National Technical University of Athens
School of Mechanical Engineering
Section of Mechanical Design and Automatic Control
Control Systems Laboratory

Diploma Thesis

On the Dynamic Modelling, Trajectory Planing and Control of a Space Robot Emulator

Nikolena Christofi

Supervising Professor: Evangelos G. Papadopoulos

February 2017

"Few people realise the immensity of vacancy in which the dust of the material universe swims."

- H.G. Wells, The War of the Worlds

--

Acknowledgments

First and foremost, I have to thank my research supervisor, Professor Evangelos Papadopoulos. Without his assistance and dedicated involvement in every step throughout the process, this thesis would have never been accomplished. I would like to thank you for your support and understanding over these past three years, and most importantly your for patience, since I was away, until the end, the completion of my thesis.

Getting through my dissertation required more than academic support, and I have many, many people to thank for listening to and, at times, having to tolerate me over the past three years. I cannot begin to express my gratitude and appreciation for their friendship.

To all the members of the space lab, who have been unwavering in their personal and professional support during the time I spent at the csl. For many memorable evenings out and in, I must thank everyone above as well as the legged lab -even in the hardest of days you would make me smile.

Most importantly, none of this could have happened without my family. My grandmother, who offered her encouragement with every opportunity – despite my own limited devotion to correspondence. With their own brand of humour, Marcos and Vassilis have been kind and supportive to me over the last several years, in every country I have been, in every stage, from leaving Athens and working at ESA, to coming back two years later to undertake my thesis at csl; a family without borders. To Rob, my parents and my brother, always there to lift me up and keep me going.

Every time I was ready to quit, you did not let me and I am forever grateful. This dissertation stands as a testament to your unconditional love and encouragement.

Preface

This diploma thesis serves as the last part of the research project undertaken by Nikolena Christofi in the Space Robotics Lab of the Control Systems Laboratory (csl) of the School of Mechanical Engineering at the National Technical University of Athens, from January 2016 to February 2017, under the supervision of Professor Evangelos Papadopoulos.

This paper tries to construe the work done on the dynamics and control of a space robot emulator with 8 degrees of freedom and a pair of two-link manipulators.

Abstract

In an attempt to construct a full dynamic model that sufficiently describes the dynamic characteristics of a space robot and its behaviour while executing certain on-board servicing tasks, a research was made, which resulted in a model-based controller, considering all the 7 degrees of freedom of the robot. Based on the mission scenario, the controller analyses the desired outcome to commands sent straight to the robot's actuators, the operation of which achieves the predefined objectives.

Using an optimisation method, the torques and forces needed to act on the robot's state variables, are translated into the forces and torques which shall act on the system' actuators, which are the three thrusters sets and servo motors mounted on the two manipulators' links, in each arm. The robot emulator is capable of performing a planar movement, within the air bearing testbed of the Space Robotics Lab of the csl and shall conduct trajectory planning to reach, grasp and move along a predefined space moving target.

This paper tackles the analysis of the approach followed for the controller and planner composition, taken under consideration the physical constraints imposed on the system, by the actuators of the actual robot, and its mechanical components. The modelling of the latter was included in both the dynamic modelling of the system as well as in the control module.

The dynamic analysis and control are simulated in Matlab 2016b and Simulink; the results are presented in Chapter 6. Further investigation on the simulation capabilities of Gazebo and ROS are being explored, in an effort to achieve a realistic simulation of the behavior of space robots in virtual environments.

Περίληψη

Ένα από τα μεγαλύτερα προβλήματα που αντιμετωπίζει η ρομποτική στο διάστημα είναι η πλήρης προσομοίωση της συμπεριφοράς των σωμάτων στη γη, ειδικά σε περιπτώσεις που αφορούν τη συνεργασία δύο και περισσότερων σωμάτων που βρίσκονται σε τροχιά. Με το πρόβλημα των διαστημικών αποβλήτων συνεχώς να αυξάνεται, καθώς και τον αυξημένο αριθμό των σωμάτων σε τροχιά γύρω από τη γη, η αναγκαιότητα της επίτευξης επιτυχούς συνεργασίας μεταξύ ενεργών και παθητικών σωμάτων είναι επίκαιρη και επιτακτική.

Στην εργασία αυτή έγινε η προσπάθεια πλήρους δυναμικής μοντελοποίησης ενός εξομοιωτή διαστημικού ρομπότ 7 βαθμών ελευθερίας και ο έλεγχός του ούτως ώστε να μπορεί να διεκτελέσει εργασίες με άλλα σώματα στο διάστημα, που βρίσκονται σε τροχιά -αφότου το διαστημικό ρομπότ έχει προσεγγίσει την τροχιά του παθητικού σώματος. Ο στόχος του ελέγχου είναι η επιτυχής συλλογή του παθητικού σώματος από το διαστημικό ρομπότ.

Η μοντελοποίηση έγινε στο διδιάστατο επίπεδο και ο έλεγχος είναι τύπου Model Based. Ο έλεγχος της θέσης και προσανατολισμού του διαστημικού εξομοιωτή επιτυγχάνεται με τον έλεγχο των επενεργητών του συστήματος - 3 (2 διευθύνσεις) thrusters & 1 reaction wheel.

Μετά την παρουσίαση της θεωρητικής ανάλυσης που ακολουθήθηκε, η οποία χρησιμοποιεί τη μέθοδο Euler-Lagrange για τη δυναμική μοντελοποίηση, παρουσιάζονται τα αποτελέσματα της προσομοίωσής των στο προγραμματιστικό περιβάλλον του Matlab, Simulink.

Η εργασία αυτή αποτελεί αποτέλεσμα της διπλωματικής εργασίας που εκπονήθηκε στο Εργαστήριο Αυτομάτου Ελέγχου, υπό την επίβλεψη του κύριου Ευάγελου Παπαδόπουλου, καθηγητή στη σχολή Μηχανολόγων Μηχανικών του ΕΜΠ, κατά τη διάρκεια του Ιανουαρίου 2016 - Φεβρουαρίου 2017.

Contents

List of Figures

List of Tables

Nomenclature

Physics Constants

g	Gravitational Constant	$6.67384 \times 10^{-11}\, N \cdot m^2/kg^2$
c	Speed of light in a vacuum inertial system	$299,792,458\, m/s$
h	Plank Constant	$6.62607 \times 10^{-34}\, Js$
π	Ratio of the circle's circumference to its diameter	$3.14928375903247590873 24.....$
p_a	Ambient static pressure	$1\, bar$
T	Kinetic Energy of the System	
V	Dynamic Energy of the System; work produced by the conservative forces	

$L = T - V$ Lagrange term

Number Sets

$\mathbb{H}$	Quaternions
$\mathbb{C}$	Complex Numbers
$\mathbb{R}$	Real Numbers

Other Symbols

$\ddot{\theta}_b$	Absolute acceleration of the robot base
$\ddot{\theta_w}$	Absolute acceleration of the reaction wheel
$\ddot{e}_j$	Error vector of the acceleration of the system's states, at any given moment

$\ddot{q}_c$	Acceleration vector of the system's state variables, calculated by the controller
$\ddot{q}_{des}$	Vector containing the desired acceleration of each state of the system, at any given moment
δq_j	Virtual displacementsδqj(marginal shifts of the chosen generalised coordinates, q_j)
δW_j	Virtual work of the non-conservative torques and forces acting on the system
$\dot{\omega}$	Angular acceleration of the load
$\dot{\theta_b}$	Absolute velocity of the robot base
$\dot{\theta_w}$	Absolute velocity of the reaction wheel
$\dot{e}$	Errors matrix denoting the velocity error values for each state variable in every calculation repetition, equal to the subtraction of the desired velocity of the state variables by their current velocity value
$\dot{e}_j$	Error vector of the velocity of the system's states, at any given moment
$\dot{m_{exp}}$	Measured gas mass flow rate (experimental)
$\dot{m_{th}}$	Gas mass flow rate -calculated (theoretical)
$\dot{m}$	Fuel mass ratio
$\dot{m}$	Gas mass flow rate
$\dot{P}$	Thrust time derivative
$\dot{q}_{des}$	Vector containing the desired velocity of each state of the system, at any given moment
$\dot{V}$	Velocity time derivative
$\dot{x_t}(t)$	Time derivative of the displacement of the target spacecraft on the x-axis; x-velocity of the target
$\dot{y_t}(t)$	on of movement of the target spacecraft on the y-axis $y_t(t) = mt + n$
η	Efficiency ratio of the planetary gears of the motion transmission system
γ, R_g	CO2 gas constants
ß$_a$	Electric current of the DC motor

t	Electro-mechanical constant of the DC motor
p_e	Momentum of each thruster exhaust
Ω	Thrust generated due the gas expansion
ω	Angular velocity of the load
ω_j	Natural frequency of the second order system
Ω_{exp}	Thrust generated by the nozzles (experimental)
Ω_{th}	Thrust generated by the nozzles -calculated (theoretical)
$\overrightarrow{\mathrm{F}}$	Force generated by the movement of the arms
$\overrightarrow{\mathrm{r}}$	Vector that connects the arm attached to the robot's body COM, to the COM of the base of the robot
$\boldsymbol{p}(t_1)$	Momentum of the spacecraft before the thrusters' ignition
$\boldsymbol{p}(t_2)$	Momentum of the spacecraft after the thrusters' ignition
ρ	Friction Index
ρ	Gas density
τ_1	Acting force on the first joint of the arm, generated by the motor attached to the joint
τ_2	Acting force on the second joint of the arm, generated by the motor attached to the joint
τ_θ	Torque acting on the COM of the robot's system
τ_m	Maximum output torque produced by the motors of the arms
τ_m	Torque generated by the motor of the Reaction Wheel
τ_{q11}	Torque acting on the COM of the first link of the first arm of the robot
τ_{q12}	Torque acting on the COM of the second link of the first arm of the robot
τ_{q21}	Torque acting on the COM of the first link of the second arm of the robot
τ_{q22}	Torque acting on the COM of the second link of the second arm of the robot

$\tau_{q_{ij}}$ Torque transferred to the system from the actuators of the arms

τ_{RW} Torque transferred from the Reaction Wheel to system

θ angle of the circle section

ζ Damping ratio of the second-order system; a real number that defines the damping properties of the system. More damping has the effect of less percent overshoot, and slower settling time. Damping is the inherent ability of the system to oppose the oscillatory nature of the system's transient response. Larger values of damping coefficient or damping factor produces transient responses with lesser oscillatory nature

ζ Friction coefficient between the gas and the tube walls

A Area of the nozzle outlet tip

$a1$ Acceleration of the robot between $t = 0 - t1s$

$a2$ Deceleration of the robot between $t = t2 - t3s$

a_{bot}^{max} Maximum acceleration of the robot; bound by the maximum thrust the thrusters can produce

a_1 Length of the first link of the arms, considered as a straight line connecting the first to the second joint of the arms

a_2 Length of the second link of the arms, considered as a straight line connecting the second joint of the arms to the End Effector

a_B Distance between the COM and the geometric centre of the main body of the robot

A_e Area of the tube's section

a_{c_1} Distance between the COM of the first link of the arms and its mounting point on the robot's main base frame

a_{c_2} Distance between the COM of the second link of the arms and the joint connecting the first to the second link

B DAMPING ratio of the Reaction Wheel

b_w Coefficient of friction of the reaction wheel

C Velocity matrix; contains Coriolis and centrifugal terms of the equation of motion - C$(\dot{q}, q)[7x1]$

C_c C matrix calculated by the controller

D Internal diameter of the nozzle

D Matrix that trans-forms the nozzle thrust (here forces) vector into planar forces

d_B Angle between the straight line connecting the COM of the main body of the robot and its geometrical centre, and the X axis of the absolute coordinate frame. This angle is used in order to define the initial orientation of the robot - if not rotated, set to zero

d_e Diameter of the tube's section

d_{a_1} Angle defining the offset of the COM of the first link of the arms in respect to the link's centre line -angle between a_1 and a_{c_1}

d_{a_2} Angle defining the offset of the COM of the second link of the arms in respect to the link's centre line -angle between a_2 and a_{c_2}

d_{B_a} Half of the angle created by the triangle connecting the first joint of the first arm, the main body's COM and the first joint of the second arm

D_h Hydraulic diameter of the nozzle

e Errors matrix denoting the displacement error values for each state variable in every calculation repetition, equal to the subtraction of the desired position of the state variables by their current position

E_1 End Effector of the upper arm

E_2 End Effector of the lower arm

e_j Error vector of the displacement of the system's states, at any given moment

$e_{\dot{m}_{real}}$ Divergence of $\dot{m}_{exp}$ and $\dot{m}_{th}$

$e_{\Omega_{real}}$ Divergence of Ω_{exp} and Ω_{th}

F Total Force generated by the thrusters, acting on the system's COM

f_m^{max} Maximum / Saturation value of torque generated by the motors of the arms

f_t^{max} Maximum / Saturation value of force generated by the thrusters

f_{RW}^{max} Maximum / Saturation value of torque generated by the Reaction Wheel

f_1 External Force acting on the first arm of the robot

f_1 Force generated by the first thruster set

f_2 External Force acting on the second arm of the robot

f_2 Force generated by the second thruster set

f_3 Force generated by the third thruster set

f_x Force acting on the COM of the robot's system, on the x-axis of the relative frame

f_y Force acting on the COM of the robot's system, on the y-axis of the relative frame

f_{1-6} force generated by each thruster of the robot, respectively

F_{max} Maximum value of the Force produced by the operation of a thruster

F_{tot} Total force acting on the COM of the robot, created by the addition of the forces generated by the projection of the force created by the thrusters' operation on the y-axis

f_{E1_x} Force acting on the EE of the first arm on the robot, on the x-axis of the relative system

f_{E1_y} Force acting on the EE of the first arm on the robot, on the y-axis of the relative system

f_{E2_x} Force acting on the EE of the second arm on the robot, on the x-axis of the relative system

f_{E2_y} Force acting on the EE of the second arm on the robot, on the y-axis of the relative system

H_{RB} Angular momentum H_{RB_1}, H_{RB_2} of the system, before and after a change of state, respectively

H_{robot} Momentum of the robot body

H_S Total momentum of the system $H_S(t_1), H_S(t_2)$, before and after a change of state, respectively

I_1 Total, centre-mass, polar Moment of Inertia of the first link of each arm

I_2 Total, centre-mass, polar Moment of Inertia of the second link of each arm

I_B Total, centre-mass, polar Moment of Inertia of the main body of the robot

I_b Moment of inertia of the robot base

I_t Total, centre-mass, polar Moment of Inertia of the secondary link of the first joint of the arms, including the two gears in carries

I_w Moment of inertia of the reaction wheel motor

I_{ma} Moment of Inertia of the driver of each electric motor

J Inertia of the load

J_E Jacobian matrix containing information on how the forces acting on the EE of the robot are transferred to the COM of the robot system [7x4]

J_w Normalisation Jacobian matrix [7x8]

J_{act} Jacobian matrix; contains information on the 1-order planar qualities of interest of the system; expressed in reference to the system's state variables [7x8]

$J_{c_{act}}$ Jacobian matrix [7x8] describing how the acting forces and torques on the system's state variables are transferred to the COM of the system, produced by the controller

K_B Kinetic Energy of the main body of the robot

K_D Velocity Gain Control matrix

K_P Displacement Gain Control matrix

K_{a_1} Kinetic Energy of the first manipulator

K_{a_2} Kinetic Energy of the second manipulator

K_{a_i1} Kinetic Energy of the first link of each arm, i=1,2

K_{a_i2} Kinetic Energy of the second link of each arm, i=1,2

K_{D_j} Velocity gain of each state variable of the system, analogous to latter natural frequency and damping ration

K_{m_i} Kinetic Energy of the motors' drivers, i=1,2

K_{P_j} Displacement gain of each state variable of the system, analogous to the square of the later's natural frequency

L Length of the nozzle

l_1 Distance between the COM of the first link of the manipulator and the second joint

l_2	Distance between the second joint and the COM of the second link of the manipulator
L_{sys}	Total Energy of the system
M	Mach number (M1 corresponds to the nozzle inlet and M2 to the nozzle outlet)
M	Mass matrix of the system's state variables - M(q) [7x7]
m	Mass values
m_0	Mass of the robot's base
m_1	Mass of the first link of the manipulator
m_1	Total mass of the first link of the each arm
m_2	Mass of the second link of the manipulator
m_2	Total mass of the second link of the each arm
m_B	Mass of the main robot frame, consisting of the circular base and all the arm parts connected to the base
M_c	M matrix calculated by the controller
m_f	Fuel mass $m_f = m_{CO_2}$
M_r	Total mass of the robot
m_{bot}	Total mass of the robot
n	REDUCTION RATIO of the planetary gears of the motion transmission system
n	
n_m^{RW}	Mechanical efficiency rate of the Reaction Wheel
P	Thrust value
p_1	Static pressure of the fluid before the nozzle outlet
p_2	Static pressure of the fluid right after it exits the nozzle
p_s	Static Pressure
p_t	Stagnation Pressure
p_t	Total pressure of the CO2 gas

p_{sc} Momentum of the spacecraft $p_{sc_{1-2}}$ before and after the thrusters' ignition

Q Vector containing the forces and torques acting on the system's state variables [7x1]

q States variables vector

Q_c Acting forces and torques to the COM of the system's state variables[8x1], generated by the controller

Q_E Acting forces vector; forces acting on the EE of the robot [4x1]

Q_f Generalised forces and torques, caused by the actuators and by the external non-conservative forces and torques, acting on the COM of the robot's system [7x1]

q_f Generalised coordinates, equal to the minimum number of the system's variables (State Variables) that can at any moment, uniquely fully describe its state

Q_j Virtual work done by Q acting along a virtual displacement δq

Q_{act} Acting Torques and Forces to the system [11x1]

$Q_{C_{act}}$ Acting forces and torques generated by the actuators of the system, including the real of the torque sent to the load (the robot) by the Reaction Wheel [8x1]

q_{des} Vector containing the desired position of each state of the system, at any given moment

$Q_{E_{act}}$ Acting forces in the EE vector [4x1]

Q_{end} Acting forces vector [7x1]; forces acting on the system's COM, generated by the forces acting on the EEs of the robot, transferred to the COM of the system by the Jacobian matrix J_E

$Q_{f_{1,2}}$ Force generated by the first set of thrusters

$Q_{f_{3,4}}$ Force generated by the second set of thrusters

$Q_{f_{5,6}}$ Force generated by the thirs set of thrusters

Q_{m_1} Maximum value for the torque required to be produced by the actuator (motor) attached to the first link of the first arm

Q_{m_2} Maximum value for the torque required to be produced by the actuator (motor) attached to the second link of the first arm

Q_{m_3} Maximum value for the torque required to be produced by the actuator (motor) attached to the first link of the second arm

Q_{m_4} Maximum value for the torque required to be produced by the actuator (motor) attached to the second link of the second arm

$Q_{act,5}$ Fifth element of the acting forces and torques vector; generated by the motor attached to the first link of the first arm

$Q_{act,6}$ Sixth element of the acting forces and torques vector; generated by the motor attached to the second link of the first arm

$Q_{act,7}$ Seventh element of the acting forces and torques vector; generated by the motor attached to the first link of the second arm

$Q_{act,8}$ Eighth element of the acting forces and torques vector; generated by the motor attached to the second link of the second arm

R Radius of circle

r Radius of the main body's circular frame; the length of the straight line connecting the geometrical centre of the base and the axis of the first link of the arm -the point connecting the first link to the main body frame

R_1^{max} Maximum value of the Path Depended Workspace; radius between the robot's base COM and the upper limit of the PDW

R_2^{max} Maximum value of the Path Independent Workspace; radius between the robot's base COM and the upper limit of the PIW

R_1^{min} Minimum value of the Path Depended Workspace; radius between the robot's base COM and the lower limit of the PDW

R_2^{min} Minimum value of the Path Independent Workspace; radius between the robot's base COM and the lower limit of the PIW

r_0 Perpendicular distance between the COM of the base and the first joint of the manipulator

r_1 Perpendicular distance between the first joint to the COM of the first link of the manipulator

r_2 Perpendicular distance between the COM of the second link and the EE of the manipulator

R_{WS} The radius withe centre the system's COM indicating the robot's EE's reachable workspace

s Root of the transfer function of the second-order system

T_f

t_s Settling time of the system

T_t Total Temperature of the CO2 gas

T_{act} Torque generated by the operation of the Reaction Wheel of the robot, resulting to an acting Torque on the system

T_C Torque acting on the robot's COM, produced by the acting internal and external Forces on the system

t_{meet} Time $t_{meet_{x,y}}$ when the chaser meets the target

T_{RW} Torque produced by the Reaction Wheel motor

T_{RW} Torque produced by the Reaction Wheel mounted on the robot

u_R Displacement velocity of the robot

u_{gas} Relative exit velocity of the gas to the nozzle

V Constant Volume

V Velocity value

V_j Gas Velocity before the tube's outlet tip

V_∞ Velocity of ambient air

V_{bot} Total velocity of the robot

$Vx_c(t)$ Velocity of the chaser spacecraft: x-vector $Vx_c(t) = \dot{x_c}(t) = 2at + b$

$Vx_c(t)$ Velocity of the chaser spacecraft: y-vector $Vy_c(t) = \dot{y_c}(t) = 2dt + e$

$Vx_t(t)$ Velocity of the target spacecraft on the x-axis $Vx_t(t) = \dot{x_t}(t)$

$Vy_t(t)$ Velocity of the target spacecraft on the y-axis $Vy_t(t) = \dot{y_t}(t)$

W Weights matrix [8x8]; normalises the acting forces and torques vector (brings the forces and torques to a common reference)

X Displacement of the robot's base on the x-axis at the absolute coordinate system

x Displacement of the robot's base on the x-axis

X_b Displacement of the robot's base on the x-axis at the relative coordinate system

$x_c(t)$ Equation of movement of the chaser spacecraft on the x-axis $x_c(t) = at^2 + bt + c$

$x_t(t)$ Equation of movement of the target spacecraft on the x-axis $x_t(t) = it+k$

x_{E_1} x-position of the End Effector of the upper arm

x_{E_2} x-position of the End Effector of the lower arm

x_{meet} Point on the x-axis that the chaser meets the target, when $t = t_{meet_x}$

$x_{q_{11}}$ Displacement of the COM of the first link of the first arm on the x-axis

$x_{q_{12}}$ Displacement of the COM of the second link of the first arm on the x-axis

$x_{q_{21}}$ Displacement of the COM of the first link of the second arm on the x-axis

$x_{q_{22}}$ Displacement of the COM of the second link of the second arm on the x-axis

x_{WS} Projection of R_{WS} on the x-axis

x_{WS_1} X-coordinate of R_{WS}

x_{WS_2} X-coordinate of the upper y-boundary of R_{WS}

x_{WS_3} X-coordinate of the lower y-boundary of R_{WS}

Y Displacement of the robot's base on the y-axis at the absolute coordinate system

y Displacement of the robot's base on the y-axis

Y_b Displacement of the robot's base on the y-axis at the relative coordinate system

$y_c(t)$ Equation of movement of the chaser spacecraft on the y-axis $y_c(t) = dt^2 + et + f$

$y_t(t)$ Equation of movement of the target spacecraft on the y-axis $y_t(t) = mt + n$

y_{E_1} y-position of the End Effector of the upper arm

y_{E_2} y-position of the End Effector of the lower arm

y_{meet}	Point on the y-axis that the chaser meets the target, when $t = t_{meet_y}$
$y_{q_{11}}$	Displacement of the COM of the first link of the first arm on the y-axis
$y_{q_{12}}$	Displacement of the COM of the second link of the first arm on the y-axis
$y_{q_{21}}$	Displacement of the COM of the first link of the second arm on the y-axis
$y_{q_{22}}$	Displacement of the COM of the second link of the second arm on the y-axis
y_{WS_1}	Y-coordinate of R_{WS}
y_{WS_2}	Y-coordinate of the upper y-boundary of R_{WS}
y_{WS_3}	Y-coordinate of the lower y-boundary of R_{WS}
Z	Friction coefficient between the gas and the nozzle material
τ_{ij}^{eff}	
τ_g^{max}	Maximum intermittently permissible torque at gear output
τ_{ma}^{max}	Stall torque value of the DC motor
Q_{act}^{eff}	Vector of the forces and torques produced by the actuators and acting on the system [8x1]

Acronyms

COM Centre of Mass. 11–14, 42, 49, 59–64, 66, 69, 70, 72, 75, 98–102, 104, 105, 107–109, 113–116, 120, 123–125, 128–131, 136, 138–142, 144–146, 151–153

CSL Control Systems Laboratory. 10, 29, 38, 39, 89

DOF Degrees of Freedom. 60, 99, 101

EE End Effector. 14, 59, 64, 71, 72, 75, 82, 100, 120, 121, 123, 136, 140, 142, 150

ESA European Space Agency. 10, 32, 33

ESOC European Space Operations Centre. 33

GC Geometrical Centre. 58, 60

I/O Input/Output. 53, 54

ISS International Space Station. 33–35, 42

LAN Local Area Network. 54

LiPo Lithium Polymer. 53

MBC Model Based Controller. 92

NASA National Aeronautics and Space Administration. 10, 32, 33

NTUA National Technical University of Athens. 29

PDW Path Dependent Workspace. 99

PIW Path Independent Workspace. 11, 99, 100, 102

PWM Pulse Width Modulation. 55

ROCC Rover Operations Control Centre. 33

ROS Robot Operating System. 54

RW Reaction Wheel. 10, 11, 49–53, 59, 60, 68, 69, 73, 78, 81, 83–88, 91, 92, 99, 133–135, 147, 148, 150

USSR Union of Soviet Socialist Republics, 1922-1991. 31

WS Working Space. 8, 11, 98, 99, 101, 102, 123, 136, 140

Chapter 1

Introduction

1.1 Motivation

The widespread use of robotics in space has direct added benefit not only to scientific growth, but to the improvement of our life on earth too. Early science fiction writers such as H. G. Wells in the War of the Worlds imagined an invasion on Earth by the Martians, which resulted in the advance of our scientific knowledge and technological advance. The need for design and construction of robotic applications in space results on the urge of creating the knowledge on the kinematics and dynamics that such models represent, and, of course, their successful guidance and control.

