



CONTROL ALLOCATION APPLIED TO ROBOTS SUBJECT TO INPUT CONSTRAINTS

Jhomolos Gomes Alves

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Engenharia Elétrica, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Mestre em Engenharia Elétrica.

Orientador: Fernando Cesar Lizarralde

Rio de Janeiro
Setembro de 2019

CONTROL ALLOCATION APPLIED TO ROBOTS SUBJECT TO INPUT
CONSTRAINTS

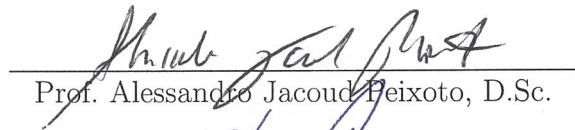
Jhomolos Gomes Alves

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DO INSTITUTO
ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE
ENGENHARIA (COPPE) DA UNIVERSIDADE FEDERAL DO RIO DE
JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A
OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA
ELÉTRICA.

Examinada por:



Prof. Fernando Cesar Lizarralde, D.Sc.



Prof. Alessandro Jacoud Feixoto, D.Sc.



Prof. Paulo Cesar Pellanda, Ph.D.

RIO DE JANEIRO, RJ – BRASIL
SETEMBRO DE 2019

Alves, Jhomolos Gomes

Control Allocation Applied to Robots Subject to Input Constraints/Jhomolos Gomes Alves. – Rio de Janeiro: UFRJ/COPPE, 2019.

XIX, 156 p.: il.; 29, 7cm.

Orientador: Fernando Cesar Lizarralde

Dissertação (mestrado) – UFRJ/COPPE/Programa de Engenharia Elétrica, 2019.

Referências Bibliográficas: p. 127 – 132.

1. Control Allocation. 2. Load Allocation. 3. Saturation. 4. Input Constraints. 5. Optimization.
I. Lizarralde, Fernando Cesar. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia Elétrica.
III. Título.

*I would like to dedicate this
dissertation to my grandmother
Venilda (in memoriam).*

Acknowledgements

First of all, may God be praised for giving me strength when I thought I could not make it. Only He knows how hard it has been and how much resilient I had to be to arrive here. Victory comes for those who wait and whose doubts have evolved into belief!

I would like to thank to the most important people in my life, my parents Cloves and Elenice. They have always been my safe harbor and have taught me the meaning of the words family and home. Moreover, I am thankful to my sister Cíntian for inspiring me to pursue my goals.

I would like to express my gratitude to my chief Capitain Hélio, who has been a big friend and has always encouraged me to keep going and never give up, to my work colleagues Miquéas, Rose and Silveira, for the friendship and for backing me up while I was attending the Master, and to the institution where I work, the Brazilian Air Force.

Special thanks to my advisor, Fernando Lizarralde, for teaching me Nonlinear Systems and for guiding me along the research and the writing of this dissertation. I also would like to thank the Federal University of Rio de Janeiro, for giving me the opportunity to continue my studies and to become a better professional.

I would also like to thank to the family I could choose - my friends. My cousin Anderson and my friends Kades, Haron, Camilo, Francielle, Elaine, Cecília, Ana Paula, Leonardo, Iuri and Pâmela have always made everything easier and have helped me not to take myself too seriously.

Finally, thanks to all my colleagues from LABCON and LEAD that have helped me along this journey.

Resumo da Dissertação apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

ALOCAÇÃO DE CONTROLE APLICADA A ROBÔS SUJEITOS A RESTRICÇÕES DE ENTRADA

Jhomolos Gomes Alves

Setembro/2019

Orientador: Fernando Cesar Lizarralde

Programa: Engenharia Elétrica

Em sistemas de controle com múltiplas entradas e múltiplas saídas (MIMO), o sinal de controle de um dado grau de liberdade pode ser distribuído entre vários atuadores através de uma estratégia denominada alocação de controle. Entretanto, o problema de alocação pode não ser trivialmente resolvido, uma vez que no caso em que os atuadores do sistema estão sujeitos a restrições (e.g. saturação), o comportamento desejado pode não ser completamente satisfeito. Neste cenário, a alocação de controle consiste em um problema de otimização, que busca a melhor distribuição possível dos controles ao mesmo tempo em que respeita estas restrições. As técnicas de alocação de controle podem ter o objetivo primário de manter a direção do controle ou de minimizar o erro, ou ainda, podem cumprir um objetivo secundário. Para englobar todas estas situações, as técnicas Alocação de Controle Direta, Programação Linear com Simplex, Mínimos Quadrados Ponderados com Conjunto Ativo e Ponto Interior Primal-Dual serão aplicadas em um experimento com um robô móvel do tipo diferencial e em simulações com um Veículo Aéreo Não Tripulado (VANT) do tipo quadricóptero e com um Veículo Submarino Operado Remotamente (ROV). Uma segunda abordagem da alocação de controle tem por objetivo a distribuição de cargas entre os agentes de uma tarefa cooperativa, que nesta dissertação será representada por meio da simulação de um sistema composto por dois manipuladores e um objeto. Todos estes sistemas serão submetidos a trajetórias com diferentes exigências, suas dinâmicas analisadas e seus resultados comparados.

Abstract of Dissertation presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

CONTROL ALLOCATION APPLIED TO ROBOTS SUBJECT TO INPUT CONSTRAINTS

Jhomolos Gomes Alves

September/2019

Advisor: Fernando Cesar Lizarralde

Department: Electrical Engineering

In control systems with multiple inputs and multiple outputs (MIMO), the control signal of a given degree of freedom can be distributed among several actuators by means of a strategy called control allocation. Nonetheless, the control allocation problem may not be trivially solved, since in the case that the system actuators are subject to constraints (e.g. saturation), the desired behavior may not be completely satisfied. In this scenario, the control allocation consists of an optimization problem which searches the best possible control distribution while it satisfies these constraints. The control allocation techniques may have as a primary objective to maintain the control direction or to minimize the error, or may even satisfy a secondary objective. To encompass all of these situations, the techniques Direct Control Allocation, Linear Programming with Simplex, Weighted Least Squares with Active Set and Primal-dual Interior Point are employed in an experiment with a differential drive mobile robot and in simulations of an Unmanned Aerial Vehicle (UAV), such as a quadrotor, and also of a Remotely Operated Underwater Vehicle (ROV). A second approach of the control allocation aims to distribute the load among the agents in a cooperative task, which in this dissertation is represented by means of the simulation of a system composed by two manipulators and an object. All these systems are subject to trajectories with different requirements, their dynamics are analyzed and their results compared.

Contents

List of Figures	xi
List of Tables	xv
List of Symbols	xvi
List of Abbreviations	xix
1 Introduction	1
1.1 Structure of Control Allocation	7
1.2 State of Art	7
1.3 Applications of Control Allocation	10
1.4 Objectives	12
1.5 Methodology	12
1.6 Organization	13
2 Control Allocation	15
2.1 Control Allocation and the Saturation Effect	15
2.2 Optimization Problem	17
2.3 Unconstrained Control Allocation	20
2.4 Constrained Control Allocation	21
2.4.1 Linear Programming	22
2.4.2 Quadratic Programming	23
2.4.3 Common Objectives in Control Allocation	24
2.5 Linear Programming with Simplex	26
2.6 Direct Control Allocation	30
2.7 Weighted Least Squares	30
2.7.1 Weighted Least Squares with Active Set	31
2.7.2 Primal-dual Interior-Point	33
2.8 Error Analysis	35
2.9 Conclusions	36

3	Robots: Modeling and Control	37
3.1	Wheeled Mobile Robots	37
3.1.1	Differential Drive Robot	38
3.1.2	Nonholonomic Constraints	39
3.1.3	Kinematics in Polar Coordinates	39
3.1.4	High-level Controller	41
3.1.5	Control Allocation Problem	43
3.2	Remotely Operated Underwater Vehicle	44
3.2.1	Kinematics	44
3.2.2	Static Model of Propellers	45
3.2.3	Dynamics	46
3.2.4	LUMA	49
3.2.5	Reference Model	49
3.2.6	High-level Controller	50
3.2.7	Control Allocation Problem	51
3.3	Quadrotors	52
3.3.1	Propellers, Forces and Torques	53
3.3.2	Gyroscopic Effects	54
3.3.3	Rotor Dynamics	54
3.3.4	Newton-Euler Equations of a Quadrotor	55
3.3.5	High-level Controller	56
3.3.6	Control Allocation Problem	58
3.4	Robotic Manipulators	58
3.4.1	Forward Kinematics	59
3.4.2	Diferential Kinematics	60
3.4.3	Dynamics	62
3.4.4	Cooperative Manipulators	64
3.4.5	Impedance Control	67
3.4.6	Kinematic Control	69
3.4.7	Static Load Allocation	70
3.4.8	Control Scheme of Cooperative Manipulators	73
3.5	Conclusions	74
4	Experiment Results and Simulations	75
4.1	Overview of the Algorithms	75
4.2	Wheeled Mobile Robot	75
4.3	Remotely Underwater Operated Vehicle	86
4.4	Quadrotor	102
4.5	Cooperative Manipulators	113

4.6	Conclusions	121
5	Discussion and Conclusions	123
5.1	Future works	125
	Bibliography	127
A	Mechanics of Rigid Body	133
A.1	Rigid Body Kinematics	133
A.1.1	Position	134
A.1.2	Orientation and Rotation Matrix	134
A.1.3	Rotation in Exponential Coordinates	136
A.1.4	Euler Angles	137
A.1.5	Angular Velocity Transformation	137
A.1.6	Quaternions	138
A.1.7	Homogeneous Transformation Matrix	139
A.1.8	Velocities of a Rigid Body	140
A.1.9	Quaternion Propagation	141
A.1.10	Generalized Velocities Transformation	141
A.2	Rigid Body Dynamics	142
A.2.1	Center of Mass	142
A.2.2	Kinetic Energy	143
A.2.3	Potential Energy	143
A.2.4	Inertia Matrix	144
A.2.5	Lagrange Equations of Motion	145
A.2.6	Newton-Euler Equations	146
A.2.7	Generalized Forces Transformation	147
A.2.8	Coriolis and Centrifugal Forces	148
A.2.9	Other Dynamic Effects	148
B	Proof of the KKT Conditions	150

List of Figures

1.1	Saturation effect in dynamical systems.	3
1.2	The operating range of the actuators define a polytope region in space \mathbb{R}^m	3
1.3	Typical control allocation structure applied to a mechanical system.	7
1.5	Model of the military transport C-17 has been widely analyzed in control allocation oriented studies ¹	10
2.1	The control effectiveness matrix B executes the mapping between the attainable set of control inputs \mathbb{U} and the attainable set of virtual controls \mathbb{A}	16
2.2	According to the norm adopted for the cost function $f(u)$, the optimization problem admits different solutions. At the left, it is depicted the $l1$ -norm, and at the left, the $l2$. The cost function $f(u)$ is represented by contours in dashed lines.	20
2.3	At each iteration, the Simplex algorithm visits every adjacent vertex to find the optimal feasible basic solution.	28
2.4	The parallelogram represents \mathbb{A} with vertices 1,2,3 and 4, and edges connecting them. The virtual control ν is solution of the vector equation $\nu\hat{\nu}_d + b\nu_{32} = \nu_2$	31
2.5	Interior-point algorithm travels through the feasible polytope towards the optimal solution x^*	33
3.1	Posture of a two-wheeled-mobile robot with respect to the inertial frame \mathcal{F}_o	38
3.2	Conversion from Cartesian to polar coordinates.	40
3.3	View of the LUMA ROV ²	45
3.4	A submerge ROV connected by a umbilical cable to the inertial frame	49
3.5	Top and frontal view of the ROV LUMA with the arrangement of the propellers, placed at R_{pi} with force direction P_i	52
3.6	Representation of a quadrotor along with the forces and torques generated by its propellers.	52

3.7	Free body diagram of a quadrotor, which is represented as a concentrated mass.	55
3.8	Denavit-Parameters at the zero position of the UR5.	60
3.9	Contact forces simulated as a spring-damper model.	64
3.10	Superior and front view of a cooperative system, where the virtual sticks r_i connect the tips of end effectors to the object frame \mathcal{F}_c	66
3.11	Block diagram of the control of cooperative manipulators.	73
4.1	The Roomba 621 manufactured by <i>iRobot</i>	76
4.3	Results when saturation is unmanaged. (\circ initial position; \times desired waypoints; — trajectory traveled; — linear velocity v ; — angular velocity ω ; (a) trajectory 1; (b) linear and angular velocities for trajectory 1; (c) trajectory 2; (d) linear and angular velocities for trajectory 2; (e) trajectory 3; (f) linear and angular velocities for trajectory 3.	80
4.4	Results with the Direct Control Allocation.	81
4.5	Results with the Simplex.	82
4.6	Results with the Weighted Least Squares with Active Set.	83
4.7	Results with the Primal-dual Interior Point.	84
4.8	Wheel velocities in the DCA (plots a, b and c) and Simplex (d, e and f). (Legend: — right wheel velocity v_r ; — left wheel velocity v_l). 85	
4.9	Wheel velocities in the WLSAS (plots a, b and c) and PDIP (d, e and f). (Legend: — right wheel velocity v_r ; — left wheel velocity v_l). 86	
4.10	The desired trajectories for the simulation 1, 2 and 3 originated from the reference trajectory. (— desired trajectory χ_{d1} , — desired trajectory χ_{d2} , — desired trajectory χ_{d3}).	89
4.11	The dynamics of the ROV when subject to saturation.(— results for χ_{d1} , — results for χ_{d1} , — results for χ_{d3}).	90
4.12	The dynamics of LUMA with the Direct Control Allocation algorithm. (— results for χ_{d1} , — results for χ_{d2} , — results for χ_{d3}).	91
4.13	The resulting dynamics of LUMA obtained with the Simplex algorithm. (— results for χ_{d1} , — results for χ_{d2} , — results for χ_{d3}).	93
4.14	Results for the Primal-dual Interior Point algorithm with $\gamma = 1000$ (— results for χ_{d1} , — results for χ_{d2} , — results for χ_{d3}). 94	
4.15	Results the Primal-dual Interior Point algorithm with $\gamma = 10$ (— results for χ_{d1} , — results for χ_{d2} , — results for χ_{d3}).	95

4.16	Forces produced by each propeller with $\gamma = 10$ and $\gamma = 10^3$ in the PDIP algorithm for the desired trajectory χ_{d2} (— $\gamma = 10$, — $\gamma = 10^3$).	96
4.17	The dynamics of LUMA with the Weighted Least Squares with Active Set algorithm. (— results for χ_{d1} , — results for χ_{d2} , — results for χ_{d3}).	97
4.18	Control inputs for the desired trajectory χ_{d1} with $\gamma = 10^3$ (— DCA, — Simplex, — WLSSA, — PDIP).	98
4.19	Control inputs for the desired trajectory χ_{d2} with $\gamma = 10^3$ (— DCA, — Simplex, — WLSSA, — PDIP).	99
4.20	Control inputs for the desired trajectory χ_{d3} with $\gamma = 10^3$ (— DCA, — Simplex, — WLSSA, — PDIP).	100
4.21	The reference signal $r(t)$ and the three desired trajectories for the quadrotor. (-.-.- reference signal χ_r , — desired trajectory $\chi_{d,1}$, — desired trajectory $\chi_{d,2}$, — desired trajectory $\chi_{d,3}$). .	104
4.22	Dynamics of the quadrotor when saturation is not managed by any control allocation technique. (-.-.- reference χ_r , — trajectory traveled χ_1 , — trajectory traveled χ_2 , trajectory traveled χ_3 —).	106
4.23	Results for the Direct Control Allocation (-.-.- reference χ_r , — trajectory traveled χ_1 , — trajectory traveled χ_2 , — trajectory traveled χ_3).	107
4.24	Virtual controls for the DCA (— trajectory traveled χ_1 , — trajectory traveled χ_2 , — trajectory traveled χ_3).	108
4.25	Results for the linear programming with Simplex (-.-.- reference χ_r , — trajectory traveled χ_1 , — trajectory traveled χ_2 , — trajectory traveled χ_3).	109
4.26	Virtual controls for the linear programming with Simplex (— trajectory traveled χ_1 , — trajectory traveled χ_2 , — trajectory traveled χ_3).	110
4.27	Results for the WLSIP and PDIP (-.-.- desired trajectory $\chi_{d,1}$, — WLSAS with $W_v = \text{diag}(1, 1, 1, 1)$, — WLSAS with $W_v = \text{diag}(5, 2, 2, 2)$, — WLSAS with $W_v = \text{diag}(50, 10, 10, 50)$, — PDIP with $\gamma = 10^4$).	111
4.28	Dynamics of the virtual controls for the WLSAS and PDIP (-.-.- desired trajectory $\chi_{d,1}$, — WLSAS with $W_v = \text{diag}(1, 1, 1, 1)$, — WLSAS with $W_v = \text{diag}(5, 2, 2, 2)$, — WLSAS with $W_v = \text{diag}(50, 10, 10, 50)$, — PDIP with $\gamma = 10^4$).	112
4.29	The UR5 manipulator made by Universal Robots.	114

4.30	Two UR5 manipulators in a cooperative task.	115
4.31	Trajectories of the object and of the manipulators UR5 ₁ and UR5 ₂ with Static Load Allocation (\dashdot desired trajectory $\chi_{d,1}$, $-\ -$ desired trajectory $\chi_{d,2}$, $\cdots\cdots$ desired trajectory $\chi_{d,3}$, --- trajectory χ_1 , --- trajectory χ_2 , --- trajectory χ_3).	118
4.32	Forces acting on the object by the end effectors and the load allocation according to the sharing policy from the Static Load Allocation (--- trajectory χ_1 , --- trajectory χ_2 , --- trajectory χ_3).	119
4.33	The internal force $h_{I,x}$ and the internal torques $h_{I,\phi}$, $h_{I,\theta}$, $h_{I,\psi}$. (--- trajectory χ_1 , --- trajectory χ_2 , --- trajectory χ_3).	120
A.1	Representation of a rigid body with body frame \mathcal{F}_b located at ${}^a p_b$	133
A.2	Rotation of body frame \mathcal{F}_b with respect to \mathcal{F}_a	135
A.3	A point p rotating about the axis ω	136
A.4	Two frames \mathcal{F}_b and \mathcal{F}_c in a rigid frame with respect to \mathcal{F}_a	141
A.5	System mass-spring under a force F , which provokes a deformation Δx in the spring and energy storage.	144
A.6	When there is a rotational motion ω , it appears Centrifugal and Coriolis forces, which changes the body acceleration.	148

List of Tables

1.1	Common strategies for solving input saturation in dynamic systems.	4
4.1	The control allocation algorithms and their main characteristics.	76
4.2	Parameters of the <i>Roomba</i> robot.	78
4.3	Parameters of the control allocation techniques.	79
4.4	Virtual control errors (VCE) and Distance Error (DE) for the trajectory 1 in <i>Roomba</i>	84
4.5	VCE and DE for the trajectory 2 in <i>Roomba</i>	85
4.6	VCE and DE for the trajectory 3 in <i>Roomba</i>	85
4.7	Position (R_{pi}) and orientation (P_i) of the propellers.	87
4.8	Physical parameters of the ROV, for describing the undersea environment and the dynamics of the propellers.	88
4.9	Gains of the PD Controller and the parameters of the control allocation algorithms.	88
4.10	Virtual control errors (VCE) and Distance Error (DE) in the ROV LUMA when $\chi_{d,1}$	101
4.11	VCE and DE of the CA algorithms in the ROV LUMA when $\chi_{d,2}$	101
4.12	VCE and DE of the CA algorithms in the ROV LUMA when $\chi_{d,3}$	102
4.13	Parameters of the quadrotor.	102
4.14	Gains of the Integral Backstepping and Proportional-Derivative Controllers.	105
4.15	VCE and DE of the CA algorithms with the desired trajectory $\chi_{d,1}$ in the quadrotor	113
4.16	VCE and DE of the CA algorithms with $\chi_{d,2}$ in the quadrotor	113
4.17	VCE and DE of the CA algorithms with $\chi_{d,3}$ in the quadrotor	113
4.18	Denavit-Hartenberg Parameters of the UR5 manipulator.	114
4.19	Parameters of the object manipulated.	115
4.20	Parameters for modeling the contact forces and the desired object dynamics for the impedance controller.	116
4.21	Parameters adopted in the CLIK algorithm and PID Position Controller.	116

List of Symbols

B	Control Effectiveness Matrix, p. 16
G	Grasp matrix, p. 66
I	Inertia tensor, p. 144
K_m	DC gain of motor, p. 54
M	Mass of a rigid body or particle, p. 142
R	Rotation Matrix, p. 135
R_r	Rotation matrix from \mathcal{F}_o to \mathcal{F}_c , p. 46
$SE(n)$	n-dimensional special euclidean group, p. 139
$SO(n)$	n-dimensional special orthornormal group, p. 135
T	Transformation Matrix, p. 139
T_z	Thrust, p. 53
W_u	Weighting matrix of the control input vector, p. 21
W_v	Weight matrix of virtual control vector, p. 26
χ	Generalized coordinates, p. 38
$\dot{\chi}$	Generalized velocities, p. 38
ϵ	Error tolerance, p. 35
κ_d	Rotor drag coefficient, p. 53
κ_t	Rotor thrust coefficient, p. 53
λ	Lagrange multipliers vector, p. 22
\mathbb{A}	Attainable set of virtual control inputs, p. 3

\mathbb{U}	Attainable set of control inputs, p. 3
\mathcal{E}	Index set related to equality constraints, p. 18
\mathcal{F}_c	Object Orthonormal Cartesian Frame, p. 37
\mathcal{F}_o	Inertial Orthonormal Cartesian Frame, p. 37
\mathcal{G}	Index set related to inequality constraints, p. 18
\mathcal{I}	Identity matrix, p. 32
\mathcal{Q}	Quaternion representation, p. 138
Q_U	Unit quaternion, p. 139
\mathcal{T}	Kinetic Energy, p. 146
\mathcal{U}	Potential energy, p. 144
ν	Virtual control input vector, p. 5
ν_d	Desired virtual control input vector, p. 5
ω	Body angular velocity, p. 39
ω_R	Residual angular velocity of the rotors, p. 54
ω_i	rotation speed of the i -th propeller, p. 46
τ_m	time constant of motor, p. 54
h_f	Contact force, p. 63
k_{fi}	Reference model parameters, p. 50
m	Number of actuators, p. 5
n	Number of DoF of the system, p. 5
o_f	Control input offset, p. 21
r_d	Half the distance between wheels, p. 40
s	Dual slack variable, p. 22
u	Control input vector, p. 1
u_d	Desired control input vector, p. 17

u_{max}	Maximum control input vector, p. 2
u_{min}	Minimum control input vector, p. 2
v	Linear velocity, p. 39
w	Slack variable, p. 23

List of Abbreviations

DCA	Direct Control Allocation, p. 13
DE	Direction Error, p. 36
DoF	Degrees of Freedom, p. 5
GSCAR	Group of Simulation, Control and Automation in Robotics, p. 44
HARV	High Alpha Research Vehicle, p. 8
IB	Integral Backstepping, p. 104
KKT	Karush-Kuhn-Tucker conditions, p. 35
LP	Linear Programming, p. 23
MIMO	Multiple input-multiple output, p. 15
NASA	National Aeronautics and Space Administration, p. 8
PDIP	Primal-dual Interior Point, p. 13
QP	Quadratic Programming, p. 35
ROV	Remotely Operated Underwater Vehicle, p. 44
SLA	Static Load Allocation, p. 70
UAV	Unmanned Aerial Vehicle, p. 11
UFRJ	Federal University of Rio de Janeiro, p. 44
VANT	<i>Veículo Aéreo Não Tripulado</i> - Unmanned Aerial Vehicle, p. vi
VCE	Virtual Control Error, p. 36
WLSAS	Weighted Least Squares with Active Set, p. 13
WLS	Weighted Least Squares, p. 30

Chapter 1

Introduction

Physical phenomena present in nature consist inherently of nonlinear systems, which can be described as systems that do not follow the superposition and homogeneity properties (CHEN, 2009). Nonetheless, this behavior is also found in every “motion generator” called actuators, that are mechanical devices responsible for providing motion and also for controlling the system they belong. Their inputs consist of an energy source and a control signal, and their outputs are force and torque, which are delivered to effectors - devices designed for interacting with the environment. Typical examples of actuators are motors and engines, and of effectors, propellers, wheels and end effectors.

Concerning its constructive aspects and physical constraints, actuators present nonlinearities, such as friction, hysteresis, dead-zone, backlash and saturation. More details concerning non-linear systems can be found in SLOTINE *et al.* (1991).

In KALMAN (1955), an important work dated back to 1955, the author states that

“(...) the engineer (...) wants to understand fully the effects on system performance of the various inevitable physical limitations and imperfection of practical equipment. The most obvious limitation may be termed ‘saturation’. It is everywhere present.”

The limitations and imperfections mentioned by the author imply that there are conditions where the superposition and homogeneity properties do not hold. In fact, despite the actuators can operate at a linear region, their range is bounded by upper and lower limits. Whenever the actuator reaches any of these limits, it is said to be saturated, since any attempt to surpass them does not result in a corresponding variation in the actual control input.

Whenever a control input signal $u \in \mathbb{R}$ is applied to an actuator, its actual position δ is driven towards u , according to the actuator dynamics. However, δ is constrained regardless of the value of u , such that

$$\delta_{min} \leq \delta \leq \delta_{max} \quad (1.1)$$

where δ_{min} and δ_{max} are the lower and upper actuator position, respectively.

One can consider also the rate limits in the actuator dynamics. In practical situations, the demand of a control input u cannot be instantaneously delivered due to a dynamic constraint OPPENHEIMER *et al.* (2010), represented by the rate limit $\dot{\delta}$.

Remark. *In this dissertation, we assume that the actuators present a response fast enough to be neglected, such that*

$$\begin{aligned} u &\triangleq \delta \\ u_{min} &\triangleq \delta_{min} \\ u_{max} &\triangleq \delta_{max} \end{aligned} \quad (1.2)$$

unless stated otherwise.

The diagram block in the figure 1.1 illustrates the saturation in actuators. Whenever the control input u surpasses u_{max} or reach values below u_{min} , the output does not respect the linearity property, rather it remains constrained to these limits. Mathematically, saturation $\text{sat}(u)$ can be described by the function

$$\text{sat}(u) = \begin{cases} u_{min}, & \text{if } u < u_{min} \\ u, & \text{if } u_{min} \leq u \leq u_{max} \\ u_{max}, & \text{if } u > u_{max} \end{cases} \quad (1.3)$$

In a dynamic system with m independent actuators and control input $u \in \mathbb{R}^m$, the function $\text{sat}(u)$ is represented by a vector in the form

$$\text{sat}(u) = \left[\text{sat}(u_1) \quad \text{sat}(u_2) \quad \dots \quad \text{sat}(u_m) \right] \quad (1.4)$$

This function allows to define the attainable set of control inputs \mathbb{U} formed by all the m actuators. In DURHAM (1993), the subset \mathbb{U} is mathematically defined as

$$\mathbb{U} = \{u \in \mathbb{R}^m \mid u_{i,min} \leq u_i \leq u_{i,max}\} \subset \mathbb{R}^m \quad (1.5)$$

Geometrically, \mathbb{U} represents a polytope in the space \mathbb{R}^m , as depicted in figure 1.2. If the control input is located inside the polytope or at its border, it is said to be attainable or feasible. Otherwise, if located outside \mathbb{U} , it is unattainable or

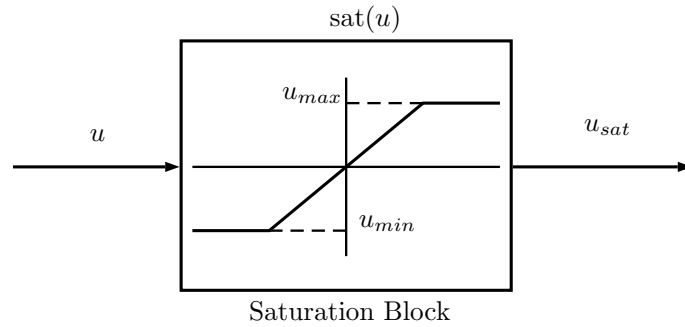


Figure 1.1: Saturation effect in dynamical systems.

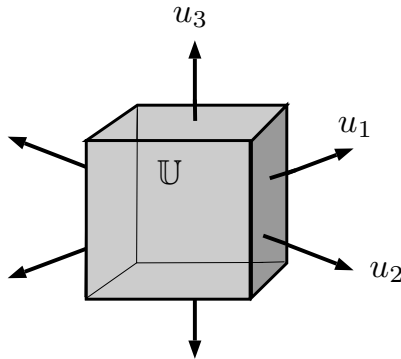


Figure 1.2: The operating range of the actuators define a polytope region in space \mathbb{R}^m .

unfeasible. However, this concept goes beyond saturation and can be expanded to any input constraint.

Many controllers and model plants can be approximated by linear models or linearized at the vicinity of an equilibrium point. However, external factors such as disturbances or set point changes can cause dynamic systems to produce large errors, to generate efforts beyond the capabilities of the actuators or even to prevent them to recover themselves back to nominal operating conditions. In 2007, a tragical accident involving two HAL Dhruv helicopters of the Indian Air Force was caused due to a phenomenon called cyclic saturation. It occurs when the system does not respond to cyclic inputs, then the lateral control for rolling moment runs out of limit (DEFENCE, 2010).

Another common saturation issue is called “wind-up”, that occurs when the control variable value reaches its limit at the presence of a PI/PID controller, because the actuator remains at its limit independently from the system output. Then it is necessary to force the controller to actuate in the linear region, which can be implemented with techniques such as back-calculation or conditional integration (GARCIA and CASTELO, 2002). The wind-up is even worsened in the presence of unstable eigenvalues in the compensator. Since wind-up is related to the dynamics of a controller, it is often referred as controller wind-up. One anti wind-up scheme employed is the observer technique (HIPPE, 2006), where the controller dynamics

Strategies
Anti-windup Design
Optimal Control
Robust Control
Adaptive Control
Predictive Control
Domain of Attraction
Sliding-mode Control

Table 1.1: Common strategies for solving input saturation in dynamic systems.

does not influence input saturation anymore, rather influences the plant. The author implements additional dynamic elements, such as filtered setpoint or additional dynamic network to prevent saturation.

In CHEN (2014), it was proposed a saturated controller combined with a dynamic time-varying controller with slope restrictions, obtained by applying the finite-time control strategy. In HUANG *et al.* (2018), a robust neural network-based control scheme is utilized to perform stabilizing, tracking, and as consequence, to solve input saturation in wheeled mobile robots. Another approach is presented in HUANG *et al.* (2013), where the authors utilize two adaptive controllers, kinematic and dynamic, respectively, whose design parameters are computed in advance in order to prevent input saturation.

Shortly, the approaches mostly utilized for solving input saturation are chronologically summarized in BERNSTEIN and MICHEL (1995), and thereafter, some other approaches have been employed. They are briefly presented in the table 1.1.

However, these controls techniques have been utilized mostly as a single strategy to deal with input saturation. Nonetheless, another approach is to combine them with a technique called *control allocation*.

Definition 1.1. *Control allocation consists of distributing control efforts among the actuators that contribute to motion concerning one or more degrees of freedom of interest for allocation purposes, so that the actuators can reproduce the moments demanded.*

A general control allocation problem can be stated as: given a set of desired moments and forces ν_d , generally denoted as desired virtual control vector, to be exerted on a system, distribute these efforts among every actuator, such that ν_d is achieved. There may exist infinite solutions for this problem - for instance, consider an object with mass $M = 1$ kg moving along a single coordinate axis, namely a task with one degree of freedom (DoF). Consider now that there are two actuators delivering forces u_1 and u_2 to the their respective effectors, responsible for providing acceleration to the object. The system object-actuators is described by the dynamics

$$\begin{cases} \dot{x} = \nu \\ \nu = u_1 + u_2 \end{cases} \quad (1.6)$$

where x is the object velocity. In this model, the dissipative forces are not considered. The object may be demanded to travel at an acceleration $\nu_d = 2\text{m/s}^2$. Hence, there are infinite combinations of forces $u = \begin{bmatrix} u_1 & u_2 \end{bmatrix}^T$ that meet this requirement, such as $u = \begin{bmatrix} 1.1 & 0.9 \end{bmatrix}^T$ or $u = \begin{bmatrix} 1.5 & 0.5 \end{bmatrix}^T$.

Now, suppose that the actuators have equal characteristics and are able to exert forces only between the limits $u_{min} = 0$ and $u_{max} = 1.2$. Forces such as $u_1 = 1.5$ and $u_2 = 0.5$ would not be an efficient solution, provided that due to saturation, $u_{sat} = \begin{bmatrix} 1.2 & 0.5 \end{bmatrix}^T$, and as consequence, $\nu = 1.7 < \nu_d$. However, despite of saturation, it is possible to find a control u that best satisfies the desired virtual control input ν_d , for instance $u = \begin{bmatrix} 1 & 1 \end{bmatrix}$ or $u = \begin{bmatrix} 1.1 & 0.9 \end{bmatrix}^T$. Note that the control was distributed wisely respecting the constraints of each actuator with the objective of satisfying the desired moment ν_d . This is a typical example of a control allocation problem, although these solutions may not be optimal.

While this example is quite simple, the problem gets more complex as the number of actuators m and the number of DoF of the system n increase, as well as more constraints are imposed. However, if there existed only one effector exerting a force on the object, the solution would be unique and straightforward, and not to mention, the control requirements could not be met at all and its overall performance would consequently be decayed. It can be concluded that systems with more independent actuators than the strictly necessary to perform a given task, that is $m > n$ and denoted overactuated, provide a wider range of options when allocating controls.

Thus, the level of redundancy of a system in the control allocation context allows to classify them into three categories:

- Overactuated: systems with more actuators than those strictly needed for the DoF of interest, i.e $n > m$;
- Fully actuated: systems where the amount of actuators is exactly the amount of degrees of freedom of interest, i.e $n = m$;
- Underactuated: systems with less actuators than those necessary to control the degrees of freedom of interest, i.e $n < m$.

It must be considered that the layout of actuators and their operating principles exert influence on the level of redundancy. If we purely observe the total amount of actuators and the amount of DoF to be controlled, the analysis may lead to

wrong conclusions, provided that multiple actuators can contribute to motion with respect to a unique or several DoF, whereas another DoF may be controlled by only a single actuator. In the last case, the control allocation is not suitable, since it is not possible to distribute the controls.

The development of overactuated systems has been necessary to provide reliability and to make them more controllable, and as it went on, research has focused in making better use of this potentiality. For instance, the controllability of the chosen states and outputs may be achieved with less control inputs JOHANSEN and FOSSEN (2013). Moreover, the control allocation may bring the following advantages:

- to promote the control redistribution to actuators respecting input constraints;
- to provide fault tolerance and reconfiguration requirements: the actuators can be rearranged to cancel the effects of an actuator that undergoes a fault, and hence, the system can be recovered back to its operational conditions when physically possible (OPPENHEIMER *et al.*, 2010);
- multiple control input choices available can satisfy secondary control objectives: one may prefer one actuator over another one, whether to reduce energy consumption or to grant better stability. An actuator can also be demanded to work at the neighborhood of a preferred position to minimize the control or to avoid stress in its structure. From these possibilities, one can achieve accuracy, better response, flexibility, ease of maintenance and power efficiency;
- enables the system to share actuators among different control systems (JOHANSEN and FOSSEN, 2013).

Nonetheless, when an actuator suffers failure and the system becomes fully actuated or underactuated, it implies that redundancy has been lost and the signal distribution is limited to a best possible solution, and therefore, the system cannot benefit from the most advantages aforementioned. Now the constraints in the actuators play an even more important role - the systems gets more susceptible to the constraints, provided that in order to deliver the desired moments, the actuators become more demanded and are more likely to work near saturation levels.

It must be considered that control allocation does not apply only to control motion, rather it can be utilized also to represent quantities like energy and mass (JOHANSEN and FOSSEN, 2013). In OPPENHEIMER *et al.* (2010), the authors summarize the objectives of control allocation so as to determine a unique solution when multiple solutions exist, to obey physical constraints of the control effectors and to determine the “best” configuration of control settings when no solution exists. These objectives are concise but extensive - they cover every kind of system, from

underactuated to overactuated, and shed light on how flexible control allocation can be, enabling its application to deal with any input constraint.

1.1 Structure of Control Allocation

The project of control algorithms for control allocation may be basically split into two levels, as shown in 1.3. In the first level, a High-level control is performed, which consists of any standard control technique and uses the actual state χ and the desired state χ_d to determine the desired virtual control input $\nu_d \in \mathbb{R}^n$. The vector ν_d contains all the n demanded forces and moments to be applied on the system for providing motion and such that the basic requirement of controllability is met (JOHANSEN and FOSSEN, 2013). If the actuators are capable of imposing a $\nu = \nu_d$ to the system, then it is said that the control objective has been reached in the first level.

In the literature, one can find control allocation schemes where the High-level controller consists of PI Controller with Feedforward (BUFFINGTON and BUFFINGTON, 1997), Lyapunov (JOHANSEN, 2004), Sliding-mode Controller (CHEN and WANG, 2011), Backstepping (MONTEIRO, 2015), among others.

In the second level, the control allocation maps the desired input vector ν_d into individual forces and torques $u \in \mathbb{U}$ such that the sum of all forces and torques on the effectors inflict the virtual control input $\nu = \nu_d$ to the mechanical system in every instant t .

Although not mandatory, there may exist also third level, which consists of a low-level controller in each effector, responsible for assuring that actuators deliver the desired force and moment (JOHANSEN and FOSSEN, 2013).

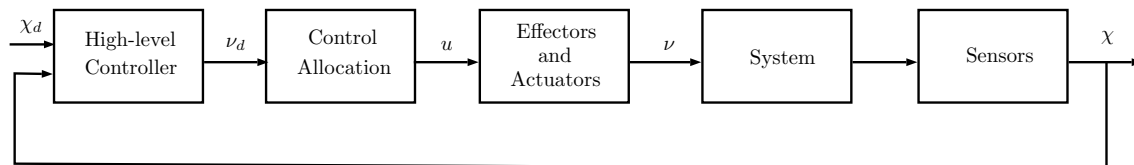


Figure 1.3: Typical control allocation structure applied to a mechanical system.

1.2 State of Art

The work published by DURHAM (1993) is a watershed concerning control allocation, because prior to it, the term had never been defined, although it had been implemented informally. Formerly it was common to control aircrafts by utilizing single controls for each of its degree of freedom, namely roll, pitch and yaw. Rolling

moments are controlled by ailerons, elevators control pitch, and yaw is controlled by rudders. However, there existed a coupling between the rolling and yawing moments, which was solved by means of a mechanical aileron-rudder interconnect (ARI). Later, the rolling surface-rudder interconnect (RSRI) was adopted to solve coupling in the F-18 aircraft. These devices were designed to generate a rolling moment without yaw, and as a result, opposing control surfaces were set free.

In the history of control allocation, the NASA F-18 High Alpha Research Vehicle (HARV), depicted in 1.4, plays an important role. It was utilized in the NASA's Dryden Flight Research Center in Edwards, CA and consists of a highly modified version of the American McDonnell Douglas F/A-18 Hornet. The aircraft has been extensively used in a three-phase investigation program, which aimed to study controlled flight at elevated angle of attack (high- α) by using thrust vectoring, actuated forebody strakes and modifications to its flight controls.

Every phase corresponded to a single aircraft configuration. In the first one, the F-18 HARV performed precisely 101 research flights with angles of attack as high as 55 degree from April 1987 to 1989. It aimed to collect data concerning aerodynamic measures and to develop the flight research techniques needed for these purposes. The aircraft suffered no modification at all, except for extensive instrumentation. The second phase was the thrust-vectoring phase, in which the vectored thrust was examined in order to achieve greater maneuverability and control at these high- α . Hardware and software modifications were carried out to provide pitch and yaw forces to improve maneuverability. This phase started in July 1991 and completed in January 1993, and as a result, the high- α was increased up to 70 degrees. Finally, the third phase used a modified forebody with a deployable nose strakes (BOWERS *et al.*, 1996) to control yaw at high angles. At this situation, the rudders become ineffective. The strakes are designed for controlling the aircraft at the presence of large side forces generated by vortices. The project was concluded in September 1996.

The aircraft is composed by the many moment generators, such as horizontal tail, right and left aileron, leading-edge flap, trailing-edge flap, left and right rudder, left and right strakes, left and right stabilators, and yaw and pitch thrust-vectoring moment generators. Thus, the control system is highly overactuated and the attitude control of the aircraft becomes a complex task. Concerning control purposes, NASA released a report (MORELLI, 1995) where parameter identification techniques were applied for optimal input design validation at 5 degrees of angle of attack, individual strake effectiveness at 40 and 50 degrees, and lateral dynamics and lateral control effectiveness at these angles.

A large amount of data has been collected and used in simulations in many researches concerning control allocation. In DURHAM (1993), the author presents



Figure 1.4: The F-18 aircraft was widely studied at NASA for achieving an angle of attack as high as 70 degrees¹.

the Direct Control Allocation technique, which consists of finding the input signals for each actuator, such that the control law demands are met and the control direction is maintained in overactuated systems. In the same work, the DCA technique is then compared with Daisy Chaining, another control allocation strategy used for splitting controls into two groups for controlling aerodynamic and thrust vectoring. Finally, the author compared the DCA with the generalized pseudoinverses. The aircraft model was also analyzed in BODSON and FROST (2011), where the authors implemented three different control allocation algorithms for the roll command and their resulting contributions from aileron and horizontal tail. In DURHAM and BORDIGNON (1996), the authors addressed the control allocation problem to the HARV, where the control effectiveness matrix and control position limits were based on a F-18 aircraft flying at 10,000 feet, Mach 0.23 and 30 degrees angle of attack. They analyzed how a control allocation algorithm can be chosen, if considering individual control effector rate demands. Traditional control allocation techniques are revisited and a new allocation scheme is presented, called moment-rate allocation.

An airplane model commonly found in the literature is based on the C-17 military transport airplane, as depicted in figure 1.5. Its name is Boeing C-17 Globemaster III, developed by McDonnell Douglas at the 80es for the United States Air Force. In PETERSEN and BODSON (2005), this model is revisited and the control allocation problem is solved by means of interior-point algorithms formulated as linear programming problems. In BODSON (2002), the author utilizes the C-17

¹Extracted from <https://www.dfrc.nasa.gov/Gallery/Photo/F-18HARV/HTML/EC94-42645-9.html>



Figure 1.5: Model of the military transport C-17 has been widely analyzed in control allocation oriented studies².

model with 8 actuators with five different control allocation strategies, which are Pseudo-inverse, Quadratic Programming, Fixed-point Method, DCA and a mixed optimization problem converted to linear programming solved with Simplex.

1.3 Applications of Control Allocation

Although the studies concerning control allocation have extensively focused in airplanes and aircrafts, it has been extended to other areas as well. In JOHANSEN and FOSSEN (2013), the authors make a detailed research covering this topic, which is summarized as follows:

- in offshore vessels in low and high speed maneuvering and repositioning by using different propellers types, which generate a desired yaw when combined, and also in ship autopilots;
- maneuvering of underwater vehicles, usually overactuated, to prevent failure and to counteract the mispositioning due to forces from the environment;
- in ground vehicles, by controlling yaw and providing stability to prevent accidents due to slippery surfaces and high speeds, by combining individual tire breaks to producing a desired yaw to counteract and correct the skidding;
- by means of electrical propulsion to coordinate individual combinations of motors to optimize power efficiency;
- incorporating roll moment with yaw moment allocation, by using brake and steering actuators to prevent vehicle rollover;

²Extracted from <https://www.af.mil/About-Us/Fact-Sheets/Display/Article/1529726/c-17-globemaster-iii/>

- in off-road vehicles to distribute traction control;
- in strategies to wheeled and legged mobile robots with active suspension;
- in legged mobile robots, since they demand a periodic and coordinated movement of each leg while distributing the force. It takes into account energy consumption and friction between legs and ground.

The offshore vessel is a good example to explain how complex the modern control/mechanical systems have become. The control allocation has been used to allocate efforts to thrusts for controlling its three degrees of freedom, namely surge, sway and yaw, in a low-speed maneuvering, when the ship dynamically positions itself during operations that demand a *quasi* static position. The thrust system is composed by the main propellers, tunnel thrusters, azimuth thrusters and water jets, used to control the longitudinal direction of the ship and its degrees of freedom as well. Because of safety and operational concernings, these systems are overactuated, so that a single failure will not interrupt a related oil extraction operation, which are always expensive and long-lasting. The vessels are designed to withstand external forces and disturbances, such as wind, waves and sea currents. The yaw control has higher priority though, since it would imply in loss of heading, and hence, loss of position due to bad weather and sea conditions.

Another scheme is presented in LAWITZKY *et al.* (2010), where the authors propose a load allocation scheme between human and robot. Based on a desired effort to be applied on the manipulated object, sharing policies are established according to the role of the agent in the task. In BAIS *et al.* (2015), the authors propose a static and a dynamic load allocation scheme for cooperative tasks and analyze the internal wrenches that may arise due to the load distribution.

In MONTEIRO *et al.* (2016a) and MONTEIRO *et al.* (2016b), control allocation techniques are presented and implemented in a quadrotor to demonstrate how they can solve actuator saturation can be solved, even though it is an underactuated system. The addressed control allocation strategies eliminate the need of optimization algorithms and reduce computational costs.

From all these examples, it can be observed that control allocation has been applied to solve different issues in a large number of systems, mostly concerning overactuated systems. As technology improves and new challenges arise, researchers concentrate their efforts to find new applications for the control allocation techniques and to improve them as well.

1.4 Objectives

This work aims to present the control allocation applied to manage input constraints in four different types of robots. The input constraints can stand for saturation in an actuators level or for load sharing in a cooperative task.

Although the control allocation theory has been spread and is well established, the literature still lacks of applications concerning the robots here addressed, such as the differential drive robots and quadrotors, which are underactuated systems, and the ROV LUMA, which consists of an overactuated system with respect to the planar motion. Therefore this work aims to contribute with more applications of the well-established control allocation algorithms.

Moreover, the specific objectives of this work are:

- to subject the robots to demanding desired trajectories in order to enforce one or more actuators to work saturated;
- to propose control allocation algorithms to optimize the control distribution among all the actuators and hence to manage saturation properly;
- to perform numerical simulations and experiments to verify the impact of each control allocation algorithm on the dynamics of the robots;
- to compare the results obtained from the control allocation algorithms by means of metrics designed to calculate the virtual control error and the direction error;
- to evaluate how another control allocation approach, called load allocation, between two agents in a cooperative task can impact the internal forces and the resulting trajectory in an impedance-admittance relationship among the end effectors and an object;
- to compare the results obtained in the cooperative task for different trajectories, which impose different control requirements to be satisfied by both manipulators.

1.5 Methodology

This text addresses the control allocation to four different robots:

- Wheeled mobile robot;
- Remotely Operated Underwater Vehicle (ROV);

- Quadrotor (Unmanned Aerial Vehicle)
- Cooperative manipulators.

The criteria adopted to choose these robots was to include the most common and widely known types. They also contain some interesting characteristics, such as a nonholonomic constraint in the wheeled mobile robot, the quadrotor is underactuated and the ROV addressed is overactuated. These robots also consider different working environments, such as the ground, the air, underwater and also in a cooperative task.

To solve the input constraints to which they are subject, the control allocation techniques employed along this dissertation are

- Direct Control Allocation (DCA);
- Weighted Least Squares with Active Set (WLSAS);
- Linear Programming with Simplex;
- Primal-dual Interior-point (PDIP);
- Static Load Allocation (SLA).

These algorithms encompass the most common control allocation objectives. The DCA is concerned with maintaining the control direction, whereas the WLSAS, the Linear Programming with Simplex and the PDIP have as a primary objective to minimize the control error, although the WLSAS and the PDIP can also satisfy a secondary objective. Finally, the SLA promotes a sharing policy among the agents in a cooperative task. Then, the robots are required to follow trajectories or to perform a given task in a closed-loop dynamics in order to satisfy any of these objectives.

1.6 Organization

The dissertation is organized as follows:

- **Chapter 2:** presents the formulations of the control allocation strategies addressed, their main characteristics and their control objectives, along with their mathematical formulations;
- **Chapter 3:** the robots utilized are presented, as well their kinematics and dynamic equations, the High-level controller and also the control allocation problem;

- **Chapter 4:** contains experiment results for a wheeled mobile robot and simulations for the ROV, quadrotor and cooperative manipulators;
- **Chapter 5:** some topics observed the development of this work and some ideas for future studies are highlighted.
- **Appendix A:** presents additional information concerning the mechanics of the rigid body.
- **Appendix B:** presents the proof of the Karush-Kuhn-Tucker theorem.

Chapter 2

Control Allocation

This chapter presents the main aspects of control allocation, such as its concepts and objectives, and presents the algorithms employed in this dissertation. In terms of input constraints, the control allocation can be split into two groups: constrained and unconstrained. If unconstrained, the control inputs are not subject to nonlinearities and thus they must match the virtual control input. An example is the pseudoinverse, that is commonly applied to the load allocation problem. On the other hand, the constrained control allocation considers the system limitations and tries to utilize all of its capabilities to deliver exactly the control requirements, or when not possible, a control input that best approximates them, according to an optimization problem and an allocation objective.

