbf{p}}$ we have

$$\begin{aligned} Ad(\underline{\mathbf{x}})\underline{\mathbf{p}} &= \overset{+}{\underline{\mathbf{h}}}(\underline{\mathbf{x}}) \left(\overset{-}{\underline{\mathbf{h}}}(\mathbf{x}_P^*) + \varepsilon\overset{-}{\underline{\mathbf{h}}}(\mathbf{x}_D^*) \right) (\mathbf{p}_P + \varepsilon\mathbf{p}_D) \\ &= \overset{+}{\underline{\mathbf{h}}}(\underline{\mathbf{x}}) \left(\overset{-}{\underline{\mathbf{h}}}(\mathbf{x}_P^*) \mathbf{p}_P + \varepsilon \left(\overset{-}{\underline{\mathbf{h}}}(\mathbf{x}_P^*) \mathbf{p}_D + \overset{-}{\underline{\mathbf{h}}}(\mathbf{x}_D^*) \mathbf{p}_P \right) \right), \end{aligned} \quad (5.6)$$

and from $\overset{+}{\mathbf{h}}(\underline{\mathbf{x}}) = \overset{+}{\mathbf{h}}(\mathbf{x}_P) + \varepsilon \overset{+}{\mathbf{h}}(\mathbf{x}_D)$,

$$\begin{aligned}
Ad(\underline{\mathbf{x}})\underline{\mathbf{p}} &= \left(\overset{+}{\mathbf{h}}(\mathbf{x}_P) + \varepsilon \overset{+}{\mathbf{h}}(\mathbf{x}_D) \right) \left(\bar{\mathbf{h}}(\mathbf{x}_P^*) \mathbf{p}_P + \varepsilon \left(\bar{\mathbf{h}}(\mathbf{x}_P^*) \mathbf{p}_D + \bar{\mathbf{h}}(\mathbf{x}_D^*) \mathbf{p}_P \right) \right) \\
&= \left[\overset{+}{\mathbf{h}}(\mathbf{x}_P) \bar{\mathbf{h}}(\mathbf{x}_P^*) \right] \mathbf{p}_P + \varepsilon \left[\overset{+}{\mathbf{h}}(\mathbf{x}_P) \bar{\mathbf{h}}(\mathbf{x}_P^*) \right] \mathbf{p}_D + \varepsilon \left[\overset{+}{\mathbf{h}}(\mathbf{x}_P) \bar{\mathbf{h}}(\mathbf{x}_D^*) \right] \mathbf{p}_P + \varepsilon \left[\overset{+}{\mathbf{h}}(\mathbf{x}_D) \bar{\mathbf{h}}(\mathbf{x}_P^*) \right] \mathbf{p}_P \\
&= \left[\overset{+}{\mathbf{h}}(\mathbf{x}_P) \bar{\mathbf{h}}(\mathbf{x}_P^*) \right] \mathbf{p}_P + \varepsilon \left[\overset{+}{\mathbf{h}}(\mathbf{x}_P) \bar{\mathbf{h}}(\mathbf{x}_P^*) \right] \mathbf{p}_D + \varepsilon \left[\overset{+}{\mathbf{h}}(\mathbf{x}_P) \bar{\mathbf{h}}(\mathbf{x}_D^*) + \overset{+}{\mathbf{h}}(\mathbf{x}_D) \bar{\mathbf{h}}(\mathbf{x}_P^*) \right] \mathbf{p}_P.
\end{aligned} \tag{5.7}$$

Now let us define the auxiliary adjunct terms, \mathcal{A}_P and \mathcal{A}_D ,

$$\mathcal{A}_P(\underline{\mathbf{x}}) = \left[\overset{+}{\mathbf{h}}(\mathbf{x}_P) \bar{\mathbf{h}}(\mathbf{x}_P^*) \right] \text{ and } \mathcal{A}_D(\underline{\mathbf{x}}) = \left[\overset{+}{\mathbf{h}}(\mathbf{x}_P) \bar{\mathbf{h}}(\mathbf{x}_D^*) + \overset{+}{\mathbf{h}}(\mathbf{x}_D) \bar{\mathbf{h}}(\mathbf{x}_P^*) \right], \tag{5.8}$$

which allow us to rewrite (5.7) as

$$\begin{aligned}
Ad(\underline{\mathbf{x}})\underline{\mathbf{p}} &= \mathcal{A}_P(\underline{\mathbf{x}})\mathbf{p}_P + \varepsilon \mathcal{A}_P(\underline{\mathbf{x}})\mathbf{p}_D + \varepsilon \mathcal{A}_D(\underline{\mathbf{x}})\mathbf{p}_P \\
&= \mathcal{A}_P(\underline{\mathbf{x}})\mathbf{p}_P + \varepsilon (\mathcal{A}_P(\underline{\mathbf{x}})\mathbf{p}_D + \mathcal{A}_D(\underline{\mathbf{x}})\mathbf{p}_P).
\end{aligned} \tag{5.9}$$

Next, by using (2.55), we may further analyze the terms in (5.8). Thus we note that $\mathcal{A}_P\boldsymbol{\mu}$, for any $\boldsymbol{\mu} = \mu_0 + \mu_1\hat{i} + \mu_2\hat{j} + \mu_3\hat{k} \in \mathbb{H}_0$, can be written as

$$\begin{aligned}
\mathcal{A}_P(\underline{\mathbf{x}})\boldsymbol{\mu} &= \overset{+}{\mathbf{h}}(\mathbf{x}_P) \bar{\mathbf{h}}(\mathbf{x}_P^*) \boldsymbol{\mu} \\
&= \overset{+}{\mathbf{h}}(\mathbf{x}_P) \left(\mathbf{x}_P^* \mu_0 + \hat{i} \mathbf{x}_P^* \mu_1 + \hat{j} \mathbf{x}_P^* \mu_2 + \hat{k} \mathbf{x}_P^* \mu_3 \right) \\
&= \left(\mathbf{x}_P \mathbf{x}_P^* \mu_0 + \mathbf{x}_P \hat{i} \mathbf{x}_P^* \mu_1 + \mathbf{x}_P \hat{j} \mathbf{x}_P^* \mu_2 + \mathbf{x}_P \hat{k} \mathbf{x}_P^* \mu_3 \right),
\end{aligned} \tag{5.10}$$

and since $\boldsymbol{\mu} \in \mathbb{H}_0$, we have that $\mu_0 = 0$ and in turn (5.10) becomes

$$\begin{aligned}
\mathcal{A}_P(\underline{\mathbf{x}})\boldsymbol{\mu} &= (\mathbf{x}_P \hat{i} \mathbf{x}_P^*) \mu_1 + (\mathbf{x}_P \hat{j} \mathbf{x}_P^*) \mu_2 + (\mathbf{x}_P \hat{k} \mathbf{x}_P^*) \mu_3 \\
&= \mathcal{A}_{P_{\hat{i}}}\mu_1 + \mathcal{A}_{P_{\hat{j}}}\mu_2 + \mathcal{A}_{P_{\hat{k}}}\mu_3,
\end{aligned} \tag{5.11}$$

with $\mathcal{A}_{P_{\hat{i}}} = (\mathbf{x}_P \hat{i} \mathbf{x}_P^*)$, $\mathcal{A}_{P_{\hat{j}}} = (\mathbf{x}_P \hat{j} \mathbf{x}_P^*)$ and $\mathcal{A}_{P_{\hat{k}}} = (\mathbf{x}_P \hat{k} \mathbf{x}_P^*)$ being the individual base transformation.

We highlight here that $\mathcal{A}_P(\underline{\mathbf{x}})\boldsymbol{\mu}$ is the adjoint quaternion transformation defined in (2.78) and here taking the form of $\mathbf{x}_P \boldsymbol{\mu} \mathbf{x}_P^*$. Such concept for adjoint transformation is considerably intuitive in the sense that the classic quaternion adjoint transformation is used to either rotate the point $\boldsymbol{\mu}$ or to observe the same point in a different frame, the quaternion adjoint $\mathcal{A}_P(\underline{\mathbf{x}})\boldsymbol{\mu}$ redefines the quaternionic elements in the frame of interest. In other words, the adjoint transformation is performed for every quaternionic unit $\hat{i}, \hat{j}, \hat{k}$ and the point $\boldsymbol{\mu}$ is simple redefined within the new base.

Similarly to $\mathcal{A}_P(\underline{\mathbf{x}})$, we may rewrite the term $\mathcal{A}_D(\underline{\mathbf{x}})\boldsymbol{\mu}$, for any $\boldsymbol{\mu} = \mu_0 + \mu_1\hat{i} + \mu_2\hat{j} + \mu_3\hat{k} \in \mathbb{H}_0$, as

$$\begin{aligned}
\mathcal{A}_D(\underline{\mathbf{x}})\boldsymbol{\mu} &= \left[\overset{+}{\mathbf{h}}(\mathbf{x}_P) \bar{\mathbf{h}}(\mathbf{x}_D^*) + \overset{+}{\mathbf{h}}(\mathbf{x}_D) \bar{\mathbf{h}}(\mathbf{x}_P^*) \right] \boldsymbol{\mu} \\
&= \overset{+}{\mathbf{h}}(\mathbf{x}_P) \left(\mathbf{x}_D^* \mu_0 + \hat{i} \mathbf{x}_D^* \mu_1 + \hat{j} \mathbf{x}_D^* \mu_2 + \hat{k} \mathbf{x}_D^* \mu_3 \right) + \overset{+}{\mathbf{h}}(\mathbf{x}_D) \left(\mathbf{x}_P^* \mu_0 + \hat{i} \mathbf{x}_P^* \mu_1 + \hat{j} \mathbf{x}_P^* \mu_2 + \hat{k} \mathbf{x}_P^* \mu_3 \right) \\
&= \left(\mathbf{x}_P \mathbf{x}_D^* \mu_0 + \mathbf{x}_P \hat{i} \mathbf{x}_D^* \mu_1 + \mathbf{x}_P \hat{j} \mathbf{x}_D^* \mu_2 + \mathbf{x}_P \hat{k} \mathbf{x}_D^* \mu_3 \right) + \left(\mathbf{x}_D \mathbf{x}_P^* \mu_0 + \mathbf{x}_D \hat{i} \mathbf{x}_P^* \mu_1 + \mathbf{x}_D \hat{j} \mathbf{x}_P^* \mu_2 + \mathbf{x}_D \hat{k} \mathbf{x}_P^* \mu_3 \right).
\end{aligned} \tag{5.12}$$

As $\mu_0 = 0$ (μ is a pure quaternions) we can rewrite the equation as

$$\begin{aligned}\mathcal{A}_{\mathcal{D}}(\underline{\mathbf{x}})\boldsymbol{\mu} &= \left(\mathbf{x}_P\hat{\mathbf{i}}\mathbf{x}_D^*\mu_1 + \mathbf{x}_P\hat{\mathbf{j}}\mathbf{x}_D^*\mu_2 + \mathbf{x}_P\hat{\mathbf{k}}\mathbf{x}_D^*\mu_3\right) + \left(\mathbf{x}_D\hat{\mathbf{i}}\mathbf{x}_P^*\mu_1 + \mathbf{x}_D\hat{\mathbf{j}}\mathbf{x}_P^*\mu_2 + \mathbf{x}_D\hat{\mathbf{k}}\mathbf{x}_P^*\mu_3\right) \\ &= (\mathbf{x}_P\hat{\mathbf{i}}\mathbf{x}_D^* + \mathbf{x}_D\hat{\mathbf{i}}\mathbf{x}_P^*)\mu_1 + (\mathbf{x}_P\hat{\mathbf{j}}\mathbf{x}_D^* + \mathbf{x}_D\hat{\mathbf{j}}\mathbf{x}_P^*)\mu_2 + (\mathbf{x}_P\hat{\mathbf{k}}\mathbf{x}_D^* + \mathbf{x}_D\hat{\mathbf{k}}\mathbf{x}_P^*)\mu_3 \\ &= (\mathbf{x}_P\hat{\mathbf{i}}\mathbf{x}_D^* - (\mathbf{x}_P\hat{\mathbf{i}}\mathbf{x}_D^*)^*)\mu_1 + (\mathbf{x}_P\hat{\mathbf{j}}\mathbf{x}_D^* - (\mathbf{x}_P\hat{\mathbf{j}}\mathbf{x}_D^*)^*)\mu_2 + (\mathbf{x}_P\hat{\mathbf{k}}\mathbf{x}_D^* - (\mathbf{x}_P\hat{\mathbf{k}}\mathbf{x}_D^*)^*)\mu_3.\end{aligned}\quad (5.13)$$

Now, for an arbitrary pure quaternion \mathbf{q}_0 , let us consider an expression such as

$$\mathbf{q}_0 - \mathbf{q}_0^*. \quad (5.14)$$

We notice that (5.14) is the sum of a quaternion with the negative of its conjugate, and from (2.12), we have

$$\mathbf{q}_0 = \text{Im}(\mathbf{q}_0) = \frac{1}{2}(\mathbf{q}_0 - \mathbf{q}_0^*). \quad (5.15)$$

With the propriety described in (5.15), and knowing that the adjoint map of a pure quaternion yields a pure quaternion, we have that for any quaternionic unit $\hat{\eta} = \{\hat{i}, \hat{j}, \hat{k}\}$

$$\mathbf{x}_P\hat{\eta}\mathbf{x}_D^* - (\mathbf{x}_P\hat{\eta}\mathbf{x}_D^*)^* = 2\text{Im}(\mathbf{x}_P\hat{\eta}\mathbf{x}_D^*). \quad (5.16)$$

And thus, we may rewrite (5.13) as

$$\begin{aligned}\mathcal{A}_{\mathcal{D}}(\underline{\mathbf{x}})\boldsymbol{\mu} &= 2\text{Im}(\mathbf{x}_P\hat{\mathbf{i}}\mathbf{x}_D^*)\mu_1 + 2\text{Im}(\mathbf{x}_P\hat{\mathbf{j}}\mathbf{x}_D^*)\mu_2 + 2\text{Im}(\mathbf{x}_P\hat{\mathbf{k}}\mathbf{x}_D^*)\mu_3 \\ &= 2\left(\text{Im}(\mathbf{x}_P\hat{\mathbf{i}}\mathbf{x}_D^*)\mu_1 + \text{Im}(\mathbf{x}_P\hat{\mathbf{j}}\mathbf{x}_D^*)\mu_2 + \text{Im}(\mathbf{x}_P\hat{\mathbf{k}}\mathbf{x}_D^*)\mu_3\right) \\ &= \mathcal{A}_{D_i}\mu_1 + \mathcal{A}_{D_j}\mu_2 + \mathcal{A}_{D_k}\mu_3,\end{aligned}\quad (5.17)$$

with $\mathcal{A}_{D_i} = 2\text{Im}(\mathbf{x}_P\hat{\mathbf{i}}\mathbf{x}_D^*)$, $\mathcal{A}_{D_j} = 2\text{Im}(\mathbf{x}_P\hat{\mathbf{j}}\mathbf{x}_D^*)$ and $\mathcal{A}_{D_k} = 2\text{Im}(\mathbf{x}_P\hat{\mathbf{k}}\mathbf{x}_D^*)$.

Finally, going back to the adjoint definition in (5.9), we have

$$\begin{aligned}\text{Ad}(\underline{\mathbf{x}})\underline{\mathbf{p}} &= \underline{\mathbf{x}}\underline{\mathbf{p}}\underline{\mathbf{x}}^* \\ &= \mathcal{A}_{\mathcal{P}}(\underline{\mathbf{x}})\underline{\mathbf{p}}_{\mathcal{P}} + \varepsilon(\mathcal{A}_{\mathcal{P}}(\underline{\mathbf{x}})\underline{\mathbf{p}}_{\mathcal{D}} + \mathcal{A}_{\mathcal{D}}(\underline{\mathbf{x}})\underline{\mathbf{p}}_{\mathcal{P}}),\end{aligned}\quad (5.18)$$

with $\mathcal{A}_{\mathcal{P}}(\underline{\mathbf{x}})$ and $\mathcal{A}_{\mathcal{D}}(\underline{\mathbf{x}})$ being defined respectively as (5.11) and (5.17).

Remark 5.1. To close of the demonstration for the alternative adjoint we should once more return to (5.17), in which, due to the fact that for any quaternionic unit $\hat{\eta} = \{\hat{i}, \hat{j}, \hat{k}\}$, we have that $\mathbf{x}_P\hat{\eta}\mathbf{x}_D^*$ is a pure quaternion, and consequently $\mathbf{x}_P\hat{\eta}\mathbf{x}_D^* = \text{Im}(\mathbf{x}_P\hat{\eta}\mathbf{x}_D^*)$, we have

$$\begin{aligned}\mathcal{A}_{\mathcal{D}}(\underline{\mathbf{x}})\boldsymbol{\mu} &= 2\left(\text{Im}(\mathbf{x}_P\hat{\mathbf{i}}\mathbf{x}_D^*)\mu_1 + \text{Im}(\mathbf{x}_P\hat{\mathbf{j}}\mathbf{x}_D^*)\mu_2 + \text{Im}(\mathbf{x}_P\hat{\mathbf{k}}\mathbf{x}_D^*)\mu_3\right) \\ &= 2\left(\mathbf{x}_P\hat{\mathbf{i}}\mathbf{x}_D^*\mu_1 + \mathbf{x}_P\hat{\mathbf{j}}\mathbf{x}_D^*\mu_2 + \mathbf{x}_P\hat{\mathbf{k}}\mathbf{x}_D^*\mu_3\right) \\ &= 2\overset{+}{\mathbf{h}}(\mathbf{x}_P)\left(\hat{\mathbf{i}}\mathbf{x}_D^*\mu_1 + \hat{\mathbf{j}}\mathbf{x}_D^*\mu_2 + \hat{\mathbf{k}}\mathbf{x}_D^*\mu_3\right) \\ &= 2\overset{+}{\mathbf{h}}(\mathbf{x}_P)\bar{\mathbf{h}}(\mathbf{x}_D^*)\boldsymbol{\mu}.\end{aligned}\quad (5.19)$$

Furthermore from (5.10) we also have that

$$\mathcal{A}_{\mathcal{P}}(\underline{\mathbf{x}})\boldsymbol{\mu} = \overset{+}{\mathbf{h}}(\mathbf{x}_P)\bar{\mathbf{h}}(\mathbf{x}_P^*)\boldsymbol{\mu}. \quad (5.20)$$

Thus we may similarly represent the adjoint transformation as introduced in [60] and Definition 2.2, that is, defining

$$\mathbf{A}_P(\underline{\mathbf{x}}) = \overset{+}{\mathbf{h}}(\mathbf{x}_P) \bar{\mathbf{h}}(\mathbf{x}_P^*) \quad \text{and} \quad \mathbf{A}_D(\underline{\mathbf{x}}) = 2\overset{+}{\mathbf{h}}(\mathbf{x}_P) \bar{\mathbf{h}}(\mathbf{x}_D^*) \quad (5.21)$$

and

$$Ad(\underline{\mathbf{x}})\underline{\mathbf{p}} = \mathbf{A}_P(\underline{\mathbf{x}})\underline{\mathbf{p}}_P + \varepsilon(\mathbf{A}_P(\underline{\mathbf{x}})\underline{\mathbf{p}}_D + \mathbf{A}_D(\underline{\mathbf{x}})\underline{\mathbf{p}}_P). \quad (5.22)$$

The representation in (5.11), (5.17) and (5.18) is equivalent to the one in Definition 2.2, which is what was used in [60] for lowering the computational costs of the algorithm. However, we believe that $\mathcal{A}_P(\underline{\mathbf{x}})$ and $\mathcal{A}_D(\underline{\mathbf{x}})$ are both more intuitive reformulations for the adjoint operator, as well as more cost efficient.