Under this scope, the Control Systems Laboratory (CSL), of the National Technical University of Athens (NTUA), has constructed a space emulator, which consists of two robots that simulate the function of robots in space, and which are under constant development. The last need that occurred was the modeling and control of the robotic emulator and its successful application. For this reason, the purpose of this work is to fully identify and develop the kinematic and dynamic model of the robotic emulator and manage to fully control it into performing basic tasks as would be required in a space environment, such as grasping free-flying objects or docking to other objects on orbit.

1.2 Literature Review

For centuries now has the human need for space exploration searched for answers of life on earth through observing the sky and trying to solve the mysteries of the universe that surrounds us. From the early astronomers in Mesopotamia and ancient Greece to today's space missions, the human thirst for exploration into deep space and the search for answers beyond our world, planet Earth, have yet not been fulfilled. Since the earliest days of astronomy, since the time of Galileo, astronomers have shared a single goal — to see more, see farther, see deeper [1].

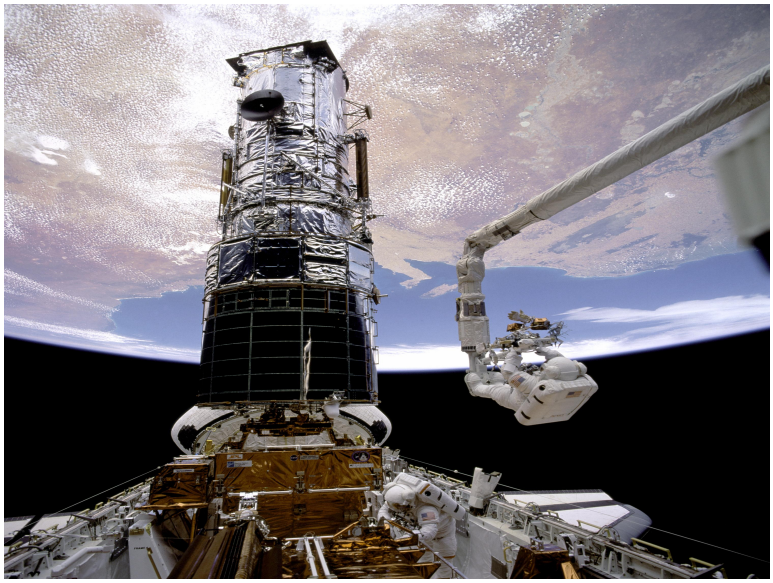


Figure 1.1: Hubble telescope undergoing astronaut repairs on first servicing mission. Credits: NASA

Stars gazing, looking for signs in the sky, humans endless search for knowledge and discovering the unknown. Human curiosity and the urge of scientific discovery has led to the vast development of science -figuring out how the world that surrounds us, works. When technology finally caught up science, humans managed to actually send objects and instruments into space that could collect and send data back to earth that would bring new scientific knowledge to light. The Hubble Space Telescope's launch in 1990 sped humanity to one of its greatest advances in that journey. Hubble is a telescope that orbits Earth. Its

position above the atmosphere, which distorts and blocks the light that reaches our planet, gives it a view of the universe that typically far surpasses that of ground-based telescopes. Before man could go to space, robotic applications were constructed in order to assist the effort of space exploration and expand the already existing knowledge about the universe. Sputnik 1 was the first robot in space, and was launched on October 4th, 1957 by the USSR. The Voyager missions are notable for the milestone of having a robot leave the Solar System. Voyager 1 and 2 were launched in 1977 are still making their way out of the Solar System, and have entered the heliopause, where the solar wind starts to drop off, and the interstellar wind picks up [2].

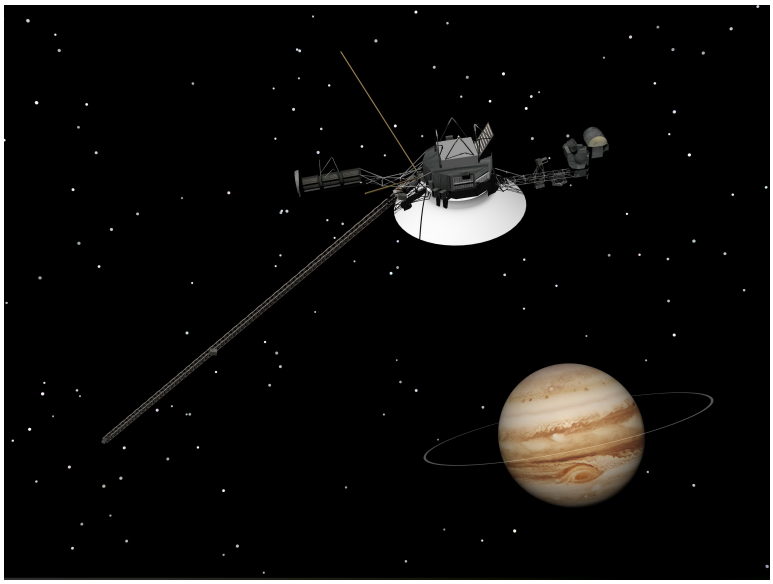


Figure 1.2: After visiting Jupiter and Saturn (while its twin Voyager 2 dropped by Uranus and Neptune), Voyager 1 spurned Pluto to visit Saturn's massive moon Titan.

The most famous robots in space have to be the series of orbiters, rovers and landers that have been sent to Mars. The first orbiter was Mariner 4, which flew past Mars on July 14, 1965 and took the first close up photos of another planet. The first landers were the Viking landers. Viking 1 landed July 20, 1976, and Viking 2 on September 3, 1976. Both landers were accompanied by orbiters that took photos and scientific data from above the planet. The landers included instruments to detect for life on the surface of Mars, but the data they returned was somewhat ambiguous, and the question of whether there is life on Mars still

requires an answer.

Currently, the Spirit and Opportunity are roving away on the Martian surface, which both landed on Mars surface in 2014, well past their expected mission lifetime, and have returned a wealth of information about the planet. The Phoenix lander descended on Mars on May 25, 2008 [3]. Mission scientists used instruments aboard the lander to search for environments suitable for microbial life on Mars, and to research the history of water there. The European Space Agency (ESA) currently has Mars Express orbiting the planet, and has the first webcam of another planet available.

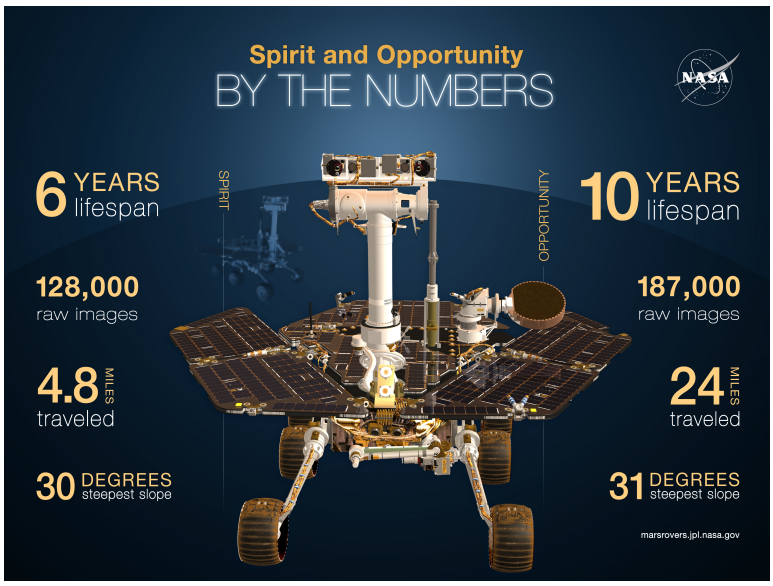


Figure 1.3: Spirit and Opportunity - NASA's rovers still roving Mars - Missions in numbers.

The ongoing 2020 ESA mission of the ExoMars [4] programme will deliver a European rover and a Russian surface platform to the surface of Mars. A Proton rocket will be used to launch the mission, which will arrive to Mars after a nine-month journey. The ExoMars rover will travel across the Martian surface to search for signs of life. It will collect samples with a drill and analyse them with next-generation instruments. The drill is designed to extract samples from various depths, down to a maximum of two metres. It includes an infrared spectrometer to characterise the mineralogy in the borehole. Once collected, a sample is delivered to the rover's analytical laboratory, which will perform min-

eralogical and chemistry determination investigations. Of special interest is the identification of organic substances. The rover is expected to travel several kilometres during its mission, and will be the first mission to combine the capability to move across the surface and to study Mars at depth. Moreover, The ExoMars Trace Gas Orbiter, part of the 2016 ExoMars mission, will support communications. The Rover Operations Control Centre (ROCC) will be located in Turin, Italy. The ROCC will monitor and control the ExoMars rover operations. Commands to the Rover will be transmitted through the Orbiter and the ESA space communications network operated at ESA's European Space Operations Centre (ESOC) [5].

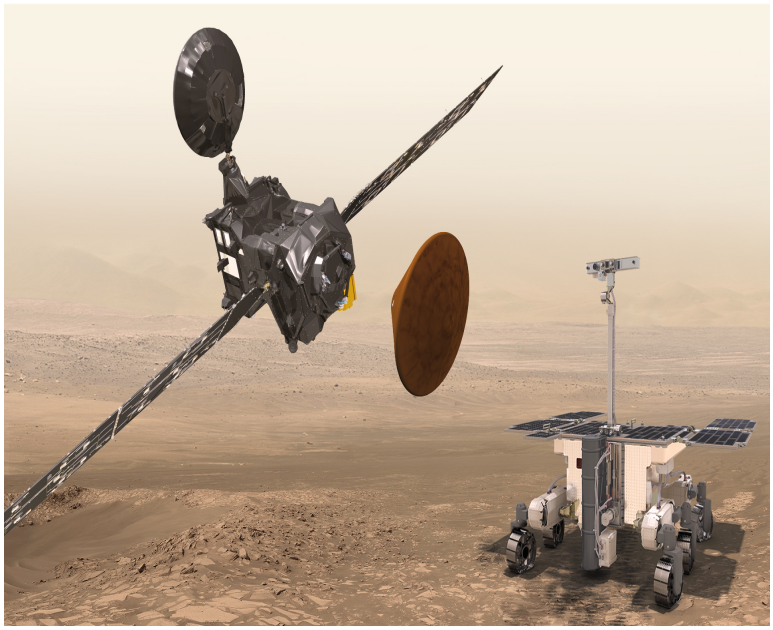


Figure 1.4: ESA's Exomars Mission: Orbiter and Lander.

The construction and the constant assembly of the International Space Station (ISS), is a remarkable demonstration of the benefits of space missions since it is an ongoing international cooperation amongst countries -United States, which through NASA, leads the ISS project, and 15 other countries involved in building and operating various parts of the station: Russia, Canada, Japan, Brazil, and 11 member nations of ESA (Belgium, Denmark, France, Germany, Italy, The Netherlands, Norway, Spain, Sweden, Switzerland, and the United Kingdom) [6]. Now essentially complete, the ISS has a pressurized living and working space approximately equivalent to the volume of a 747 jumbo-jet or a conventional

five-bedroom house, and can accommodate up to seven astronauts. It has a gymnasium, two bathrooms, and a bay window. The solar panels, spanning more than half an acre, supply 84 kilowatts. The ISS has been continuously occupied since November 2, 2000, and is visible, at times, in the night sky to the naked eye.

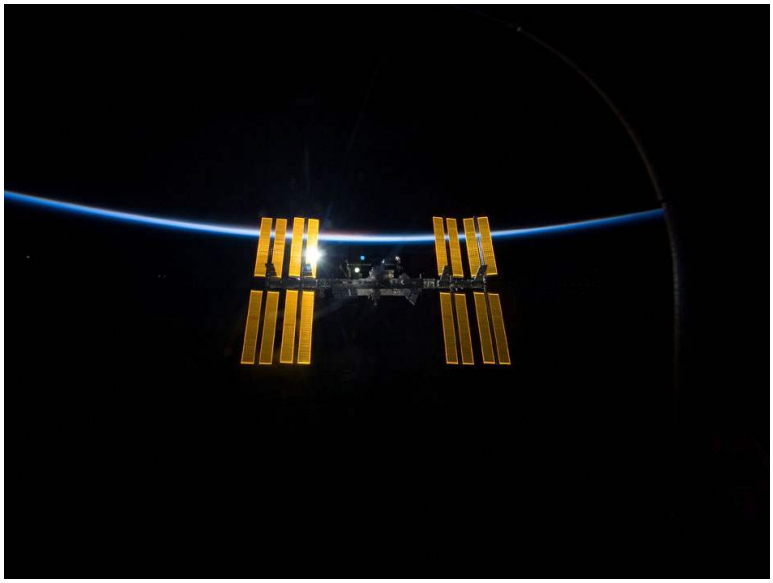


Figure 1.5: International Space Station On-Orbit Status 16 May 2016.

Robotic applications on board of the ISS have assisted the assembly of the ISS modules, as well as to various repairs, spacewalks and many other missions. The Shuttle's robotic arm, called Canadarm [7], has performed many kinds of tasks over the years. It has set satellites into orbit and retrieved others for repair. The first time Canadarm was used in one of the many ISS assembly missions was during Mission STS-88, December 1998. After the design and building of the arm, also known as the Shuttle Remote Manipulator System, the Canadarm continued its operation on board and wrapped up 30 years of successful operations when it was retired along with the Space Shuttle program after mission STS-135, which marked the robotic arm's 90th flight.

Now Canadarm2 [8] took its place on the ISS; it is a 17 metre-long robotic arm that assembled the ISS while in space. It is routinely used to move supplies, equipment and even astronauts. As well as supporting the Station's maintenance and upkeep, it is responsible for performing "cosmic catches," the capturing and

Figure 1.6: Canadarm being used for the repair of Hubble Telescope.

docking of unpiloted spacecraft that carry everything from science payloads to necessities for the 6-person crew on board the ISS.

The latest robotic addition on the ISS is Dextre, whose role is to perform maintenance work and repairs like changing batteries and replacing cameras outside the ISS. Having Dextre on call reduces the amount of risky spacewalks to do to routine chores, thus giving astronauts more time for science, the main goal of the ISS. Dextre's special skills and awesome location also offer a unique testing ground for new robotics concepts like servicing satellites in space. Dextre can ride on the end of Canadarm2 to move from worksite to worksite, or simply hitch a ride on the Mobile Base [9].

The knowledge created by the design and the on board assembly and use of the robotic arms has provided a significant added benefit to numerous applications on earth. The robotic technology used in Canadarm provides humanlike dexterity here on Earth in a variety of environments. These may include servicing nuclear power stations, welding and repairing pipelines on the ocean floor, remote servicing of utility power lines, or cleaning up radioactive and other hazardous wastes. An example is the Light Duty Utility Arm system, which was designed to inspect and analyze radioactive waste in underground storage tanks. This system consists of a modular, seven-joint manipulator attached to a tele-

Figure 1.7: Dextre and Canadarm2.

scopic vertical positioning mast. A mobile system deploys the manipulator in the tank. Remotely operated robotic systems have had wide application in industry and other fields. In medicine, they have aided the development of techniques involving robotic surgery operated from a remote location.

1.3 Contributions of this thesis

The work done and presented in this thesis shall provide a method of the full dynamic modeling of a space robot emulator with multiple manipulators (in this case 2). Planning and control have as an objective minimum fuel consumption, in scenarios of In-Space Robotic Servicing (ISRS) [10], like In-Space Maintenance [11] and In-Space Assembly [12], or space-junk removal [13].

1.4 Organisation of this thesis

The first chapter presents the space robot emulator of the CSL and describes its main function principles. The dynamic modeling analysis presented in the second chapter is followed by the third chapter, which presents the control equations and

the trajectory planning. The fourth chapter attempts an extended elaboration on the design and execution of the planer used in the Simulink model, which is presented right after, in Chapter 5, along with the simulation results and evaluation of the model. Chapter 6 tackles an approach to simulating the robot's response to the Simulink controller in a virtual simulation environment, Gazebo. Chapter 7 includes the conclusions drawn and the continuation capabilities of this work.

Chapter 2

The CSL Space Emulator

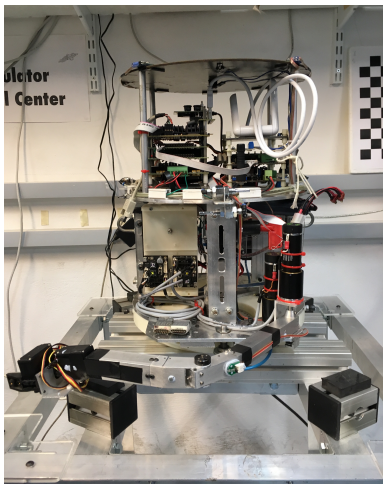
2.1 Testbed

The CSL space emulator consists of a low friction testbed in order to run zero-gravity (zero-g) and zero-friction experiments. For this purpose, a 2200 x 1800 x 300 mm granite table was procured, as shown in Figure 2.1, which ensures a flat surface (Class 0:<0.013mm) with very low roughness ($<\pm 5\mu$m –when new) is provided for experimental purposes.

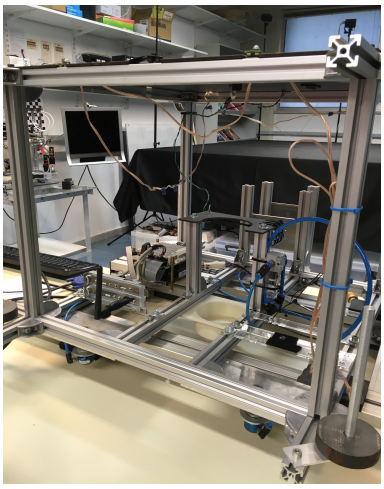
The granite table consists, along with robots equipped with air bearings [14], a testbed for various space-environment emulating experiments. The flatness of the table and its minimum roughness provide the zero friction element in conducting the experiments, and the function of the air bearings eliminate the gravity factor in order to simulate zero gravity conditions. The table deflection is negligible and its tilt after leg adjustment is less than 0.01°.

The table's surface finish was hand-polished and weighs approximately $3.5tn$, with density equal to $2.7 \cdot 10^{-3}$. Six slots are used for its lifting from the floor, each of them 900 mm tall. Three robots have been constructed by the CSL to be placed on the testbed -two active and one passive- which simulate the movement of robots in free-friction environments. There are currently two in use, Cepheus being the active (chaser) and Vanguard the passive robot (target).

Figure 2.1: The planar granite testbed.



(a) The Cepheus robot.

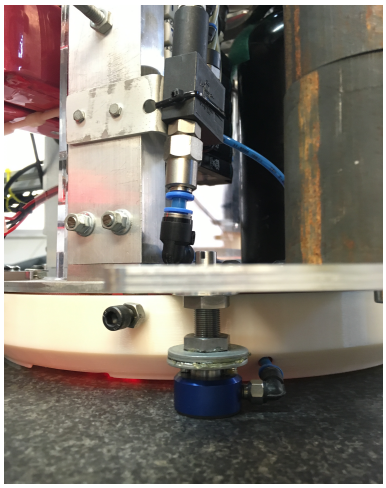


(b) The Vanguard passive robot.

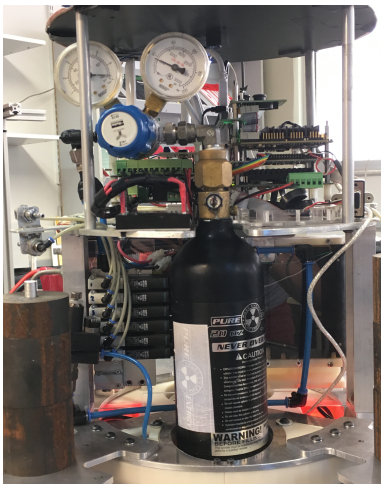
Figure 2.2: The CSL's robots: (a) Cepheus (active) & (b) Vanguard (passive) robots.

2.2 Eliminating friction

The latter is achieved with the use of air-bearings attached at the bottom of the robots, as shown in Figure 2.3(a), placed diametrically at 120 degrees angular distance from one another. The gas that flows through the porous surface of the air bearings forms a film underneath the robotic structures. This film creates a significant distance between the table and the robot bottom, which leads to the free-flow-like motion of the robots. To be noted that depending on the weight, the lift of the robots from the granite table varies. By increasing the weight of the robots, more pressure is to be distributed to the air bearings in order to lift the robots.



(a)



(b)

Figure 2.3: Air-bearring technology: (a) Mounting & (b) bottle and regulator.

2.2.1 Propellant

The gas flow is provided by a tubing system from a CO2 paintball bottle, of 50bars gas capacity, as shown in Figure 2.3(b).

Currently, 6.4 bars of CO2 gas are being used to lift the robots and thus achieving simulated movement in a free fall environment[1]. This is achieved by using a pressure regulator placed on top of the gas bottle. The same system is used to

[1]The condition of moving freely in an environment in which gravity, and nothing else, is causing acceleration.

provide gas flow to the air bearings system as well as to the thrusters used for the movement of the robot, as shown in Figure 2.4.

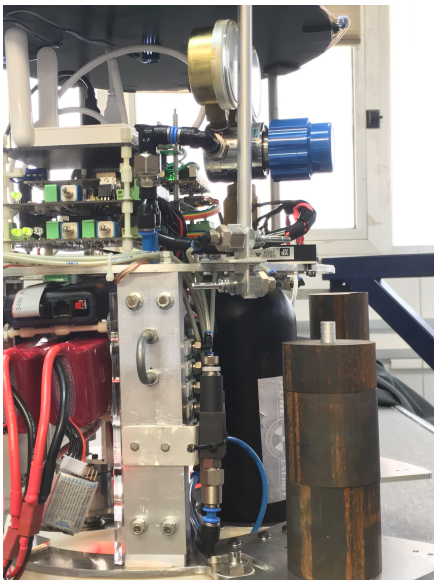


Figure 2.4: Pressure regulator and penumatics.

2.3 Robot System Motion

2.3.1 Thrusters

Three sets of thrusters are placed on the robot to allow translational movement on the table. The thrusters are set to operate at 7 bars each. Depending on the direction, one thruster of each pair is activated in order to reach the desired position given by the controller. The thrusters are placed at 120 degrees distance from each other diametrically on the robot, as shown in Figure 2.5.

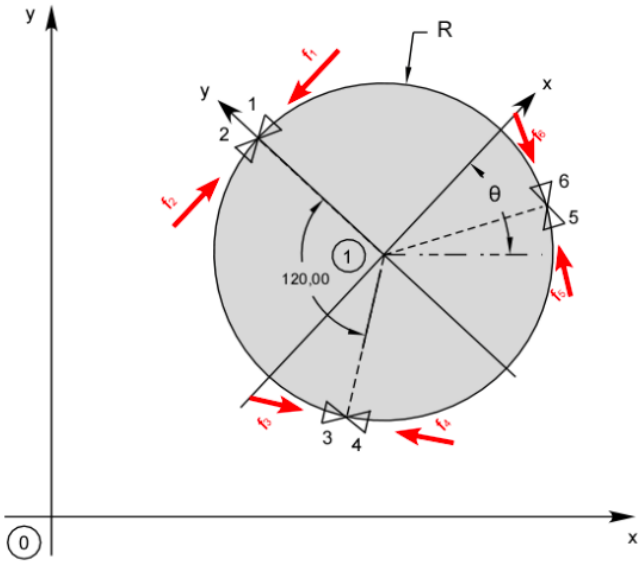


Figure 2.5: The thrusters geometry.

2.3.1.1 Operation

Thrusters serve many purposes in space propulsion. They are used for rocket launching, as shown in Figure 2.6, during the separation stages for reaching the orbit altitude, and being set into orbit as in Figure 2.7. Moreover, they are widely used among spacecrafts, from small satellites e.g. nanosatellites [15], to the International Space Station (ISS), for maneuvering and orbit correction, as shown in Figure 2.8.

The function of thrusters on spacecraft is based on the principle of conservation of momentum. They produce high-pressure gases which leave the engine (in the case of spacecraft, where the fuel is usually hypergolic and ignites on contact; in our case, the CO2 escaping of the thrusters) and cause the body to which they are attached to move in the direction opposite to the gas velocity.

Thrusters provide linear thrust in the direction opposite to the nozzle. If the thruster is directly connected to the Grid a technical term for any independent collection of blocks that form together any ship or station) it will transfer momentum from the ejected gas, to the Grid. Any component of the thrust vector which is not aligned with spacecraft Centre of Mass (COM) will apply a torque and impart angular momentum on the spacecraft. The use of thrusters for the

Figure 2.6: Thrusters used for rocket propulsion : Galileo Satellites Launch.

Figure 2.7: Thrusters during stage separation : Galileo second stage separation.

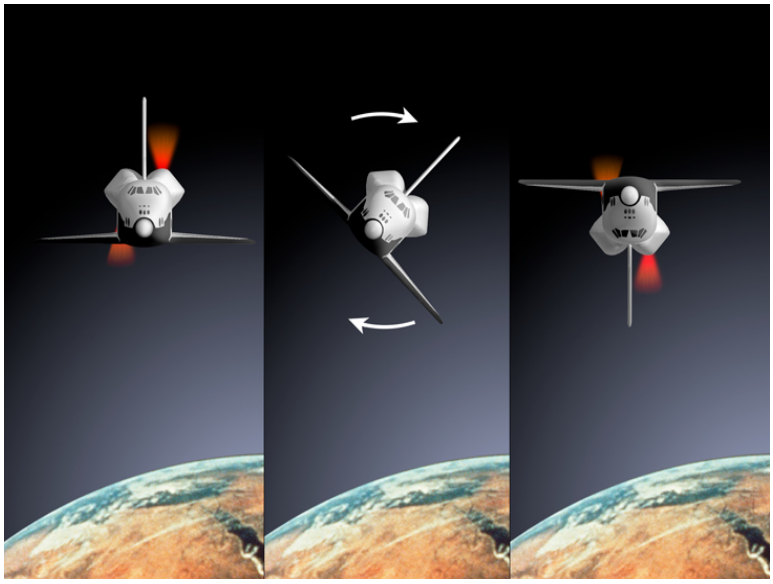


Figure 2.8: Thrusters for spacecraft maneuvering

Figure 2.9: ATV-3 thrusters for propulsion.

needs of propulsion of the ATV-3 (Automated Transfer Vehicle) is shown in figure 2.9.