All the techniques addressed in this chapter belongs to this class, which are the Direct Control Allocation (DCA), the Linear Programming with Simplex, the Weighted Least Squares with Active Set (WLSSA) and the Primal-dual Interior Point (PDIP). The load allocation strategy, called Static Load Allocation (SLA), will be presented in the next chapter, because it is necessary to present some concepts about cooperative tasks prior to presenting the strategy.

2.1 Control Allocation and the Saturation Effect

Consider a general nonlinear mechanical system with multiple input-multiple output (MIMO), described by the following dynamic model:

$$\begin{cases} \dot{x} = f(x, t) + g(x, t)\nu \\ y = l(x, t) \end{cases} \quad (2.1)$$

where $x \in \mathbb{R}^n$ is the vector of state variables, $y \in \mathbb{R}^m$ is the vector of outputs to be controlled and the virtual control vector $\nu \in \mathbb{R}^n$ corresponds to desired quantities, such as forces and torques.

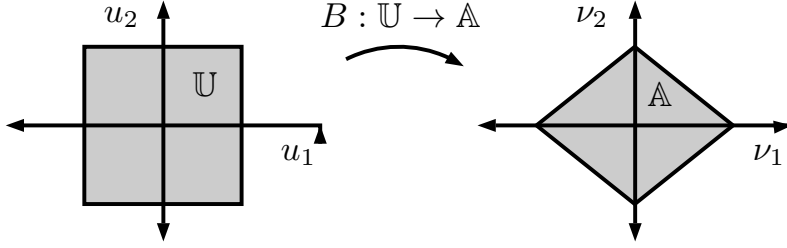


Figure 2.1: The control effectiveness matrix B executes the mapping between the attainable set of control inputs \mathbb{U} and the attainable set of virtual controls \mathbb{A} .

The term $f(x, t)$ stands for the contribution of the actual states to the system dynamics, whereas $g(x, t)\nu$ is the contribution due to the controls. Thus, to change the dynamics of the system, it is needed to impose forces and torques to it according to the high-level controller output, here denoted virtual control input vector ν_d . However, ν is not an input directly applied to the system, rather the actuators exert forces and moments, denoted control input vector u , which contribute to motion respect the degrees of freedom of the system, and hence to ν , such that

$$\nu = h(x, t, u) \quad (2.2)$$

where $h(x, t, u)$ is the static effector model (JOHANSEN and FOSSEN, 2013). However, the effectors models can be regarded as linear about u , which results in the model

$$\nu = B(x, t)u \quad (2.3)$$

where $B(x, t)$ is the control effectiveness matrix that maps the attainable controls onto a n -dimensional attainable set of virtual control inputs \mathbb{A} , also known as attainable moment set. A general example of this mapping is presented in figure 2.1 and can be mathematically defined as

$$\mathbb{A} = \{\nu \in \mathbb{R}^n | Bu = \nu, u \in \mathbb{U}\} \subset \mathbb{R}^n \quad (2.4)$$

In overactuated systems, a necessary condition is that $B(x, t)$ is a "large" matrix and a necessary and sufficient condition is that the number of linearly independent columns is greater than of rows (OPPENHEIMER *et al.*, 2010). If $B(x, t)$ is square and invertible, the system is fully actuated and the control input vector u for the system can be obtained by merely performing the inverse mapping of (2.3).

Remark. *The control effectiveness matrix often presents constant elements. Thus, from now on, it will be represented by B , unless stated otherwise.*

Given a desired virtual control ν_d and that B is square and invertible, its inverse mapping can be written as

$$u_d = B^{-1}\nu_d \quad (2.5)$$

Nonetheless, constraints due to saturation have not been regarded while calculating u_d . If the virtual control ν_d does not lie in the attainable set of virtual controls \mathbb{A} , one or more actuators will be saturated, and its response is consequently deteriorated, according to the function $\text{sat}(u)$, such that

$$u_{sat} = \text{sat}(B^{-1}\nu_d) \quad (2.6)$$

Finally, it results in the virtual control input

$$\nu = B u_{sat} \quad (2.7)$$

Of course, u_{sat} is not an optimal solution, since it corresponds to a $\nu < \nu_d$ and does not fulfill any optimization criteria. Therefore, the saturation effect cannot be ignored, provided that it results in an undesired change in the controls direction and in unplanned controls inputs, which may trigger instability. We can conclude that a straightforward solution, such as the matrix inversion or the pseudoinverse, are not suitable for allocating controls, since they do not take the saturation effect into account.

2.2 Optimization Problem

Systems are frequently subject to nonlinearities, present constructive uncertainties and other characteristics that may turn their modeling a difficult task. A good approach is to consider the variables that best represent the problem in question to obtain a solution as close as possible to the exact one. This modelling is regarded as an important tool of conceptualization and analysis, rather than as a principle yielding the philosophically correct solution (LUENBERGER and YE, 2008).

The optimization poses as an important tool for complex decision taking and analysis. Therefore, it can be also expanded to control allocation problems, since it establishes relation between related variables and delivers a solution that can be quantified in terms of performance or any other quality parameter. The optimal solution may be chosen so as to maximize or minimize a control objective.

A general optimization problem (NOCEDAL and WRIGHT, 1999) can be formulated and brought to the context of control allocation, in the form

$$\begin{aligned}
& \text{minimize} && f(u) \\
& \text{subject to} && c_i(u) = 0, \quad i \in \mathcal{E} \\
& && c_i(u) \geq 0, \quad i \in \mathcal{G} \\
& && u \in \mathbb{U}
\end{aligned} \tag{2.8}$$

where \mathcal{E} is the index set related to equality constraints, \mathcal{G} is the index set related to inequality constraints, $f(u)$ is a objective function vector also denoted objective or cost function and is subject to the equality constraints $c_i(u) = 0$ and inequality constraints $c_i(u) \geq 0$. Both constraints form the attainable set of control inputs \mathbb{U} , as defined in equation 1.5. Not every optimization problem consists of both equality and inequality constraints. Typically, inequality constraints are found in constrained problems, such as those subject to saturation, whereas the equality ones are found in the unconstrained type, although in both cases $u \in \mathbb{U}$ must hold.

Optimality Conditions

The optimality conditions are those that the a solution must fulfill and are often referred as Karush-Kuhn-Tucker conditions. They consist of first derivative tests and hence are also known as first-order conditions. Furthermore, they generalize the method of Lagrange multipliers, which allows only equality constraints. The Karush-Kuhn-Tucker (KKT) conditions pose as a theorem, as follows:

Theorem 2.1. *The vector u^* is a solution of (2.8) if and only if there exists vectors $u^* \in \mathbb{R}^p$ and $\lambda^* \in \mathbb{R}^q$ for which the following conditions hold for $(u, \lambda) = (u^*, \lambda^*)$:*

$$\nabla_x \mathcal{L}(u^*, \lambda^*) = 0, \tag{2.9a}$$

$$c_i(u^*) = 0, \forall i \in \mathcal{E}, \tag{2.9b}$$

$$c_i(u^*) \geq 0, \forall i \in \mathcal{G}, \tag{2.9c}$$

$$\lambda_i^* \geq 0, \forall i \in \mathcal{G}, \tag{2.9d}$$

$$\lambda_i^* c_i(u^*) = 0, \forall i \in \mathcal{E} \cap \mathcal{G}. \tag{2.9e}$$

Proof. It can be consulted on the Appendix B. □

The vector λ is also known as the Lagrange multipliers for the constraints. Besides containing the constraints from 2.8, the Lagrange multipliers must be non-negative and in (2.21c), whether λ_i^* or $c_i(u^*)$ must be zero, so that (2.9e) is zero. Therefore, it is called *complementarity condition*.

P-norm

The norm adopted in the optimization problem plays an important role in control allocation problems. The cost function $f(u)$ is usually defined with $l1$ or $l2$ -norm, corresponding to a linear programming (LP) or a quadratic programming (QP) problem, respectively.

Consider a cost function on the n -Euclidean space given by

$$f(u) = \left\| \begin{array}{c} f_1(u) \\ f_2(u) \\ \vdots \\ f_n(u) \end{array} \right\|_p \quad (2.10)$$

whose p -norm is mathematically defined as

$$\|f(u)\|_p = \left(\sum_{i=1}^n |f_i(u)|^p \right)^{\frac{1}{p}} \quad (2.11)$$

As p approaches infinity, it results in the $l-\infty$ norm, defined as

$$\lim_{p \rightarrow \infty} \|f(u)\| = \|f(u)\|_\infty = \max |f(u)| \quad (2.12)$$

Although control allocation problems often consist of linear constraints, the norm of the objective function determines a surface in the space \mathbb{R}^n and impacts the optimal solution directly, as depicted in the figure 2.2, which illustrates two optimization problems, one with $l1$ -norm at the left and another with $l2$ -norm at the right. The cost functions are represented by dashed-line contours projected onto the feasible region in gray and delimited by the constraint inequalities $c(u)$. Despite both plots present the same feasible region, their optimal solutions u^* are located in different points - whereas in the linear programming problem it always coincides with the vertices of the feasible region, in the quadratic programming it can be located anywhere on an edge or even inside the feasible region.

Thus, the location of u^* provides an important implication - solutions placed at the vertices tend to demand more some actuators to the detriment of others, while those located inside provided a more balanced distribution. Control balancing is a desired characteristic in aircrafts BODSON and FROST (2011), since it allows a homogeneous convergence of the control surfaces. In fact, the $l2$ -norm has been more extensively utilized to minimize errors in control allocation problems (JOHANSEN and FOSSEN, 2013).

In BODSON and FROST (2011), the authors propose a control allocation problem formulated as an optimization problem with infinity norm and its conversion to

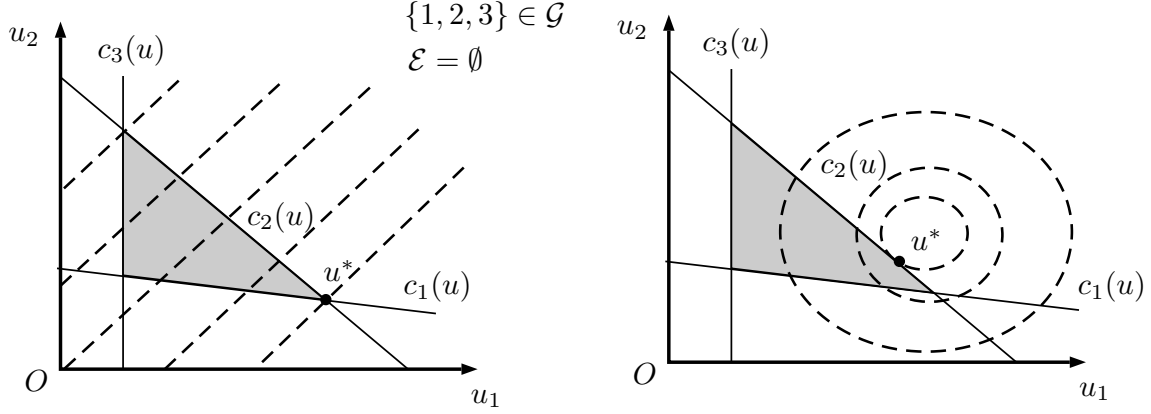


Figure 2.2: According to the norm adopted for the cost function $f(u)$, the optimization problem admits different solutions. At the left, it is depicted the l_1 -norm, and at the right, the l_2 . The cost function $f(u)$ is represented by contours in dashed lines.

a linear program. The use of infinity norm to the detriment of the l_1 -norm is justified based on the solution properties. The optimal solution obtained from the infinity norm minimizes the deflection of the actuators and the maximum deflection will be as small as possible, providing a better load distribution. On the other hand, from the characteristics of the linear programming derived from the l_1 norm, a control input u may attain n values whether at the upper limit or in the lower. In another words, if a virtual control ν_d cannot be achieved in any direction, n the actuators will whether at one of their limits or at a preferred position. This property may be questioned, since it may cause elevated deflection and the control surfaces cannot converge simultaneously towards the desired moment.

2.3 Unconstrained Control Allocation

In the unconstrained control allocation problems, the saturation effects in the actuators are ignored and consists of finding a control input u such that $\nu_d = Bu$. The matrix inversion aforementioned belongs to this class of control allocation, for instance.

In the general control allocation allied to saturation presented in (2.6), it was considered only the situation where B is square and invertible. However, a problem arises when the system is overactuated with a "fat" control effectiveness matrix and has full row rank. In this situation, a pseudoinverse solution can be applied to the control allocation problem formulated as a l_2 -norm optimization problem, in the form

$$\begin{aligned} \min: \quad & f(u) = \frac{1}{2}(u + o_f)^T W_u (u + o_f) \\ \text{s.t.} \quad & \nu_d = Bu \end{aligned} \tag{2.13}$$

where $W_u \in \mathbb{R}^{m \times m}$ is a positive definite weighting matrix and $o_f \in \mathbb{R}^m$ an offset to represent an off-nominal condition with one or more control effectors (OPPENHEIMER *et al.*, 2010). The solution of the optimization problem is obtained by calculating its Lagrangian, i.e by taking its derivatives with respect to u and its Lagrange multipliers, and finally, setting it to zero, which yields

$$u = -o_f + B^\dagger [\nu_d + B o_f] \quad (2.14)$$

such that the pseudoinverse matrix B^\dagger is

$$B^\dagger = W_u^{-1} B^T (B W_u^{-1} B^T)^{-1} \quad (2.15)$$

If $W_u = \mathcal{I}$ and $o_f = 0$, the vector of control input is defined by the Moore-Penrose pseudoinverse (JOHANSEN and FOSSEN, 2013), which results in

$$u = B^T (B B^T)^{-1} \nu_d \quad (2.16)$$

Another unconstrained control allocation method is the Explicit Ganging (OPPENHEIMER *et al.*, 2006) for combining effectors to reduce the number of effective control devices. For instance, the position of the left and right ailerons in an airplane can be linearly combined to produce a single effective roll control device, and thus, to create a pseudo vector of control input and diminish the control space dimension of the system. However, the Explicit Ganging can be utilized only if the contribution of the actuators to the control surface is known. Otherwise, the pseudoinverse must undergo a Singular Value Decomposition, where the eigenvalues higher than a preferred threshold value partitions the input and output unitary matrices to create a new control effectiveness matrix in order to produce similar controls, with the benefit of reducing the dimension of the problem, just as the Explicit Ganging. If the errors due to the reduction cannot be neglected, a second submatrix with secondary eigenvalues might be regarded.

Finally, the Daisy Chaining method can be utilized for setting a hierarchy among the actuators, such that the desired control input is sequentially allocated. Given a desired virtual control ν_d , the first actuator in the hierarchy is commanded to actuate. However, if it saturates, the remaining control input required is allocated to the next actuator hierarchically and so on.

2.4 Constrained Control Allocation

This section will cover methods for finding an optimal solution in problems where input constraints are regarded and will presented more details concerning linear and

quadratic programming, as well the main control allocation objectives and how they can be converted into a standard linear programming problem.

Finally, the control allocation algorithms revisited in the text are presented, including their main characteristics and implementation details, which are

- Direct Control Allocation;
- Linear Programming with Simplex;
- Weighted Least Squares with Active Set;
- Quadratic Programming with Primal-dual Interior-point algorithm.

2.4.1 Linear Programming

The linear programming (LP) consists of a mathematical method to optimize a linear objective function subject to linear constraints, represented by a set of m linear equalities and inequalities, which define the feasible region \mathbb{U} . If a candidate solution u satisfies all the constraints, then it is called a feasible point. A standard linear programming problem is, according to WRIGHT (1987), defined as

$$\begin{aligned} \min: \quad & f(u) = c^T u \\ \text{s.t.} \quad & Au = b \\ & u \geq 0 \end{aligned} \tag{2.17}$$

where $c \in \mathbb{R}^p$ is the coefficient vector, $A^{q \times p}$, $b \in \mathbb{R}^q$ and $x \in \mathbb{R}^q$ is the searched optimal solution that satisfies the constraints, while minimizing the cost function.

The Lagrange multipliers of (2.17) can be split into the vectors $\lambda \in \mathbb{R}^p$ and $s \in \mathbb{R}^n$, where λ is related to the equality constraints and s to the inequality constraints, respectively. Its Lagrangian is given by

$$\mathcal{L}(u, \lambda, s) = c^T u - \lambda^T (Au - b) - s^T u \tag{2.18}$$

Slack variables

Not rarely, the linear programs are not presented in the standard form and hence cannot be directly solved by mostly algorithms available. It is necessary to implement some variable substitution for this purpose, called slack variables (WRIGHT, 1987), that when added to a inequality constraint, transforms it into an equality.

Consider that the linear program in (2.17) is formulated with an inequality constraint $Au \leq b$ instead. If a slack variable $w \leq 0$ is added, the same expression can be restated with equality and inequality constraints, which results in

$$\begin{cases} Au + w = b \\ w \leq 0 \end{cases} \quad (2.19)$$

and can be solved by means of any LP solver.

Dual program

The standard optimization problem is associated with another similarly formulated linear program, called dual, in the form

$$\begin{aligned} \max: \quad & g(\lambda) = b^T \lambda \\ \text{s.t.} \quad & A^T \lambda + s = c \\ & s \geq 0 \end{aligned} \quad (2.20)$$

where $\lambda \in \mathbb{R}^q$ is a vector of dual variables and denotes the Lagrange multipliers for the equality constraints, and $s \in \mathbb{R}^p$ is the vector of dual slacks, namely the Lagrange multipliers for the inequality constraints (WRIGHT, 1987). The linear problem in equation 2.17 is often referred as primal, and both approaches are called primal-dual pair.

Optimality Conditions

The vector u^* is a solution of (2.17) if and only if there exists vectors $s^* \in \mathbb{R}^p$ and $\lambda^* \in \mathbb{R}^q$ for which the following KKT conditions hold for $(u, \lambda, s) = (u^*, s^*, \lambda^*)$:

$$A^T \lambda + s = c, \quad (2.21a)$$

$$Au = b, \quad (2.21b)$$

$$u_i s_i = 0, \quad i = 1, 2, \dots, p \quad (2.21c)$$

$$u \geq 0 \quad (2.21d)$$

$$s \geq 0. \quad (2.21e)$$

Note that in (2.21c) whether u_i or s_i must be zero for the equation to be zero, and this condition is called complementarity condition. We can conclude as well that the vector (u^*, s^*, λ^*) is a primal-dual solution if and only if u^* solves the primal problem (2.17) and (λ^*, s^*) solves the dual one (2.20).

2.4.2 Quadratic Programming

The quadratic programming (QP) stands for another type of optimization problem, which consists of a quadratic objective function subject to linear constraints. Ac-

According to NOCEDAL and WRIGHT (1999), a standard quadratic program can be stated as

$$\begin{aligned} \min: \quad & f(u) = \frac{1}{2}u^T Q u + c^T u \\ \text{s.t.} \quad & Au \geq b \end{aligned} \tag{2.22}$$

where $Q \in \mathbb{R}^{q \times q}$ is the Hessian matrix, that is the second partial derivative of the quadratic problem.

Optimality Conditions

Similar to the LP problems, any optimal solution u^* in QP problems must hold for the first-order conditions presented in the KKT conditions 2.1.

The Lagrangian of equation 2.22 is given by

$$\mathcal{L}(x, \lambda) = \frac{1}{2}u^T Q u + c^T u - \lambda(Au - b) \tag{2.23}$$

From the KKT conditions, any solution u^* must hold for

$$\begin{aligned} \frac{1}{2}u^T Q u + c^T u - \lambda(Au - b) &= 0, \\ Au^* &\geq b, \\ \lambda_i^*(a_i^T u^* - b_i) &= 0, \quad i = 1, \dots, m \\ \lambda^* &\geq 0 \end{aligned} \tag{2.24}$$

A quadratic programming problem always admits solution, although the computational cost depends on the objective function and on the number of constraints. If the Hessian Q is positive semidefinite, then the problem is called convex and can be easily solved with a few iterations, demanding a computational cost similar to the LP problems. Otherwise, if non-convex, the QP problem can present local solutions and stationary points (NOCEDAL and WRIGHT, 1999), which hinders the convergence to a global solution in the feasible region.

2.4.3 Common Objectives in Control Allocation

Common optimization problems are to minimize whether the error or the control in the control allocation problem $\nu = Bu$ (BODSON, 2002). Generally, the most common and primary objective in control allocation consists of minimizing the error that a given optimal solution, namely control input, may inflict to the DoF of a system. This objective calls error minimization and aims to find a control input u that satisfies

$$\begin{aligned}
\min: \quad & f(u) = \|Bu - \nu_d\| \\
\text{s.t.} \quad & u_{min} \leq u \leq u_{max}
\end{aligned} \tag{2.25}$$

On the other hand, when the error minimization problem results in a function cost $f(u) = 0$ and the system is overactuated, the system has enough authority to satisfy plainly the control requirements and hence the error minimization problem may result in multiple optimal solutions. In this case, a secondary optimization problem must be employed to choose among all solutions the one that best satisfies a second criteria, called control minimization. In OPPENHEIMER *et al.* (2010), the authors specify some situations where the control minimization is useful, such as to minimize control deflection, wing loading, drag and actuator power in aircraft applications. In minimum control deflection, the control effectors are driven towards zero position, whereas in minimum wind loading the outboard aerodynamic surfaces are likely to produce higher wing root bending moments when compared to the inboard surfaces, then weights can be related to each effector so as to penalize the outboard surfaces.

Thus, both objectives can be combined to produce a mixed optimization problem, such that

$$\begin{aligned}
\min: \quad & f(u) = \gamma \|Bu - \nu_d\| + \|u - u_p\| \\
\text{s.t.} \quad & u_{min} \leq u \leq u_{max}
\end{aligned} \tag{2.26}$$

where $\gamma > 0$ is chosen to set priority on the error minimization over the control minimization, and u_p is a preferred control with the preferred values for each effector according to the issue found in each application, as mentioned.

When $f(u) > 0$, it means also that the control requirements are beyond the system capabilities, which implies that the redundancy is not able of providing multiple solutions. Therefore, γ may receive a high value, such that the error minimization is not hindered.

The most commonly norms of the cost functions in formulations above are $l1$, which can be transformed to the linear programming to be solved with any of the available methods, such as Simplex and Interior-point (BODSON, 2002), and $l2$ norm, which results in a quadratic programming problem and can be solved through the Weighted Least Squares with Active Set and the Sequential Least Squares HÄRKEGÅRD (2002) and Iterative Fixed-point Method (JOHANSEN and FOSSEN, 2013).

Conversion to Linear Programming

Despite consisting of optimization problems, the control allocation problems are not written as a standard LP problem. Then it is necessary to convert them properly,

by formulating them with a cost function defined as a l_1 -norm. In OPPENHEIMER *et al.* (2006), the authors present a conversion of the error minimization problem, in the form

$$\begin{aligned} \min: \quad & f(u) = \begin{bmatrix} 0 & \dots & 0 & 1 & \dots & 1 \end{bmatrix} \begin{bmatrix} u \\ w \end{bmatrix} \\ \text{s.t.} \quad & \begin{bmatrix} w \\ -u \\ u \\ -Bu + w \\ Bu + w \end{bmatrix} \geq \begin{bmatrix} 0_m \\ -u_{max} \\ u_{min} \\ -\nu_d \\ \nu_d \end{bmatrix} \end{aligned} \quad (2.27)$$

where 0_m is a vector of zeros in \mathbb{R}^m and $w \in \mathbb{R}^m$ is a vector of slack variables, which represents how much control power demand exceeds supply in any given axis. In case the function cost equals zero, the control is attainable. Otherwise, a weighting vector $W_v \in \mathbb{R}^{n \times 1}$ may be employed by formulating the cost function as

$$\min: \quad f(u) = \begin{bmatrix} 0 & \dots & 0 & W_v^T \end{bmatrix} \begin{bmatrix} u \\ w \end{bmatrix} \quad (2.28)$$

to correct the strictness of the problem and hence to penalize control power deficiencies in some axes greater than others OPPENHEIMER *et al.* (2010), given the control unbalance due to the location of the optimal solution in the LP problems.

2.5 Linear Programming with Simplex

Consider the error minimization problem converted to a linear programming formulation, as detailed in the previous section, in the form

$$\begin{aligned} \min: \quad & f(u) = c^T x \\ \text{s.t.} \quad & Ax \geq b \end{aligned} \quad (2.29)$$

where

$$x = \begin{bmatrix} u \\ w \end{bmatrix} \quad (2.30)$$

and $x \in \mathbb{R}^p$ and the matrix A contains the vectors associated to x . The problem is rewritten in the LP standard as

$$\begin{aligned}
\min: \quad & f(x) = c^T x \\
\text{s.t.} \quad & Ax - w = b \\
& w \leq 0
\end{aligned} \tag{2.31}$$

For solving it, the Simplex method was chosen, because it presents optimal solution in finite time, a good performance and has already been utilized in control algorithm works, like in BODSON (2002). In this work, the author utilizes the Simplex method to allocate controls in some aircraft models, such as a complete C-17 and a tailless model.

As a starting point, consider that A has full row rank and the problem has already undergone a preprocessing phase for removing unnecessary variables and redundancies in the constraints. Prior to solving the LP problem with the Simplex method, it is important to introduce the concept of basic feasible solution, as defined in NOCEDAL and WRIGHT (1999).

Consider that x is a feasible solution with at most p nonzero components, in which it is possible to identify a subset $\mathcal{B}(x)$ originated from the index set $1, 2, 3, \dots, q$, such that

- $\mathcal{B}(x)$ has exactly p indices;
- $i \notin \mathcal{B}(x) \Rightarrow x_i = 0$;
- a matrix $A_B \in \mathbb{R}^{p \times p}$ defined as

$$A_B = [A_i]_{i \in \mathcal{B}(x)} \tag{2.32}$$

is non-singular, and A_i is the i -th column of A . In case every condition here stated holds, then x is a feasible basic solution.

Geometrically, $Ax \geq b$ represents a polytope in the space \mathbb{R}^q . Algebraically, its vertices coincide with the feasible basic solutions, what allows to state that every feasible basic solution is a vertex of this polytope and vice-versa (NOCEDAL and WRIGHT, 1999).

How Simplex works

The algorithm consists of two phases. In the first, the initial feasible basic solution is found with of the Simplex algorithm itself in an auxiliary optimization problem. In the second phase, successive new feasible basic solutions are searched with the objective of lowering the function cost $f(x)$, as shown in figure 2.3. This search consists of moving from a vertex to another adjacent, permuting a variable which is at its limit with another that is not. If it is not possible to reduce the cost

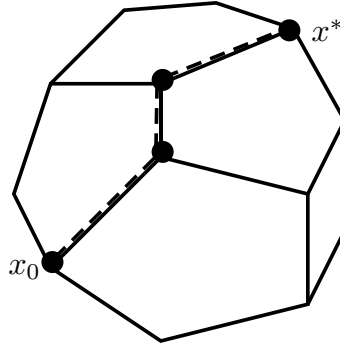


Figure 2.3: At each iteration, the Simplex algorithm visits every adjacent vertex to find the optimal feasible basic solution.

any more, the optimal feasible basic solution x^* has been found and the algorithm is terminated.

The Simplex Algorithm

According to NOCEDAL and WRIGHT (1999), some considerations are needed prior to solving the LP problem, which are

- let A_B be a submatrix of A with all its linearly independent columns;
- x_B stands for the basic variables associated to A_B ;
- let A_N be the remaining columns of A ;
- x_N stands for the nonbasic variables associated to A_N ;
- $i = 1, 2, \dots, q$ is the column index of A ;
- \mathcal{B} is an index set composed by indices related to A_B ;
- \mathcal{N} is the complement of \mathcal{B} related to A_N ;
- The vectors s and c are also split into basic and non basic vectors, according to the i indices of \mathcal{B} , which gives s_B, s_N, c_B and c_N ;
- if x is a feasible solution, then $x^T s = 0$.

In order to find the optimal solution x^* , the constraint matrix A is rewritten according to the definitions above, which results in

$$Ax = A_B x_B + A_N x_N = b \quad (2.33)$$

By setting $x_N = 0$, the primal variable is found by calculating

$$x_B = A_B^{-1} b \quad (2.34)$$

which satisfies the KKT conditions, since $x_B \geq 0$. To satisfy the complementarity condition, we set $s_B = 0$ and the remaining components of the algorithm are calculated, as follows

$$\begin{aligned}\lambda &= A_B^{-T} c_B \\ s_N &= c_N - A_N^T \lambda\end{aligned}\tag{2.35}$$

If $s_N \geq 0$, the optimal feasible solution was found and the algorithm is terminated.

Otherwise, a new index $q \in \mathcal{N}$ to enter \mathcal{B} is elected, such that $s_q < 0$. With this choice, it is granted the decrease of the function cost $f(x)$. Then, the equation

$$A_B \xi = A_q\tag{2.36}$$

is solved for ξ . If $\xi \leq 0$ the problem is unbounded and the algorithm is terminated. Otherwise, calculate

$$x_q^{k+1} = \min_{j|\xi_j > 0} \frac{(x_B)_j}{\xi_j}\tag{2.37}$$

and j is defined as the index of the basic variable for which the minimum is achieved. Then, the basic and non-basic variables are updated by proceeding to the calculation of

$$\begin{aligned}x_B^{k+1} &= x_B - \xi x_q^{k+1} \\ x_N^{k+1} &= \left[0 \quad \dots \quad 0 \quad x_q^{k+1} \quad 0 \quad \dots \quad 0 \right]^T\end{aligned}\tag{2.38}$$

which forms a new feasible solution x^{k+1} . Finally, q is added to \mathcal{B} and j is removed. At this point, an iteration is concluded and the next one starts at equation (2.33). When the algorithm is terminated, i.e $s_N \geq 0$, then $x^* = x^{k+1}$ and the optimal control input u is given by

$$u^* = \left[x_1^* \quad x_2^* \quad \dots \quad x_m^* \right]^T\tag{2.39}$$

More information concerning Simplex can be found in NOCEDAL and WRIGHT (1999) and WRIGHT (1987), although other linear programming solvers can be utilized as desired, such as Dual-simplex or Interior Point (PETERSEN and BODSON, 2005).

Still concerning the Simplex algorithm, its efficiency is worth mentioning. Despite it has an exponential complexity, as observed in KLEE and MINTY (1972), where the algorithm had to visit every $2^n - 1$ vertices to find the optimal feasible basic solution, the majority of practical situations demands as many iterations as two to three times the row rank of A .

2.6 Direct Control Allocation

In DURHAM (1993), the author proposes a method for allocating overactuated controls by projecting a desired virtual control ν_d onto the boundary of the attainable set of virtual controls, denoted $\delta\mathbb{A}$. First, the direction of ν_d is obtained, according to the equation

$$\hat{\nu}_d = \frac{\nu_d}{\|\nu_d\|_2} \quad (2.40)$$

Secondly, one determines the vertices and edges of $\delta\mathbb{A}$, to which the desired virtual control ν_d points to and their intersection $\|\nu\|_2 \hat{\nu}_d$, such that $\|\nu\| > 0$. Otherwise, the control u that generates the intersection ν is calculated, by rescaling ν_d to lie on the boundary $\delta\mathbb{A}$, such that

$$u^* = \alpha u_d, \text{ with } \alpha = \frac{\|\nu_d\|_2}{\|\nu\|_2} \quad (2.41)$$

In the original work, Durham utilized a geometrical approach combined with the addition of vectors to compound a linear system in order to find ν , as shown in figure 2.4. However, in BODSON (2002), the author restated the problem as a LP problem with equality constraints to determine α and u , in the form

$$\begin{aligned} \max: \quad & f(\alpha) = \alpha \\ \text{s.t.} \quad & Bu = \alpha \nu_d \\ & \alpha \nu_d \in \mathbb{A} \end{aligned} \quad (2.42)$$

which can be solved with any LP solver, such as Simplex.

Since ν_d has been rescaled, it results that the vector direction is maintained, and provided that ν lies on $\delta\mathbb{A}$, at least one actuator is required to work at its limit position. The algorithm provides an attainable u with the maximum control utilization, although it requires a $u_{min} \leq 0$ and $u_{max} \geq 0$, which makes the application difficult (BODSON, 2002). The algorithm is also criticized for not allowing to prioritize any DoF.

2.7 Weighted Least Squares

The Weighted Least squares (WLS) belongs to the QP problem, provided that its cost function is quadratic. The problem is algebraically formulated as

$$\begin{aligned} \min: \quad & f(u) = \|W_v(\nu - \nu_d)\|_2 \\ \text{s.t.} \quad & c(u) = \begin{bmatrix} u - u_{min} \\ u_{max} - u \end{bmatrix} \geq 0 \end{aligned} \quad (2.43)$$

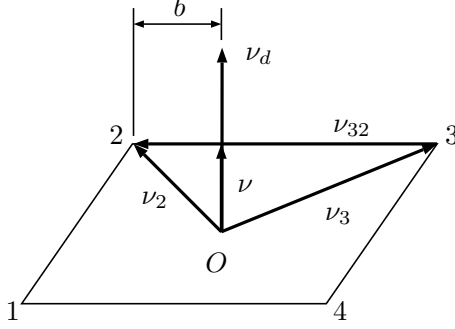


Figure 2.4: The parallelogram represents \mathbb{A} with vertices 1,2,3 and 4, and edges connecting them. The virtual control ν is solution of the vector equation $\nu \hat{\nu}_d + b \nu_{32} = \nu_2$.

where $W_v \in \mathbb{R}^{n \times n}$ is a full-rank weighting diagonal matrix, useful for setting assigning weights to each DoF. In W_v , the diagonal elements may receive normalized values for setting priorities to the moment generators in ν .

In MONTEIRO *et al.* (2016a), the authors propose a WLS solver which utilizes the Lagrange multipliers of the cost function to derive an algebraic solution. The solver constraints every effector at its lower and upper bounds to calculate a candidate solution u that minimizes the function cost $f(u)$. Other solvers found in the literature are the Weighting Least Squares with Active Set (WLSSA) and the Primal-dual Interior Point (PDIP), which will be detailed next.

2.7.1 Weighted Least Squares with Active Set

The quadratic programming approach in equation (2.43) can be enhanced by adding a control minimization term, which results in the equation

$$\begin{aligned} \min : f(u) &= \|W_u(u - u_p)\|_2 + \gamma \|W_v(Bu - \nu_d)\|_2 \\ \text{s.t.} : u_{min} &\leq u \leq u_{max} \end{aligned} \quad (2.44)$$

As proposed by HÄRKEGÅRD (2002), this mixed minimization problem can be rewritten as

$$\begin{aligned} \min: \quad & f(u) = \|Au - b\|_2 \\ \text{s.t.} \quad & Cu \leq U \end{aligned} \quad (2.45)$$

where the matrices A , b , C and U stand for

$$A = \begin{pmatrix} \gamma W_v B \\ W_u \end{pmatrix}, \quad b = \begin{pmatrix} \gamma W_v \nu \\ W_u u_p \end{pmatrix}, \quad C = \begin{pmatrix} \mathcal{I} \\ -\mathcal{I} \end{pmatrix}, \quad U = \begin{pmatrix} u_{min} \\ -u_{max} \end{pmatrix} \quad (2.46)$$

where \mathcal{I} is an identity matrix and $W_u \in \mathbb{R}^{m \times m}$ is a weighting diagonal matrix utilized for setting priorities among the actuators.

The WLSAS was proposed in HÄRKEGÅRD (2002) to demonstrate that an optimal solution u can be found in finite time, unlike other solvers, which converge when iterations go to infinity. Thus, it can be utilized without restriction in practical situations.

The active set algorithm splits the controls into a working set \mathcal{W} for the saturated controls and in a free set for the nonsaturated ones. An initial estimate of $u_0 \in \mathbb{U}$ is defined and at each iteration, an attainable control u^{k+1} is obtained at a successively smaller cost $f(u)$.

As the algorithm iterates, some inequality constraints are treated like equalities and stand for the working set \mathcal{W} , while the remaining ones are disregarded. The working set at its optimal is known as active set.

In the algorithm, every actuator may be affected by an optimal perturbation $p \in \mathbb{R}^m$ to be determined from

$$\begin{aligned} \min: \quad & f(p) = \|A(u^k + p) - b\|_2 \\ \text{s.t.} \quad & Bp = 0, \\ & p_i = 0, \quad i \in \mathcal{W} \end{aligned}$$

In case $u^k + p$ remains feasible, u^{k+1} receives the sum and the Lagrange multipliers λ associated with the active constraints are calculated. Similar to other quadratic programming algorithms, its Lagrange multipliers (λ, μ) are calculated from the equation

$$A^T(Au - b) = \begin{pmatrix} B^T & C_o^T \end{pmatrix} \begin{pmatrix} \mu \\ \lambda \end{pmatrix} \quad (2.47)$$

where μ are the Lagrange multipliers associated with $\nu = Bu$ and λ are those associated with the active constraints $Cu \leq U$, and C_o^T is composed by the rows of C related to the constraints in the working set.

If all λ are nonnegative, u^{k+1} is the optimal solution and the algorithm is terminated. Otherwise, the constraint related to the most negative λ is removed from the working set.

Nevertheless, if $u^k + p$ is unfeasible, a step α must be determined, such that

$$u^{k+1} = u^k + \alpha p \quad (2.48)$$

remains feasible. Afterwards, the primary bounding constraint is added to the working set \mathcal{W} and an iteration is complete. The algorithm goes back to equation (2.7.1) and searches a new perturbation p . This process is repeated until every

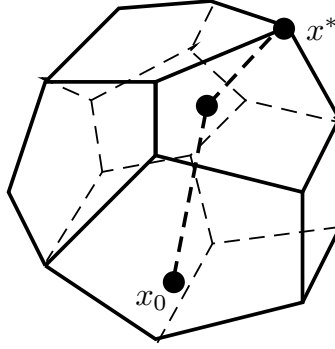


Figure 2.5: Interior-point algorithm travels through the feasible polytope towards the optimal solution x^* .

$\lambda_i \geq 0$ holds or a predefined maximal number of iterations has been reached.

This algorithm is more effective when a good estimate of the active working set is available. Provided that a control input u calculated by the optimization problem does not differ much between two sampling times, the last optimal solution may be utilized as a good estimate u_0 to reduce the amount of iterations.

2.7.2 Primal-dual Interior-Point

The Interior point methods are a class of algorithms which arise to compete with Simplex, provided that the latter has presented little efficiency with complex problems with a large number of variables and demands large space for storing data.

The Interior point methods share a few characteristics that distinguish them from Simplex, such as the computational cost - although each iteration of Interior point has an elevated cost, they converge faster to the optimal solution. The algorithm travels within a polytope, formed by the inequality constraints, towards their limit borders, whereas the Simplex visits every adjacent vertex of the feasible polytope.

In NOCEDAL and WRIGHT (1999) and PETERSEN and BODSON (2005), the authors present methods for linear and quadratic programming problems by using Interior-point methods. However, in this dissertation the Primal-dual Interior Point algorithm for QP is employed for control allocation purposes.

Now, we consider that the mixed optimization problem (2.26) is written in a standard QP problem and undergoes a variable change, in the form

$$\begin{cases} x = u - u_{min}, & x_0 = u_p - u_{min} \\ x_{max} = u_{max} - u_{min}, & \nu_0 = \nu_d - B u_{min} \end{cases} \quad (2.49)$$

that results in the constraint set

$$\begin{cases} x + w = x_{max} \\ x \geq 0 \\ w \geq 0 \end{cases} \quad (2.50)$$

where w is a slack variable utilized to grant the upper constraint in x . An expansion of the l_2 -norm of the mixed minimization objective over x is performed, which results in the cost function

$$\begin{aligned} f(x) &= (Bx - \nu_0)^T(Bx - \nu_0) + \gamma(x - x_0)^T(x - x_0) \\ &= \frac{1}{2}x^T Qx + c^T x + k \end{aligned} \quad (2.51)$$

where $H = 2(B^T B + \gamma I)$, $c^T = -2(\nu_0 B + \gamma x_0^T)$ and $k = \nu_0^T \nu_0 + \gamma x_0^T x_0$. Provided that k is constant, it cannot influence on the optimal solution, and therefore can be neglected. Thus, the optimization problem can be rewritten as

$$\begin{aligned} \min: \quad & f(x) = \frac{1}{2}x^T Qx + c^T x \\ \text{s.t.} \quad & x + w = x_{max} \\ & x \geq 0, \quad w \geq 0 \end{aligned} \quad (2.52)$$

The objective function in (2.52) is convex and the KKT conditions hold globally. Therefore, a smooth barrier function $\log(x)$ is adopted to approximate the nonnegative constraints x and w , which tend to $-\infty$ when the analyzed variable tends to 0.

Now we proceed to the calculation of the Lagrangian of the equation (2.51), which gives

$$\mathcal{L}(x, \lambda) = \frac{1}{2}x^T Qx + c^T x + \lambda^T(x + w - x_{max}) - \mu \sum_{i=1}^n \log(x_i) - \mu \sum_{i=1}^n \log(w_i) \quad (2.53)$$

where $\mu > 0$ is a penalty parameter (PETERSEN and BODSON, 2005) and λ a Lagrange multiplier for the dual variable. Then, the partial derivatives of the Lagrangian with respect to x and λ are calculated in order to obtain the optimal conditions of the problem. The derivatives result in the system of equations

$$\begin{cases} Qx + c + \lambda - s = 0 \\ x + w - x_{max} = 0 \\ Xs - \mu \mathbf{1} = 0 \\ W\lambda + \mu \mathbf{1} = 0 \\ x > 0, w > 0, \lambda > 0, s > 0 \end{cases} \quad (2.54)$$

where the elements of the vectors x and w form the diagonals of X and W , re-

spectively, and $\mathbf{1}$ is a vector in \mathbb{R}^n . Provided x and λ cannot attain 0, it is needed to impose equality constraints by performing the variables change $s = \mu X^{-1}\mathbf{1}$ and $\lambda = \mu W^{-1}\mathbf{1}$. In order to satisfy the Karush-Kuhn-Tucker conditions, the equation 2.54 must be valid for $\mu = 0$, and therefore, $Xs = 0$ and $W\lambda = 0$ hold. On the other hand, μ is the complementarity gap, since it guides the solution through a trajectory named central path towards the optimal solution x^* . As the feasible solution approximates x^* , μ tends to zero.

Algebraically, μ is defined as

$$\mu = \sigma\gamma, \text{ with } \gamma = \frac{x^T s + w^T \lambda}{2n} \quad (2.55)$$

where $0 < \sigma < 1$, and σ is an adjust parameter to set the convergence rate. Provided that the intention of this method is to undertake an iterative search for the optimal solution along the central path at successively lower μ , the candidate solution must also be located at the neighborhood of the central path. Now by means of the equation 2.54, the algorithm searches for the step direction, calculates the residuals and the maximum step, whose implementation details can be consulted on BODSON and FROST (2011). The optimal solution is found when all the residuals and the complementarity gap are zero.

In PETERSEN and BODSON (2005), the authors revisit the problem for the case with bilateral limits from the linear problem, which is then expanded. It is assumed that a lower bound is zero, and for the upper, a second Lagrange multiplier λ is added. Furthermore, the finite upper bound is manipulated through the use of a slack variable w . Despite the adaptations result in equations similar to those in 2.54, structurally the formulation is still originated from the l_1 norm.

2.8 Error Analysis

With the objective of validating and comparing the results obtained from the control allocation techniques here addressed, it is important to define two suitable metrics to deal with the main primary control allocation objectives. The first one is derived from the error minimization objective, since it compares the error between the desired and the effective virtual control vectors ν_d and ν . It is denoted Virtual Control Error (VCE) and is calculated by

$$VCE = \|\nu_d - Bu\|_2 \quad (2.56)$$

and computed at each code iteration. This metrics was utilized in HÄRKEGÅRD (2002) to compare some Weighted Least Squares algorithms, such as the Sequential Least Squares, the Minimal Least Squares, the Redistributed Pseudoinverse and the

Fixed-point Iteration.

To compare how the CA algorithms change the direction of ν_d and provided that the DCA is also utilized to allocate controls in the simulations and experiments, we propose a metrics to calculate it, called Direction Error (DE), given by

$$DE = \left\| \frac{\nu_d}{\|\nu_d\|} - \frac{Bu}{\|Bu\|} \right\|_2 \quad (2.57)$$

and computed at each iteration and stored in a vector. The vectors are normalized to allow a fair comparison of the directions. For both error analysis metrics, the average and maximum values will be displayed and hence the CA algorithms will be compared.

2.9 Conclusions

The main concepts of control allocation and the algorithms presented in this chapter form the fundamentals of this dissertation, because they can be applied not only to the systems to come, but also to any other robot desired, along with any desired High-level controller.

The knowledge of the types of control allocation, such as constrained and unconstrained, allow to comprehend how the allocation problem can be dealt with and the respective optimization problem involved, besides providing the necessary conditions for an optimal solution to exist. Next, the norm, the constraints and the objectives allow to elect a strategy or an algorithm to address the control allocation problem. Finally, the metrics to compare the experiments and simulations were presented in order to provide a reliable tool to compare the results.

Chapter 3

Robots: Modeling and Control

This chapter discusses the kinematics and the dynamics of the four robot types here considered, presents the High-level controller adopted and provides also an overview about how the control allocation can be applied to deal with the input constraints to which they are subject.

3.1 Wheeled Mobile Robots

Among the mobile robots, the wheeled type is most common and presents simplified constructive characteristics. They can adapt well to different terrain conditions and can move not only in indoor environments, but also on general regular surfaces. Their motion with respect to an inertial frame \mathcal{F}_o can be provided whether by a kinematic or a dynamic model. The kinematic model describes the configuration of the robot as function of velocities, whereas the dynamic model describes its configuration as function of the generalized forces delivered by the motors attached to the wheels. Nonetheless, the kinematic model is the most adopted to describe their motion (LAGES, 1998).

Consider a vector $\chi \in \mathbb{R}^e$ as the vector of generalized coordinates which describes fully the robot configuration. A general kinematic model for a wheeled mobile robot can be written as

$$\dot{\chi} = G(\chi)\nu \quad (3.1)$$

where $G(\chi) \in \mathbb{R}^{e \times n}$ is composed by the input vector fields columns $g_i(\chi)$ with $i = 1, \dots, n$, the vector $\nu \in \mathbb{R}^n$ corresponds to the virtual control input vector composed by the generalized velocities of the vehicle with respect to the body frame \mathcal{F}_c and e is the number of DoF of the system SICILIANO *et al.* (2009).

Due to rolling and sliding constraints in the wheels, such systems have more DoF e than the number n of virtual control inputs, and therefore, the matrix $G(\chi)$ has less columns than rows.

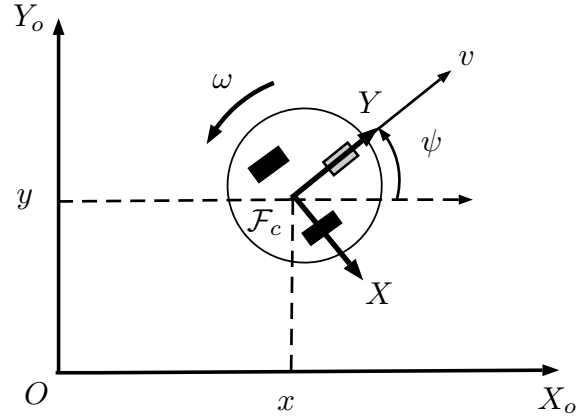


Figure 3.1: Posture of a two-wheeled-mobile robot with respect to the inertial frame \mathcal{F}_o .

This system also contains $e-n$ nonholonomic constraints, which reduces the maneuverability of the vehicle SIEGWART and NOURBAKHS (2004) and can be described by the equation below in the Pfaffian form

$$a_i^T(\chi)\dot{\chi} = 0, \quad i = 1, \dots, e - n \quad (3.2)$$

3.1.1 Differential Drive Robot

As an example, consider a wheeled mobile robot composed by two rigid differential rear wheels and a front castor wheel to provide directional stability and to prevent wandering, which does not contribute to the kinematic modeling. The wheels do not suffer deformation and the vehicle moves on a horizontal flat plane.

The body frame \mathcal{F}_c is located at half the distance from its wheels and it is used as a reference with respect to an inertial frame \mathcal{F}_o to describe the vehicle position and orientation. It results that the robot travels in a 2D plane and can rotate by a steering angle ψ about the Z axis, as depicted in 3.1. Thus, the generalized coordinates χ of the robot are given by

$$\chi = \begin{bmatrix} p \\ \eta \end{bmatrix} \quad (3.3)$$

where $p \in \mathbb{R}^2$, such that $p = \begin{bmatrix} x & y \end{bmatrix}^T$, and $\eta \triangleq \psi$ can be represented by the rotation matrix

$$R_z(\psi) = \begin{bmatrix} \cos \psi & -\sin \psi & 0 \\ \sin \psi & \cos \psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (3.4)$$

3.1.2 Nonholonomic Constraints

Although the full configuration of the robot is described by three DoF, the whole configuration space can be accessed by means of the vector of generalized velocities $\nu \in \mathbb{R}^2$, since the differential drive robot is subject to 1 nonholonomic constraint, which can be expressed in the Pfaffian form as

$$\dot{x} \sin \psi - \dot{y} \cos \psi = \begin{bmatrix} \sin \psi & -\cos \psi & 0 \end{bmatrix} \dot{\chi} = 0 \quad (3.5)$$

as described in equation 3.2. This constraint stands for a pure rolling constraint, which means that in the absence of slipping, the velocity of the contact point has zero component in the direction orthogonal to the sagittal plane (SICILIANO *et al.*, 2009), or in another words the wheel cannot move directly aside. Despite the non-holonomic constraint, the vehicle can reach any desired posture in finite time.