Remark 5.2. Finally, we also remark here how this new representation of the adjoint works with our Algorithm ??.

In this case, we shall compute $\mathcal{A}_P(\underline{\mathbf{x}})$ and $\mathcal{A}_D(\underline{\mathbf{x}})$ (or instead $\mathbf{A}_P(\underline{\mathbf{x}})$ and $\mathbf{A}_D(\underline{\mathbf{x}})$) and only use the value stored in the memory for the subsequent transformations.

Thus, we have Algorithm 5.1, in which we add the computation of $\mathcal{A}_P(\underline{\mathbf{x}})$ and $\mathcal{A}_D(\underline{\mathbf{x}})$.

Algorithm 5.1 Dual-quaternion based recursive Newton-Euler Inverse Dynamics for n DoF Manipulator

Initialization:

At the home config., set the frames \mathcal{F}_0 to the base, \mathcal{F}_1 to \mathcal{F}_n to the n -links' center of mass, and \mathcal{F}_{n+1} to the end-pose.

Set $\underline{\delta}_{i-1}^i, \underline{\mathbf{a}}_i$ as the pose of \mathcal{F}_{i-1} and the screw axis of joint i expressed in \mathcal{F}_i , with null vel. $\underline{\omega}_0=0$ and gravity accel. $\underline{\dot{\omega}}_0 = -\varepsilon \mathbf{g}$ at base and end-pose wrench $\underline{\mathbf{f}}_{-EF} = \underline{\mathbf{f}}_{EF} + \varepsilon \mathbf{m}_{EF}$.

Forward iteration

$$\begin{aligned} \underline{\mathbf{x}}_i^{i-1}(\theta_i(t)) &= \underline{\delta}_{i-1}^i \exp(\underline{\mathbf{a}}_i \frac{\Delta\theta_i}{2}) \\ \text{Compute } \mathcal{A}_P(\underline{\mathbf{x}}_i^{i-1}) \text{ and } \mathcal{A}_D(\underline{\mathbf{x}}_i^{i-1}) &\text{ from (2.91)} \\ \underline{\omega}_i &= \underline{\mathbf{a}}_i \dot{\theta}_i(t) + Ad(\underline{\mathbf{x}}_i^{i-1})\underline{\omega}_{i-1} \\ \underline{\dot{\omega}}_i &= \underline{\mathbf{a}}_i \ddot{\theta}_i(t) + Ad(\underline{\mathbf{x}}_i^{i-1})\underline{\dot{\omega}}_{i-1} + (\underline{\omega}_i \times \underline{\mathbf{a}}_i) \dot{\theta}_i(t) \end{aligned}$$

Backward iteration

$$\begin{aligned} \underline{\mathbf{f}}_{R,i} &= \underline{\omega}_i \times (G_i(\underline{\omega}_i)) + G_i(\underline{\dot{\omega}}_i) \\ \underline{\mathbf{f}}_i &= \underline{\mathbf{f}}_{R,i} + Ad(\underline{\mathbf{x}}_i^{i+1})\underline{\mathbf{f}}_{i+1} \\ \tau_i &= \underline{\mathbf{f}}_i \odot \underline{\mathbf{a}}_i \end{aligned}$$

5.2.3.1 Computational Costs for the new adjoint representation

When computing the costs of the adjoint transformations presented in this section we must take in consideration two different steps:

1. The costs for calculating the adjoint operands ($\mathcal{A}_P(\underline{\mathbf{x}})$ and $\mathcal{A}_D(\underline{\mathbf{x}})$ or $\mathbf{A}_P(\underline{\mathbf{x}})$ and $\mathbf{A}_D(\underline{\mathbf{x}})$);
2. The costs of the adjoint transformation as described in (5.18).

Regarding $\mathcal{A}_P(\underline{\mathbf{x}})$ in (5.18), we may analyze the cost equations and notice that (5.11) has the individual elements $\mathcal{A}_{P_i} = (\mathbf{x}_P \hat{i} \mathbf{x}_P^*)$, $\mathcal{A}_{P_j} = (\mathbf{x}_P \hat{j} \mathbf{x}_P^*)$ and $\mathcal{A}_{P_k} = (\mathbf{x}_P \hat{k} \mathbf{x}_P^*)$. In this case,

$$\text{cost}_{\text{Total}}(\mathbf{x}_P \hat{i}) = (4f, 4\times, 0+) \quad (5.23)$$

and $\text{cost}_{\text{Total}}((\mathbf{x}_P \hat{i}) \mathbf{x}_P^*)$ will be equal to the cost for (5.23) plus the cost for the the multiplication of two quaternions, that is

$$\text{cost}_{\text{Total}}(\mathbf{x}_P \hat{i} \mathbf{x}_P^*) = (4f, 4\times, 0+) + (0f, 16\times, 12+) = (4f, 20\times, 12+). \quad (5.24)$$

Thus, noting that the costs for \mathcal{A}_{P_j} and \mathcal{A}_{P_k} are the same as (5.24), we have a total cost for computing $\mathcal{A}_P(\underline{\mathbf{x}})$

$$\text{cost}_{\text{Total}}(\mathcal{A}_P(\underline{\mathbf{x}})) = 3(4f, 20\times, 12+) - \underbrace{2(4f, 0\times, 0+)}_{\text{Repeated } \mathfrak{x}_P} = (4f, 60\times, 36+).$$

Taking into consideration the cost for the adjoint transformation once $\mathcal{A}_P(\underline{\mathbf{x}})$ is already built and stored, we must look at (5.11) once more. That is we must compute

$$\text{cost}_{\text{Total}}(\mathcal{A}_{P_i} \mu_1 + \mathcal{A}_{P_j} \mu_2 + \mathcal{A}_{P_k} \mu_3) = \underbrace{\text{cost}_{\text{Total}}(\mathcal{A}_{P_i} \mu_1)}_{(5f, 3\times, 0+)} + \underbrace{\text{cost}_{\text{Total}}(\mathcal{A}_{P_j} \mu_2)}_{(5f, 3\times, 3+)} + \underbrace{\text{cost}_{\text{Total}}(\mathcal{A}_{P_k} \mu_3)}_{(5f, 3\times, 3+)}. \quad (5.25)$$

Which yields,

$$\text{cost}_{\text{Total}}(\mathcal{A}_P(\underline{\mathbf{x}}) \boldsymbol{\mu}) = (15f, 9\times, 6+). \quad (5.26)$$

Similarly we may calculate the costs for \mathcal{A}_D . In this case, from (5.17) we have the individual elements that form \mathcal{A}_D being $\mathcal{A}_{D_i} = 2 \text{Im}(\mathbf{x}_P \hat{i} \mathbf{x}_D^*)$, $\mathcal{A}_{D_j} = 2 \text{Im}(\mathbf{x}_P \hat{j} \mathbf{x}_D^*)$ and $\mathcal{A}_{D_k} = 2 \text{Im}(\mathbf{x}_P \hat{k} \mathbf{x}_D^*)$. Here we notice that, due to (2.81)

$$\begin{aligned} \mathcal{A}_{D_i} &= 2 \text{Im}(\mathbf{x}_P \hat{i} \mathbf{x}_D^*) = 2 \text{Im}(\mathbf{x}_P \hat{i} (0.5 \underline{\mathbf{x}}. \text{translation} \mathbf{x}_P)^*) \\ &= \text{Im}(\mathbf{x}_P \hat{i} \mathbf{x}_P (\underline{\mathbf{x}}. \text{translation})), \end{aligned} \quad (5.27)$$

but $\mathbf{x}_P \hat{i} \mathbf{x}_P$ ($\mathbf{x}_P \hat{j} \mathbf{x}_P$ or $\mathbf{x}_P \hat{k} \mathbf{x}_P$) has already been calculated in 5.24, thus we do not need to compute this again only retrieve it from memory. Thus, the total cost will be only one quaternion multiplication

$$\text{cost}_{\text{Total}}(\text{Im}((\mathbf{x}_P \hat{i} \mathbf{x}_P) (\underline{\mathbf{x}}. \text{translation}))) = (8f, 16\times, 12+). \quad (5.28)$$

Furthermore the cost for \mathcal{A}_D will be

$$\text{cost}_{\text{Total}}(\mathcal{A}_D(\underline{\mathbf{x}})) = 3(8f, 16\times, 12+) - \underbrace{2(8f, 0\times, 0+)}_{\text{Repeated } \mathfrak{x}_P \ \mathfrak{x}_D} = (8f, 48\times, 36+). \quad (5.29)$$

With \mathcal{A}_D built and stored we can calculate the cost for the $\mathcal{A}_D(\underline{\mathbf{x}}) \boldsymbol{\mu}$, thus, from (5.17) we have the exact same cost as for (5.25),

$$\begin{aligned} \text{cost}_{\text{Total}}(\mathcal{A}_{D_i} \mu_1 + \mathcal{A}_{D_j} \mu_2 + \mathcal{A}_{D_k} \mu_3) &= \underbrace{\text{cost}_{\text{Total}}(\mathcal{A}_{D_i} \mu_1)}_{(5f, 3\times, 0+)} + \underbrace{\text{cost}_{\text{Total}}(\mathcal{A}_{D_j} \mu_2)}_{(5f, 3\times, 3+)} + \underbrace{\text{cost}_{\text{Total}}(\mathcal{A}_{D_k} \mu_3)}_{(5f, 3\times, 3+)} \\ \text{cost}_{\text{Total}}(\mathcal{A}_D(\underline{\mathbf{x}}) \boldsymbol{\mu}) &= (15f, 9\times, 6+). \end{aligned} \quad (5.30)$$

Finally, we have the cost for the whole adjoint transformation, defined in (5.18). Here we consider that \mathcal{A}_P and \mathcal{A}_D have already been computed, thus we shall only consider the costs for fetching them from the memory and multiplying by a pure quaternion, as in (5.26) and (5.30), thus

$$\begin{aligned} \text{cost}_{\text{Total}}(Ad(\underline{\mathbf{x}}) \boldsymbol{p}) &= \text{cost}_{\text{Total}}(\mathcal{A}_P(\underline{\mathbf{x}}) \boldsymbol{p}_P) + \text{cost}_{\text{Total}}(\mathcal{A}_P(\underline{\mathbf{x}}) \boldsymbol{p}_D + \mathcal{A}_D(\underline{\mathbf{x}}) \boldsymbol{p}_P) \\ &= \underbrace{(15f, 9\times, 9+)}_{\mathcal{A}_P(\underline{\mathbf{x}}) \boldsymbol{p}_P} + \underbrace{(15f, 9\times, 9+)}_{\mathcal{A}_P(\underline{\mathbf{x}}) \boldsymbol{p}_D} + \underbrace{(15f, 9\times, 9+)}_{\mathcal{A}_D(\underline{\mathbf{x}}) \boldsymbol{p}_P} + \underbrace{(0f, 0\times, 3+)}_{\text{Addition in } \mathbb{H}_0} - \underbrace{(15f, 0\times, 0+)}_{\text{Repeated } \mathcal{A}_P, \boldsymbol{p}_P}, \end{aligned} \quad (5.31)$$

yielding a total cost of

$$\text{cost}_{\text{Total}}(Ad(\underline{\mathbf{x}})\underline{\mathbf{p}}) = (30f, 27\times, 30+). \quad (5.32)$$

The cost for operators $\mathbf{A}_P(\underline{\mathbf{x}})$ and $\mathbf{A}_D(\underline{\mathbf{x}})$ is a bit more straightforward to calculate. In this case we have that $\mathbf{A}_P(\underline{\mathbf{x}})$ has the same cost as a Hamiltonian operators for quaternion, (2.55). Thus

$$\text{cost}_{\text{Total}}(\mathbf{A}_P(\underline{\mathbf{x}})) = \text{cost}_{\text{Total}}\left(\overset{+}{\mathbf{h}}(\mathbf{x}_P)\bar{\mathbf{h}}(\mathbf{x}_P^*)\right) = (4f, 16\times, 12+) \quad (5.33)$$

and $\mathbf{A}_D(\underline{\mathbf{x}})$ has also the same cost, with the added scalar multiplication,

$$\text{cost}_{\text{Total}}(\mathbf{A}_D(\underline{\mathbf{x}})) = \text{cost}_{\text{Total}}\left(2\overset{+}{\mathbf{h}}(\mathbf{x}_P)\bar{\mathbf{h}}(\mathbf{x}_D^*)\right) = (8f, 20\times, 12+). \quad (5.34)$$

Furthermore the cost for $\mathbf{A}_P(\underline{\mathbf{x}})\underline{\boldsymbol{\mu}}$ (or $\mathbf{A}_D(\underline{\mathbf{x}})\underline{\boldsymbol{\mu}}$) is the cost of a further Hamiltonian operator

$$\text{cost}_{\text{Total}}(\mathbf{A}_P(\underline{\mathbf{x}})\underline{\boldsymbol{\mu}}) = (14f, 20\times, 12+). \quad (5.35)$$

Finally the adjoint transformation as in Definition 2.2 has the total cost of

$$\begin{aligned} \text{cost}_{\text{Total}}(Ad(\underline{\mathbf{x}})\underline{\mathbf{p}}) &= \text{cost}_{\text{Total}}(\mathbf{A}_P(\underline{\mathbf{x}})\underline{\mathbf{p}}_P) + \text{cost}_{\text{Total}}(\mathbf{A}_P(\underline{\mathbf{x}})\underline{\mathbf{p}}_D + \mathbf{A}_D(\underline{\mathbf{x}})\underline{\mathbf{p}}_P) \\ &= \underbrace{(14f, 20\times, 12+)}_{\mathbf{A}_P(\underline{\boldsymbol{\mu}})\underline{\mathbf{p}}_P} + \underbrace{(14f, 20\times, 12+)}_{\mathbf{A}_P(\underline{\boldsymbol{\mu}})\underline{\mathbf{p}}_D} + \underbrace{(14f, 20\times, 12+)}_{\mathbf{A}_D(\underline{\boldsymbol{\mu}})\underline{\mathbf{p}}_P} + \underbrace{3(0f, 0\times, 3+)}_{\text{Addition in } \mathbb{H}_0} - \underbrace{(14f, 0\times, 0+)}_{\text{Repeated } \mathbf{A}_P, \underline{\mathbf{p}}_P}, \end{aligned} \quad (5.36)$$

yielding a total cost of

$$\text{cost}_{\text{Total}}(Ad(\underline{\mathbf{x}})\underline{\mathbf{p}}) = (28f, 60\times, 39+). \quad (5.37)$$

Before closing up this section we also present Table 5.3, in which we see the total weight the costs of the adjoint transformations would have in the whole Algorithm 4.2—that is the cost for constructing the matrix and the cost for all three times it appears throughout the algorithm.

We notice here that, as we are not retrieving from memory $\mathcal{A}_P(\underline{\mathbf{x}})$ and $\mathcal{A}_D(\underline{\mathbf{x}})$ (or $\mathbf{A}_P(\underline{\mathbf{x}})$ and $\mathbf{A}_D(\underline{\mathbf{x}})$) the cost for the storage for the traditional adjoint transformation is considerably less than the for the other representations, however, the number of additions and multiplications in this case are much greater. We also notice here that for the Adjoint Map of Subsection 5.2.3 the operator's construction cost is higher than for Adjoint Map in Definition 2.2, however the added costs for each recursion are generally lesser. Indeed, for our particular algorithm we believe the two more optimized solutions are better, with (5.18) having a slight advantage.

5.3 COMPUTATIONAL COST OF DQRNEA

In this section we shall use the data of cost operations within the unit dual quaternion framework, as presented in Tables 5.1 and 5.2 in order to calculate the costs for dqRNEA algorithm. We can compute the total cost for the forward and backward iteration of the proposed dual quaternion based Newton-Euler inverse dynamics algorithm for a n -joints serial manipulator. Such results are presented on Table III.

Starting out from (3.56) we have, as presented in [40], we have the cost for n consecutive dual quaternion multiplication

$$\text{cost}_{\text{Total}}(\underline{\mathbf{x}}_{EF}) = \text{cost}_{\text{Total}}(\underline{\mathbf{x}}_1^0 \underline{\mathbf{x}}_2^1 \dots \underline{\mathbf{x}}_{n+1}^n) = n(8f, 48\times, 40+). \quad (5.38)$$

Table 5.3: Comparison between different manners of performing the adjoint mapping

Mapping Type	Cost			Observation
	Storage	\times/\div	$+/-$	
Traditional Adjoint Map (2.89)	$14f$	$84\times$	$70+$	Costs from (2.89)
	$42f$	$252\times$	$210+$	Total for Algorithm 5.1
Adjoint Map in Definition 2.2	$4f$	$16\times$	$12+$	$\mathcal{A}_P(\underline{\mathbf{x}})$: Needs to be computed only once per recursion
	$8f$	$20\times$	$12+$	$\mathcal{A}_D(\underline{\mathbf{x}})$: Needs to be computed only once per recursion
	$28f$	$60\times$	$39+$	$Ad(\underline{\mathbf{x}})\underline{\mathbf{p}}$: Computed multiple times per recursion
	$96f$	$216\times$	$141+$	Total for Algorithm 5.1
Adjoint Map of Subsection 5.2.3	$4f$	$60\times$	$36+$	$\mathcal{A}_P(\underline{\mathbf{x}})$: Needs to be computed only once per recursion
	$8f$	$48\times$	$36+$	$\mathcal{A}_D(\underline{\mathbf{x}})$: Needs to be computed only once per recursion
	$30f$	$27\times$	$30+$	$Ad(\underline{\mathbf{x}})\underline{\mathbf{p}}$: Computed multiple times per recursion
	$102f$	$189\times$	$162+$	Total for Algorithm 5.1

Before proceeding for the costs of the other equations, we also find it interesting highlight that the cost of (3.56) when parametrized with the PoE formulation is much smaller than the one with DH Parameters [40]. That is, due to (3.57) we have that each element in (3.56) has three extra dual quaternion multiplications imbued in this computation, thus

$$\text{cost}_{\text{Total}}(\underline{\mathbf{x}}_{EF}^{DH}) = 3n(8f, 48\times, 40+), \quad (5.39)$$

which corroborate the choice for using the PoE representation in our cost effective algorithm.

Next, we calculate the cost for the twist equation, (4.7),

$$\begin{aligned} \text{cost}_{\text{Total}}(\underline{\boldsymbol{\omega}}_i) &= \text{cost}_{\text{Total}}\left(\underline{\mathbf{a}}_i \Delta \dot{\boldsymbol{\theta}}_i(t) + Ad(\underline{\mathbf{x}}_i^{i-1}) \underline{\boldsymbol{\omega}}_{i-1}\right) \\ &= \underbrace{\text{cost}_{\text{Total}}\left(\underline{\mathbf{a}}_i \Delta \dot{\boldsymbol{\theta}}_i(t)\right)}_{\text{Pure DQ times Scalar}} + \text{cost}_{\text{Total}}(\text{Pure DQ Add.}) + \text{cost}_{\text{Total}}\left(Ad(\underline{\mathbf{x}}_i^{i-1}) \underline{\boldsymbol{\omega}}_{i-1}\right) \\ &= (7f, 6\times, 0+) + (0f, 0\times, 6+) + (30f, 27\times, 30+) \\ &= (37f, 33\times, 36+). \end{aligned} \quad (5.40)$$

As the algorithm is repeated n -times we have the total cost being $n(37f, 33\times, 36+)$. Here we must also add that the adjoint is in our new formulation proposed in Subsection 5.2.3. When using the traditional adjoint we would need to replace $(30f, 27\times, 30+)$ by $(14f, 84\times, 70+)$, thus resulting in a total cost of $n(21f, 90\times, 76+)$.