This is explained with the conservation of momentum principle. As shown in (2.1), the momentum lost through the mass of high-velocity gas escaping the system, results in the movement of the object in the opposite direction in order to preserve the total momentum of the system, as no external forces are applied.

$$\begin{aligned} \boldsymbol{p}(t_1) &= \boldsymbol{p}(t_2) \\ \Rightarrow \boldsymbol{p}_{sc_1} &= \boldsymbol{p}_{sc_2} + \boldsymbol{p}_e \cdot n \\ &= \mathrm{p}_{sc_2} - \mathrm{p}_e \cdot n \\ \Rightarrow \mathrm{p}_{sc_2} &= \mathrm{p}_{sc_1} + \mathrm{p}_e \cdot n \end{aligned} \tag{2.1}$$

where $\boldsymbol{p}(t_1), \boldsymbol{p}(t_2)$ is the total momentum of the spacecraft before and after the thrusters ignition respectively, p_e is the momentum of each thruster exhaust and n is the number of the thrusters (assuming they all produce the same amount of thrust, in the same direction and with the same effectiveness rate), p_{sc_1}, p_{sc_2} being the directional momentum of the spacecraft before and after the thrusters' ignition respectively, equal and opposite the total momentum produced by the thrusters.

2.3.1.2 Cepheus thrusters

The thruster forces acting on the robot result in the movement of Cepheus, as a result of the CO2 expansion taking place outside the thruster nozzle. Due to the limited gas supply of each nozzle, it is important to monitor the supply of each one, as well as the thrust generated from each expansion.

2.3.1.2.1 Thrust & Mass Flow

The CO2 flow in the nozzles can be considered as an adiabatic flow of perfect gas, since the valve (which controls the gas flow) is open for very short time hence no heat exchange occurs between the environment, the CO2 gas and the connected nozzle. The flow is approached as a FANNO type flow, considering the constant diameter of the nozzles and the presence of friction in the gas route throughout them.

The total CO2 gas pressure at the nozzle inlet is 7 bars (equal to the pressure of the first pressure regulator) and its total temperature, which is related to the fluid energy, is equal to the temperature inside the CO2 bottle. Since the latter is in thermal equilibrium with the environment, a temperature equal to 20deg Celsius (293K) is assumed. Therefore the 2 following equations apply, the FANNO flow equation and the conservation of mass flow equation, as shown below in 2.2, 2.3 and 2.4:

$$\zeta \frac{L_{max}}{D_h} = \frac{1 - M^2}{\gamma M^2} + \frac{\gamma + 1}{2\gamma} \ln \left[\frac{(\gamma + 1) M^2}{2(1 + \frac{\gamma+1}{2}) M^2} \right] \tag{2.2}$$

$$\zeta \frac{L}{D_h} = \left(\frac{\zeta L_{max}}{D_h} \right)_{M=M1} - \left(\frac{\zeta L_{max}}{D_h} \right)_{M=M2} \tag{2.3}$$

$$\dot{m} = A \sqrt{\frac{\gamma}{R_g}} \frac{p_t}{\sqrt{T_t}} \frac{M}{\left(\frac{(\gamma+1)M^2}{1+\frac{\gamma-1}{2}M^2} \right)^{\frac{\gamma+1}{2(\gamma-1)}}} \tag{2.4}$$

where:

- M is the Mach number (M1 corresponds to the nozzle inlet and M2 to the nozzle outlet)
- D_h is the hydraulic diameter of the nozzle
- Z is the friction coefficient between the CO2 gas and the nozzle material
- $\dot{m}$ if the gas mass flow through the nozzle
- A is the area of the nozzle outlet tip
- γ, R_g are the CO2 gas constants
- p_t, T_t are the total pressure and Temperature of the CO2 gas, respectively

Both (2.3) and (2.4) are represented in Fig. 2.10.

The horizontal axis represents the current Mach number M_1 at the nozzle inlet while the lateral the Mach number M_2 at the nozzle outlet. The length of the nozzle is equal to $L = 6mm$, its diameter to $D = 0.9mm$ and a friction coefficient was chosen as a typical value from the Moody diagram, equal to $\zeta = 0.02$. The red curve represents the FANNO flow equation, while the green curves the mass flow conservation equation, for various environmental pressure values. The solution derives from the section points of the red curve with the green curves depending the environmental pressure. Taking under consideration the 2nd Axiom of Thermodynamics, and the FANNO curve, it is concluded that

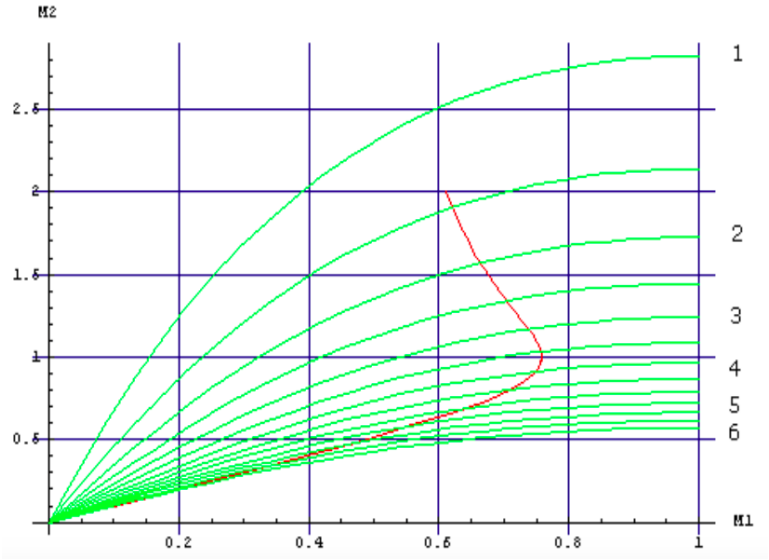


Figure 2.10: Graphical representation of the FANNO flow equation and the mass flow conservation equation at the nozzle exit.

for environmental pressure equal to $Pa = 1bar$, the fluid (CO2 gas) enters the nozzle with [16]

$$M_1 = 0.76 \tag{2.5}$$

and exits with

$$M_2 = 1 \tag{2.6}$$

with pressure p_2 given by Fig. 2.10 as:

$$p_2 = 3.65bar \tag{2.7}$$

Therefore, as it emerges from the mass supply with

$$\dot{m} = 1.2gr/s \tag{2.8}$$

the thrust generated due the CO2 gas expansion is calculated as:

$$\Omega = A(p_2 - p_a) + \dot{m}(u_{gas} - u_R) \tag{2.9}$$

where u_{gas} is the relevant exit velocity of the CO2 to the nozzle, A the area of the nozzle exit tube and u_R the velocity of the robot. Taking under consideration that u_{gas} is significantly greater than u_R and equal to:

$$u_{gas} = 248m/s \gg u_R \Rightarrow \Omega = 0.66N \tag{2.10}$$

The conclusions of the above calculations are shown in Table 2.1.

Table 2.1: Calculated Values - Thrusters - Analytical Solution.

Name	Value
Environmental Pressure	$p_a = 1bar$
CO_2 Pressure at nozzle outlet	$p_2 = 3.65bar$
Mach number at nozzle inlet	$M_1 = 0.76$
Mach number at nozzle outlet	$M_2 = 1$
Nozzle mass supply	$\dot{m} = 1.2g/s$
Nozzle generated Trust	$\Omega = 0.66N$
CO_2 velocity at nozzle outlet	$u_{gas} = 248m/s$

2.3.1.2.2 Values Comparison

It was calculated experimentally that the Thrust generated by the nozzles is equal to $\Omega_{exp} = 0.52N$. The divergence of the two values is calculated as (2.11):

$$e_{\Omega_{real}} = \frac{(\Omega_{exp} - \Omega_{th})}{\Omega_{th}} = 21.73\% \tag{2.11}$$

Calculating the divergence between the theoretically calculated mass flow and the experimentally measured value, equal to $\dot{m}_{exp} = 1.53g/s$ (2.12):

$$e_{\dot{m}_{real}} = \frac{(\dot{m}_{exp} - \dot{m}_{th})}{\dot{m}_{th}} = 27.5\% \tag{2.12}$$

It is concluded that the analytical calculations predicted with sufficient accuracy the thrust and mass flow, which is shown by the small value of the absolute error.

2.3.2 Reaction Wheel

2.3.2.1 Motivation

Since the CO2 bottles have a limited gas capacity, they were refilled repetitively during the execution of experiments. The process is costly and time consuming

thus another system has to be used in order to minimise the gas consumption of the thrusters. This would allow:

(a) longer exception of experiments

(b) maximum gas flow to the air bearings

(c) greater controllability and stability and

(d) redundancy for the system.

Therefore it was decided that a Reaction Wheel (RW) [17] shall be added to the system, in reliance with the thrusters. The RW is mounted at the bottom of the robot and parallel to the table, as shown in Figure 2.11.

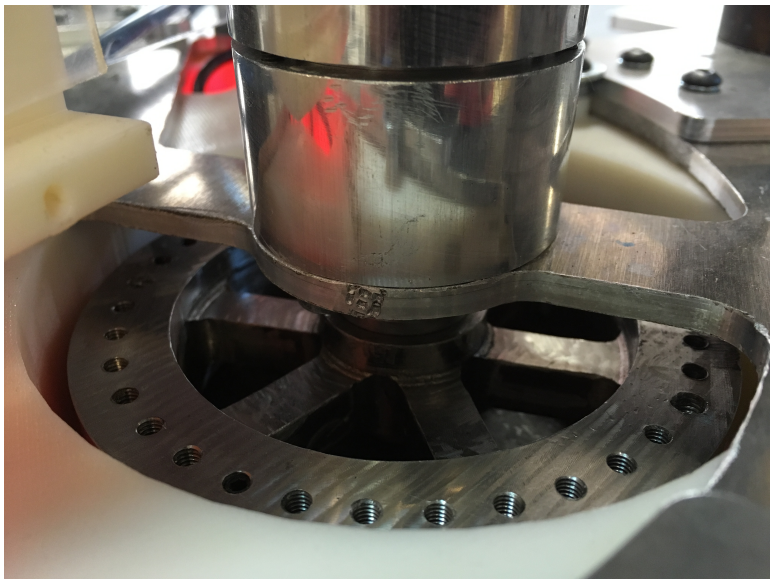


Figure 2.11: The Reaction Wheel mounted on the robot.

2.3.2.2 Operation

The function of reaction wheels makes them ideal for use in spacecraft for attitude control. Their operation is based on the principle of conservation of angular momentum. This is accomplished by attaching an electric motor (Figure 2.12) to a flywheel which, when its rotation speed is changed, causes the spacecraft to begin to counter-rotate proportionately due to the conservation of angular

momentum. Reaction wheels can only rotate a spacecraft around its COM; they are not capable of generating translational movement.

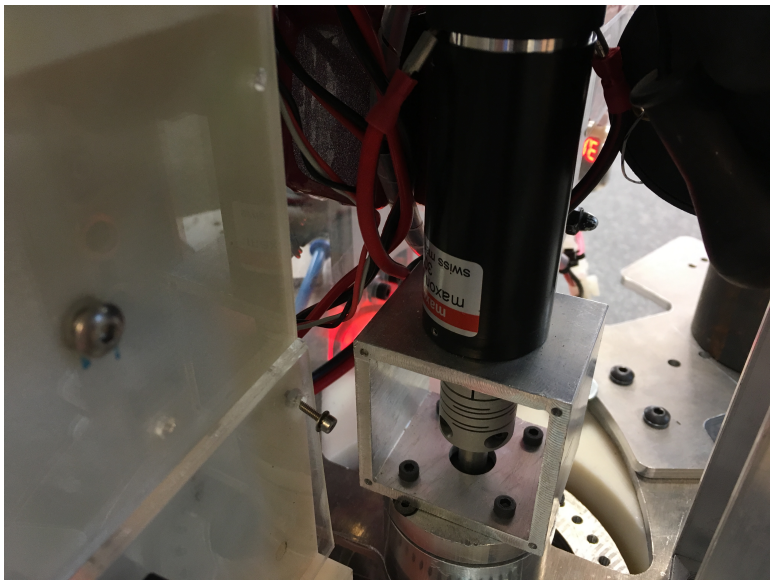


Figure 2.12: The maxon motor that rotates the RW.

As described in 2.13, momentum exchange takes place when necessary through the function of the RW, in order for the total momentum of the system to remain constant (conservation of momentum):

$$H_S(t_1) = H_S(t_2) \tag{2.13}$$

The operation of the electric motor alters the system balance. The torque produced by the motor T_{RW} acts to change the rotational speed of its axis ω_{RW}, on which the flywheel is mounted. The angular momentum of the RW has a change rate equal to the motor torque (2.14):

$$\begin{aligned} \Sigma M &= \frac{dH}{dt} \\ \Rightarrow T_{RW} &= \frac{dH_{RW}}{dt} \end{aligned} \tag{2.14}$$

The angular momentum of the robot H_{RB_2} is then equal to the sum of the initial system momentum value H_{RB_1} and the extra momentum ΔH_{RB} needed

for the conservation of the total angular momentum of the system robot - RW (2.15):

$$H_{RB_2} = H_{RB_1} + \Delta H_{RB} \tag{2.15}$$

The initial and final values of the momentum of the system are defined in eq. 2.16 and 2.18, while the momentum of the RW in eq. 2.20:

$$H_{SYS_1} = H_{RB_1} \tag{2.16}$$
$$H_{SYS_2} = H_{RB_2} + H_{RW} \tag{2.17}$$
$$= (H_{RB_1} + \Delta RB) + H_{RW} \tag{2.18}$$
$$2.16 \Rightarrow H_{RB_1} = (H_{RB_1} + \Delta RB) + H_{RW} \tag{2.19}$$
$$\Rightarrow \Delta H_{RB} = -H_{RW} \tag{2.20}$$

As a result, the extra amount of momentum is equal in value to and opposite in direction to the contribution of the RW in order to conserve the total momentum (eq. 2.21)

$$\begin{aligned} H_1 &= H_2 \\ H_{robot} &= H_{robot} + \Delta H_{robot} + H_{RW} \end{aligned} \tag{2.21}$$

Where H_1 and H_2 the value of the total momentum of the system in time $t = t_1$ and $t = t_2$, respectively.

2.3.2.3 Balancing

Due to Cepheus' role as a space robot simulator, a high level of accuracy is expected, thus it was necessary to ensure any possible part failure was eliminated. In particular, the mounting of the RW to the robot body requires high accuracy in order for it to function properly -avoiding lateral forces and introducing momentum to the system in axes other than the z-axis.

The ball bearing is to retain the stabilisation of the RW during its spin, however manufacturing unbalance could endanger the system's stability. For this reason a balancing procedure for the RW wheel took place, in order to identify the unbalancing points. As a forethought, holes were manufactured along the wheel's perimeter, with the provision of adding extra masses so as to balance the flywheel. The experiment was executed at 1800 rpm (maximum speed of the balancing machine, whereas it should have been tested in 4774 rpm, which is the maximum speed of the RW) and 0.2g and 0.03g unbalance was found at 44 deg and 94 deg respectively, and were corrected with the addition of the correcting masses.

2.3.2.4 Assembly

The reaction wheel consists of a motor and a driver, coupled with a steel flywheel, as shown in Figure 2.13. Moreover, a double row ball bearing was chosen to carry

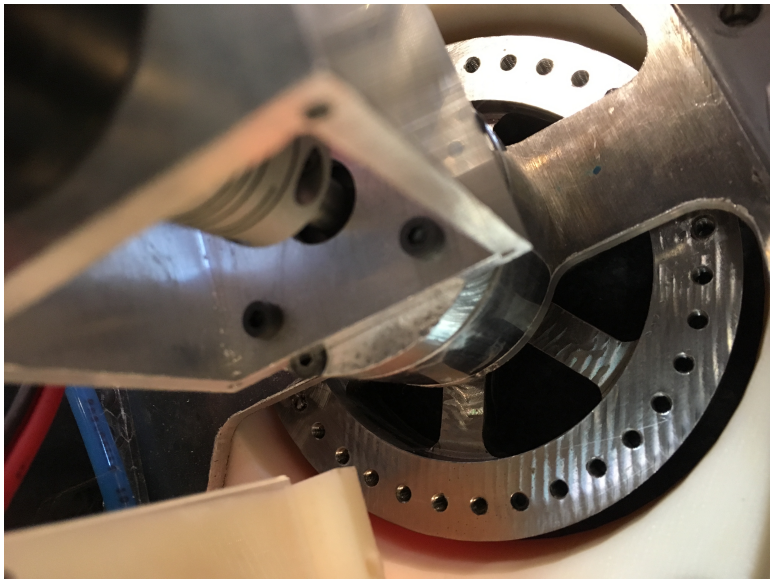


Figure 2.13: The RW coupled with the maxon motor.

Figure 2.14: Double row ball bearing.

the load, which offers considerably higher load carrying capacity than single row bearings, shown in Figure 2.14. Moreover, the latter offers both lateral and angular balancing of the reaction wheel.

2.3.2.4.1 Parts

The RW consists of the following parts assembly:

1. Electronics:
    - Maxon Motor RE 30 268216 Ø30 mm, Graphite Brushes, 60 Watt
    - Nominal torque (max. continuous torque): 88.2 mNm
    - Max. Efficiency: 88%
    - 4-Q-DC Servo Amplifier ADS 50/5 (driver)
2. Mechanical Systems:
    - Misumi Slit Coupler
    - SKF Angular Contact Ball Bearing double row
    - Flywheel (constructed by CSL):
        - Material: Steel [18]
        - Diameter: 110mm
        - Thickness: 24mm

2.4 Power

The system is powered by two 14.8V four-cell (4S) LiPo batteries [19], as shown in Figure 2.15, and can deliver 27.2-33.6V (3.4x8-4.2x8 V) to the system. Since the voltage delivered from the batteries varies, a DC/DC converter is used as a regulator, set to deliver 24V to the system through the Power Board.

The Power Board then supplies 24V to the robot's computer and high power electronics (valves, motor driver) through 2 manual switches. The board also features a remote shut-off operation for the high power electronics, which acts as an emergency button. The on-board computer consists of a 3-set stack of PC104. The Power supply unit, the single board computer and a I/O card for interfacing the robot's hardware, while the thrusters are operated through

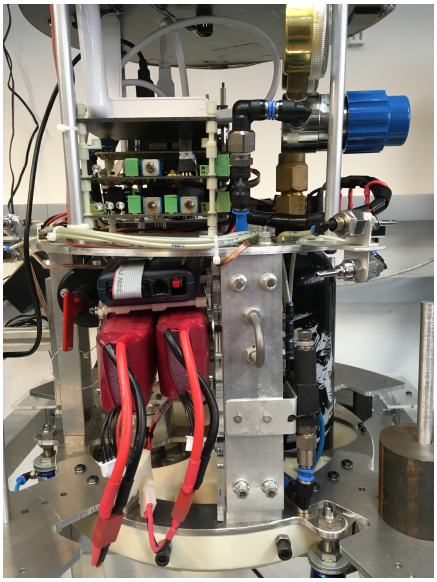


Figure 2.15: The robot's power system.

switching on and off one of the 6 valves which are controlled from the thrusters amplifier card, converting the 0-5V signals from the I/O card to 0-24V.

2.5 Control

The system is Linux-based and running on Robot Operating System (ROS). The robot's computer controls the function of the thrusters, the rotation wheel through the Input/Output (I/O) board, and acts as a host for the USB camera and force-sensor.

The code embedded in the layers below is ROS enabled. This means that in these dedicated pieces of code, the communication is abstract and platform independent; standard message types are being used which support distributed computing over Local Area Network (LAN).

The control logic of the robot is split into three layers. The hardware layer includes all the hardware dedicated software that controls:

- the digital I/O pins responsible for the pneumatic valves &
- the reaction wheel current controller.

This layer also incorporates low level safety features that prevent higher level commands from damaging the hardware.

The middle layer consists of all the software that creates input data, interfaces with the sensors and the low level controllers. Some examples are the reaction wheel velocity/torque controller, the thrusters force to PWM duty cycle converter, the optical mouse's software that creates odometry data, the pose tracking of robot based on LED-known position and the robot camera software that produces the relative pose of robot and the target, based on aruco markers [20].

Finally the higher layer embodies the robot base controller and the *Base Planner*. The former takes the robot base's desired pose as input, as well as its velocity and acceleration in a known reference frame, then transforms this pose to the inertial reference frame. Then, based on the current robot's state, it produces the thrusters' forces and reaction wheel torque needed to eliminate the error between the desired and the current values. The base controller is a PD controller with increased weight on the reaction wheel torque, aiming to reduce the CO2 and electrical energy consumption from the on-board batteries. The *Base Planner* is the software which, depending on the chosen experiment, creates the desired poses of the robot in a known frequency. For example, in an approach testing experiment, the desired poses array is a range of positions with respect to the target reference frame, with gradually decreased distance.

Chapter 3

Dynamic Modeling

To fully comprehend the contribution of each subsystem, and system's overall behaviour, the construction of a dynamic model that describes a physical system and can adequately represent its behaviour is essential. It provides the fundamentals for analysing, as well as controlling, the system. The approach can be as detailed as the knowledge of the system's parameters allows, but not more detailed than what is required.

The theoretical solution and the simulated results of their application could represent the real response to stimulants (external forces) depending on the mathematical approach chosen, the assumptions made, the number of the known parameters and those chosen to be omitted. For the dynamic modelling of the robot being examined in this paper the Euler - Lagrange Systems Dynamic Modelling method is applied, and is described next.

3.1 The Euler-Lagrange Method

The method used for the system analysis was the Euler-Lagrange. The dynamic equations of the system under analysis are given by (3.1):

$$\frac{d}{dt}\Big(\frac{\partial L}{\partial \dot{q_f}}\Big) - \frac{\partial L}{\partial q_f} = Q_f \tag{3.1}$$

Where:

- L = T - V : the Lagrange term

- T : the Kinetic Energy of the System
- V : the Dynamic Energy of the System, meaning the work produced by the conservative forces (e.g. weight, spring forces)
- Q_f : the generalised forces and torques, that are applied by the actuators and by the external non-conservative forces and torques acting on the system
- q_f : the generalised coordinates, which is equal to the minimum number of the system's variables state = $[q_f, \dot{q}_f]$ that can at any moment, uniquely fully describe its position.

As for the generalised coordinates, its definition implies that the ought to be:

- (mathematically) independent to each other
- in number, equal to the number of the independent Degrees of Freedom (DOF)
- sufficient to define the orientation of the position of each part of the system at any given moment
- holonomic so as the description of the system shall not require knowledge of previous states.

The chosen generalized coordinates shall meet the above requirements in order to apply the Euler-Lagrange method as a means of analysing and describing the system. If more than one set of variables meet the requirements, then the set is chosen based on which one best serves the purposes of the study.

3.1.1 Generalised Torques and Forces

The generalised torques and forces, Q_f, acting on the system, are defined by the method of the virtual work of the non-conservative torques and forces acting on the system δW_j, which cause the virtual displacements δq_j (marginal shifts of the chosen generalised coordinates, q_j).

If Q is a generalised force applied to the system, then the virtual work done by Q acting along a virtual displacement δq is given by

$$\delta W_j = Q_j \cdot \delta q_j. \tag{3.2}$$

Therefore, a systematic approach to apply the Euler-Lagrange method in order to create the dynamic model of the space robotic emulator arises. It shall be applied

as: defining the system's DOF, choosing the appropriate generalised coordinates, calculating the T and V terms, calculating the left part of the Lagrange equation for every q_j and vectoring the result, calculating the generalised torques and forces.

3.2 The dynamic model of the robot

A definition of the robot's description and variables taken into consideration in the dynamic analysis shall be presented, prior to the modelling of the space robot emulator.

It shall be stated that no modelling of any friction forces applied to the system takes place, nor the elasticity of the belts of the arms. Therefore, no friction nor elasticity terms are part of the generalised forces values of the Lagrange equation. Moreover, the time delay of the sensors' signals are not taken into account. The delay of the Incremental Encoders and of the optical sensors (type PC mouse) are also treated as negligible.

3.2.1 Modeling the physical system

The robot is modeled as a circular base with two actuated arms attached on it, as shown in Figure 3.1.

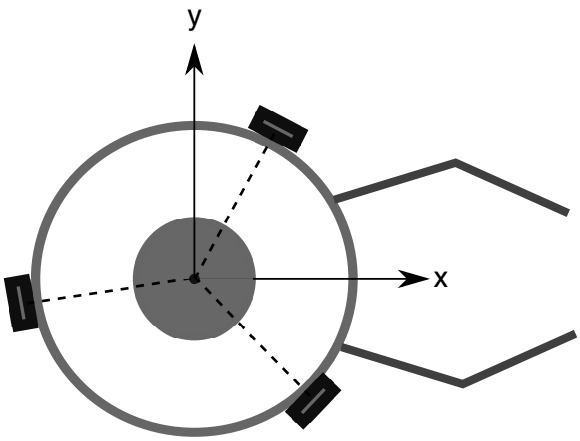


Figure 3.1: The robot model

Three thrusters are placed diametrically along the circumference, with 120°distance -the circle centre being the Geometrical Centre (GC) of the circle, as shown in Figure 3.2. Each thruster generates force equal and opposite to the thrust created by the pulse escape of the CO2 gas from the thrusters nozzle. Therefore the thrusters come in pairs of two. One thruster out of the two operates at each time, depending on the directional need for the robot.

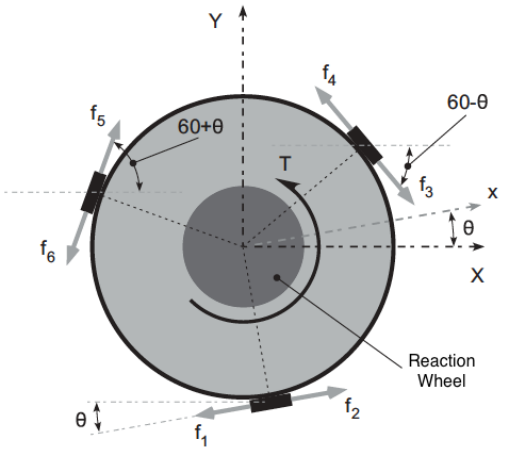


Figure 3.2: The robot base model

The RW is placed near the centre of the robot. The torque generated by its operation applies a torque on the robot, with direction opposite to the RW's. Therefore,

$$T_{act} = -T_{RW}. \tag{3.3}$$

The movement of the arms in a zero-friction environment causes the robot to move according to the torque applied for the movement. When an arm moves, the motion requires a torque equal to:

$$\boldsymbol{\tau} = \boldsymbol{r} \times \boldsymbol{f} \tag{3.4}$$

where $\boldsymbol{r}$ the distance between the system's COM and the arm's COM.