3.1.3 Kinematics in Polar Coordinates

Consider that the virtual control inputs of a mobile robot are its linear velocity v and angular velocity ω . Hence, the kinematic model SICILIANO *et al.* (2009) in Cartesian coordinates is

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} \cos \psi & 0 \\ \sin \psi & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} \quad (3.6)$$

The vehicle must travel from a given posture (p_1, ϕ) to a desired posture $(p_2, \theta)^T$, as depicted in the figure 3.2, which illustrates a parking maneuver task.

In order to implement the proposed control law, the system representation must be converted from Cartesian to polar coordinates, as depicted in figure 3.2, where r is the distance error between p_1 and p_2 , θ denotes the desired orientation angle at pose p_2 with respect to the X axis, and α stands for the orientation error between the θ and ψ . Algebraically, the coordinate system transformation is given by

$$\begin{cases} r = \sqrt{\Delta x^2 + \Delta y^2} \\ \theta = \text{atan2}(\Delta y, \Delta x) \\ \alpha = \theta - \psi \end{cases} \quad (3.7)$$

with $r > 0$, $\alpha \in (-\pi, \pi]$ and $\theta \in (-\pi, \pi]$. Then, the time derivative of the system of equations in (3.7) is calculated to obtain the new kinematic model, which yields

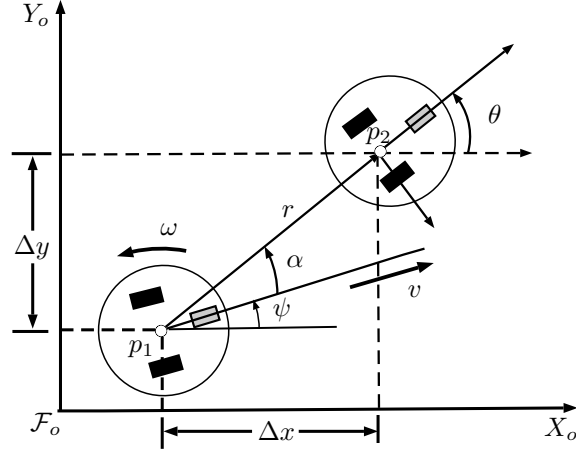


Figure 3.2: Conversion from Cartesian to polar coordinates.

$$\begin{bmatrix} \dot{r} \\ \dot{\alpha} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} -\cos \alpha & 0 \\ \frac{\sin \alpha}{r} & -1 \\ \frac{\sin \alpha}{r} & 0 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} \quad (3.8)$$

It is important to mention that the equation presented in 3.8 is utilized only for differential drive vehicles. Moreover, the case $r = 0$ must be avoided, since it corresponds to a singularity and make α and ϕ become undefined. It implies that the polar coordinate system cannot be related to the Cartesian coordinates.

It must be considered that commercial differential drive robots are not directly controlled by the linear velocity v and angular velocity ω . In fact, these velocities are converted into the wheel velocities, namely the left and right wheel velocities v_l and v_r , respectively. These velocities are related by the mapping

$$\begin{bmatrix} v \\ \omega \end{bmatrix} = \frac{1}{2} \begin{bmatrix} 1 & 1 \\ \frac{1}{r_d} & -\frac{1}{r_d} \end{bmatrix} \begin{bmatrix} v_r \\ v_l \end{bmatrix} \quad (3.9)$$

where r_d stands for half the distance between the rear wheels. In order to find the corresponding individual velocities, the inverse mapping results in

$$\begin{bmatrix} v_r \\ v_l \end{bmatrix} = \begin{bmatrix} 1 & r_d \\ 1 & -r_d \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} \quad (3.10)$$

Odometry

The odometry is based on the concept that a wheel revolution can be converted into a relative linear displacement. Its implementation is simple, since the required sensors are commonly present in commercial mobile robots. The movement of the robot is sensed whether by the wheel encoders or heading sensors, which are used to compute the linear and angular velocities.

Let β_l be the encoder measurement from the left wheel rotation between two measurement instants and β_r the respective encoder measurement from the right wheel, r_w the wheel radius and c_b is the number of transitions per wheel revolution. The driven distance Δd and the steering angle variation $\Delta\psi$ are obtained from the formulas

$$\Delta d = \frac{\pi r_w (\beta_r + \beta_l)}{c_b}, \quad \Delta\psi = \frac{\pi r_w (\beta_r - \beta_l)}{r_d c_b} \quad (3.11)$$

which result in the linear velocity v and the angular velocity ω when integrated SICILIANO *et al.* (2009).

In discrete time, given an actual pose χ_k , the next robot pose χ_{k+1} with respect to an inertial frame can be mathematically derived by using the Euler method (SICILIANO *et al.*, 2009), which yields

$$\chi_{k+1} = \chi_k + \begin{bmatrix} \Delta d \cos \Delta\psi \\ \Delta d \sin \Delta\psi \\ \Delta\psi \end{bmatrix} \quad (3.12)$$

One must consider that the sensor readings Δd and $\Delta\psi$ are not taken with respect to the inertial frame, but rather to the frame where the measurements were last taken, which implies that errors are propagated to every future measurement. Besides the measurement errors due to odometry, other systematic errors may also be present, such as range, turn and drift errors, or may also be originated from environmental factors (SIEGWART and NOURBAKHS, 2004), such as:

- limited resolution during integration;
- misalignment of the wheels;
- uncertainty in the wheel diameter;
- variation in the contact point of the wheel;
- unequal floor contact;
- irregular floor surface;
- slipping during motion.

3.1.4 High-level Controller

According to the Brockett's Theorem, a nonholonomic system cannot be asymptotically stabilized by a time-invariant control law in the Lyapunov sense (BROCKETT, 1983). Therefore, effective control laws have been extensively researched.

The Lyapunov-based tracking control utilized in AICARDI *et al.* (1995) can be employed to control a wheeled mobile robot in polar coordinates and has been elected as High-level controller for the control allocation scheme.

Without loss of generality, we consider that at every time instant, the condition for the distance error $r > 0$ holds and the state variables (r, α, θ) can be directly measured. We search a $\nu_d = \begin{bmatrix} v_d & \omega_d \end{bmatrix}^T$ that make the state variables converge asymptotically to the limiting point $\begin{pmatrix} 0 & 0 & 0 \end{pmatrix}^T$, while $r = 0$ is avoided.

Consider the following candidate Lyapunov function:

$$\begin{aligned} V &= V_1 + V_2 \\ &= \frac{1}{2}\zeta r^2 + \frac{1}{2}(\alpha^2 + k_\theta \theta^2) \end{aligned} \quad (3.13)$$

where $\zeta > 0$ and $k_\theta > 0$ are parameters utilized for adjusting the controller output. Provided that V is quadratic, the Lyapunov function is also a positive definite function. Then, we proceed to the calculation of its time derivative, which yields

$$\begin{aligned} \dot{V} &= \dot{V}_1 + \dot{V}_2 \\ &= \zeta r \dot{r} + (\alpha \dot{\alpha} + k_\theta \theta \dot{\theta}) \\ &= -\zeta r v \cos \alpha + \alpha \left[-\omega + v \frac{\sin \alpha}{\alpha} \frac{(\alpha + k_\theta \theta)}{r} \right] \end{aligned} \quad (3.14)$$

Now, the main challenge resides in finding input laws for generating the desired virtual control inputs v_d and ω_d , such that $\dot{V} \leq 0$. Then, the control laws suggested in AICARDI *et al.* (1995) are

$$\nu_d = \begin{cases} v_d = k_r r \cos \alpha \\ \omega_d = k_\alpha \alpha + k_r \cos \alpha \frac{\sin \alpha}{\alpha} (\alpha + k_\theta \theta) \end{cases} \quad (3.15)$$

where $k_r > 0$ and $k_\alpha > 0$ are controller design constants. Finally, \dot{V} results in

$$\dot{V} = -\zeta k_r r^2 \cos^2 \alpha - k_\alpha \alpha^2 \leq 0 \quad (3.16)$$

As a result, \dot{V} is a negative semidefinite function as desired, and provided that V is continuous, the closed loop stability of the system is granted. Hence, the following observations can be made:

- $V = 0$ if and only if $r = 0$ and $\alpha = 0$, i.e. the robot is exactly at p_2 and at the desired orientation θ ;

- V é bounded from below;
- V is not a increasing function due to (3.16);
- V is differentiable and \dot{V} é uniformly continuous. Hence, from the Barbalat's Lemma, \dot{V} tends to zero, and by analyzing (3.16), r e α also tends to zero.

By replacing (3.15) into (3.8), the following closed-loop system equations are obtained to represent the system dynamics:

$$\begin{cases} \dot{r} = -k_r r \cos^2 \alpha, \\ \dot{\alpha} = -k_\alpha \alpha - k_r \mu \theta \cos \alpha \frac{\sin \alpha}{\alpha}, \\ \dot{\theta} = k_r \sin \alpha \cos \alpha. \end{cases} \quad (3.17)$$

The polar coordinates r and α tend to zero, thus \dot{r} and $\dot{\theta}$ will also tend to zero. However, $\dot{\alpha}$ will converge to a constant value $k_1 \mu \bar{\theta}$. On the other hand, we already know that $\dot{\alpha}$ is uniformly continuous and that α tends to zero, then we conclude from the Barbalat's Lemma that $\dot{\alpha}$ also tends to zero, which implies that $\bar{\theta} = 0$ (LAGES, 1998).

3.1.5 Control Allocation Problem

The differential drive vehicle consists of two independent wheels, used to control three DoF, namely x , y and ψ , such that any desired posture χ_d can be reached by means of the generalized velocities v and ω in finite time, and the linear and angular velocities v and ω can be reproduced by a combination of the right wheel velocity v_r and left wheel velocity v_l .

The control allocation problem for a wheeled mobile robot would typically be concerned with the dynamics and would find the maximum and minimum torque for the actuators. However, the control allocation range can be expanded also to solve a kinematic problem, given that

$$\nu = \begin{bmatrix} v \\ \omega \end{bmatrix} = \frac{1}{2} \begin{bmatrix} 1 & 1 \\ \frac{1}{r_d} & -\frac{1}{r_d} \end{bmatrix} \begin{bmatrix} v_r \\ v_l \end{bmatrix} = Bu \quad (3.18)$$

and B is an invertible control effectiveness matrix. If the High-level controller computes a desired virtual control input vector ν_d that cannot be attained by the right and left wheel velocities, the control allocation algorithms must find a “best” vector of wheel velocities v_r and v_l .

3.2 Remotely Operated Underwater Vehicle

Originally created to substitute divers, the Remotely Operated Underwater Vehicles (ROV) are employed in deep-water works, such as scientific studies concerning geology, deep-water fauna and flora and their ecosystems, rescue, inspection and maintenance of undersea cables. Operations in such depth are executed in a harsh environment under extreme pressure and cold temperatures, which impedes direct human activities.

The ROVs play an important role in the oil and gas industry, as the demand for oil worldwide increases and the onshore and shallow-water-offshore production decreases, the search for fossil fuels is moving towards deep-water and ultra-deep-water oil fields (SHUKLA and KARKI, 2016). The ROVs also perform human jobs more effectively and are able to work uninterruptedly, what improves cost efficiency and reduces the risk of environmental disasters, such as the oil spill in the Gulf of Mexico in 2010. These facts have motivated improvements towards developing more intelligent robotic technologies to improve safety during oil extraction related procedures.

In this context, the Group of Simulation, Control and Automation in Robotics (GSCAR), located at the UFRJ, has developed a ROV named LUMA, as shown in figure (3.3). Formerly, the vehicle had been used for inspecting diversion tunnels in dams. Later, it underwent modifications for resisting the adverse underwater environment in the Admiralty Bay in Antarctica in order to gather high-quality images from the seabed and from various forms of life found in waters up to 300 meters depth.

3.2.1 Kinematics

Given the inertial frame \mathcal{F}_o and the ROV frame \mathcal{F}_c , the position of the vehicle can be described by the vector $p = \begin{bmatrix} x & y & z \end{bmatrix}^T$, as shown in figure 3.4. On the other hand, its orientation can be described through the raw-pitch-yaw Euler angles $\eta = \begin{bmatrix} \phi & \theta & \psi \end{bmatrix}^T$. Thus, its complete configuration is described by

$$\chi = \begin{bmatrix} p \\ \eta \end{bmatrix}, \chi \in SE(3) \quad (3.19)$$

where $SE(3)$ is defined in equation A.27 and the Euler-angles derivatives are related to the angular velocities as in equation A.17.

¹Extracted from ANDRADE, Mariana R. Simulador para o ROV LUMA usando Gazebo. Poli Monografias. UFRJ, 2017.

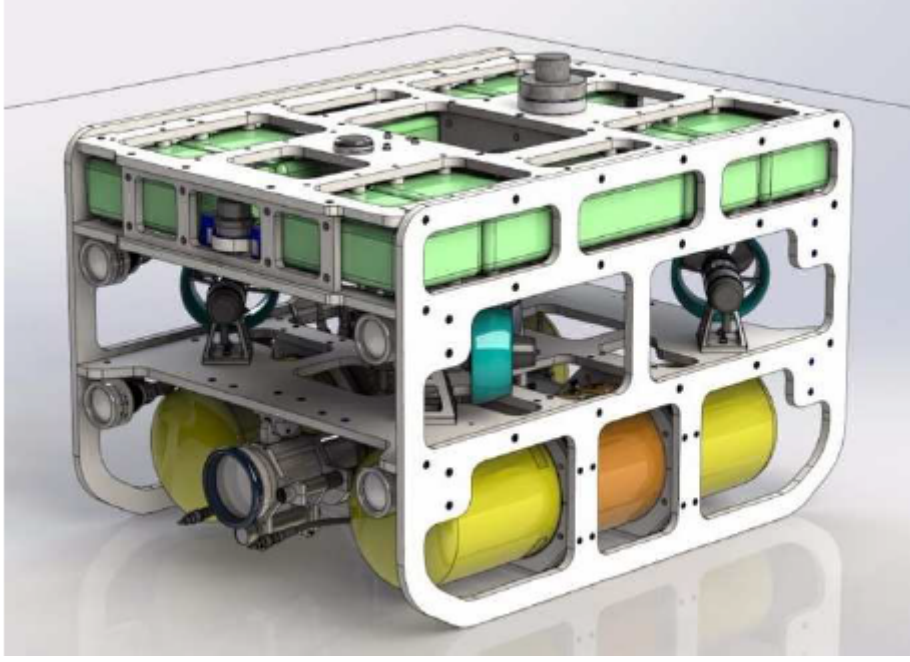


Figure 3.3: View of the LUMA ROV¹.

3.2.2 Static Model of Propellers

A ROV consists of a rigid structure with built-in propellers, whose layout does not follow a strict pattern. There are many models released, which may vary the number of propellers, their arrangement and constructive characteristics.

According to HSU *et al.* (2000), a general static model for a ROV with m propellers produces a thrust f_i in the P_i direction and an axial moment τ_i , which are related by the equation

$$\begin{bmatrix} f_c \\ \tau_c \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^n f_i P_i \\ \sum_{i=1}^n [\tau_i P_i + R_{pi} \times (f_i P_i)] \end{bmatrix} \quad (3.20)$$

where R_{pi} is the thrust position of the i -th propeller with respect to the body frame \mathcal{F}_c , and force f_c and torque τ_c are vectors in \mathbb{R}^3 , in the form

$$f_c = \begin{bmatrix} f_x \\ f_y \\ f_z \end{bmatrix}, \quad \tau_c = \begin{bmatrix} \tau_\phi \\ \tau_\theta \\ \tau_\psi \end{bmatrix} \quad (3.21)$$

In HSU *et al.* (2000), the static model of the blades is described, such that the forces f_c and the angular velocity of the blades ω_i presented in the model are related by

$$f_i = C_T^*(\sigma) \frac{\rho}{8} [v_{wi}^2 + (0.7\pi\omega_i D^2)] \pi D^2 \quad (3.22)$$

where $\sigma = \arctan [v_{wi}/(0.7\pi\omega_i D)]$, v_{wi} is the water speed that goes through the disk

of the i -th blade, ρ is the water density and D is the blade diameter.

In order to calculate the thrust coefficient $C_T^*(\sigma)$, a linear interpolation utilizes the angle σ and a table with values of σ and thrust coefficients collected experimentally. In this experiment, the propeller actuators receive input currents in such a way that the rotors rotate in both directions. The least squares method correlate the measured thrusts generated by the i -th propeller at a rotation speed ω_i . This relation is not straightforward, but rather ω_i is quadratic and is related to f_i by a factor α , such that

$$f_i = \alpha^* \omega_i^2; \quad \alpha^* = \begin{cases} \alpha^+, & \text{if } \omega_i \geq 0 \\ \alpha^-, & \text{if } \omega_i < 0 \end{cases} \quad (3.23)$$

where α^+ and α^- are the thrust coefficients of the blade with respect to the direct and reverse direction of rotation, respectively. These coefficients are utilized to calculate the limit values $C_T^*(0^\circ)$ and $C_T^*(180^\circ)$ of the thrust coefficient, given by the equations

$$\begin{aligned} C_T^*(0^\circ) &= 8\alpha^+ / (\rho 0.7^2 \pi^3 D^4) \\ C_T^*(180^\circ) &= 8\alpha^- / (\rho 0.7^2 \pi^3 D^4) \end{aligned} \quad (3.24)$$

and utilized in a linear interpolation to compute $C_T^*(\sigma)$.

In this formulation, for simplification purposes, the cross coupling due to interference in the water flow from a propeller to another is neglected, and also, the axial component of the velocity of the water which enters a propeller is equal the component of the relative velocity of the ROV, parallel to the helix rotation axis (HSU *et al.*, 2000). Other dynamics neglected are the moments generated by each propeller, since they are small when compared to the thrusts; the rotor dynamics, since the motion of the ROV is too slow when compared to the rotor velocities; and also the dead-zone in the actuators.

3.2.3 Dynamics

From the Newton-Lagrange Equations, the dynamics of the ROV can be written as

$$\begin{cases} (M_{CR} + M_A)\ddot{\chi} + C(\dot{\chi})\dot{\chi} + D(\chi) + G(\chi) = \nu \\ \dot{\eta} = R_r^{-1}(\eta)\omega \end{cases} \quad (3.25)$$

where M_{CR} is the rigid body inertia matrix due to the rigid body dynamics, M_A is the inertia matrix due to additional mass, C is the centripetal and Coriolis forces, D is the matrix due to friction, G represents the buoyant and gravitational forces and ζ is the dynamics due to the umbilical cable.

For simplicity, hydrodynamic damping and the umbilical cable dynamics are

neglected. Consider also that the ROV has symmetry with respect to the plane xz and yz and \mathcal{F}_c coincides with the center of mass. A more complete modeling of a ROV can be consulted on HSU *et al.* (2000) and a model of LUMA can be consulted on GOULART (2007).

The M_{CR} matrix is composed by block submatrices, such as

$$M_{CR} = \begin{bmatrix} M_{11} & 0 \\ 0 & M_{22} \end{bmatrix} \quad (3.26)$$

and its submatrices are given by

$$M_{21} = \begin{bmatrix} 0 & -Mz_G & 0 \\ Mz_G & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad M_{22} = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix} \quad (3.27)$$

where M is the mass of the ROV. On the other hand, the additional inertia M_A is composed by the submatrices

$$M_A = \begin{bmatrix} A_T & A_{RT} \\ A_{RT}^T & A_R \end{bmatrix} \quad (3.28)$$

whose elements are

$$A_T = - \begin{bmatrix} A_{11} & 0 & 0 \\ 0 & A_{12} & 0 \\ 0 & 0 & A_{13} \end{bmatrix}, \quad A_{RT} = - \begin{bmatrix} 0 & A_{21} & 0 \\ A_{22} & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad (3.29)$$

$$A_R = - \begin{bmatrix} A_{31} & 0 & 0 \\ 0 & A_{32} & 0 \\ 0 & 0 & A_{33} \end{bmatrix} \quad (3.30)$$

and each element is the additional mass with respect to an axis for motion on it or with respect to another axis, and $p_G \in \mathbb{R}^3$ stands for the position of the gravity center of the vehicle with respect to the body frame \mathcal{F}_c and can be written as

$$p_G = \begin{bmatrix} x_G \\ y_G \\ z_G \end{bmatrix} \quad (3.31)$$

The Coriolis and Centripetal forces are calculated based on the rigid body dynamics C_1 and also on the additional mass C_2 , which form the matrix

$$C(\dot{\chi}) = \begin{bmatrix} 0_{3 \times 3} & C_1 \\ -C_1^T & C_2 \end{bmatrix} \quad (3.32)$$

whose submatrices are given by

$$C_1 = \begin{bmatrix} M(y_G\omega_y + z_G\omega_z) & -M(x_G\omega_y - \dot{z}) & -M(x_G\omega_z + \dot{y}) \\ -M(y_G\omega_x + \dot{z}) & M(z_G\omega_z + x_G\omega_x) & -M(y_G\omega_z - \dot{x}) \\ -M(z_G\omega_x - \dot{y}) & -M(z_G\omega_y + \dot{x}) & M(x_G\omega_x + y_G\omega_y) \end{bmatrix} \quad (3.33)$$

$$C_2 = \begin{bmatrix} 0 & -I_{yz}\omega_y - I_{xz}\omega_x + I_{zz}\omega_z & I_{yz}\omega_z + I_{xz}\omega_1 - I_{yy}\omega_y \\ I_{yz}\omega_y + I_{xz}\omega_x - I_{zz}\omega_z & 0 & -I_{xz}\omega_z - I_{xy}\omega_y + I_{xx}\omega_x \\ -I_{yz}\omega_z - I_{xz}\omega_x + I_{yy}\omega_y & I_{xz}\omega_z + I_{xy}\omega_y - I_{xx}\omega_x & 0 \end{bmatrix} \quad (3.34)$$

where $\omega_c \in \mathbb{R}^3$ stands for $\omega_c = \begin{bmatrix} \omega_x & \omega_y & \omega_z \end{bmatrix}^T$ and corresponds to the angular velocity with respect to the body frame \mathcal{F}_c .

There exists a force that arises on immersed bodies called buoyancy force, which pushes the body towards the surface. The buoyancy force can be intensified by means of built-in floats to increase the displaced water volume. Another force common to all rigid bodies is gravitational, which is exerted on the contrary direction. Thus, the gravitational and buoyant forces are given by

$$G = \begin{bmatrix} G_1 \\ G_2 \end{bmatrix} \quad (3.35)$$

and the submatrices consist of

$$G_1 = R_c^{-1} \begin{bmatrix} 0 & 0 & (Mg - \rho g \nabla V) \end{bmatrix}^T \quad (3.36)$$

$$G_2 = p_G \times \left(R_c^{-1} \begin{bmatrix} 0 & 0 & Mg \end{bmatrix}^T \right) - p_B \times \left(R_c^{-1} \begin{bmatrix} 0 & 0 & \rho g \nabla V \end{bmatrix}^T \right) \quad (3.37)$$

where $R_c \in SO(3)$ is the rotation matrix relative to the ZYX-Euler angle described in equation A.6, ρ is the water density, ∇V is the displaced water volume. The parameter $p_B \in \mathbb{R}^3$ is the position of the buoyancy center (HSU *et al.*, 2000), given by

$$p_B = \begin{bmatrix} x_B \\ y_B \\ z_B \end{bmatrix} \quad (3.38)$$

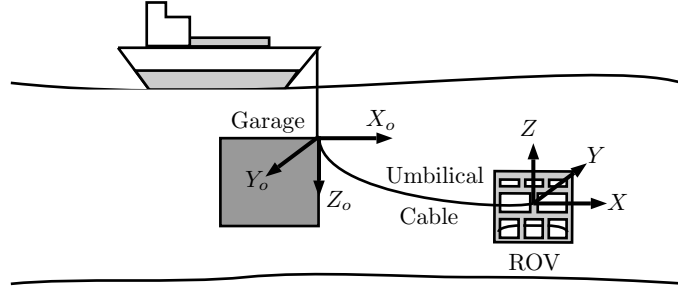


Figure 3.4: A submersible ROV connected by a umbilical cable to the inertial frame .

3.2.4 LUMA

LUMA has four propellers fixedly arranged at the bottom of its structure with orientation R_i and placed at P_i with respect to the body frame \mathcal{F}_c . These propellers are responsible for producing thrusts on the axis X and Y , as depicted in 3.5.

The ROV LUMA also has one additional propeller, responsible for generating thrust for controlling its depth with respect to the Z axis. However, since no control allocation can be performed to control its depth, this propeller will be neglected.

However, the moments exerted by the propellers will be neglected since they are too small when compared to the forces. Thus, the total forces and moments exerted on LUMA due to the force generated by each propeller is given by

$$\nu = \begin{bmatrix} f_x \\ f_y \\ \tau_\psi \end{bmatrix} = Bu \quad (3.39)$$

where $u = [f_1 \ f_2 \ f_3 \ f_4]^T$ and the mapping between forces and torques acting on the body frame and those produced by each propeller is performed by the uncoupling matrix B (GOULART, 2007), given by

$$B = \begin{bmatrix} P_1 & P_2 & P_3 & P_4 \\ R_1 \times P_1 & R_2 \times P_2 & R_3 \times P_3 & R_4 \times P_4 \end{bmatrix} \quad (3.40)$$

but without the rows corresponding to forces on the Z axis and torques τ_ϕ and τ_θ about the X and Y axis, respectively, which yields a matrix $B \in \mathbb{R}^{3 \times 4}$.

3.2.5 Reference Model

When planning a path, one may desire to drive the ROV from an initial pose to a desired one. However, depending on the control law adopted, it is necessary to generate the desired dynamics of the trajectory between these poses in order to supply the controller with information such as the desired velocity $\dot{\chi}_d$ and acceleration $\ddot{\chi}_d$.

This information can be obtained by means of a reference model, which is based

on the filtering of the input signal and is utilized to obtain its first p time derivatives. This filtering consists of implementing a p -th order filter MONTEIRO (2015).

For a second order filter, the corresponding transfer function is

$$H(s) = \frac{k_{f1}k_{f2}}{s^2 + (k_{f1} + k_{f2})s + k_{f1}k_{f2}} \quad (3.41)$$

with $k_{f1} > 0$ and $k_{f2} > 0$ as design constants to determine the bandwidth. The second order transfer function $H(s)$ can be converted into an observable canonical space state form, given by

$$\begin{cases} \dot{\chi}_{in}(t) &= \begin{bmatrix} -(k_{f1} + k_{f2}) & 1 \\ -k_{f1}k_{f2} & 0 \end{bmatrix} \chi_{in}(t) + \begin{bmatrix} 0 \\ k_{f1}k_{f2} \end{bmatrix} \chi_r(t) \\ \chi_d(t) &= \begin{bmatrix} 1 & 0 \end{bmatrix} \chi_{in}(t) \end{cases} \quad (3.42)$$

where $\chi_r(t)$ is the reference trajectory for a given DoF of the system and χ_{in} corresponds to the internal states of the reference model. The state space representation can be rewritten so as to generate the desired trajectory, velocity and acceleration as output, which yields

$$\begin{cases} \chi_d(t) = \chi_{in,1}(t) \\ \dot{\chi}_d(t) = -(k_{f1} + k_{f2}) \chi_{in,1}(t) + \chi_{in,2}(t) \\ \ddot{\chi}_d(t) = -(k_{f1} + k_{f2}) \dot{\chi}_d + k_{f1}k_{f2} [\chi_r(t) - \chi_{in,1}(t)] \end{cases} \quad (3.43)$$

This model must be implemented for each DoF and hence it is possible to implement different dynamics by setting individual values for the design parameters k_{f1} and k_{f2} .

3.2.6 High-level Controller

The High-level controller can be executed with a PD Controller, which is widely used for control purposes and easy to implement. The proportional gain $K_p \in \mathbb{R}^{n \times n}$ corresponds to the response rate to an actual error signal, whereas the derivative gain $K_d \in \mathbb{R}^{n \times n}$ actuates as an error predictor.

Consider the pose error $e_\chi = \chi_d - \chi$, whose time derivatives yield the error velocity $\dot{e}_\chi = \dot{\chi}_d - \dot{\chi}$ and the acceleration error $\ddot{e}_\chi = \ddot{\chi}_d - \ddot{\chi}$. In order to correct the acceleration, let $\ddot{\chi} = \bar{\nu}$ where $\bar{\nu}$ is given by

$$\bar{\nu} = \ddot{\chi}_d + K_p e_\chi + K_d \dot{e}_\chi \quad (3.44)$$

Thus, the error dynamics becomes

$$\ddot{e}_\chi + K_p e_\chi + K_d \dot{e}_\chi = 0 \quad (3.45)$$

which is asymptotically stable for $K_p > 0$ and $K_d > 0$ according to the Routh-Hurwitz criterion (OGATA, 2008).

In order to enhance the High-level controller, we include in the control law the additional mass (3.26)), the Coriolis and Centripetal forces (3.32) and the effects due to gravity and buoyancy (3.35) for compensation purposes, which yields the following High-level controller:

$$\nu_d = (M_{CR} + M_A)^{-1} [\bar{\nu} - C(\dot{\chi}) - G(\chi)] \quad (3.46)$$

3.2.7 Control Allocation Problem

The ROV LUMA can move to any direction by means of a combination of forces and torque, whose components are summed up to generate the desired forces and torques with respect to the body frame \mathcal{F}_c and hence to impose an acceleration to the ROV, according to the Newton's Second Law (A.63).

The nominal rotation direction of propellers is clockwise, although they can also rotate counterclockwise. However, the maximum thrusts generated at both directions is not symmetric - when clockwise, propellers may generate a maximum force u_{max} , and when counter-clockwise, u_{min} , such that $|u_{max}| > |u_{min}|$. In fact, a desired virtual control input, after processed by the uncoupling matrix B may require the propellers to produce forces u , such that $u \notin \mathbb{U}$.

If every propeller produces a force of equal intensity on its nominal direction, the ROV will move forward because of the vector sum of the forces, which causes the components with respect to X to cancel each other. However, an issue arises when the forces required are not properly distributed among the propellers - there are trajectories that require a propeller to rotate clockwise and hence can reach u_{max} , whereas another propeller is required to rotate reversely and hence can reach u_{min} . Some components of these forces may be supposed to cancel each other, but it is not possible because of the asymmetries $|u_{max}| > |u_{min}|$. Due to saturation, undesired residual thrusts arise as well, that deviates the robot from the desired path.

In the control allocation context, the uncoupling matrix B is the control effectiveness matrix, the control input vector u is the forces produced by the propellers and the desired virtual control input vector $\nu_d = f_{c,d}$ stands for the desired forces required by the High-level controller.

The High-level controller only computes the desired forces $f_{c,d}$, since the moments are much smaller than the forces. The fifth propeller is concerned only with the depth control and therefore it does not contribute to the control allocation problem.

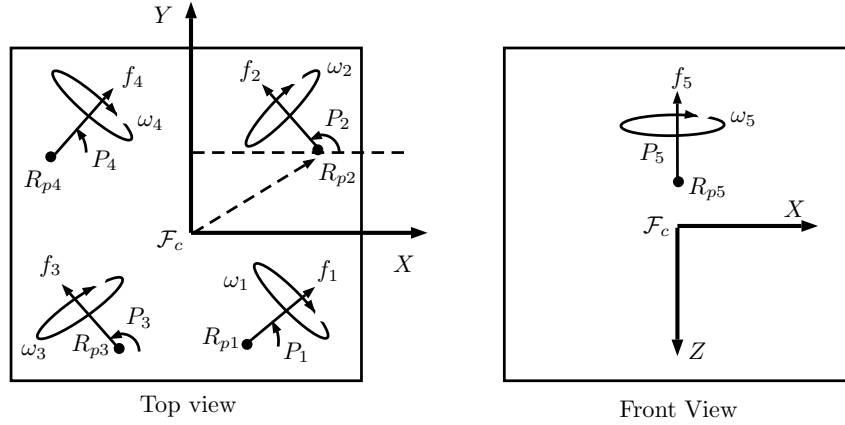


Figure 3.5: Top and frontal view of the ROV LUMA with the arrangement of the propellers, placed at R_{pi} with force direction P_i .

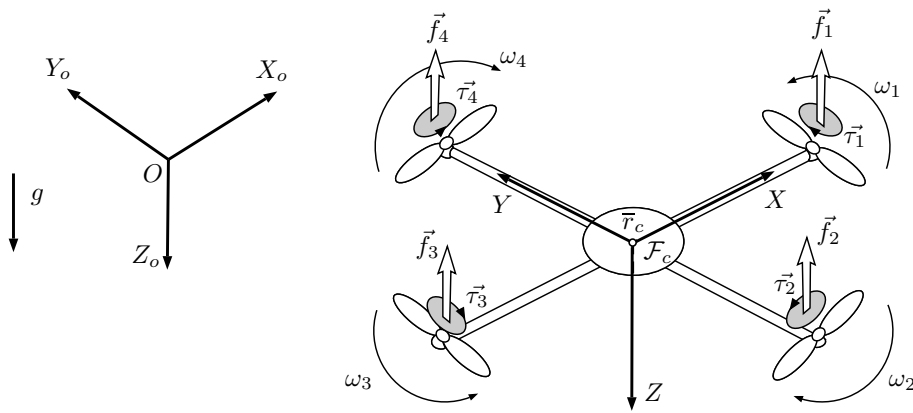


Figure 3.6: Representation of a quadrotor along with the forces and torques generated by its propellers.

3.3 Quadrotors

A quadrotor, also known as quadrotor helicopter or quadcopter, is a type of helicopter with four propellers and rotors situated in an ordered position, laid out in a cross equidistant configuration with respect to its center of mass \bar{r}_c , which contains all the embedded electronics and control boards. The propellers consist of rotational blades, responsible for generating forces and torques due to the Bernoulli's principle and the Newton's Third Law, as depicted in the figure 3.6, and are controlled by adjusting the angular velocities ω_i of their rotors. Every pair of rotors rotate in opposite directions, clockwise and counterclockwise, and hence the need for tail rotors is eliminated. Quadrotors have been employed in search and rescue, surveillance and reconnaissance, and several other applications (BOUABDALLAH, 2007).

3.3.1 Propellers, Forces and Torques

Let \mathcal{F}_o be the inertial frame with the axis Z_o pointing down and \mathcal{F}_c the body frame, whose origin is located in center of mass \bar{r}_c and the Z axis points down as well. The rotation matrix R_c stands for the orientation of the frame \mathcal{F}_c with respect to \mathcal{F}_o , such that $R_c \in SO(3)$. The position of the quadrotor is defined as p and its attitude in the XYZ-Euler angles as η , both with respect to the inertial frame \mathcal{F}_o . Therefore, the full configuration of the quadrotor is given by

$$\chi = \begin{bmatrix} p \\ \eta \end{bmatrix} \quad (3.47)$$

The figure 3.6 illustrates the quadrotor and its four propellers. Each propeller rotates at an angular velocity ω_i , and as consequence, it generates a force f_i upwards and a torque τ_i about the rotor axis. Assuming that the propellers can rotate at a unique direction, the forces and torques are related to their respective angular velocities (BOUABDALLAH, 2007) according to

$$\begin{aligned} f_i &= \kappa_t \omega_i^2 \\ \tau_i &= \kappa_d \omega_i^2 \end{aligned} \quad (3.48)$$

where $\kappa_t > 0$ is the rotor thrust coefficient and $\kappa_d > 0$ is the rotor drag coefficient.

It is important to notice that the robot has four control inputs, although its full configuration is given by the generalized position vector $\chi \in \mathbb{R}^6$. It implies that the system is underactuated and it is not possible to control every DoF directly.

To control the altitude, a thrust T_z is produced on the Z axis by the sum of the forces generated by each propeller, according to the equation

$$T_z = \kappa_t \sum_{i=1}^4 \omega_i^2 \quad (3.49)$$

On the other hand, in order to control the attitude and to produce the torques τ_ϕ , τ_θ and τ_ψ about the axis X , Y and Z , respectively, the following formulas are given

$$\begin{bmatrix} \tau_\phi \\ \tau_\theta \\ \tau_\psi \end{bmatrix} = \begin{bmatrix} l\kappa_t (-\omega_2^2 + \omega_4^2) \\ l\kappa_t (-\omega_1^2 + \omega_3^2) \\ \sum_{i=1}^4 \tau_i \end{bmatrix} \quad (3.50)$$

where l represents the distance between the center of mass of the vehicle and any propeller. Both equations for thrust and torques can be reunited in the matricial

form

$$\underbrace{\begin{bmatrix} T_z \\ \tau_\phi \\ \tau_\theta \\ \tau_\psi \end{bmatrix}}_{\nu} = \underbrace{\begin{bmatrix} \kappa_t & \kappa_t & \kappa_t & \kappa_t \\ 0 & -l\kappa_t & 0 & l\kappa_t \\ l\kappa_t & 0 & -l\kappa_t & 0 \\ -\kappa_d & \kappa_d & -\kappa_d & \kappa_d \end{bmatrix}}_B \underbrace{\begin{bmatrix} \omega_1^2 \\ \omega_2^2 \\ \omega_3^2 \\ \omega_4^2 \end{bmatrix}}_u \quad (3.51)$$

which performs a transformation between the thrust and torques, here denoted as the virtual control input ν , and the propeller velocities squared ω_i^2 , namely control input u , in the context of control allocation.

3.3.2 Gyroscopic Effects

When a torque is applied perpendicularly to an axis of a rotating body and to its angular momentum, the gyroscopic effect appears as a reaction against this torque, which makes the rigid body rotate about an axis perpendicular to both torque and momentum, according to the conservation of angular momentum. The faster the body rotates, the stronger is this reaction.

Let ω_R be the residual angular velocity of the rotors, such that

$$\omega_R = \omega_1 + \omega_3 - \omega_2 - \omega_4 \quad (3.52)$$

In the quadrotor, the gyroscopic effect due to the rotation of propellers (BOUAB-DALLAH, 2007) is given by

$$\begin{cases} \tau'_\phi = I_r \omega_y \omega_R \\ \tau'_\theta = -I_r \omega_x \omega_R \end{cases} \quad (3.53)$$

where I_r is the rotor inertia about its own axis. In the equation (A.63), it is represented by the Coriolis force. On the other hand, τ'_ψ is an induced torque due to a variation in the rotation speed, also known as *inertial counter torque* (BOUAB-DALLAH, 2007), which is

$$\tau'_\psi = I_r \dot{\omega}_R \quad (3.54)$$

3.3.3 Rotor Dynamics

In the quadrotor, the propellers are actuated by brushless DC motors, which are controlled by tension V_i and its output variable is the rotor velocity ω_i . According to MONTEIRO (2015), the relation output-input can be approximated by a first-order differential equation, similar to the equation

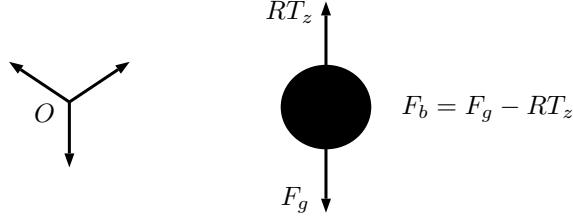


Figure 3.7: Free body diagram of a quadrotor, which is represented as a concentrated mass.

$$\frac{d\omega_i}{dt} = -\frac{1}{\tau_m}\omega_i + \frac{K_m}{\tau_m}U_i - Q_i \quad (3.55)$$

with $K_m > 0$ the DC gain of the motor, $\tau_m > 0$ is the time constant, $U_i \in \mathbb{R}$ is the input signal and Q_i is the torque on the motor axis.

3.3.4 Newton-Euler Equations of a Quadrotor

Consider the Newton-Euler equation A.63 in the quadrotor context. The forces that act on the vehicle are F_g due to the gravitational acceleration and thrust T_z , as shown in figure 3.7. The Centrifugal and Coriolis forces disappear, since the dynamic system is analyzed from the inertial frame. Therefore the dynamic model results in

$$\left\{ \begin{array}{l} M\ddot{x} = (\sin \psi \sin \phi + \cos \psi \sin \theta \cos \phi)T_z \\ M\ddot{y} = (\sin \psi \sin \theta \cos \phi - \cos \psi \sin \phi)T_z \\ M\ddot{z} = Mg - (\cos \psi \cos \phi)T_z \\ I_{xx}\dot{\omega}_x = \omega_z\omega_y (I_{yy} - I_{zz}) + \tau'_\phi + \tau_\phi \\ I_{yy}\dot{\omega}_y = \omega_x\omega_z (I_{zz} - I_{xx}) - \tau'_\theta + \tau_\theta \\ I_{zz}\dot{\omega}_z = \omega_y\omega_x (I_{xx} - I_{yy}) + \tau'_\psi + \tau_\psi \\ \dot{\eta} = R_r^{-1}(\eta)\omega \end{array} \right. \quad (3.56)$$

An aspect to consider is that T_z is coupled with respect to motion on the X , Y and Z axis because of the pitch and roll angles. This coupling disappears only in the quasi-stationary flight, where these angles become so small that can be neglected, and therefore, T_z lies mostly on the Z direction. For motion with respect to the X and Y axis, a pitching or rolling motion is required to change the direction of T_z and produce thrust components on the X and Z axis, in response to a deviation in the x_d and y_d references, respectively (BOUABDALLAH, 2007). The dynamic model can be written in the control allocation context along with a small angle approximation, which results in

$$\left\{ \begin{array}{l} M\ddot{x} = (\phi \sin \psi + \theta \cos \psi)\nu_1 \\ M\ddot{y} = (\theta \sin \psi - \phi \cos \psi)\nu_1 \\ M\ddot{z} = Mg - \nu_1 \\ I_{xx}\dot{\omega}_x = \omega_z\omega_y(I_{yy} - I_{zz}) + \tau'_\phi + \nu_2 \\ I_{yy}\dot{\omega}_y = \omega_x\omega_z(I_{zz} - I_{xx}) - \tau'_\theta + \nu_3 \\ I_{zz}\dot{\omega}_z = \omega_y\omega_x(I_{xx} - I_{yy}) + \tau'_\psi + \nu_4 \\ \dot{\eta} = R_r^{-1}(\eta)\omega \end{array} \right. \quad (3.57)$$

Despite friction, hub forces, rolling moments and other aerodynamic effects action were neglected in this dynamic model, it delivers a good approximation to accomplish with the purpose of this dissertation. More detailed concerning this model can be consulted on MONTEIRO (2015) and a more enhanced model can be found in BOUABDALLAH (2007).

3.3.5 High-level Controller

Consider the mapping between the virtual control inputs and the propellers velocities in the equation 3.51. The equations relate thrust and their components on the Z , Y and X axis, and their respective torques τ_ϕ , τ_θ and τ_ψ . Thus, the control can be split into two steps in a cascaded approach, where the first is responsible for generating the reference signals for the second step and for computing the thrust T_z to control the altitude. In the second step, the reference signals calculated are utilized to compute the torques τ_ϕ , τ_θ and τ_ψ to control the attitude and the position of the vehicle.

The altitude controller can be straightforwardly undertaken with a PD Controller, which is widely used for this control purpose and easy to implement. The proportional gain $k_{p,z}$ corresponds to the response rate to an actual error signal, whereas the derivative gain $k_{d,z}$ actuates as an error predictor.

Consider the altitude error $e_z = z_d - z$, whose derivatives yield the error velocity $\dot{e}_z = \dot{z}_d - \dot{z}$ and the acceleration error $\ddot{e}_z = \ddot{z}_d - \ddot{z}$. Let $\ddot{z} = \bar{\nu}_{d,1}$ for implementing a PD Controller in the form

$$\bar{\nu}_{d,1} = \ddot{z}_d + k_{p,z}e_z + k_{d,z}\dot{e}_z \quad (3.58)$$

The PD Controller for the altitude control yields the error dynamics

$$\ddot{e}_z + k_{p,z}e_z + k_{d,z}\dot{e}_z = 0 \quad (3.59)$$

which is asymptotically stable for $k_{p,z} > 0$ and $k_{d,z} > 0$, according to the Routh-Hurwitz criterion (OGATA, 2008). With the gravity compensation, the control law

becomes

$$\begin{aligned}\nu_{d,1} &= \bar{\nu}_{d,1} + Mg \\ &= \ddot{z}_d + k_{p,z}e_z + k_{d,z}\dot{e}_z + Mg\end{aligned}\quad (3.60)$$

On the other hand, the position control concerning the x and y coordinates is not directly performed. From the rolling and pitching, the PD Controller proposed in MONTEIRO *et al.* (2016a) calculates the desired values ϕ_d and θ_d for the inner loop, implemented as

$$\begin{bmatrix} \bar{\phi}_d \\ \bar{\theta}_d \end{bmatrix} = K_p \begin{bmatrix} x_d - x \\ y_d - y \end{bmatrix} + K_d \begin{bmatrix} \dot{x}_d - \dot{x} \\ \dot{y}_d - \dot{y} \end{bmatrix} + g^{-1} \begin{bmatrix} \ddot{x}_d \\ \ddot{y}_d \end{bmatrix}\quad (3.61)$$

where $K_p = \text{diag}(k_{p,x}, k_{p,y})$ and $K_d = \text{diag}(k_{d,x}, k_{d,y})$ are positive definite proportional and derivative gain matrices, respectively. The desired values for pitch and yaw are finally obtained by rotating the desired coordinates found in 3.61 in a small-angle approximation, calculated according to

$$\begin{cases} \phi_d = (\phi \sin \psi + \theta \cos \psi) \bar{\phi}_d \\ \theta_d = (\theta \sin \psi - \phi \cos \psi) \bar{\theta}_d \end{cases}\quad (3.62)$$

For the attitude control, in BOUABDALLAH (2007) the author proposes an inner loop control with a Integral Backstepping, whose approach is utilized to control the dynamics of the angles θ , ϕ and ψ . Consider a roll error $e_1 = \phi_d - \phi$ and its derivative $\dot{e}_1 = \dot{\phi}_d - \omega_x$. Now we assume that ω_x can be a control input, whose desired value is given by

$$\omega_{x,d} = \dot{\phi}_d + k_1 e_1 + k_2 \bar{e}_1, \quad \text{with } \bar{e}_1 = \int_{t_0}^t e_1(\sigma) d\sigma\quad (3.63)$$

for stabilizing the e_1 dynamics, and $k_1 > 0$ and $k_2 > 0$ are proportional and integral gains, respectively. Moreover, we compute the angular velocity error $e_2 = \omega_{x,d} - \omega_x$ and its time derivative, given by

$$\begin{aligned}\dot{e}_2 &= \dot{\omega}_{x,d} - \dot{\omega}_x \\ &= \ddot{\phi}_d + k_1 \dot{e}_1 + k_2 e_1 - \dot{\omega}_x\end{aligned}\quad (3.64)$$

Thus, the dynamics of e_1 can be written as function of e_2 , which turns out to be

$$\dot{e}_1 = e_2 - k_1 e_1 - k_2 \bar{e}_1\quad (3.65)$$

This equation is substituted in equation 3.64 along with the dynamics of ω_x from

the equation 3.57, which yields

$$\dot{e}_2 = \ddot{\phi}_d + k_1(e_2 - k_1 e_1 - k_2 \bar{e}_2) + k_2 e_1 - \omega_y \omega_z \frac{(I_{yy} - I_{zz})}{I_{xx}} - \frac{\tau'_\phi}{I_{xx}} - \frac{u_2}{I_{xx}} \quad (3.66)$$

Now, the desired dynamics for the angular velocity error e_2 is $\dot{e}_2 = -k_3 e_2 - e_1$, such that $k_3 > 0$ is also a controller design constant. Thus, by equating it with (3.66), the control u_2 results in

$$\nu_{d,2} = I_{xx} \left[e_1(1 - k_1^2 + k_2) + e_2(k_3 + k_1) - k_1 k_2 \bar{e}_1 + \ddot{\phi}_d \right] - \omega_x \omega_y (I_{yy} - I_{zz}) - \tau'_\phi \quad (3.67)$$

Similarly, the steps for calculating the virtual control input $\nu_{d,2}$ can be repeated for $\nu_{d,3}$ and $\nu_{d,4}$, which yield the virtual control inputs

$$\begin{cases} \nu_{d,3} = I_{yy} \left[e_3(1 - k_4^2 + k_5) + e_4(k_6 + k_4) - k_4 k_5 \bar{e}_3 + \ddot{\theta}_d \right] - \omega_z^b \omega_x^b (I_{zz} - I_{xx}) + \tau'_\theta \\ \nu_{d,4} = I_{zz} \left[e_5(1 - k_7^2 + k_8) + e_6(k_9 + k_7) - k_7 k_8 \bar{e}_5 + \ddot{\psi}_d \right] - \omega_x^b \omega_y^b (I_{xx} - I_{yy}) - \tau'_\psi \end{cases} \quad (3.68)$$

These control laws look similar to a PID control, although it compensates the gyroscopic effects and the inertial counter torques in a feedback linearization strategy.

3.3.6 Control Allocation Problem

The quadrotor is an underactuated system with 6 DoF and only 4 actuators. Provided that the system lacks two actuators to become fully actuated, the control of a quadrotor tends to present a high sensibility when subject to an input constraint, such as saturation.

The quadrotor is highly subject to saturation, specially in the initial instants if it starts from a still position. Therefore, control allocation is necessary for setting suitable controls for each rotor, so that it generates the best possible forces and torques for tracking purposes, while presenting an acceptable performance and respecting the constraints imposed by the minimal and maximal angular velocities and hence the maximum and minimum forces and torques.

3.4 Robotic Manipulators

Manipulators are a fixed-base class of robot, which consists of joints and links sequentially interconnected. Their joints grant the manipulator the ability to perform

a required task within its operational space and can be prismatic or revolute. The prismatic joint creates a relative translational motion, whereas the revolute creates a rotational one between two links.