Regarding 4.18 we have

$$\begin{aligned} \text{cost}_{\text{Total}}(\dot{\underline{\boldsymbol{\omega}}}_i) &= \text{cost}_{\text{Total}}\left(\underline{\mathbf{a}}_i \Delta \ddot{\boldsymbol{\theta}}_i(t) + Ad(\underline{\mathbf{x}}_i^{i-1}) \dot{\underline{\boldsymbol{\omega}}}_{i-1} + (\underline{\boldsymbol{\omega}}_i \times \underline{\mathbf{a}}_i) \Delta \dot{\boldsymbol{\theta}}_i(t)\right) \\ &= \underbrace{\text{cost}_{\text{Total}}\left(\underline{\mathbf{a}}_i \Delta \ddot{\boldsymbol{\theta}}_i(t)\right)}_{\text{Pure DQ times Scalar}} + 2\text{cost}_{\text{Total}}(\text{Pure DQ Add.}) + \text{cost}_{\text{Total}}\left(Ad(\underline{\mathbf{x}}_i^{i-1}) \dot{\underline{\boldsymbol{\omega}}}_{i-1}\right) + \underbrace{\text{cost}_{\text{Total}}\left((\underline{\boldsymbol{\omega}}_i \times \underline{\mathbf{a}}_i) \Delta \dot{\boldsymbol{\theta}}_i(t)\right)}_{\substack{\text{DQ cross Product} \\ \text{Pure DQ times Scalar}}} \\ &= (7f, 6\times, 0+) + 2(0f, 0\times, 6+) + (30f, 27\times, 30+) + (6f, 18\times, 12+) + (1f, 6\times, 0+) \\ &= (44f, 57\times, 54+). \end{aligned} \quad (5.41)$$

Once more, we reiterate that this was computed using the adjoint, of Subsection 5.2.3. When using the traditional adjoint the cost would go up to $(28f, 114\times, 94+)$. Also, (5.41) is repeated n -times bringing the total cost to $n(44f, 57\times, 54+)$ (or $n(28f, 114\times, 94+)$)

Now, we go to the backward iteration of the algorithm. In this case we shall start with the cost for the Dual Quaternion Inertia Transformation, from (3.44),

$$\begin{aligned}
\text{cost}_{\text{Total}}(G_i(\underline{\mathbf{q}})) &= \text{cost}_{\text{Total}}\left(m\mathcal{D}(\underline{\mathbf{q}}) + \varepsilon\left(\mathbf{I}_x^b q_x + \mathbf{I}_y^b q_y + \mathbf{I}_z^b q_z\right)\right) \\
&= \underbrace{\text{cost}_{\text{Total}}(m\mathcal{D}(\underline{\mathbf{q}}))}_{\text{Pure Quat. times Scalar}} + \underbrace{\text{cost}_{\text{Total}}\left(\mathbf{I}_x^b q_x + \mathbf{I}_y^b q_y + \mathbf{I}_z^b q_z\right)}_{\substack{3\text{times Pure Quat. times Scalar} \\ 2\text{ Pure quaternion Add}}} \\
&= (16f, 12\times, 6+).
\end{aligned} \tag{5.42}$$

Then, we may calculate the cost of (4.19)

$$\begin{aligned}
\text{cost}_{\text{Total}}(\underline{\mathbf{f}}_{R,i}) &= \text{cost}_{\text{Total}}(\underline{\boldsymbol{\omega}}_i \times (G_i(\underline{\boldsymbol{\omega}}_i)) + G_i(\underline{\dot{\boldsymbol{\omega}}}_i)) \\
&= \underbrace{\text{cost}_{\text{Total}}(\underline{\boldsymbol{\omega}}_i \times (G_i(\underline{\boldsymbol{\omega}}_i)))}_{G \text{ function} + \text{Cross Product}} + \text{cost}_{\text{Total}}(\text{Pure DQ Add.}) + \text{cost}_{\text{Total}}(G_i(\underline{\dot{\boldsymbol{\omega}}}_i)) \\
&= (16f, 12\times, 6+) + (0f, 18\times, 12+) + (0f, 0\times, 6+) + (6f, 12\times, 6+) \\
&= (22f, 42\times, 30+),
\end{aligned} \tag{5.43}$$

which becomes $n(22f, 42\times, 30+)$ when considering the calculation for all n links.

The next cost is for the resulting force, in (4.20),

$$\begin{aligned}
\text{cost}_{\text{Total}}(\underline{\mathbf{f}}_i) &= \text{cost}_{\text{Total}}\left(\underline{\mathbf{f}}_{R,i} + \text{Ad}(\underline{\mathbf{x}}_i^{i+1}) \underline{\mathbf{f}}_{i+1}\right) \\
&= \text{cost}_{\text{Total}}(\underline{\mathbf{f}}_{R,i}) + \text{cost}_{\text{Total}}(\text{Pure DQ Add.}) + \text{cost}_{\text{Total}}\left(\text{Ad}(\underline{\mathbf{x}}_i^{i+1}) \underline{\mathbf{f}}_{i+1}\right) \\
&= (6f, 0\times, 0+) + (0f, 0\times, 6+) + (30f, 27\times, 30+) \\
&= (36f, 27\times, 36+).
\end{aligned} \tag{5.44}$$

In this case we will have a total of $n(36f, 27\times, 36+)$ for the whole n -link manipulator and $n(20f, 84\times, 76+)$ for the traditional adjoint operation.

Finally, the cost for the torque is

$$\begin{aligned}
\text{cost}_{\text{Total}}(\tau_i) &= \text{cost}_{\text{Total}}\left(\underline{\mathbf{f}}_i \odot [\mathcal{D}(\underline{\mathbf{a}}_i) + \varepsilon\mathcal{P}(\underline{\mathbf{a}}_i)]\right) \\
&= \text{cost}_{\text{Total}}(\text{DG Product}) \\
&= (12f, 6\times, 5+),
\end{aligned} \tag{5.45}$$

and $n(12f, 6\times, 5+)$ for all links.

Table 5.4 shows the total cost of operations for different formulations of the the algorithm. Our preferred version uses the PoE formulation, dual quaternion algebra and the adjoint proposed in this chapter is presented

Table 5.4: Cost of operations for UDQ

Oper.	DQ - PoE - New Adjoint			DQ - PoE - Old Adjoint			DQ - DH - New Adjoint			DQ - DH - Old Adjoint		
	Storage(f)	\times/\div	$+/-$	Storage(f)	\times/\div	$+/-$	Storage(f)	\times/\div	$+/-$	Storage(f)	\times/\div	$+/-$
Eq. (3.56)	$8n$	$48n$	$40n$	$8n$	$48n$	$40n$	$24n$	$144n$	$120n$	$24n$	$144n$	$120n$
Eqs. (5.11, 5.17)	$12n$	$112n$	$72n$	0	0	0	$12n$	$112n$	$72n$	0	0	0
Eq. (4.7)	$37n$	$33n$	$36n$	$21n$	$90n$	$76n$	$37n$	$33n$	$36n$	$21n$	$90n$	$76n$
Eq. (4.18)	$44n$	$57n$	$54n$	$28n$	$114n$	$94n$	$44n$	$57n$	$54n$	$28n$	$114n$	$94n$
Eq. (4.19)	$22n$	$42n$	$30n$	$22n$	$42n$	$30n$	$22n$	$42n$	$30n$	$22n$	$42n$	$30n$
Eq. (4.20)	$36n$	$27n$	$36n$	$20n$	$84n$	$76n$	$36n$	$27n$	$36n$	$20n$	$84n$	$76n$
Eq. (4.22)	$12n$	$6n$	$5n$	$12n$	$6n$	$5n$	$12n$	$6n$	$5n$	$12n$	$6n$	$5n$
TOTAL	$171n$	$325n$	$273n$	$111n$	$384n$	$321n$	$187n$	$421n$	$353n$	$127n$	$480n$	$401n$

in the first column of the Table. This version proves to be quite good as it has a lesser general cost, even though storage is a little higher in this case. The last two columns in Table 5.4 show how the cost would be with the forward kinematics being calculated with DH parameters. These two columns were just added here for comparison purposes, however, we remark that there must be a transformation of the center of mass for us to use the DH parameters with our formulation, and this would imply in an even bigger cost for all subsequent operations.

5.4 COMPUTATIONAL COSTS OF HOMOGENEOUS TRANSFORMATION MATRICES

In this Section we shall present the costs for an alternative RNEA algorithm based on Homogeneous Transformation Matrices (HTM), as per Appendix A. Here we have calculated the computational costs based on the same methodology used in Section 5.1, in which we divide the costs between storage addition/subtraction and multiplication/division. In Subsection 5.4.1 we show both the cost for HTM operations and for vectors in \mathbb{R}^6 . Moreover, in Subsection 5.4.2 we show the the costs for the algorithm as a whole.

5.4.1 Cost of HTM operations

In this Subsection we present the costs for the main operations used for the RNEA algorithm based on HTM.

Table 5.5 shows the costs for the main operations for the vectors introduced in Subsection A.1.1, with $\mathbf{X}_1, \mathbf{X}_2 \in \mathbb{R}^6$ and $\alpha \in \mathbb{R}$. We highlight that we also show the costs for the vector adjoint multiplying a vector in \mathbb{R}^6 , such as in (A.8).

Furthermore, Table 5.6 is concerned with the costs for the homogeneous transformation matrices from Subsection A.1.3, with $\mathbf{X}_1 \in \mathbb{R}^6$, $\alpha \in \mathbb{R}$ and $\mathbf{T}_1, \mathbf{T}_2 \in \mathbb{R}^{4 \times 4}$. Moreover, the first four lines of the table deal with the traditional multiplication and addition operations, but the las two lines deal with the adjoint

Table 5.5: Cost requirements for vector operations

Operation	Storage	\times/\div	$+/-$
Addition ($\mathbf{X}_1 + \mathbf{X}_2$)	$12f$	$0\times$	$6+$
Multiplication by Scalar ($\alpha\mathbf{X}_1$)	$7f$	$6\times$	$0+$
Vector Multiplication $\mathbf{X}_1^T \mathbf{X}_2$	$12f$	$6\times$	$5+$
Vector Adjoint (A.8)	$42f$	$36\times$	$30+$

transformation. Adjoint construction is the cost for building the adjoint matrix, in (A.17)—that is, the cost for the product $[p] \mathbf{R}$, and the adjoint transformation is the cost for the a 6×6 matrix multiplying a \mathbb{R}^6 vector.

Table 5.6: Cost requirements for HTM operations

Operation	Storage	\times/\div	$+/-$
Addition ($\mathbf{T}_1 + \mathbf{T}_2$)	$32f$	$0\times$	$16+$
Multiplication by Scalar ($\alpha\mathbf{T}_1$)	$17f$	$16\times$	$0+$
Matrix-by-Matrix Multiplication $\mathbf{T}_1 \mathbf{T}_2$	$32f$	$64\times$	$48+$
Matrix-by-Vector Multiplication $\mathbf{T} \mathbf{X}_1$	$20f$	$16\times$	$12+$
Adjoint Construction (A.17)	$12f$	$27\times$	$18+$
Adjoint transformation (A.18)	$42f$	$36\times$	$30+$

5.4.2 Cost of HTM Algorithm

In this Subsection we shall use the costs from Tables 5.5 and 5.6 together with the methodology in 5.1 to calculate the whole cost of the HTM based algorithm described in (A.20) to (A.25) (Appendix A).

Thus, we start by calculating the costs for (A.20). This is the cost for a series of n HTM multiplications, that is,

$$\text{cost}_{\text{Total}}(\mathbf{T}_{EF}) = n(16f, 64\times, 48+). \quad (5.46)$$

We also note here that, as was the case for the dual quaternions, this equation is parametrized in the PoE formulation. However, if we choose to use the DH notation the cost for each transformation will triplicate, as there will also be three extra transformations. Thus

$$\text{cost}_{\text{Total}}(\mathbf{T}_{EF}^{DH}) = 3n(16f, 64\times, 48+). \quad (5.47)$$

Next we calculate the costs for (A.21),

$$\begin{aligned} \text{cost}_{\text{Total}}(\mathbf{W}_i) &= \text{cost}_{\text{Total}}\left(\left[\text{Ad}_{\mathbf{T}_i^{i-1}}\right] \mathbf{W}_{i-1} + \mathbf{A}_i \dot{\theta}_i\right) \\ &= \underbrace{\text{cost}_{\text{Total}}\left(\left[\text{Ad}_{\mathbf{T}_i^{i-1}}\right] \mathbf{W}_{i-1}\right)}_{\text{Cost of Adjoint Transform}} + \text{cost}_{\text{Total}}(\text{Vector Add.}) + \underbrace{\text{cost}_{\text{Total}}(\mathbf{A}_i \dot{\theta}_i)}_{\text{Cost for vec. scalar mult.}} \\ &= (42f, 36\times, 30+) + (0f, 0\times, 6+) + (7f, 6\times, 0+) \\ &= (49f, 42\times, 36+), \end{aligned} \quad (5.48)$$

and the costs for (A.22)

$$\begin{aligned}
\text{cost}_{\text{Total}}(\dot{\mathbf{W}}_i) &= \text{cost}_{\text{Total}}\left(\left[\text{Ad}_{\mathbf{T}_i^{i-1}}\right]\dot{\mathbf{W}}_{i-1} + \text{ad}_{\mathbf{W}_i}(\mathbf{A}_i) + \mathbf{A}_i\ddot{\theta}_i\right) \\
&= \underbrace{\text{cost}_{\text{Total}}\left(\left[\text{Ad}_{\mathbf{T}_i^{i-1}}\right]\dot{\mathbf{W}}_{i-1}\right)}_{\text{Cost of Adjoint Transform}} + \underbrace{\text{cost}_{\text{Total}}\left(\text{ad}_{\mathbf{W}_i}(\mathbf{A}_i)\dot{\theta}_i\right)}_{\substack{\text{Cost of vector Adjoint} \\ \text{Cost of vec. by scalar}}} + \underbrace{\text{cost}_{\text{Total}}\left(\mathbf{A}_i\ddot{\theta}_i\right)}_{\text{Cost for vec. scalar mult.}} + 2\text{cost}_{\text{Total}}(\text{Vec. Add.}) \\
&= (42f, 36\times, 30+) + [(42f, 36\times, 30+) + (1f, 6\times, 0+)] + (1f, 6\times, 0+) + 2(0f, 0\times, 6+) \\
&= (86f, 84\times, 72+). \tag{5.49}
\end{aligned}$$

We note that both the costs for (5.48) and (5.49) are repeated n -times, thus resulting in a total of cost of $n(49f, 42\times, 36+)$ and $n(86f, 84\times, 72+)$ respectively. Furthermore, we also highlight here that we are considering that the adjoint matrix has already been constructed and is just being retrieved from memory. In fact this is the approach we will be following for both the matrix and vector adjoint throughout the remainder of this section.

Now we move to the backward iteration, and before calculating the costs for (A.23), we must account for the costs for the inertia matrix, that is

$$\text{cost}_{\text{Total}}(\mathcal{G}) = \text{cost}_{\text{Total}}\left[\begin{array}{cc} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & m\mathbf{I}_3 \end{array}\right] = (10f, 9\times, 0+). \tag{5.50}$$

We remark that (5.50) should only be calculated once at the beginning of the algorithm, and retrieved from memory in the subsequent iterations of the algorithm. After we construct \mathcal{G} we can calculate the costs for the resulting forces

$$\begin{aligned}
\text{cost}_{\text{Total}}(\mathbf{F}_{R,i}) &= \text{cost}_{\text{Total}}\left(\mathbf{G}_i\dot{\mathbf{W}}_i - \text{ad}_{\mathbf{W}_i}^T(\mathbf{G}_i\mathbf{W}_i)\right) \\
&= \underbrace{\text{cost}_{\text{Total}}\left(\mathbf{G}_i\dot{\mathbf{W}}_i\right)}_{\text{Cost of Multiplication}} + \text{cost}_{\text{Total}}(\text{Vec. Sub.}) + \underbrace{\text{cost}_{\text{Total}}\left(\text{ad}_{\mathbf{W}_i}^T(\mathbf{G}_i\mathbf{W}_i)\right)}_{\substack{\mathbf{G}_i\mathbf{W}_i \text{ multiplication} \\ \text{Cost of Vector Adjoint}}} \\
&= (42f, 36\times, 30+) + (0f, 0\times, 6+) + [(6f, 36\times, 30+) + (36f, 36\times, 30+)] \\
&= (84f, 108\times, 96+), \tag{5.51}
\end{aligned}$$

in which we must remember that for n -iterations of the algorithm the cost would be $n(84f, 108\times, 96+)$. Following that we calculate the costs for (A.24)

$$\begin{aligned}
\text{cost}_{\text{Total}}(\mathbf{F}_i) &= \text{cost}_{\text{Total}}\left(\left[\text{Ad}_{\mathbf{T}_i^{i+1}}\right]^T\mathbf{F}_{i+1} + \mathbf{F}_{R,i}\right) \\
&= \underbrace{\text{cost}_{\text{Total}}\left(\left[\text{Ad}_{\mathbf{T}_i^{i+1}}\right]^T\mathbf{F}_{i+1}\right)}_{\text{Cost of Adjoint Transform}} + \text{cost}_{\text{Total}}(\text{Vec. Addition}) + \text{cost}_{\text{Total}}(\mathbf{F}_{R,i}) \\
&= (42f, 36\times, 30+) + (0f, 0\times, 6+) + (6f, 0\times, 0+) \\
&= (48f, 36\times, 36+), \tag{5.52}
\end{aligned}$$

with the costs for n -iterations being $n(48f, 36\times, 36+)$.

Finally, we have the costs for the torque in (A.25),

$$\begin{aligned}
\text{cost}_{\text{Total}}(\tau_i) &= \text{cost}_{\text{Total}}\left(\mathbf{F}_i^T\mathbf{A}_i\right) \\
&= \text{cost}_{\text{Total}}(\text{Vector Multiplication}) \\
&= (12f, 6\times, 5+), \tag{5.53}
\end{aligned}$$

Table 5.7: Cost of operations for HTM

Oper.	HTM - PoE			HTM - DH		
	Storage(f)	\times/\div	$+/-$	Storage(f)	\times/\div	$+/-$
Eq. (3.42)	10	9	0	10	9	0
Eq. (A.20)	$16n$	$64n$	$48n$	$48n$	$192n$	$144n$
Eq. (A.17)	$12n$	$27n$	$18n$	$12n$	$27n$	$18n$
Eq. (A.8)	$12n$	0	0	$12n$	0	0
Eq. (A.21)	$49n$	$42n$	$36n$	$49n$	$42n$	$36n$
Eq. (A.22)	$86n$	$84n$	$72n$	$86n$	$84n$	$72n$
Eq. (A.23)	$84n$	$108n$	$96n$	$84n$	$108n$	$96n$
Eq. (A.24)	$48n$	$36n$	$36n$	$48n$	$36n$	$36n$
Eq. (A.25)	$12n$	$6n$	$5n$	$12n$	$6n$	$5n$
TOTAL	$319n + 10$	$367n + 9$	$311n$	$351n + 10$	$495n + 9$	$407n$

with a total cost for the whole algorithm being $n(12f, 6\times, 5+)$.