The forces acting on the End Effector (EE) of the arms are also being modelled. Based on the same principle, if an external force acts on the EE of one or both the arms simultaneously, a torque is generated, equal to the cross product of the acting force and the distance to the base Centre of Mass (COM):

$$T_C = \boldsymbol{r_1} \times \boldsymbol{f_1} + \tau_1 \tag{3.5}$$

The dynamic modelling approach uses the system's COM as the centre of movement (and not the GC of the base). Therefore, all the forces acting to the system are transferred to the system's COM, as shown in Figure 3.3.

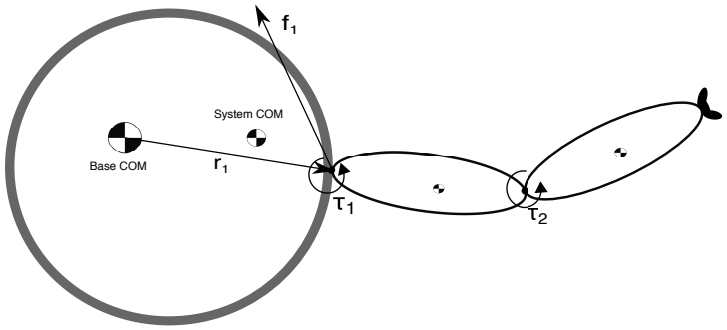


Figure 3.3: The acting force on the robot's COM

Before moving on with the calculation of the result of the forces acting on the robot, we shall first define the robot's Degrees of Freedom (DOF). Essentially, this is a definition of the robot's ability to move, on a 2D plane.

3.2.1.1 Degrees of Freedom

Cepheus has 7 DDOF, actuated by the robot's actuators, which are the three pairs of thrusters, the RW and the four arms' motors. The thrusters provide motion on the xy (planar) and around the z axis (combination of thrusters to create circular motion) and the RW can generate a rotation, thus motion around the z axis, still on the 2D plane. The arms have the ability of angular motion, hence each joint has one degree of freedom. The presence of 4 manupulatior degrees of freedom sums up to 7 DOF for the robot in total, along with the planar displacement and rotation of the base, where GC denotes the robot's base geometrical centre.

3.2.1.2 Generalized Coordinates

Proper designation of a system's generalized coordinates denote the full modeling of its behaviour.

The generalized coordinates, in respect to the absolute system (X,Y), can therefore be expressed as a vector q:

$$q = \begin{bmatrix} x \\ y \\ \theta \\ q_{11} \\ q_{12} \\ q_{21} \\ q_{22} \end{bmatrix} \tag{3.6}$$

as shown in Figure 3.4:

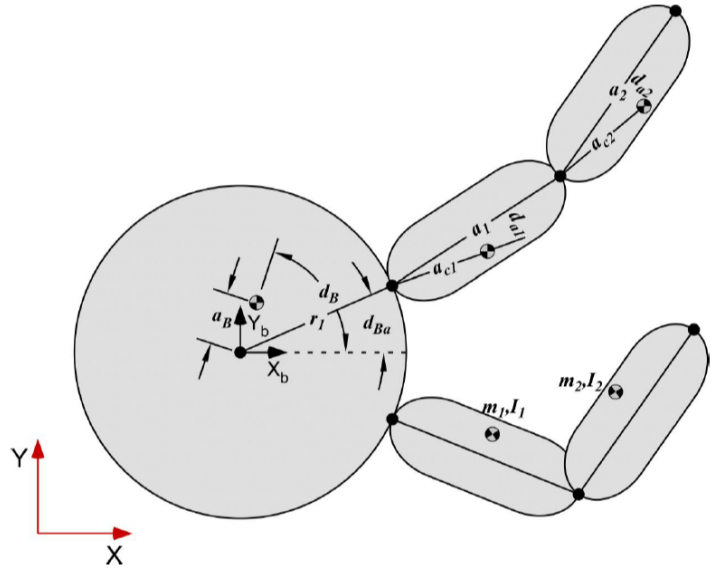


Figure 3.4: The generalized coordinates of the system, in the inertial frame.

where x and y represent the system's displacement in the x-axis and y-axis respectively, in the inertial frame, and θ the latter's planar rotation, while q_{11} denotes the angular displacement of the first joint of the fist manipulator, q_{12} of the second joint of the first manipulator, q_{21} of the first joint of the second manipulator, and q_{22} of the second joint of the second maipulator. The values x_b, y_b indicate the position of the robot's base GC in the x-axis and y-axis respectively, in respect to the absolute coodrinate system (X,Y).

In order to properly and fully model the system, the actuator forces need to be expressed to a common point of reference, this chosen to be the system's COM. Implied by the system's geometry, below are shown the positions of the system's

actuators' COM, in reference to the system's COM.

$$x_{q_{11}} = x_b + ab \cdot cos(db + \theta) + r1 \cdot cos(dba + \theta) + ac1 \cdot cos(q_{11}) \tag{3.7}$$
$$y_{q_{11}} = y_b + ab \cdot sin(db + \theta) + r1 \cdot sin(dba + \theta) + ac1 \cdot sin(q_{11}) \tag{3.8}$$
$$x_{q_{12}} = x_b + ab \cdot cos(db + \theta) + r1 \cdot cos(dba + \theta) + ac1 \cdot cos(q_{11}) + ac2 \cdot cos(q_{12}) \tag{3.9}$$
$$y_{q_{12}} = y_b + ab \cdot sin(db + \theta) + r1 \cdot sin(dba + \theta) + ac1 \cdot sin(q_{11}) + ac2 \cdot sin(q_{12}) \tag{3.10}$$
$$x_{q_{21}} = x_b + ab \cdot cos(db + \theta) + r1 \cdot cos(dba + \theta) + ac1 \cdot cos(q_{21}) \tag{3.11}$$
$$y_{q_{21}} = y_b + ab \cdot sin(db + \theta) + r1 \cdot sin(dba + \theta) + ac1 \cdot sin(q_{21}) \tag{3.12}$$
$$x_{q_{22}} = x_b + ab \cdot cos(db + \theta) + r1 \cdot cos(dba + \theta) + ac1 \cdot cos(q_{21}) + ac2 \cdot cos(q_{22}) \tag{3.13}$$
$$y_{q_{22}} = y_b + ab \cdot sin(db + \theta) + r1 \cdot sin(dba + \theta) + ac1 \cdot sin(q_{21}) + ac2 \cdot sin(q_{22}) \tag{3.14}$$

The first index attests to the arm number -first being the top and second the bottom arm- while the second index attests the joint number -first being the actuator attached to the base and second the one attached to the end of the first arm.

3.2.1.2.1 Values Notation The analysis takes place in a 2-dimensional plane and as such, the distances shall represent the planar projections of the real-spacial distances. Due to the fact that the two arms are identical, the same notation is used to designate the corresponding values for both. The symbolism is in accordance to that depicted in Figure 3.4. Ensuingly are conferred the main main values symbolism used in the paper.

- $m_B = 13.776kg$ the mass of the main robot frame, consisting of the circular base and all the arm parts connected to the base, e.g. motors, etc.
- $I_B = 0.1312kgm^2$ the total, centre of mass, polar moment of Inertia of the main body of the robot
- $a_B = 0mm$ the distance between the COM of the system and the GC of the robot's body
- $d_B = 0^\circ$ the angle between the straight line connecting the COM of the main body of the robot and its geometrical centre, and the X axis of

the absolute coordinate frame. This angle is used to define the initial orientation of the robot - if not rotated, set to zero

- $d_{B_a} = 15^\circ$ the half value of the angle created by the triangle connecting the first joint of the first arm, the main body's COM and the first joint of the second arm (position of the arm joints in reference to the system's COM)
- $r_1 = 0.15mm$ the radius of the main body's circular frame; the length of the straight line connecting the geometrical centre of the base and the axis of the first link of the arm -the point connecting the first link to the main body frame
- $m_1 = 0.086kg$ the total mass of the first link of the arms
- $m_2 = 0.079kg$ the total mass of the second link of the arms
- $I_1 = 2 \cdot 10^{-4} kgm^2$ the total, centre-mass, polar moment of Inertia of the first link of the arms
- $I_2 = 2 \cdot 10^{-4} kgm^2$ the total, centre-mass, polar moment of Inertia of the second link of the arms
- $a_1 = 0.18mm$ the length of the first link of the arms, considered as a straight line connecting the first to the second joint of the arms
- $a_2 = 0.13mm$ the length of the second link of the arms, considered as a straight line connecting the second joint of the arms to the End Effector ($x, y_{E_{1,2}}$, i=1,2)
- $a_{c_1} = 0.09mm$ the distance between the COM of the first link of the arms and its mounting point on the robot's main base frame
- $a_{c_2} = 0.065mm$ the distance between the COM of the second link of the arms and the joint connecting the first to the second link
- $d_{a_1} = 3^\circ$ the angle defining the offset of the COM of the first link of the arms in respect to the link's centre line -angle between a_1 and a_{c_1}, as defined above
- $d_{a_2} = 2^\circ$ the angle defining the offset of the COM of the second link of the arms in respect to the link's centre line -angle between a_2 and a_{c_2}, as defined above
- $I_{ma} = 11.2 \cdot 10^{-7} kgm^2$ the moment of Inertia of the driver of each electric motor

- $I_t = 1 \cdot 10^{-5} kgm^2$ the total, centre-mass, polar moment of Inertia of the secondary link of the first joint of the arms, including the two gears it carries
- $n = 1.9011$ the reduction ratio of the planetary gears of the motion transmission system
- $\eta = 0.7$ the efficiency ratio of the planetary gears of the motion transmission system

3.2.1.2.2 End Effector

Amongst the primary goals of the project is to include the functionality of the arms' End Effector (EE) in the modelling, which introduces the ability to the system of receiving external disturbances, through the arms' EEs. It is therefore essential to determine the position of the end effector of each arm, as shown below:

$$x_{E_1} = x_b + ab \cdot cos(db + \theta) + r1 \cdot cos(dba + \theta) + a1 \cdot cos(q_{11}) + a2 \cdot cos(q_{12}) \tag{3.15}$$

$$y_{E_1} = y_b + ab \cdot sin(db + \theta) + r1 \cdot sin(dba + \theta) + a1 \cdot sin(q_{11}) + a2 \cdot sin(q_{12}) \tag{3.16}$$

$$x_{E_2} = x_b + ab \cdot cos(db + \theta) + r1 \cdot cos(dba + \theta) + a1 \cdot cos(q_{21}) + a2 \cdot cos(q_{22}) \tag{3.17}$$

$$y_{E_2} = y_b + ab \cdot sin(db + \theta) + r1 \cdot sin(dba + \theta) + a1 \cdot sin(q_{21}) + a2 \cdot sin(q_{22}) \tag{3.18}$$

The index E_1 designates the position of the end effector of the upper hand (index 1) and E_2 of the other arm (index 2).

3.3 Kinetic and Dynamic Energy

The calculation of the total energy of the body at any given moment is essential for the solving of the Euler-Lagrange equation, which is used for the dynamic modelling of the system. The Kinetic Energy is symbolised with the letter T and the Dynamic Energy with the letter D.

The Kinetic Energy of the robot consists of the individual Kinetic Energy of:

- the main robot body

- the Reaction Wheel
- the first link of each arm (11, 12) - 1i (i=1,2)
- the second link of each arm (21, 22) - 2i (i=1,2)
- the driver of both the joints motor in each arm (2 placed in each arm base, to actuate the 2 links, 4 in total)

summed to produce the T term in the Euler - Lagrange Equation, as shown in eq. (3.25):

$$T = K_B + K_{a_1} + K_{a_2} + K_{ma_1} + K_{ma_2} \tag{3.19}$$

where K_B the Kinetic Energy of the base, K_{a_1} of the first arm, K_{a_2} of the second arm, K_{ma_1} the Energy produced by the motor of the first arm and K_{ma_2} by the motor of the second arm.

It is to be noted that the Kinetic Energy produced by the operation of the timing belts used to transfer movement from the motors placed on the base of each arm to the arms' links will not be taken into consideration in the calculation of the Kinetic Energy of the system. This is due to the fact that the mass of the timing belts is considerably low due to its small mass.

The information contained in the generalized coordinates vector q, delineates the position and orientation of the robot, at any given moment. Hence by differentiating the generalized coordinates vector elements, can be calculated the velocity of the robot and thus its Kinetic Energy, T, at any given moment.

The Kinetic Energy of the main body of the robot (base) is described in eq. 3.20 below:

$$\begin{aligned} K_B &= \frac{1}{2} m_B V^2 + \frac{1}{2} I_B \omega^2 \\ &= \frac{1}{2} m_B (\sqrt{\dot{x}^2 + \dot{y}^2})^2 + \frac{1}{2} I_B \dot{\theta}^2 \\ &= \frac{1}{2} m_B (\dot{x}^2 + \dot{y}^2) + \frac{1}{2} I_B \dot{\theta}^2 \end{aligned} \tag{3.20}$$

By integrating the positions of the COM of each body contributing to the Total Kinetic Energy of the system, in reference to the absolute coordinate system, the Total Kinetic Energy can thus be calculated.

For the first link of each arm:

$$K_{a_{i1}} = \frac{1}{2} m_{i1} (\dot{x_{i1}}^2 + \dot{y_{i1}}^2) + \frac{1}{2} I_{i1} (\dot{\theta} + \dot{q_{i1}})^2 \tag{3.21}$$

For the second link of each arm:

$$K_{a_{i2}} = \frac{1}{2} m_{i2} (\dot{x_{i2}}^2 + \dot{y_{i2}}^2) + \frac{1}{2} I_{i2} (\dot{\theta} + (\dot{q_{i2}} - \dot{q_{i1}}))^2 \tag{3.22}$$

The calculation of the parameters consisting the above equations are calculated with the software provided in the package of Wolfram Mathematica.

The above Kinetic Energies consist the Kinetic Energies of each individual body. In order to define the Kinetic Energy of the system, in addition to the Kinetic Energies of each module separately (main body & arms) the effect of the arms' movement to the base thus to the system, shall be studied. Since the lack of friction as the study case, momentum transfer takes place with each movement, meaning that the momentum produced by the movement of a body placed in a finite distance from the main body, is transferred back to the system, leading to a change of each state and shall therefore be investigated.

The rotation of each link results from the function of the motors mounted on the base of the first arm. The output of the idler gears produces the values of the generalized coordinates $q_{11}, q_{12}, q_{21}, q_{22}$. Consequently, the rotation angle of each link in respect to the generalized coordinates can be described by the following equations:

$$\begin{aligned} \theta_{11} &= n.q_{11} \\ \theta_{21} &= n.q_{21} \\ \theta_{12} &= n(q_{12} - q_{11}) \\ \theta_{22} &= n(q_{22} - q_{21}) \end{aligned} \tag{3.23}$$

where n is the gear ratio, as defined in 3.2.1.2.1.

The values of the Kinetic Energy of the motors' drivers, Km_1 & Km_2, shall be fitly described by the following equations, using eq.3.23:

$$\begin{aligned} Km_{i1} &= \frac{1}{2} I_{ma_{i1}} (q_{i1}^2) \\ Km_{i2} &= \frac{1}{2} I_{ma_{i2}} (q_{i2}^2) \end{aligned} \tag{3.24}$$

Resultantly, the Total Kinetic Energy of the system, T, shall be described by eq. 3.25:

$$T = K_B + K_{a_{11}} + K_{a_{12}} + K_{a_{21}} + K_{a_{22}} + K_{ma_{11}} + K_{ma_{12}} + K_{ma_{21}} + K_{ma_{22}} \tag{3.25}$$

3.3.0.0.1 Dynamic Energy V

The Dynamic Energy of the system V is set to zero, because the system operates in a 2D space and gravity is perpendicular to the plane of motion. Moreover, the lack of elements capable of energy storing, sets the Total Energy of the system, L, equal to the Total Kinetic Energy of the system, T, neglecting belt compliance, as shown in eq.3.26:

$$L_{sys} = T + \cancelto{0}{V} \rightarrow L_{sys} = T \tag{3.26}$$

3.4 Generalised Torques and Forces

The External Generalised Torques and Forces acting on the system under study are the forces generated by the operation of the thrusters, $f_{1\rightarrow 6}$, 3 sets of thrusters, six in total- the torque generated by the RW, τ_m and the torques generated by the motor actuators of the arms, τ_{i1} & τ_{i2} (i = 1,2). It is to be noted that any other external force or torque, as well as friction, is not taken into account.

The acting Torques and Forces to the system consist the elements of the vector Q_{act}:

$$Q_{act} = \left[\begin{array}{c} f_1 \\ f_2 \\ f_3 \\ f_4 \\ f_5 \\ f_6 \\ \hline \tau_m \\ \hline \tau_{11} \\ \tau_{12} \\ \tau_{21} \\ \tau_{22} \end{array}\right] \tag{3.27}$$

The Forces and Toques Vector shall result in accelerations of the generalized variables, as defined in eq.3.6. Therefore the acting forces and torques vector shall result to a vector which is in reference of the generalized coordinates,

thus:

$$Q = \begin{bmatrix} f_x \\ f_y \\ \tau_\theta \\ \tau_{q11} \\ \tau_{q12} \\ \tau_{q21} \\ \tau_{q22} \end{bmatrix} \tag{3.28}$$

Note that Q_{act} is an [11x1] vector, while Q a [7x1] vector, same as the generalized coordinates'.

Hence the necessity of expressing the acting forces and torques vector, Q_{act}, in the form of $[Q]$ arises.

Firstly, it should be noted that for practical reasons, the thrusters are not modelled as six individual units but as three sets of thrusters, which can produce either positive or negative thrust. This happens because at the low level control, the output of controller software sends a value to the hardware, equal to the amount of thrust that shall be generated in a specific direction in order to meet the initial set requirements. This value has either positive or negative sign. With the same set of values, thence have 3^2 different commands.

Moreover, the torque included in the Q_{act} vector is equal to the acting force on the system, and thus, as explained in 2.3.2.2, the output of the torque generated by the RW, in the opposite direction. The torque to the system (base) is equal to [21]:

$$\begin{aligned} \tau_m &= K_t i_a \\ &= I_w \ddot{\theta_w} + b_w(\dot{\theta_w} - \dot{\theta_b}) \end{aligned} \tag{3.29}$$

where θ_w the absolute velocity of the wheel, and θ_b the absolute velocity of the base.

The torque on the base (if no other actuators act) is thus:

$$\begin{aligned} \tau_{RW} = -I_w \ddot{\theta_w} &= I_b \ddot{\theta_b} \\ &= -\tau_m + b_w(\dot{\theta_w} - \dot{\theta_b}) \\ &= -K_t i_a + b_w(\dot{\theta_w} - \dot{\theta_b}) \end{aligned} \tag{3.30}$$

The torque produced is less than the torque generated by the motor of the RW, as a result of the mechanic losses. The torque finally transferred from the RW

to the system is then:

$$\begin{aligned} \tau_{RW} &= -n_m^{RW}.\tau_m + b_w(\dot{\theta_w} - \dot{\theta_b}) \\ &= -n_m^{RW}.K_t i_a + b_w(\dot{\theta_w} - \dot{\theta_b}) \end{aligned} \tag{3.31}$$

where n_m^{RW} the mechanical efficiency rate of the RW.

As for the arms' actuators, the torque transferred to the system is equal to:

$${\tau_{ij}}^{eff} = n.\nu.\tau_{q_{ij}} \tag{3.32}$$

The Q_{act} vector hence becomes:

$${Q_{act}}^{eff} = \begin{bmatrix} f_{1,2} \\ f_{3,4} \\ f_{5,6} \\ \tau_{RW} \\ n.\nu.\tau_{q_{11}} \\ n.\nu.\tau_{q_{12}} \\ n.\nu.\tau_{q_{21}} \\ n.\nu.\tau_{q_{22}} \end{bmatrix} \tag{3.33}$$

which is a [8x1] vector. The Q_{act} is a vector containing elements whose values represent the ones of the system's actuators. Q_{act} should come to a [7x1] form, in order to be in accordance to the size of the generalized coordinates vector. Note that all the forces and torques are acting on the COM of the system.

3.4.0.0.1 The Jacobian Matrix Jacobian matrix is the fundamental quantity that describes all the 1rst-order planar qualities (length, angles) of interest, therefore, it is appropriate to focus the building of the forces and torques vector on the Jacobian matrix or the associated metric tensor [22]. The elements of this Jacobian matrix, J_{act}, are expressed in reference to the system's generalized coordinates, thus

$$J_{act} \rightarrow J_{act}(\overline{q}) \tag{3.34}$$

The Jacobian matrix contains information that relates the actuators' space to the general relative space, acting on the COM of the system, as shown below.

$$J_{act}(\overline{q}).Q_{act} = Q = {Q_{act}}^{eff} \tag{3.35}$$

The Jacobian matrix was calculated in Wolfram Mathematica and is included in the Appendix B at the end of this paper. The Jacobian J_{act} is a matrix of size [7x8], therefore when multiplied with the Q_{act} vector whose size is [8x1], the Q matrix is produced, with the correct size of [7x1].

3.5 Solving the Euler - Lagrange Equation

As presented in Chapter 3.1, the Euler-Lagrange equation:

$$\frac{d}{dt}\Big(\frac{\partial L}{\partial \dot{q}}\Big) - \frac{\partial L}{\partial q} = Q \tag{3.36}$$

requires the calculation of the Q_f vector, which contains the external generalized torques and forces acting on the system, which is represented from the Q vector, as described above. The left part of the Euler-Lagrange Equation needs further analysis, taking under consideration that the $V = 0$, as explained in Paragraph 3.3.0.0.1.

The left part of the Euler-Lagrange equation is hence analysed as (3.26):

$$\begin{aligned}\frac{d}{dt}\Big(\frac{\partial L}{\partial \dot{\overline{q}}}\Big) - \frac{\partial L}{\partial \overline{q}} = \frac{d}{dt}\Big(\frac{\partial T}{\partial \dot{\overline{q}}}\Big) - \cancel{\frac{d}{dt}\Big(\frac{\partial T}{\partial \dot{\overline{q}}}\Big)} - \frac{\partial T}{\partial \overline{q}} + \cancel{\frac{\partial V}{\partial \overline{q}}} \\ = \frac{d}{dt}\Big(\frac{\partial T}{\partial \dot{\overline{q}}}\Big) - \frac{\partial T}{\partial \overline{q}}\end{aligned} \tag{3.37}$$

The terms of the Euler-Lagrange Equation are being analytically calculated for each of the generalized coordinates $[xy\theta q_{11}q_{12}q_{21}q_{22}]$ so there are 7 equations in total; each corresponding to the solving of the Euler-Lagrange equation of a state variable. The terms are put into matrices and divided based on their dependencies from the acceleration of the generalized coordinates (vector) $\ddot{\overline{q}}$ only, and the ones from the velocity and displacement of each variable, $\dot{\overline{q}}$ and $\overline{q}$. The results produce the equation of motion for one generalized coordinate in a multibody system. The combination of the seven scalar equations lead to the vector form:

$$\frac{d}{dt}\Big(\frac{\partial T}{\partial \dot{\overline{q}}}\Big) - \frac{\partial T}{\partial \overline{q}} = M(\overline{q})\ddot{\overline{q}} + C(\overline{q}, \dot{\overline{q}}) \tag{3.38}$$

where $M(\overline{q})$ is the mass matrix, $C(\overline{q}, \dot{\overline{q}})$ is the Coriolis and centrifugal term of the equation of motion, and Q is the vector of generalized forces for all the degrees of freedom (DOFs) in the system [23]. **M** only depends on q and **C**

depends quadratically on $\dot{q}$. The mass-inertia matrix M is of size [7x7] while the non-linear terms matrix C of [7x1].

Through eq. (3.33) and (3.35) we then conclude to:

$$M(\overline{q})\ddot{\overline{q}} + C(\overline{q}, \dot{\overline{q}}) = J_{act}(\overline{q}).Qact = Q = \begin{bmatrix} f_x \\ f_y \\ \tau_\theta \\ \tau_{q11} \\ \tau_{q12} \\ \tau_{q21} \\ \tau_{q22} \end{bmatrix} \tag{3.39}$$

3.5.0.0.1 Forces at the end effector In order to keep the size of the generalised forces and torques vector, Qact -[11x1], reduced to [8x1]- as low as possible, the acting forces on the EE were not added to the Q_{act} vector, but treated individually and then integrated to the Qact vector.

The full generalised forces and torques vector would be:

$$\left[\begin{array}{c} f_x \\ f_y \\ \tau_\theta \\ \tau_{q11} \\ \tau_{q12} \\ \tau_{q21} \\ \tau_{q22} \\ \hline f_{E1_x} \\ f_{E1_y} \\ f_{E2_x} \\ f_{E2_y} \end{array}\right] \tag{3.40}$$

Therefore, the acting forces at the End Effector, Q_E, is treated as an individual [4x1] vector, as shown below:

$$Q_E = \begin{bmatrix} f_{E1_x} \\ f_{E1_y} \\ f_{E2_x} \\ f_{E2_y} \end{bmatrix} \tag{3.41}$$

The forces acting on the EE are transferred to the system's COM, consisting part of the Q matrix. The Jacobian matrix, J_E, contains this information. Therefore,

$$J_E(\overline{q}).Q_E = Q_{end} \tag{3.42}$$

The Jacobian J_E is a [7x4] matrix, since it contains information of each element of Q_E in reference to the generalized coordinates vector, $\overline{q}$ (7x1). The multiplication of the J_E matrix with the Q_E matrix $\rightarrow$ [7x4].[4x1] produces a [7x1] matrix, that shall be added to the Q matrix, to include the effect of the end effector to the dynamic behaviour of the system.

$$Q_{end} + Q = Q_F \tag{3.43}$$

Chapter 4

Control and Trajectory Planning

Following the modelling of the system and thus defining the equations that describe its dynamic behaviour in a 2-Dimensional plane, the attempt to control the robot shall take place. Manipulating the robot space emulator as a unit implies the control of each active or passive component and subsystem.