A manipulator is chosen according to the required task. For instance, a planar manipulator has 3 DoF and can perform a positioning task, while a 6-DoF manipulator is demanded for both position and orientation in the three-dimensional space. However, a 4-DoF planar manipulator for a positioning task has more additional DoF available than the strictly needed. If more DoFs than task variables are available, the manipulator is called redundant (SICILIANO *et al.*, 2009). At the end of the sequence of joints and links there is an end effector, responsible for accessing the environment inside the workspace.

3.4.1 Forward Kinematics

The configuration of an end effector can be described by

$$\chi = \begin{pmatrix} p_e \\ \eta_e \end{pmatrix} \in SE(3), \quad (3.69)$$

where p_e is the position of the end effector with respect to \mathcal{F}_o , η_e is its orientation and $SE(3)$ is the 3-dimensional special Euclidean group, as defined in (A.27). Given a manipulator with a total of n joints, let $q \in \mathbb{R}^n$ be with the position of the joints. The mapping between q and the end effector configuration χ is denoted forward kinematics and is represented by

$$\chi = k(q) \quad (3.70)$$

This relation can be obtained by fixing a coordinate frame at each link and calculating its respective homogeneous transformation matrix (A.29) in order to obtain its relative position and orientation. The transformation matrices sweeps all the robot, from its inertial frame to the base, then passing by every joint sequentially until the end effector, which results in a transformation matrix from the inertial frame to the end effector, given by

$$\chi = {}^bT_0 {}^0T_n(q) {}^nT_e(q) \quad (3.71)$$

For computing 0T_n , a common approach is the Denavit-Hartenberg convention, which based on establishing a relative position and orientation of two consecutive links (SICILIANO *et al.*, 2009). In this convention, coordinate frames X_i and Y_i are fixed at the joint i and are related by the parameters a_i for the link length, link twist α_i , link offset d_i and link angle θ_i , as illustrated in the figure 3.8. Thus, as defined in

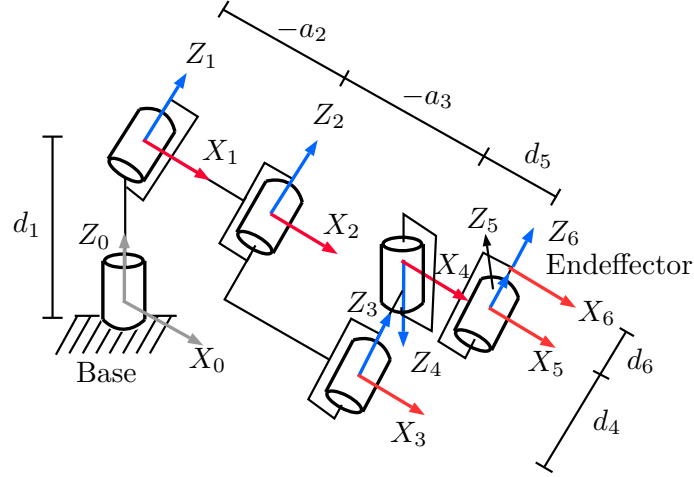


Figure 3.8: Denavit-Parameters at the zero position of the UR5.

CRAIG (2004), the transformation matrix obtained from the Denavit-Hartenberg convention from the link $i-1$ to i is given by

$${}^{i-1}T_i = \begin{bmatrix} c\theta_i & -s\theta_i & 0 & a_{i-1} \\ s\theta_i c\alpha_{i-1} & c\theta_i c\alpha_{i-1} & -s\alpha_{i-1} & -s\alpha_{i-1}d_i \\ s\theta_i s\alpha_{i-1} & c\theta_i s\alpha_{i-1} & c\alpha_{i-1} & c\alpha_{i-1}d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.72)$$

where $c\theta_i = \cos \theta_i$, $c\alpha_{i-1} = \cos \alpha_{i-1}$, $s\theta_i = \sin \theta_i$ and $s\alpha_{i-1} = \sin \alpha_{i-1}$.

Thus, by defining a task, it is possible to plan it whether as function of q , also called joint space, or as function of χ , namely operational space.

3.4.2 Differential Kinematics

The linear and angular velocities of an end effector, \dot{p} and ω respectively, are related to the generalized velocities \dot{q} by means of a matrix $J(q) \in \mathbb{R}^{6 \times n}$ called *geometrical Jacobian*. This matrix is very important, since it can be used to analyze singularities, redundancies and determining the inverse kinematics (SICILIANO *et al.*, 2009). The Jacobian J can be split into two block matrices J_p and J_o , where $J_p \in \mathbb{R}^{3 \times n}$ is denoted position Jacobian and $J_o \in \mathbb{R}^{3 \times n}$ is the orientation Jacobian. Thus, the joint velocities \dot{q} and end effector velocities are related by

$$\begin{bmatrix} \dot{p}_e \\ \omega_e \end{bmatrix} = \begin{bmatrix} J_p(q) \\ J_o(q) \end{bmatrix} \dot{q} \quad (3.73)$$

Consider a coordinate frame \mathcal{F}_i placed at the joint i and consider also the end effector frame \mathcal{F}_e . The contribution of the joint i to the velocities of the end effector is given by an adjoint transformation matrix (MURRAY *et al.*, 1994), given by

$$\begin{bmatrix} \dot{p}_e \\ \omega_e \end{bmatrix} = \begin{bmatrix} \mathcal{I} & -{}^i p_e \times \\ 0 & \mathcal{I} \end{bmatrix} \begin{bmatrix} \dot{p}_i \\ \omega_i \end{bmatrix} \quad (3.74)$$

Let \vec{h}_i be a unit vector corresponding to the axis of the joint i . Since there are two different joint types, prismatic and revolute, the following relations can be established:

$$\begin{bmatrix} \dot{p}_i \\ \omega_i \end{bmatrix} = \begin{cases} \begin{bmatrix} 0 \\ \vec{h}_i \end{bmatrix} \dot{q}_i, & \text{if } i \text{ is revolute} \\ \begin{bmatrix} \vec{h}_i \\ 0 \end{bmatrix} \dot{q}_i, & \text{if } i \text{ is prismatic} \end{cases} \quad (3.75)$$

By replacing (3.74) into (3.73), the adjoint matrix is adapted to consider the possible motion of the prismatic and revolute joints, which results in

$$\begin{bmatrix} \dot{p}_e \\ \omega_e \end{bmatrix} = \begin{bmatrix} J_{ip}(q) \\ J_{io}(q) \end{bmatrix} \dot{q}_i = \begin{cases} \begin{bmatrix} \vec{h}_i \times {}^i p_e \\ \vec{h}_i \end{bmatrix} \dot{q}_i, & \text{if } i \text{ is revolute} \\ \begin{bmatrix} \vec{h}_i \\ 0 \end{bmatrix} \dot{q}_i, & \text{if } i \text{ is prismatic} \end{cases} \quad (3.76)$$

where J_{ip} performs the mapping from the joint i to the linear velocity of the end effector, and J_{io} , to the angular velocity.

The adapted revolute and prismatic adjoint matrices can be combined to consider all the joints and their contributions to motion. In a manipulator with n joints, $J(q)$ is defined as

$$\begin{bmatrix} \dot{p}_e \\ \omega_e \end{bmatrix} = \begin{bmatrix} J_{1p} & J_{2p} & \cdots & J_{np} \\ J_{1o} & J_{2o} & \cdots & J_{no} \end{bmatrix} \dot{q} \quad (3.77)$$

An important issue about $J(q)$ arises when it loses rank, i.e two or more columns become linearly dependent. It implies that $J(q)$ cannot be inverted and the manipulator has reached a singular condition. In another words, the correspondence between the velocities \dot{q} and $\dot{\chi}$ is lost and the end effector cannot move to certain directions.

As a matter of fact, the vicinity of a singular position must be avoided - as the joint angles are driven to a singular configuration, the control over one of its DoF is lost and the task to which it is designed cannot be fulfilled. When implementing a control law, it is common also to perform a mapping from the joint space to the end effector space, and vice-versa. For instance, when mapping from the task space

to the joint space, the pseudo-inverse of $J(q)$ leads to elevated joint velocities in response to a minor motion of the end effector. It also affects the forces on the end effector, since it is a result of the transposed Jacobian from the torques on the joints (MURRAY *et al.*, 1994).

3.4.3 Dynamics

From the Euler-Lagrange approach, the kinetic energy of a link is calculated based on the position of its center of mass r_i^c with respect to a link frame \mathcal{F}_i . Let \dot{p}_i^c be the linear velocity of the center of mass in link i . As defined in equation A.50, the total kinetic energy of a manipulator can be obtained from the summation of the kinetic energy in its links, such that

$$\mathcal{T} = \frac{1}{2} \sum_{i=1}^n (m_i \|\dot{p}_i^c\|^2 + \frac{1}{2} \omega_i^T I_i \omega_i) \quad (3.78)$$

where \dot{p}_i^c and ω_i are obtained from the partial geometric Jacobian (3.77), and therefore, they are function of q and \dot{q} . The total kinetic energy \mathcal{T} can be rewritten as

$$\mathcal{T} = \frac{1}{2} \dot{q}^T M(q) \dot{q} \quad (3.79)$$

with $M(q)$ as the inertia matrix of the manipulator (SICILIANO *et al.*, 2009), defined as

$$M(q) = \left[\sum_{i=1}^n (m_i J_{p_i}^T J_{p_i} + J_{o_i}^T R_i I_i R_i J_{o_i}) \right] \quad (3.80)$$

On the other hand, the potential energy, defined in equation A.46, is calculated according to

$$\begin{aligned} \mathcal{U} &= - \sum_{i=1}^n m_i g \cdot {}^o r_i^c(q) \\ &= g(q) \end{aligned} \quad (3.81)$$

Thus, the Lagrangian of the manipulator corresponds to the energy balance of the system, which is given by

$$L(q, \dot{q}) = \frac{1}{2} \dot{q}^T M(q) \dot{q} - g(q) \quad (3.82)$$

By taking the partial derivatives according to equation A.57, it results in the Lagrange equations

$$M(q)\ddot{q} + \dot{M}(q, \dot{q})\dot{q} - \begin{bmatrix} \frac{1}{2}\dot{q}^T M_1(q)\dot{q} \\ \vdots \\ \frac{1}{2}\dot{q}^T M_n(q)\dot{q} \end{bmatrix} + G(q) = \tau \quad (3.83)$$

where τ is the generalized forces on the joints and $G(q)$ is the vector of gravitational forces. This equation corresponds to the manipulator dynamics and can be rewritten in the standard form commonly found in literature

$$M(q)\ddot{q} + C(q, \dot{q})\dot{q} + G(q) = \tau \quad (3.84)$$

where $C(q, \dot{q}) \in \mathbb{R}^n$ represents the centripetal and Coriolis forces. Friction has neglected in this dynamic model, although it is present in every mechanism and increases the torque required to move the manipulator considerably in typical situations (CRAIG, 2004).

In this dynamic model, the contact forces h_f may arise when the end effector interacts with the environment and hence they can be regarded, resulting in

$$M(q)\ddot{q} + C(q, \dot{q})\dot{q} + G(q) = \tau - J^T(q)h_f \quad (3.85)$$

As mentioned, the manipulator can be controlled in the operational space. Thus, its dynamics can be also written in the Cartesian space after some calculation using the differential kinematics presented in the equation 3.77. Then, the dynamic model in the operational space is

$$\bar{M}(\ddot{\chi}) + \bar{C}(\dot{\chi})\dot{\chi} + \bar{G}(\chi) = F - h_f \quad (3.86)$$

such that

$$\begin{cases} \bar{M} = (JM^{-1}J^T)^{-1} \\ \bar{C}\dot{\chi} = \bar{M}JM^{-1}C\dot{q} - \bar{M}\dot{J}\dot{q}; \\ \bar{G} = \bar{M}JM^{-1}G; \\ F = J^{-T}\tau \end{cases}$$

Contact Forces

In practical situations, end effectors are supposed to interact with the environment, and as a result, the environment exerts forces and moments on them, called contact forces. The contact forces play an important role in grasping tasks because it is necessary to balance external wrenches on the payload by applying appropriate wrenches at the contact points (MURRAY *et al.*, 1994), otherwise the end effector will not be able to hold the payload properly. In tasks which demand force control,

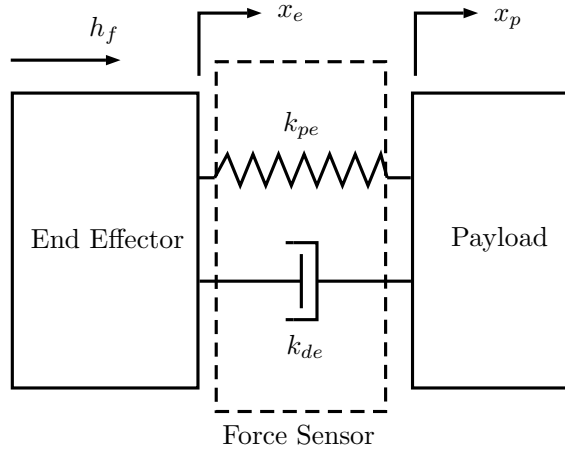


Figure 3.9: Contact forces simulated as a spring-damper model.

such as manipulator-aided surgeries, the actuator must interact with precision on skin and other body tissues. Therefore, contact forces must be measured by sensors built at the tip of the robotic arm, such that the sensors undergo a deformation according to the intensity of interaction.

For the estimation of the contact forces, consider the figure 3.9, which depicts an interaction between an end effector and a rigid payload on the X axis. Between them, there exists a compliant force sensor, which can be modeled as a spring-damper model (EPPINGER and SEERING, 1987), according to the equation

$$h_f = k_{pe}(x_e - x_p) + k_{de}(\dot{x}_e - \dot{x}_p) \quad (3.87)$$

where k_{pe} is the spring constant and k_{de} is the damping constant. Typical values for these constants are $k_{pe} = 10000$ N/m and $k_{de} = 25$ kg/s, which are values commonly found in sensor materials (REN *et al.*, 2016).

Nonetheless, a $x_e > x_p$ implies in an interaction between the end effector and the environment, whereas a $x_e < x_p$ means that no contact forces arise since no interaction occurs. This interaction details can be expressed as a penalty function and can be written as

$$h_f = \max(0, k_{pe}(x_e - x_p) + k_{de}(\dot{x}_e - \dot{x}_p)) \quad (3.88)$$

3.4.4 Cooperative Manipulators

Cooperative Manipulators are referred as two or more manipulators assigned to carry out a certain task. They are used in tasks impossible to be executed by a single manipulator, such as carrying heavy or large payloads. As they are employed cooperatively, the control of interaction in such systems is particularly complex and

has received attention of researchers since the early 1970's.

Many strategies have been adopted for control purposes, such as master/slave control, force/compliance control and task-space control (CACCAVALE *et al.*, 2008). In the task-space control, the control is concerned only with the kinematics, where the cooperative system is described by the absolute and relative motions to the detriment of the interaction forces.

Consider a cooperative system composed by N manipulators, such that each of them contains n_i joints. Let \mathcal{F}_i be the end effector frame located at its tip, whose pose is described by the position $p_i \in \mathbb{R}^3$ and orientation $R_i \in SO(3)$ with respect to an inertial frame \mathcal{F}_o . Consider also that the cooperative system manipulates an object with an inner fixed reference point C , at which is located the object frame \mathcal{F}_c . The end effector frames are connected to \mathcal{F}_c by means of virtual sticks r_i . These sticks simulate an extension of the manipulator, and therefore, they emulate a manipulator link with length r_i , as depicted in figure 3.10. Hence, each manipulator now has a virtual end effector frame $\mathcal{F}_{r,i}$, which coincides with the object frame \mathcal{F}_c . Its position and orientation are given by

$$\begin{cases} p_{r,i} = p_c \\ R_i = R_c \end{cases} \quad (3.89)$$

However, we must take into account that the virtual sticks assumption is valid only if the grasp remains tight and presents no slipping during all the task, in such a way that the displacements between the end effectors and the contact points can be neglected. It implies that the equation 3.89 imposes a kinematic constraint between every manipulator involved in the cooperative task and the object and therefore they can be considered as a system of rigid bodies.

The kinematic constraint can be expanded to include also the positions, velocities and the acceleration (CACCAVALE *et al.*, 2008) of each end effector, which results in

$$\begin{cases} p_i = p_c + R_c r_i \\ R_i = R_c \\ \dot{p}_i = \dot{p}_c - (R_c r_i) \times \omega_c \\ \omega_i = \omega_c \\ \ddot{p}_i = \ddot{p}_c - \omega_c \times (R_c r_i) \times \omega_c - (R_c r_i) \times \dot{\omega}_c \\ \dot{\omega} = \dot{\omega}_c \end{cases} \quad (3.90)$$

Since a manipulator exerts a force on the object, the contact forces at the frame \mathcal{F}_i can be transformed into the object frame \mathcal{F}_c through the adjoint transformation matrix

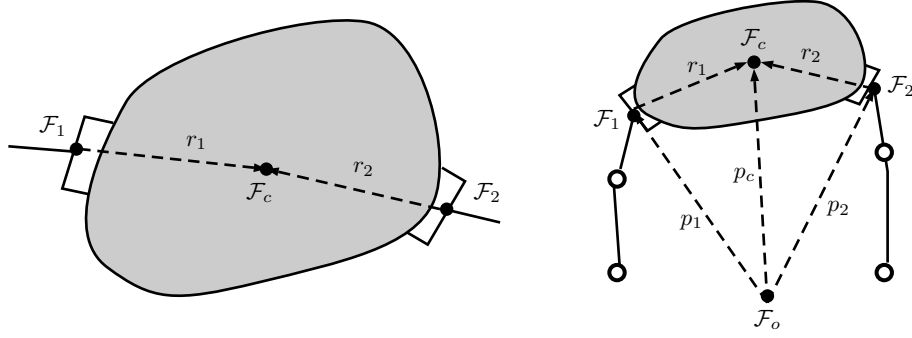


Figure 3.10: Superior and front view of a cooperative system, where the virtual sticks r_i connect the tips of end effectors to the object frame \mathcal{F}_c .

$${}^{r,i}G_i = \begin{bmatrix} \mathcal{I}_3 & 0_{3 \times 3} \\ r_i \times & \mathcal{I}_3 \end{bmatrix} \quad (3.91)$$

The wrench h_o exerted by the manipulators on the object, called external forces, and the N forces exerted by each end effector at the contact point can be related by combining their respective adjoint transformation matrices into a grasp matrix G , such that

$$h_o = \begin{bmatrix} {}^{r,1}G_1 & \dots & {}^{r,N}G_N \end{bmatrix} \begin{bmatrix} h_{f,1} \\ \vdots \\ h_{f,N} \end{bmatrix} = Gh_f \quad (3.92)$$

The grasp matrix G is very important in cooperative tasks, since it relates the individual contact forces with the wrench h_o . However, a remainder of these forces may arise and they do not cause motion at all, rather produce mechanical stresses in the object, namely internal forces h_I (CACCAVALE *et al.*, 2008). These forces belong to the null-space of the grasp matrix, namely $\ker(G)$, and can be mathematically written as

$$\ker(G) = \{h_I | Gh_I = 0\} \quad (3.93)$$

and calculated by the equation

$$h_I = \ker(G) \ker(G)^\dagger h_f \quad (3.94)$$

Topics concerning internal forces have been covered in many works, such as CACCAVALE *et al.* (2008) and REN *et al.* (2016), where the authors limit the internal stress on the object to avoid its damage. On the other hand, to some extent, internal forces are also important for ensuring the grasping and no occurrence of slippage. Therefore, they must be located inside a range in order to satisfy the kinematic constraints and not to cause stresses on the object and on the manipulators.

3.4.5 Impedance Control

In a task, manipulators can be required to interact with the environment. When an end effector exerts a force on the environment, this force is a suitable quantity to describe how the interaction is occurring. Conventional position controllers are employed in simple tasks, like moving an object to a desired position, although they are not suitable since they do not take the interaction forces into account. As a result, elevated contact forces may arise, causing stress and damage to the object and to the end effector (SICILIANO *et al.*, 2009).

Proposed by HOGAN (1984), the impedance control is one of the existing force control approaches and belongs to the category of indirect force control, since it does not control force directly, but rather it utilizes the concept of impedance and admittance. Similar to impedance in electrical systems, the motion of an end effector can be related to a flow of electric charges and the applied force can be related to the electric potential. Both physical systems can be divided into impedance and admittance. In impedance, the motion of the end effector is regarded as an input, or a flow of charges which produces a force or an electric potential as output. In admittance, from a force or an electric potential as input, motion or flow of charges are produced as an output.

From the side of mechanical system, with respect to any DoF, if the end effector acts like impedance, the environment must be an admittance and vice-versa (HOGAN, 1984). However, since the end effector is the element to be controlled, it must present the behavior of an impedance - if the force it exerts is not suitable, the desired position will suffer deviation in a closed loop control system.

Thus, the impedance scheme produces a desired mechanical behavior between the environment deviations and the forces due to interaction described by

$$M_d \ddot{e} + D_d \dot{e} + K_d e = h_{env} \quad (3.95)$$

where $e = \chi - \chi_d$ represents the deviation of the end effector, M_d stands for the desired apparent inertia, D_d is the desired apparent damping and K_d represents the desired environment stiffness, chosen so to achieve the desired compliant behavior of the environment. Moreover, these matrices are positive definite and symmetric.

Concerning a manipulator, consider the dynamic model expressed by the equation 3.86 and a control law given by

$$\tau = M(q)y + C(q, \dot{q}) + G(q) + J^T(q)h_f \quad (3.96)$$

The main idea behind this control law is to perform a global linearization in the dynamic system. As contact forces arise, its dynamics becomes

$$M(q)\ddot{q} + C(q, \dot{q}) + G(q) = M(q)y + C(q, \dot{q}) + G(q) + J^T(q)h_f - J(q)^T h_f \quad (3.97)$$

that after canceling out its terms, both sides of the equation are left-multiplied by $M^{-1}(q)$, which results in

$$\ddot{q} = y \quad (3.98)$$

This equation reveals that any acceleration y can be imposed to the joints, provided that the contact forces can be measured and feedback into the torque-based position controller. Thus, the control law

$$y = J^{-1}(q)M_d^{-1} \left[M_d\ddot{\chi} + D_d\dot{e}_\chi + K_d e_\chi - M_d\dot{J}(q, \dot{q})\dot{q} - h_{env} \right] \quad (3.99)$$

will result in the dynamics described by the equation 3.95 in a closed loop dynamic system. Nevertheless, it is important to emphasize that this implementation does not perform a force control, but rather the impedance is controlled.

Given that h_{env} cannot be directly measured, the equation 3.95 needs to be transformed to include the object dynamics. Given that $\{\chi_d, \dot{\chi}_d, \ddot{\chi}_d\}$ denotes the desired object trajectory, the inverse dynamics of the object is given by

$$M_o\ddot{\chi}_d + D_o\dot{\chi}_d + G(\chi_d) = h_o - h_{env} \quad (3.100)$$

where M_o is the object inertia matrix, $D_o\dot{\chi}$ is the vector of the centrifugal and Coriolis force and $G(\chi)$ is the force due to gravitation. This equation is solved for h_{env} and substituted in the equation 3.95, which results in

$$M_d\ddot{e} + D_d\dot{e} + K_d e = h_o - M_o\ddot{\chi}_d - D_o\dot{\chi}_d - G(\chi_d) \quad (3.101)$$

A remaining problem consists of obtaining an accurate object acceleration $\ddot{\chi}$. This issue may be solved by using the last acceleration $\ddot{\chi} = \ddot{\chi}_l$, or using the desired acceleration $\ddot{\chi}_d$ (SCHNEIDER and CANNON, 1992). In their work, both approaches granted acceptable experimental results. With the last acceleration $\ddot{\chi}_l$, the equation of the impedance controller results in

$$\ddot{\chi}_c = \ddot{\chi}_d + M_d^{-1} [h_o - M_o\ddot{\chi}_l - D_o\dot{\chi} - G(\chi) - D_d\dot{e}_c - K_d e_c] \quad (3.102)$$

where $e_c = \chi_d - \chi_c$ is the position error of the object.

3.4.6 Kinematic Control

The kinematic control here presented computes a torque-based control in a manipulator, such that its end effector tracks a desired position and orientation and the error is driven to zero.

Remark. *For simplicity of notation and provided that the control aims the end effector level, the subscript i will be dropped from this moment on in this subsection.*

Consider the differential kinematics equation

$$\dot{\chi} = J(q)\dot{q} \quad (3.103)$$

In order to establish a relationship between the joint and the task frames in terms of acceleration, its time derivative yields

$$\ddot{\chi} = J(q)\ddot{q} + \dot{J}(q, \dot{q})\dot{q} \quad (3.104)$$

For the reference trajectory $\{\chi_r, \dot{\chi}_r, \ddot{\chi}_r\}$ for a given end effector, it is needed to grant a perfect tracking of the Cartesian reference position and orientation. This objective can be achieved by solving (3.104) for \ddot{q} and by implementing a PD Controller, such that the corresponding desired acceleration of the joints is given by

$$\ddot{q}_d = J^{-1}(q) \left[\ddot{\chi}_r + K_{DV}\dot{e}_\chi + K_{PV}e_\chi - \dot{J}(q)\dot{q}_d \right] \quad (3.105)$$

where $e_\chi = \chi_r - \chi$, $K_{PV} > 0$ and $K_{DV} > 0$. By substituting (3.105) into (3.104), the error dynamics become

$$\ddot{e}_\chi + K_{VD}\dot{e}_\chi + K_{VP}e_\chi = 0 \quad (3.106)$$

which is a second-order exponentially stable and decoupled time-invariant error representation.

However, provided that the manipulator is controlled by torque, the driving torques τ are computed by using a computed torque controller (CACCAVALE *et al.*, 1997). Given the manipulator dynamics presented in equation 3.84, a control input τ composed by two branches is employed in the form

$$\tau = \tau_{FF} + \tau_{PID} \quad (3.107)$$

where τ_{FF} is a feed-forward nonlinear control and τ_{PI} is a feedback linear control. The control branch τ_{FF} is given by

$$\tau_{FF} = \hat{M}(q)\ddot{q}_d + \hat{C}(q, \dot{q}) + \hat{G}(q) \quad (3.108)$$

where \hat{M} , \hat{C} and \hat{G} are estimates for minimizing the effects of C and G and for imposing the desired dynamics.

For the second branch, a standard PID control is proposed to grant perfect tracking, in the form

$$\tau_{PID} = K_{PC}e_q + K_{DC}\dot{e}_q + \int_{t_0}^t K_{IC}e(\sigma)d\sigma \quad (3.109)$$

where error $e_q = q_d - q$ and q are the measured joint variables. The computed torque τ in the manipulator dynamics results in the following closed loop dynamics

$$M(q)\ddot{e} - K_{PC}e_q - \int_{t_0}^t K_{IC}e(\sigma)d\sigma - K_{DC}\dot{e}_q = 0 \quad (3.110)$$

With the assumption that $\hat{M}(q) = M(q)$, it turns out that the dynamics of the error, given by the equation

$$\ddot{e}_q - K_{PC}e_q - K_{DC}\dot{e}_q - \int_{t_0}^t K_{IC}e(\sigma)d\sigma = 0 \quad (3.111)$$

is exponentially stable for $K_{PC} > 0$, $K_{IC} > 0$ and $K_{DC} > 0$. This algorithm is known as second-order closed loop inverse kinematics (CLIK) and provides accuracy, fast tracking of reference trajectories and robustness against disturbances.

3.4.7 Static Load Allocation

As an application of control allocation in manipulators, the load distribution in cooperative tasks is addressed in this text. When two or more agents are involved in a cooperative task, the load can be shared among them efficiently. The motion of every agent in the task has to be synchronized, while observing some aspects, such as the possibility of arising elevated internal forces and the individual maximum payload.

In BAIS *et al.* (2015), the authors propose two different load allocation strategies, static and dynamic, which utilize weighting coefficients to set the load to each agent. The strategies consider the influence of internal forces and different payload capacities as well. According to the load allocated to the agents, a virtual center of mass of the object other than the real one may arise. As a consequence, the desired wrench is applied to the virtual center of mass, which results in undesired torques exerted on the physical center of mass.

Consider N robots responsible for carrying or grasping a rigid object, where the forces exerted by the end effectors $h = \begin{bmatrix} h_1 & \dots & h_N \end{bmatrix}^T$ and the total force on the object h_c are related by

$$h_c = Gh \quad (3.112)$$

The desired wrench h_c is calculated by the inverse dynamics, which yields

$$h_c = h_o - M_o\ddot{\chi}_d - D_o\dot{\chi}_d - G(\chi_d) \quad (3.113)$$

and must be shared to every manipulator involved in the cooperative task. The impedance controller is implemented at the end effector level, such that the force h_i exerted by each of them contributes to the forces applied to \mathcal{F}_c according to equation 3.112.

Consider now a desired object trajectory χ_d . The kinematic constraint will result in a desired trajectory to be performed joint by all end effectors involved, which results in the desired trajectory $\chi_{d,i}$ for the i -th end effector. Thus, the impedance controller is given by

$$\ddot{\chi}_{r,i} = \ddot{\chi}_{d,i} + M_d^{-1}(h_i - D_d\dot{e}_i - K_d e_i) \quad (3.114)$$

such that $e_i = \chi_{r,i} - \chi_{d,i}$.

In order to find the forces h_i to be exerted by the end effectors, the inverse mapping of (3.112) can be written as

$$h = G^\dagger h_c \quad (3.115)$$

which is a typical pseudoinverse problem and can be formulated as an optimization problem with equality constraints, in the form

$$\begin{aligned} \min : \quad & f(h) = h^T W_u h \\ \text{s.t.} \quad & h_c = Gh \end{aligned} \quad (3.116)$$

where W_u is a weighting diagonal matrix for the task-space weights. A common solution for this constrained problem would be a weighted-pseudoinverse for computing h , as presented in (SCHNEIDER and CANNON, 1992), but it is also possible to pre-compute a new grasp matrix to allocate the loads promptly (BAIS *et al.*, 2015), where the authors propose a parametrized grasp matrix $G(W_u)^\dagger$ that considers the shift of the virtual center of mass. The authors define also a vector $W_u \in \mathbb{R}^N$ with normalized values to set priority for each agent, such that

$$\sum_{i=1}^N W_{u,i} = 1 \quad (3.117)$$

As a result, every agent will be in charge of employing a $W_{u,i}$ percentage from the total wrench h_c to execute the task. By means of the Lagrange multipliers, the

optimization problem is solved to obtain the grasp matrix directly, without the need of calling the solver every time.

Let Δ be a vector that points from the object physical center of mass to the virtual one. The displacement vector Δ is given by

$$\Delta = \sum_{i=1}^N W_i r_i \quad (3.118)$$

Thus, the individual force h_i exerted by an end effector is an optimal unique solution, given by

$$h_i = \beta_i \begin{bmatrix} I_3 - \bar{S}K_\Delta\Delta \times & \bar{S}K_\Delta \\ -K_\Delta\Delta \times & K_\Delta \end{bmatrix} h_c \quad (3.119)$$

where $\bar{S} = \Delta \times - r_i \times$, $K_\Delta = (\mathcal{I}_3 + \mathcal{M}_\Delta)^{-1}$, $\mathcal{M}_\Delta = \mathcal{I}^t - (\Delta \times)^T \Delta \times$ and

$$I^t = \sum_{i=1}^N W_{u,i} r_i \times (r_i \times)^T \quad (3.120)$$

The matrix \mathcal{M}_Δ represents the weighted inertia tensor of the system of end effectors with respect to the virtual center of mass Δ and I_c is the inertia tensor with respect to the center of mass of the object. Thus, the undesired internal torques derived from the displacement of the virtual center of mass are given by

$$h_I = \Delta \times f_d \quad (3.121)$$

where f_d is the desired force related to the translational motion. From this equation, one can notice that the undesired torques disappear for $\Delta = 0$. Nonetheless, the internal forces h_I is regarded as a disturbance and limit the possible choices of W_u , which must satisfy:

- if $N = 2$, there exists a unique solution if and only if the grasp sticks are related by $r_1 = cr_2$, where $c \in \mathbb{R}^+$.
- if $N = 3$, there is only a unique solution and the grasp sticks r_i span \mathbb{R} ;
- if $N \geq 3$, there are multiple solutions.

The total wrench h_c on the object and the individual manipulators forces h can be related to the control allocation definitions, which yields

$$h_c \triangleq \nu_d, \quad h \triangleq u \quad (3.122)$$

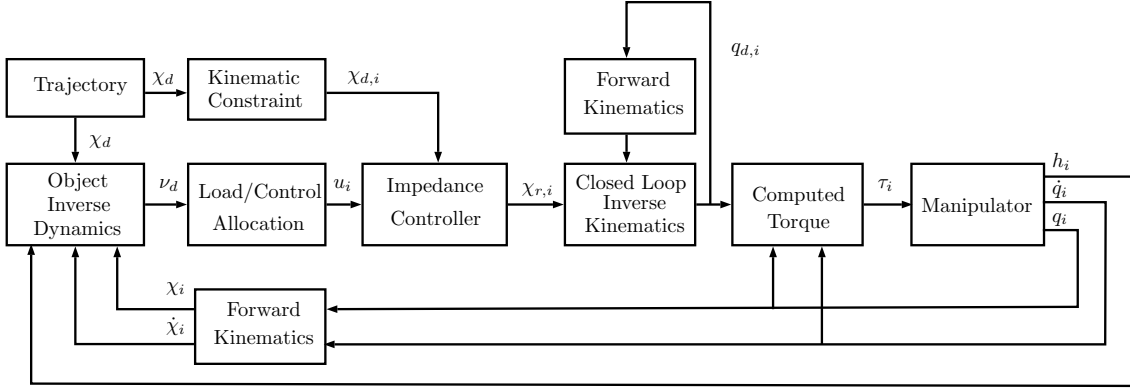


Figure 3.11: Block diagram of the control of cooperative manipulators.

3.4.8 Control Scheme of Cooperative Manipulators

The figure 3.11 illustrates how the control of cooperative manipulators is implemented. In a task where N manipulators are involved, the figure illustrates how the control is conceived for any i -th manipulator.

First, a reference trajectory χ_r for the object is generated, which forms the base for computing the kinematic constraints to be respected by every end effector, as presented in the equation 3.90. The kinematic constraints result in the desired trajectory $\chi_{d,i}$. The acceleration $\ddot{\chi}_d$ of the trajectory generator is also utilized as an estimate of the object acceleration and to compute the inverse dynamics of the object, which is compensated by the external forces due to the contact forces. It yields an estimate of the desired force $\nu_d \triangleq h_c$ to be applied on the object, which is distributed to the manipulators according to a predefined sharing policy.

Then, the impedance controller promotes an impedance-admittance relation between the end effector and the environment in order to generate the force h_i at the cost of provoking a deviation in the reference acceleration $\ddot{\chi}_{r,i}$. The acceleration is integrated, which yields the velocity $\dot{\chi}_{r,i}$ and pose $\chi_{r,i}$. However, this trajectory must be converted into the joint space by means of the Closed Loop Inverse Kinematics (CLIK), which utilizes the values of the previous control loop $\dot{q}_{d,i}$ and $q_{d,i}$ to generate the new desired trajectory $q_{d,i}$.

The Computed-torque controller receives the desired trajectory $q_{d,i}$ and $\dot{q}_{d,i}$, the actual joint angles q_i and the joint velocities \dot{q}_i for the PID Controller branch, which is responsible for ensuring a good tracking and accuracy, and also for compensating gravity and the centrifugal and Coriolis force. Thus, the Computed-torque controller finally generates the torques τ_i to control the i -th manipulator.

Finally, the manipulator tracks the desired trajectory and provides new contact forces, velocity \dot{q}_i and position q_i of the joints to a new execution of the control scheme. Similar approaches can be found in REN *et al.* (2016) and CACCAVALE *et al.* (2008). However, this approach differs from those because the authors have

implemented an impedance control at the object level to find a desired trajectory and to implement an internal impedance control at the manipulators level to impose internal forces to the object. These algorithms also do not allow sharing policies, such as to distribute a load percentage $W_{u,i}$ to each agent.

3.5 Conclusions

The robotic systems addressed in this dissertation have been presented in this chapter, along with their main aspects and the dynamic and kinematic equations. For the wheeled mobile robots, the differential drive robot has been presented, along with its nonholonomic constraint and a Lyapunov-based feedback controller. The ROV LUMA is an overactuated system with respect to x , y and ψ and the PD controller is responsible for the High-level Controller. The quadrotor is an underactuated system and a Integral Backstepping controller has been elected to control it. Finally, the cooperative task consists of manipulators, whose motion is provided by a impedance controller along with a Closed-loop Inverse Kinematics and torque-computed to control the manipulators. Nonetheless, the suggested High-level controllers may be exchanged for any other traditional control law.

Chapter 4

Experiment Results and Simulations

This chapter is concerned with experiment and simulation results, where the control strategy consists of a High-level controller and a control allocation algorithm, as defined in the previous chapter, for each robot detailed. The experiment and simulations were executed in MATLAB® R2017a environment in a notebook with processor Intel Core®i7 2.70 GHz and 8 GB RAM memory.

4.1 Overview of the Algorithms

The control allocation algorithms detailed in this chapter are summarized in the table 4.1 for better visualization, where their related optimization problems are mathematically expressed along with their main characteristics.

4.2 Wheeled Mobile Robot

The control allocation algorithms are validated with the autonomous vacuum robot *Roomba*® 621 designed by *iRobot*, depicted in figure 4.1, with the Lyapunov-based feedback controller. It is a differential drive robot composed by right and left wheels with independent actuators. Under the faceplate indicated in the figure 4.2a, there is a female connector designed for a serial cable with 7 pins, depicted in 4.2b. This connector contains a RXD and a TXD pin for receiving and transmitting data, which use serial communication TTL at 0-5 Volts levels. The *iRobot* has released the iRobot Create USB cable, which converts from serial to USB and allows the computer and *Roomba* to communicate at a 115,200 baud rate.

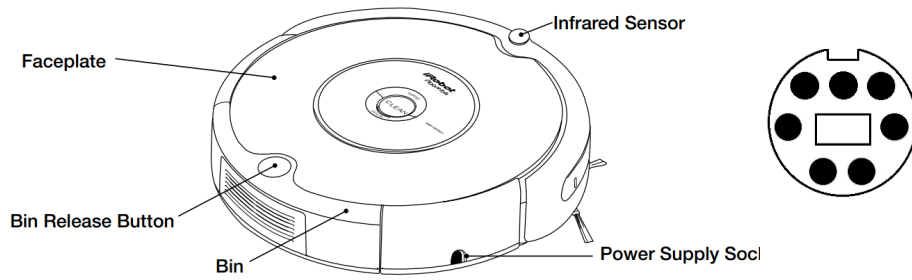
When connected to a computer, the USB cable turns its LEDs on and the computer recognizes it automatically through any communication port COM available.

Algorithm	Optimization Problem	Cost Function	Objective
DCA	max: $f(\alpha) = \alpha$ s.t.: $Bu = \alpha v_d, \alpha v_d \in \mathbb{A}$	Linear	Direction maintenance
Simplex	min: $f(x) = c^T x$ s.t.: $Ax \geq b$	Linear	Error minimization
PDIP	min: $f(u) = \ u - u_p\ _2 + \gamma \ v_d - Bu\ _2$ s.t.: $u_{min} \leq u \leq u_{max}$	Quadratic	Mixed minimization
WLSSA	min: $f(u) = \ W_u(u - u_p)\ _2 + \gamma \ W_v(v_d - Bu)\ _2$ s.t.: $u_{min} \leq u \leq u_{max}$	Quadratic	Mixed minimization
SLA	min: $f(h) = h^T G(\beta)h$ s.t.: $h_o = Gh$	Quadratic	Load allocation

Table 4.1: The control allocation algorithms and their main characteristics.



Figure 4.1: The Roomba 621 manufactured by *iRobot*.



(a) The anatomy of Roomba. ¹ (b) The female connector located under the faceplate of *Roomba*.

The robot has also four operating modes, denoted off, passive, safe and full, where the last three modes are sorted in an increasing order of access. The safe mode grants full control of the robot - it allows to send commands the wheels, to read sensors, but the robot still detects cliffs or when it leaves the ground, wheel drop and charger plugged in. When any of these situations is detected, the robot leaves the safe mode. In order to prevent a misuse, all the experiments are executed in safe mode.

The MATLAB Toolbox for the iRobot Create 2 (ESPOSITO, 2015) eases a lot the communication between the computer and Roomba. This toolbox was developed to run in MATLAB environment and contains a list of user-friendly commands. Originally, for accessing Roomba it was necessary to write one-byte opcodes. The toolbox translates these opcodes to intuitive commands, such that the user does not need to worry about sequences of codes to send and read data. The robot contains commands to read the distance traveled and the angle, which are based on the encoder readings of the right and left wheel. The readings are then converted based on the odometry equations in 3.11. The *Roomba* robot contains an encoder at each wheel, with resolution equal 508.8 counts per revolution and each wheel can travel at a maximum velocity of 0.5 m/s forward and reversely, although the odometry presents huge errors at such rates. Therefore, for the experiment, the maximum and minimum velocities were set at ± 0.2 m/s.

The constructive parameters utilized for modeling the kinematics of *Roomba* and to convert the encoder reading to distance and angle traveled are presented in the table 4.2.

In the control allocation context, the desired virtual control input ν_d and the control input vector u are respectively given by

¹Figure extracted from www.irobotweb.com/-/media/Files/Support/Home/Roomba/600/Roomba-600-Manual.pdf

Parameter	Description	Unit	Value
r_d	half the distance between wheels	m	0.1175
r_w	wheel radius	m	0.036

Table 4.2: Parameters of the *Roomba* robot.

$$\nu_d = \begin{bmatrix} v_d \\ \omega_d \end{bmatrix}, \quad u = \begin{bmatrix} v_r \\ v_l \end{bmatrix} \quad (4.1)$$

The virtual control input vector $\nu \in \mathbb{R}^2$ and the control input $u \in \mathbb{R}^2$ are related by the control effectiveness matrix, which for the robot *Roomba* is

$$B = \begin{bmatrix} 0.5 & 0.5 \\ 4.2553 & -4.2553 \end{bmatrix} \quad (4.2)$$

with $\text{rank}(B) = 2$.

Then, the *Roomba* robot is required to navigate through waypoints, which form a trajectory similar to a lane-change maneuver. Every waypoint is circumscribed by a circumference of radius $\epsilon = 0.1$ m utilized as a trigger to set the next desired waypoint as the vehicle touches its limits, namely $r \leq \epsilon$. In order to impose different control requirements and hence to apply the control allocation strategies at such circumstances, it is necessary to impose different trajectories. Provided that the Lyapunov-based feedback controller formula proposed by AICARDI *et al.* (1995) does not contain desired velocities or accelerations, we can impose it by varying the distance error r , since it is the polar coordinate that mostly interferes in the desired virtual control vector ν_d . The further the desired waypoint is located, the higher is the distance error r and also the linear velocity v , or depending on the sequence of waypoints, for instance in a tricky trajectory with obstacles to be avoided, the high-level controller may also require demanding trajectories in terms of the angular velocity ω .

Thus, a solution for imposing higher control requirements is to increase the distance between the desired coordinates $x_{d,i}$ and $x_{d,i+1}$, which results in the following trajectories

- desired trajectory 1: $x_d = [-3 \quad -2.5 \quad -2 \quad -1.5 \quad \dots \quad 3 \quad 3.5 \quad 4 \quad 4.5 \quad 5]$
- desired trajectory 2: $x_d = [-2.75 \quad -2 \quad -1.25 \quad \dots \quad 3.25 \quad 4 \quad 4.75 \quad 5.5]$
- desired trajectory 3: $x_d = [-2.5 \quad -1.5 \quad -0.5 \quad 0.5 \quad 1.5 \quad 2.5 \quad 3.5 \quad 4.5 \quad 5.5]$

To complete the desired waypoint coordinate, the corresponding $y_{d,i}$ is calculated according to the function

Parameter	Unit	Value
k_r	-	1
k_α	-	1
k_θ	-	0.5
ϵ	m	0.1
W_u	-	\mathcal{I}_2
W_v	-	\mathcal{I}_2
γ	-	10^6
$u_{i,min}$	m/s	-0.2
$u_{i,max}$	m/s	0.2

Table 4.3: Parameters of the control allocation techniques.

$$y_{d,i} = \begin{cases} -1, & \text{if } x_{d,i} < -1.25 \\ \sin(x_{d,i}), & \text{if } -1.25 \leq x_{d,i} < 1.5 \\ 1, & \text{if } x_{d,i} \geq 1.5 \end{cases} \quad (4.3)$$

Finally, the experiment starts with the vehicle at the posture $\chi(0) = [-4 \quad -1 \quad \pi/2]^T$ in the Cartesian coordinates. The experiment lasts 50 seconds, regardless of whether the trajectory or the control allocation algorithm utilized.

The figure 4.3 depicts the three trajectories assigned to *Roomba* when no control allocation algorithm is applied and saturation remains unmanaged. The plots (a), (b) and (c) represent the trajectories 1, 2 and 3, and (b), (d) and (f) their virtual controls, respectively. When the vehicle enters the circumference defined by $r \leq \epsilon$ and the next desired waypoint is set, the desired virtual control input vector ν_d in the following seconds corresponds to wheel velocities at the saturation values. It results in a virtual control input ν that does not respect any optimization criteria and causes a navigation towards undesired directions. As the control requirements increase, performance becomes more deteriorated, as shown in the trajectories 1, 2 and 3. The trajectories traveled are not the shortest path, although they still can reach some desired waypoints. In the trajectory 3 between $t = 25.6$ and $t = 28$ s, the vehicle is even required to travel backwards in order to correct its orientation and navigate towards the desired waypoint $x_{d,4}$. In order to circumvent the input constraints imposed by the maximum and minimum wheel velocities and to grant smooth and efficient travelling, the results for the control allocation techniques employed are shown and discussed next.

Direct Control Allocation

The figure 4.4 illustrates the experimental results obtained with the Direct Control Allocation. This algorithm provides motion towards the direction desired by the

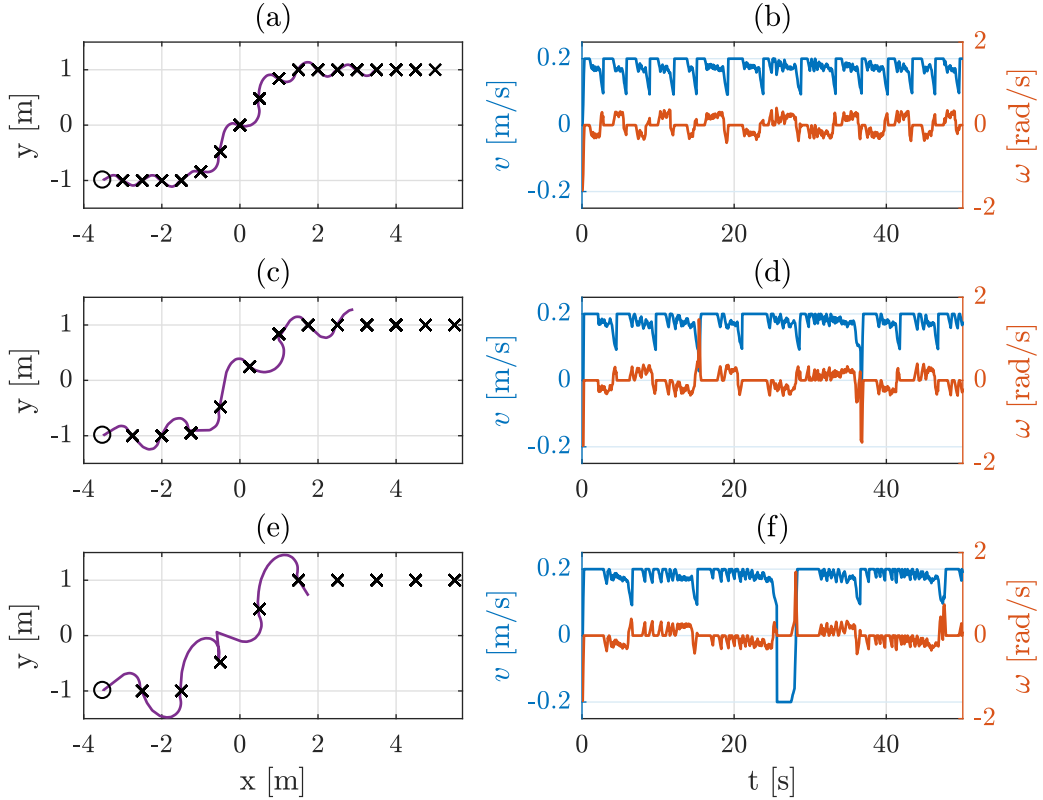


Figure 4.3: Results when saturation is unmanaged. (\circ initial position; \times desired waypoints; — trajectory traveled; — linear velocity v ; — angular velocity ω ; (a) trajectory 1; (b) linear and angular velocities for trajectory 1; (c) trajectory 2; (d) linear and angular velocities for trajectory 2; (e) trajectory 3; (f) linear and angular velocities for trajectory 3.