In Table 5.7 we show, in the first column the total cost of operations for the HTM based algorithm from in [49]. In the second column we show the added cost if we had used a DH notation for the forward kinematics. As was the case for the dqRNEA we highlight that the actual costs for an algorithm of this kind using the DH parameters would be even greater, as we would need to change the position of the center of mass. Furthermore, a full comparison of the algorithms is presented in Subsection 5.5.2.

5.5 COMPUTATIONAL COST COMPARISON

In this Section we elaborate more on the discussion presented in the end of Section 5.3, in which we compared the computational costs of different formulations of the dqRNEA algorithm. Furthermore, given the costs for an HTM based recursive Newton Euler algorithm found in the literature (see Appendix 5.4) we shall also discuss some of the differences in performance for this cases.

5.5.1 Results in the Literature

Before looking at our algorithm, we believe it is interesting to account for some of the discussions found in the literature regarding this subject. Indeed, ever since the 1990's people have been making comparisons between the computational costs of quaternions, dual quaternions and other algebras.

In [41] four different formalisms are compared regarding their efficiency for representing a rigid body's screw displacement, the dual orthogonal 3×3 matrix, unit dual quaternions, dual special unitary 2×2 matrix, and dual Pauli spin matrices. The paper argues that the most advantageous formalism is has to be compact and computationally efficient—as both [41, 110] address, the costs for fetching an operand may sometimes exceed the costs for performing basic arithmetic operations, thus making the storage an important point for us

to consider in our analysis. Furthermore, [41] concludes their analysis by stating that for sequential operation the unit dual quaternions are indeed the most compact and computationally efficient manner to represent a screw displacement of a line in space.

Following the results of [41], quite a few works have presented a computation efficiency calculation for the forwards and inverse kinematics problem of a serial manipulator, it is shown in [37, 40, 77, 110] that, for the kinematics, a DQ based implementation performs better than an HTM in terms of computational cost. In particular [40] states that the use of the D-H parameters also make costs increase the cost of the forward dynamics by three, as shown in (5.39).

In [110] the authors make an analyzes that concludes that the inverse kinematics is also more efficient in dual quaternion algebra than when using quaternions or exponential maps. However [37] concludes that when calculating the Jacobian matrix, fewer multiplications are necessary for a geometric Jacobian than the dual quaternion Jacobian (as the former is of the size $6 \times n$ and the latter is $8 \times n$).

Finally, we must also highlight here most of our design choices for the dqRNEA algorithm were made taking into consideration the results in the literature which were more efficient. However, sometimes a greater computational cost may be well justified. As [37] argues, the dual quaternion may perform poorly in some accounts, but it does have some attractive proprieties that can justify its use: the DQs are more compact on themselves, they are singularity free, we may use the Hamiltonian operators in order to commute elements and we may use the same set of variables for the kinematics, dynamics and control. And those are only to cite a few of those advantages.

In this manner, we do aim to find a more compact and cost efficient solution for the dynamics, however, the representation of the dual quaternion based recursive Newton Euler algorithm can be justified for its novelty alone.

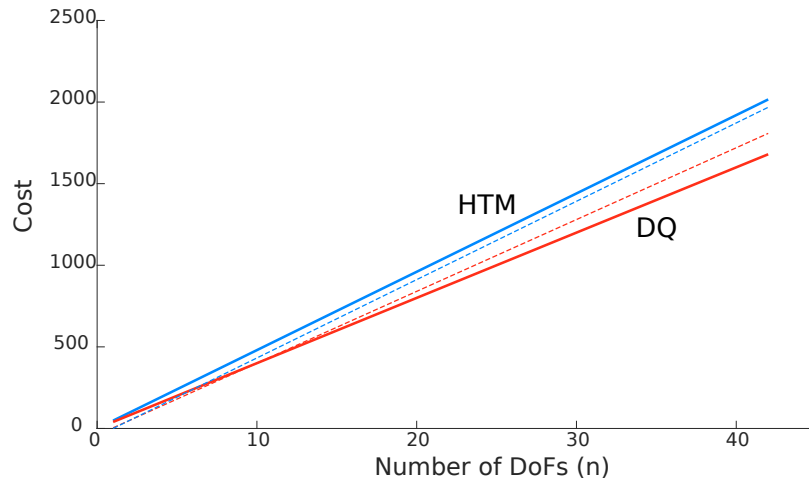
5.5.2 Algorithms Comparison

In this subsection we compare the different costs for both the HTM based algorithm, based on both Table 5.4 and Table 5.7 (See Appendix A).

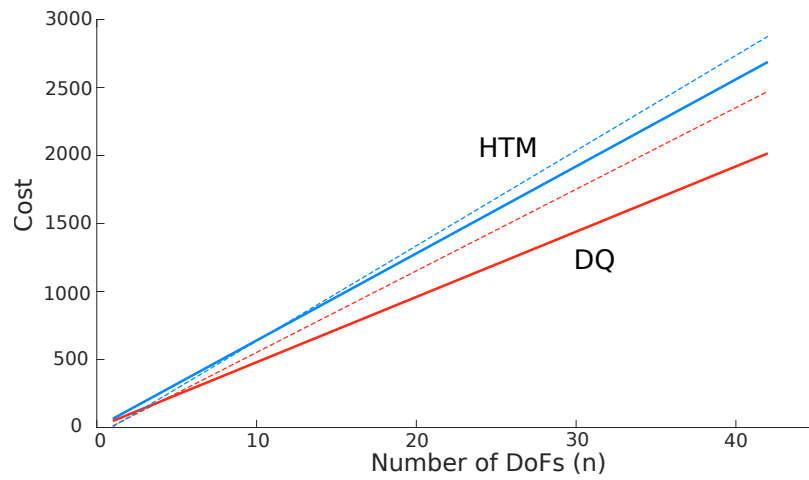
To start our analysis of the costs for the algorithm we believe it is interesting to look back some the discussion in Subsection 5.5.1. In particular, we shall look Figure 5.1 which shows how the costs for forward kinematics transformation from 3.55 grow. Indeed, it is shown in [37], [40] that the dual quaternions perform better than the HTM for the FKM transformation, and that the PoE formulation has an advantage over the DH parameters. As it can be seen, for a manipulator with few degrees of freedom the difference might be small, but it becomes relevant as n —the number of DoFs—increases (and more so with the algorithm running many times in order to control a robot).

Furthermore, in Figure 5.2 we show the cost comparison between the dqRNEA with PoE and the new adjoint in comparison to the HTM recursive Newton-Euler presented in Appendix A. Indeed, one of our goals throughout this work was to create a version of the RNE algorithm with dual quaternions and evaluate whether it is cost efficient when compared to a similar representation, and, in this manner, Figure 5.2 provides us this answer. We notice that for all three variables we have measured the dual quaternion representation performs significantly better.

Still regarding the results of Figure 5.2, we must add that there are other versions of this algorithm through-



(a) Costs for Addition



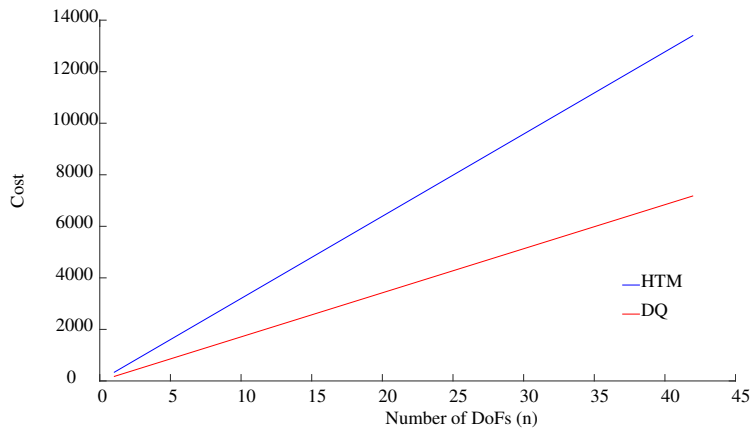
(b) Costs for Multiplication

Figure 5.1: Costs for Forward Kinematics computation using different representations. In blue we have HTM and in red we have DQ representation. With the dotted line is the cost for D-H based parametrization and the straight line is the screw theory parametrization. The value of n ranges from 1-42.

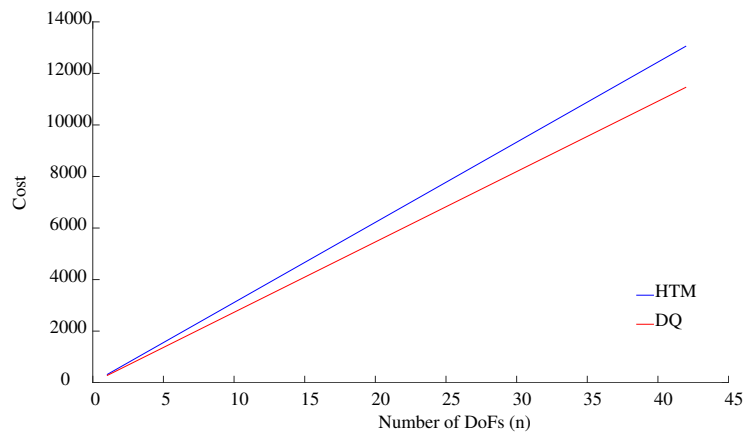
out the literature and some of them highly optimized for specific applications, thus presenting lower costs than the ones in this manuscript. More so, the manner in which different authors calculate the costs may also differ, making it difficult to make a comparison between results such as in [55, 111], where the costs for some RNE algorithms are lower.

In this manner, we have chosen to compare our results to the classic version of Homogeneous Transformation Matrix based RNE presented in Appendix A, and to calculate the costs of that algorithm by using our methodology. One of the main reasons for our choice of algorithm was that HTM are comparable to dual quaternions in being both singularity free and for having the translation and rotation coupled together in only one structure, which by itself is one great advantage for applications in path planning and control. Another reason for choosing the version in Appendix A of the algorithm was that it made some similar design choices to our own and did not use any novel proprieties that would complicate the cost calculation.

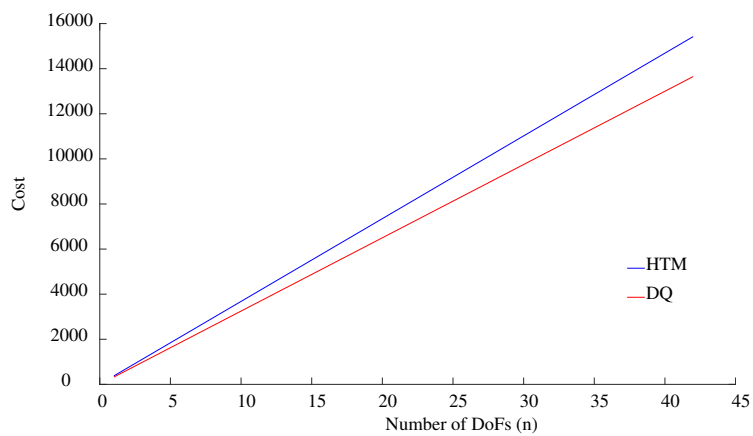
Finally, before finishing this section we also present, in Figure 5.3 the cost comparison for all the versions of the algorithm. Which depending on what we are evaluating they perform differently. We consider the best version of our algorithm to be the one with the New Adjoint and the PoE (in Figure 5.3 represented by the red line) and the one that performs more poorly to be the DH based version with the old adjoint (in Figure 5.3 represented by the black line).



(a) Storage Costs

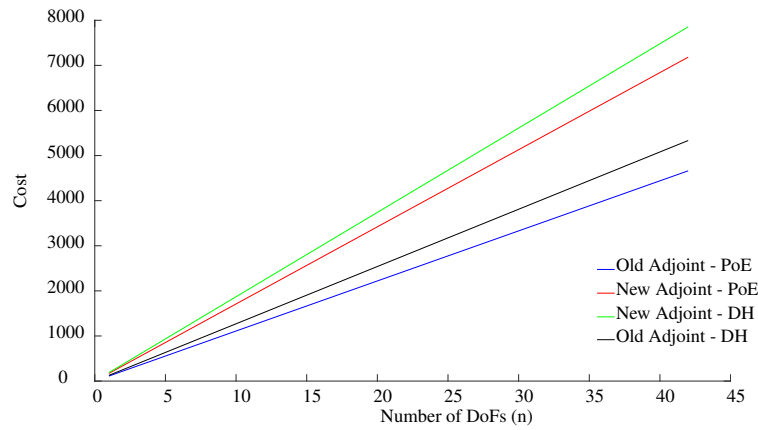


(b) Addition Costs

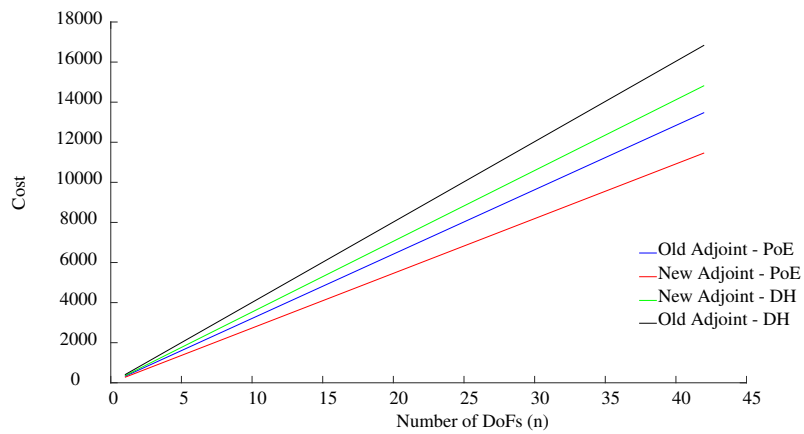


(c) Multiplication Costs

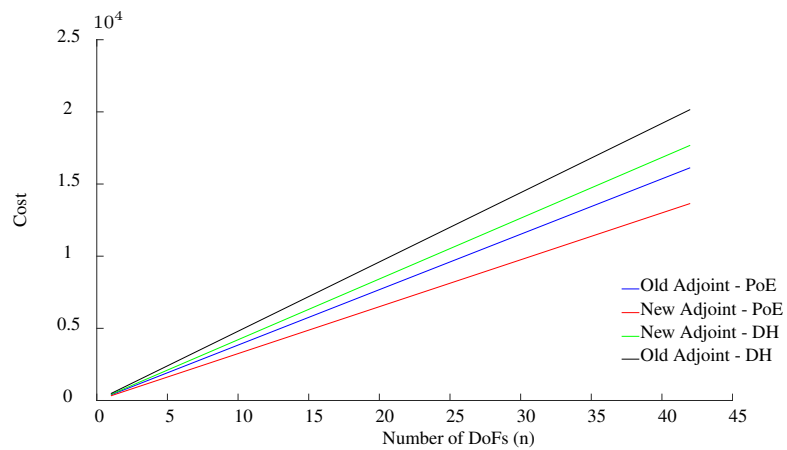
Figure 5.2: Total costs for RNEA of both HTM and DQ. In the plots, the HTM is represented in blue and the DQ is represented in Red. Also n ranges from 1-42.



(a) Storage Costs



(b) Addition Costs



(c) Multiplication Costs

Figure 5.3: Total costs the different versions of the algorithm presented in Table 5.4. with n ranging from 1-42.

6

SIMULATION RESULTS

Simulation is an incredibly important tool for validating and testing algorithms. Thus in this Chapter we show some of the tests and results we achieved when using the algorithms and equations proposed in this manuscript.

To start off this Chapter, in Section 6.1 we first discuss the methodology we used for algorithm development and for analyzing our results. Furthermore, in Section 6.2 we provide a very didactic example of how to use the algorithms to program and control a simple Two Link Planar arm. In Section 6.3 we move on to a more complex example: that of the 7-DoF Kuka robot.

6.1 METHODOLOGY

In order to implement and validate our results we have used mainly the MATLAB software and the DQ_Robotics Toolbox¹. Our main method of analysis was comparing the results of our package with the results for similar algorithms in another package (in this case the robotics toolbox). To further analyze the results we have implemented a computed torque controller in different situations and analyzed its convergence.

Taking advantage of these tools our methodology can be divided into three main parts which will be further explored in the next subsections:

1. Implementation of a MATLAB package for the functions described in Chapter 4. Here we have used native MATLAB routines coupled together with the DQ_Robotic toolbox which allowed us to promptly use quaternion and dual quaternion algebra. This is presented in Subsection 6.1.1;
2. The Peter Corke robotics toolbox has many different functions available, including the recursive Newton-Euler algorithm for dynamics. As this toolboxes are widely used, we believe they are an important tool for validation. Thus, in Subsection 6.1.2 we describe in a bit more depth.
3. Computational Setup for the experiments.

6.1.1 MATLAB Package

In order to validate our algorithm we have created a MATLAB package with the functions of Chapter 4. Here we added the dual quaternion functionality by means of DQ_Robotics Toolbox. Table 6.1 shows the main functions implemented and their description.

6.1.2 Validation Packages

For validation purposes we have used a well know robotic toolbox: Peter Corke Robotic toolbox [112]. As both this toolbox is widely used and have functions for the recursive Newton-Euler Algorithm we have set it up for benchmarking the results we have obtained in our algorithms.

¹The DQ Robotics Toolbox can be found at <https://dqrobotics.github.io/>

Table 6.1: Function description for dqRNEA MatLab Package. Variables from Subsection 4.3.2

Function name		Description	Input
Main dqRNEA functions	dqRNE	The dual quaternion recursive Newton Euler Function as described in Algorithm ??.	Robot Model; Joint trajectories; Joint Velocity; Joint Accelerations
	matrix_invdyn	The matrix-quaternion version of the dqRNEA algorithm: that is the equations in Table 4.1	Robot Model; Joint trajectories; Joint Velocity; Joint Accelerations
	closed_invdyn	Closed version of the algorithm. As in (4.71).	Robot Model; Joint trajectories; Joint Velocity; Joint Accelerations
Closed form sub-functions	M_function	Mass function, as in (4.72). Returns the inertial terms of the inverse dynamic equation.	$\underline{A}, \underline{L}, \underline{L}^*$, robot model, Number of Degrees of Freedom.
	C_function	Centripetal and Coriolis function, as in (4.73). Returns the Centripetal and Coriolis terms of the inverse dynamic equation.	$\underline{A}, \underline{L}, \underline{L}^*, \underline{C}, \underline{W}, \underline{W}_{BASE}, \dot{\theta}$, robot model, Number of Degrees of Freedom.
	G_function	Gravity function as in (4.74). Returns the gravity terms of the inverse dynamics equation.	$\underline{A}, \underline{L}, \underline{L}^*, \underline{W}, \dot{\underline{W}}_{BASE}$, robot model, Number of Degrees of Freedom.
	f_jacob	Contact Jacobian as in (4.75). Used when there is a force at the end-effector.	$\underline{A}, \underline{L}^*$, Number of Degrees of Freedom
Auxiliary Functions	DG_product	Double Geodesic Product as described in (2.42).	Two pure dual quaternions
	cross_Vec	Element wise cross product between two dual quaternion vectors. As in (2.71).	Two dual quaternion vectors
	G_	Dual Quaternion Inertia Transformation as in (3.44).	Dual Quaternion, robot model, iteration
	G_vec	Vectorial version of the Dual Quaternion Inertia transformation as in (4.51).	Dual Quaternion vector, robot model
	G_Mat	Matrix version of the Dual Quaternion Inertia Transformation.	Dual Quaternion Matrix, robot model
	vec2DQ	Real Vector to quaternion vector transformation.	Vector
	Mat2DQMat	Real Matrix to dual quaternion matrix transformation.	Matrix
	DQ2vec	Dual quaternion vector to real vector transformation.	Dual Quaternion Vector
DQMat2Mat	Dual quaternion matrix to real matrix transformation.	Dual Quaternion Matrix	

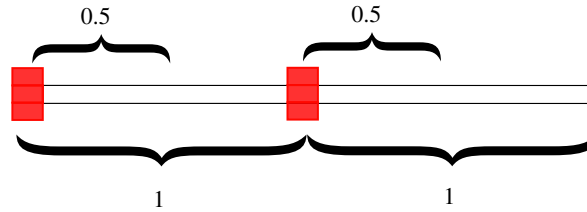


Figure 6.1: Schematic Upper View of a two link planar arm in which both links measure 1m and the center of mass is located exactly at the middle point of the link.