As defined by the states variable vector, Q_{act}:

$$Q_{act} = \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \\ f_5 \\ f_6 \\ \tau_m \\ \tau_{11} \\ \tau_{12} \\ \tau_{21} \\ \tau_{22} \end{bmatrix} \tag{4.1}$$

the actuators that are being modelled and their function shall be controlled are the Thrusters, the Reaction Wheel (RW) and the Motors of the arms. However, as explained in (3.33), the six thrusters are modelled as three sets of coupled

thrusters (with opposite direction). Therefore the acting Torques and Forces on the system are the following:

$$Q_{act}{}^{eff} = \begin{bmatrix} f_{1,2} \\ f_{3,4} \\ f_{5,6} \\ \tau_{RW} \\ n.\nu.\tau_{q_{11}} \\ n.\nu.\tau_{q_{12}} \\ n.\nu.\tau_{q_{21}} \\ n.\nu.\tau_{q_{22}} \end{bmatrix} \tag{4.2}$$

The controller, using the model of the system, which represents the system's reaction to external stimulants, calculates the acting Torques and Forces vector, $Q_{act}{}^{eff}$, based on the given command. It then transforms the $Q_{act}{}^{eff}$ vector to the corresponding Q_{act}, which is in accordance to the system's rank, and can be used to calculate the new state of the system -hence its state variables, as shown below:

$$M(\overline{q})\ddot{\overline{q}} + C(\overline{q}, \dot{\overline{q}}) = Q = \begin{bmatrix} f_x \\ f_y \\ \tau_\theta \\ \tau_{q11} \\ \tau_{q12} \\ \tau_{q21} \\ \tau_{q22} \end{bmatrix} \tag{4.3}$$

The transformation from the actuators' space to the model's takes place with the following equation, as explained in (3.35):

$$J_{act}(\overline{q}).Q_{act} = Q = Q_{act}{}^{eff} \tag{4.4}$$

4.1 Defining the High Level Objectives

Before proceeding to the description of the controller, the objectives shall first be set. The primary concern was the ability to manipulate the robot's full motion capabilities, including both the movement of the base and the arms. This means that the control command shall expect to move the robot's:

- Base attitude to a specific one (Senario 1)
- End Effector (EE) to a specific point in the planar space, and execute a specific task e.g. grasp another moving object -in orbit, near the robot's initial position, within the limits provided by the testbed's dimensions (Senario 2)

The Figure 4.1 below shows a servicing mission taking place, amongst two satellites, a smaller and a larger one, the servicer; the latter attempting to grasp the former with a robotic arm.

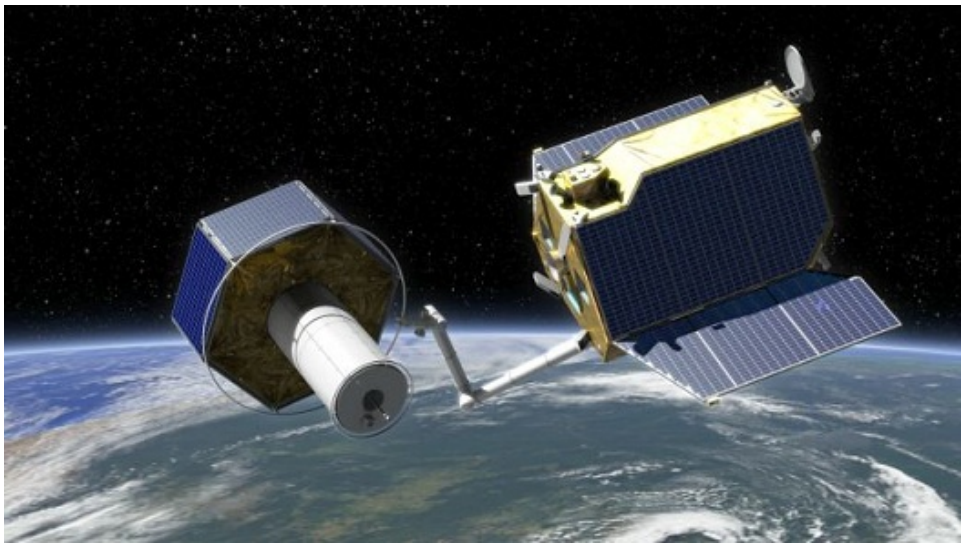


Figure 4.1: DEOS satellite servicing spacecraft. (Credit: Astrium).

4.2 The Control Equations

The method of non-linear model-based control is used, for applying Force/Torque control. Using the known dynamic model of the robot, the needed actuators' Forces and Torques can be constantly calculated, while the error produced in each interval -repetition of the calculations, until reaching the desirable state, are 'eliminated' by applying a model-based controller, in this case, a PD controller.

A PD Controller calculates the value of the acceleration of the state vector, $\ddot{\overline{q}}^*$, at each repetition, by taking under consideration the calculation of the calue of

the error between the desired position and velocity vector state, $\overline{q}_{des}$ and $\dot{\overline{q}}_{des}$, and their current value, as shown in the Figure 4.2:

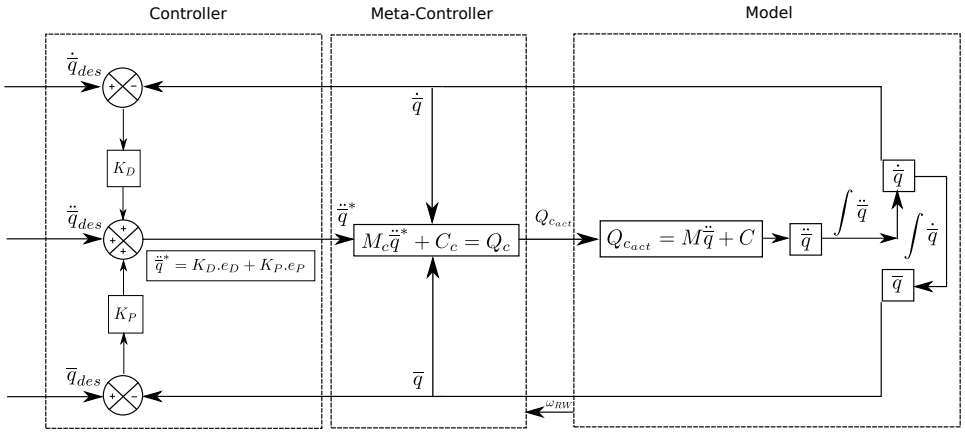


Figure 4.2: Model Based PD Control.

The Control Command consists of the desired set position, $\overline{q}_{des}$ and velocity, $\dot{\overline{q}}_{des}$, of each state and compares it each time with their current state, in order to reach the desired state. The Controller is doing that by taking as Feedback, at each repetition, the value of the position $\overline{q}_{des}$ and of the velocity $\dot{\overline{q}}_{des}$ vector and calculates the error between them. It then inserts these errors in the Control Equation, using the K_P and K_D control terms, respectively, and calculates a temporary value of the acceleration state vector, $\ddot{\overline{q}}^*$, as shown in (4.5) below. The value of the acceleration $\ddot{\overline{q}}^*$, as well as the feedback from the position and velocity, $\overline{q}$ and $\dot{\overline{q}}$, are needed to be calculated in order to calculate the dynamics of the system, at each time state, using the Euler-Lagrange Equation [24]:

$$\ddot{\overline{q}}^* = \ddot{\overline{q}}_{des} + K_D(\dot{\overline{q}}_{des} - \dot{\overline{q}}) + K_P(\overline{q}_{des} - \overline{q}) \tag{4.5}$$

The above values are then inserted in the Euler-Lagrange Equation, as shown in the Equation 4.6 below, that describes the state of the system, since the $\overline{M}$ and $\overline{C}$ matrices represent the dynamic behaviour of the system. Thus the matrices are calculated for each value of the position and velocity of the system's generalized coordinates, at each given time.

$$\overline{M}(\overline{q})\ddot{\overline{q}}^* + \overline{C}(\overline{q}, \dot{\overline{q}}) = Q \tag{4.6}$$

Assuming that the system parameters included in the calculation of the M and C matrices are precisely known, it can be assumed that:

$$\begin{aligned} \overline{M} &\simeq M \\ \overline{C} &\simeq C \Rightarrow \end{aligned} \tag{4.7}$$

$$\begin{aligned} Q &= \overline{M}\ddot{\overline{q}}^{*} + \overline{C} \\ &= M\ddot{\overline{q}} + C \end{aligned} \tag{4.8}$$

where Q the vector containing the generalised Forces and Torques of the actuators, calculated based on the dynamic model of the system, resulting to the movement of the robot according to the control command.

Using the (4.8), the Closed Loop equation is written as:

$$\begin{aligned} &\overline{M}(\ddot{\overline{q}}_{des} + K_D(\dot{\overline{q}}_{des} - \dot{\overline{q}}) + K_P(\overline{q}_{des} - \overline{q})) + \overline{C} = M\ddot{q} + C \Rightarrow \\ &\overline{M}((\ddot{\overline{q}}_{des} - \ddot{q}) + K_D(\dot{\overline{q}}_{des} - \dot{\overline{q}}) + K_P(\overline{q}_{des} - \overline{q})) = 0 \end{aligned} \tag{4.9}$$

Since the M matrix is defined as positive, for all the possible values of the variables, from the (4.9) result seven linear and uncoupled error equations, their number being equal to the number of the state variables, as shown below:

$$\ddot{e}_j + K_{Dj}\dot{e}_j + K_{Pj}e_j = 0, \quad j = 1, ..., 7 \tag{4.10}$$

where e the position error, $\dot{e}$ the velocity error and $\ddot{e}$ the acceleration error value of each state variable in each loop.

Equation (4.10) is treated as a Homogeneous Linear Equation, of the form [25]:

$$s^2 + 2\zeta\omega_j s + \omega_j^2 = 0 \tag{4.11}$$

Therefore, from (4.10) & (4.11), it can be easily concluded that:

$$\begin{aligned} K_{Pj} &= \omega_j^2 \\ K_{Dj} &= 2\zeta\omega_j \end{aligned} \tag{4.12}$$

where ω the closed-loop natural frequency[1], and ζ the damping coefficient[2] of each state variable of the system.

The value of ζ is chosen equal to 1, so that the system has critical damping and therefore no overshoot. Equation (4.12) then becomes:

$$\zeta = 1 \Rightarrow K_{Pj} = \omega_j^2, K_{Dj} = 2\omega_j \tag{4.13}$$

Hence the settling time of the system equals to [26]:

$$t_s = \frac{6}{\omega} \Rightarrow \omega = \frac{6}{t_s} \tag{4.14}$$

The settling time was chosen equal to $t_s = 8.5s$, based on observations from the simulations of the early Simulink models. The settling time should be:

- realistic (the system is able to reach the final state in the chosen time)
- as small as possible, so that the system reaches its desired state fast, however
- not too small, to avoid excessive power use

On that account, the settling time t_s should be a balance between the above restrictions.
In conclusion, the values of ω, K_P and K_D are calculated as follows:

$$\begin{aligned} \omega = \frac{6}{t_s} \Rightarrow \omega = 68.5 \Rightarrow \omega = 0.0706 \\ K_P = \omega^2 \Rightarrow K_P = 0.005 \\ K_D = 2\zeta\omega = 2 \cdot 1 \cdot 0.0706 \Rightarrow K_D = 0.1412 \end{aligned} \tag{4.15}$$

To reduce any oscillations, it is generally preferred to use the same settling time for all variables -so that all the variables are set to reach their desired state within the same time constraints. However, the thrusters affecting the 3 first state variables x, y and θ, are bounded by the constraints of low frequency and non-continuous operation. Moreover, the angle is also affected by the operation of the RW, which has different levels of operation as well. At the same time the motors actuating the manipulator joints, thus affecting the values of the

[1] the frequency at which a system oscillates when not subjected to a continuous or repeated external force

[2] critical damping occurs when the damping coefficient is equal to the undamped resonant frequency of the oscillator

state variables $q_{11}, q_{12}, q_{21}, q_{22}$ operate with less power and continuously, so they can reach their desired state faster and therefore can have smaller settling times.

The settling time was used as base for setting the time for all the variables, and each time was multiplied by a number, based on simulations' observations and on the knowledge of how fast can each error be minimised, as stated in the paragraph above.

The matrices then take the form of:

$$K_P = \begin{bmatrix} K_{P_1} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & K_{P_2} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & K_{P_3} & 0 & 0 & 0 & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & 0 & 0 & 0 & K_{P_7} \end{bmatrix}$$

$$K_D = \begin{bmatrix} K_{D_1} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & K_{D_2} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & K_{D_3} & 0 & 0 & 0 & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & 0 & 0 & 0 & K_{D_7} \end{bmatrix}$$

where $K_{Pi}, K_{D_i}, i = (1, ..., 7)$ the state and velocity gain vectors, respectively, each one depending on the settling time of the system. The matrices are multiplied to the error vectors, which include the error value for each state variable in each calculation repetition, as shown below:

$$e = \begin{bmatrix} e_1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & e_2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & e_3 & 0 & 0 & 0 & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & 0 & 0 & 0 & e_7 \end{bmatrix}$$

$$\dot{e} = \begin{bmatrix} \dot{e}_1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \dot{e}_2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \dot{e}_3 & 0 & 0 & 0 & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & 0 & 0 & 0 & \dot{e}_7 \end{bmatrix}$$

4.2.1 The Meta-Controller

The acting forces and torques vector, Q_c, includes, by definition, elements of different nature: Forces and Torques. The Q_c vector is calculated by:

$$Q_c = C_c + M_c \cdot \ddot{q}_c \tag{4.16}$$

where the index 'c' designates the vectors and matrices calculated through the controller module.

Therefore, these elements shall be brought to the point of reference, for the consistency of the Q_c vector (the vector produced by the calculations taking place within the meta-controller module). This is made possible with the normalisation of the values of the elements of the Q_c vector, by creating a weights matrix, W, which shall be multiplied by the Q_c, to produce the normalised vector $Q_{c_{act}}$.

The elements of the diagonal matrix, W, consist of the division by the maximum possible value of each actuator -either empirical, or taken from the actuators' specification sheets. However, since the maximum value of the Reaction Wheel is greater by orders of magnitude, therefore, for the normalisation of the acting and forces vector Q_c, the following matrix was created:

$$W = \begin{bmatrix} \frac{1000}{f_t^{max}} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \\ 0 & \frac{1000}{f_t^{max}} & 0 & 0 & 0 & 0 & 0 & 0 & \\ 0 & 0 & \frac{1000}{f_t^{max}} & 0 & 0 & 0 & 0 & 0 & \\ 0 & 0 & 0 & \frac{1}{f_{RW}^{max}} & 0 & 0 & 0 & 0 & \\ 0 & 0 & 0 & 0 & \frac{1000}{f_m^{max}} & 0 & 0 & 0 & \\ 0 & 0 & 0 & 0 & 0 & \frac{1000}{f_m^{max}} & 0 & 0 & \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{1000}{f_m^{max}} & 0 & \\ 0 & 0 & 0 & 0 & 0 & 0 & & 0 & \frac{1000}{f_m^{max}} \end{bmatrix}$$

where $f_t^{max} = 0.7N$, $f_{RW}^{max} = 0.0925N$ and $f_m^{max} = 6.5N$, the selected saturation value of the thrusters, the RW and the manipulators' motors, respectively.

The acting Torques and Forces explicitly have number equal to the number of the acting actuators to the system, eight in number. However the state variables are seven -not eight: $x, y, \theta, q_{11}, q_{12}, q_{21}, q_{22}$. Ergo, for the needs of consistency of the Q_{act} vector within the Euler-Lagrange equation that describes the dynamic

behaviour of the physical system, the Q_c vector has to be brought transformed from a [1x8] to a [1x7] vector.

This is possible with the multiplication of the Q_c vector with the Jacobian matrix, J_{act}, of size [8x7]. Evidently, the multiplication of a [8x1] vector by a [7x8] matrix produces a [7x8].[8x1] = [7x1] vector. The adaption of the theory elaborately described in [18], leads to the following equation, which results to the J_w matrix, size [8x7]:

$$J_w = (W^T \cdot J_{c_{act}}^T \cdot J_{c_{act}} \cdot W)^{-1} \cdot W^T \cdot J_{c_{act}}^T \tag{4.17}$$

The $J_{c_{act}}$ matrix is included in the Appendix B.

Hence the $Q_{c_{act}}$ vector is produced by the multiplication of the J_w matrix by the Q_c vector:

$$Q_{c_{act}} = J_w \cdot Q_c \tag{4.18}$$

4.2.1.1 The Reaction Wheel Torque

The controller developed is model-based, which means that it uses the model (a representation of the dynamic behaviour of the physical system) to compute the actuator inputs. Essentially, the controller produces the forces and torques whose effect is direct on the system's state variables, $x, y, \theta, q_{11}, q_{12}, q_{21}$, and q_{22}.

For the case of the RW, the Torque produced is consumed by the static friction T_f developed during the operation of the wheel and the torque due to damping, $B \cdot \omega$, while the rest is what is actually used to drive the load, as shown below:

$$T_{RW} = T_f + B \cdot \omega + J \cdot \dot{\omega} \tag{4.19}$$

where T_f the static friction losses, B the damping coefficient, ω the motor speed in rad/s, J the inertia coefficient. and $\dot{\omega}$ the wheel acceleration.

In 4.19 the part $J \cdot \dot{\omega}$ represents the Torque consumed by the load for its acceleration (torque due to inertia). The latter is the value produced by the controller, since the controller produces the necessary values to be acted on the load, to move the system to its desired state.

However, the $Q_{c_{act}}$ vector constrains the information sent to the actuators, the control command, of the forces and torques they shall produce. For the case of the RW though, to the acting torque value, $Q_{c_{act}}$ as calculated by the controller, is also added the value of the friction and damping losses, in order for

the controller to send the command to the RW to produce the torque actually required by the system to reach its final state, after the extraction of the losses. This means that the wheel motor needs to produce torque equal to the torque needed for the load to reach the desired value, plus the torque that will be lost due to friction and samping. For this reason, a final calculation is made before the value of the $Q_{c_{act}}$ vector is sent to the model, as a forethought for this phenomena:

$$Q_{c_{act}} = Q_{c_{act}} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ T_f + B \cdot \omega \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{4.20}$$

The $Q_{c_{act}}$ vector is then sent to the model, where the saturation of the torques and forces will take place, before the inverse kinematics calculations, which shall produce the new value of the acceleration, $\ddot{q}$, velocity, $\dot{q}$ and position, q vectors.

4.2.2 The Model's physical constraints

In order to have a realistic approach to the system's function, before the $Q_{c_{act}}$ vector passes through the model, it is filtered' through the physical constraints of the actuators' performance. After the saturation of the actuators' forces and torques, the inverse kinematics calculate the acceleration of the system, given actuation acting upon the system. The acceleration $\ddot{q}$ is then integrated once to produce the velocity $\dot{q}$, and twice, to produce the position q vector of the system.

The acceleration $\ddot{q}$ is calculated as follows:

$$\ddot{q} = M^{-1}(Q - C) \tag{4.21}$$

Acting External Force

The system has the provision to eliminate any external force acting upon the system through the arms' EE, in the case of contact or other physical reasons.

When detecting an acting force on the EE, the $Q_{E_{act}}$ force is added to the $Q_{c_{act}}$ vector, to produce the Q vector, which shall be used in the inverse kinematics calculations:

$$Q = Q_{c_{act}} + Q_{E_{act}} \tag{4.22}$$

A final check takes place, bounded by the constraints of the arms' accessible workspace, as elaborately explained in [27]. Before sending the position vector to the controller module, the current position of the arms is confirmed to be within the limits or not. If the position exceeds the limits, the variable is given the maximum possible value. The manipulator joints are bounded as follows:

$$\begin{aligned} -53^{\circ} \leq q_{11}, q_{21} \leq 150^{\circ} \\ -172^{\circ} \leq q_{12}, q_{22} \leq 95^{\circ} \end{aligned} \tag{4.23}$$

where point 0 is considered the point where the manipulator joints are facing the centre (straight joints), the positive direction is the outwards direction and negative the inwards direction (where the manipulators are facing each other).

The values for acceleration, velocity and position of the system's state variables are then sent to the controller modules, in a loop, until the system reaches its desired state.

4.3 Actuators Saturation

Below is presented the implementation of the saturation of the system's actuators, in an attempt to simulate the behaviour of the physical system.

4.3.1 Thrusters

The thrusters are set to deliver Thrust equal to 0.7 NM. Therefore the saturation for the first three elements of the $Q_{c_{act}}$ vector is set to 0.7:

$$\begin{bmatrix} Q_{f_{1,2}} \leq 0.7 \\ Q_{f_{3,4}} \leq 0.7 \\ Q_{f_{5,6}} \leq 0.7 \end{bmatrix} \tag{4.24}$$

4.3.2 Reaction Wheel

The operation of the Reaction Wheel (RW) is described by the following Equation:

$$T = K_T \cdot i_a = T_f + B(\omega_w - \omega_b) + J_w \dot{\omega}_w \tag{4.25}$$

where T the output (produced) Torque by the RW, T_f the Torque lost by static friction (during its operation), B the damping coefficient, J_w the inertia of the wheel, ω_b the absolute speed of the robot base, ω_w the absolute speed of the wheel, and $\dot{\omega_w}$ its acceleration.

While the speed of the system is fixed (ω_w-$\omega_b = const.$), and $\omega_w = const.$ the acceleration

$$\dot{\omega}_w = 0 \tag{4.26}$$

Therefore (4.25) becomes:

$$\begin{aligned} &T = T_f + B(\omega_w - \omega_b) + J\cancelto{0}{\dot{\omega}_w} \Rightarrow \\ &T = T_f + B(\omega_w - \omega_b) \end{aligned} \tag{4.27}$$

Meanwhile, the RW needs to acquire a minimum amount of power in order for the wheel to start spinning, since the Torque produced is consumed by the static friction inside the wheel. When the power exceeds that limit, equal to the value of static friction of the wheel, the produced Torque is then able to spin the wheel.

$$T = T_f + B\cancelto{0}{\omega} \Rightarrow T = T_f \tag{4.28}$$

During an experiment conducted, in order to determine the parameters of the RW, it was found that the losses out of static friction in the wheel are equal to:

$$T_f = 0.012Nm \tag{4.29}$$

Then, (4.27) is written as:

$$T = 0.012 + B\omega \tag{4.30}$$

In order to calculate the value of the damping coefficient, B, (4.30) is written as:

$$B = \frac{T - 0.012}{\omega_w - \omega_b} \Rightarrow B = \frac{T}{\omega_w - \omega_b} - \frac{0.012}{\omega_w - \omega_b} \tag{4.31}$$

The above equation (4.31) is of the form of:

$$y = ax + b \tag{4.32}$$

meaning a first order linear equation.

4.3.2.1 The experiment - Determining the Parameters

For this reason, an experiment was executed, in order to determine the value of the damping coefficient, B, by taking measurements of the Torque produced by the RW, in relevance to its speed. The RW was given a certain value of the speed it needed to reach, and the Torque produced at each moment was observed and recorded. The RW was taking command through a ROS module, and the values were recorded each in a rosbag file. These files were extracted in a Matlab environment into Matlab recognised time-series and were plotted, in respect to each other.

In order to analyse the data collected, they had to be brought to a known form, where further conclusions could be made. For this reason, the set of values of torque and speed were transformed to a linear equation, of the form of (4.32). The value of 'a' would equal to $a = \frac{T}{\omega}$ and of b, respectively, to $b = \frac{0.012}{\omega}$.

Below are shown the figures drown by the post-processing of the data collected during the experiment.

4.3.2.2 Experiment 1

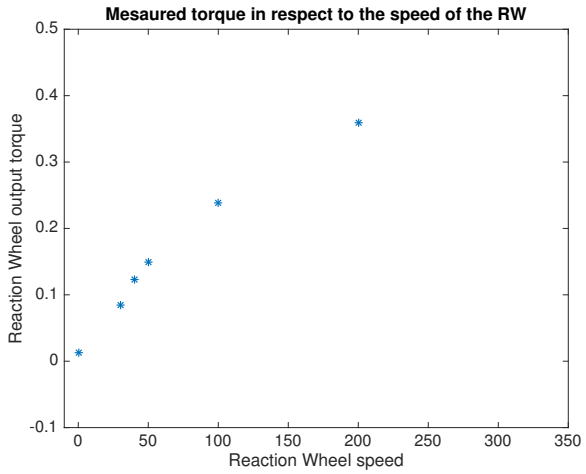


Figure 4.3: Plot of the output torque (T_{RW}) Vs. speed (ω) of the RW.

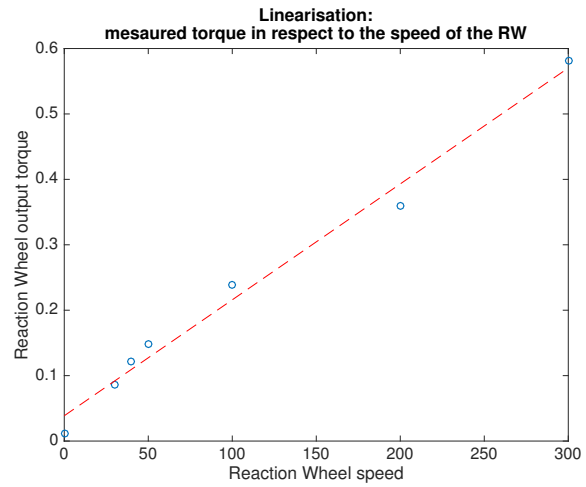


Figure 4.4: Linearisation of the equation of the output torque (T_{RW}) Vs. speed (ω) of the RW.

$$\begin{aligned} a &= 0.0018 \\ b &= 0.0387 \\ \Rightarrow y &= 0.0018x + 0.0387 \end{aligned} \tag{4.33}$$

$$\Rightarrow B = 0.0018, \quad T_f = 0.0387 \tag{4.34}$$

4.3.2.3 Experiment 2

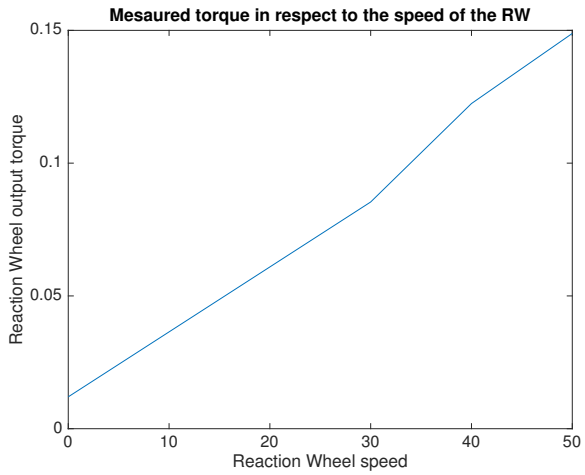


Figure 4.5: Plot of the output torque (T_{RW}) Vs. speed (ω) of the RW - Measurements before the RW's saturation.