High-level controller, but with lower performance. When experiment starts for the three desired trajectories, the robot rotates clockwise, travels forth and back, and forth again while rotates counterclockwise until it reaches a steering a orientation error α low enough to travel linearly towards the first desired waypoint $x_{d,1}$. From the plots, one can notice also that the High-level is mainly concerned with prioritizing the linear velocity, whereas the angular velocity ω presents minor values along the rest of the experiment, which allows the robot to travel smoothly and to converge the degrees of freedom simultaneously. We can observe a more constant angular velocity as the distance between $x_{d,i}$ and $x_{d,i+1}$ increase, provided that the robot tends to travel in a straight line after correcting its orientation when a new $x_{d,i}$ is set.

Linear Programming with Simplex

A simultaneous convergence of the degrees of freedom is not observed with Simplex, as shown in figure 4.5. When experiment starts, the Simplex changes the desired

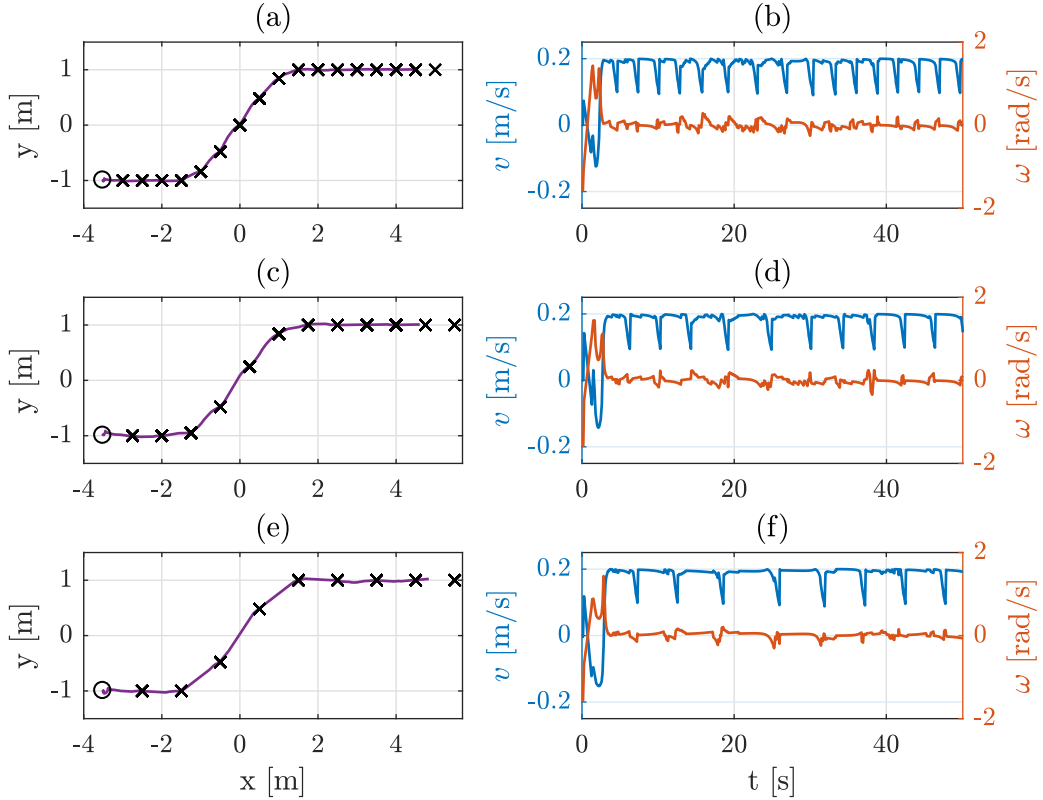


Figure 4.4: Results with the Direct Control Allocation.

dynamics to prioritize the orientation of the robot. Then, it rotates clockwise and starts decreasing the angular velocity and increases the linear velocity, while travels towards the first waypoint. Everytime the robot reaches the circumference $r \leq \epsilon$, it first corrects the orientation, and then, travels to reduce the distance error r . However, the oscillatory response observed in ω in (b), (d) and (f) shows that the Simplex is always correcting the orientation to a lesser extent.

Despite the choice of the parameters k_r , k_α and k_θ to provide a damped response and simultaneous convergence of the degrees of freedom, as noticed with the DCA, we can notice that the dynamic behavior of the angular velocity ω corresponds to higher values of k_α and k_θ . Therefore we can conclude that the Simplex changes the design parameters of the controller. The transient response observed is undesired in most situations, although the vehicle still visits every desired waypoint, with exception of the last one due to time restriction.

Weighted Least Squares with Active Set

Despite the WLSAS changes the control direction to minimize the error, the error minimization term in the *Roomba* robot results in a unique solution, and therefore, a secondary control objective cannot be satisfied. The figure 4.6 presents the ex-

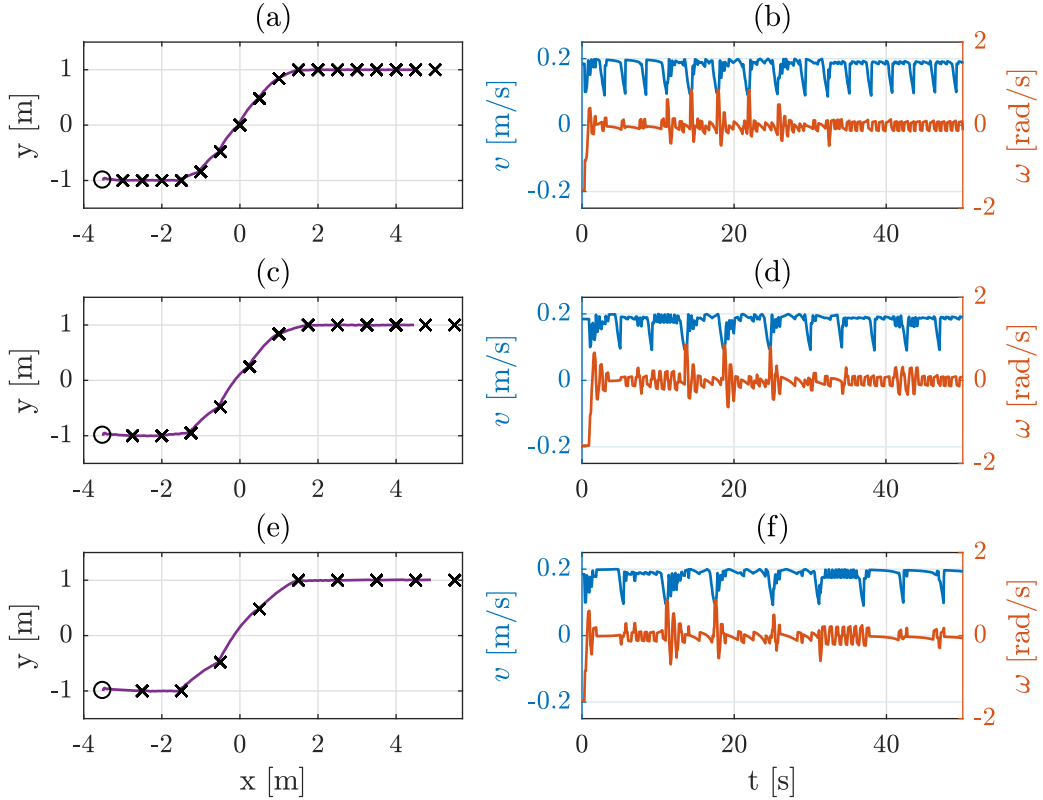


Figure 4.5: Results with the Simplex.

periment results for the WLSAS. Provided that the quadratic cost function allows a simultaneous convergence of every DoF, it does not present transient responses as intense as those observed in Simplex, although they cannot be visually observed, except for the outlier peaks with the angular velocity $\omega > 0.1$ rad/s, as observed in plots (b), (d) and (f). As expected, the trajectory traveled becomes more stable and smooth as the distance between the desired waypoints increase.

Primal-dual Interior Point

The last algorithm here considered is the PDIP, whose results are displayed in figure 4.7 and are similar to those observed with the WLSAS, although it presents a smoother response, specially the angular velocity ω in the plot (f) for the trajectory 3. Just like the WLSAS, the algorithm only minimizes the error and provides a unique solution, since the vehicle works with at least one wheel at the saturation levels and the system is underactuated.

Considerations

In the figures 4.8 and 4.9 are depicted the control input signals for each wheel, obtained from the control allocation algorithms, such that the figure 4.8 contains

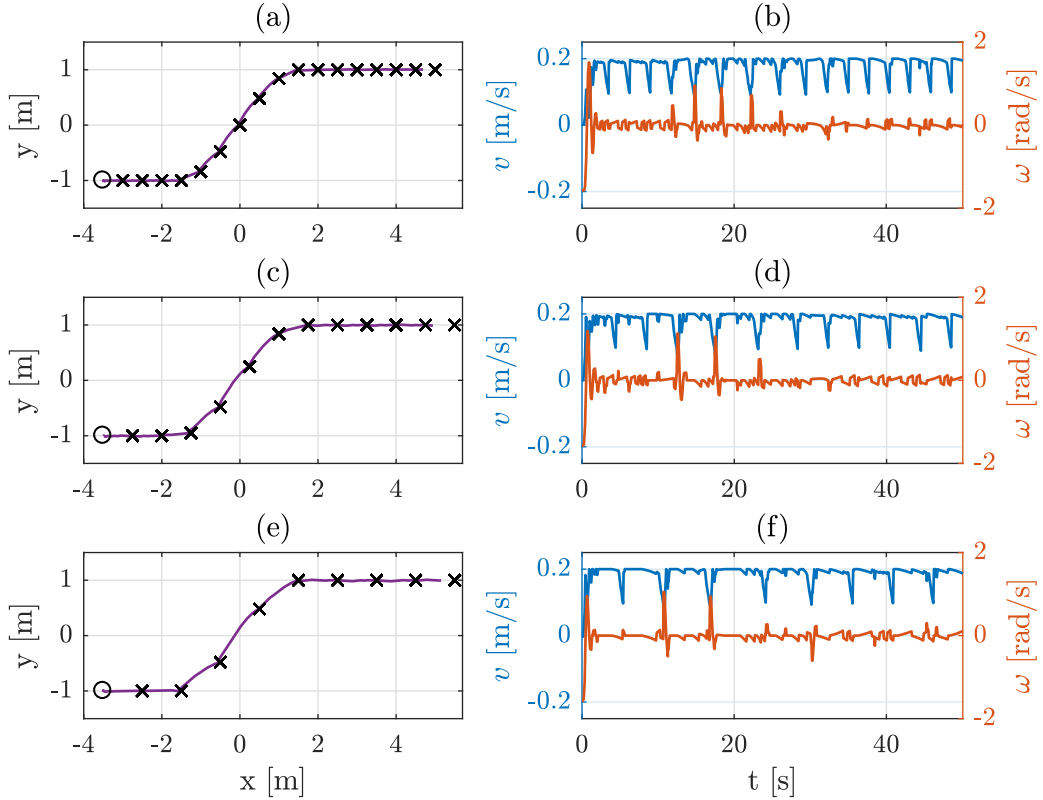


Figure 4.6: Results with the Weighted Least Squares with Active Set.

the optimal solutions obtained from DCA and Simplex, and in 4.9 are presented the control inputs for the WLSAS and PDIP algorithms. In all the plots, the saturation levels are respected and the control signals are constrained to these limits.

The control inputs for Simplex present an oscillatory behavior, as detailed previously, when compared with the DCA, which provokes a nonuniform trajectory of *Roomba*. On the other hand, the algorithms, WLSAS and PDIP, also provided good results and a smooth trajectory. Although the behavior presented by Simplex still converges to the desired waypoints, it can be undesired in more sensible tasks, such as for transporting dangerous materials or people.

For providing a more precise analysis of the results obtained with the algorithms, the virtual control input error is displayed in the tables 4.4, 4.5 and 4.6. The tables are composed by three columns, to register the maximum and average norm values, calculated according to equation 2.56. The values were rounded to the nearest second decimal.

In the Virtual Control Error (VCE), the DCA presents the highest errors among all the control algorithms, not only the maximum errors but also the average ones. Such values are expected since optimization problem is not concerned with the error minimization, but focuses on only with the direction maintenance, although it still able to reach the desired waypoints, as observed in the experiments. The maxi-

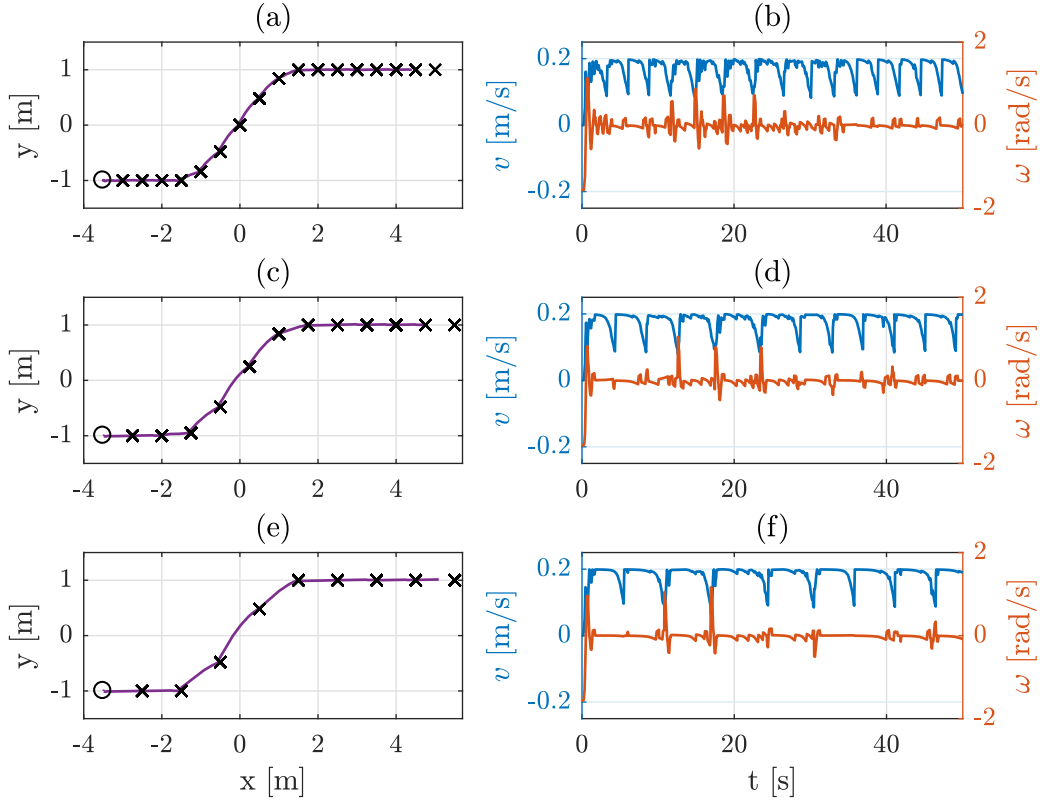


Figure 4.7: Results with the Primal-dual Interior Point.

imum and average errors in Simplex, WLSAS and PDIP do not present meaningful differences.

On the other hand, when analyzing the Direction Error (DE), we see that the DCA presents error 0 for the maximum and average values found. The WLSAS and PDIP present similar nonzero values, as expected, provided that both utilize QP formulations. However, the Simplex algorithm presents the highest DE for all the three trajectories and these discrepancy cannot be neglected. In fact, the Simplex is a LP optimization problem, which finds solutions located only at the vertices and hence the results confirm the theory.

Table 4.4: Virtual control errors (VCE) and Distance Error (DE) for the trajectory 1 in *Roomba*.

Algorithm	VCE		DE	
	Maximum	Average	Maximum	Average
DCA	3.42	0.22	0	0
Simplex	0.71	0.19	0.70	0.17
WLSAS	0.64	0.18	0.63	0.08
PDIP	0.65	0.19	0.63	0.10

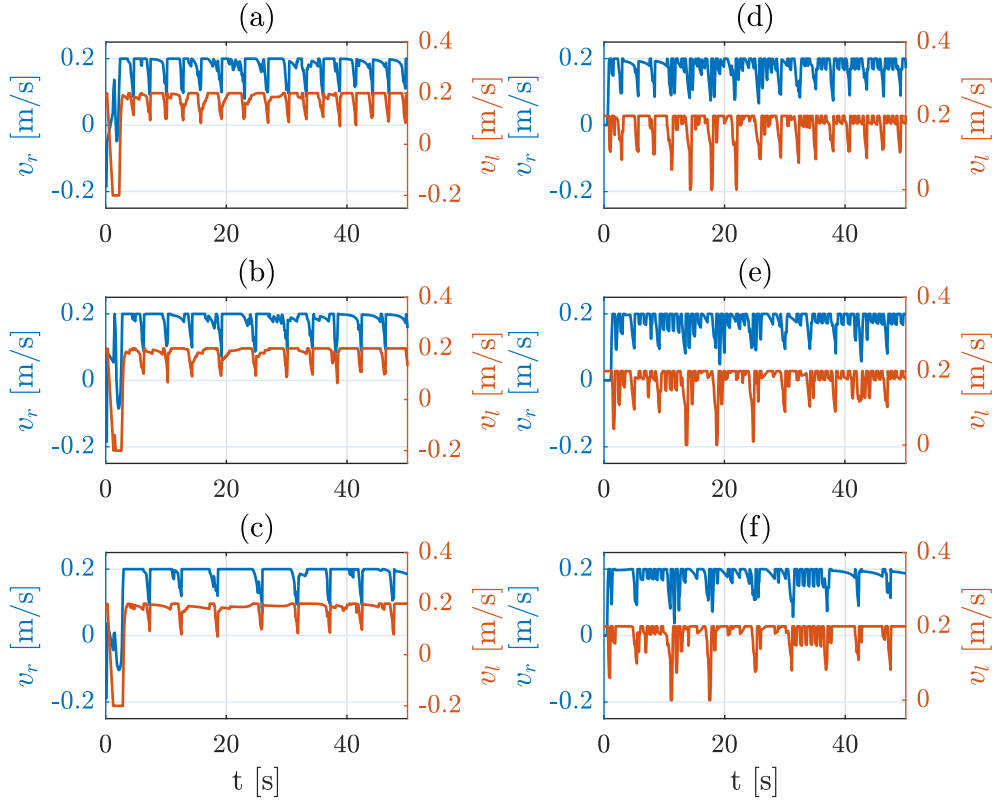


Figure 4.8: Wheel velocities in the DCA (plots a, b and c) and Simplex (d, e and f). (Legend: — right wheel velocity v_r ; — left wheel velocity v_l).

Table 4.5: VCE and DE for the trajectory 2 in *Roomba*.

Algorithm	VCE		DE	
	Maximum	Average	Maximum	Average
DCA	3.61	0.36	0	0
Simplex	0.97	0.33	0.81	0.27
WLSAS	0.97	0.32	0.74	0.12
PDIP	0.98	0.31	0.76	0.10

Table 4.6: VCE and DE for the trajectory 3 in *Roomba*.

Algorithm	VCE		DE	
	Maximum	Average	Maximum	Average
DCA	3.76	0.50	0	0
Simplex	1.31	0.47	0.92	0.24
WLSAS	1.29	0.45	0.82	0.11
PDIP	1.30	0.45	0.84	0.08

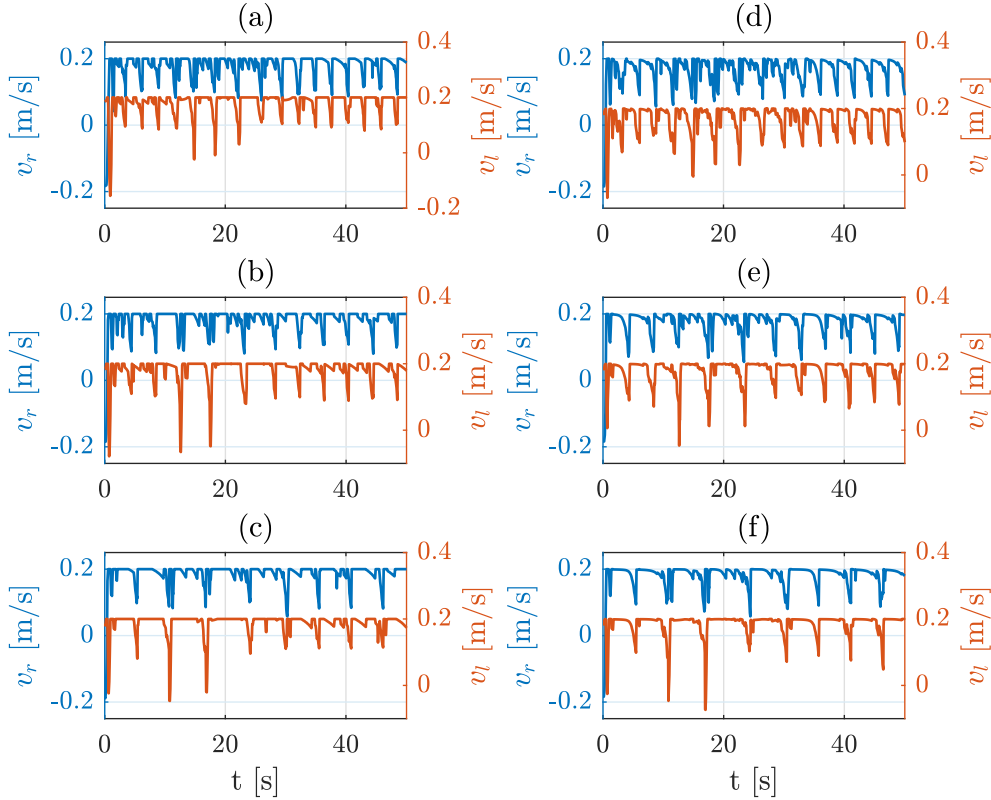


Figure 4.9: Wheel velocities in the WLSAS (plots a, b and c) and PDIP (d, e and f). (Legend: — right wheel velocity v_r ; — left wheel velocity v_l).

4.3 Remotely Underwater Operated Vehicle

The simulation for the ROV LUMA is implemented on MATLAB [®]. Due to the propellers layout, the ROV cannot be directly controlled with respect to the roll and pitch angles. In fact, the ROV present four propellers placed horizontally, which control the robot with respect to x , y and ψ , and a fifth propeller to control its depth (z). In face of these aspects, it is assumed that

1. The vehicle operates at the horizontal position, with $\phi \approx 0$ and $\theta \approx 0$;
2. The propellers do not contribute to motion with respect to ϕ and θ ;

As the depth is controlled only by a single propeller, which does not contribute to motion with respect to any other DoF because of the first assumption, its virtual control can not be allocated to the other four propellers. Therefore, this propeller and the dynamics of the depth z are not regarded for control allocation purposes. It results in a control effectiveness matrix with more columns than rows and in an overactuated system with respect to x , y and ψ . It allows the control allocation algorithms WLAS and the PDIP algorithms to benefit from redundancy and to

Parameter	Unit	Value
R_{p1}	m	(0.2378, -0.2917, 0)
R_{p2}	m	(0.2378, 0.22917, 0)
R_{p3}	m	(-0.3667, -0.2128, 0)
R_{p4}	m	(-0.3667, 0.2128, 0)
P_1	-	$[\cos(\pi/4), \sin(\pi/4), 0]$
P_2	-	$[\cos(3\pi/4), \sin(3\pi/4), 0]$
P_3	-	$[\cos(\pi/4), \sin(\pi/4), 0]$
P_4	-	$[\cos(3\pi/4), \sin(3\pi/4), 0]$

Table 4.7: Position (R_{pi}) and orientation (P_i) of the propellers.

satisfy secondary control objectives when the control requirements correspond to $u \in \mathbb{U}$.

The table 4.7 details the position and orientation of the propellers extracted from DE ANDRADE (2017) to calculate the control effectiveness matrix B , according to the equation (3.39), which results in

$$B = \begin{bmatrix} 0.7071 & -0.7071 & 0.7071 & -0.7071 \\ 0.7071 & 0.7071 & 0.7071 & 0.7071 \\ 0.3477 & 0.3744 & 0.4233 & 0.4233 \end{bmatrix} \quad (4.4)$$

with $\text{rank}(B) = 3$.

Then, the configuration of LUMA for the simulation is described by

$$\chi = \begin{bmatrix} x \\ y \\ \psi \end{bmatrix} \quad (4.5)$$

In terms of control allocation, the desired virtual control input vector ν_d and the control input vector u are respectively given by

$$\nu_d = \begin{bmatrix} f_{x,d} \\ f_{y,d} \\ \tau_{\psi,d} \end{bmatrix}, \quad u = \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \end{bmatrix} \quad (4.6)$$

The table 4.8 displays the physical parameters, such as the center of gravity (p_G), center of buoyancy (p_B) and the LUMA volume (ΔV). Since only the motion with respect to x , y and ψ regarded, the simplified inertia matrix becomes

$$M_{CR} + M_A = \begin{bmatrix} 46 & 0 & 0 \\ 0 & 61 & 0 \\ 0 & 0 & 30.5 \end{bmatrix} \quad (4.7)$$

Parameter	Unit	Value
p_G	m	$[0 \ 0 \ 0.2]^T$
p_B	m	$[0 \ 0 \ -0.3]^T$
∇V	m^3	0.0396
v_{wi}	m/s	$[0 \ 0 \ 0]^T$
ρ	kg/m^3	1025
$C_T^*(0^\circ)$	-	0.0293
$C_T^*(180^\circ)$	-	0.0179
α^+	-	0.0395
α^-	-	0.0233

Table 4.8: Physical parameters of the ROV, for describing the undersea environment and the dynamics of the propellers.

Parameter	Unit	Value
K_P	-	diag(30, 30, 30)
K_D	-	diag(60, 60, 95)
γ	-	10^3
W_v	-	diag(1, 1, 1)
W_u	-	\mathcal{I}_4
u_p	N	$0_{4 \times 1}$
$u_{i,max}$	N	16
$u_{i,min}$	N	-10

Table 4.9: Gains of the PD Controller and the parameters of the control allocation algorithms.

Moreover, other important parameters are presented in the table 4.8 and consist of the seawater density ρ and the velocity that the water goes through the propellers v_{wi} to simulate the undersea environment. Finally, the dynamics of the propellers are characterized by the forward and reverse thrust coefficients ($C_T^*(0^\circ)$ and $C_T^*(180^\circ)$), the forward and reverse blade thrust coefficients (α^+ and α^-).

Finally, the proportional K_P and derivative K_D gains of the PD Controller and the parameters of the control allocation algorithms utilized in the simulation are presented in the table 4.9.

The gains of the High-level controller presented in the table 4.9 were chosen after successive attempts with the intent to grant tracking, to provide a damped response in all degrees of freedom under analysis and to impose a trajectory demanding enough to bring the actuators to saturation. The simulation runs with parameters $k_1 = 6$ and $k_2 = 3$ for the reference model for every DoF, except for the yaw angle ψ , which runs with $k_1 = 1$ and $k_2 = 0.5$ for a slower dynamics.

The simulation runs during $t_s = 60$ s with sampling time equal 0.05 s. The ROV LUMA is initially placed at $\chi = [0 \ 0 \ 0]^T$ and is required to follow a circular

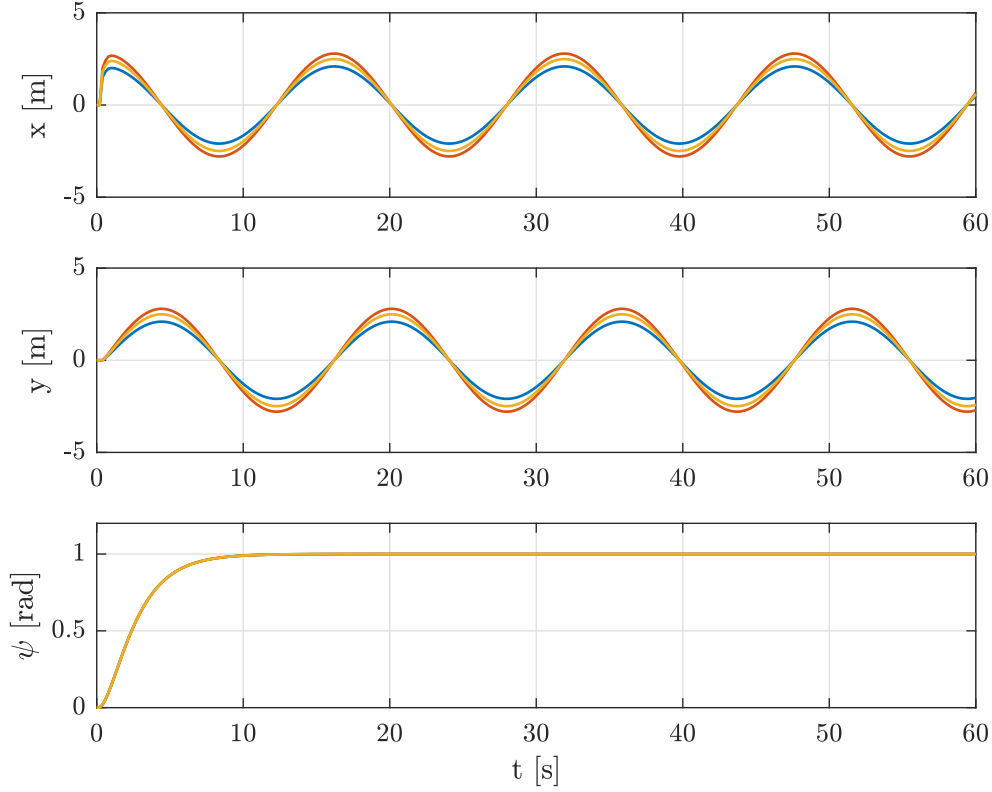


Figure 4.10: The desired trajectories for the simulation 1, 2 and 3 originated from the reference trajectory. (— desired trajectory χ_{d1} , — desired trajectory χ_{d2} , — desired trajectory χ_{d3}).

trajectory generated by the reference trajectory

$$\chi_r = \begin{cases} x_r(t) = A_n \cos(0.4t) \\ y_r(t) = A_n \sin(0.4t) \\ \psi_r(t) = 1 \end{cases} \quad (4.8)$$

such that the amplitude A_n attains the following values

- $A_n = 2.1$ to generate the desired trajectory χ_{d1} ;
- $A_n = 2.5$ to generate χ_{d2} ;
- $A_n = 2.8$ to generate χ_{d3} .

With these reference trajectories, the reference model generates the desired trajectories depicted in the figure 4.10, and also the first and second derivatives required by the High-level controller to compute the desired virtual control input vector ν_d . The desired dynamics of the yaw angle ψ are the same for every trajectory.

At this time, the simulation is executed for the three reference trajectories without any control allocation algorithm. Therefore the vehicle is totally subject to the

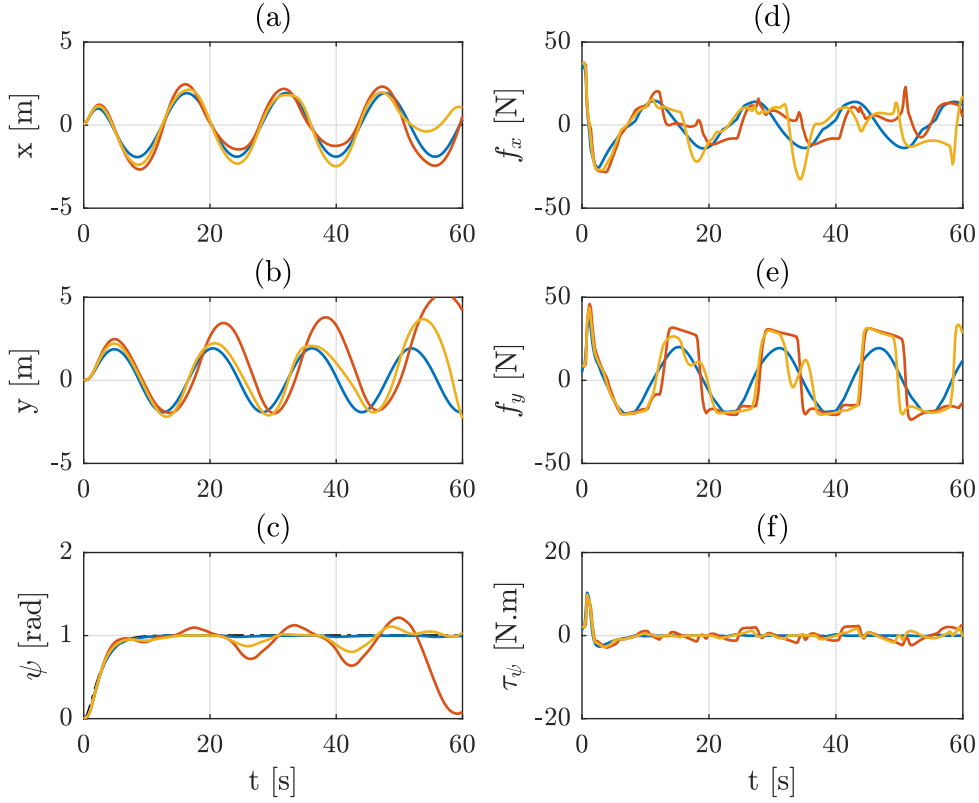


Figure 4.11: The dynamics of the ROV when subject to saturation. (— results for χ_{d1} , — results for χ_{d1} , — results for χ_{d3}).

saturation effects, as depicted in the figure 4.11. Although LUMA can follow the desired trajectory for a while in the trajectories 2 (b) and 3 (c), the tracking error increases as time elapses and leads the vehicle to instability.

The desired trajectory 1 is less demanding and hence LUMA can track it without problems, despite being affected by saturation in the initial seconds of the simulation. Then, the controls become less demanding and the robot can perform tracking without taking its actuators into saturation region.

Direct Control Allocation

The first algorithm analyzed here is the Direct Control Allocation and its results are depicted in the figure 4.12. The PD Controller allied with the DCA are able to keep the system stable for the simulations 1 and 2. Along all the simulation time, the ROV tracks the desired trajectory without presenting overshoot with respect to x , y and ψ . By observing the virtual controls displayed in (d), (e) and (f), the more the actuators are required to operate near saturation levels, the more the virtual controls degrade and present distortion, when compared with the results obtained for χ_{d1} .

The DCA scales the desired virtual control vector ν_d in order to find a feasible

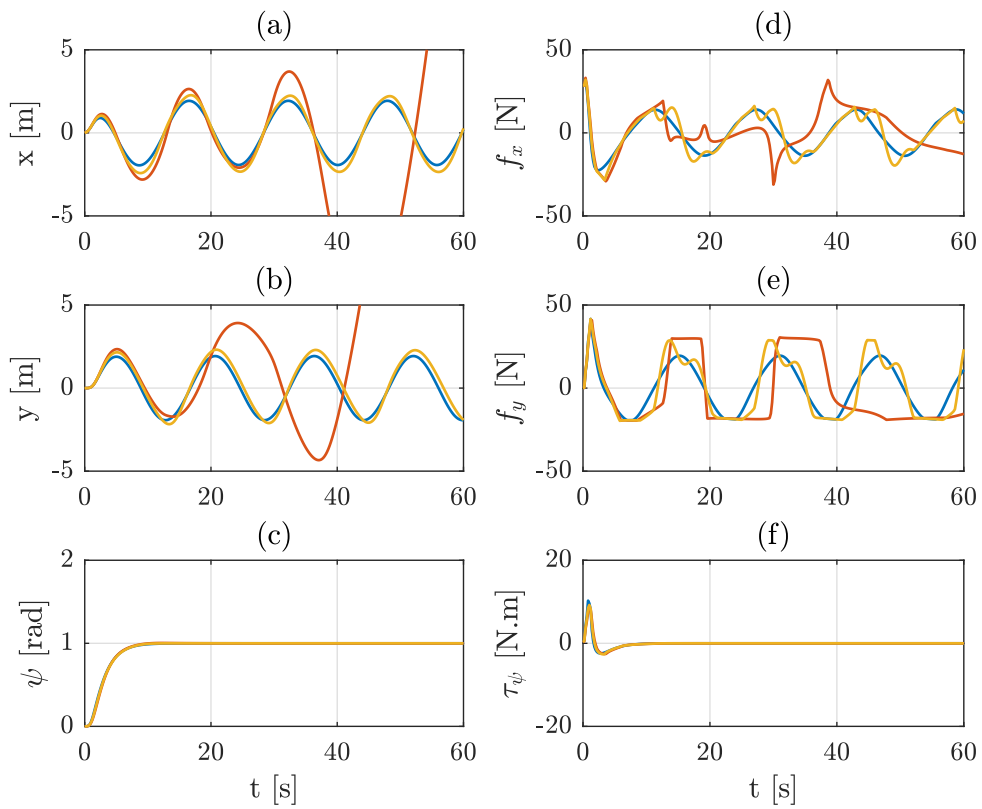


Figure 4.12: The dynamics of LUMA with the Direct Control Allocation algorithm. (— results for χ_{d1} , — results for χ_{d2} , — results for χ_{d3}).

control input u that produces a virtual control ν with the same direction. As a result, ν provides a convergence of the degrees of freedom similar to those observed with ideal actuators, but with lower performance. The results for χ_{d1} and χ_{d2} confirm it, provided that the three DoF considered, x , y and ψ , converge smoothly and without overshooting. On the other hand, we can observe some degradation in the virtual controls for χ_{d2} , although the trajectory performed is not deteriorated at all. However, there is a limit where the DCA can properly manage saturation to fulfill the control requirements, as demonstrated with the results for χ_{d3} . Although the ROV can track the desired trajectory initially, the rotors of 3 and 4 are forced to operate saturated at the minimum and maximum positions, as shown in figure 4.20. As a result, the DCA is not capable of tracking the desired trajectory with respect to x and y , although the yaw angle ψ is properly tracked.

Simplex

Unlike the DCA method, the Simplex algorithm concerns with the virtual control error minimization and does not satisfy any secondary control objective. Provided that its solutions are located on the vertices of the feasible region, it tends to prioritize some control inputs to the detriment of the remaining ones. As a result, some degrees of freedom may converge faster or even be prioritized. The figure 4.13 illustrates this situation, where the controls tend to prioritize the convergence of x and y , specially in the results for χ_{d3} , where saturation is more present - in fact, during the first seconds of the simulations, f_x and f_y reaches the highest values and acceptable tracking is performed with respect to x and y to the detriment of ψ , which registers intense deviation periodically to grant the tracking of x and y .

This prioritization can be noticed by analyzing the virtual controls in the plots (d), (e) and (f). The signals f_x and f_y become nearly sinusoidal after about 20 seconds of simulation. The same cannot be verified in τ_ψ , which periodically deviates from the desired value in order to allow a good tracking of x and y .

Although the demanding trajectories degrade the system and the virtual controls, all the simulations proposed lead to acceptable results and are able to perform tracking, unlike the DCA.

Primal-dual Interior Point

The dynamics due to the Primal-dual Interior Point algorithm are displayed in the figure 4.14, that shows that the input constraint imposed by saturation is successfully managed for the three simulations considered.

Simulation 1 demands less efforts and hence, the virtual controls ν are mostly located inside the feasible set of virtual controls \mathbb{A} . Despite it deviates from the

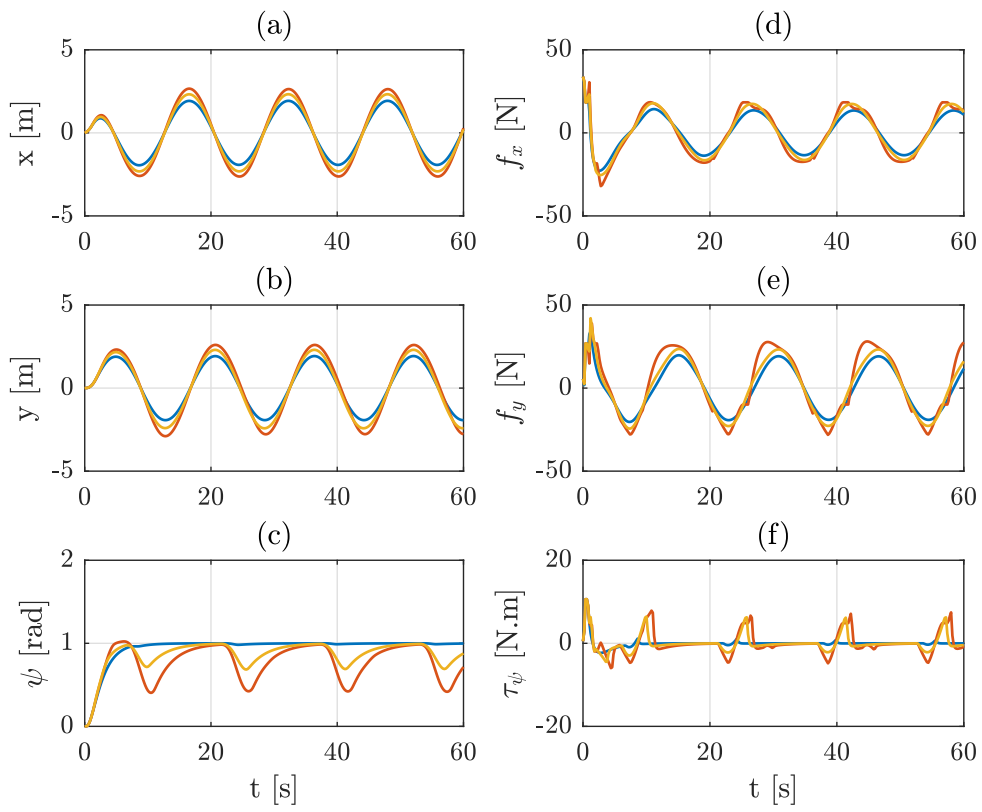


Figure 4.13: The resulting dynamics of LUMA obtained with the Simplex algorithm. (— results for χ_{d1} , — results for χ_{d2} , — results for χ_{d3}).

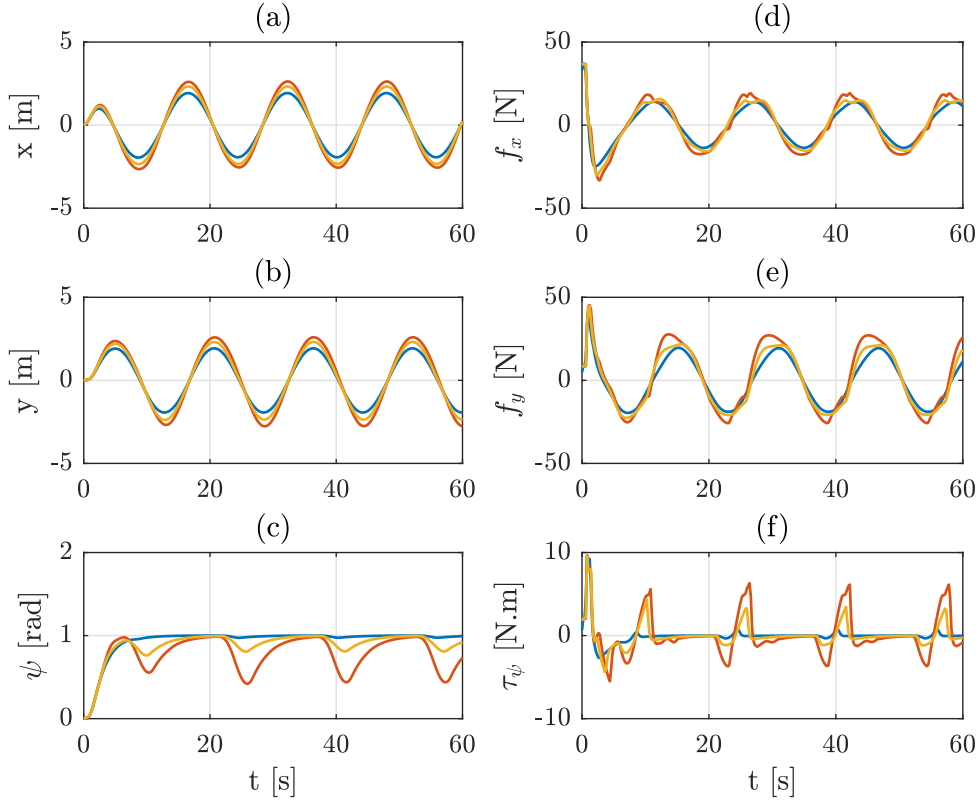


Figure 4.14: Results for the Primal-dual Interior Point algorithm with $\gamma = 1000$ (— results for χ_{d1} , — results for χ_{d2} , — results for χ_{d3}).

expected sinusoidal dynamics concerning f_x and f_y in the first time instants, ν soon stabilizes in this sense. However, by observing the virtual control τ_ψ , there is a small periodical deviation from the desired value, occasioned by the optimization program adopted, which in search of the minor error, modifies the virtual control directions and degrades each DoF to some extent.

As the trajectory becomes faster and more demanding, the deviation in the direction of the virtual controls become more remarkable, although they can still converge. The results for χ_{d2} show how the dynamics are impacted when compared to the results obtained with χ_{d1} . This impact becomes even more intense in the results originated from the desired trajectory χ_{d3} , especially the yaw angle, which periodically deviates from the reference value $\psi_r = 1$ rad.

The weight of the error minimization over the control minimization term is worth mentioning, since it has been set a $\gamma = 10^3$. According to the prioritization factor γ , the control minimization exerts an important influence on the results. For better visualization of the control minimization, the figure 4.15 depicts the same trajectory, but with a weight $\gamma = 10$. There is no noticeable difference concerning the dynamics of x and y , but the yaw angle ψ presents a more degraded convergence.

As we have set the preferred input vector $u_p = 0_4$, the optimal solution of the QP

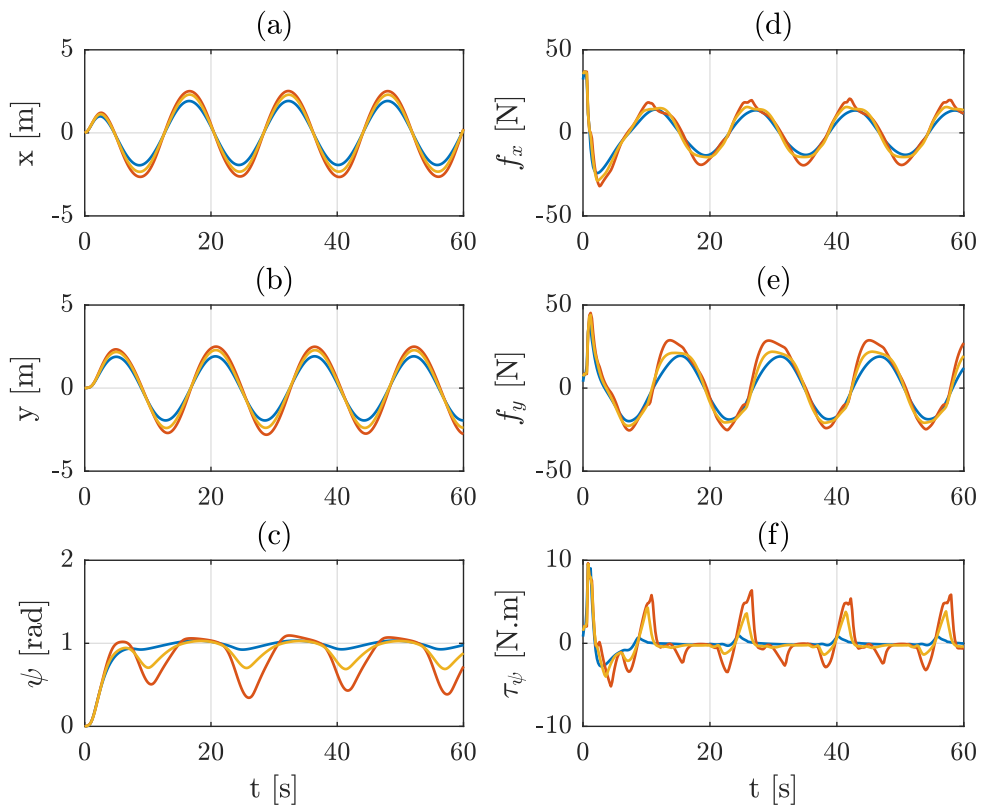


Figure 4.15: Results the Primal-dual Interior Point algorithm with $\gamma = 10$ (— results for χ_{d1} , — results for χ_{d2} , — results for χ_{d3}).

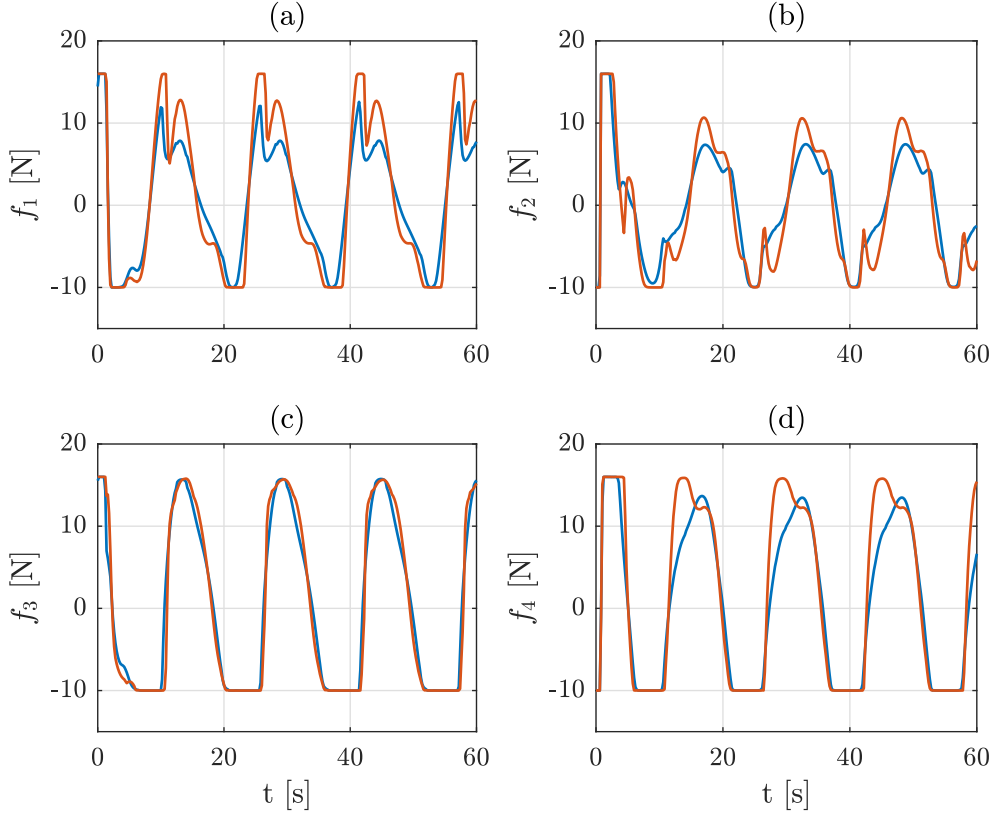


Figure 4.16: Forces produced by each propeller with $\gamma = 10$ and $\gamma = 10^3$ in the PDIP algorithm for the desired trajectory χ_{d2} (— $\gamma = 10$, — $\gamma = 10^3$).

problem will result in decayed control inputs and hence less energy consumption, as depicted in figure 4.16 for the desired trajectory χ_{d2} , where the forces delivered by the actuator reduce when a lower prioritization factor $\gamma = 10$ is applied to the PDIP. However, as the trajectory often demands the propellers to work at saturation levels, the solution for the error minimization problem is also unique and a higher priority for the control minimization will result in a poorer performance.