Regarding Peter Corke’s toolbox, we have the RNE algorithm developed based on the equations presented in [56]. These are written as decoupled vector-based equations and use the Denavit-Hartenberg notation.

6.1.3 Computational Setup

For all the experiments detailed in this chapter we have adopted as hardware setup a Dell Inspiron with 16GB of RAM running on an Intel Core i7-7500U CPU at 2.70GHz with 4 cores. Furthermore, all algorithms were tested for MATLAB R2017b, running on a Ubuntu 16.04 64-bit version.

Some of the experiments also use MATLAB-Simulink in order to control the robot, in this case we have set the simulation to use a variable-step with maximum step size of 0.05s and an ODE23t (mod. stiff/Trapezoidal) solver.

6.2 TWO LINK PLANAR ARM: MODELING AND CONTROLLING A SIMPLE ROBOT

In this section we shall present a simple example of how to model a robot and implement it in the MATLAB framework we have designed. For that we shall use a simple two-link planar arm. In Subsection 6.2.1 we shall present the model of our robot, in Subsection 6.2.2 we show the numerical results and in Subsection 6.2.3 we show the implementation of the computed torque control as in 4.4.1

6.2.1 Robot Model

Here we present the model for the two link planar arm we used for this experiment. We note that the model plotted in Figure 6.2 was taken from the Peter Corke’s robotic toolbox—and as such, uses the DH notation to describe its kinematics. In this case the transformation from DH parameters to the PoE formulation is quite straightforward, as all joint axis are parallel to each other, that is, all can be defined in the same axis of the referenced frame. Figure 6.1 shows the layout of the robot, in which each link is measured one meter and the center of mass is located at 0.5m from each joint axis

Following Figure 6.1 we can program our screw-based model, which in this case shall follow Listing 6.1. We highlight here that our model is structured in two parts: the first one being the transformation from the beginning of the link (where the joint is) to the object’s center of mass. This is followed by the transformation

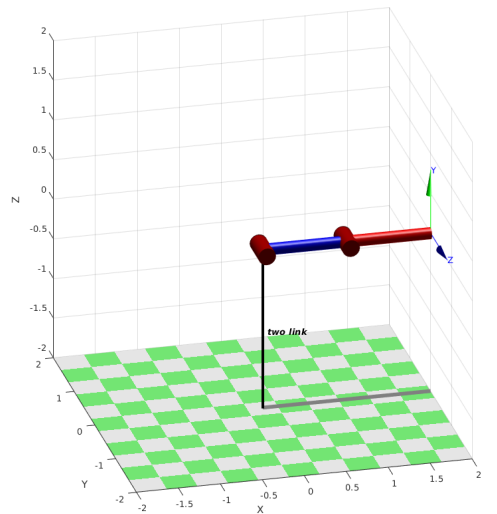


Figure 6.2: Matlab 3D Robot Plot for the two-link planar arm.

from one link to the other.

Listing 6.1: Two Link Arm PoE model for dqRNEA algorithm

```
function [linktwist,MT,NT] = twolink_dq_screw()

% Screw axis on each joint is the same
linktwist = DQ([0 0 0 1]);

%%
% Link transformation: Must remember to divide by two for dual
% quaternion motion.
%%

% Transformation from each link to its center of mass
MT(1) = DQ([1 0 0 0, 0 0.5/2 0 0]);
MT(2) = DQ([1 0 0 0, 0 0.5/2 0 0]);

% Transformation from each link to the other
NT(1) = DQ([1 0 0 0, 0 1/2 0 0]);
NT(2) = DQ([1 0 0 0, 0 1/2 0 0]);

end
```

Once the model is defined, as in Listing 6.1 we can easily use them to calculate the forward kinematics. As shown in (3.60) we need both the axis of rotation \underline{a}_i which is defined as the variable “linktwist” and the

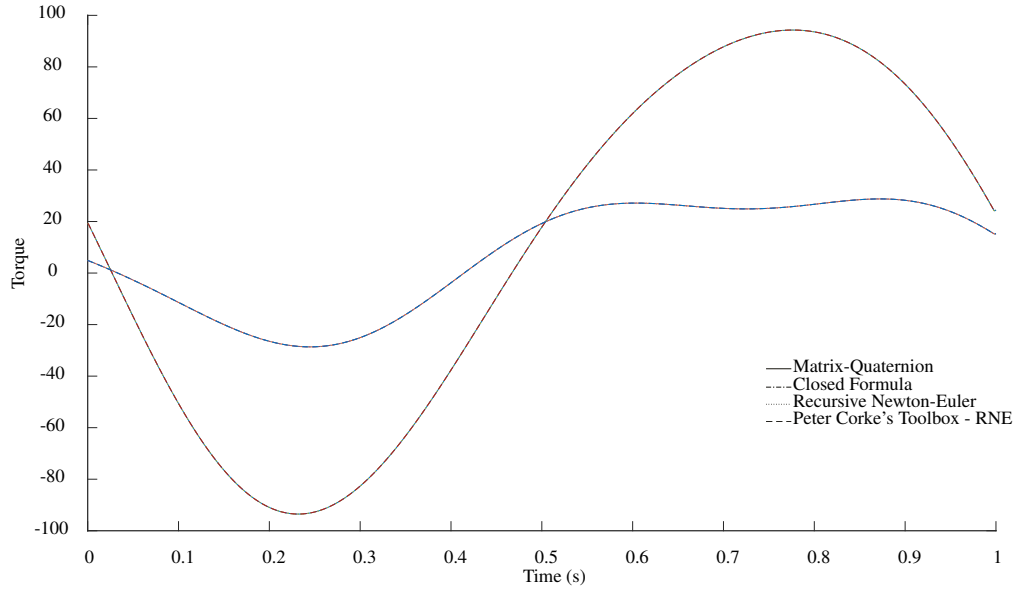


Figure 6.3: Torque plot for each of the joints in different versions of the algorithm. The straight lines are the results for the matrix-quaternion formulation, the dash-dotted lines refers to the closed equation and the dotted line are the torques for the recursive algorithm. Finally the dashed line are the resulting torques for the recursive Newton-Euler algorithm from Peter Corke’s Toolbox.

home position link-transformations δ_{i-1}^i , which are obtained from the variables “MT” and “NT”, such that the resulting transformation can be calculated as

$$\underline{x}_i^{i-1}(\theta_i(t)) = MT_i^* (NT_i (\exp(-0.5\underline{a}_i\theta_i))) MT_{i+1}. \quad (6.1)$$

As described in Listing 6.1 NT is the transformation from one joint to the next, and MT is the transformation from the joint to the center of mass. To make the models more intuitive we combine the two expressions such that we have (6.1).

6.2.2 Numerical Results

In order to validate our algorithms we have ran all of the three main dqRNEA functions from Table 6.1, and compared it to the results for the recursive Newton-Euler function provided by Peter Corke’s toolbox.

As input parameters for the inverse dynamics functions we have input a random sinusoidal trajectory (and their derivatives) and the model for the robot followed what was described in subsection 6.2.1 (for Peter Corke’s toolbox we have used the equivalent DH Parameters). The plot in Figure 6.3 shows the resulting torques for each of the algorithms.

We note that for this model all algorithms result in the exact same value for the torques, as expected since different modeling or computation strategies should not provide different values for the mechanical system. This exercise is presented herein purely to endorse such analysis.

6.2.3 Control Algorithm

Here we present the results for the computed torque controller implemented in the two link planar arm. In order to implement this controller and simulate its results we need also to implement the forward dynamic equations. To this aim, we have taken advantage of a MATLAB-SimuLink model provided by Peter Corke's Robotic toolbox [112]. The Simulink model was adapted to compute the torque-based control inputs following the proposed inverse dynamics algorithms, dqRNEA. Figure 6.4 illustrates the control strategy on Simulink.

Furthermore, we have set our control input to a few different trajectories. In Figures 6.5 and 6.6 we show the progression of the position error for some of the tests performed.

We notice here that from all the setups the controller is converging to the desired pose. Depending on the configuration, however, we can see there is a small stationary error—the biggest one being shown in Figure 6.6b.

6.3 KUKA ROBOT: THE DQRNEA BASED CONTROLLER

In order to better validate our models we have also implemented our algorithms in a more realistic robot: the KUKA [2]. In this manner, we start this section, in Subsection 6.3.1, by presenting the robot model for the KUKA robot—in particular, we detail here how to interpret the DH parameters in order to describe the robot in our PoE formulation. Furthermore, we present the numerical results for the dqRNEA algorithm in Subsection 6.3.2 and the controller implemented in MATLAB-Simulink in Subsection 6.3.3.

6.3.1 Robot Model

In this section we will present the robot model of the KUKA LWR4, which can be seen in Figure 6.7. The correspondent DH parameters can be seen in Table 6.2 and the position for the center of mass used in the PoE formulation is seen in Table 6.3.

In order to transform the parameters on Table 6.2 to the ones in Table 6.3 we simply need to look at the schematic for the robot, presented in Figure 6.7. We first notice that the DH frames are rotated to coincide with each rotation axis, thus the centers of mass given in Table 6.3 are with relation to those rotated frames. In our model, however, we start from the notion that all frames are aligned at home position and the center of mass is given by taking the joints as references. The transformation from the DH to PoE is then given as a rotation of each link's axis so that they are all aligned.

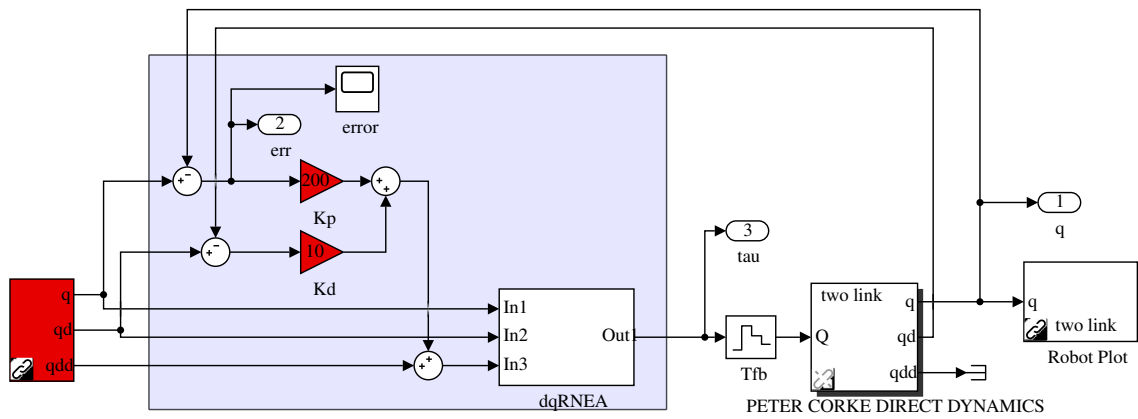
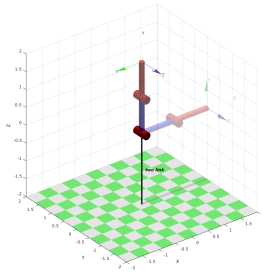
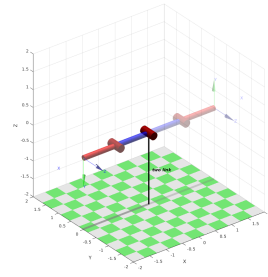


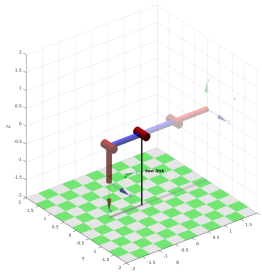
Figure 6.4: Controller Design on Simulink—From the Peter Corke Toolbox, with the addition of our dual quaternion based recursive newton euler function.



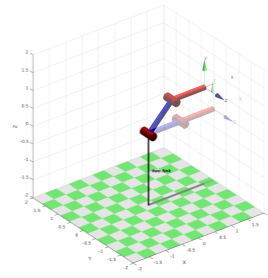
(a) Robot initial and final position: $\theta_0 = \{0, 0\}$, $\theta_{end} = \{0, \frac{\pi}{2}\}$.



(b) Robot initial and final position: $\theta_0 = \{0, 0\}$, $\theta_{end} = \{0, \pi\}$

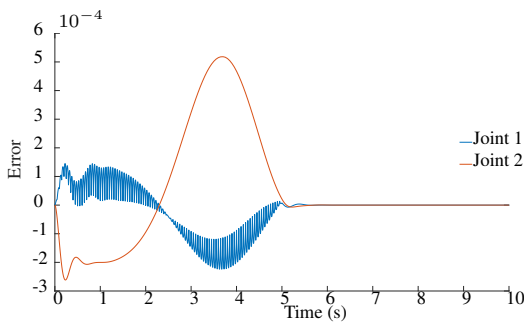


(c) Robot initial and final position: $\theta_0 = \{0, 0\}$, $\theta_{end} = \{\pi, \frac{\pi}{2}\}$

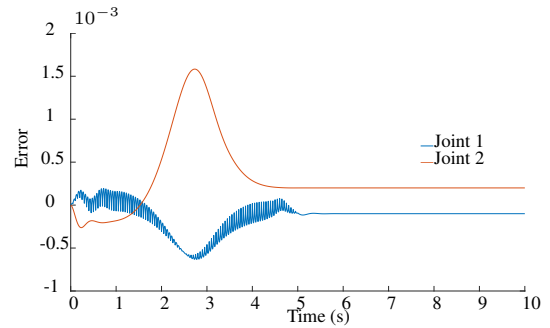


(d) Robot initial and final position: $\theta_0 = \{0, 0\}$, $\theta_{end} = \{\pi, \frac{\pi}{2}\}$

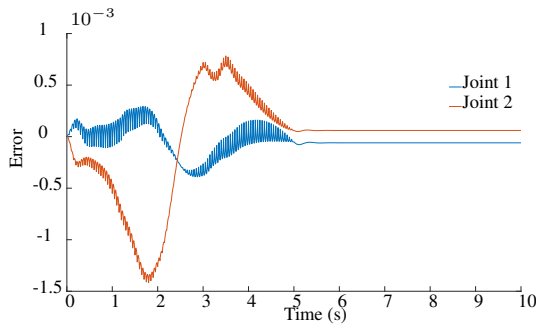
Figure 6.5: Robot plot at different configurations. Here we merged the images for the plot at the initial joint configuration θ_0 and the final joint configuration θ_{end} .



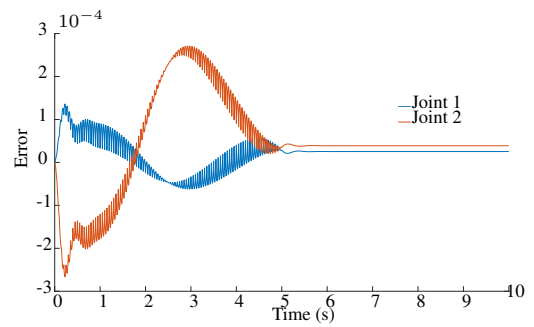
(a) Error progression for achieving the final position in Figure 6.5a



(b) Error progression for achieving the final position in Figure 6.5b



(c) Error progression for achieving the final position in Figure 6.5c



(d) Error progression for achieving the final position in Figure 6.5d

Figure 6.6: Plot for the Error Progression for the experiments in Figure 6.5

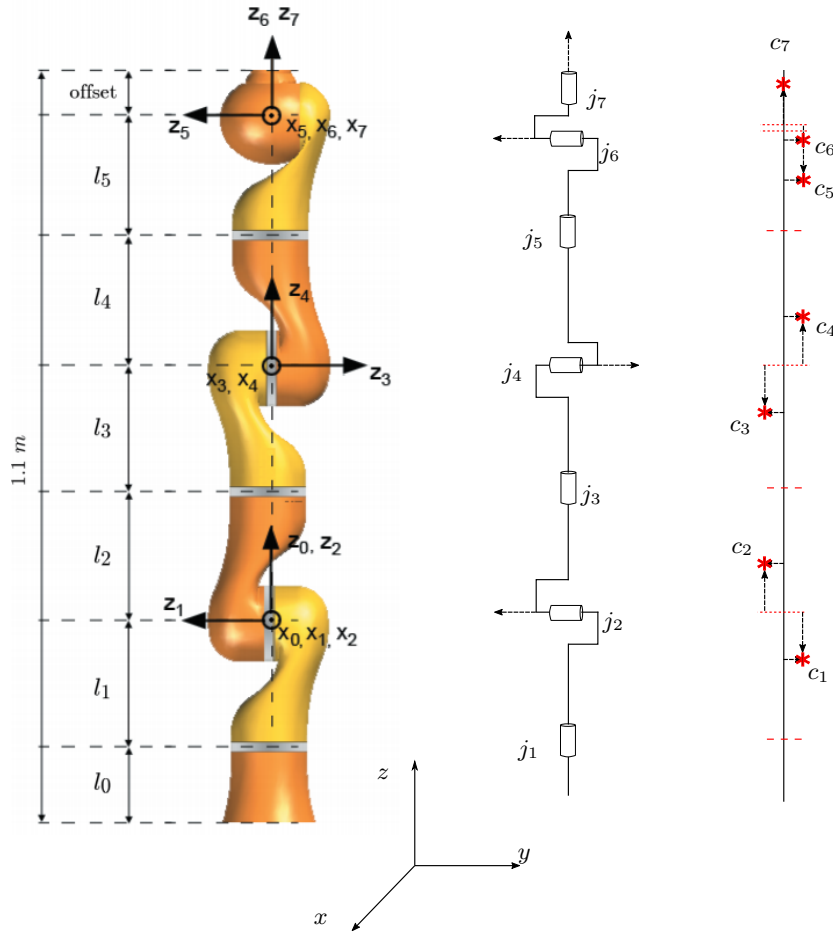


Figure 6.7: Model of the robot, from left to right we have the DH model of the KUKA LWR4, taken from [2], followed by the joint position and orientation and finally the center of mass for each link—with positions c_1 to c_7 shown in Table

Table 6.2: DH parameters for the KUKA robot, based on [2]

(a) DH Parameters (b) Center of mass for each of the links. The frame of reference is the DH frame.