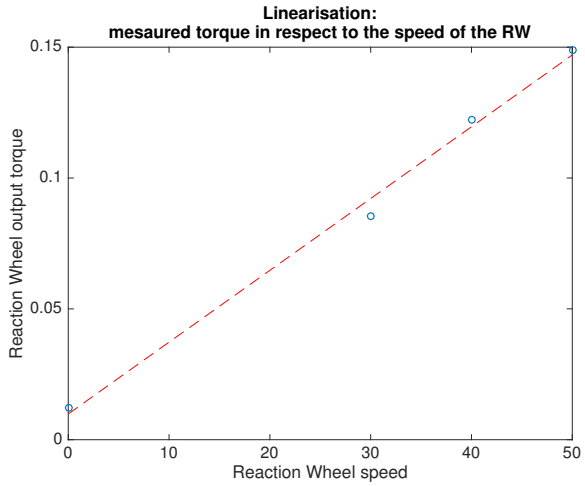


Figure 4.6: Linearisation of the equation of the output torque (T_{RW}) Vs. speed (ω) of the RW - Measurements before the RW's saturation.

$$
\begin{aligned}
a &= 0.0027 \\
b &= 0.0099 \qquad (4.35) \\
\Rightarrow y &= 0.0027x + 0.0099 \\
\Rightarrow B = 0.0027, &\quad T_f = 0.0099 \qquad (4.36)
\end{aligned}
$$

4.3.2.4 Conclusion

Since the value of 0.0387 is a more realistic value for torque due to static friction losses, the value of $B = 0.0018$ is chosen for the the damping coefficient. Therefore, the losses from static friction are taken as $T_f = 0.0387Nm$.

4.3.3 Arms' Motors

The arms' servo motors' output torque is bounded by the:

- stall torque value of the DC motor $\tau_{ma}^{max} = 136mNm$,
- planetary gear's specifications, with:
  - reduction gear ratio $n = 190 : 1$,
  - number of stages $= 3$,
  - maximum intermittently permissible torque at gear output $\tau_g^{max} = 6.5Nm$ and
  - efficiency $\eta = 70\%$

Therefore, the maximum output torque produced by the motors of the arms can be calculated by:

$$\tau_m = \tau_g^{max} \cdot n \cdot \boldsymbol{n} = 0.136 \cdot 190 \cdot 0.7 = 0.18Nm \leq \tau_g^{max} = 6.5Nm \qquad (4.37)$$

Hence the maximum value for the torque required to be produced by the arms' actuators of the system shall be defined as:

$$
\begin{bmatrix}
Q_{m_1} = Q_{act,5} \leq 0.18Nm \\
Q_{m_2} = Q_{act,6} \leq 0.18Nm \\
Q_{m_3} = Q_{act,7} \leq 0.18Nm \\
Q_{m_4} = Q_{act,8} \leq 0.18Nm
\end{bmatrix} \qquad (4.38)
$$

Chapter 5

The Planner

5.1 Presentation of the case study

The case under study is the control and trajectory planning of a space robot, therefore the scenario must be as realistic as possible to real life space missions; grasping space junk, docking and on-orbit servicing. Therefore the scenario suggests that the chaser/servicer, approaches the target (space dynamics not included in the study) and then attempts to grasp the targeted spacecraft.

The controller assumes that the chaser is aligned, at a planar level, with the target, and aims to orient the acting face of the chasing robot towards the targeted spacecraft, approach the latter and manage to grasp the moving object. The chaser end-effectors and target eventually must have the same planar speed.

The actuators are working on the chasing spacecraft until the approach. When the target is within the reachable workspace of the chaser, momentarily, no external forces are acting on the body of the robot, except from the ones causing its angular orientation and the movement of the arms - arms' motors.

5.1.1 Optimisation Criteria

The objective of the planner is the minimisation of the fuel consumption of the chasing spacecraft. Therefore, the optimisation objective was chosen to be the mass flow rate of the fuel consumed during the whole motion of the robot; from the approach manoeuvres, to the grasping of the target. For the space robot

emulator of the CSL, the fuel used is CO_2, which is stored in on-board cartridge, set to deliver the CO_2 across the system (to the air bearings and the 3 thrusters sets).

The CO_2 consumption is defined as the amount of the CO_2 exiting the nozzles of the thrusters mechanisms, per time rate (mass flow rate). The CO_2 cartridge is set to a fixed pressure of $6bar$, where $1bar = 1 \times 10^5$ Pa.

The mass flow rate of the CO_2 is defined as:

$$\dot{m} = \frac{mass\ of\ CO_2\ exiting\ the\ thruster\ nozzles\ [ks]}{unit\ of\ time\ [s]} \tag{5.1}$$

The fuel mass flow $m_f = m_{CO_2}$ and its exiting velocity from the thruster tube, Vj, are the ones causing the planar movement of the robot, from the generation of Thrust, P, which they are analogous to, as shown in (5.2):

$$P = m \cdot V = m_f \cdot Vj \tag{5.2}$$

We consider the mass fuel consumption m_f to be proportional to the acceleration a_R of the robot, thus to the acting Force on the body, in this case, the total force generated by the thrusters, which is equal to the time derivative of the Thrust P, as shown in (5.3):

$$F = \dot{P} = \dot{m} \cdot V + m \cdot \dot{V}, \quad V_{CO_2} >> V_{base} \tag{5.3}$$

We make the assumption that the fluid Velocity is constant, since the fluid is set to exit the tube with fixed pressure, regardless the pressure inside the CO_2 bottle (which is constantly reducing during operation). Also, the tube's section diameter d_e is constant, and equal to $d_e = 1.85mm$, and thus its area is equal to:

$$A_e = \pi \frac{d_e^2}{4} = \pi \cdot \frac{1.85 \times 10^{-3^2}}{4} = 2.69mm^2 \tag{5.4}$$

Therefore the 5.3 yeilds:

$$\begin{aligned} F = \dot{P} &= \dot{m} \cdot V + m \cdot \cancelto{0}{\dot{V}} \\ &= \dot{m}_f \cdot (V_j - V_\infty) \\ &= \dot{m}_f \cdot (V_J - \cancelto{0}{V_\infty}) \\ &= \dot{m} \cdot V_j \end{aligned} \tag{5.5}$$

We would be able to measure the Force generated by each nozzle by having a force sensor attached to each thruster tube outlet, the mass flow rate with a mass flow meter and the velocity through a pitot-tube, with the use of the following calculation:

$$V_j = \sqrt{\frac{2(p_t - p_s)}{\rho}} \tag{5.6}$$

p_t being the stagnation pressure and p_s the static pressure, whereas the pitot-tube measures the stagnation pressure, and the static pressure is known and equal to 1 bar.

However this kind of system would be complicated to be procured, installed and properly put into use, with a lot of errors in play.

When controlling a physical system a level of redundancy is considered, high enough to cover any disregard of fluid dynamics phenomena. The approach is then focusing on the robot's acceleration, in regard to the function of the thrusters" operation (not the RW).

As explained in [28], minimising fuel consumption can be achieved following different methods, one of which is the use of generating functions and Hamiltonian Dynamics [29]. While this method is mathematically solid, it is not straightforward to apply to nonlinear systems. Optimal Control is more suitable for nonlinear systems, and easier to program.

The paper suggests the use of a [2x3] D matrix, which is the matrix that transforms the nozzle thrust vector into forces in axes x and y and, as shown in (5.7):

$$F = D \cdot \begin{pmatrix} f_1 \\ f_2 \\ f_3 \end{pmatrix} = \begin{pmatrix} f_x \\ f_y \end{pmatrix} \tag{5.7}$$

By minimising (5.7), $|f_1| + |f_2| + |f_3| = min$, hence the minimum force vector f can be found:

$$F = D \cdot f \Rightarrow f = D^+ \cdot F \tag{5.8}$$

where f_1, f_2, f_3 are the forces gerenated by thruster 1, 2, 3 respectively, and can take positive and negative values, as shown in Figure 5.1.

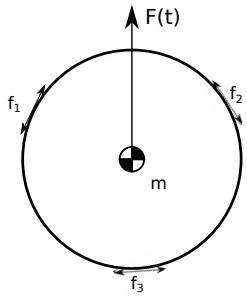


Figure 5.1: The thrusters sets sum up to a total acting force F on the robot.

The fuel consumption optimisation aims to achieve the minimum use of the thrusters, when the robot is executing a certain task. In this case, the robot needs to reach a moving object, with a certain distance and orientation, grasp it and then move with the same velocity as the target. For this reason, a combination of thrusters and the RW need to be put in use to achieve the task.

Essentially, as thrusters' operation shall generate the acceleration the robot needs, produced by the Model Based Controller (MBC), in order to reach a specific point on the plane, in a specific time, already calculated and provided by the planner of the model.

In the case that the robot is to move on a straight line, compared to its initial position, the needed thrust can be provided with the use of only one thruster, which applies its maximum output value, $F_{max} = 0.7N$. The latter would then cause the robot body to rotate, since the absence of friction hence the RW shall operate to generate torque equal in value and opposite to the generated torque, as shown in Figure 5.2.

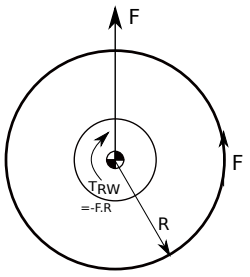


Figure 5.2: One thruster set generating the total thrust needed to move the robot, $F = F_{max} = 1N$.

A possible alternative would be the simultaneous use of two thrusters, which shall create a vector in the y-direction, and a vector in the x-direction, equal in value and opposite in direction which each other. as shown to Figure 5.3.

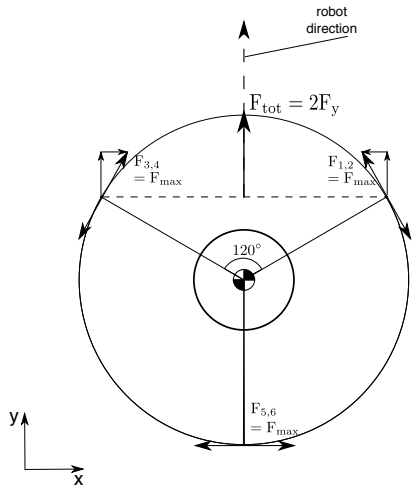


Figure 5.3: Two thrusters sets generating the total Thrust to move the robot, $F = F_{tot} = \sqrt{2}N$; the x-vectors of the acting forces neutralise each other.

The sum of the vectors in the y-direction would be equal to:

$$F_{tot} = 2 \cdot F_x = 2 \cdot F_{max} \cdot \cos 30^\circ = 2 \cdot 0.7 \cdot \frac{\sqrt{3}}{2} = 0.7\sqrt{3}N \tag{5.9}$$

The RW does not need to operate in this case. However, force equal to 2 N is wasted into counterbalancing the thrusters' forces y-vectors. Apparently this case is not considered as optimal.

Another solution would be the minimisation of the integral of the acceleration, as explained in the section below.

5.1.1.1 Acceleration

The fuel mass consumption is directly related to the acceleration of the body of the robot, since the need of accelerating the robot is what initiates the function of the actuators, as shown below:

$$F = m_{bot} \cdot a \tag{5.10}$$

Therefore, the maximum acceleration is bound by the maximum thrust the thrusters can produce, as shown in (5.11):

$$\begin{aligned} F_{max} = m_{bot} \cdot a_{bot}^{max} \Rightarrow a_{bot}^{max} &= \frac{F_{max}}{m_{bot}} \\ \Rightarrow a_{bot}^{max} &= 0.088 \ m/s^2 \end{aligned} \tag{5.11}$$

Thus the maximum acceleration of the system of the robot is $a_{bot}^{max} = 0.088 m/s^2$.

This sets a constrain to the movement of the robot towards the target; the robot could never accelerate more than that number. If the latter is not taken under consideration, the model would be unrealistic hence not accurate.

If the values for the acceleration of the model driving from the controller, depending the position of the target in respect to the chaser, are larger, then the robot should accelerate with the maximum acceleration rate until it reaches the target. This however would result in a high fuel consumption since it would require from the body of the robot to constantly accelerate, with the maximum rate.

To minimise the fuel consumption, we should find the optimum acceleration rate, for the corresponding amount of time, in order for the chasing spacecraft to reach its final destination using as least fuel as possible. Minimising the integral of the acceleration of the robot, thus of the derivatice of the velocity, in respect to time, could be a possible solution to the problem, as shown below.

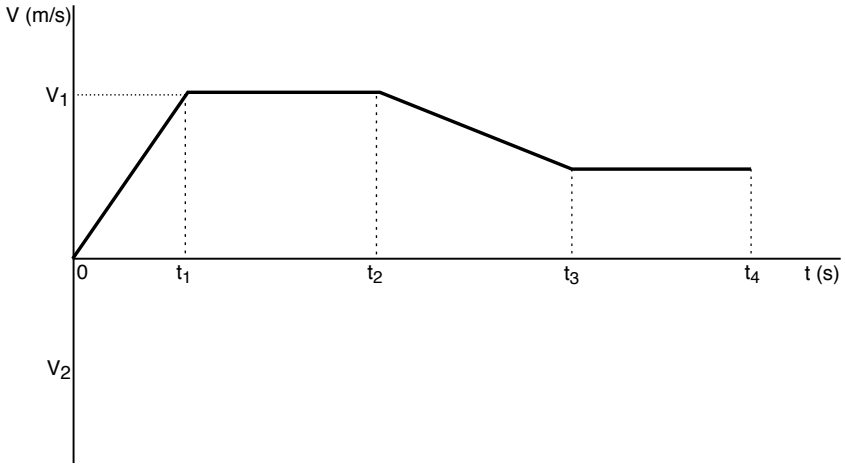


Figure 5.4: The profile of the velocity of the robot's base, V_{bot}.

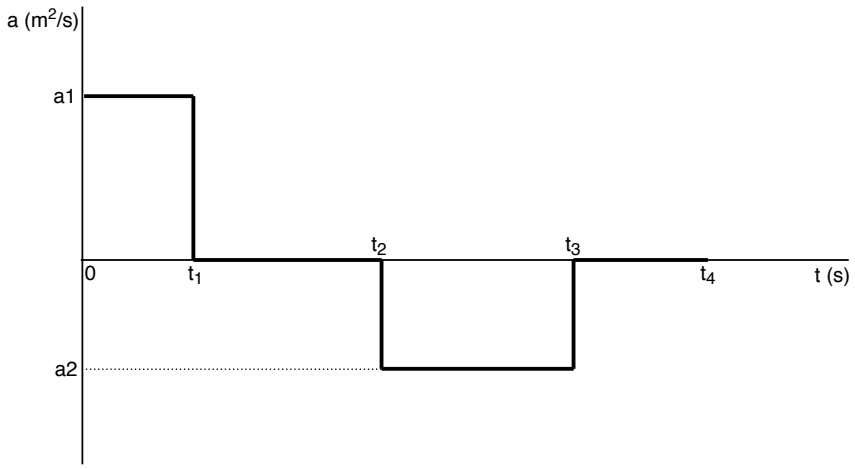


Figure 5.5: The time integral of acceleration $\int_0^t a_{bot}dt = V_{bot}$.

Figure 5.4 shows the profile of the velocity of the robot's base in respect to time, and 5.5 of the latter's acceleration.

However attractive this approach might seem, it was decided that it is not a feasible solution for this case study. The reason is that the integral of the robot's acceleration equals to the value of its velocity, $\int_0^t a_{bot}dt = V_{bot}$, yet the desired Velocity is given, that needs to be reached in a certain time, and provided by the planner, since the trajectory of the target is foreknown. Thus the integral of the acceleration by time shall always have the same value, regardless the different combinations of a1-t1, a2-t3, since the acceleration and deceleration of the robot shall always result to the same value of the robot's velocity, reached in a specific point in time. Consequently, this solution is not suitable for this case study.

Taking under consideration all the above, another approach was chosen, as explained in the following sections.

5.2 Trajectory planning

The approach chosen for the planning of the chaser trajectory is taking into account the fact that the trajectory of the target is known. By knowing the initial position of the target and its equations of motion, the planner receives the calculated x and y position so that the chaser is au courant with the planar

position of the target.

The position of the target spacecraft is described by the following equations:

$$x_t(t) = it + k \tag{5.12}$$
$$y_t(t) = mt + n \tag{5.13}$$

were $k, n = 1$ the initial position of the target and

$$V_{x,t}(t) = \dot{x_t}(t) = i \tag{5.14}$$
$$V_{y,t}(t) = \dot{y_t}(t) = m \tag{5.15}$$

The chaser is doing a projectile motion. The formulae describing its motion are the following:

$$x_c(t) = at^2 + bt + c \tag{5.16}$$
$$y_c(t) = dt^2 + et + f \tag{5.17}$$

where a, d the acceleration of the robot in the x and y axis respectively, at every point in time, c, f the initial position of the robot in the x and y axis respectively, and

$$V_{x,c}(t) = \dot{x_c}(t) = 2at + b \tag{5.18}$$
$$V_{y,c}(t) = \dot{y_c}(t) = 2dt + e \tag{5.19}$$

the x and y vector of the chaser Velocity.

5.2.0.1 Objective

The aim is that in given time, deriving from the geometry of the trajectories of the chaser and the target, as shown in Figure 5.6 below, the chaser reaches the same position as the target, and they have equal velocity, as described in the equations 5.20 & 5.21 below.

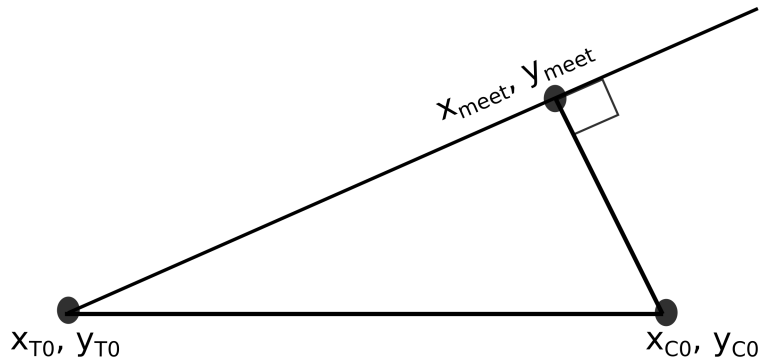


Figure 5.6: The chaser meets the target traversing the minimum distance - target to the target's trajectory.

$$x_c(t) = x_t(t) \Rightarrow at^2 + bt + c = it + k \tag{5.20a}$$
$$y_c(t) = y_t(t) \Rightarrow dt^2 + et + f = mt + n \tag{5.20b}$$

$$Vx_c(t) = \dot{x_c}(t) = Vx_t(t) = \dot{x_t}(t) \Rightarrow 2at + b = i \tag{5.21a}$$
$$Vy_c(t) = \dot{y_c}(t) = Vy_t(t) = \dot{y_t}(t) \Rightarrow 2dt + e = m \tag{5.21b}$$

By solving (5.20a), (5.20b), (5.21a) & (5.21b), we get:

$$a = \frac{i^2}{9(c-k)} \tag{5.22}$$
$$b = \frac{i}{3} \tag{5.23}$$
$$d = \frac{m^2}{9(f-n)} \tag{5.24}$$
$$e = \frac{m}{3} \tag{5.25}$$

$$t_{meet_x} = \frac{i}{3a} \tag{5.26}$$
$$t_{meet_y} = \frac{m}{3d} \tag{5.27}$$

So when the $time <= t_{meet_x} || t_{meet_y}$, the robot shall have reached the desired position, (x_{meet}, y_{meet}), and its planar Velocity shall then be equal to the target's planar velocity, and follow the latter's trajectory.

5.3 Target grasping

The controller is programmed so that, using inverse kinematics, since all the dynamic equations are have as a centre of reference the robot's COM, to calculate the end effectors' position and that to follow the trajectory of the target spacecraft. When the target is within the Working Space (WS) of the 2-DOF robot manipulator, the chaser's manipulators start moving in a preconfigured way, in order to grasp the target spacecraft.

The robot starts with identifying the position and orientation of the target spacecraft in relation to the former's position and orientation. The chaser then starts rotating in order to achieve the same orientation as the target spacecraft, in the Global Reference Point -Absolute Zero- of the system. In this point, the robot is only rotating and having zero displacement.

Continuing, with zero change in the chaser's orientation, the robot moves towards the target's position, always checking whether the target is yet within its Workspace or not. When so, the robot has zero rotation from then and on, as well as zero acceleration, since its Velocity is stable and equal to the target's velocity. Therefore no external forces are acting on the chaser robot and it attempts to grasp the target spacecraft, only by moving its arms. At the time of the grasping, the second links of the arm are in 45° angle in respect to the first link, since this configuration is considered to be optimal in grasping and avoids singularities. The methodology is thoroughly explained below.

5.3.1 Working Space

As explained in [30], for a 2 DOF robot manipulator, when no external forces are acting on the system, the equations for calculating the Path Dependent Workspace (PDW) and Path Independent Workspace (PIW) depend on the model's (base and manipulators) parameters. The latter statement is making the assumption that when the chaser has already approached the target, no external actuators are operating (thrusters & Reaction Wheel (RW)), except from the arms' motors, hence the robot's velocity is equal to zero, and the velocity is stable and equal to the target's velocity, as explained in 5.2.

The equations are adjusted to the case under study, which is a robot base with two 2 - DOF manipulators, with two links. R_1^{min} and R_2^{max} define the Reachable Workspace Boundaries, with respect to the centre is the COM of the robot. $R_1^{min} \rightarrow R_1^{max}$ define the limits of the PDW as well as $R_1^{max} \rightarrow R_2^{max}$. $R_1^{max} \rightarrow R_2^{min}$ consist the Path Independent Workspace (PIW) area and that shall be the desired WS for the robot to be able to manipulate the target object.

The two PDWs, constrained by (R_1^{min}, R_1^{max}) and (R_2^{min}, R_2^{max}) respectively, are calculated as follows:

$$\begin{aligned} R_1^{min} &= \beta + \gamma - \alpha \\ R_1^{max} &= \alpha + \gamma - \beta \\ R_2^{min} &= \alpha + \beta - \gamma \\ R_2^{max} &= \alpha + \beta + \gamma \end{aligned} \tag{5.28}$$

where:

$$\alpha = r_0^* = \frac{1}{M_r} r_0 m_B \tag{5.29}$$

$$\beta = r_1^* = \frac{1}{M_r} \{ r_1 (m_B + m_1) + l_1 m_B \} \tag{5.30}$$

$$\gamma = c_2^* + r_2 = \frac{1}{M_r} l_2 (m_B + m_1) + r2 \tag{5.31}$$

where:
M the mass of the robot
m_B the mass of the robot's base
m_1 the mass of the first link of the manipulator
m_2 the mass of the second link of the manipulator
l_1 the distance between the COM of the first link of the manipulator and the second joint
l_2 the distance between the second joint and the COM of the second link of the manipulator
r_0 the perpendicular distance between the COM of the base and the first joint of the manipulator
r_1 the perpendicular distance between the first joint to the COM of the first link of the manipulator
r_2 the perpendicular distance between the COM of the second link and the EE of the manipulator.

Therefore, according to the model's symbolism:

$$
\begin{aligned}
r_1 &= a_{c_1} \cos d_{a_1} \\
l_1 &= a_1 - a_{c_1} \cos d_{a_1} \\
r_2 &= a_{c_2} \cos d_{a_2} \\
l_2 &= a_2 - a_{c_2} \cos d_{a_2}
\end{aligned}
\tag{5.32}
$$

Hence the limits of the two Path Dependent Workspace and the Path Independent Workspace are calculated as shown in (5.33):

$$
\begin{aligned}
R_1^{min} &= 0.1163m \\
R_1^{max} &= 0.1416m \\
R_2^{min} &= 0.2401m \\
R_2^{max} &= 0.4980m
\end{aligned}
\tag{5.33}
$$

It was decided that the manipulators shall attempt to grasp the target spacecraft only when the latter is within the PIW, in order to avoid the risk of any singularity points. The area could by anywhere within (R_1^{max}, R_2^{min}), thus the arc created $45°$ from the main axis was chosen to be the workspace of the two 2-DOF manipulators, as shown in Figure 5.7

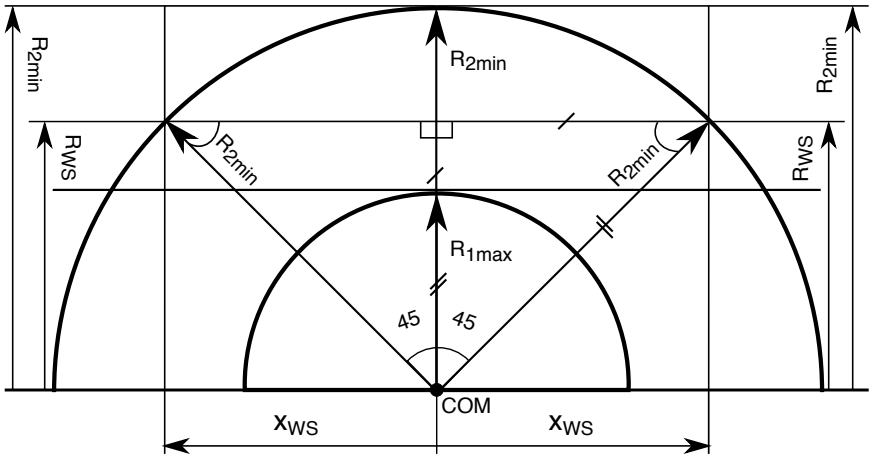


Figure 5.7: Calculation of the robot's manipulators WS, within (R_1^{max}, R_2^{min}).

where:

$$
\begin{aligned}
R_{WS} &= R_{2_{min}} \cdot \cos 45° \\
&= 0.2401 \cdot \cos 45° \\
=> R_{WS} &= 0.1698m
\end{aligned}
\tag{5.34}
$$

$$
\begin{aligned}
x_{WS} &= R_{WS} \cdot \tan 45^{\circ} \\
\Rightarrow x_{WS} &= R_{WS} = 0.1698m
\end{aligned}
\tag{5.35}
$$

The straight line, parallel to the x-axis, as shown in Figure 5.7, connecting the intersection points of the 45° tilted lines, with origin the COM of the robot, and the arc with R_2^{min} radius, was selected to be the grasping workspace, as shown in Figure 5.8.