Weighted Least Squares with Active Set

The Weighted Least Squares with Active Set algorithm is executed with the following virtual control weighting matrix

$$W_v = \begin{bmatrix} 10 & 0 & 0 \\ 0 & 10 & 0 \\ 0 & 0 & 25 \end{bmatrix} \quad (4.9)$$

as an attempt to improve the yaw dynamics, provided that the Primal-dual Interior Point does not allow to set weights among the degrees of freedom. In fact, better dynamics can be achieved if there are suitable weights, otherwise the system may

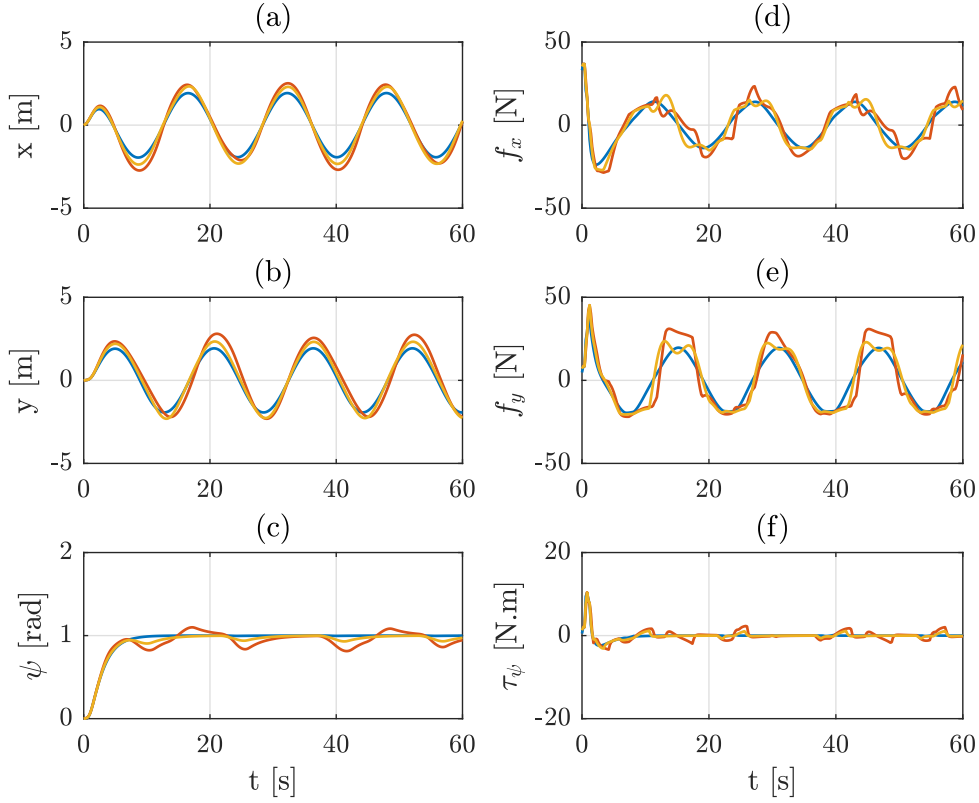


Figure 4.17: The dynamics of LUMA with the Weighted Least Squares with Active Set algorithm. (— results for χ_{d1} , — results for χ_{d2} , — results for χ_{d3}).

become unstable. After some attempts to find a suitable weighting matrix W_v , the results obtained with the W_v above are displayed in the figure 4.17, which present better dynamics for the yaw angle ψ , when compared with the PDIP.

By analyzing the simulations 1, 2 and 3, one can notice that with a higher priority set for the yaw angle and it was still possible to get an acceptable tracking for x and y . Nonetheless, the virtual controls f_x and f_y became degraded in order to pursue a better tracking of ψ , specially in the simulations with desired trajectories χ_{d2} and χ_{d3} . Even with the priority adjustment, the yaw angle still deviates from the desired value at the cost of the better tracking of x and y .

The Control Inputs

Below in the figure 4.18, 4.19 and 4.20 are presented the control inputs registered for the desired trajectories χ_{d1} , χ_{d2} and χ_{d3} and will help to get more reliable information concerning the control requirements and the control allocation algorithms.

As χ_{d1} is less demanding, the rotors work saturated in the first seconds of simulation and also when they rotate reversely at -10 N. However, the rotors do not saturate when rotating forward.

However, in the control inputs for χ_{d2} , the propellers that provide the forces f_3

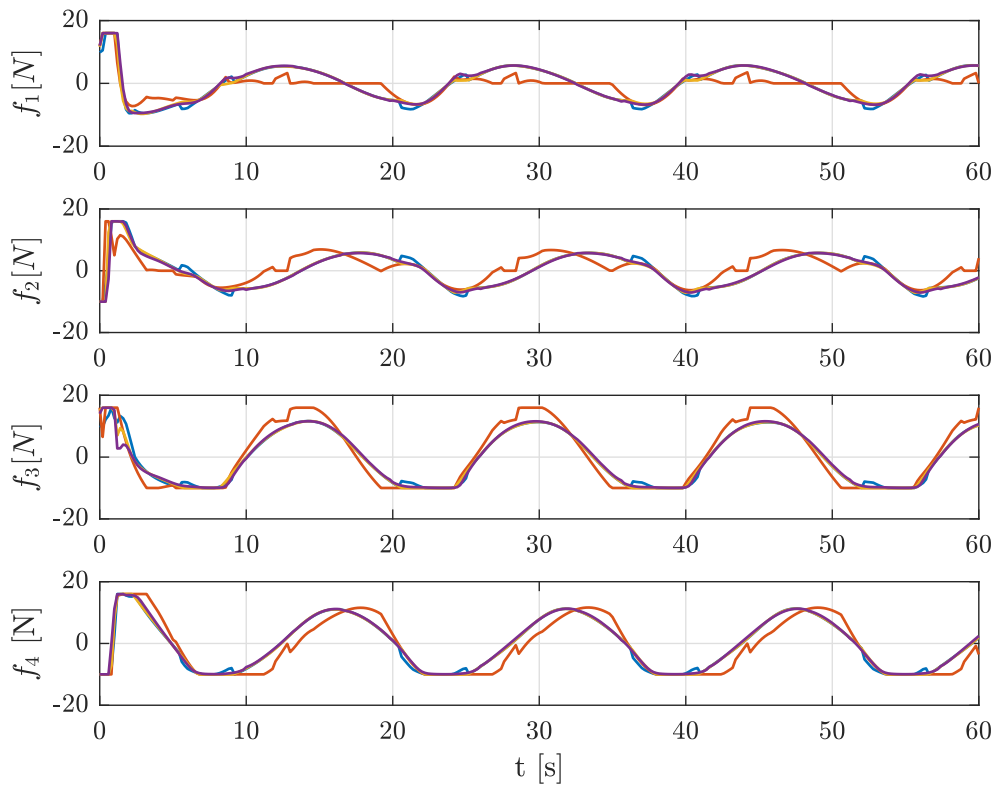


Figure 4.18: Control inputs for the desired trajectory χ_{d1} with $\gamma = 10^3$ (— DCA, — Simplex, — WLSSA, — PDIP).

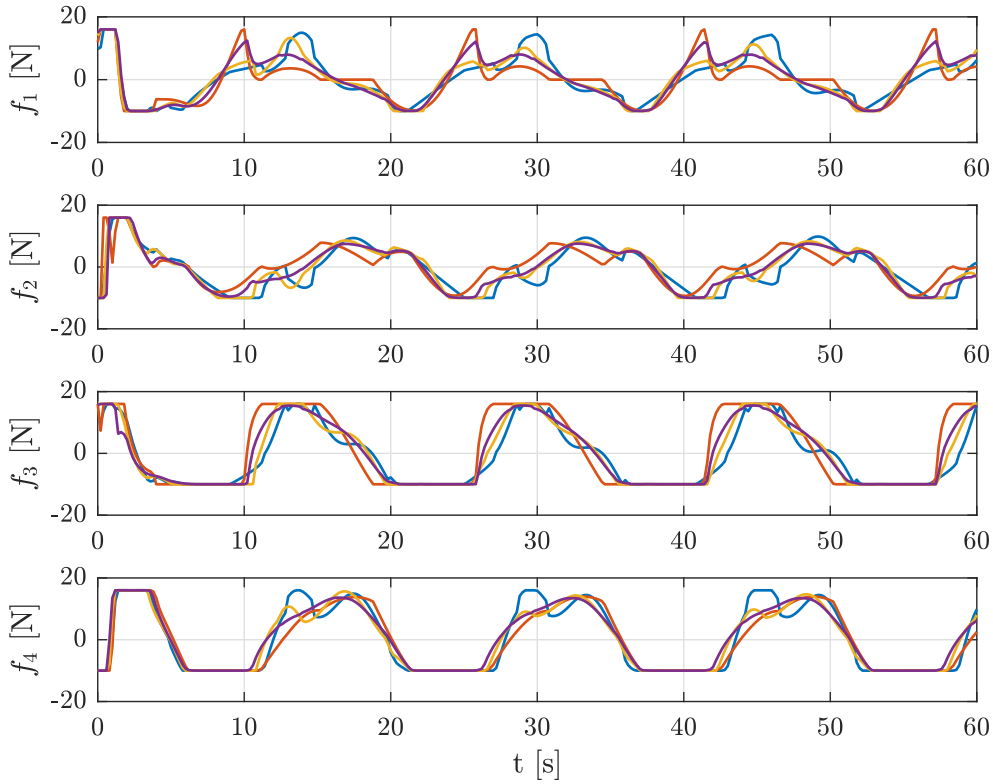


Figure 4.19: Control inputs for the desired trajectory χ_{d2} with $\gamma = 10^3$ (— DCA, — Simplex, — WLSSA, — PDIP).

and f_4 present also saturation in the upper bound $u_{i,max}$, besides presenting also the saturation issues observed in χ_{d1} . In the first instants of the simulation, every control allocation distributes the controls similarly, and their differences can be noticed only close to the the upper limits $u_{max,i}$. For instance, the Simplex algorithm stands out, since it tends to saturate the propellers 1 and 3 in the upper limits, whereas the remaining ones never reach such condition.

On the other hand, the DCA presents also a different pattern for the forces f_1 and f_4 - in the propeller 1, the maximum forces delivered by the DCA present some delay, hitting values close to saturation levels while the other CA algorithms are in a descending dynamics. In the propeller 4, the DCA allocates the higher forces right after leaving the lower limit $u_{min,i}$, while the other CA algorithms increase the control inputs at a lower rate. Constructively, both propellers point to the same direction, and therefore, there multiple control inputs that result in the same virtual control with respect to x and y , although the dynamics of the yaw angle ψ may be impacted.

The figure 4.20 displays even more demanding control inputs, specially for the propellers 3 and 4, which are remain saturated along all the simulation for all CA algorithms, whereas the propellers 1 and 2 remain capable of providing forces in the

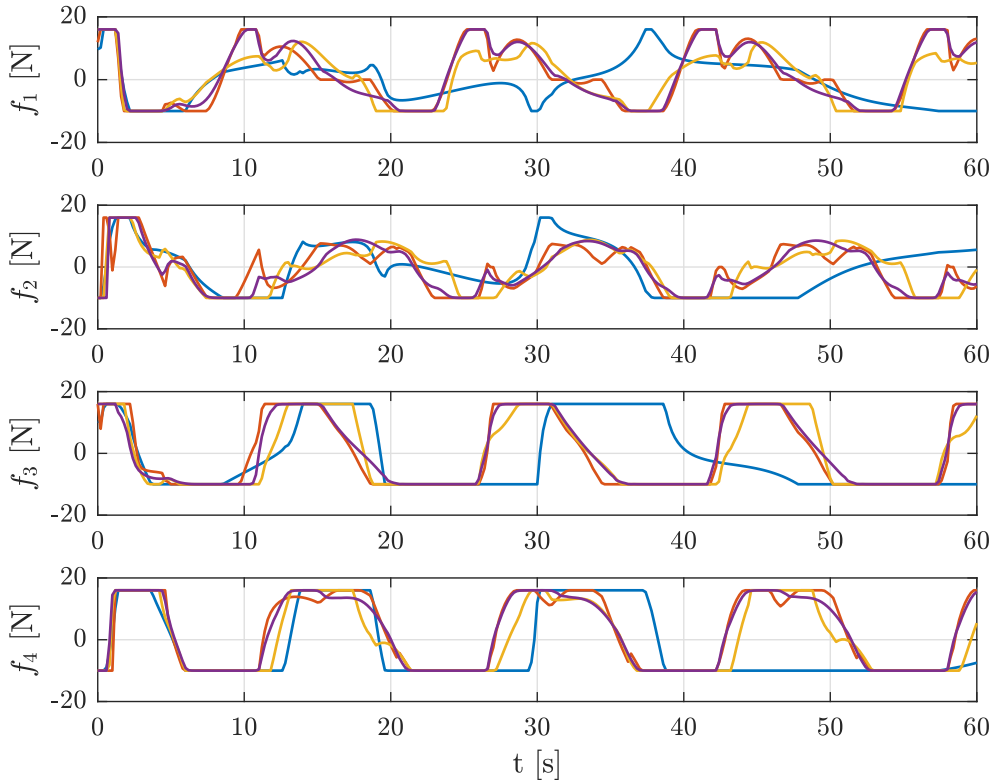


Figure 4.20: Control inputs for the desired trajectory χ_{d3} with $\gamma = 10^3$ (— DCA, — Simplex, — WLSSA, — PDIP).

linear range. However, the control inputs for the DCA stands out and highlights the instability of the vehicle - the dynamics of the propellers significantly differ from the dynamics presented by the other CA algorithms, which indicates that the robot is traveling towards an undesired direction.

Another point is that the optimal solutions of the CA algorithm differ among themselves as the control requirements become more demanding. However, there is a limit to which control allocation can handle saturation properly. The virtual controls may become so degraded that the instability cannot be avoided, although the sensibility changes according to the CA algorithm proposed.

Considerations

The performance of the control allocation algorithms are now mathematically analyzed. For this purpose, we calculate the Virtual Control Error (VCE) and the Direction Error (DE) according to the equation 2.56. The first 9 entries of ν_d and u were excluded since they present outlier values when compared to the other entries in the vector. The results are presented in the tables 4.10, 4.11 and 4.12. For the PDIP, we have considered the data collected for $\gamma = 10^3$.

In the table 4.10, the errors are quite small, provided that the ROV travels mostly

with propellers in the linear region. In general, the average DE is quite neglectable for all the algorithms, although some considerations can be made concerning VCE, which presents the highest maximum and average values for the PDIP, followed by Simplex. The DCA and WLSAS present similar results. If the WLSAS had been run with a identity matrix for W_v , their results would have been similar to those obtained for the PDIP. Therefore, a weighting matrix adopted provided better results in this situation.

Table 4.10: Virtual control errors (VCE) and Distance Error (DE) in the ROV LUMA when $\chi_{d,1}$.

Algorithm	VCE		DE	
	Maximum	Average	Maximum	Average
DCA	1.37	0.09	0	0
Simplex	1.87	0.12	0.10	0.01
WLSAS	1.18	0.09	0.02	0.01
PDIP	2.16	0.26	0.10	0.01

On the other hand, the table 4.11 presents higher values for all the errors VCE and DE. The trajectory is more demanding the the DCA presents the highest errors among the CA algorithms. In fact, the performance deteriorates fast as the trajectory becomes faster. In the table 4.12 confirms it by means of a huge error, corresponding to the stability that the ROV presents in the figure 4.12 for the desired trajectory $\chi_{d,3}$.

Table 4.11: VCE and DE of the CA algorithms in the ROV LUMA when $\chi_{d,2}$.

Algorithm	VCE		DE	
	Maximum	Average	Maximum	Average
DCA	31.57	7.31	0	0
Simplex	18.39	3.19	0.72	0.13
WLSAS	17.57	3.38	0.23	0.04
PDIP	14.88	2.59	0.46	0.08

When comparing the QP solvers, i.e WLSAS and PDIP, we can notice that apparently the WLSAS presents better dynamics mainly because of the response observed in the yaw angle. However, the virtual controls f_x and f_y for the PDIP presents less signal deterioration than the WLSAS and hence presents a lower VCE. The same observation can be made between the Simplex and the WLSAS as well, since the Simplex and PDIP algorithms delivered similar results, while the WLSAS presents a higher VCE. However, among the three algorithms, the WLSAS presented

the lowest DE, despite virtual controls present deterioration for the forces f_x and f_y .

Table 4.12: VCE and DE of the CA algorithms in the ROV LUMA when $\chi_{d,3}$.

Algorithm	VCE		DE	
	Maximum	Average	Maximum	Average
DCA	1.30e3	330.13	0	0
Simplex	35.46	7.96	0.90	0.20
WLSAS	41.86	14.03	0.40	0.09
PDIP	34.20	7.77	0.72	0.17

4.4 Quadrotor

Now we proceed to the simulation of the quadrotor detailed in Chapter 3 to demonstrate how control allocation can handle saturation imposed by the demanding trajectories. The parameters utilized for modeling the quadrotor are presented in the table 4.13.

From these parameters allied to the equation 3.51, the resulting control effectiveness matrix is

$$B = \begin{bmatrix} 0.8704 & 0.8704 & 0.8704 & 0.8704 \\ 0 & -0.1323 & 0 & 0.1323 \\ 0.1323 & 0 & -0.1323 & 0 \\ -0.1399 & 0.1399 & -0.1399 & 0.1399 \end{bmatrix} \quad (4.10)$$

with $\text{rank}(B)=4$. In order to bring the control allocation problems the quadrotor context, the desired virtual control vector ν_d and the control input vector u are

Description	Parameter	Unit	Value
Vehicle body mass	m	kg	0.89
Inertia matrix	I	$10^{-3} \text{ kg}\cdot\text{m}^2$	diag(9.6, 1.86, 25.5)
Rotor inertia	I_r	$10^{-5} \text{ kg}\cdot\text{m}^2$	6.0
Arm length	l	m	0.152
Rotor thrust coefficient	κ_t	$10^{-5} \text{ N}\cdot\text{s}^2$	8.74
Rotor drag coefficient	κ_d	$10^{-7} \text{ N}\cdot\text{s}^2$	1.4
DC Gain	K_m	-	1
Motor time constant	τ_m	s	0.02
Min. velocity of propellers	ω_{min}	rad/s	0
Max. velocity of propellers	ω_{max}	rad/s	230

Table 4.13: Parameters of the quadrotor.

declared as

$$\nu_d = \begin{bmatrix} T_{z,d} \\ \tau_{\phi,d} \\ \tau_{\theta,d} \\ \tau_{\psi,d} \end{bmatrix}, \quad u = \begin{bmatrix} \omega_1^2 \\ \omega_2^2 \\ \omega_3^2 \\ \omega_4^2 \end{bmatrix} \quad (4.11)$$

Provided that $u_i = \omega_i^2$, the lower and upper control limits are given by $u_{min,i} = 0$ and $u_{max,i} = 52,900 \text{ rad}^2/\text{s}^2$. Thus, from equation 3.48, the forces and torques produced by the propellers are located in the range

$$\begin{aligned} 0 &\leq f_i \leq 4.60 \quad \text{N}, \\ 0 &\leq \tau_i \leq 0.74 \quad \text{N} \cdot \text{m} \end{aligned} \quad (4.12)$$

The full configuration of the quadrotor in any time instant t is given by the vector

$$\chi = \begin{bmatrix} x \\ y \\ z \\ \phi \\ \theta \\ \psi \end{bmatrix}, \quad \chi \in SE(3) \quad (4.13)$$

for describing its position and orientation in the 3D Euclidean space.

Initially, the quadrotor is located at $\chi = [0 \ 0 \ 0 \ 0 \ 0 \ 0]^T$ and it is required to follow a parabola-shaped trajectory, originated from the reference signal

$$\chi_r(t) = \begin{cases} x_r &= \sin(\frac{2\pi}{3}t + \frac{\pi}{2}) \\ y_r &= \sin(\frac{\pi}{3}t) \\ z_r &= 1 \\ \psi_r &= 0 \end{cases} \quad (4.14)$$

whose values at every instant t are input of the reference model presented in the equation 3.2.5 to generate the desired trajectories as output. As previously mentioned, the reference values ϕ_r and θ_r are internally computed by the PD Controller in a cascaded implementation and are input for the Integral Backstepping controller. The reference model is used for generating the desired position, velocity and acceleration of every DoF of the system.

The simulation is executed for three different sets of filter poles k_{f1} and k_{f2} for the reference model, with the purpose of generating trajectories with different control requirements. These parameters are

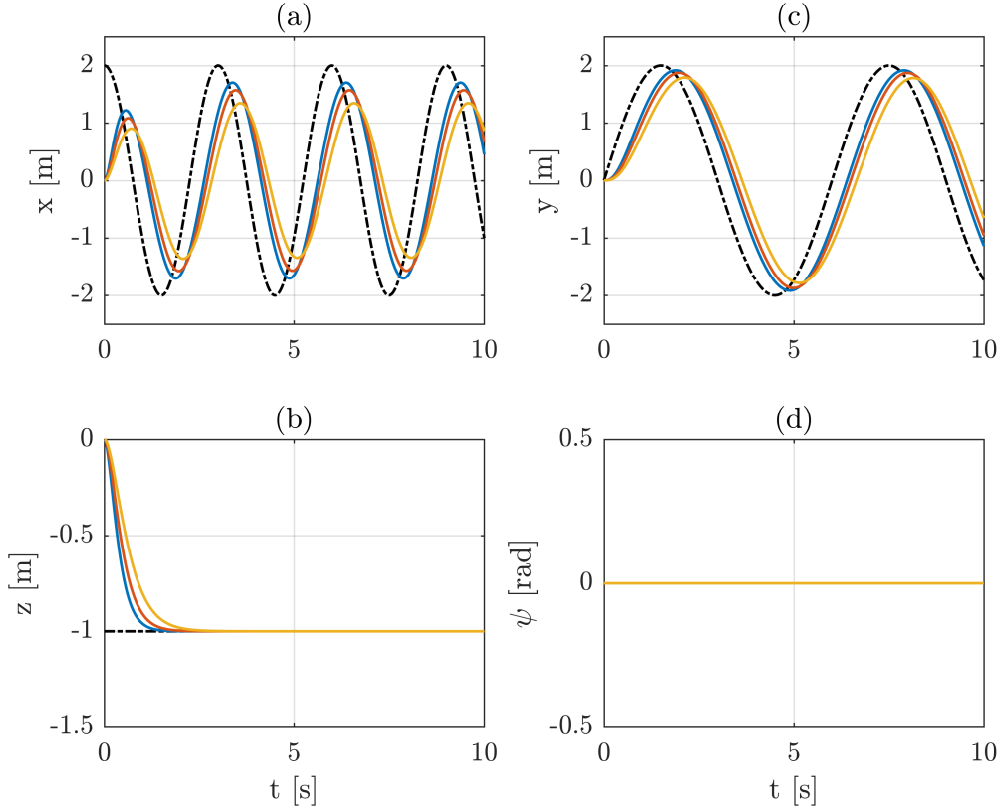


Figure 4.21: The reference signal $r(t)$ and the three desired trajectories for the quadrotor. (--- reference signal χ_r , — desired trajectory $\chi_{d,1}$, — desired trajectory $\chi_{d,2}$, — desired trajectory $\chi_{d,3}$).

- desired trajectory $\chi_{d,1}$: $k_{f1} = k_{f2} = 30$ for ϕ_d and θ_d , and $k_{f1} = k_{f2} = 3$ for ψ_d , z_d , x_d and y_d ;
- desired trajectory $\chi_{d,2}$: $k_{f1} = k_{f2} = 40$ for ϕ_d and θ_d , and $k_{f1} = k_{f2} = 4$ for ψ_d , z_d , x_d and y_d ;
- desired trajectory $\chi_{d,3}$: $k_{f1} = k_{f2} = 50$ for ϕ_d and θ_d and $k_{f1} = k_{f2} = 5$ for ψ_d , z_d , x_d and y_d ;

The figure 4.21 displays the resulting desired trajectories $\chi_{d,i}$ and shows their dynamics in the simulation. The gains of the PD and Integral Backstepping controllers used in the simulation are presented in the table 4.14. The simulation lasts $t_s = 10$ seconds and the sampling time is 0.001 seconds. More information concerning this quadrotor model can be consulted on MONTEIRO (2015).

The figure 4.22 illustrates the dynamics of the UAV for the three trajectories considered when saturation is not managed by any control allocation technique. As a result, the desired virtual controls ν_d produced by the High-level controller correspond to a control input vector u beyond the operating range of the rotors. The High-level controller then tries to recover the UAV back to the desired trajectory

IB Roll		IB Pitch		IB Yaw		PD	
Parameter	Value	Parameter	Value	Parameter	Value	Parameter	Value
k_1	20	k_4	20	k_7	5.5	k_p	0.25
k_2	0	k_5	0	k_8	0.1	k_d	0.4
k_3	20	k_6	20	k_9	4	-	-

Table 4.14: Gains of the Integral Backstepping and Proportional-Derivative Controllers.

at the cost of generating higher control inputs successively. Finally, every propeller becomes saturated whether in the lower limit or in the upper, deviating the vehicle from the desired trajectory and leading it to instability.

Direct Control Allocation

The figure 4.23 depicts the dynamics of the vehicle during all simulation time and displays also the rotation speed of the propellers for every desired trajectory. The DCA always maintains the direction of the virtual controls while keeps them inside feasible set of virtual controls \mathbb{A} . However, there are situations where direction cannot be kept because of the quadratic characteristic of the forces and torques (3.48). The inverse mapping from ν_d to u_d can lead to negative control inputs and hence its square roots cannot be computed. In this case, a feasible solution does not exist because their square roots are not real numbers. If a control input u_i results in an angular velocity below $\underline{\omega} = 0$ and another control input demands an angular velocity above the upper limit $\bar{\omega}$, the DCA cannot deliver a proper solution, but rather a null vector. Thus, it is not possible to adjust both velocities and still maintain the direction in such situation.

As time elapses, the reference model generates the desired trajectory, velocity and acceleration for each DoF, until u_d becomes nonnegative and the DCA can compute the a valid control input vector u , such that motion is produced. If the initial pose of the UAV is on the ground, the vehicle remains still for a while, whereas if it starts in the air, it would undergo a free fall initially, as depicted in the plot (c), until the desired trajectories $\chi_{d,i}$ result again in nonnegative control inputs. Therefore, the UAV can accomplish the task in trajectory 1 and 2 with some reservations, since a null control input implies in a quadrotor out of service only in the initial instants of the simulation. Another trajectory could avoid this issue and the DCA would fully work without any reservation.

As depicted in the figure 4.24, initially the propellers do not rotate since the DCA finds only a scale factor $\alpha = 0$, but soon the actual and the desired states produce a valid control input and the propellers start rotating, while prioritize the virtual controls T_z , τ_ϕ and τ_ψ , in this sequence. Although the DCA can provide

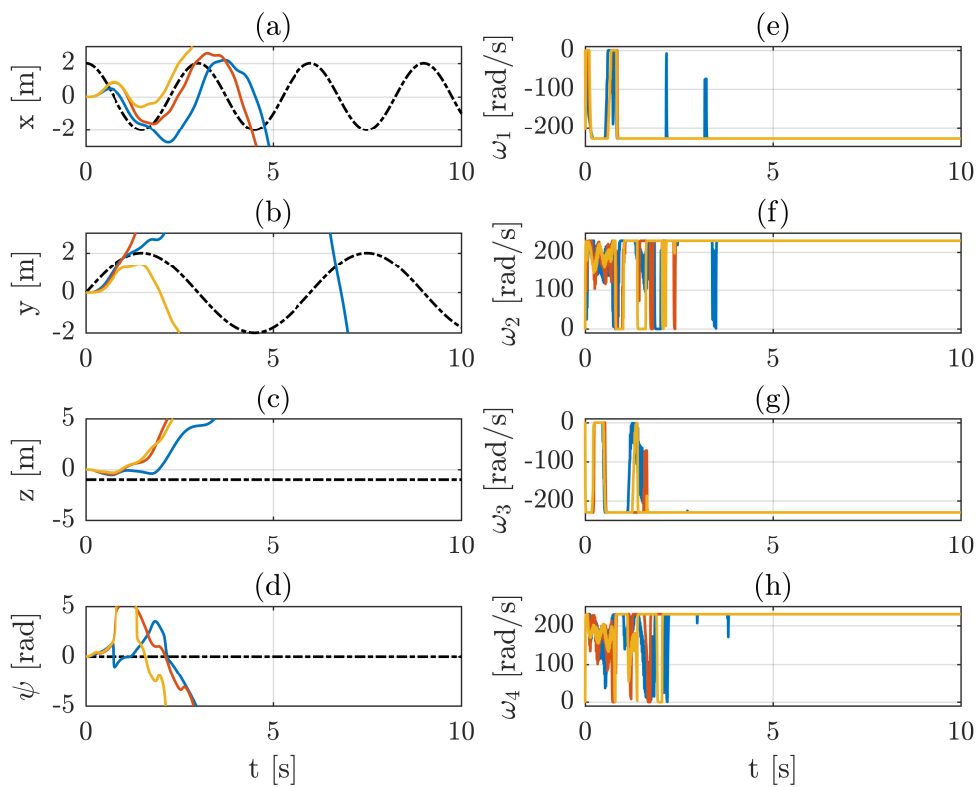


Figure 4.22: Dynamics of the quadrotor when saturation is not managed by any control allocation technique. (--- reference χ_r , — trajectory traveled χ_1 , — trajectory traveled χ_2 , — trajectory traveled χ_3 —).

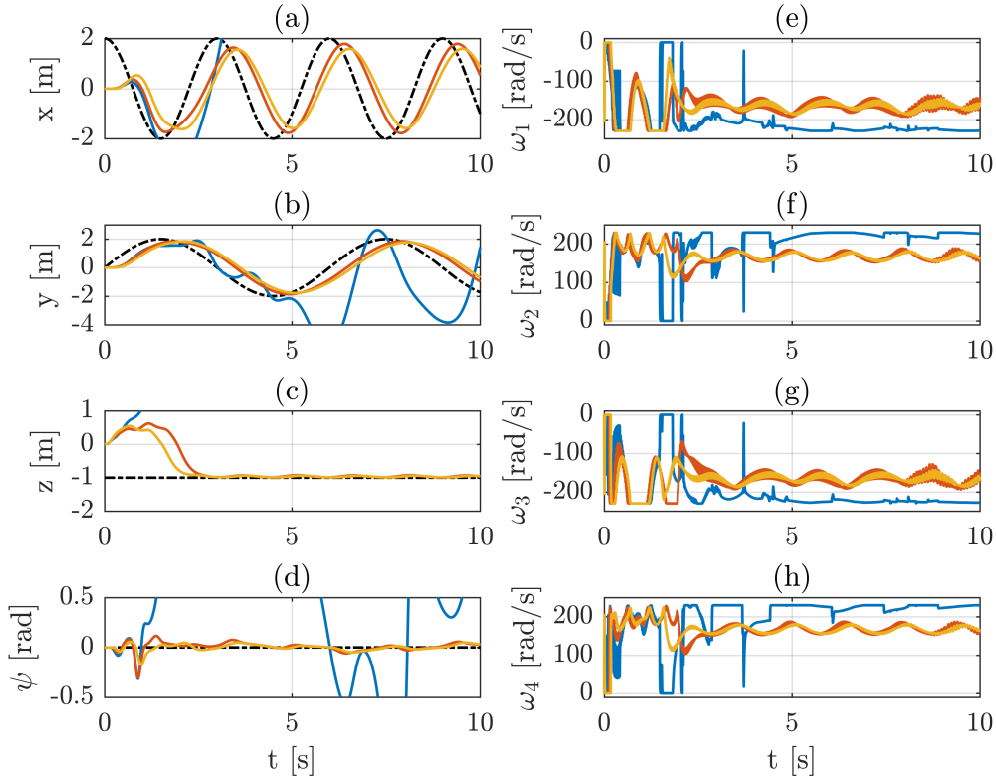


Figure 4.23: Results for the Direct Control Allocation (--- reference χ_r , — trajectory traveled χ_1 , — trajectory traveled χ_2 , — trajectory traveled χ_3).

tracking in the initial instants of χ_3 , from the time $t = 1.5$ s on, the dynamics of the quadrotor becomes unstable and it is not able to fly back towards the desired trajectory, since all propellers work at saturation levels and the High-level controller produces a high thrust T_z in an attempt to correct the height and position with respect to x and y .

Linear Programming with Simplex

The simulation for the linear programming with Simplex is depicted in figure 4.25. The vehicle could track the desired trajectories properly, even for the trajectory 3, where the DCA is not able to perform tracking. The Simplex algorithm is able to find an optimal feasible solution during all simulation time and a demonstrates robustness to manage input saturation. The figure also shows that the initial instants are the most concerning in terms of saturation - the High-level controller is concerned at stabilizing every DoF, and therefore, the propellers are taken to its upmost bounds to fulfill the control requirements, as it can be observed for the plots (e), (f), (g) and (h), where every propeller starts at a saturated position.

The more the altitude of the vehicle approximates the desired value, the less the

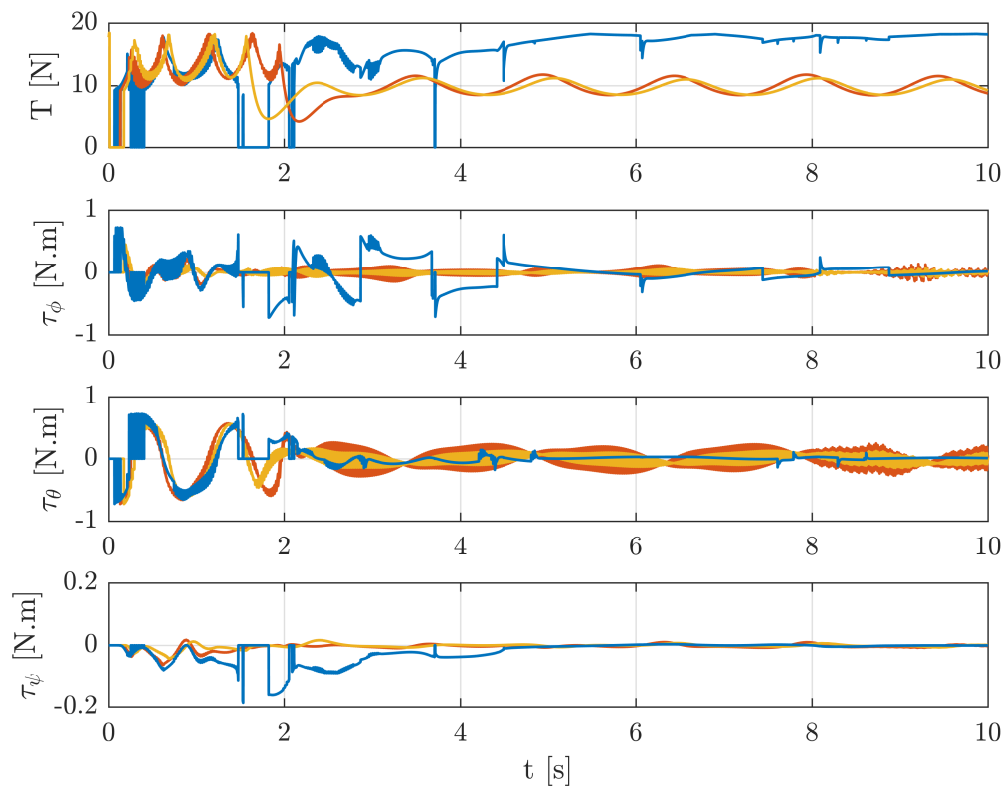


Figure 4.24: Virtual controls for the DCA (— trajectory traveled χ_1 , — trajectory traveled χ_2 , — trajectory traveled χ_3).

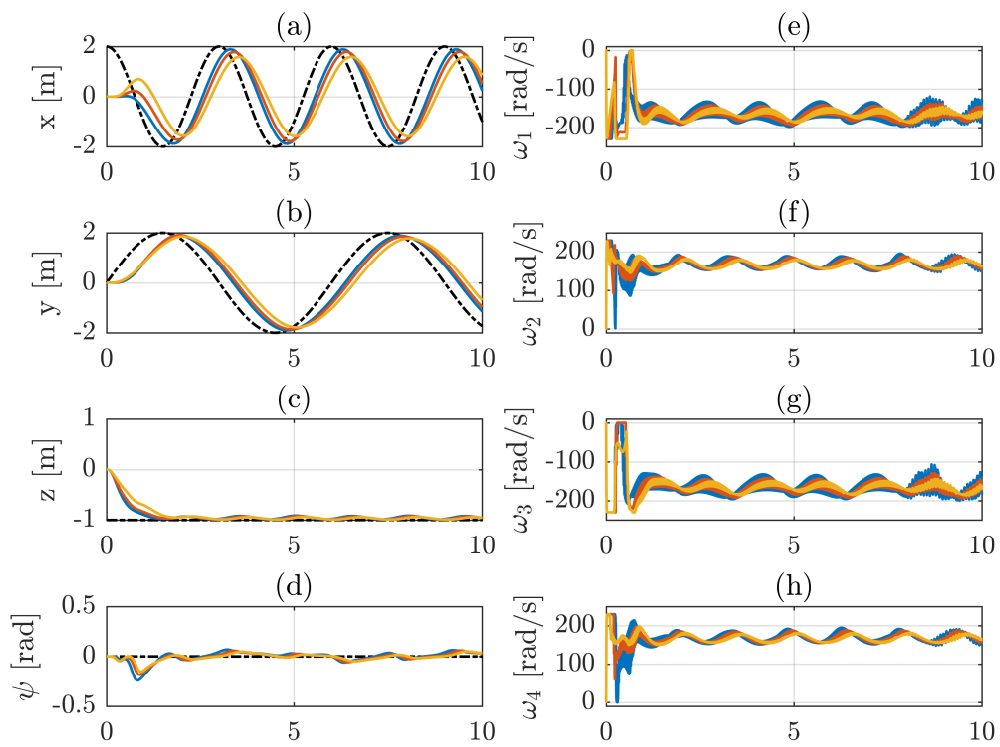


Figure 4.25: Results for the linear programming with Simplex (--- reference χ_r , — trajectory traveled χ_1 , — trajectory traveled χ_2 , — trajectory traveled χ_3).

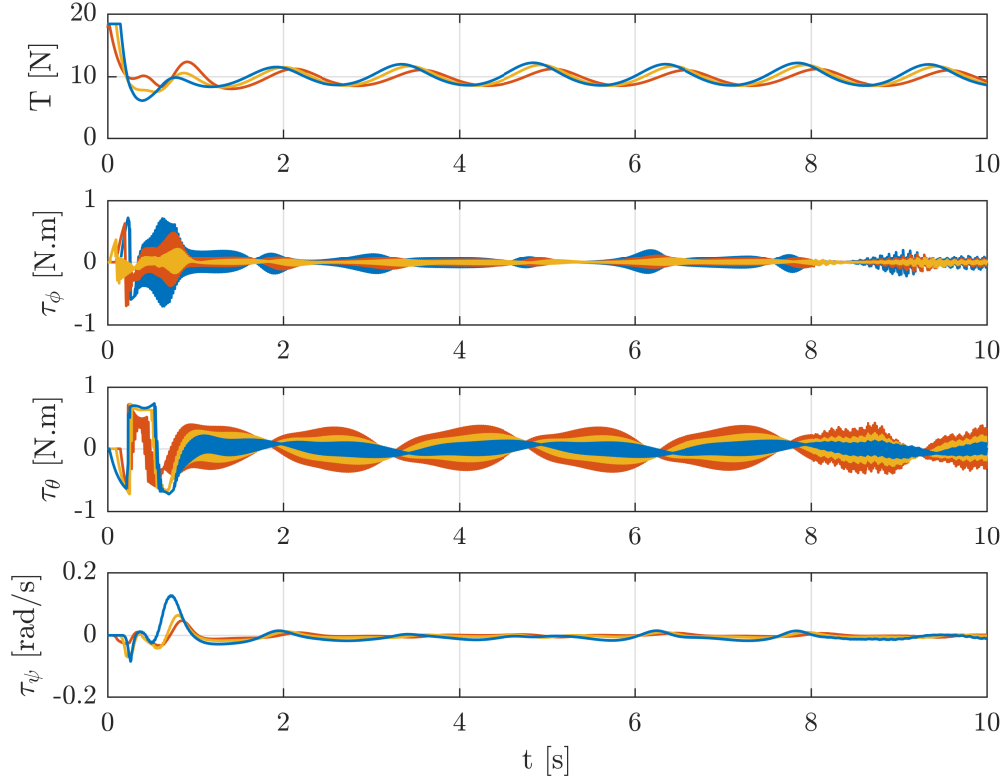


Figure 4.26: Virtual controls for the linear programming with Simplex (— trajectory traveled χ_1 , — trajectory traveled χ_2 , — trajectory traveled χ_3).

propellers are demanded and the priority to track x_d and y_d increases, as one can observe in figure 4.24 in the plots for τ_θ , τ_ϕ and to a lesser extent the torque τ_ψ . From this moment on, the trajectory becomes less demanding and the propellers can operate fully inside its operating range.

By analyzing the plots of the virtual inputs ν and propeller speeds ω_i , one can notice that the more demanding is the trajectory, the bigger is the range that the signals oscillate. It is not present in every ω_i , although it is quite intense the plots of ν .

Weighted Least Squares

The PDIP and the WLAS algorithms were also employed to allocate controls with the quadrotor. However, none of the algorithms were able to handle saturation, even for less demanding trajectories. The simulation has been run for the desired trajectory $\chi_{d,1}$ and the results are displayed in the figure 4.27.

Several parameters were tried to make the system stable for both algorithms. The figures display results for the WLSAS run with weighting matrices $W_v = \text{diag}(5, 2, 2, 2)$, $W_v = \text{diag}(50, 10, 10, 10)$ and $W_v = \text{diag}(1, 1, 1, 1)$, and for the PDIP the simulation was run with prioritization factor $\gamma = 10^4$.

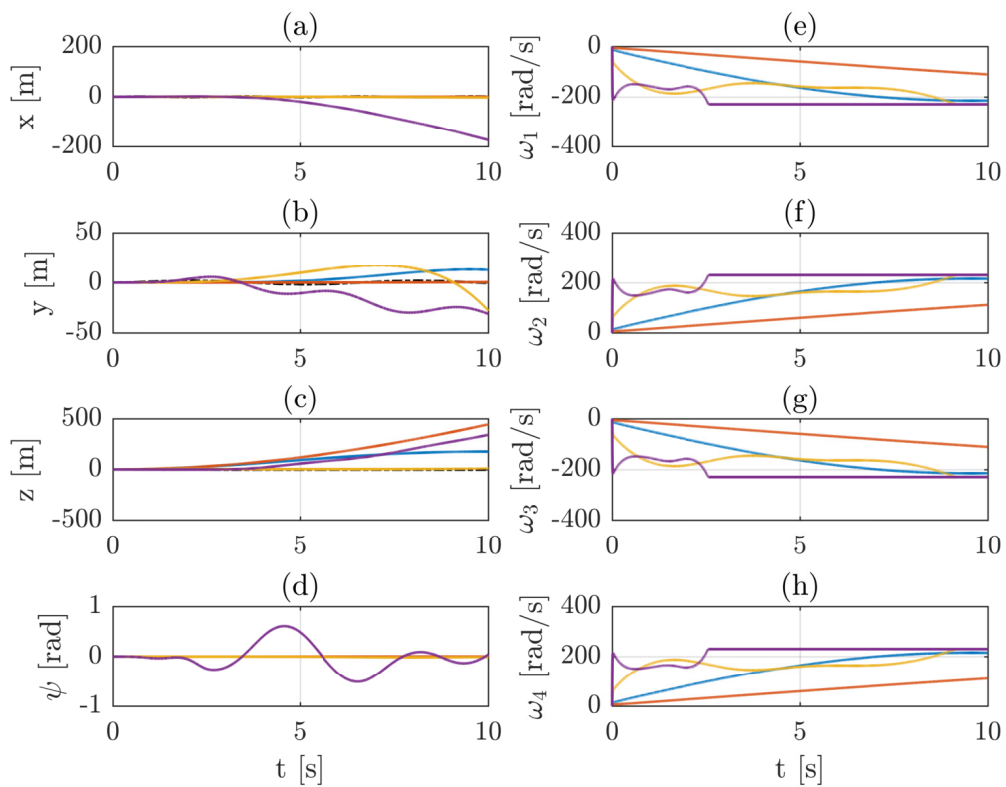


Figure 4.27: Results for the WLSIP and PDIP (- - - desired trajectory $\chi_{d,1}$,
— WLSAS with $W_v = \text{diag}(1, 1, 1, 1)$, — WLSAS with $W_v = \text{diag}(5, 2, 2, 2)$,
— WLSAS with $W_v = \text{diag}(50, 10, 10, 50)$, — PDIP with $\gamma = 10^4$).

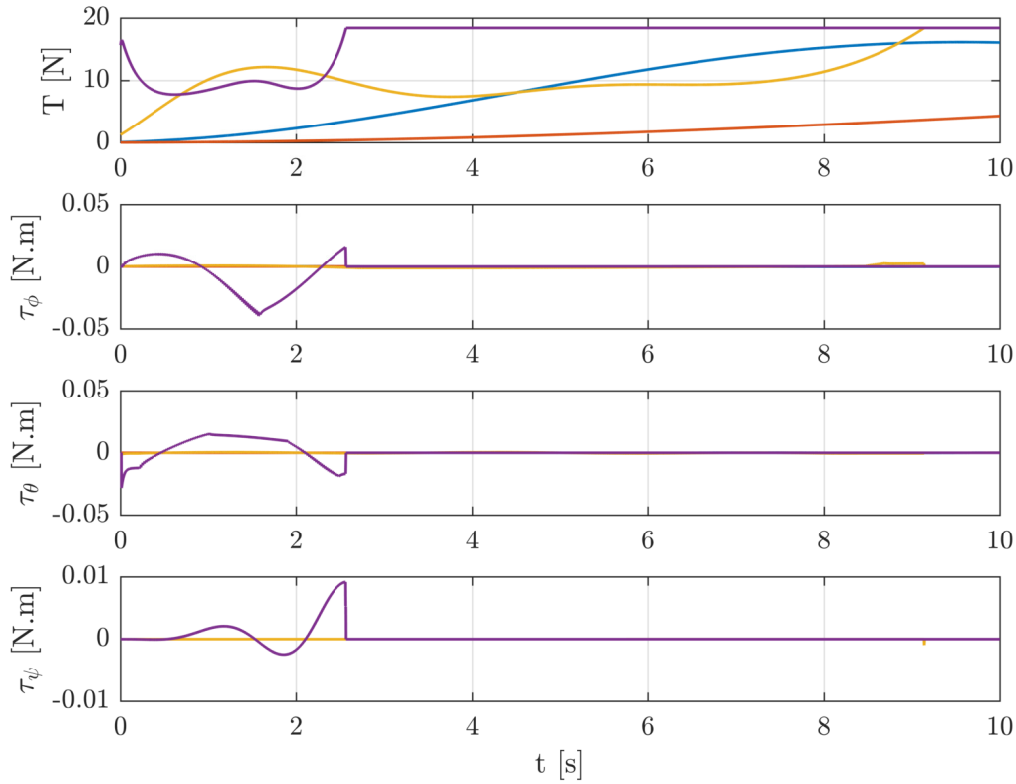


Figure 4.28: Dynamics of the virtual controls for the WLSAS and PDIP (--- desired trajectory $\chi_{d,1}$, — WLSAS with $W_v = \text{diag}(1, 1, 1, 1)$, — WLSAS with $W_v = \text{diag}(5, 2, 2, 2)$, — WLSAS with $W_v = \text{diag}(50, 10, 10, 50)$, — PDIP with $\gamma = 10^4$).