DH-Parameters				Center of mass			
d	a	α	θ		c_x	c_y	c_z
0.2	0	$\frac{\pi}{2}$	θ_1	c_1	0.001340	-0.087777	-0.026220
0	0	$-\frac{\pi}{2}$	θ_2	c_2	0.001340	-0.026220	0.087777
0.4	0	$-\frac{\pi}{2}$	θ_3	c_3	-0.001340	0.087777	-0.026220
0	0	$\frac{\pi}{2}$	θ_4	c_4	-0.001340	0.026220	0.087777
0.39	0	$\frac{\pi}{2}$	θ_5	c_5	-0.000993	-0.111650	-0.026958
0	0	$-\frac{\pi}{2}$	θ_6	c_6	-0.000259	-0.005956	-0.005328
0	0	0	θ_7	c_7	0	0	0.063

Listing 6.2: Two Link Arm PoE model for dqRNEA algorithm

```

function [linktwist, NT, MT] = kuka_dq_screw()
% Screw from each joint
linktwist(1) = DQ([0 0 0 1 0 0 0 0]);
linktwist(2) = DQ([0 0 -1 0 0 0 0 0]);
linktwist(3) = DQ([0 0 0 1 0 0 0 0]);
linktwist(4) = DQ([0 0 1 0 0 0 0 0]);
linktwist(5) = DQ([0 0 0 1 0 0 0 0]);
linktwist(6) = DQ([0 0 -1 0 0 0 0 0]);
linktwist(7) = DQ([0 0 0 1 0 0 0 0]);

% Transformation from each link to the other

NT(1) = DQ([1 0 0 0, 0 0 0 0.2/2]);
NT(2) = DQ([1 0 0 0, 0 0 0 0.2/2]);
NT(3) = DQ([1 0 0 0, 0 0 0 0.2/2]);
NT(4) = DQ([1 0 0 0, 0 0 0 0.2/2]);
NT(5) = DQ([1 0 0 0, 0 0 0 0.19/2]);
NT(6) = DQ([1 0 0 0, 0 0 0 0]);
NT(7) = DQ([1 0 0 0, 0 0 0 0.078/2]);

% Transformation from each link to its center of mass
a = 0.087777;
b = 0.02622;
c = 0.001340;

MT(1) = DQ([1 0 0 0, 0 c/2 b/2 a/2
]);
MT(2) = DQ([1 0 0 0, 0 c/2 -b/2 (0.2-a)/2
]);
MT(3) = DQ([1 0 0 0, 0 -c/2 -b/2 a/2
]);
MT(4) = DQ([1 0 0 0, 0 -c/2 b/2 (0.2-a)/2
]);
MT(5) = DQ([1 0 0 0, 0 -0.0009930/2 0.026958/2 0.11165/2
]);
MT(6) = DQ([1 0 0 0, 0 -0.000259/2 -0.005328/2 0.005956/2
]);
MT(7) = DQ([1 0 0 0, 0 0 0 (0.078-0.063)/2
]);

end

```

Table 6.3: PoE formulation for the KUKA robot–center of mass position with regards to equivalent joint, as in Figure 6.7

Center of mass			
	c_x	c_y	c_z
c_1	0.001340	0.02622	0.087777
c_2	0.001340	-0.02622	0.1122
c_3	-0.001340	-0.02622	0.087777
c_4	-0.001340	0.02622	0.1122
c_5	-0.000930	-0.026958	0.111650
c_6	-0.000259	-0.005328	0.005956
c_7	0	0	0.0150

Finally, following what was done in Subsection 6.2.1 we could program the KUKA model in the function presented in Listing 6.2.

6.3.2 Numerical Results

In this Subsection we present the numerical results for our algorithm with the three functions in the first section of Table 6.1, as well as the comparison with the RNE toolbox. Here, like in Subsection 6.2.2, we input to the functions our robot model as well as random trajectories for the joints (and their derivatives). Figure 6.8 shows the numerical results for the different implementations of our algorithm as well as the result from Peter Corke’s Toolbox.

We notice that for our quaternion based inverse dynamics the results are exactly the same in any of the forms of our algorithm. However, when we compare to Peter Corke’s RNE there is a difference. In particular the algorithm we have created is very dependent on the model for the robot, and any uncertainties may cause deviations of the output. The effects of this uncertainties, as in any system, may generate negligible error, but also it may generate more considerable deviations in the results. In this particular case, there was some uncertainty in the inertia parameters that we used, therefore, although we have already validated the algorithm there might be some slight differences in the resulting torque due to some difference in the robot model.

6.3.3 MATLAB Controller

Here we have implemented the computed torque control, once more, by taking advantage of the Simulink blocks from Peter Corke’s robotic toolbox. The model we present in Figure 6.9 is indeed identical to the one used on Subsection 6.2.3, with the only changes being the input parameters, and robot model.

Again, as in Subsection 6.2.3 we have set our control target to different positions and analyzed our results in Figures 6.10 and 6.11.

We notice the results are also satisfactory in this case, with all the robot joints achieving a low steady state error in the end of the experiments, as was the case for the two link planar arm.

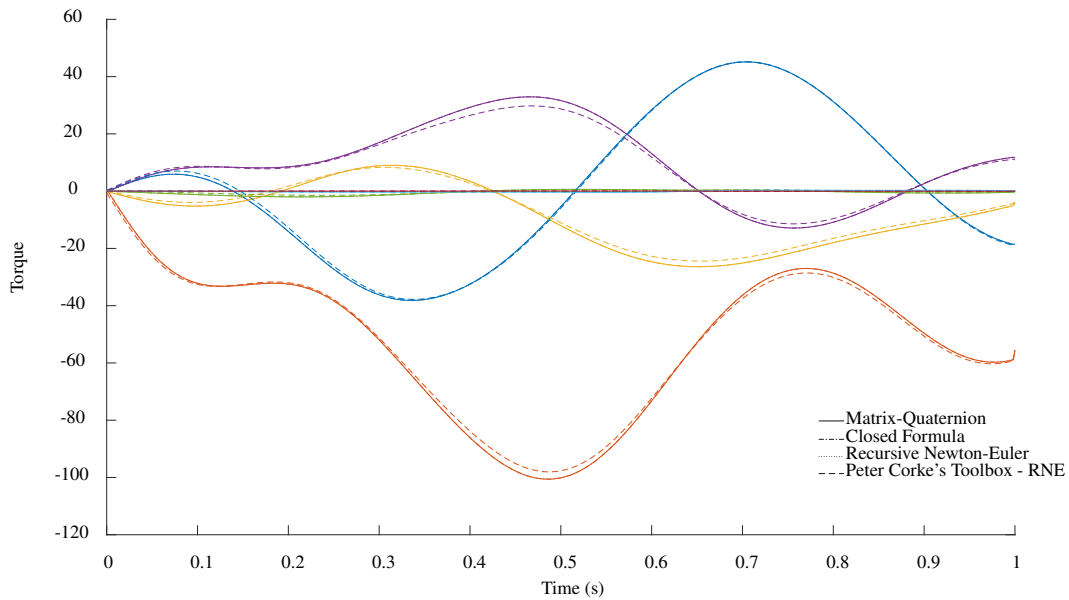


Figure 6.8: Torque plot for each of the joints in different versions of the algorithm. The straight lines are the results for the matrix-quaternion formulation, the dash-dotted lines refers to the closed equation and the dotted line are the torques for the recursive algorithm. Finally the dashed line are the resulting torques for the recursive Newton-Euler algorithm from Peter Corke's Toolbox.

As one last remark, we would like to add on Subsection 6.3.2 discussion about the error. As can be seen in Figure 6.9, we have the blocks referring to the computed torque control, which we are using our dqRNEA algorithm, and we have Peter Corke's toolbox's direct dynamics. One of the reasons we chose not to design the direct dynamics ourselves was to ensure a more realistic depiction of the controller: if the direct dynamics was also our own, we would always have the error equal to zero, as we would be just canceling the terms we provided ourselves. However, this is not the case, for this experiment, as there are uncertainties between our model and the toolbox's. As we have discussed, this uncertainties were visible with the resulting torques of Figure 6.8, and they are also visible on the steady state errors in the results of Figure 6.11, which have a higher order of magnitude than the errors in Figure 6.6.

In the case of our experiments with our Kuka model were able to reach its target position and work as expected, despite the steady state error. This result however, can motivate a further research on the topic of computed torque control. Indeed, as discussed in Subsection 4.4.1, we have only implemented a simple PD controller that uses feedback linearization in order to negate the non-linear terms. This controller is very powerful and operates very well in scenarios in which the robot model is well known, as was the case for the Two Link planar arm. For cases when the robot model is not completely known or in the presence of more uncertainties it starts to fail, creating the need for more sophisticated controllers. Although this is not in the scope of this manuscript, there are many works discussing different, more sophisticated solutions can be found in the works in [108, 109], in which the authors review many different solutions for improving the control strategy.

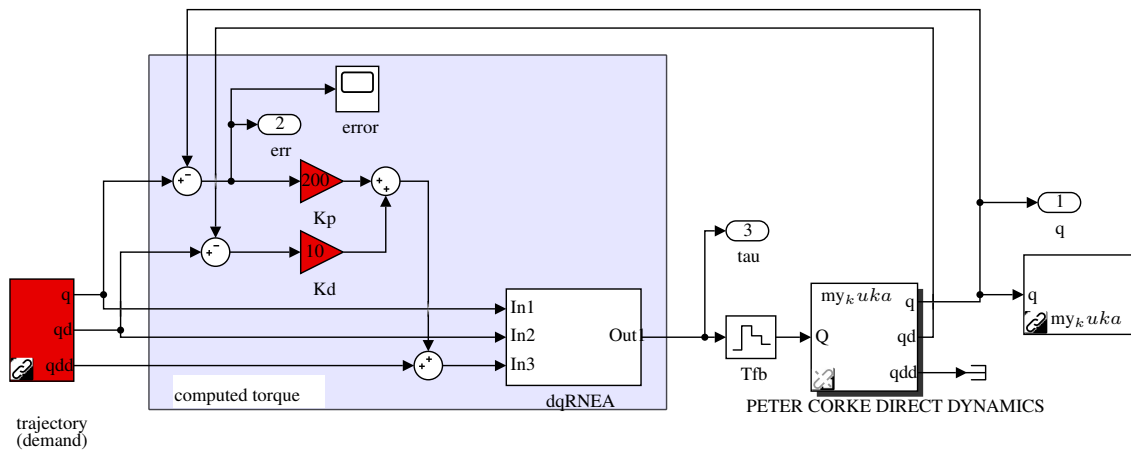
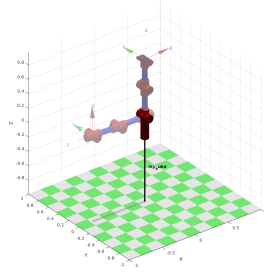
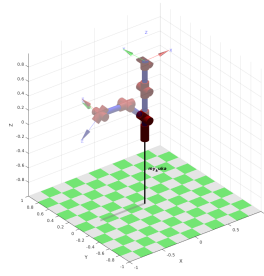


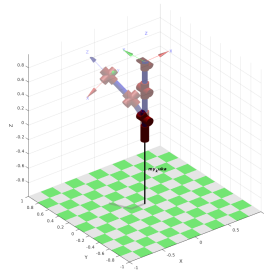
Figure 6.9: Simulink Computed Torque Control Diagram for KUKA LWR4. This is based on Peter Corke robotic toolbox with the addition of our dual quaternion based recursive Newton Euler algorithm.



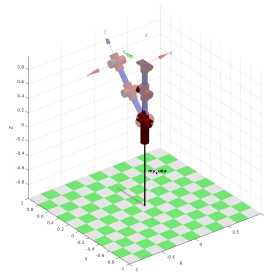
(a) Robot initial and final position: $\theta_0 = \{0, 0, 0, 0, 0, 0, 0\}$, $\theta_{end} = \{0, \frac{\pi}{2}, 0, 0, 0, 0, 0\}$.



(b) Robot initial and final position: $\theta_0 = \{0, 0, 0, 0, 0, 0, 0\}$, $\theta_{end} = \{0, \frac{\pi}{4}, 0, -\frac{\pi}{4}, 0, \frac{\pi}{4}, 0\}$

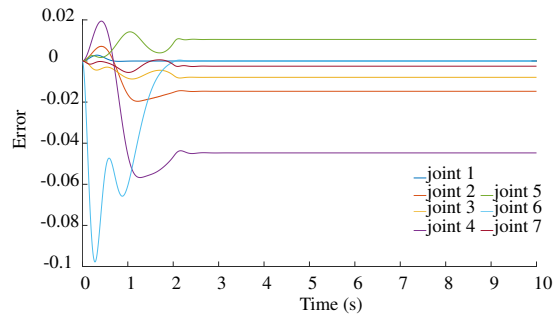


(c) Robot initial and final position: $\theta_0 = \{0, 0, 0, 0, 0, 0, 0\}$, $\theta_{end} = \{0, \frac{\pi}{6}, \frac{\pi}{2}, \frac{\pi}{4}, 0, 0, \frac{\pi}{2}\}$

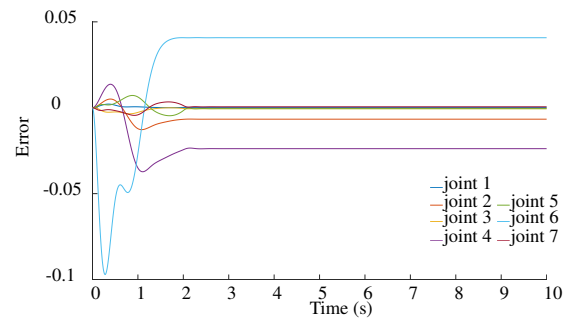


(d) Robot initial and final position: $\theta_0 = \{0, 0, 0, 0, 0, 0, 0\}$, $\theta_{end} = \{\frac{\pi}{2}, -\frac{\pi}{4}, \pi, 0, \pi, 0, \frac{\pi}{4}\}$

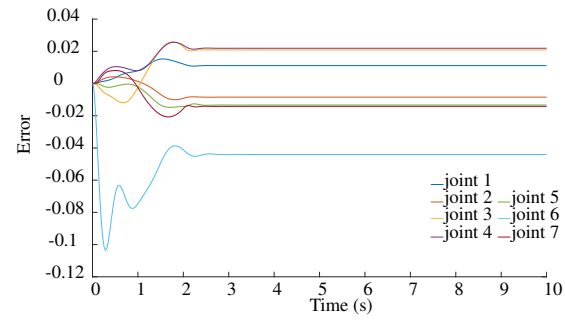
Figure 6.10: Robot plot at different configurations. Here we merged the images for the plot at the initial joint configuration θ_0 and the final joint configuration θ_{end} .



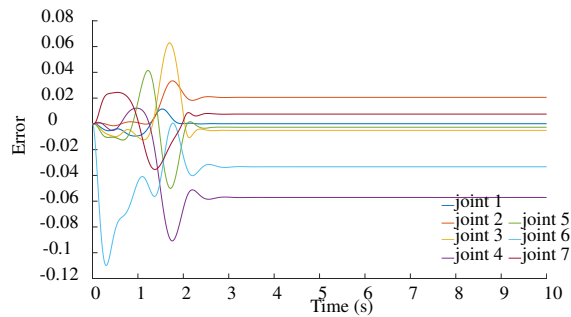
(a) Error progression for achieving the final position in Figure 6.10a



(b) Error progression for achieving the final position in Figure 6.10b



(c) Error progression for achieving the final position in Figure 6.10c



(d) Error progression for achieving the final position in Figure 6.10d

Figure 6.11: Plot for the error progression for the experiments in Figure 6.10

7

CONCLUSION

This manuscript focused on the study and development of a novel, cost efficient, formalization for a dual quaternion based recursive Newton-Euler algorithm. Here we have explored different concepts in the literature and developed different tools in order to better implement our dqRNEA thus contributing to strengthening a concept that is still lacking in the literature.

At the beginning of this work we set out not only to design a model for the inverse dynamics of a serial manipulator based on DQ, but also, to answer whether this model would retain all the advantages that dual quaternions usually have when used to represent the kinematics of system. Our main approach to answer this question was an extensive computational cost analysis of the algorithm and comparison with a similar model described in another algebra—after all, one of the main characteristics of dual quaternions is that it is more efficient than other non-minimal representations. Indeed we noticed from our analysis that our algorithm performs better than an algorithm based on Homogeneous Transformation Matrices.

One particularity of our model is that we have also introduced some new mathematical tools to allow us to create the closed form Newton-Euler equations, namely the dual quaternion vectors and matrices. Still, perhaps our main contribution in the the form of dual quaternion basic functions was probably our alternative description of the adjoint transformation, which perform much better than the traditional map, allowing for even more efficiency.

More so, we also aimed at ensuring the validity of our model by means of a few experiments in a simulated environment. In this manner, we have developed a package for MATLAB to implement our functions and we analyzed its validity both compared to another implementation of the RNEA and by itself on a controller. As our algorithm rely on an accurate robot description, we have been able to produce both very precise results as well as some results with some level of error (in the case of the experiment with the Kuka). However, the errors were still small and the controllers we have implemented still converged.

Finally, the dual quaternion description itself also endows the dqRNEA with some of the DQ proprieties: the representation is singularity free and the linear and angular velocities are coupled together—be it in as unit dual quaternions or representing twists and wrenches. Therefore, we believe the answer to our question is yes. It is indeed possible to represent the dynamics of a serial manipulator in dual quaternion algebra and it does retain the proprieties that make this representation attractive. And with this notion we also believe the equations provided in this manuscript opens up the possibilities of many subsequent research with dual quaternion and on the more ways we can further empower a robotic manipulator to act on our world.

7.1 FUTURE WORK

Although we were able to describe the dqRNEA algorithm, create a MATLAB package for it and validate our results in a simple controller, there is still a lot that can be done in this work. Indeed, as we have created a formulation for modeling a robotic arm using [37]’s framework for dual quaternions, we can use the many tools at our disposal both with regards to the dual quaternion framework and robotics theory to follow this work in many different directions.

One obvious direction regarding the possible future works that we can follow is to delve deeper in control theory and create more powerful controllers using the models proposed in this manuscript. Indeed, as we have discussed, the computed torque controller we used to validate our experiments is very powerful, however, it requires knowledge of robot model. This, of course, is an unrealistic scenario in the real world—models are never fully available to us, and when they are, they usually have many uncertainties and noise. In this manner, dealing with this is imperative for real world solutions that would operate safely and reliably in a number of applications. In [108, 109] there is a number of different suggestions on how to improve the computed torque controller. More so, a popular solution in the literature for multi-variable non-linear systems are sliding mode controllers [113–115].

Besides exploring the many options of controllers within the joint space, we may also transform our equations to the task-space and focus on controlling the end-effector itself. Indeed, this approach should open the possibility for many different applications, one of them being exploiting the already existing works on cooperative robotics within the dual quaternion framework to also use cooperative tasks primitives that encompass the forces of the system.

One other way of extending this work can also be to extend the model itself to be more general. In this work we have only considered serial robots with a fixed based and revolute joints. It could be advantageous to expand this to a more universal framework. Another choice for expanding this model is, by following the work in [116], to create the dqERNEA—a dual quaternion based elastic recursive Newton-Euler algorithm—describing a robot with elastic joints.

Of course, these are only some of the many possibilities of future works that can stem from what has been presented in this manuscript. We strongly believe our approach may also result in different works with varied scopes, as well as insights other descriptions would not provide.