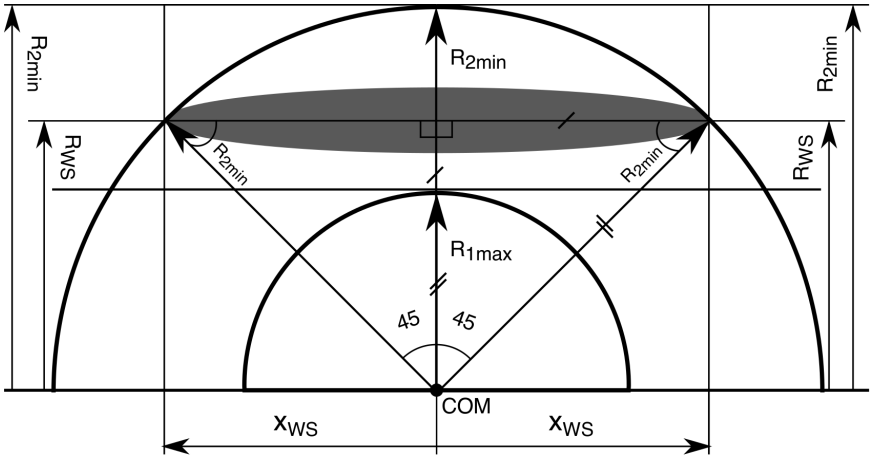


Figure 5.8: The robot's manipulators selected WS, within the PIW constraints, taking under consideration the limits set by the physical structure of manipulators.

5.3.1.1 The controller

The robot follows the trajectory generated by the controller, as defined by the PD controller, aiming to reach the position of the target spacecraft. The controller is constantly checking whether the y-position of the target is within (y_{WS_2}, y_{WS_3}), in distance to the robot's COM, and when that requirement is fulfilled, for the x-position of the target. If the distance between the x-position of the target and the robot's COM is less than $R_{2_{min}}$, when it is equal to x_{WS}, the manipulators attempt to grasp the object, as explained in Figure 5.9.

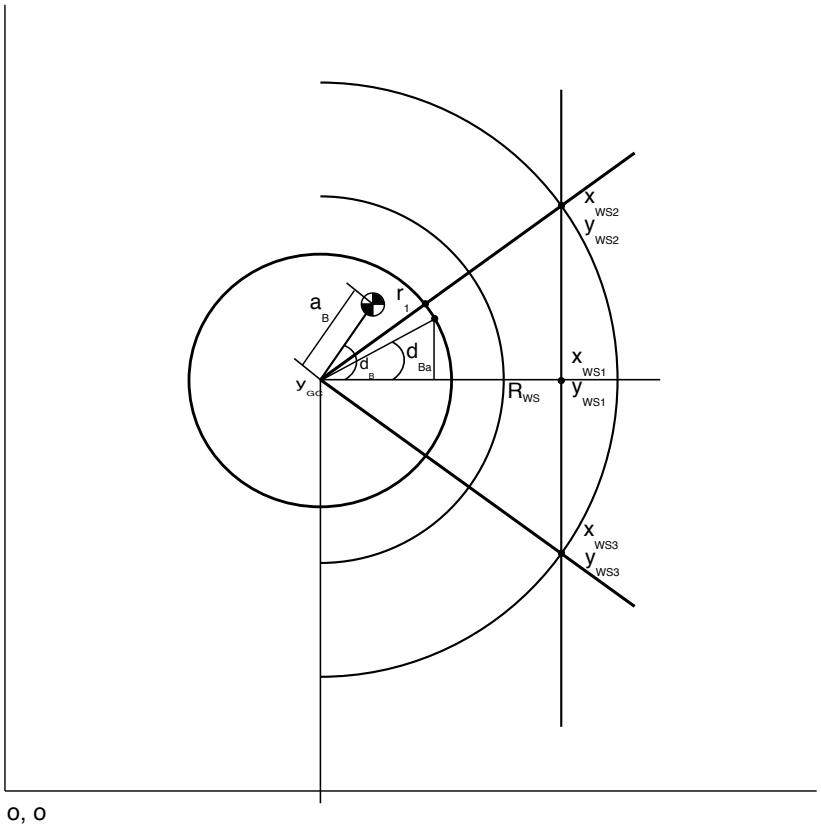


Figure 5.9: Calculation of the selected Workspace limit points.

The intersection points between the edges of the $45°$ arc and the R_2^{min} arc, as well as of the line with R_{WS} length abd the line crossing (x_{WS_2}, y_{WS_2}) and (x_{WS_3}, y_{WS_3}), (x_{WS_1}, y_{WS_1}), are calculated as shown in (5.36):

$$y_{GC} = q_{2,1} - a_B \sin d_B$$

$$\begin{aligned} x_{WS_1} &= q_{1,1} + x_{WS} \\ y_{WS_1} &= y_{GC} \\ x_{WS_2} &= q_{1,1} + x_{WS} \\ y_{WS_2} &= y_{GC} + x_{WS} \\ x_{WS_3} &= q_{1,1} + x_{WS} \\ y_{WS_3} &= y_{GC} \end{aligned} \tag{5.36}$$

where:

- y_{GC} the y-coordinate of the robot's base Geometrical Centre
- x_{WS} the projection of R_{WS} on the x-axis
- x_{WS_1} the x-coordinate of R_{WS}
- y_{WS_1} the y-coordinate of R_{WS}
- x_{WS_2} the x-coordinate of the upper y-boundary of R_{WS}
- $y_{WS_2}$the y-coordinate of the upper y-boundary of R_{WS}
- $x_{WS_3}$the x-coordinate of the lower y-boundary of R_{WS}
- y_{WS_3} the y-coordinate of the lower y-boundary of R_{WS}

Chapter 6

Results - Evaluation of the Simulink model

6.1 Model verification

6.1.1 Validation of the dynamic model

Before assigning to the robot more complex tasks, an initial model was developed in order to validate the correctness of the dynamic model and the accuracy of the controller. The controller is taking the vectors of the desired position q_{des} and velocity $\dot{q}_{des}$ as inputs, and aims to direct the system's COM to the points given by the control command. Since the robot is to stop moving when it reaches the desired position, the $\dot{q}_{des}$ vector is always equal to:

$$\dot{q}_{des}(t_{final}) = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{6.1}$$

In the position command are included the desired positions on the X-Y - plane of the robot's base and orientation. Hence the desired positions (control command) describe the position and orientation of the robot's base COM, as well as of

the angle of the arms' COM, which are set to zero, so as to control only the displacement of the base. The controller recieves feedback from the position and velocity of the generalized coordinates, at each loop, and the control command includes the hposition and velocity of the generalized coordinates vector. Figure 6.1 demonstrates the Simulink Model of the PD-Model Based Controller and the Dynamic Model Representation.

Essentially the user defines the robot chaser and target initial positions, which are sent to the controller, which calculates the Forces and Torques that need to be sent to the system and passes them through the meta-controller. The latter calculates the error vectors and sends them to the model, where the values of the actuators' torques and forces are saturated and the acceleration of the model is produced, and integrated to the system's velocity and position vectors, to be sent back to the loop.

The model-based controller is giving position and velocity commands to the system, and receives freedback of the states of the generalized coordinates at each loop. The variables under control are essentially the elements of the q vector, which are the planar position and orientation of the robot body, x, y, & θ, and the angular displacement of the arms' joints, $q_{11}, q_{12}, q_{21}, q_{22}$. In later examples, inverse kinematics are used to control the position of the end effectors of the manipulators' joints, while the system's state variables remain unchanged (the position and orientation of the COM of the robot body and of the manipulators' joints.

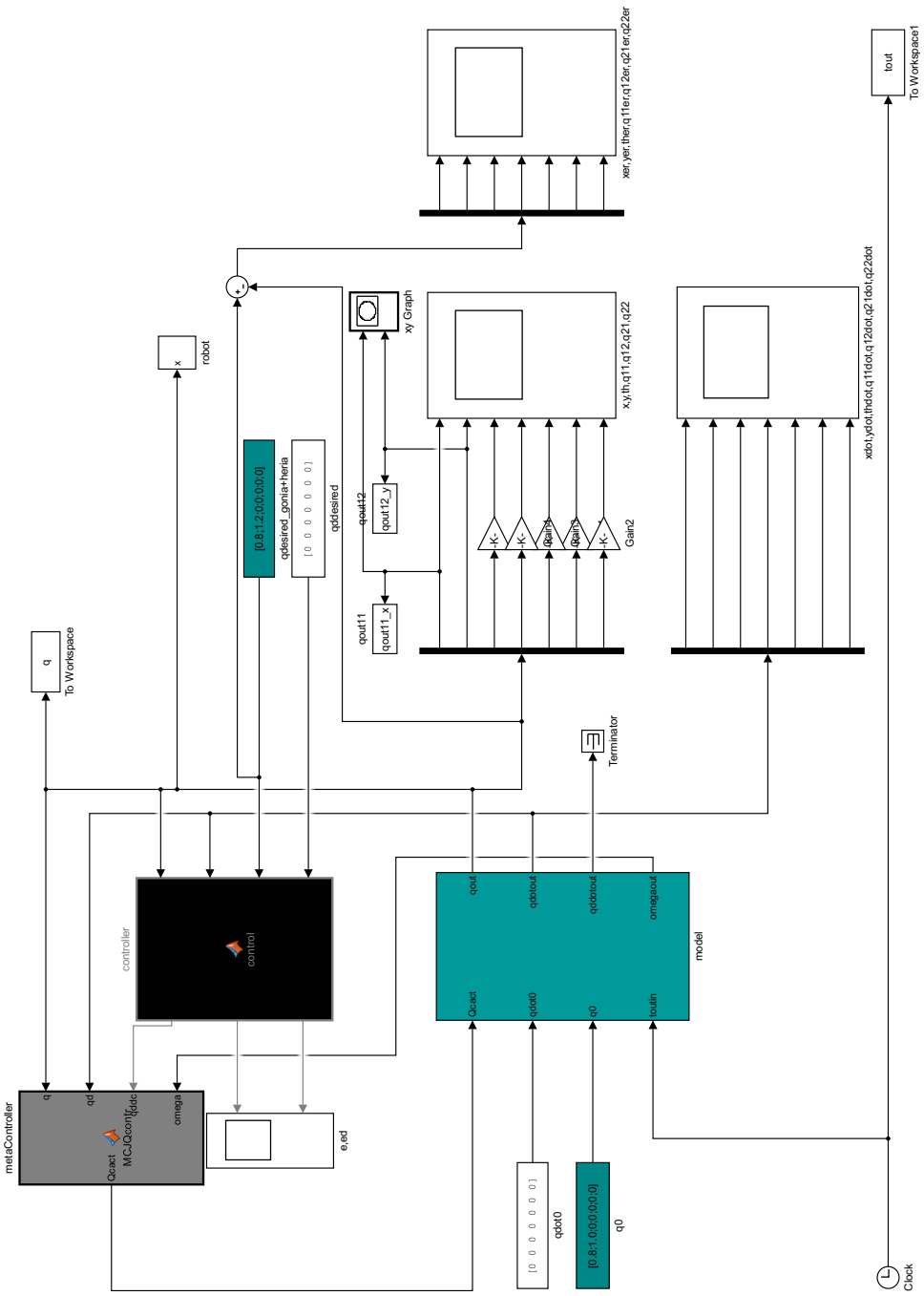


Figure 6.1: The Simulink Model for the Validation of the Dynamic Model.

6.1.2 Simulation Results

6.1.2.1 Case: XY Displacement

Firstly, the validity of the dynamic model and the controller should be verified. Therefore, the first scenario requires a simple task execution from the robot, in this case moving the robot's COM to a certain X-Y location.

The control command requires that the base moves by 0.2m on the X and Y axis respectively. The robot's COM initial position is equal to:
$(x_{COM_{init}}, y_{COM_{init}}) = (0.8, 1.0)$
and is commanded to reach the position:
$(x_{COM_{final}}, y_{COM_{final}}) = (1, 1.2)$.

The command is a set-point command and the controller is a model-based PD controller.

Based on experimental observations, the settling time was chosen to be equal to $t_s = 8.5$ s and $\zeta = 1$, as explained in 4.13. Therefore $\omega = 0.7059$, $K_P = 0.4983$ and $K_D = 1.4118$.

After the weights multiplication with K_P and K_D, the position and velocity gains matrices result to:

$$K_P = \begin{bmatrix} 0.4983 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.4983 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.7474 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 99.6540 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 99.6540 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 99.6540 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 99.6540 \end{bmatrix}$$

$$K_D = \begin{bmatrix} 1.4118 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1.4118 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1.4118 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 70.5882 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 141.1765 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 70.5882 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 70.5882 \end{bmatrix}$$

As shown in Figures 6.2, 6.3 and 6.4, the robot reaches smoothly the desired xy-position.

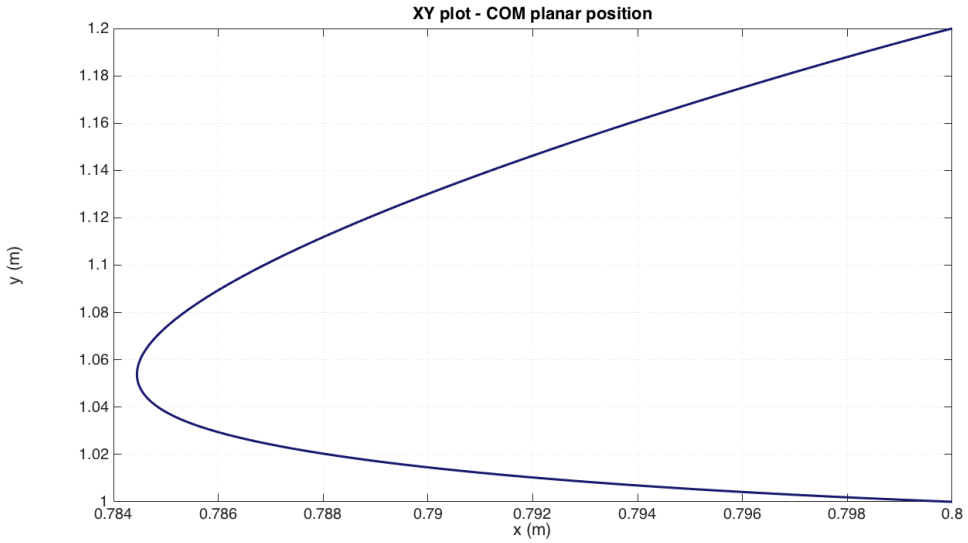


Figure 6.2: Planar motion of the robot's COM

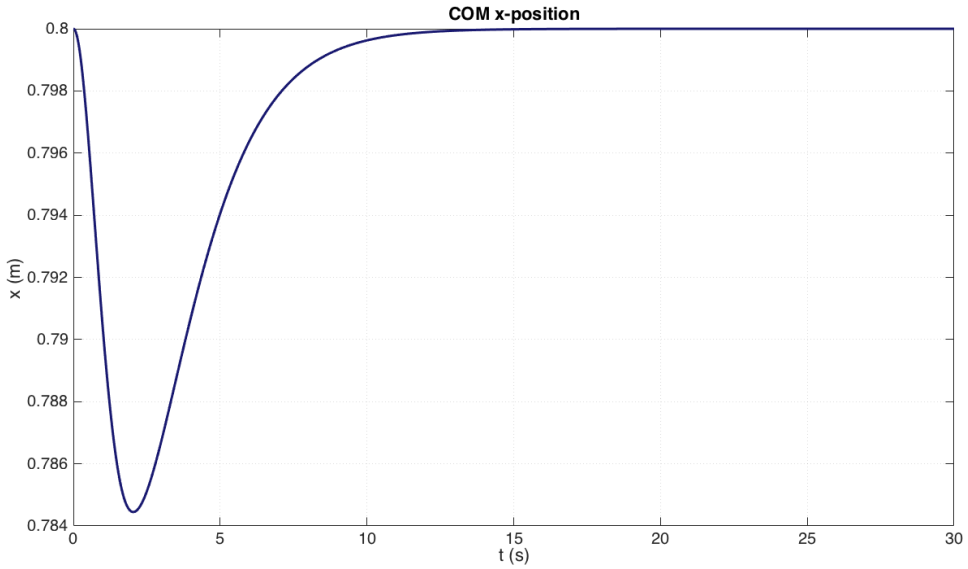


Figure 6.3: The robot's COM reaching the desired x-position.

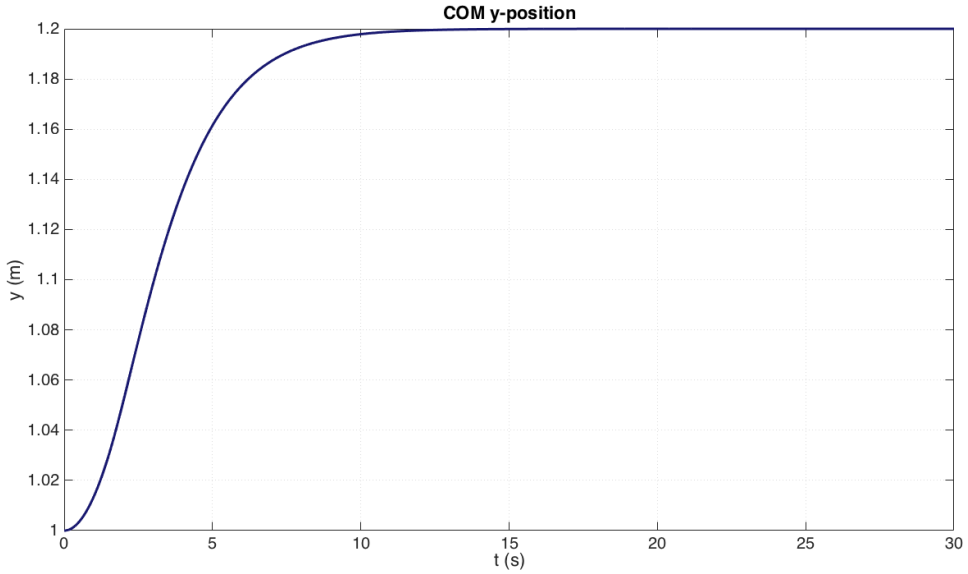


Figure 6.4: The robot's COM advancing to the desired y-position.

As shown in Figure 6.5, the planar motion of the robot creates a disturbance in the orientation of the robot, which reaches the value of 20° in 5s and then goes back to zero. As we can observe in Figures 6.6 and 6.8, this disturbance, caused by momentum exchange and the absence of friction, is more obvious in the first link of each manipulator, which are mounted on the base. While the second link of the first manipulator shows little disturbance, relatively to the first link's -Figure 6.7, the second link of the second manipulator -Figure 6.9, has a large overshoot in the first seconds of motion and goes back to zero by the 10^{th} second of motion. We can see that all the states are stabilised by the 10^{th} second.

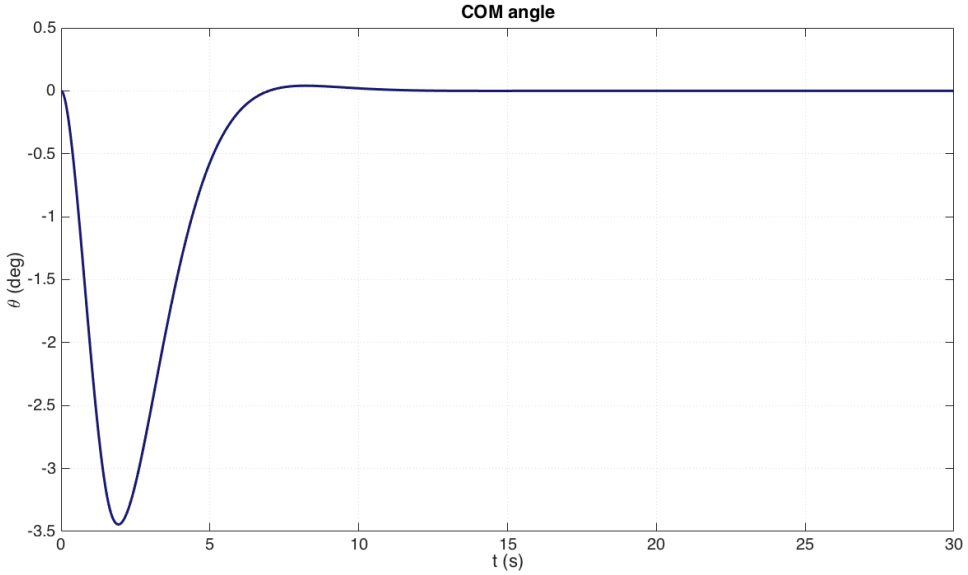


Figure 6.5: Orientation of the robot's COM.

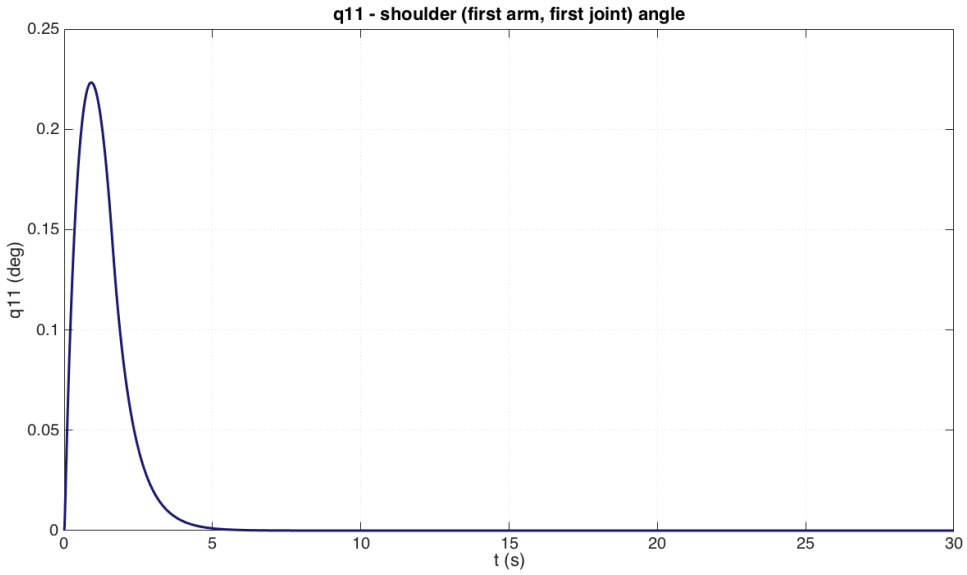


Figure 6.6: Position of the 1^{st} link of the 1^{st} manipulator in respect to time.

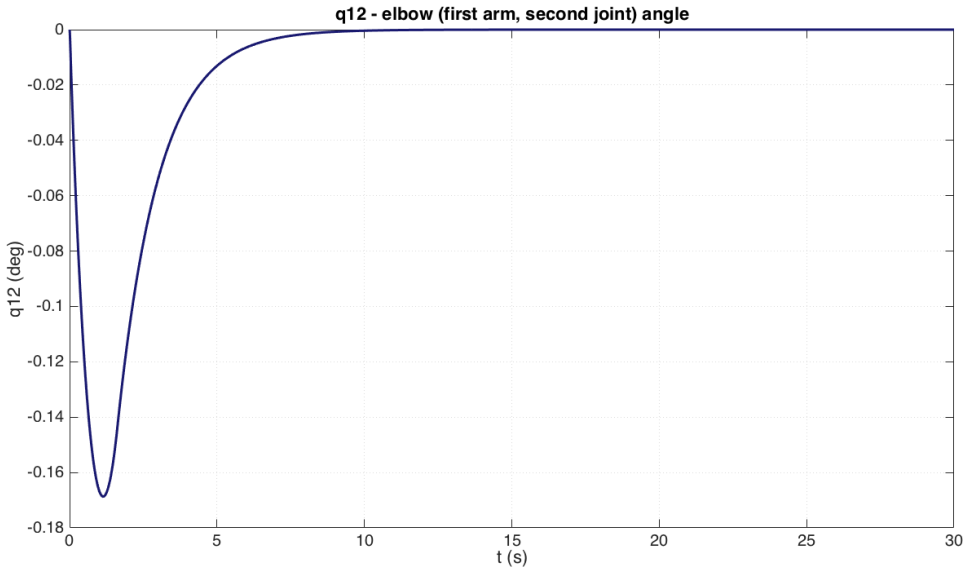


Figure 6.7: Position of the 2^{nd} link of the 1^{st} manipulator in respect to time.

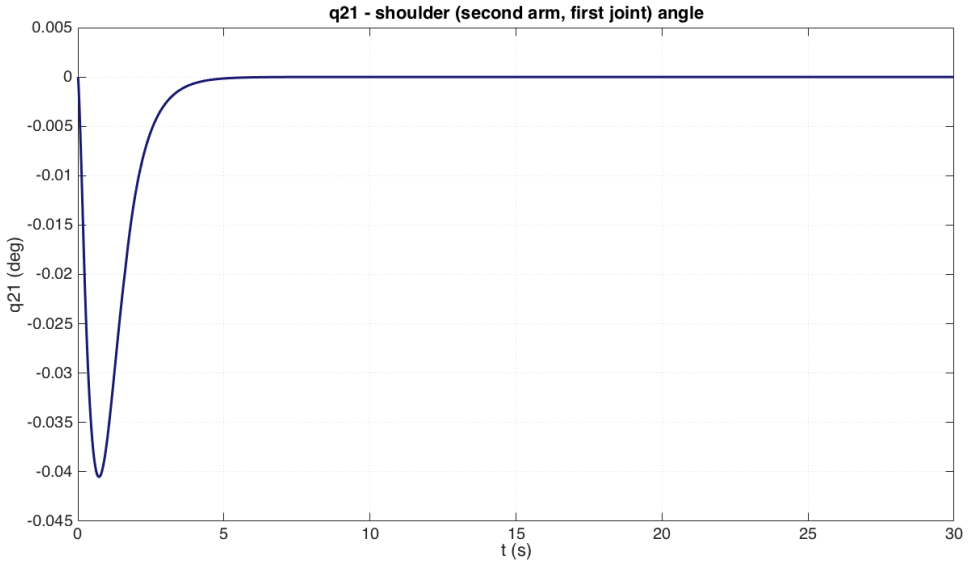


Figure 6.8: Position of the 1^{st} link of the 2^{nd} manipulator in respect to time.