The figure 4.28 presents the dynamics of the virtual controls for the simulations with the WLSAS and PDIP. The weighting vector $W_v = \text{diag}(1, 1, 1, 1)$ presents a slow dynamics for both algorithms. Other attempts to accelerate it by means of the other W_v matrices led the vehicle to unstable behavior either. For instance, running the WLSAS with weighting matrix $W_v = \text{diag}(50, 10, 10, 50)$ led the rotors to saturation at $t = 2.5$ seconds and it remained so along all the simulation, whereas for $W_v = \text{diag}(5, 2, 2, 2)$, the rotors became saturated only near the end of simulation.

Considerations

The UAV has presented instability for the WLSAS and PDIP, and therefore, it is not possible to extract meaningful information concerning the algorithms and compare them with the DCA and Simplex. Therefore, their errors are not displayed in the tables 4.15, 4.16 and 4.17 to come.

Numerically, the Simplex has presented better results not only for the Virtual Control Error (VCE), but also for the Direction Error (DE). The better results for VCE were expected provided that Simplex is concerned with the error minimization

in its cost function. However, since the DCA cannot find a solution that maintains the direction of the virtual control input ν_d when the rotor is required to rotate reversely and the unique possible control input u was a null vector, the normalized resulting virtual control input vector ν is $0 \in \mathbb{R}^4$. Initially the quadrotor is also required to ascend vertically, which results in a normalized $\nu_d = \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix}^T$. This limitation in the DCA impacted the DE computed. Although the Simplex algorithm tends to present DE at every control loop, the maximum and average errors presented better overall results.

Table 4.15: VCE and DE of the CA algorithms with the desired trajectory $\chi_{d,1}$ in the quadrotor .

Algorithm	VCE		DE	
	Maximum	Average	Maximum	Average
DCA	42.66	4.73	1	1.6e-2
Simplex	7.52	3.19	0.38	7.5e-3

Table 4.16: VCE and DE of the CA algorithms with $\chi_{d,2}$ in the quadrotor .

Algorithm	VCE		DE	
	Maximum	Average	Maximum	Average
DCA	50.18	6.88	1	1.2e-2
Simplex	15.15	3.19	0.48	8.4e-3

Table 4.17: VCE and DE of the CA algorithms with $\chi_{d,3}$ in the quadrotor .

Algorithm	VCE		DE	
	Maximum	Average	Maximum	Average
DCA	8.2e3	2.5e3	1	5.0e-2
Simplex	26.19	0.13	0.54	7.5e-3

4.5 Cooperative Manipulators

The manipulator simulated in this work is the UR5[®], a 6-DoF manipulator manufactured by Universal Robots, as shown in figure 4.29. It is a robot-arm like manipulator, designed for cooperative tasks, highly customizable and weighs approximately

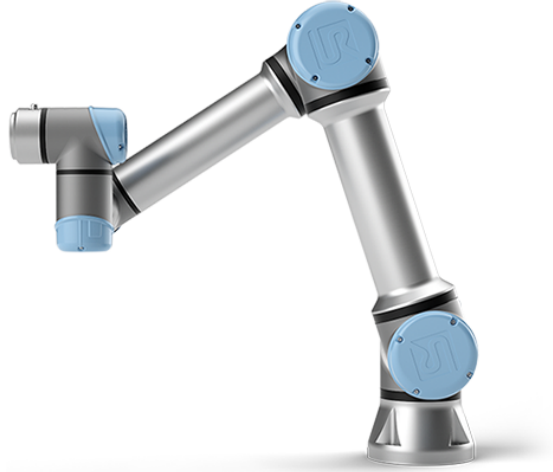


Figure 4.29: The UR5 manipulator made by Universal Robots.

18 kg. It can be employed in overall industry activities and it admits a maximum payload of 5kg.

To simulate properly the manipulator, the first step is to obtain its Denavit-Hartenberg parameters, which were extracted from the manufacturer website and are presented in the table 4.18.

Table 4.18: Denavit-Hartenberg Parameters of the UR5 manipulator.

Link	θ [rad]	a [m]	d [m]	α [rad]	M (kg)
1	0	0	0.089159	$\pi/2$	3.7
2	0	-0.425	0	0	8.393
3	0	-0.39225	0	0	2.33
4	0	0	0.10915	$\pi/2$	1.219
5	0	0	0.09465	$-\pi/2$	1.219
6	0	0	0.0823	0	0.1879

For the cooperative task, the UR5 model of the Peter Corke's Robotics Toolbox version 10.2 is utilized and improved for simulating also the dynamics of the robot and to allow it to be controlled by torque. The inertia matrices and their perspectives centers of mass were obtained from the Model C presented in KUFETA (2014), where the author divided the UR5 links into cylinders, such that their resulting centers of mass approximates those disclosed by the manufacturer. The manipulators are denoted UR5₁ and UR5₂ and are located at $\begin{bmatrix} -0.4 & 0 & 0 \end{bmatrix}^T$ and $\begin{bmatrix} 0.4 & 0 & 0 \end{bmatrix}^T$ with respect to the inertial frame, as depicted in figure 4.30. The manipulators are also not attached to any vehicle and hence cannot move their bases with respect to the inertial frame \mathcal{F}_o .

¹Figure extracted from <https://www.universal-robots.com/products/ur5-robot/>

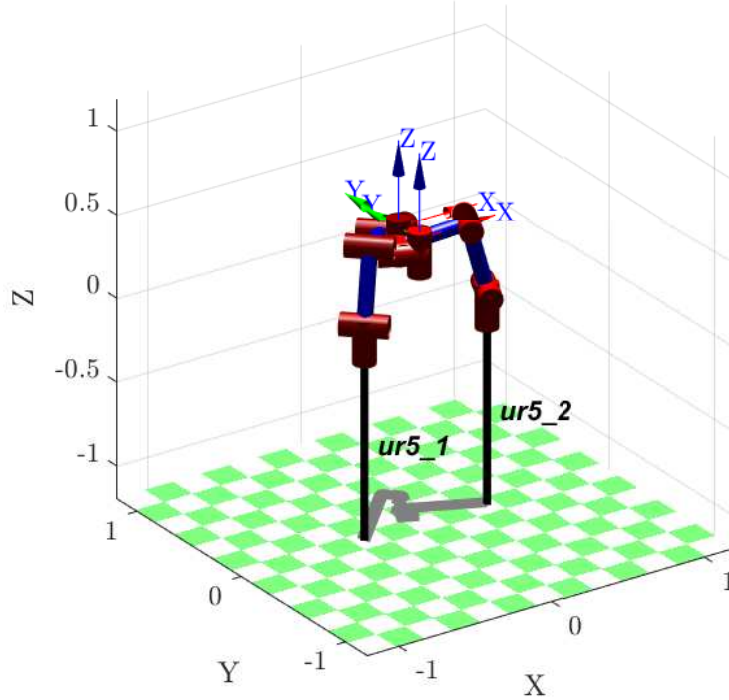


Figure 4.30: Two UR5 manipulators in a cooperative task.

Parameter	Description	Unit	Value
M	Mass	kg	7
M_o	Inertia matrix	kg.m ²	$\text{diag}(M\mathcal{I}_3, 0.03\mathcal{I}_3)$
G_o	Gravity	m/s ²	$[0 \ 0 \ mg \ 0 \ 0 \ 0]^T$
l	Length	m	0.2

Table 4.19: Parameters of the object manipulated.

The task that the manipulators must carry out consist of transporting an cube-shaped object with mass $M = 7$ kg, such that every edge has length $l = 0.2m$. The inertia matrix M_o and the gravitational vector G_o to describe the entire object dynamics are presented in the table 4.19. The centrifugal and Coriolis force are neglected here for simplicity.

From the length l and considering that the object body frame is placed exactly at its center of mass \bar{r} , the virtual sticks vectors are therefore given by $r_1 = [-0.1 \ 0 \ 0]^T$ and $r_2 = [0.1 \ 0 \ 0]^T$ and are used for modeling the kinematic constraints, which result in the grasp matrix

$$G = \begin{bmatrix} \mathcal{I}_3 & 0_3 & \mathcal{I}_3 & 0_3 \\ r_1 \times & \mathcal{I}_3 & r_2 \times & \mathcal{I}_3 \end{bmatrix} \quad (4.15)$$

Contact Forces		Impedance Controller	
Parameter	Value	Parameter	Value
k_{pe}	10000	M_d	$\text{diag}(10\mathcal{I}_3, 0.1\mathcal{I}_3)$
k_{de}	25	D_d	$\text{diag}(50\mathcal{I}_3, 10\mathcal{I}_3)$
-	-	K_d	$\text{diag}(500\mathcal{I}_3, 10\mathcal{I}_3)$

Table 4.20: Parameters for modeling the contact forces and the desired object dynamics for the impedance controller.

Controller	Parameter	Gain	Value
CLIK	K_{PV}	Proportional	$\text{diag}(500, 150, 150, 50, 50, 50)$
	K_{DV}	Derivative	$\text{diag}(50, 50, 50, 13.8, 13.8, 13.8)$
PID	K_{PC}	Proportional	$\text{diag}(233, 233, 150, 25, 17, 0.4)$
	K_{IC}	Integral	$\text{diag}(1, 0.73, 0.18, 0.077, 0.11, 0.0027)$
	K_{DC}	Derivative	$\text{diag}(0.77, 1.55, 1, 0.33, 0.17, 0.06)$

Table 4.21: Parameters adopted in the CLIK algorithm and PID Position Controller.

with $\text{rank}(G)=6$.

The task consists of holding the object and transporting it by following a trajectory given by the equation

$$\chi_d(t) = \begin{cases} 0 \\ 0.3 \cos(\omega_n t + \pi/6) - 0.1 \\ 0.4 + 0.2 \cos(\omega_n t + \pi/8) \end{cases} \quad (4.16)$$

The simulation is run for different angular velocities ω_n to impose different control requirements to the cooperative task. The angular velocities ω_n for the desired trajectories are

- $\omega_n = 2\pi/5$ for the desired trajectory $\chi_{d,1}$;
- $\omega_n = 2\pi/4$ for $\chi_{d,2}$;
- $\omega_n = 2\pi/3$ for $\chi_{d,3}$;

When simulation starts, the manipulators are already holding the object, which is located at position $\chi_c(0) = \begin{bmatrix} 0 & 0.1566 & 0.4804 \end{bmatrix}^T$. The simulation lasts $t_s = 10$ s and the sampling time adopted for the simulation is 0.01 s.

The end effectors are demanded to act as an impedance against the object, which is the environment in the task and treated as an admittance. Thus, the desired parameters for the object, such as stiffness and damping, are detailed in table 4.20. Finally, the parameters for the Closed-loop Inverse Kinematics (CLIK) and Computed Torque with PID are presented in table 4.21.

It is worth mentioning that the mass was chosen so that it satisfies the recommendation of the manufacturer - a single manipulator admits a maximal payload of 5 kg. In order to grasp the object proposed, the cooperative task must consist of at least 2 manipulators. Therefore, we have simulated the cooperative task with 2 manipulators to demonstrate how the control allocation can be used for load sharing purposes.

For the control allocation purposes, the desired virtual control input vector ν_d is given by the total load to be exerted on the object, which will be represented by $h_c \in \mathbb{R}^6$ and consists of

$$h_c = \begin{bmatrix} h_x \\ h_y \\ h_z \\ h_\phi \\ h_\theta \\ h_\psi \end{bmatrix} \quad (4.17)$$

and the control input vector u here is denoted h and consists of the forces to be exerted by each the tip of both end effectors, which is

$$h = \begin{bmatrix} h_1 \\ h_2 \end{bmatrix}, i = 1, 2 \text{ and } h_i = \begin{bmatrix} h_{i,x} \\ h_{i,y} \\ h_{i,z} \\ h_{i,\phi} \\ h_{i,\theta} \\ h_{i,\psi} \end{bmatrix} \quad (4.18)$$

Static Load Allocation

The load distribution to each manipulator is performed with the Static Load Allocation for tracking the desired trajectories, as presented in the figure 4.31. Although the object and the end effectors are supposed to follow the desired trajectories, the impedance controller exchanges the accelerations for the desired force in an impedance-admittance relationship, and as a result, the trajectory undergoes a deviation.

The trajectories with respect to Y axis present no deviation, provided that no external forces in this axis are exerted on the object along the simulation. However, there are deviations present with respect to the X axis due to interaction between the manipulators, and to the Z axis, due to the weight of the object. Thus, the impedance controller promotes this deviation in order to deliver the required forces.

Another point that one can notice is the effects of the load sharing policy, which

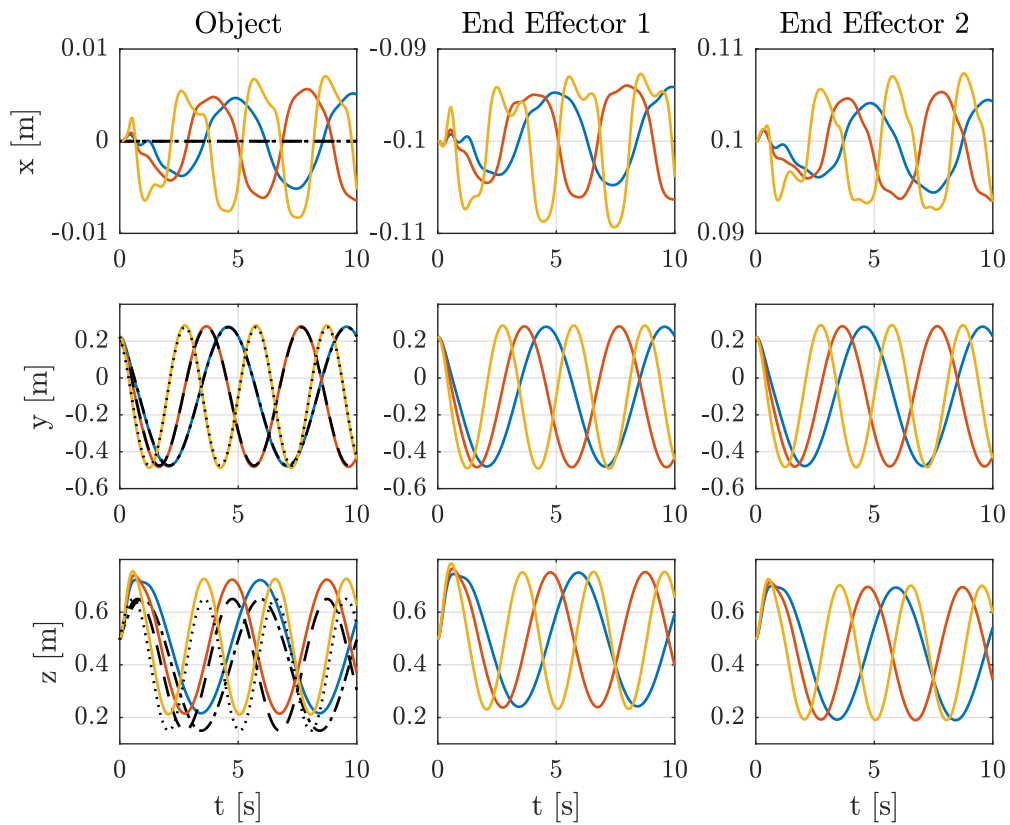


Figure 4.31: Trajectories of the object and of the manipulators UR5₁ and UR5₂ with Static Load Allocation (\dashdash desired trajectory $\chi_{d,1}$, \dash desired trajectory $\chi_{d,2}$, \cdots desired trajectory $\chi_{d,3}$, --- trajectory χ_1 , --- trajectory χ_2 , --- trajectory χ_3).

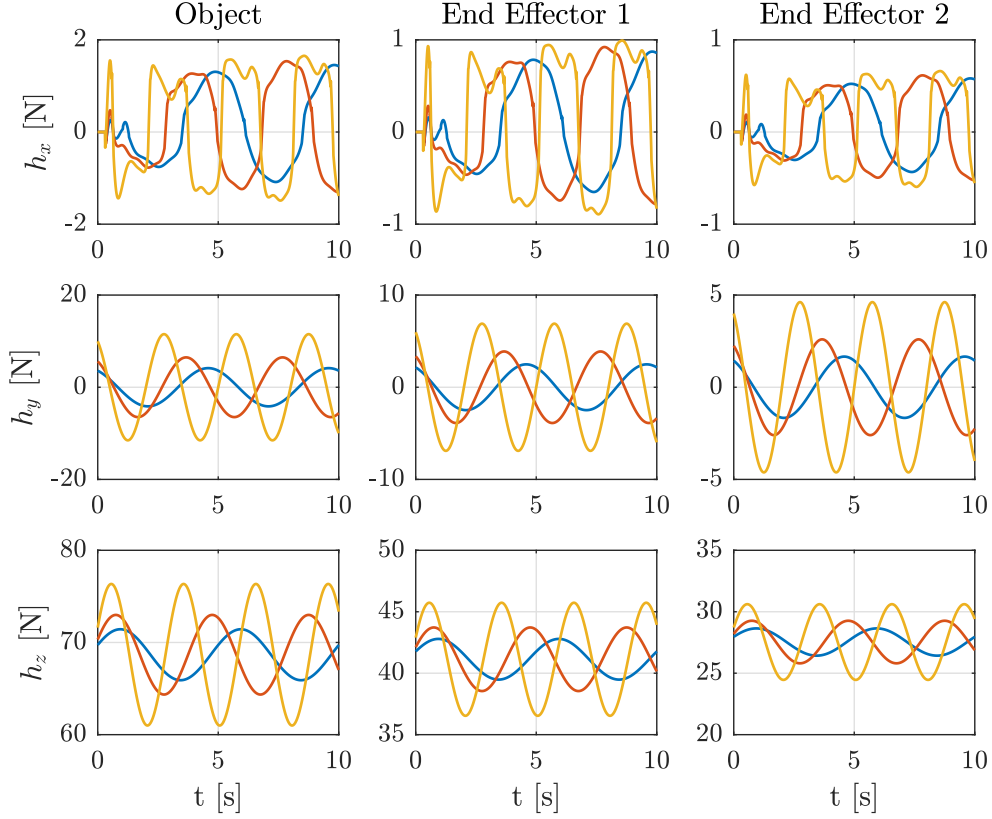


Figure 4.32: Forces acting on the object by the end effectors and the load allocation according to the sharing policy from the Static Load Allocation (— trajectory χ_1 , — trajectory χ_2 , — trajectory χ_3).

allocates 60% of the total load to the manipulator 1 and the remaining load to the manipulator 2. As the manipulator 1 must exert a higher force along this direction to compensate the gravity effects, its trajectory undergoes a deviation, reaching its peak at a height $z = 0.75$ m, whereas the manipulator 2 reaches values close to $z = 0.70$ m. On the other hand, the lowest heights performed by the end effector 1 reaches values as low as $z = 0.24$ m, whereas the end effector 2 reaches $z = 0.19$ m.

The figure 4.32 depicts all the forces involved in the interaction between the manipulators and the object. The forces on the X axis are impacted by the contact forces, which are depicted in 4.33. The contact forces result also in external forces, that deviate the deviate the object from the desired trajectory due to the impedance-admittance relation. We can also observe that the forces on the object with respect to the X axis increase as the trajectory becomes faster. The same observation can be made about the forces on the Y and Z axis, which also gain intensity because of the relationship force-acceleration in the object dynamics.

In the plot (a) of the figure 4.33, one can observe that the object is squeezed along all the simulation, except for the initial instants, when the end effectors are stabilizing their contacts on the object to start the trajectory. For the record, we

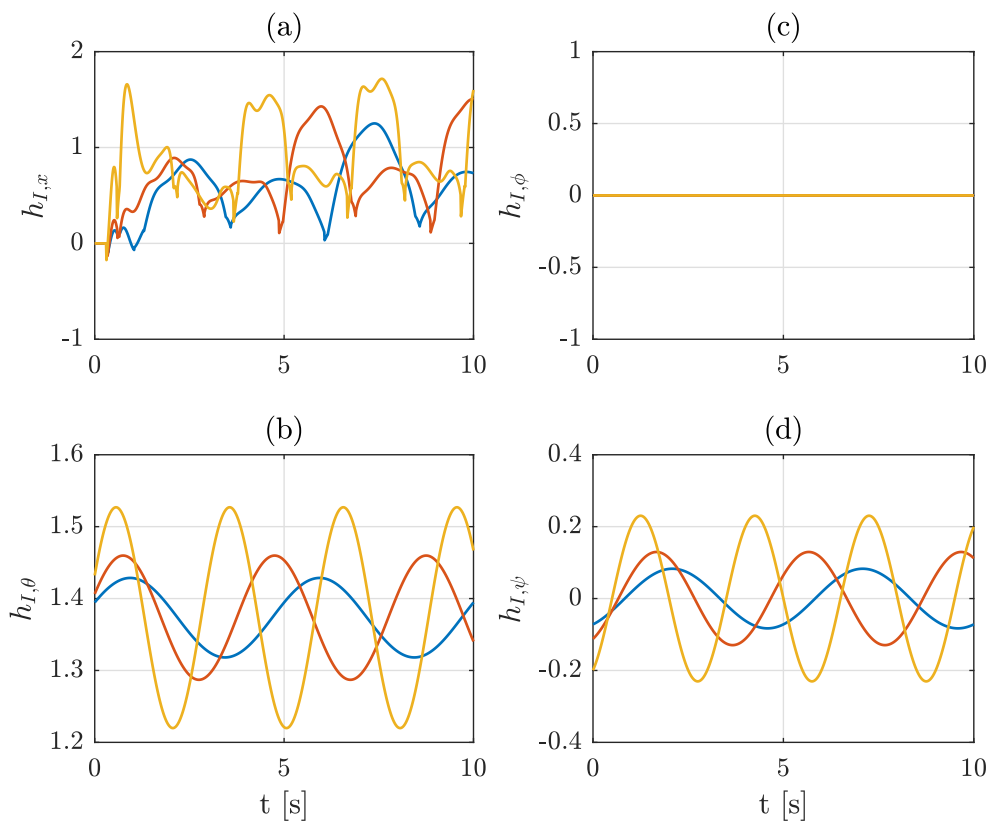


Figure 4.33: The internal force $h_{I,x}$ and the internal torques $h_{I,\phi}$, $h_{I,\theta}$, $h_{I,\psi}$. (— trajectory χ_1 , — trajectory χ_2 , — trajectory χ_3).

repeat here for simplicity the equation 3.121, which is

$$h_I = \Delta \times f_d \quad (4.19)$$

where Δ is equal

$$\Delta = \begin{bmatrix} 0.2 & 0 & 0 \end{bmatrix} \quad (4.20)$$

The task would not result in torques, only if the sharing policy and the position of the end effectors with respect to the center of mass were balanced and hence it would result in $\Delta = 0$. In another words, there is no displacement between the virtual center of mass and the object center of mass, which is not the situation here concerned. In fact we can observe torques $h_{I,\theta}$ and $h_{I,\psi}$ in the plots (b) and (d), as expected.

Considerations

Although we have covered other control allocations in this text, they are not suitable for load allocation. They are all concerned with other control objectives, such as the error minimization or the distance maintenance. For instance, in the vector sum for the DCA, it is not possible to consider during implementation the grasp matrix to calculate the wrenches in the center of mass of the object, since the force exerted by the end effector on the surface of the object is not the same felt at the center of mass of the object.

As proposed by BAIS *et al.* (2015) and SCHNEIDER and CANNON (1992), the load allocation is performed by optimization problems with equality constraints. Algorithms such as Simplex, PDIP and WLSAS are of the inequality type, and therefore, result in virtual control errors. It would mean that the load allocated to the end effectors could correspond to the total load required by the object.

4.6 Conclusions

In this chapter, experiment and simulations have been carried out for the systems differential drive robot, ROV, quadrotor and cooperative manipulators, and their results discussed and mathematically compared. Every experiment and simulation have been run for three different demanding desired trajectories for each of the control allocation algorithms proposed. The results provided meaningful to compare them with the theory addressed in the Chapter 2 and also, to analyze aspects of the algorithms in terms of the impact of their norms, their constraints, the desired control objectives and the sensibility of each of the systems as the control requirements increase.

The results have also been compared by means of the Virtual Control Error (VCE) and the Direction Error (DE) in order to provide a fair comparison of the algorithms, since both errors regard their main control objectives.

A cooperative task has been executed with two UR5 manipulators to transport an object. Three demanding desired trajectories have also been assigned and it was possible to analyze their impact to the internal forces and torques on the object and also to see the impact of the impedance-admittance relationship in the interaction between the agents and the object.

Chapter 5

Discussion and Conclusions

Four types of robots are addressed in the context of control allocation to solve the saturation issue and the load allocation in cooperative tasks. From the ground to the air, from underwater to the industrial environment, these robots cover a considerable range of environments and help to enrich the applications of control allocation present in the literature.

In the beginning, the dissertation brings the definition of saturation and demonstrates how matrix inversion or pseudoinverse cannot be utilized to deal with it, even though they belong to the unconstrained class and the pseudoinverse can be formulated as an optimization problem with equality constraints. However, this dissertation addresses systems subject to input constraints and provides tools to overcome the limitations of the unconstrained control allocation approaches. Then optimization is then briefly defined, as well the main topics concerning control allocation.

It is important to understand clearly what objectives are pursued when distributing controls. Although control allocation may provide many benefits, it is highly dependent on the redundancy of the system and also how the allocation problem is formulated. It can be set up to fulfill primary objectives such as the error minimization or the direction maintenance. Some objectives can be combined to produce a more refined formulation for the allocation problems, such as the mixed optimization problem, composed by error and control minimization, which in this dissertation are revisited by the Primal-Dual Interior Program and the Weighted Least Squares with Active Set.

Every CA algorithm here proposed presents a different formulation, which highly depends on the cost function and its norm. The function cost is defined according to its norm, which exerts influence on the projection of the cost function onto the feasible region and hence the location of the optimal solution. For instance, the DCA can be formulated as a LP problem with equality constraints for maintaining the control direction, whereas the Simplex is also formulated as a LP program with

inequality constraints for minimizing the error. Despite being defined as linear programs, their optimal solution result in different dynamics. The l_1 -norm optimization problems contain solutions at the vertices of the feasible region and tend not to converge the DoF simultaneously, whereas the DCA demonstrated not to follow this pattern, since the convergence is just like the one provided by the High-level controller, although it delivers a poorer performance when subject to saturation. QP algorithms interfere in the dynamics required by the High-level controller as well, but to a much lesser extent when compared with the LP with Simplex.

The characteristics of each algorithm could be clearly visualized in the experiment with Roomba. The DCA presented the best results, since it was able to reach the waypoints smoothly by combining harmoniously both angular and linear velocities, according to the dynamics provided by a proper tuning of the Lyapunov-based feedback controller. Nonetheless, the Simplex algorithm tends to alternate the prioritization of the linear and angular velocities and hence the travel tends to present undesired oscillations. Such irregular response must be avoided when transporting fragile or dangerous objects, for instance. Moreover, it confirms that the LP with Simplex delivers a response not according to the controller tuning, as if it were tuned with different controller gains. The WLSAS and PDIP presented dynamics not so irregular as Simplex, but close to those observed with the DCA.

The ROV LUMA is an overactuated system with respect to the DoF of interest, although it has an additional propeller for controlling depth. However, despite this redundancy, the desired trajectories required the vehicle to work near its limits, with some of its propellers at saturation levels. Therefore, the system did not have enough control authority to benefit from the mixed minimization algorithms, i.e WLSAS and PDIP, and the algorithms could not satisfy a secondary objective, since the attempt to use lower values of γ resulted in deteriorated dynamics.

On the other hand, the Simplex and DCA the only algorithms capable of allocating controls in the quadrotor, while keeping it stable. However, between both algorithms, the LP with Simplex demonstrated to be less sensitive to variations in the desired trajectories. On the other hand, the DCA remained stable only for the two less demanding desired trajectories, but it failed in the most demanding one. The dynamics of the quadrotor was simulated with the WLSAS and PDIP as well, but these algorithms were not capable of performing proper control allocation. Despite consisting of quadratic programs, the WLSAS provides more options to prioritize some control inputs or virtual controls by means of weighting matrices, whereas the Primal-dual presents a more strict structure in this sense and allows only to prioritize the error minimization to the detriment of the control minimization. Several simulations for different weighting matrices and prioritization factor γ were performed, but none could manage saturation properly and failed at delivering

suitable control inputs. As the system input is the square of the angular velocities of the rotors, this quadratic characteristic along with the gyroscopic effects pose as a challenge for CA purposes in the quadrotor. In BOUABDALLAH (2007), the Integral Backstepping has shown to provide good results for quadrotors, although this work is not concerned with dealing with saturation.

Among all the algorithms, Simplex was the unique that could allocate controls properly while keeping stable every robotic system here addressed to all the desired trajectories, although its resulting dynamics may not be the most suitable. The DCA presented excellent results with a proper convergence of the DoF, but at the cost of degenerating the dynamics of the systems at a faster pace.

In the cooperative task, the SLA provided good results for allocating loads between the two UR5 manipulators. The control scheme formed by the impedance controller at the manipulator level according to the pre-established sharing policy, the Closed Loop Inverse Kinematics and the PID Position Controller resulted in an impedance-admittance relationship between the end effectors and the load, which make the end effectors deviate from their desired trajectory in order to deliver the desired forces, as demanded by the SLA. The success of a cooperative task is not measured only by satisfying the kinematic constraints, but also by resulting in small internal forces in the object, which was achieved in the simulation, despite it increases as the trajectories become more demanding. In fact, the strategies adopted kept the internal forces into a desirable range while the kinematic constraints were satisfied.

5.1 Future works

A dissertation does not suffice to cover every aspect of control allocation and how it can be utilized to deal with input constraints. In the cooperative manipulators, for instance, it was considered the situation where the objective was purely to allocate loads while satisfying its maximum payload according to the manufacturer. Hence, the main assumption is that since the maximum payload is respected, the manipulators are able to perform the task without excessive torques at the joints level. Hence, the control allocation could be extended to the joint level to manage the impact of a demanding trajectory in a cooperative task.

As mentioned, the ROV LUMA is overactuated with respect to the motion in a plane, and therefore, two important topics of interest are the control redistribution when not subject to saturation in order to analyze how the system can benefit from redundancy and the control reconfiguration in case of failure of a propeller.

The control allocation problems here addressed have adopted only the l_1 and the l_2 -norm. Optimization problems with the l_∞ -norm has been utilized in airplanes to

provide a simultaneous convergence of the control surfaces, as described in BODSON and FROST (2011), on the opposite direction of what was observed with Simplex. Because of this feature, the l_∞ -norm is also a good topic of future research to apply it to the robots here addressed.

Another topic to be futurally studied is to analyze the influence of the High-level controller on the control allocation itself. The trajectory can be important to create control requirements at different levels, but how a controller may impact the sensibility of the system when combined to a given CA algorithm and why one should choose a control law instead of other is certainly a good topic . The High-level controllers influence directly on the optimal solution that the control allocation algorithms finds and is also responsible for setting the dynamics of the DoF, although a CA algorithm my change their directions.

Finally, the literature still lacks of stability analysis in systems when control allocation is employed. It is clear that the CA algorithms changes the dynamics imposed by the High-level controller and hence the stability analysis of the High-level controller does not suffice to describe the local or global stability of the system anymore. An evaluation of the CA algorithm and the characteristics of its optimal solution, combined with a stability analysis of the High-level controller would be important to fully understand the sensibility of the system as more demanding trajectories are imposed, how the CA algorithm interferes on the desired dynamics, its influence on the control direction and to determine the conditions in which the system should remain stable.

Bibliography

CHEN, C.-T. *Linear System Theory and Design*. Oxford University Press, 2009. ISBN: 0195392078.

SLOTINE, J.-J. E., LI, W., OTHERS. *Applied nonlinear control*, v. 199. Prentice Hall Englewood Cliffs, NJ, 1991. ISBN: 9780130408907.

KALMAN, R. E. “Analysis and design principles of second and higher order saturating servomechanisms”, *Transactions of the American Institute of Electrical Engineers, Part II: Applications and Industry*, v. 74, n. 5, pp. 294–310, nov 1955. doi: 10.1109/tai.1955.6367072.

OPPENHEIMER, M. W., DOMAN, D. B., BOLENDER, M. A. “Control Allocation”. In: *The Control Handbook - Control System Applications*, 2 ed., CRC Press, chap. 8 - Control Allocation, 2010.

DURHAM, W. C. “Constrained control allocation”, *Journal of Guidance, Control, and Dynamics*, v. 16, n. 4, pp. 717–725, 1993.

DEFENCE, L. “Damning New Light On What Caused 2 HAL Dhruv Crashes, Nasty IAF-HAL Spat Detailed”. aug. 2010. Available at: <https://www.livefistdefence.com/2010/08/damning-new-light-on-thing-that-caused.html>. Access on: 03 sep. 2019, 22:35:20.

GARCIA, R., CASTELO, F. “Conditional integration on PID for types 1 and 2 control systems”. In: *Proceedings of the International Conference on Control Applications*. IEEE, 2002. doi: 10.1109/cca.2002.1040249.

HIPPE, P. “Windup in Control: Its Effects and Their Prevention”. In: *Advances in Industrial Control*, Springer-Verlag, pp. 1–20, London, 2006. doi: 10.1007/1-84628-323-x.

CHEN, H. “Robust stabilization for a class of dynamic feedback uncertain non-holonomic mobile robots with input saturation”, *International Journal of*

- Control, Automation and Systems*, v. 12, n. 6, pp. 1216–1224, oct 2014. doi: 10.1007/s12555-013-0492-z.
- HUANG, H., ZHOU, J., DI, Q., ZHOU, J., LI, J. “Robust neural network–based tracking control and stabilization of a wheeled mobile robot with input saturation”, *International Journal of Robust and Nonlinear Control*, v. 29, n. 2, pp. 375–392, oct 2018. doi: 10.1002/rnc.4396.
- HUANG, J., WEN, C., WANG, W., JIANG, Z.-P. “Adaptive stabilization and tracking control of a nonholonomic mobile robot with input saturation and disturbance”, *Systems & Control Letters*, v. 62, n. 3, pp. 234–241, mar 2013. doi: 10.1016/j.sysconle.2012.11.020.
- BERNSTEIN, D. S., MICHEL, A. N. “A chronological bibliography on saturating actuators”, *International Journal of Robust and Nonlinear Control*, v. 5, n. 5, pp. 375–380, 1995. doi: 10.1002/rnc.4590050502.
- JOHANSEN, T., FOSSEN, T. “Control allocation— survey”, *Automatica*, v. 49, n. 5, pp. 1087–1103, 2013.
- BUFFINGTON, J., BUFFINGTON, J. “Tailless aircraft control allocation”. In: *Guidance, Navigation, and Control Conference*. American Institute of Aeronautics and Astronautics, aug 1997. doi: 10.2514/6.1997-3605.
- JOHANSEN, T. “Optimizing nonlinear control allocation”. In: *2004 43rd IEEE Conference on Decision and Control (CDC) (IEEE Cat. No.04CH37601)*. IEEE, 2004. doi: 10.1109/cdc.2004.1429240.
- CHEN, Y., WANG, J. “Energy-efficient control allocation with applications on planar motion control of electric ground vehicles”. In: *Proceedings of the 2011 American Control Conference*. IEEE, jun 2011. doi: 10.1109/acc.2011.5991160.
- MONTEIRO, J. C. E. *Modelagem e Controle de um Veículo Quadricóptero*. mathesis, Federal University of Rio de Janeiro, Rio de Janeiro:UFRJ/POLI, mar. 2015.
- BOWERS, A. H., PAHLE, J. W., WILSON, R. J., FLICK, B. C., ROOD, L. R. “An Overview of the NASA F-18 High Alpha Research Vehicle”. 1996. Available at: <<https://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/19970003009.pdf>>. Access on: 04 sep. 2019, 00:15.
- MORELLI, E. A. “F-18 High Alpha Research Vehicle (HARV) Parameter Identification Flight Test Maneuvers for Optimal Input Design

Validation and Lateral Control Effectiveness”. dec. 1995. Available at: <<https://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/19960012190.pdf>>. Access on: 03 sep. 2019, 22:38:15.

BODSON, M., FROST, S. A. “Load Balancing in Control Allocation”, *Journal of Guidance, Control, and Dynamics*, v. 34, n. 2, pp. 380–387, mar 2011. doi: 10.2514/1.51952.

DURHAM, W. C., BORDIGNON, K. A. “Multiple Control Effector Rate Limiting”, *Journal of Guidance, Control, and Dynamics*, v. 19, n. 1, pp. 30–37, 1996.

PETERSEN, J. A. M., BODSON, M. “Interior-Point Algorithms for Control Allocation”, *Journal of Guidance, Control, and Dynamics*, v. 28, n. 3, pp. 471–480, may 2005. doi: 10.2514/1.5937.

BODSON, M. “Evaluation of Optimization Methods for Control Allocation”, *Journal of Guidance, Control, and Dynamics*, v. 25, n. 4, pp. 703–711, jul 2002. ISSN: 0731-5090.

LAWITZKY, M., MORTL, A., HIRCHE, S. “Load sharing in human-robot cooperative manipulation”. In: *19th International Symposium in Robot and Human Interactive Communication*. IEEE, sep 2010. doi: 10.1109/roman.2010.5598627.

BAIS, A. Z., ERHART, S., ZACCARIAN, L., HIRCHE, S. “Dynamic load distribution in cooperative manipulation tasks”. In: *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, sep 2015. doi: 10.1109/iros.2015.7353699.

MONTEIRO, J. C., LIZARRALDE, F., HSU, L. “Optimal control allocation of quadrotor UAVs subject to actuator constraints”. In: *American Control Conference (ACC), 2016*, pp. 500–505. IEEE, 2016a.

MONTEIRO, J. C., LIZARRALDE, F., HSU, L. “Control Allocation: Enhancing the Performance of Quadrotors High-level Controllers”. In: *XXI Congresso Brasileiro de Automática - CBA2016*, pp. 1071–1076, 2016b.

LUENBERGER, D. G., YE, Y. *Linear and Nonlinear Programming (International Series in Operations Research & Management Science Book 116)*. Springer, 2008. ISBN: 978-0-387-74503-9.

NOCEDAL, J., WRIGHT, S. J. *Numerical Optimization*. Springer, 1999.

- OPPENHEIMER, M. W., DOMAN, D. B., BOLENDER, M. A. “Control Allocation for Over-actuated Systems”. In: *2006 14th Mediterranean Conference on Control and Automation*. IEEE, jun 2006. doi: 10.1109/med.2006.328750.
- WRIGHT, S. J. *Primal-Dual Interior-Point Methods*. Society for Industrial and Applied Mathematics, 1987. ISBN: 978-0898713824.
- HÄRKEGÅRD, O. “Efficient active set algorithms for solving constrained least squares problems in aircraft control allocation”. In: *Proceedings of the 41st IEEE Conference on Decision and Control, 2002.*, v. 2, pp. 1295–1300 vol.2, Dec 2002. doi: 10.1109/CDC.2002.1184694.
- KLEE, V., MINTY, G. J. “How Good Is The Simplex Algorithm?” In: Press, A. (Ed.), *Inequalities, O. Shisha*, pp. 159–175, New York, 1972.
- PETERSEN, J., BODSON, M. “Constrained Quadratic Programming Techniques for Control Allocation”, *IEEE Transactions on Control Systems Technology*, v. 14, n. 1, pp. 91–98, jan. 2005. ISSN: 1063-6536. doi: 10.1109/TCST.2005.860516.
- LAGES, W. F. *Controle e Estimação de Posição e Orientação de Robôs Móveis*. phdthesis, Instituto Tecnológico de Aeronáutica, 1998.
- SICILIANO, B., SCIAVICCO, L., VILLANI, L., ORIOLO, G. *Robotics: modelling, planning and control*. Springer-Verlag London Limited, 2009. ISBN: 978-1-84628-642-1.
- SIEGWART, R., NOURBAKHSI, I. R. *Introduction to Autonomous Mobile Robots*. Scituate, MA, USA, Bradford Company, 2004. ISBN: 026219502X.
- BROCKETT, R. W. “Asymptotic stability and feedback stabilization”. In: *Differential Geometric Control Theory*, pp. 181–191. Birkhauser, 1983.
- AICARDI, M., CASALINO, G., BICCHI, A., BALESTRINO, A. “Closed loop steering of unicycle like vehicles via Lyapunov techniques”, *IEEE Robotics Automation Magazine*, v. 2, n. 1, pp. 27–35, mar. 1995. ISSN: 1070-9932. doi: 10.1109/100.388294.
- SHUKLA, A., KARKI, H. “Application of robotics in offshore oil and gas industry— A review Part II”, *Robotics and Autonomous Systems*, v. 75, pp. 508–524, jan 2016. doi: 10.1016/j.robot.2015.09.013.

- HSU, L., COSTA, R., LIZARRALDE, F., SOARES DA CUNHA, V. “Avaliação Experimental da Modelagem e Simulação da Dinâmica de um Veículo Submarino de Operação Remota”, *Revista Controle & Automação*, v. 11, pp. 82–93, 05 2000.
- GOULART, C. *Modelagem, Simulação e Controle de um Veículo Submarino e Operação Remota*. phdthesis, COPPE/UFRJ, 2007.
- OGATA, K. *Modern Control Engineering: International Version*. Prentice Hall, 2008. ISBN: 0137133375.
- BOUABDALLAH, S. *Design and Control of Quadrotors with Application to Autonomous Flying*. phdthesis, École Polytechnique Fédérale de Lausanne, Lausanne, feb. 2007. Available at: <https://infoscience.epfl.ch/record/95939/files/EPFL_TH3727.pdf>. Access on: 03 sep. 2019, 22:40:10.
- CRAIG, J. J. *Introduction to Robotics: Mechanics and Control (3rd Edition)*. Pearson, 2004. ISBN: 978-0201543612.
- MURRAY, R. M., SASTRY, S. S., ZEXIANG, L. *A Mathematical Introduction to Robotic Manipulation*. 1st ed. Boca Raton, FL, USA, CRC Press, Inc., 1994. ISBN: 0849379814.
- EPPINGER, S., SEERING, W. “Introduction to dynamic models for robot force control”, *IEEE Control Systems Magazine*, v. 7, n. 2, pp. 48–52, apr 1987. doi: 10.1109/mcs.1987.1105274.
- REN, Y., LIU, Y., JIN, M., LIU, H. “Biomimetic object impedance control for dual-arm cooperative 7-DOF manipulators”, *Robotics and Autonomous Systems*, v. 75, pp. 273–287, jan 2016. doi: 10.1016/j.robot.2015.09.018.
- CACCAVALE, F., CHIACCHIO, P., MARINO, A., VILLANI, L. “Six-DOF Impedance Control of Dual-Arm Cooperative Manipulators”, *IEEE/ASME Transactions on Mechatronics*, v. 13, n. 5, pp. 576–586, oct 2008. doi: 10.1109/tmech.2008.2002816.
- HOGAN, N. “Impedance Control: An Approach to Manipulation”. In: *1984 American Control Conference*. IEEE, jul 1984. doi: 10.23919/acc.1984.4788393.
- SCHNEIDER, S., CANNON, R. “Object impedance control for cooperative manipulation: theory and experimental results”, *IEEE Transactions on Robotics and Automation*, v. 8, n. 3, pp. 383–394, jun 1992. doi: 10.1109/70.143355.

- CACCAVALE, F., CHIAVERINI, S., SICILIANO, B. “Second-order kinematic control of robot manipulators with Jacobian damped least-squares inverse: theory and experiments”, *IEEE/ASME Transactions on Mechatronics*, v. 2, n. 3, pp. 188–194, 1997. doi: 10.1109/3516.622971.
- ESPOSITO, J. M. “Matlab Toolbox for iRobot Create 2”. www.usna.edu/Users/weapsys/esposito/roomba.matlab/, 2015. Access on: 03 sep. 2019, 22:41:16.
- DE ANDRADE, M. R. *Simulador para o ROV LUMA utilizando Gazebo*. Master’s Thesis, Escola Politécnica, UFRJ, 2017.
- KUFIETA, K. *Force Estimation in Robotic Manipulators: Modeling, Simulation and Experiments*. mathesis, NTNU Norwegian University of Science and Technology, Department of Engineering Cybernetics, jan. 2014.
- ALDERETE, T. S. “Simulator Aero Model Implementation”. 1997. Available at: <https://www.aviationsystemsdivision.arc.nasa.gov/publications/hitl/rtsim/Toms.pdf>. Access on: 03 sep. 2019, 23:30:20.
- MURPHY, R. R. “Trial by fire [rescue robots]”, *IEEE Robotics & Automation Magazine*, v. 11, n. 3, pp. 50–61, 2004.
- HALLIDAY, D., RESNICK, R., WALKER, J. *Fundamentals of Physics*. Wiley, 2013. ISBN: 9781118230718.
- HAY, W. W. “The Coriolis Effect”. In: *Experimenting on a Small Planet*, Springer International Publishing, pp. 524–555, 2016. doi: 10.1007/978-3-319-27404-1_23.

Appendix A

Mechanics of Rigid Body

The fundamentals of Robotics were built over physical fundamentals concerning rigid bodies, and from there, the knowledge was expanded to develop every class of robot nowadays, such as manipulators, airplanes, mobile robots and so on.

One of these fundamentals is the kinematics, which is a branch of mechanical physics concerned with describing the motion of points, rigid bodies and sets of bodies. Intrinsically bound to motion it can be mentioned position, orientation and velocity. However, these characteristics are superficial, as it neglects inherent characteristics of bodies, such as its geometry, the material the body is constituted, its mass, friction, and other parameters related, which are covered by the dynamics.

In order to analyze completely the behavior of rigid bodies, these branches of mechanical physics are addressed here.

A.1 Rigid Body Kinematics

Consider a particle moving in the 3D Euclidean space, whose location changes at every time t relative to a reference frame \mathcal{F}_a and is composed by three orthonormal axis, which are $(X_a, Y_a, Z_a) \in \mathbb{R}^3$.

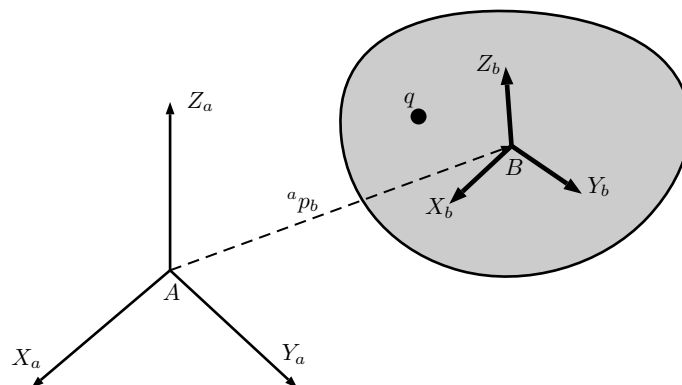


Figure A.1: Representation of a rigid body with body frame \mathcal{F}_b located at ${}^a p_b$.

Nevertheless, Robotics deals with a set of particles called *rigid body*. A rigid body is said to be any object in a continuous movement of particles, whose distance among them remains fixed at every time instant t and cannot be deformed even though it moves or any force is exerted on it. The net movement of a rigid body from a location to another via a rigid motion is called a rigid displacement (MURRAY *et al.*, 1994).

In figure A.1, consider the point p and $q \in \mathbb{R}^3$ located inside the rigid body. A mapping $g : \mathbb{R}^3 \rightarrow \mathbb{R}^3$ is called a *rigid body transformation* (MURRAY *et al.*, 1994) if the following properties can be observed in the rigid body for both points:

- $\|g(p) - g(q)\| = \|p - q\|$;
- $g(v \times w) = gv \times gw, \forall$ vector v and $w \in \mathbb{R}^3$.

In fact, these properties grant that the body is rigid, and as consequence, it can be fully tracked as long as any of its point is tracked, provided that the rigid body has a Cartesian frame attached and there exists also a fixed relative frame to be related to.

A.1.1 Position

We still consider the point p in the rigid body, in which is located the frame \mathcal{F}_b , composed by the coordinates (X_b, Y_b, Z_b) . The position of p with respect to \mathcal{F}_a is given by

$${}^a p_b = \begin{bmatrix} {}^a x_b \\ {}^a y_b \\ {}^a z_b \end{bmatrix}, \quad {}^a p_b \in \mathcal{R}^3 \quad (\text{A.1})$$

In fact, since the body frame and the inertial frame are defined and related, the trajectory can be tracked at any time instant.