A

INVERSE DYNAMIC ALGORITHM WITH HOMOGENEOUS MATRICES

In this chapter we introduce some of the basic concepts for the Homogeneous Transformation Matrices (HTM) as described in [5, 36, 49, 85, 93]. In Section A.1 we present some of the main operations when using HTM, in particular the ones needed for a clear understanding of the classic version of recursive Newton-Euler inverse dynamics algorithm, which we will present in Section A.2.

A.1 HTM

In order to represent positions and rigid motions in space we may use a notation with matrices and vectors. In this section we shall start out by showing how we may separately write a translation (or a point) as a vector in \mathbb{R}^3 and a rotation matrix in $\mathbb{R}^{3 \times 3}$, this shall be presented, respectively, in Subsections A.1.1 and A.1.2. Finally, in Subsection A.1.3 we show how to construct a Homogeneous transformation matrix, which simultaneously describes motions and rotations in space (analogous to the dual quaternion), and describe some of the most pertinent properties it has.

We highlight here that there are many properties regarding the mathematical tools introduced in this section, however, it is not the scope of this manuscript to review all particularities relating to HTM. Our aim in this Chapter is to present the recursive Newton-Euler algorithm in terms of an algebra that may be comparable to dual quaternions, and, in this manner, the purpose of this is only to overview some of the structures and operations that are used in the RNE algorithm we have chosen to present in Section A.2.

A.1.1 Positions, Twists, Forces and Wrenches as vectors

A rigid body's position \mathbf{p} with respect to the coordinate frame $\mathcal{F} = \{\mathbf{x}, \mathbf{y}, \mathbf{z}\}$ can be expressed as

$$\mathbf{p} = p_x \mathbf{x} + p_y \mathbf{y} + p_z \mathbf{z},$$

in which p_x, p_y, p_z are the coordinates for each axis. In order to compactly express this position we may use the vector formulation

$$\mathbf{p} = \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix}. \quad (\text{A.1})$$

We highlight here, that the formulation in (A.1) is isomorphic to the pure quaternion set, as we have discussed in Subsection 2.1.1.1.

Moreover, it is also interesting to note here that we may use vector algebra to write some other important

variables. For instance, the linear and angular velocities are represented, respectively, by

$$\mathbf{v} = \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix}, \text{ and } \boldsymbol{\omega} = \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix}. \quad (\text{A.2})$$

Following from (A.2) we can represent a body twist as

$$\mathbf{W} = \begin{bmatrix} \mathbf{v} \\ \boldsymbol{\omega} \end{bmatrix} \in \mathbb{R}^6. \quad (\text{A.3})$$

In this section, we also find it interesting to add that forces and moments can be represented similarly to how the linear and angular velocities of (A.2) are represented in \mathbb{R}^3 . That is,

$$\mathbf{f} = \begin{bmatrix} f_x \\ f_y \\ f_z \end{bmatrix}, \text{ and } \mathbf{m} = \begin{bmatrix} m_x \\ m_y \\ m_z \end{bmatrix}, \quad (\text{A.4})$$

with the subsequent wrench being

$$\mathbf{F} = \begin{bmatrix} \mathbf{m} \\ \mathbf{f} \end{bmatrix} \in \mathbb{R}^6. \quad (\text{A.5})$$

We also present here Definition A.1 for the Skew-symmetric representation of a vector.

Definition A.1. (Skew-symmetric representation of a vector) Given a vector $\mathbf{p} = [p_1 \ p_2 \ p_3] \in \mathbb{R}^3$, we may define a skew-symmetric matrix of \mathbf{p} , that is

$$[\mathbf{p}] = \begin{bmatrix} 0 & -p_3 & p_2 \\ p_3 & 0 & -p_1 \\ -p_2 & p_1 & 0 \end{bmatrix}. \quad (\text{A.6})$$

Being skew symmetric it follows from Definition A.1 that

$$[\mathbf{p}] = -[\mathbf{p}]^T. \quad (\text{A.7})$$

Another important propriety in this section is the adjoint transformation of a vector in \mathbb{R}^6 by another vector in \mathbb{R}^6 . In this case let us use the vector \mathbf{W} and \mathbf{F} , thus we have the mapping

$$\text{ad}_{\mathbf{W}}^T = \begin{bmatrix} [\boldsymbol{\omega}] & 0 \\ [\mathbf{v}] & [\boldsymbol{\omega}] \end{bmatrix}^T, \quad (\text{A.8})$$

which leads to a transformation of the type

$$\text{ad}_{\mathbf{W}}^T(\mathbf{F}) = \begin{bmatrix} [\boldsymbol{\omega}] & 0 \\ [\mathbf{v}] & [\boldsymbol{\omega}] \end{bmatrix}^T \begin{bmatrix} \mathbf{m} \\ \mathbf{f} \end{bmatrix} \quad (\text{A.9})$$

A.1.2 Rotation Matrix

While we may use a vector to describe a position, we use matrices to describe the orientation and rotation of a frame attached to an object. Thus, let us consider an orthonormal frame $\mathcal{F}_0 = \{\mathbf{x}, \mathbf{y}, \mathbf{z}\}$ attached to an object. The rotation from frame \mathcal{F}_0 to $\mathcal{F}_1 = \{\mathbf{x}^1, \mathbf{y}^1, \mathbf{z}^1\}$ can be defined through its individual components

$$\begin{aligned}\mathbf{x}^1 &= x_x^1 \mathbf{x} + x_y^1 \mathbf{y} + x_z^1 \mathbf{z}, \\ \mathbf{y}^1 &= y_x^1 \mathbf{x} + y_y^1 \mathbf{y} + y_z^1 \mathbf{z}, \\ \mathbf{z}^1 &= z_x^1 \mathbf{x} + z_y^1 \mathbf{y} + z_z^1 \mathbf{z}.\end{aligned}\tag{A.10}$$

Furthermore, those nine components can be expressed in the form of a rotation matrix,

$$\mathbf{R} = \begin{bmatrix} x_x^1 & y_x^1 & z_x^1 \\ x_y^1 & y_y^1 & z_y^1 \\ x_z^1 & y_z^1 & z_z^1 \end{bmatrix} \in \mathbb{R}^{3 \times 3}.\tag{A.11}$$

Although the rotation matrix is described as a set of nine components, we must add that the orientation of an object in space has only three degrees of freedom. In this manner only three entries in (A.11) can be chosen independently—or, in other words, the for \mathbf{R} to be a rotation matrix, it has six constrains:

1. Unit Norm: As we are only rotating the orthogonal frame \mathcal{F}_0 (in which all axis are unitary), we must have a resulting frame with its axis being restricted to the unit norm as well. That is, from (A.10) we have

$$\begin{aligned}(x_x^1)^2 + (x_y^1)^2 + (x_z^1)^2 &= 1, \\ (y_x^1)^2 + (y_y^1)^2 + (y_z^1)^2 &= 1, \\ (z_x^1)^2 + (z_y^1)^2 + (z_z^1)^2 &= 1.\end{aligned}\tag{A.12}$$

2. Orthogonality: All three axis of the new frame must remain orthogonal. Thus,

$$\mathbf{x}^1 \cdot \mathbf{y}^1 = \mathbf{x}^1 \cdot \mathbf{z}^1 = \mathbf{y}^1 \cdot \mathbf{z}^1 = 0.\tag{A.13}$$

Indeed, all those constrains can be compactly express through the condition

$$\mathbf{R}^T \mathbf{R} = \mathbf{I}_3,\tag{A.14}$$

in which \mathbf{I}_3 is the identity matrix.

Finally, to account for the handedness of the matrix, as the above conditions do not distinguish between a right-handed frame (in which $\mathbf{x}^1 \times \mathbf{y}^1 = \mathbf{z}^1$) or a left-handed frame (in which $\mathbf{x}^1 \times \mathbf{y}^1 = -\mathbf{z}^1$). Therefore, let us consider the determinant equation for our matrix (A.1.2)

$$\det R = (\mathbf{x}^1)^T (\mathbf{y}^1 \times \mathbf{z}^1) = (\mathbf{z}^1)^T (\mathbf{x}^1 \times \mathbf{y}^1) = (\mathbf{y}^1)^T (\mathbf{z}^1 \times \mathbf{x}^1).\tag{A.15}$$

If we have a right-handed matrix then our determinant will be $\det R = 1$, otherwise we will have $\det R = -1$.

We also mention here that the rotation matrices which follows the constrains imposed by (A.13) and that are right-handed ($\det R = 1$) form the special orthogonal group $SO(3)$.

A.1.3 Homogeneous Transformation Matrices

To close this section we shall present the formulation for the homogeneous transformation matrix, which takes advantage of both (A.1) and (A.14) to represent the translation and rotation of a body simultaneously. Thus, we have

$$\mathbf{T} = \begin{bmatrix} \mathbf{R} & \mathbf{p} \\ 0 & 1 \end{bmatrix} \in \mathbb{R}^{4 \times 4}, \quad (\text{A.16})$$

which is also an element of the Special Euclidean Group $SE(3)$, denoting a full rigid body motion.

The many proprieties and uses of the HTMs can be found in the works of [5, 49, 93, 117]. In this section, though, we find it important to show the adjoint mapping, which will be used in on the RNE algorithm of next section.

Let us take the twist \mathbf{W}_1 in frame \mathcal{F}_1 , considering the transformation \mathbf{T} that would lead to \mathbf{W}_2 in \mathcal{F}_2 , we can define the adjoint map in a matrix formulation to be

$$[\text{Ad}_{\mathbf{T}}] = \begin{bmatrix} \mathbf{R} & 0 \\ [\mathbf{p}] \mathbf{R} & \mathbf{R} \end{bmatrix} \in \mathbb{R}^{6 \times 6}. \quad (\text{A.17})$$

Thus, making the frame transformation become

$$\mathbf{W}_2 = [\text{Ad}_{\mathbf{T}}] \mathbf{W}_1. \quad (\text{A.18})$$

We remark here that this is only one of the the formulations for this adjoint transformation, there are a number of other manners to represent it in the literature.

A.2 THE NEWTON-EULER ALGORITHM AS PRESENTED IN THE LITERATURE

There are many different formulations for the Recursive Newton-Euler algorithm in the literature [5, 36, 49, 56, 118–120], in this section, particularly, we shall present the RNEA as shown in [49] as we believe that this version is particularly well suited when comparing to our version of the dqRNEA, as it uses HTM. For one we have the similarities between the dual quaternions and the homogeneous transformation matrices, as the two representations can fully describe both the translation and rotation of a body in a single structure. Furthermore, both DQ and HTM are free from representations singularities, making them more reliable for many applications.

This algorithm is initialized by attaching frame \mathcal{F}_0 to the base, frames \mathcal{F}_1 to \mathcal{F}_n to the center of mass of each link and frame \mathcal{F}_{n+1} to the end-effector (considering an n -link manipulator). Furthermore the initial twist is $\mathbf{W}_0 = 0$ and $\mathbf{W}_0 = \begin{bmatrix} 0 & -\mathbf{g} \end{bmatrix}^T$, the initial forces at the end effector are $\mathbf{F}_{n+1} = \begin{bmatrix} \mathbf{m}_{\text{tip}} & \mathbf{f}_{\text{tip}} \end{bmatrix}^T$ and the transformation from link $i - 1$ to link i at the home configuration is given by \mathbf{M}_i^{i-1} . The inertia matrix is given by $\mathcal{G} \in \mathbb{R}^{6 \times 6}$, the joint variables are θ , $\dot{\theta}$ and $\ddot{\theta}$ and the screw axis is given by $\mathbf{A} \in \mathbb{R}^6$.

We then have the main steps of the algorithm, which are similar to the dqRNEA—or, for that matter, to any other version of the recursive Newton-Euler algorithm. We first we calculate the link's transformation

$$\mathbf{T}_i^{i-1} = \exp(-[\mathbf{A}_i] \theta_i) \mathbf{M}_i^{i-1}, \quad (\text{A.19})$$

and the forward kinematics of a serial chain

$$\mathbf{T}_{EF} = \mathbf{T}_1^0 \mathbf{T}_2^1 \cdots \mathbf{T}_{n+1}^n. \quad (\text{A.20})$$

Then we calculate the twist and its derivative

$$\mathbf{W}_i = \left[\text{Ad}_{\mathbf{T}_i^{i-1}} \right] \mathbf{W}_{i-1} + \mathbf{A}_i \dot{\theta}_i, \quad (\text{A.21})$$

$$\dot{\mathbf{W}}_i = \left[\text{Ad}_{\mathbf{T}_i^{i-1}} \right] \dot{\mathbf{W}}_{i-1} + \text{ad}_{\mathbf{W}_i}(\mathbf{A}_i) \dot{\theta}_i + \mathbf{A}_i \ddot{\theta}_i. \quad (\text{A.22})$$

Next we have the backward iteration, here we start out calculating the resulting forces for each link,

$$\mathbf{F}_{R,i} = \mathbf{G}_i \dot{\mathbf{W}}_i - \text{ad}_{\mathbf{W}_i}^T(\mathbf{G}_i \mathbf{W}_i). \quad (\text{A.23})$$

Then we have that \mathbf{F}_i is given by

$$\mathbf{F}_i = \left[\text{Ad}_{\mathbf{T}_{i+1}^i} \right]^T \mathbf{F}_{i+1} + \mathbf{F}_{R,i}, \quad (\text{A.24})$$

and the torque is

$$\tau_i = \mathbf{F}_i^T \mathbf{A}_i. \quad (\text{A.25})$$

B PUBLICATIONS

Some of the results presented in this manuscript have been published in the following paper:

- C. M. Farias, L. F. C. Figueredo and J. Y. Ishihara. Performance Study on dqRNEA - A Novel Dual Quaternion based Recursive Newton-Euler Inverse Dynamics Algorithms. In 2019 IEEE Third International Conference on Robotic Computing (IRC), 2019 (Accepted - 27% acceptance rate). Naples, Italy, Feb 2019, pp. 94-101. doi: 10.1109/IRC.2019.00022

Bibliography

- [1] J. Yan-Bin, “Dual quaternion handout,” May, 2018. [Online]. Available: "<http://web.cs.iastate.edu/cs577/handouts/>"
- [2] M. Cefalo, “Notes on the kuka lwr4 dynamic model,” 2015. [Online]. Available: "<http://www.coppeliarobotics.com/contributions/>"
- [3] T. Logsdon, *The Robot Revolution*. Simon & Schuster Trade, 1984, vol. 1.
- [4] A. Gasparetto, *Robots in History: Legends and Prototypes from Ancient Times to the Industrial Revolution*, 2016, vol. 32, pp. 39–49.
- [5] B. Siciliano and O. Khatib, *Springer Handbook of Robotics*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2007.
- [6] J. Wallén, *The History of the Industrial Robot*. Linköping: Linköping University Electronic Press, 2008.
- [7] A. Gasparetto and L. Scalera, “A brief history of industrial robotics in the 20th century,” *Advances in Historical Studies*, vol. 08, pp. 24–35, 2019.
- [8] I. Zamalloa, R. Kojcev, A. Hernández, I. Muguruza, L. U. S. Juan, A. Bilbao, and V. Mayoral, “Dissecting robotics - historical overview and future perspectives,” *CoRR*, vol. abs/1704.08617, 2017.
- [9] L. P. E. Toh, A. Causo, P.-W. Tzuo, I.-M. Chen, and S. H. Yeo, “A review on the use of robots in education and young children,” *Journal of Educational Technology & Society*, vol. 19, no. 2, pp. 148–163, 2016.
- [10] I. R. Nourbakhsh *et al.*, “Robots and education in the classroom and in the museum: On the study of robots, and robots for study,” 2000.
- [11] O. Mubin, C. J. Stevens, S. Shahid, A. Al Mahmud, and J.-J. Dong, “A review of the applicability of robots in education,” *Journal of Technology in Education and Learning*, vol. 1, no. 209-0015, p. 13, 2013.
- [12] T. Belpaeme, J. Kennedy, A. Ramachandran, B. Scassellati, and F. Tanaka, “Social robots for education: A review,” *Science Robotics*, vol. 3, no. 21, 2018.
- [13] A. R. Lanfranco, A. E. Castellanos, J. P. Desai, and W. C. Meyers, “Robotic surgery: a current perspective,” *Annals of surgery*, vol. 239, no. 1, p. 14, 2004.
- [14] H. S. Ahn, S. Zhang, M. H. Lee, J. Y. Lim, and B. A. MacDonald, “Robotic healthcare service system to serve multiple patients with multiple robots,” in *Social Robotics*, S. S. Ge, J.-J. Cabibihan, M. A. Salichs, E. Broadbent, H. He, A. R. Wagner, and Á. Castro-González, Eds. Cham: Springer International Publishing, 2018, pp. 493–502.
- [15] J. Abdi, A. Al-Hindawi, T. Ng, and M. P. Vizcaychipi, “Scoping review on the use of socially assistive robot technology in elderly care,” *BMJ open*, vol. 8, no. 2, 2018.
- [16] S. M. Silverstein and K. Yamane, *Humanoid Robots for Entertainment*. Dordrecht: Springer Netherlands, 2019, pp. 2599–2615.

- [17] V. A. Ziparo, M. Zaratti, G. Grisetti, T. M. Bonanni, J. Serafin, M. Di Cicco, M. Proesmans, L. van Gool, O. Vysotska, I. Bogoslavskyi, and C. Stachniss, "Exploration and mapping of catacombs with mobile robots," in *2013 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, Oct 2013, pp. 1–2.
- [18] L. Bordoni, F. Mele, and A. Sorgente, *Artificial Intelligence for Cultural Heritage*. Cambridge Scholars Publishing, 2016.
- [19] O. Khatib, X. Yeh, G. Brantner, B. Soe, B. Kim, S. Ganguly, H. Stuart, S. Wang, M. Cutkosky, A. Edsinger *et al.*, "Ocean one: A robotic avatar for oceanic discovery," *IEEE Robotics & Automation Magazine*, vol. 23, no. 4, pp. 20–29, 2016.
- [20] P. Nilsson, S. Haesaert, R. Thakker, K. Otsu, C. I. Vasile, A.-A. Agha-Mohammadi, R. M. Murray, and A. D. Ames, "Toward specification-guided active mars exploration for cooperative robot teams." in *Robotics: Science and Systems*, 2018.
- [21] T. Huntsberger, G. Rodriguez, and P. S. Schenker, "Robotics challenges for robotic and human mars exploration," *Proceedings of ROBOTICS2000, Albuquerque, NM*, 2000.
- [22] M. F. Stieber, C. P. Trudel, and D. G. Hunter, "Robotic systems for the international space station," in *Proceedings of International Conference on Robotics and Automation*, vol. 4, April 1997, pp. 3068–3073 vol.4.
- [23] H. Li, B. Qin, Z. Jiang, and M. Ceccarelli, "Multi-arm motion planning of beijing astronaut robot," in *Advances in Mechanism and Machine Science*, T. Uhl, Ed. Cham: Springer International Publishing, 2019, pp. 2873–2882.
- [24] J. E. Scott and C. H. Scott, "Models for drone delivery of medications and other healthcare items," in *Unmanned Aerial Vehicles: Breakthroughs in Research and Practice*. IGI Global, 2019, pp. 376–392.
- [25] R. Bogue, "Disaster relief, and search and rescue robots: the way forward," *Industrial Robot: the international journal of robotics research and application*, vol. 46, no. 2, pp. 181–187, 2019.
- [26] M. Moniruzzaman, M. S. R. Zishan, S. Rahman, S. Mahmud, and A. Shaha, "Design and implementation of urban search and rescue robot," *Int. J. Eng. Manuf.(IJEM)*, vol. 8, no. 2, pp. 12–20, 2018.
- [27] J. Heinzmann and A. Zelinsky, "A safe-control paradigm for human–robot interaction," *Journal of Intelligent and Robotic Systems*, vol. 25, no. 4, pp. 295–310, Aug 1999.
- [28] B. Vilhena Adorno, "Robot Kinematic Modeling and Control Based on Dual Quaternion Algebra — Part I: Fundamentals." Feb. 2017, working paper or preprint.
- [29] A. Valverde and P. Tsiotras, "Dual quaternion framework for modeling of spacecraft-mounted multibody robotic systems," *Frontiers in Robotics and AI*, vol. 5, Nov, 2018.
- [30] —, "Modeling of spacecraft-mounted robot dynamics and control using dual quaternions," June 2018, pp. 670–675.
- [31] N. Filipe and P. Tsiotras, "Simultaneous position and attitude control without linear and angular velocity feedback using dual quaternions," in *Proceedings of the 2013 American Control Conference (ACC)*, 2013, pp. 4808–4813.