A.1.2 Orientation and Rotation Matrix

Now the translational motion is neglected and one considers that ${}^a p_b = 0$, namely \mathcal{F}_a and \mathcal{F}_b coincide initially. The frame \mathcal{F}_b is then rotated and the body frame \mathcal{F}_b can be related to \mathcal{F}_a through the following equations, as defined in SICILIANO *et al.* (2009), which yields

$$\begin{bmatrix} {}^a X_b & {}^a Y_b & {}^a Z_b \end{bmatrix} = \begin{bmatrix} x_b \cdot x_a & y_b \cdot x_a & z_b \cdot x_a \\ x_b \cdot y_a & y_b \cdot y_a & z_b \cdot y_a \\ x_b \cdot z_a & y_b \cdot z_a & z_b \cdot z_a \end{bmatrix} \quad (\text{A.2})$$

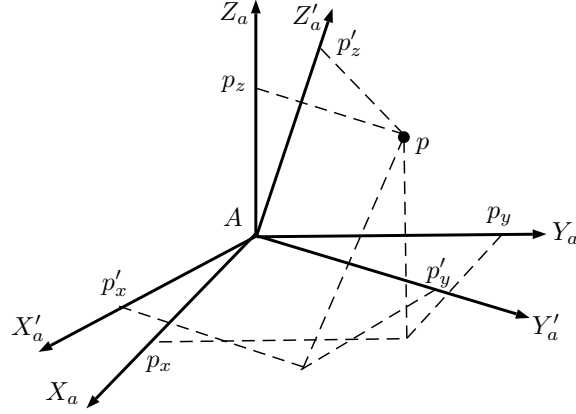


Figure A.2: Rotation of body frame \mathcal{F}_b with respect to \mathcal{F}_a .

where each inner product is a direction cosine. Hence, every related motion between frames can be represented by a *rotation matrix* in the form

$${}^a R_b = \begin{bmatrix} {}^a X_b & {}^a Y_b & {}^a Z_b \end{bmatrix} \in \mathbb{R}^{3 \times 3} \quad (\text{A.3})$$

As the columns of the rotation matrix are orthonormal and the axis of both coordinate frames \mathcal{F}_a and \mathcal{F}_b are derived from the right hand rule, the following properties hold

$$\begin{aligned} {}^a R_b^T &= {}^a R_b^{-1} \\ \det({}^a R_b) &= 1 \end{aligned} \quad (\text{A.4})$$

In fact, rotation matrices belong to the special orthonormal group $SO(3)$, because it is necessary to use a set of three angles for a minimal representation (SICILIANO *et al.*, 2009). Orthonormal groups can be generalized for n dimensions, such as

$$SO(n) = \{R \in \mathbb{R} \mid RR^T = \det(R) = +1\} \quad (\text{A.5})$$

When a point p undergoes a rotation, as depicted in A.2, its position in the coordinate frame \mathcal{F}_b with respect to \mathcal{F}_a is given by

$$p_a = {}^a R_b p_b \quad (\text{A.6})$$

A sequence of rotations from a given frame \mathcal{F}_a to \mathcal{F}_c can be concatenated by multiplying the rotation matrices from \mathcal{F}_a to \mathcal{F}_b and from \mathcal{F}_b to \mathcal{F}_c . It can be algebraically declared as

$${}^a R_c = {}^a R_b {}^b R_c \quad (\text{A.7})$$

The same approach can be applied to any other desired coordinate frame.

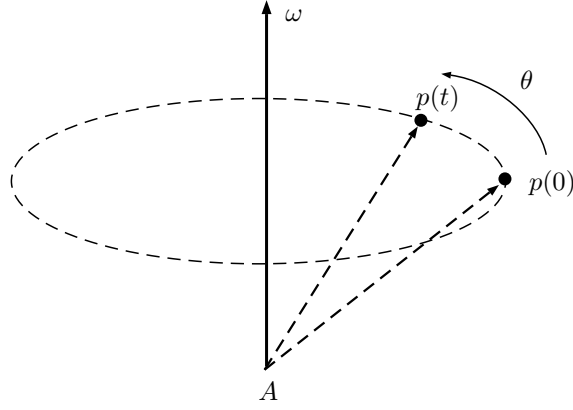


Figure A.3: A point p rotating about the axis ω .

A.1.3 Rotation in Exponential Coordinates

A common motion found in Robotics is the rotation of a rigid body about an axis. Consider a point p attached to a body rotating at a constant unit velocity θ around the axis $\omega \in \mathbb{R}^3$. The instant velocity \dot{p} can be expressed as

$$\dot{p} = \omega \times p(t) \quad (\text{A.8})$$

that after integrating, results in

$$p(t) = e^{\hat{\omega}\theta} p(0) \quad (\text{A.9})$$

It is equivalent to state that this point was rotated about the axis ω at unit velocity for θ units of time (MURRAY *et al.*, 1994). This rotation can be expressed as

$$R_\omega(\theta) = e^{\hat{\omega}\theta} \quad (\text{A.10})$$

and $\hat{\omega}$ is a skew-symmetric matrix, which stands for

$$\hat{\omega} = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix} \quad (\text{A.11})$$

The exponential at the right side of the equation A.10 can be efficiently computed by the Rodrigues' formula when $\|k\| \neq 1$, which is

$$e^{\hat{\omega}\theta} = I + \hat{\omega} \sin \theta + \hat{\omega}^2 (1 - \cos \theta) \quad (\text{A.12})$$

The inverse problem can be solved by computing $\hat{\omega}$ and θ from the equation

$$\begin{aligned}\theta &= \arccos\left(\frac{\text{tr}(R) - 1}{2}\right) \\ \hat{\omega} &= \frac{1}{2\sin\theta}(R - R^T)\end{aligned}\tag{A.13}$$

A.1.4 Euler Angles

Rotation matrices are characterized by nine elements which are not independent but related by six constraints due to the orthogonality constraints (SICILIANO *et al.*, 2009). As a consequence, three parameters are sufficient to describe completely the orientation of a rigid body and there are here two worth mentioning set of Euler angles $\phi \in \mathbb{R}^3$ among all possible, which are the ZYZ and the XYZ (also known as Roll-Pitch-Yaw or Tait-Bryan angles). These nomenclatures correspond to the sequence of basic rotations about the axis ZYZ or XYZ, in this sequence, respectively.

The basic rotations are:

$$R_X(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{bmatrix}\tag{A.14}$$

$$R_Y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix}\tag{A.15}$$

$$R_Z(\psi) = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}\tag{A.16}$$

A.1.5 Angular Velocity Transformation

A rigid body that undergoes a rotation has its orientation written in terms of roll, pitch and yaw angles with respect to a fixed reference frame \mathcal{F}_a . On the other hand, the angular velocity ω is the angular displacement rate of this rigid body about its body frame \mathcal{F}_b placed at its center of inertia and can be directly measured by devices such as gyroscopes.

Thus, the Euler angles ϕ , θ and ψ can be related to ω by means of a transformation matrix (ALDERETE, 1997) in the form

$$\begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} = \begin{bmatrix} 1 & 0 & -\sin \theta \\ 0 & \cos \phi & \sin \phi \cos \theta \\ 0 & -\sin \phi & \cos \phi \cos \theta \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \quad (\text{A.17})$$

$$\omega = R_r(\eta)\dot{\eta}$$

A.1.6 Quaternions

In 3D rotations, quaternions are preferred over the Euler angles, because they give a global parametrization of $\text{SO}(3)$, at the cost of using four numbers, rather than three (MURPHY, 2004). This results in less mathematical operations and no singularities. Besides it, it eases the interpolation between two different quaternions, specially useful in cooperative tasks for calculating relative (CACCAVALE *et al.*, 2008).

Quaternions are considered a extension of complex numbers and are defined as $\mathcal{Q} = a + bi + cj + dk$, where i, j and $k \in \mathbb{C}$ are unit imaginary numbers and $q_i \in \mathbb{R}$, with $i = 0, 1, \dots, 3$. It is composed by a scalar component $\mathcal{Q}_s = a$ and a vectorial one $\mathcal{Q}_v = \begin{bmatrix} b & c & d \end{bmatrix}^T$. These components are be obtained by calculating

$$\begin{bmatrix} \mathcal{Q}_s \\ \mathcal{Q}_v \end{bmatrix} = \begin{bmatrix} \cos \frac{\theta}{2} \\ \omega \sin \frac{\theta}{2} \end{bmatrix} \quad (\text{A.18})$$

Given two quaternions \mathcal{Q}_1 and \mathcal{Q}_2 , their multiplication is defined as

$$\mathcal{Q}_1 \mathcal{Q}_2 = \begin{bmatrix} \mathcal{Q}_{1s} \mathcal{Q}_{2s} - \mathcal{Q}_{1v}^T \mathcal{Q}_{2v} \\ \mathcal{Q}_{1s} \mathcal{Q}_{2v} + \mathcal{Q}_{2s} \mathcal{Q}_{1v} + \hat{\mathcal{Q}}_{1v} \mathcal{Q}_{2v} \end{bmatrix} \quad (\text{A.19})$$

Like vectors, quaternions can also be conjugated. The conjugate of a quaternion \mathcal{Q} is $\mathcal{Q}^* = \begin{bmatrix} \mathcal{Q}_s & -\mathcal{Q}_v \end{bmatrix}^T$, and their product is given by

$$\mathcal{Q} \mathcal{Q}^* = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad (\text{A.20})$$

In quaternions, a sequence of rotations can also be concatenated, just like the rotation matrices. A rotation from a frame \mathcal{F}_a to \mathcal{F}_b and from \mathcal{F}_b to \mathcal{F}_c can be resumed as

$${}^a \mathcal{Q}_c = {}^a \mathcal{Q}_b {}^b \mathcal{Q}_c \quad (\text{A.21})$$

A quaternion can also be converted into a rotation matrix by proceeding to the calculating of

$$R = (2\mathcal{Q}_s^2 - 1)\mathcal{I} + 2(\mathcal{Q}_v \mathcal{Q}_v^T + \mathcal{Q}_s \widehat{\mathcal{Q}}_v) \quad (\text{A.22})$$

Unit Quaternion

Unit Quaternion, also known as versor and denoted here as \mathcal{Q}_U , is a subset of all $\mathcal{Q} \in \mathbb{Q}$ such that $\|\mathcal{Q}\| = 1$ (MURRAY *et al.*, 1994). Given a quaternion \mathcal{Q} , \mathcal{Q}_U is obtained by proceeding to the calculation of

$$\mathcal{Q}_U = \frac{\mathcal{Q}}{\|\mathcal{Q}\|} \quad (\text{A.23})$$

which consists of a quaternion normalization. Such representatio is preferred over general quaternions when dealing with orientation control, provided that the calculations lie on the same basis.

For controlling orientation, quaternions are often preferred, and thus, it is necessary to calculate the orientation error. Consider two quaternions \mathcal{Q}_U and \mathcal{Q}_{Ud} , associated to the rotation matrices R and R_d , respectively. The quaternion \mathcal{Q}_U is the representation of the actual rigid body unit quaternion orientation and \mathcal{Q}_{Ud} is the desired orientation. The orientation error is given by $R_d R^T$, whose quaternion representation is

$$\begin{bmatrix} Q_{os} \\ Q_{ov} \end{bmatrix} = \mathcal{Q}_U \mathcal{Q}_{Ud}^* \quad (\text{A.24})$$

When \mathcal{Q}_{Ud} and \mathcal{Q}_U are aligned, the error becomes

$$e_o = \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix} \quad (\text{A.25})$$

Thus, it suffices to define the orientation error as \mathcal{Q}_{ov} , which stands for

$$e_o = \mathcal{Q}_{1s} \mathcal{Q}_{2v} - \mathcal{Q}_{2s} \mathcal{Q}_{1v} - \hat{\mathcal{Q}}_{1v} \mathcal{Q}_{2v} \quad (\text{A.26})$$

A.1.7 Homogeneous Transformation Matrix

Still considering the figure A.1 and with the definitions presented, it is possible to bind position and orientation for analyzing completely the aspects concerning the motion of a rigid body. Given that ${}^a p_b \in \mathbb{R}^3$ and ${}^a R_b \in SO(3)$, the full rigid body configuration space is therefore defined as

$$SE(3) = \{({}^a p_b, {}^a R_b) | {}^a p_b \in \mathbb{R}^3, {}^a R_b \in SO(3)\} = \mathbb{R}^3 \times SO(3) \quad (\text{A.27})$$

and is called 3-dimensional special euclidean group.

The position of any point p_a belonging to the rigid body with respect to \mathcal{F}_a can be expressed as

$$p_a = {}^a p_b + {}^a R_b p_b \quad (\text{A.28})$$

This relation represents the coordinate transformation, i.e rotation and translation combined, of a bound vector between two frames (SICILIANO *et al.*, 2009). It can be rewritten in the matricial form as

$$\begin{bmatrix} p_a \\ 1 \end{bmatrix} = \begin{bmatrix} {}^a R_b & {}^a p_b \\ 0 & 1 \end{bmatrix} \begin{bmatrix} p_b \\ 1 \end{bmatrix} \quad (\text{A.29})$$

and thus, the *homogeneous transformation matrix* is defined as

$${}^a T_b = \begin{bmatrix} {}^a R_b & {}^a p_b \\ 0 & 1 \end{bmatrix}, \quad {}^a T_b \in SE(3) \quad (\text{A.30})$$

where $T_{ab} \in SE(3)$. Like rotation matrices, it is also possible to concatenate sequential homogeneous transformations for relating position and orientation in chain, in different coordinate frames, in the form

$${}^a T_c = {}^a T_b {}^b T_c \quad (\text{A.31})$$

A.1.8 Velocities of a Rigid Body

Prior to defining linear velocity, consider the following property of a rotation matrix of the body frame \mathcal{F}_b with respect to a given frame \mathcal{F}_a at any instant t and calling ${}^a R_b = R$ for simplicity:

$$RR^T = \mathcal{I} \quad (\text{A.32})$$

By applying its time derivative by means of the chain rule, it yields

$$\dot{R}R^T + R\dot{R}^T = 0 \quad (\text{A.33})$$

Then, $S = \dot{R}R^T$ is set and hence S is skew-symmetric. Both sides of equation A.33 are left-multiplied by R , which gives

$$\begin{aligned} \dot{R} &= SR \\ &= \omega \times R \end{aligned} \quad (\text{A.34})$$

and relates the angular velocity ω and the derivative of the rotation matrix. Now,

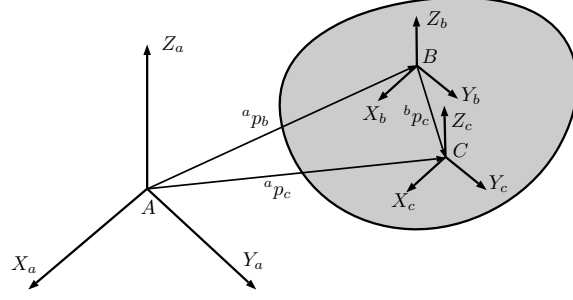


Figure A.4: Two frames \mathcal{F}_b and \mathcal{F}_c in a rigid frame with respect to \mathcal{F}_a .

considering full motion, from the derivative of (A.28), the velocity of a rigid body with respect to \mathcal{F}_a is given by

$$\begin{aligned}\dot{p}_a &= {}^a\dot{p}_b + \dot{R}p_b + Rp_b \\ &= {}^a\dot{p}_b + \omega \times Rp_b + Rp_b\end{aligned}\tag{A.35}$$

where ω denotes the angular velocity of frame R with respect to \mathcal{F}_a . However, provided that p_b is constant, the equation can be simplified to

$$\dot{p}_a = {}^a\dot{p}_b + \omega \times Rp_b\tag{A.36}$$

A.1.9 Quaternion Propagation

When quaternions are elected to represent the a rigid body orientation, it must be observed that sensors usually deliver the angular velocity, rather than the time derivative of the orientation error. Then, it must be considered a relationship between the angular velocity and quaternions (SICILIANO *et al.*, 2009), given by

$$\begin{cases} \dot{Q}_s = -\frac{1}{2}Q_v^T\omega \\ \dot{Q}_v = \frac{1}{2}(Q_s\mathcal{I} - \hat{Q}_v)\omega \end{cases}\tag{A.37}$$

This relationship is also known as quaternion propagation.

A.1.10 Generalized Velocities Transformation

Now, we consider that two different frames \mathcal{F}_b and \mathcal{F}_c are located in the same rigid body, as shown in A.4. Therefore, the following equality holds:

$${}^a\omega_b = {}^a\omega_c\tag{A.38}$$

With respect to position, both frames can be related as

$${}^a p_c = {}^a p_b + {}^b p_c \quad (\text{A.39})$$

Taking its derivative, it results in

$${}^a \dot{p}_c = {}^a \dot{p}_b + \omega_b \times {}^c \dot{p}_b \quad (\text{A.40})$$

By omitting the fixed reference frame for simplicity, the definitions above can be put together in the matricial form as

$$h_c = \begin{bmatrix} \dot{p}_c \\ \omega_c \end{bmatrix} = \begin{bmatrix} \mathcal{I} & -{}^c p_b \times \\ 0 & \mathcal{I} \end{bmatrix} \begin{bmatrix} \dot{p}_b \\ \omega_b \end{bmatrix} = {}^c G_b^T h_b \quad (\text{A.41})$$

where ${}^c G_b$ is called adjoint transformation matrix (MURRAY *et al.*, 1994).

This relation is important for propagating generalized velocities along different coordinate frames. For instance, each joint of a manipulator produces a velocity, which contributes to the end effector velocity. Therefore, the joint velocity must be propagated to the task frame (SICILIANO *et al.*, 2009).

A.2 Rigid Body Dynamics

The dynamics of a rigid body are useful to study their motion under the action of external forces. When only its kinematics is observed, many aspects which influences motion are neglected, such as friction, material stiffness, inertia, etc. Hence, the kinematic model may not be enough to describe a body motion properly.

There are two important formalisms to derive suitably the dynamics of a system, which are:

- Lagrange
- Newton-Euler

However, prior to defining them, it is important to present some important concepts concerning dynamics.

A.2.1 Center of Mass

Consider a rigid body which occupies a volume $V \in \mathbb{R}^3$ and with mass distribution $r \in V$. The rigid body is made up of any material with mass density $\rho(r)$ (MURRAY *et al.*, 1994). Its mass is given by

$$M = \int_V \rho(r) dV \quad (\text{A.42})$$

and the center of mass \bar{r} is a hypothetical point where all the mass of a body is concentrated and it behaves as if all the efforts were applied to this point. It depends on the geometry and its mass distribution, which is related to mass density of the material that the rigid body is constituted. Hence, \bar{r} is calculated by

$$\bar{r} = \frac{1}{M} \int_V \rho(r) r dV \quad (\text{A.43})$$

When a rigid body is composed by the same material, the center of mass will correspond to its centroid.

A.2.2 Kinetic Energy

Kinetic energy is the energy associated to the motion state once a rigid body is in motion (HALLIDAY *et al.*, 2013). The kinetic energy is also defined as the work necessary to put a rigid motion into motion from the rest state. If friction is neglected and provided that the rigid body is in motion, its acquired acceleration is kept unless a force is exerted against its motion. The same work is necessary to bring the body from motion to the state of rest.

Consider a force F exerted on a rigid body with mass M , which is moved by a distance p_a . From the Second Newton Law, the force applied is

$$F(p_a) = M\ddot{p}_a \quad (\text{A.44})$$

Provided that work due to the kinetic energy is the integral of force and with the definitions from the equation A.42, it results in the volume integral given by

$$\mathcal{T} = \frac{1}{2} \int_V \rho(r) \|\dot{p}_a\|^2 dV \quad (\text{A.45})$$

A.2.3 Potential Energy

The potential energy of a conservative field \mathcal{U} is the energy associated to a rigid body due to its position with respect to other bodies. This energy is stored according to its situation in a force field or to the system configuration in which the rigid body is inserted. It does not depend on the trajectory, but rather on the initial and final state.

Concerning the gravitational field, the gravitational potential energy can be calculated by

$$\mathcal{U} = g \int_V \rho(r)^a z_p dV \quad (\text{A.46})$$

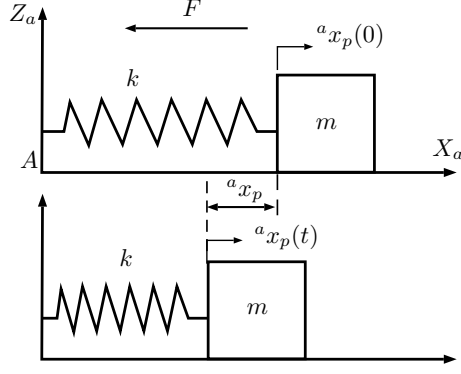


Figure A.5: System mass-spring under a force F , which provokes a deformation Δx in the spring and energy storage.

where ${}^a z_p$ is the height of a particle at position p with respect to frame \mathcal{F}_a and that occupies a volume dV .

Another important type of potential energy is the elastic potential energy, which is the energy stored in a system due to a displacement ${}^a x_p$ caused by an elastic force F in a spring with elastic constant $k > 0$, as depicted in the figure A.5. The elastic force is defined by the Hooke's Law, which is

$$F(p_a) = -k {}^a x_p \quad (\text{A.47})$$

When integrated, it results in the elastic potential energy given by

$$\mathcal{U}_e = \frac{1}{2} k {}^a x_p^2 \quad (\text{A.48})$$

The elastic potential energy does not apply in the definitions to come when modeling the dynamics of a rigid body. However, the elastic force will be important when dealing with manipulators, provided that contact forces can be modeled by a mass-spring system (REN *et al.*, 2016).

A.2.4 Inertia Matrix

The Inertia Matrix of a rigid body is close related to its motion. Its velocity can be obtained by calculating the time derivative of equation A.28, which yields

$$\dot{p}_a = {}^a \dot{p}_b + {}^a \dot{R}_b r \quad (\text{A.49})$$

Consider a body volume V with a mass distribution $\rho(r)$. Its kinetic energy corresponds to the energy it possesses due to motion and is given by

$$\mathcal{T} = \frac{1}{2} \int_V \rho(r) \left\| {}^a \dot{p}_b + {}^a \dot{R}_b r \right\|^2 dV \quad (\text{A.50})$$

By evaluating the equation A.50, it can be split into translational and rotational terms. Concerning the latter, the derivative of the rotation matrix is related to the angular velocity by $\dot{R}_{ab} = R_{ab}\hat{\omega}$, where ω is the body angular velocity. The kinetic energy due to rotation is given by

$$\mathcal{T}_\omega = \frac{1}{2}\omega^T I \omega \quad (\text{A.51})$$

where

$$I = \begin{bmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{yx} & I_{yy} & I_{yz} \\ I_{zx} & I_{zy} & I_{zz} \end{bmatrix} = - \int_V \rho(r) \hat{r}^2 dV \quad (\text{A.52})$$

where I is the *inertia tensor* expressed in the body frame (MURRAY *et al.*, 1994) and the matrix is also positive definite and symmetric. On the other hand, the translational term of the kinetic energy is

$$\mathcal{T}_p = \frac{1}{2}M \|\dot{p}_b\|^2 \quad (\text{A.53})$$

By adding both terms, \mathcal{T}_ω and \mathcal{T}_p , and after some algebraic calculations, it results in

$$\begin{aligned} \mathcal{T} &= \frac{1}{2}M \|\dot{p}_b\|^2 + \frac{1}{2}\omega^T I \omega \\ &= \frac{1}{2} \begin{bmatrix} \dot{p}_b \\ \omega \end{bmatrix}^T \underbrace{\begin{bmatrix} M\mathcal{I} & 0 \\ 0 & I \end{bmatrix}}_{\mathcal{M}} \begin{bmatrix} \dot{p}_b \\ \omega \end{bmatrix} \end{aligned} \quad (\text{A.54})$$

where \mathcal{M} is the *generalized inertia matrix* with respect to \mathcal{F}_b , such that $\mathcal{M} > 0$ and $\mathcal{M}^T = \mathcal{M}$, and ω is the angular velocity with respect to the body frame.

A.2.5 Lagrange Equations of Motion

It consists of a reformulation of the classical mechanics, in which the motion of a system of particles can be described by a set of first-order differential equations, obtained by its balance of energy.

A rigid body in motion is subject to constraints between the positions of its particles and has limited DoF. From the Newton's Second Law, a dynamic system of n particles can be written as

$$f = \begin{bmatrix} M_1 \mathcal{I} & & 0 \\ & \ddots & \\ 0 & & M_n \mathcal{I} \end{bmatrix} \begin{bmatrix} \ddot{r}_1 \\ \vdots \\ \ddot{r}_n \end{bmatrix} + \sum_{j=1}^k \Gamma_j \lambda_j \quad (\text{A.55})$$

where r_i is the particle position with respect to \mathcal{F}_b , M_i is the mass, the vectors $\Gamma_j \in \mathbb{R}^{3n}$ where $j = 1, \dots, k$ form a basis for the forces of constraint and λ_j are the Lagrange multipliers. Given a system with generalized coordinates $q \in \mathbb{R}^n$, the *Lagrangian* of a mechanical system is defined as the difference between its total kinetic and potential energy and can be represented by the equation

$$\mathcal{L}(q, \dot{q}) = \mathcal{T}(q, \dot{q}) - \mathcal{U}(q) \quad (\text{A.56})$$

and the Lagrange equations are given by

$$\frac{d}{dt} \frac{\delta \mathcal{L}}{\delta \dot{q}_i} - \frac{\delta \mathcal{L}}{\delta q_i} = \xi_i, \quad i = 1, \dots, n \quad (\text{A.57})$$

where ξ_i is the generalized force associated with q_i . Thus, these equations represent the relation between generalized forces ξ_i applied to the body and its position, velocity and acceleration (SICILIANO *et al.*, 2009). Hence, it is a friendly way to formulate the dynamics of a mechanical system, whereas it reduces the number of equations necessary to describe it to the number of generalized coordinates (MURRAY *et al.*, 1994).

A.2.6 Newton-Euler Equations

Another common way to formulate the dynamics of a system is by means of the Newton-Euler equations. They are mostly used when the Lagrange equations cannot be directly employed to determine the equations of motion, unless one chooses a local parametrization for the configuration space (SICILIANO *et al.*, 2009). Thus, the Newton-Euler formalism allows to describe globally the dynamics of a rigid body subject to forces and torques.

With the definitions explained so far, now it is possible to provide a general dynamics equation of a rigid body according to the Newton's Second Law. Let p and R be the pose of the center of mass with respect to an inertial frame, consider a force F applied to its center of mass and a torque τ which makes rigid body rotate. The dynamics is given by:

$$\begin{aligned} f &= M \frac{d}{dt} \dot{p} \\ \tau &= \frac{d}{dt} (I_a \omega) \end{aligned} \quad (\text{A.58})$$

By defining $I_a = RIR^T$ and calculating the right side of the torque equation, it results in

$$\tau = I_a \dot{\omega} + \omega \times I_a \quad (\text{A.59})$$

which is called *Euler equation* (MURRAY *et al.*, 1994). Nevertheless, the dynamics of a rigid body is written as seen from the inertial frame. For further calculations, it is necessary to convert it to the body frame, and to proceed so, the transformation of these equations is carried out by doing $v_b = R^T \dot{p}$ and setting the body force $f_b = R^T f$ (MURRAY *et al.*, 1994). Then, the equations present in A.58 are rewritten as

$$\begin{aligned} f &= \frac{d}{dt}(M\dot{p}) \\ &= \frac{d}{dt}(MR\dot{v}_b) \\ &= RM\dot{v}_b + \dot{R}Mv_b \end{aligned} \quad (\text{A.60})$$

By left-multiplying both sides by R^T , it turns out to be

$$\begin{aligned} f_b &= R^T RM\dot{v}_b + R^T \dot{R}Mv_b \\ &= M\dot{v}_b + \omega \times Mv_b \end{aligned} \quad (\text{A.61})$$

On the other hand, similar calculations are done for the torques. By converting torques, the inertia tensor I and the body angular velocity ω , it yields

$$\tau_b = I\dot{\omega}_b + \omega_b \times I\omega_b \quad (\text{A.62})$$

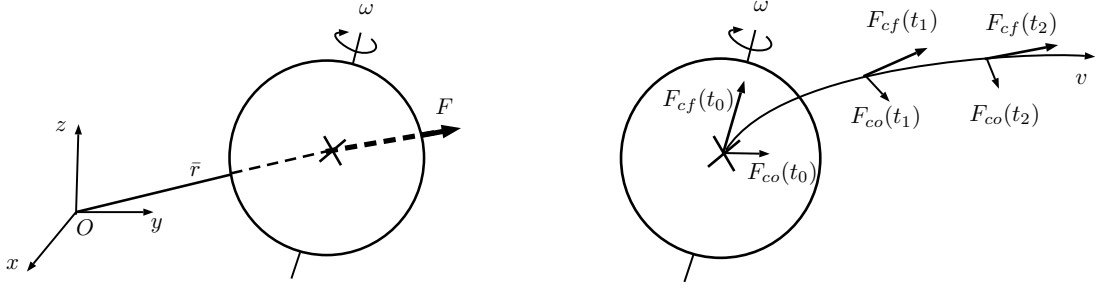
which is also known as Euler equation. Both equations can be rewritten in a matricial form as

$$\begin{bmatrix} M\mathcal{I} & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} \dot{v}_b \\ \dot{\omega}_b \end{bmatrix} + \begin{bmatrix} \omega_b \times Mv_b \\ \omega_b \times I\omega_b \end{bmatrix} = \begin{bmatrix} f_b \\ \tau_b \end{bmatrix} \quad (\text{A.63})$$

and are known as the *Newton-Euler equation*.

A.2.7 Generalized Forces Transformation

Similarly to (A.41), generalized forces can also be propagated along frames. If a force f and a torque τ are applied to a body, they are sensed at frames \mathcal{F}_b and \mathcal{F}_c located inside the rigid body. From the power conservation, the relation between these wrenches is given by



(a) A force f is applied on rotating body. (b) Centrifugal (f_{cf}) and Coriolis Force (f_{co}).

Figure A.6: When there is a rotational motion ω , it appears Centrifugal and Coriolis forces, which changes the body acceleration.

$$h_c = \begin{bmatrix} f_c \\ \tau_c \end{bmatrix} = \begin{bmatrix} \mathcal{I} & 0 \\ {}^c p_b \times & \mathcal{I} \end{bmatrix} \begin{bmatrix} f_b \\ \tau_b \end{bmatrix} = {}^c G_b h_c \quad (\text{A.64})$$

with respect to the inertial frame, where ${}^b G_c$ is the adjoint transformation matrix, defined in the equation A.41.

A.2.8 Coriolis and Centrifugal Forces

Coriolis and Centrifugal forces are inertial forces which arise in rigid bodies in motion and whose reference frame rotates with respect to an inertial frame, as depicted in the figure A.6a. As a result, these forces are seen by an observer on a rotating surface but not by a stationary observer at the inertial frame (HAY, 2016) and therefore they are also called "fictitious forces". In the equation A.63, they are present in the term

$$\begin{bmatrix} f_{cf} \\ f_{co} \end{bmatrix} = \begin{bmatrix} \omega_b \times M v_b \\ \omega_b \times I \omega_b \end{bmatrix} \quad (\text{A.65})$$

where the upper block matrix corresponds to the centrifugal f_{cf} , and the lower, to the Coriolis force f_{co} . These forces are orthogonal to each other, as shown in the equation A.6b, and they actuate on the applied force f by changing the direction of f as time elapses. The equation A.65 proves that these fictional forces are always present as long as the body presents a rotational motion. However, the centrifugal force appears only when there exists a translational motion along.

A.2.9 Other Dynamic Effects

There are others dynamic effects which influence directly the body acceleration. Depending on the application, they might be taken into account when modeling, simulating and controlling a robotic system. These effects will be briefly explained

next.

Gravity

Gravity, also defined as the Newton's Law of Universal Gravitation, is a natural law which states that every body attracts another in the Universe with a force that is directly proportional to their masses and inversely proportional to the square of the distance between their centers of mass. This statement can be mathematically described by

$$f_g = G \frac{M_1 M_2}{d^2} \quad (\text{A.66})$$

where G is the gravitational constant, M_i is the body mass and d is the distance between them.

On Earth, one can consider that every rigid body has a mass negligible, and as a consequence, the force $f_g \in \mathbb{R}^3$ that the Earth attracts a rigid body is given by

$$f_g = \begin{bmatrix} 0 \\ 0 \\ Mg \end{bmatrix} \quad (\text{A.67})$$

where g is the gravity acceleration.

Friction

Friction is the resistance that a body, fluid or particle encounters when sliding on another. There are two main types of friction:

- Static, which is the friction between two or more non-moving bodies relative to each other. To put a body into motion, it is necessary to overcome the static friction;
- Dynamic, when a body moves or tends to move relative to another and their surfaces rub against each other. Then, it is the resistance against its motion.

Appendix B

Proof of the KKT Conditions

Here is a proof of the KKT conditions presented in 2.21, according to NOCEDAL and WRIGHT (1999). Consider an optimization problem, which concerns with minimizing an objective function subject to equality and inequality constraints. A general formulation is

$$\begin{aligned} \min.: & f(u) \\ \text{s.t.} & c_i(u) = 0, i \in \mathcal{E} \\ & c_i(u) \geq 0, i \in \mathcal{G}. \end{aligned} \tag{B.1}$$

where f is the objective function and $c_i \in \mathbb{R}^n$. \mathcal{G} and \mathcal{E} are set of indices, related to equality and inequality constraints, respectively, which define all the points u where the constraints hold, called feasible region.

$$S = \{u | c_i(u) = 0, i \in \mathcal{E}; c_i(u) \geq 0, i \in \mathcal{G}\} \tag{B.2}$$

Prior to proving the KKT theorem, it is necessary to make some definitions, which are:

Definition B.1. *The active set \mathcal{W} at any feasible u is the union of the set \mathcal{E} with the indices of the active inequality constraints, that is*

$$\mathcal{W}(u) = \mathcal{E} \cup \{i \in \mathcal{G} | c_i(u) = 0\}. \tag{B.3}$$

Definition B.2 (LICQ). *Given the point u^* and the active set \mathcal{W} , we say that the linear constraint qualification (LICQ) holds if the active set constraint gradients $\{\nabla c_i(u^*), i \in \mathcal{W}(u^*)\}$ is linearly independent.*

Theorem B.1 (First-Order Necessary Conditions). *Suppose that u^* is a local solution of (B.1) and that the LICQ holds at u^* . There is a Lagrange multiplier vector λ^* , with components $\lambda_i, i \in \mathcal{E} \cup \mathcal{G}$, such that the following conditions are satisfied at (u^*, λ^*) :*

$$\nabla_u \mathcal{L}(u^*, \lambda^*) = 0, \quad (\text{B.4a})$$

$$c_i(u^*) = 0, \forall i \in \mathcal{E}, \quad (\text{B.4b})$$

$$c_i(u^*) \geq 0, \forall i \in \mathcal{G}, \quad (\text{B.4c})$$

$$\lambda_i^* \geq 0, \forall i \in \mathcal{G}, \quad (\text{B.4d})$$

$$\lambda_i^* c_i(u^*) = 0, \forall i \in \mathcal{E} \cap \mathcal{G}. \quad (\text{B.4e})$$

where \mathcal{L} is the Lagrangian of the optimization problem. These conditions are also known as the Karush-Kuhn-tucker conditions. From the complementarity condition, the Lagrange multipliers related to the inactive inequality constraints are zero, then B.4a can be rewritten as

$$0 = \nabla_u \mathcal{L}(u^*, \lambda^*) = \nabla f(u^*) - \sum_{i \in \mathcal{W}} (u^*) \lambda_i^* \nabla c_i(u^*) \quad (\text{B.5})$$

Before presenting a formal proof of the First-Order Necessary Conditions theorem, it is necessary to establish a few more theorems and lemmas with their respective proofs.

Definition B.3 (Strict Complementarity). *Given a local solution u^* and a vector λ^* satisfying (B.4), if exactly one of λ_i^* and $c_i(u^*)$ is zero for each index $i \in \mathcal{G}$, then the strict complementarity condition holds.*

Definition B.4. *Suppose that the KKT conditions (B.4) hold for u^* . We say that an inequality constraint c_i is strongly active or binding if $i \in \mathcal{W}(u^*)$ and $\lambda_i^* > 0$ for a Lagrange multiplier that satisfies (B.4). If $i \in \mathcal{W}(u^*)$ and $\lambda_i = 0$ for all λ^* , then c_i is called weakly active.*

Definition B.5 (Feasible Sequences). *Given a feasible point u^* , a sequence $\{z_k\}_{k=0}^{\infty}$ with $z_k \in \mathbb{R}^n$ is a feasible sequence if $z_k \neq u^*$ for all k , $\lim_{k \rightarrow \infty} z_k = u^*$ and z_k is feasible for all sufficiently large values of k . The set of all possible feasible sequences approaching u^* is denoted $\mathcal{T}(u^*)$.*

Definition B.6 (Limiting Direction). *Given \mathcal{S}_d as some subsequence of $\{z_k\}_{k=0}^{\infty}$ towards a local solution u^* , the limiting directions of a feasible sequence are vectors d such that*

$$\lim_{z_k \in \mathcal{S}_d} \frac{z_k - u^*}{\|z_k - u^*\|} \rightarrow d \quad (\text{B.6})$$

Theorem B.2. *If u^* is a local solution of B.1, then all feasible sequences $\{z_k\}$ in $\mathcal{T}(u^*)$ must satisfy*

$$\nabla f(u^*)^T d \geq 0 \quad (\text{B.7})$$

Proof. Consider that there exists a $\{z_k\}$ with the property $\nabla f(u^*)^T d < 0$. From the Taylor's theorem, for any $z_k \in \mathcal{S}_d$, it results that

$$f(z_k) < f(u^*) + \frac{1}{2} \|z_k - u^*\| d^T \nabla f(u^*), \text{ for all } k \text{ sufficiently large.} \quad (\text{B.8})$$

where the superior order terms are ignored because of the property set. Hence, for any neighborhood of u^* , a sufficiently large k can be chosen, such that z_k lies within this neighborhood and has a lower value of the objective function f . Therefore, u^* is not a local solution. \square

Lemma B.1. *The following two statements are true.*

(i) *If $d \in \mathbb{R}^n$ is a limiting direction of a feasible sequence, then*

$$\begin{aligned} d^T \nabla c_i^* &= 0, \quad \forall i \in \mathcal{E} \\ d^T \nabla c_i^* &\geq 0 \quad \forall \mathcal{W}(u^*) \cap \mathcal{G} \end{aligned} \quad (\text{B.9})$$

(ii) *If (B.9) holds for $\|d\| = 1$ and the LICQ condition is satisfied, then $d \in \mathbb{R}^n$ is a limiting direction of some feasible sequence.*

Proof. Without loss of generality, assume that all constraints $c_i(\cdot)$, with $i = 1, 2, \dots, m$ are active. To prove (i), let $\{z_k\} \in \mathcal{T}(u^*)$ be some feasible sequence and assume that

$$\lim_{k \rightarrow \infty} \frac{z_k - u^*}{\|z_k - u^*\|} = d \quad (\text{B.10})$$

which is rewritten as

$$z_k = u^* + \|z_k - u^*\| d + o(\|z_k - u^*\|) \quad (\text{B.11})$$

From the Taylor's theorem and with $i \in \mathcal{E}$, it results in

$$\begin{aligned} 0 &= \frac{1}{\|z_k - u^*\|} c_i(z_k) \\ &= \frac{1}{\|z_k - u^*\|} [c_i(u^*) + \|z_k - u^*\| \nabla c_i^T d + o(\|z_k - u^*\|)] \\ &= \nabla c_i^T d + \frac{o(\|z_k - u^*\|)}{\|z_k - u^*\|} \end{aligned} \quad (\text{B.12})$$

When $z \rightarrow \infty$, the superior order term is negligible and $\nabla c_i^T d = 0$, as desired to prove. For the active inequality constraints, it similarly results in

$$0 \geq c_i + \frac{o\|z_k - u^*\|}{\|z_k - u^*\|} \quad (\text{B.13})$$

which for $k \rightarrow \infty$ results in $\nabla c_i^T d \geq 0$, as we wanted to prove. For proving (ii), the LICQ condition holds and the matrix $A \in \mathbb{R}^{m \times n}$ of active constraint gradients has full row rank m . Let Z be a matrix whose columns are a basis for the null space of A . Suppose also that $\{t_k\}_{k=0}^\infty$ is any sequence of positive scalars such that $\lim_{k \rightarrow \infty} t_k = 0$. Define the parametrized system of equations $R: \mathcal{R}^n \times \mathcal{R} \rightarrow \mathbb{R}^n$ by

$$R(z, t) = \begin{bmatrix} c(z) - tAd \\ Z^T(z - u^* - td) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad (\text{B.14})$$

We need to prove that for each $t = t_k$, the solutions $z = z_k$ for small $t > 0$ give a feasible sequence towards u^* . For $t = 0$, the solution is $z = u^*$, and the Jacobian of R is

$$\nabla_R(u^*, 0) = \begin{bmatrix} A \\ Z^T \end{bmatrix} \quad (\text{B.15})$$

which is nonsingular. Hence, from the implicit function theorem, the system has a unique solution z_k for all values of t_k sufficiently small. From (i) and (B.14), we have that

$$i \in \mathcal{E} \Rightarrow c_i(z_k) = t_k \nabla c_i^T d = 0, \quad (\text{B.16a})$$

$$i \in \mathcal{W}(u^*) \cap \mathcal{G} \Rightarrow c_i(z_k) = t_k \nabla c_i^T d \geq 0. \quad (\text{B.16b})$$

so z_k is feasible. Moreover, given a $t = \bar{t} > 0$, we cannot have $z(t) = u^*$, since otherwise by replacing $(z, t) = (u^*, \bar{t})$ into (B.14), it results in

$$\begin{bmatrix} c(u^*) - \bar{t}Ad \\ -Z^T(\bar{t}d) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad (\text{B.17})$$

Since $c(u^*) = 0$ and $\bar{t} > 0$ and (B.15) has full rank, it results that $d = 0$, which contradicts $\|d\| = 1$ and hence $z_k = z(t_k) \neq u^*$ for all k .

The second part of proof consists of showing that d is a limiting direction of $\{z_k\}$. As $R(z_k, t_k) = 0$ for all k with the Taylor's theorem, we have that

$$\begin{aligned}
0 = R(z_k, t_k) &= \begin{bmatrix} c(z_k) - t_k A d \\ Z^T(z_k - u^* - t_k d) \end{bmatrix} \\
&= \begin{bmatrix} A(z_k - u^*) + o(\|z_k - u^*\|) - t_k A d \\ Z^T(z_k - u^* - t_k d) \end{bmatrix} \\
&= \begin{bmatrix} A \\ Z^T \end{bmatrix} (z_k - u^* - t_k d) + o(\|z_k - u^*\|)
\end{aligned} \tag{B.18}$$

By dividing this equation by $\|z_k - u^*\|$ and using non-singularity of the coefficient matrix in the first term, it yields

$$\lim_{k \rightarrow \infty} d_k - \frac{t_k}{\|z_k - u^*\|} d = 0, \text{ where } d_k = \frac{z_k - u^*}{\|z_k - u^*\|} \tag{B.19}$$

Provided that $\|d_k\| = 1$ for all k and since $\|d\| = 1$, it results in

$$\lim_{k \rightarrow \infty} \frac{t_k}{\|z_k - u^*\|} = 1 \tag{B.20}$$

Hence, from (B.19) it results in $\lim_{k \rightarrow \infty} d_k = d$, as we wanted to prove. \square

Definition B.7. Given a point u^* and the active constraint set $\mathcal{W}(u^*)$, the set F_1 is defined as

$$F_1 = \left\{ \alpha d \mid \alpha > 0, \begin{array}{l} d^T \nabla c_i^* = 0, \quad \forall i \in \mathcal{E}; \\ d^T \nabla c_i^* \geq 0, \quad \forall i \in \mathcal{W}(u^*) \cap \mathcal{G} \end{array} \right\} \tag{B.21}$$

Lemma B.2. There is no direction $d \in F_1$ for which $d^T \nabla f^* < 0$ if and only if there exists a vector $\lambda \in \mathbb{R}^m$ with

$$\nabla f^* = \sum_{i \in \mathcal{W}(u^*)} \lambda_i \nabla c_i^* = A(u^*)^T \lambda, \quad \lambda_i \geq 0 \text{ for } i \in \mathcal{W}(u^*) \cap \mathcal{G} \tag{B.22}$$

Proof. Consider a cone N defined by

$$N = \left\{ s \mid s = \sum_{i \in \mathcal{W}(u^*)} \lambda_i \nabla c_i^*, \quad \lambda_i \geq 0 \text{ for } i \in \mathcal{W}(u^*) \cap \mathcal{G} \right\} \tag{B.23}$$

then B.22 is equivalent to $\nabla f^* \in N$, such that N is closed. Suppose now that (B.22) holds and d is chosen to satisfy (B.9). Then it turns out that

$$d^T \nabla f^* = \sum_{i \in \mathcal{E}} \lambda_i (d^T \nabla c_i^*) + \sum_{i \in \mathcal{W}(u^*) \cap \mathcal{G}} \lambda_i (d^T \nabla c_i^*) \tag{B.24}$$

For $i \in \mathcal{E}$, $d^T \nabla c_i^* = 0$ and the first summation results in zero, whereas the second term is nonnegative because $\lambda_i \geq 0$ and $d^T \nabla c_i^* \geq 0$ for $i \in \mathcal{W}(u^*) \cap \mathcal{G}$. Therefore, $d^T \nabla f^* \geq 0$.

Reversely, we can demonstrate that if ∇f^* does not satisfy (B.22), then one can proceed to search a vector d , such that it satisfies $d^T \nabla f^* < 0$ and (B.9). Let $\hat{s} \in N$ be a vector close to ∇f^* . Since N is closed, \hat{s} is well-defined and solves the following constrained optimization problem

$$\begin{aligned} \min: \quad & f(s) = \|s - \nabla f^*\|_2^2 \\ \text{s.t.}: \quad & s \in N \end{aligned} \tag{B.25}$$

Since $\hat{s} \in N$, then $t\hat{s} \in N$ for all scalars $t \geq 0$. Since $\|t\hat{s} - \nabla f^*\|_2^2$ is minimized at $t = 1$, we have that

$$\left. \frac{d}{dt} \|t\hat{s} - \nabla f^*\|_2^2 \right|_{t=1} = \hat{s}^T (\hat{s} - \nabla f^*) = 0 \tag{B.26}$$

Now let another vector $s \in N$. Since N is convex, from the minimizing property of \hat{s} , we have that

$$\|\hat{s} + \theta(s - \hat{s}) - \nabla f^*\|_2^2 \geq \|\hat{s} - \nabla f^*\|_2^2 \quad \forall \theta \in [0, 1] \tag{B.27}$$

and hence

$$2\theta(s - \hat{s})^T (\hat{s} - \nabla f^*) + \theta^2 \|s - \hat{s}\|_2^2 \geq 0. \tag{B.28}$$

If this equation is divided by θ and by calculating its limit as $\theta \rightarrow 0$ because of (B.26), it yields

$$s^T (\hat{s} - \nabla f^*) \geq 0, \quad \forall s \in N. \tag{B.29}$$

Now, we need to prove that the vector

$$d = \hat{s} - \nabla f^* \tag{B.30}$$

is valid for (B.9) and $d^T \nabla f^* < 0$. Note that $d \neq 0$ because ∇f^* does not belong to cone N . We have from (B.26) that

$$\begin{aligned} d^T \nabla f^* &= d^T (\hat{s} - d) \\ &= (\hat{s} - \nabla f^*)^T \hat{s} - d^T d \\ &= -\|d\|_2^2 < 0 \end{aligned} \tag{B.31}$$

so that d satisfies the descent property. If one proceeds to a proper choice of λ_i , with $i = 1, 2, \dots, m$, we have that

$$\begin{aligned} i \in \mathcal{E} &\Rightarrow \nabla c_i^* \in N \text{ and } -\nabla c_i^* \in N; \\ i \in \mathcal{W}(u^*) \cap \mathcal{G} &\Rightarrow \nabla c_i^* \in N. \end{aligned} \tag{B.32}$$

Hence, from (B.29), we perform a substitution $d = \hat{s} - \nabla f^*$ and the particular choices $s = \nabla c_i^*$ and $s = -\nabla c_i^*$, which results in

$$\begin{aligned} i \in \mathcal{E} &\Rightarrow d^T \nabla c_i^* \geq 0 \text{ and } -d^T \nabla c_i^* \geq 0 \Rightarrow d^T \nabla c_i^* = 0 \\ i \in \mathcal{W}(u^*) \cap \mathcal{G} &\Rightarrow d^T \nabla c_i^* \geq 0. \end{aligned} \tag{B.33}$$

which satisfies (B.9), so the reverse implication is proved. \square

Finally, the First-Order Conditions can be proved, as follows:

Proof. First, assume that $u^* \in \mathbb{R}^n$ is a feasible point at which LICQ holds. The theorem says that if u^* is a local solution of (B.1), then there is a vector λ^* that satisfies the KKT conditions (B.4).

First, it is important to demonstrate that λ_i satisfies (B.22). The theorem B.2 states that $d^T \nabla f^* \geq 0$ is valid for every d of feasible sequences $\{z_k\}$. From Lemma B.1 the LICQ conditions hold and the set of all possible d matches the set of vectors that satisfy conditions (B.9). From both statements, it implies that all limiting directions d that satisfy (B.9) must have $d^T \nabla f^* \geq 0$. Hence, from Lemma B.2, there must exist a vector λ for which (B.22) holds.

Consider a vector λ^* , such that

$$\lambda_i^* = \begin{cases} \lambda_i, & i \in \mathcal{W}(u^*), \\ 0, & \text{otherwise} \end{cases} \tag{B.34}$$

which shows that λ^* and u^* satisfies the KKT conditions.

By checking the conditions, we have that the condition (B.4a) is achieved from (B.22), from the Lagrangian function and from (B.34). The conditions (B.4b) and (B.4c) are satisfied as well, provided that u^* is feasible. From (B.22), $\lambda_i^* \geq 0$ for $i \in \mathcal{W}(u^*) \cap \mathcal{G}$, while from (B.34), $\lambda_i^* = 0$ for $i \in \mathcal{G} \setminus \mathcal{W}(u^*)$. Hence, $\lambda_i^* \geq 0$ for $i \in \mathcal{G}$, so that (B.4d) is satisfied. Finally, for $i \in \mathcal{W}(u^*) \cap \mathcal{G}$, $c_i(u^*) = 0$, while for $i \in \mathcal{G} \setminus \mathcal{W}(u^*)$, we have $\lambda_i^* = 0$. Thus, $\lambda_i^* c_i(u^*) = 0$ for $i \in \mathcal{G}$, so that (B.4e) holds, which completes the proof. \square