- [32] —, “Adaptive Model-Independent Tracking of Rigid Body Position and Attitude Motion with Mass and Inertia Matrix Identification using Dual Quaternions,” *AIAA Guidance, Navigation, and Control (GNC) Conference*, no. January 2015, pp. 1–15, 2013.
- [33] X. Wang and C. Yu, “Feedback linearization regulator with coupled attitude and translation dynamics based on unit dual quaternion,” in *IEEE International Symposium on Intelligent Control - Proceedings*, 2010, pp. 2380–2384.
- [34] J. R. Dooley and J. M. McCarthy, “Spatial rigid body dynamics using dual quaternion components,” in *Proceedings of 1991 IEEE International Conference on Robotics and Automation*, 1991, pp. 90–95.
- [35] D. Han, Q. Wei, and Z. Li, “A dual-quaternion method for control of spatial rigid body,” in *2008 IEEE International Conference on Networking, Sensing and Control*, April 2008, pp. 1–6.
- [36] B. Siciliano, L. Sciacivco, L. Villani, and G. Oriolo, *Robotics: modelling, planning and control*. Springer Verlag, 2009.
- [37] B. V. Adorno, “Two-arm Manipulation: From Manipulators to Enhanced Human-Robot Collaboration,” PhD thesis, Laboratoire d’Informatique, de Robotique et de Microélectronique de Montpellier (LIRMM) - Université Montpellier 2, Montpellier, France, 2011.
- [38] X. Wang and H. Zhu, “On the comparisons of unit dual quaternion and homogeneous transformation matrix,” *Advances in Applied Clifford Algebras*, vol. 24, pp. 213–229, 2014.
- [39] N. A. Aspragathos and J. K. Dimitros, “A Comparative Study of Three Methods for Robot Kinematics,” *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 28, no. 2, pp. 135–145, 1998.
- [40] E. Özgür and Y. Mezouar, “Kinematic modeling and control of a robot arm using unit dual quaternions,” *Robotics and Autonomous Systems*, vol. 77, pp. 66–73, 2016.
- [41] J. Funda and R. Paul, “A computational analysis of screw transformations in robotics,” *IEEE Transactions on Robotics and Automation*, vol. 6, no. 3, pp. 348–356, 1990.
- [42] S. Chiaverini, “Singularity-robust task-priority redundancy resolution for real-time kinematic control of robot manipulators,” *IEEE Transactions on Robotics and Automation*, vol. 13, no. 3, pp. 398–410, Jun. 1997.
- [43] X. Wang, D. Han, C. Yu, and Z. Zheng, “The geometric structure of unit dual quaternion with application in kinematic control,” *Journal of Mathematical Analysis and Applications*, vol. 389, no. 2, pp. 1352–1364, May 2012.
- [44] B. V. Adorno, P. Fraisse, and S. Druon, “Dual position control strategies using the cooperative dual task-space framework,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Taipei, Oct. 2010, pp. 3955–3960.
- [45] L. F. C. Figueredo, B. V. Adorno, J. Y. Ishihara, and G. A. Borges, “Robust kinematic control of manipulator robots using dual quaternion representation,” in *2013 IEEE International Conference on Robotics and Automation*, 2013, pp. 1949–1955.
- [46] C. M. de Farias, Y. G. Rocha, L. F. C. Figueredo, and M. C. Bernardes, “Design of singularity-robust and task-priority primitive controllers for cooperative manipulation using dual quaternion representation,” in *IEEE Conference on Control Technology and Applications (CCTA)*, 2017, pp. 740–745.

- [47] K. Daniilidis, "Hand-eye calibration using dual quaternions," *The International Journal of Robotics Research*, vol. 18, pp. 286–298, 1999.
- [48] J. Cheng, J. Kim, Z. Jiang, and W. Che, "Dual quaternion-based graphical slam," *Robotics and Autonomous Systems*, vol. 77, pp. 15 – 24, 2016.
- [49] K. Lynch and F. Park, "Modern robotics." Cambridge University Press, 2017, ch. 8.
- [50] R. Featherstone, *Rigid Body Dynamics Algorithms*. Berlin, Heidelberg: Springer-Verlag, 2007.
- [51] J. Carpentier, "Analytical Derivatives of Rigid Body Dynamics Algorithms," May 2018, robotics: Science and Systems.
- [52] T. R. Kane and D. A. Levinson, "The use of kane's dynamical equations in robotics," *The International Journal of Robotics Research*, vol. 2, no. 3, pp. 3–21, 1983.
- [53] X. Wang and C. Yu, "Feedback linearization regulator with coupled attitude and translation dynamics based on unit dual quaternion," in *IEEE International Symposium on Intelligent Control - Part of 2010 IEEE Multi-Conference on Systems and Control*, 2010, pp. 2380–2384.
- [54] N. Filipe and P. Tsiotras, "Adaptive position and attitude-tracking controller for satellite proximity operations using dual quaternions," *Journal of Guidance, Control, and Dynamics*, vol. 38, no. 4, pp. 566–577, 2014.
- [55] J. M. Hollerbach, "A recursive lagrangian formulation of manipulator dynamics and a comparative study of dynamics formulation complexity," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 10, pp. 730–736, 1980.
- [56] J. Y. S. Luh, M. W. Walker, and R. P. C. Paul, "On-line computation scheme for mechanical manipulators," *Journal of Dynamic Systems Measurement and Control-transactions of The Asme - J DYN SYST MEAS CONTR*, vol. 102, 06 1980.
- [57] M. Nasser, "Recursive newton-euler formulation of manipulator dynamics," in *NASA Conference on Space Telerobotics*, 1989, p. 309.
- [58] X. L. Yang, H. T. Wu, Y. Li, S. Z. Kang, and B. Chen, "Computationally Efficient Inverse Dynamics of a Class of Six-DOF Parallel Robots: Dual Quaternion Approach," *Journal of Intelligent and Robotic Systems: Theory and Applications*, pp. 1–13, 2018.
- [59] V. Brodsky and M. Shoham, "The Dual Inertia Operator and Its Application to Robot Dynamics," *Journal of Mechanical Design*, vol. 116, no. 4, p. 1089, 1994.
- [60] C. Miranda de Farias, L. F. da Cruz Figueredo, and J. Yoshiyuki Ishihara, "Performance study on dqrne - a novel dual quaternion based recursive newton-euler inverse dynamics algorithms," in *2019 Third IEEE International Conference on Robotic Computing (IRC)*, Feb 2019, pp. 94–101.
- [61] R. Palais, "The classification of real division algebras," *American Mathematical Monthly*, vol. 75, pp. 366–, April 1968.
- [62] A. Buchmann, "A brief history of quaternions and of the theory of holomorphic functions of quaternionic variables," vol. 1, Nov 2011.

- [63] W. Clifford, "Preliminary sketch of biquaternions," *Proc. London Mathematical Society*, vol. 4, no. 64, pp. 381–395, 1873.
- [64] W. R. Hamilton, "On quaternions, or on a new system of imaginaries in algebra: Copy of a letter from Sir William R. Hamilton to John T. Graves, esq. on quaternions," *Journal of Mechanical Design - Transactions of ASME*, vol. 25, no. 3, pp. 489–495, 1844.
- [65] L. Ahlfors, *Complex Analysis: An Introduction to the Theory of Analytic Functions of One Complex Variable*, ser. International Series in pure and applied mathematics. McGraw-Hill Interamericana, 1953.
- [66] L. F. da Cruz Figueredo, "Kinematic control based on dual quaternion algebra and its application to robot manipulators," Ph.D. dissertation, Universidade de Brasilia, July 2016.
- [67] J. M. Selig, *Geometric Fundamentals of Robotics*, 2nd ed. Springer-Verlag New York Inc., 2005.
- [68] H. Anton, *Elementary linear algebra*. John Wiley & Sons, 2010.
- [69] A. A. Harkin and J. B. Harkin, "Geometry of generalized complex numbers," *Mathematics Magazine*, vol. 77, no. 2, pp. 118–129, April 2004.
- [70] B. Akyar, "Dual quaternions in spatial kinematics in an algebraic sense," *Turk. J. Math*, vol. 32, pp. 373–391, 2008.
- [71] B. Kenwright, "A beginners guide to dual-quaternions: What they are, how they work, and how to use them for 3d," in *The 20th International Conference on Computer Graphics, Visualization and Computer Vision, WSCG 2012 Communication Proceedings*, 2012, pp. 1–13.
- [72] F. Bullo and R. Murray, "Proportional derivative (PD) control on the euclidean group," Division of Engineering and Applied Science, California Institute of Technology, Technical Report Caltech/CDS 95–010, 1995.
- [73] F. C. Park, "Distance Metrics on the Rigid-Body Motions with Applications to Mechanism Design," *ASME. Journal of Mechanical Design*, vol. 117, no. 1, pp. 48–54, 1995.
- [74] J. C. K. Chou, "Quaternion kinematic and dynamic differential equations," *IEEE Transactions on Robotics and Automation*, vol. 8, no. 1, pp. 53–64, Feb. 1992.
- [75] O. P. Agrawal, "Hamilton operators and dual-number-quaternions in spatial kinematics," *Mechanism and Machine Theory*, vol. 22, no. 6, pp. 569–575, 1987.
- [76] B. Kenwright, "Dual quaternion - from classical mechanics to computer graphics and beyond," 2012.
- [77] A.-N. Sharkawy and N. Aspragathos, "A comparative study of two methods for forward kinematics and Jacobian matrix determination," in *2017 International Conference on Mechanical, System and Control Engineering, ICMSC 2017*, 2017.
- [78] I. Yaglom, *Complex numbers in geometry*. Academic Press, Translated by E. J.F. Primrose, 1968.
- [79] J. Kuipers, *Quaternions and Rotation Sequences: A Primer with Applications to Orbits, Aerospace, and Virtual Reality*. Princeton University Press, 1999.

- [80] Y. Wu, X. Hu, D. Hu, T. Li, and J. Lian, "Strapdown inertial navigation system algorithms based on dual quaternions," *IEEE Transactions On Aerospace And Electronic Systems*, vol. 41, no. 1, pp. 110–132, 2005.
- [81] H. S. M. Coxeter, "Review: Methods of algebraic geometry." *Bull. Amer. Math. Soc.*, vol. 2, no. 6, pp. 678–679, 11 1952.
- [82] H. Coxeter, *Non-Euclidean Geometry: Fifth Edition*, ser. Heritage. University of Toronto Press, Scholarly Publishing Division, 1965.
- [83] M. Mason, *Mechanics of Robotic Manipulation*, ser. Intelligent Robotics and Autonomous Agents series. MIT Press, 2001.
- [84] M.-j. Kim, M.-s. Kim, and S. Y. Shin, "A compact differential formula for the first derivative of a unit quaternion curve," *The Journal of Visualization and Computer Animation*, vol. 7, no. 1, pp. 43–57, 1996.
- [85] S. S. S. Richard M. Murray, Zexiang Li, *A Mathematical Introduction to Robotic Manipulation*. CRC Press, 3 1994.
- [86] M. P. M. Paulo Marcos de Aguiar, Glauco Augusto de Paula Caurin, "Análise cinestática de uma garra antropomórfica de três dedos utilizando a teoria helicoidal," in *II Congresso Nacional de Engenharia Mecânica*. ABMC, 2012.
- [87] T. Schon, "Chasles theorem," November 2008. [Online]. Available: <https://www.control.isy.liu.se/student/graduate/DynVis/Lectures/le1Chasles.pdf>
- [88] J. Gallardo-Alvarado, *Kinematic Analysis of Parallel Manipulators by Algebraic Screw Theory*. Springer, June 2016.
- [89] B. Busam, T. Birdal, and N. Navab, "Camera pose filtering with local regression geodesics on the riemannian manifold of dual quaternions," *2017 IEEE International Conference on Computer Vision Workshops (ICCVW)*, pp. 2436–2445, 2017.
- [90] L. Kavan, S. Collins, C. OâSullivan, and J. Zara, "Dual quaternions for rigid transformation blending," *Technical report, Trinity College Dublin*, 2006.
- [91] D. S. Byung-EulJun and N. HarrisMcClamroch, "Identification of the inertia matrix of a rotating body based on errors-in-variables models," *IFAC Proceedings Volumes*, vol. 40, no. 12, pp. 182 – 187, 2007, 7th IFAC Symposium on Nonlinear Control Systems.
- [92] A. Valverde, "Dynamic modeling and control of spacecraft robotic systems using dual quaternions," Ph.D. dissertation, Georgia Institute of Technology, 2018.
- [93] M. Spong, S. Hutchinson, and M. Vidyasagar, *Robot modeling and control*. John Wiley & Sons, 2006.
- [94] A. S. de Oliveira, E. R. De Pieri, and U. F. Moreno, "A new method of applying differential kinematics through dual quaternions," *Robotica*, vol. 35, no. 4, p. 907â921, 2017.
- [95] R. M. Murray, Z. Li, and S. S. Sastry, *A Mathematical Introduction to Robotic Manipulation*. CRC, 1994.

- [96] L. F. C. Figueredo, B. V. Adorno, J. Y. Ishihara, and G. A. Borges, "Switching strategy for flexible task execution using the cooperative dual task-space framework," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2014, pp. 1703–1709.
- [97] J. Vilela, L. F. C. Figueredo, J. Y. Ishihara, and G. A. Borges, "Quaternion-based H_∞ kinematic attitude control subjected to input time-varying delays," in *2015 54th IEEE Conference on Decision and Control (CDC)*, 2015, pp. 7066–7071.
- [98] M. M. Marinho, L. F. C. Figueredo, and B. V. Adorno, "A dual quaternion linear-quadratic optimal controller for trajectory tracking," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015, pp. 4047–4052.
- [99] J. Cavalcanti, L. F. C. Figueredo, and J. Y. Ishihara, "Quaternion-based H_∞ attitude tracking control of rigid bodies with time-varying delay in attitude measurements," in *2016 IEEE 55th Conference on Decision and Control (CDC)*, 2016, pp. 1423–1428.
- [100] —, "Robust controller design for attitude dynamics subjected to time-delayed state measurements," *IEEE Transactions on Automatic Control*, 2016.
- [101] H. T. Kussaba, L. F. C. Figueredo, B. V. Adorno, and J. Y. Ishihara, "Hybrid kinematic control for rigid body pose stabilization using dual quaternions," *Journal of the Franklin Institute*, 2017.
- [102] B. Nemeč, N. Likar, A. Gams, and A. Ude, "Bimanual human robot cooperation with adaptive stiffness control," in *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*, Nov 2016, pp. 607–613.
- [103] G. Magnani, P. Rocco, L. Trevisan, A. M. Zanchettin, and A. Rusconi, "Torque control in the joint of a space robotic arm," in *IEEE 2009 International Conference on Mechatronics, ICM 2009*, 01 2009.
- [104] A. De Santis, V. Lippiello, B. Siciliano, and V. Luigi, *Human-Robot Interaction Control Using Force and Vision*, June 2007, vol. 353, pp. 51–70.
- [105] H.-S. Liu and Y. Huang, "Bounded adaptive output feedback tracking control for flexible-joint robot manipulators," *Journal of Zhejiang University-SCIENCE A*, vol. 19, no. 7, pp. 557–578, Jul 2018.
- [106] S. Ulrich, J. Z. Sasiadek, and I. Barkana, "Nonlinear adaptive output feedback control of flexible-joint space manipulators with joint stiffness uncertainties," *Journal of Guidance Control Dynamics*, vol. 37, pp. 1961–1975, 2014.
- [107] G. Garofalo, A. Werner, F. Loeffl, and C. Ott, "Joint-space impedance control using intrinsic parameters of compliant actuators and inner sliding mode torque loop," *IEEE Control Systems Letters*, vol. 3, no. 1, pp. 1–6, Jan 2019.
- [108] F. Lewis, D. Dawson, and C. Abdallah, *Robot Manipulator Control: Theory and Practice*, ser. Automation and Control Engineering. CRC Press, 2003.
- [109] V. S. R. Kelly and A. Loria, *Control of robot manipulators in joint space*, ser. Advanced Textbooks in Control and Signal Processing. Springer-Verlag London Limited 2005, 2005.
- [110] E. Sariyildiz and H. Temeltas, "A Singularity Free Trajectory Tracking Method for the Cooperative Working of Multi-Arm Robots using Screw Theory," in *Proceedings of the 2011 IEEE International Conference on Mechatronics*, Istanbul, Turkey, Apr. 2011, pp. 451–456.

- [111] C. P. Neuman and J. J. Murray, "Computational robot dynamics: Foundations and applications," *Journal of Robotic Systems*, vol. 2, no. 4, pp. 425–452, 1985.
- [112] P. Corke, *Robotics, Vision and Control: Fundamental Algorithms In MATLAB*. Springer International Publishing, 2017.
- [113] F. Piltan and N. B. Sulaiman, "Review of sliding mode control of robotic manipulator," 2012.
- [114] Y.-P. Chen and J.-L. Chang, "Sliding-mode force control of manipulators," *Proc. Natl. Sci. Counc. ROC(A)*, vol. 23, March 1999.
- [115] D. Shiferaw and A. Jain, "Comparision of joint space and task space integral sliding mode controller implementations for a 6-dof parallel robot," in *Proceedings of the 11th WSEAS International Conference on Robotics, Control and Manufacturing Technology, and 11th WSEAS International Conference on Multimedia Systems*. Stevens Point, Wisconsin, USA: World Scientific and Engineering Academy and Society (WSEAS), 2011, pp. 163–169.
- [116] G. Buondonno and A. De Luca, "A recursive newton-euler algorithm for robots with elastic joints and its application to control," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sep. 2015, pp. 5526–5532.
- [117] R. M. Murray, "Optimization-based control," in *Feedback Systems*, K. J. Åström and R. M. Murray, Eds. Princeton Univ. Press, 2010.
- [118] D. Orin, R. McGhee, M. Vukobratovic, and G. Hartoch, "Kinematic and kinetic analysis of open-chain linkages utilizing newton-euler methods," *Mathematical Biosciences*, vol. 43, no. 1, pp. 107 – 130, Feb. 1979.
- [119] R. Featherstone, *Robot dynamics algorithms*. The University of Edinburgh, 1984.
- [120] W. Khalil, "Dynamic modeling of robots using recursive newton-euler techniques." in *ICINCO 2010 - Proceedings of the 7th International Conference on Informatics in Control, Automation and Robotics*, vol. 1, Jun 2010, pp. 19–31.