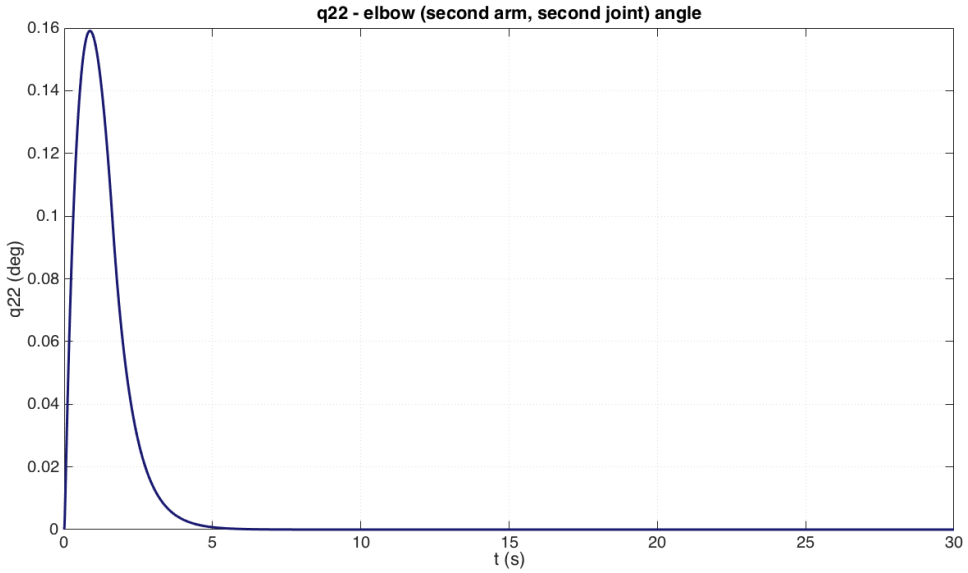


Figure 6.9: Position of the 2^{nd} link of the 2^{nd} manipulator in respect to time.

As shown in Figures 6.10 and 6.11, the error reduces smoothly, from the initial distance from the desired point to zero, within 10s.

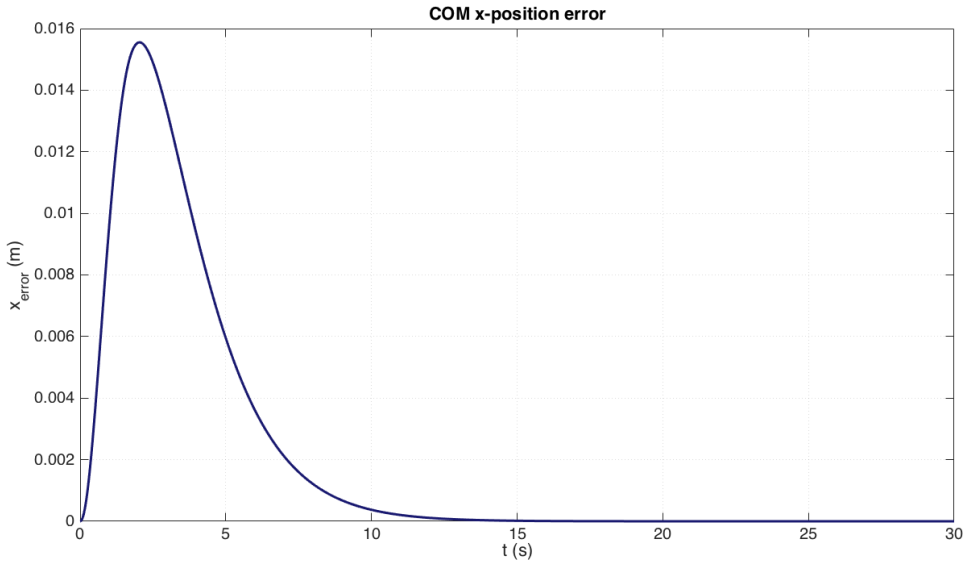


Figure 6.10: Error of the robot's x-position.

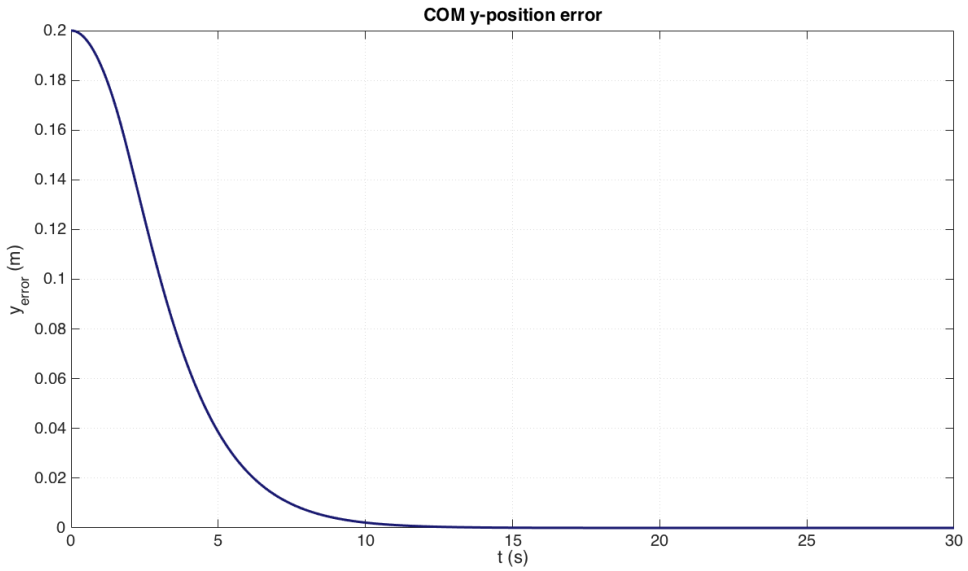


Figure 6.11: Error of the robot's y-position.

Figure 6.12 shows that the robot's angle error -in reference with the absolute coordinate system, has a minor overshoot of 0.27° on the 5th second and then goes to zero by the 10^{th}.

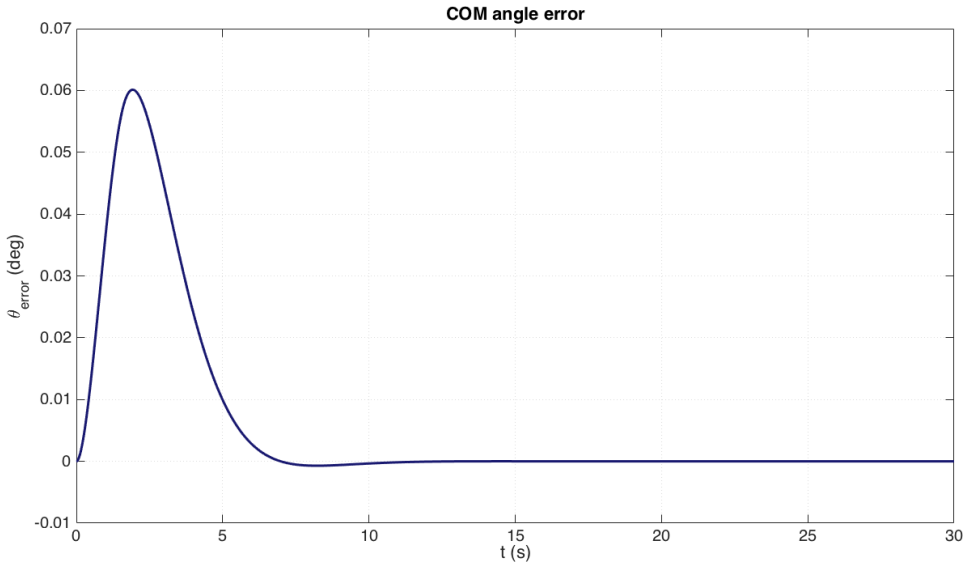


Figure 6.12: The robot's COM error of orientation.

Figures 6.13 and 6.14 indicate a minor overshoot in the errors of the manipulators' joints' displacement, of magnitude 10^{-3}. We can also observe the proper dynamics taking place in the momentum transfer within the system. A negative error on the first joint of the arm creates a positive error in the second link, and a positive error on the 5th second of the first joint, a negative of the second, respectively. This means that when the first joint moves towards a negative angle, the second joint moves towards a positive angle. On the 5th second we can observe that the error value of the second joint is half the value of the first one, which is mounted on the robot body.

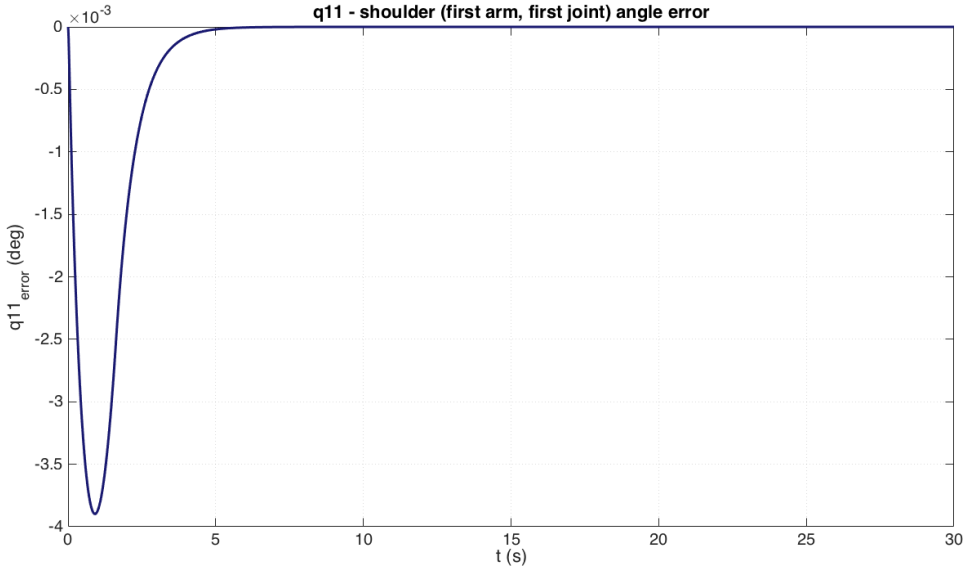


Figure 6.13: Angle displacement of the COM of the 1^{st} link of the 1^{st} manipulator.

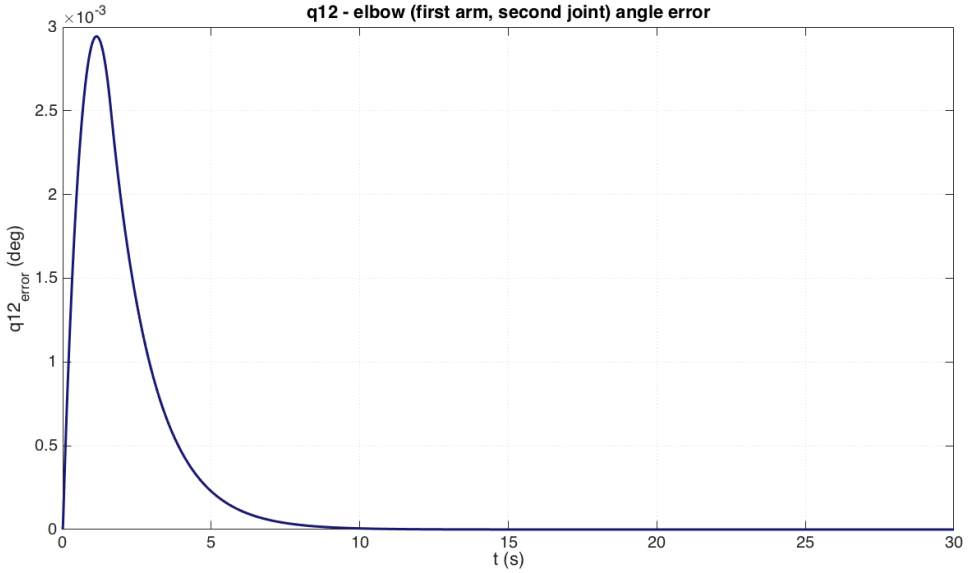


Figure 6.14: Angle displacement of the COM of the 2^{nd} link of the 1^{st} manipulator.

Similar observations can be made about the joint displacement errors of the second manipulator -Figures 6.15 and 6.16, with the difference that the second joint displacement error shows an acute overshoot in the beginning of motion, as observed for the same state displacement in Figure 6.9. This happened because the robot starts moving suddenly in the direction opposite where the second manipulator is mounted (on the robot), thus the sudden change in the variables' state.

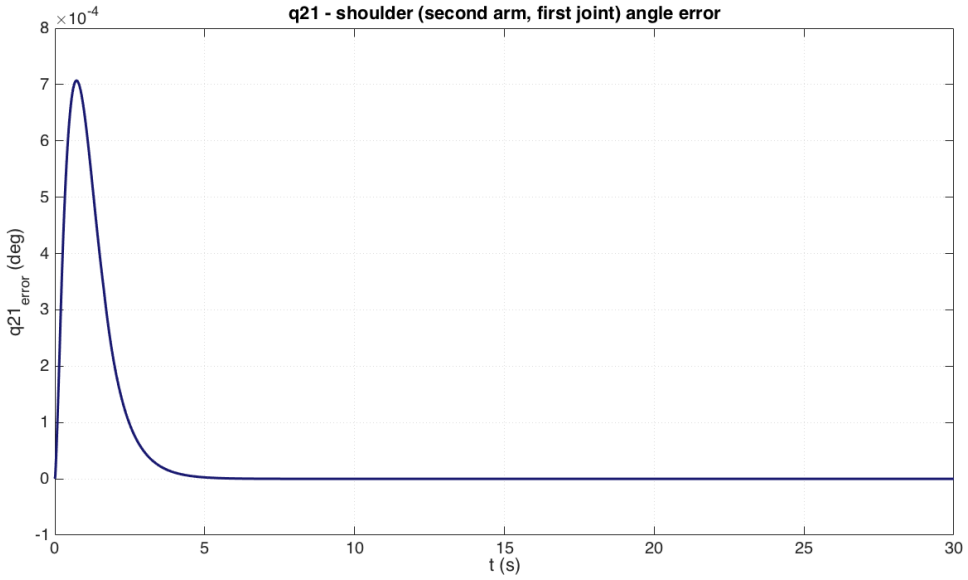


Figure 6.15: Angle displacement of the COM of the 1^{st} link of the 2^{nd} manipulator.

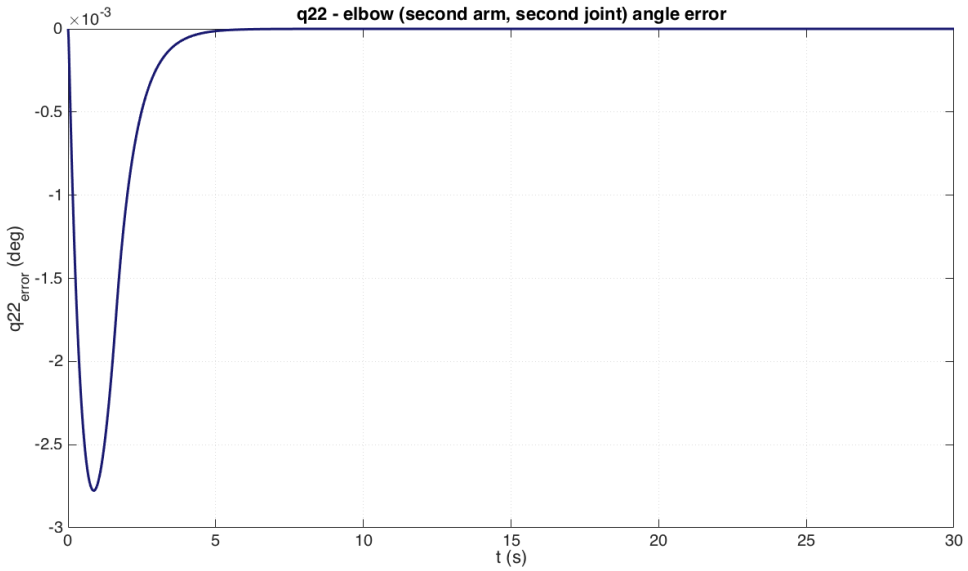


Figure 6.16: Angle displacement of the COM of the 2^{nd} link of the 2^{nd} manipulator.

Figure 6.17 shows that the error of the arms' links is insignificant, while the x and y position variables move smoothly towards zero, in the effort of covering the initial distance to the desired position. This planar motion of the robot's body however affects the orientation of the robot, the peak of which seems to take place on the 5th second and then goes back to zero on the 10^{th} second. No residual errors are left in any of the states, which means that the controller is working as expected.

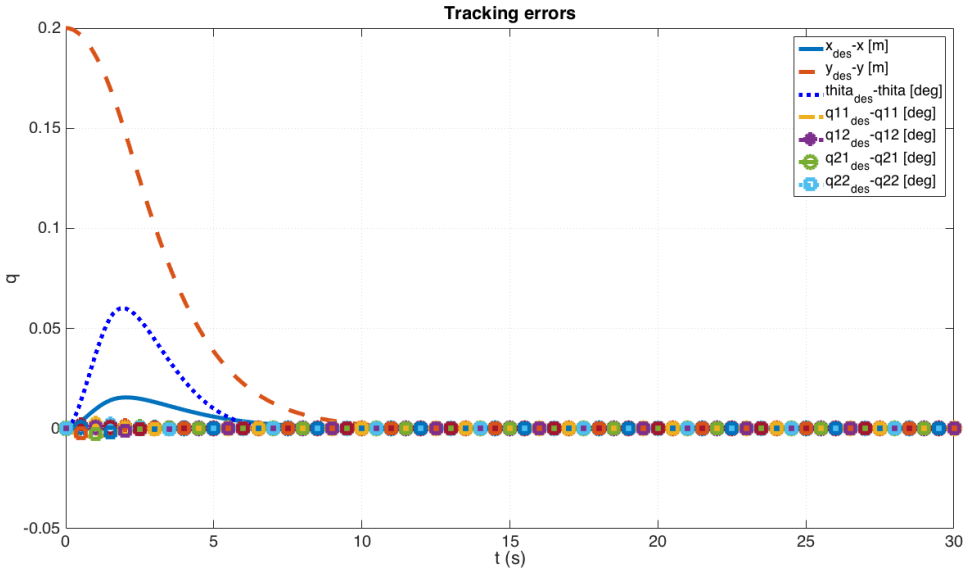


Figure 6.17: The errors of the variables' states compared to the desired values, in respect to time.

The same observations can be made in the Figure 6.18, where the velocity of the robot base on the x and y axis increase steadily and after the second second start decreasing until they are zero, on the 10^{th} second. At the same time, the angle velocity has the same behaviour with a triple rate of decrease, then increases to 0.1deg/s to go back to zero shortly after the 10^{th} second. The arms show to have insignificant velocity throughout the motion of the robot.

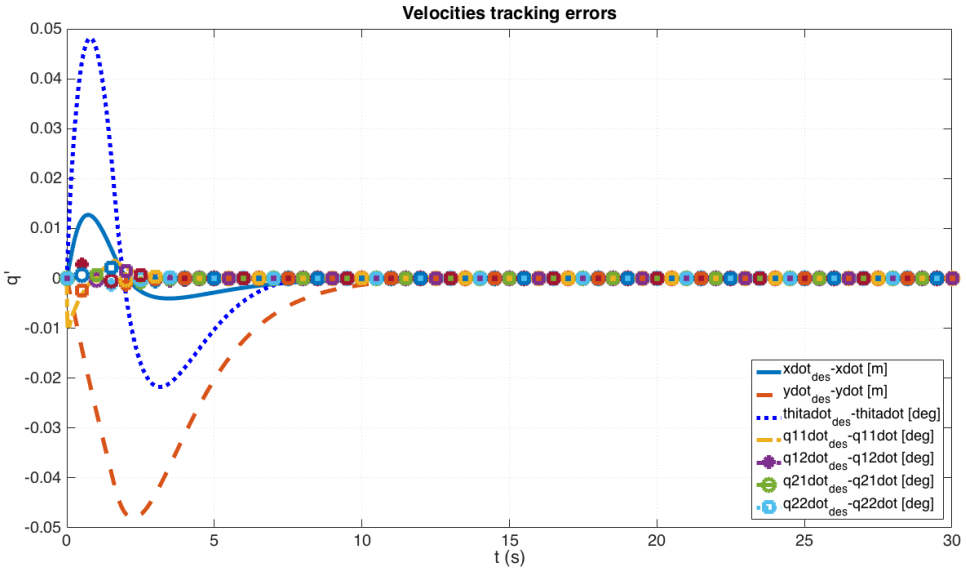


Figure 6.18: The robot's states' velocity in respect to time.

In Figure 6.19 we can see the values produced by the controller, to be sent to the actuators, based on the displacement needs of each of the robot's states, in every time interval. Based on the controllers' calculations, the first set of thrusters needs to give 3N of force in the beginning of the robot's motion, -2N and 0.7N the second and third thruster respectively. The reaction wheel is producing torque less than 0.5Nm, while the arm's motors shall produce zero torque, since the control command implies that the arms does not move at all.

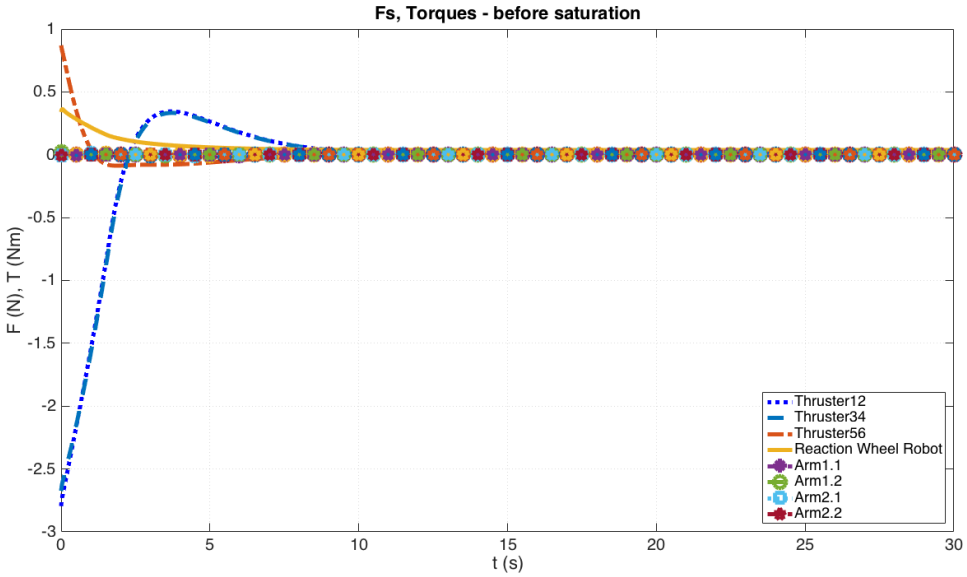


Figure 6.19: The forces and torques needed to be produced by the actuators, as calculated by the controller.

Figure 6.20 shows how the forces and torques produced by the actuators actually change the forces that are eventually delivered to the robot, after the imposed saturation limits in the model. The first and second set of thrusters operate in their maximum value for the first seconds of motion, in opposite directions, to produce the required thrust to move the robot to the required direction, while the third thruster starts operating from its maximum value to gradually decrease to zero by the 10^{th} second. The reaction wheel seems to deliver very low torque, while the arms' servo motors none.

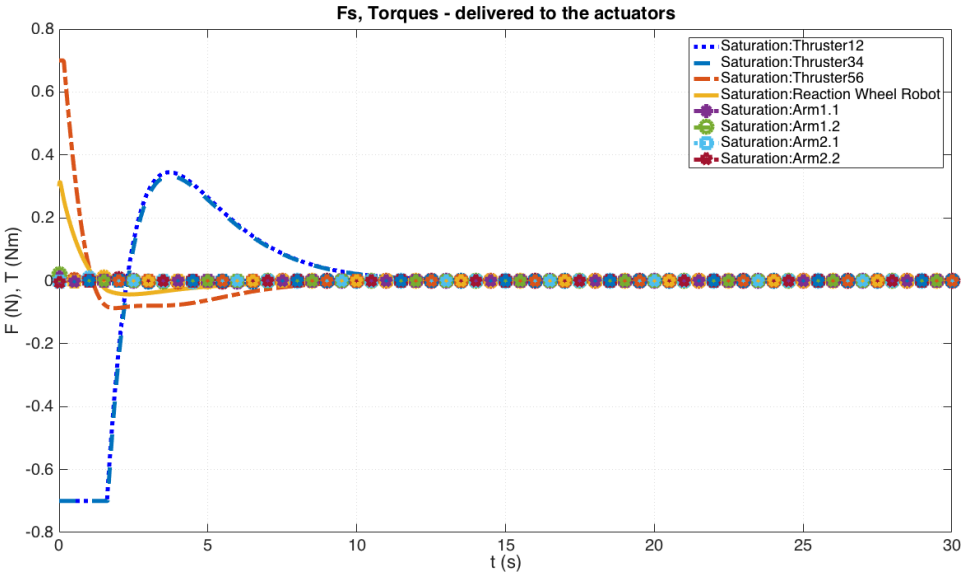


Figure 6.20: The required force and torque values delivered to the actuators, after the saturation of the thrusters and motors.

6.1.2.2 Discussion

The system responds to the control command, reaching its desired final state, with no residual errors. The saturation imposed on the actuators has insignificant modification on the forces and toques initially asked to deliver from the controller. The elements that stay stable have a small overshoot in the beginning of the motion, as a result of the sudden change of the system's state: the robot suddenly gains velocity, and as result the orientation is affected so it has to go back to zero. At the same time, the arms -whose COM are in distance from the system's COM- move to the opposite direction of the motion of the base of the robot, because of their inertia and the lack of friction. This overshoot is short and is eliminated in less than the $1/7^{th}$ of the total time of motion.

6.2 Target Chase

This controller focuses on the position of the EE of the arms, instead of the system's COM. Since the robot's planar position state variables are defined in respect to the system's COM, essentially, with the use of inverse kinematics,

the position of the arms' EE is being calculated in order to achieve the high level objective of reaching and grasping a moving target. Thus, this case targets scenarios of on-orbit servicing by space robots, or grasping scenarios; a robotic mechanism is commanded to grasp an object on orbit and continue moving with the same speed, in orbit.

The planner calculates the time of grasping, gives the command to the robot to gain the required orientation (based on the optimisation criteria of minimum fuel consumption, as explained in chapter 5.2.0.1) and start approaching the moving target. When $t = t_{meet}$, the robot's EE start grasping the target object with a predefined manipulators' configuration. The second links of the arms are to have a final angle of 45° at the competition of the grasping and the start of the joint motion of the chaser and target.

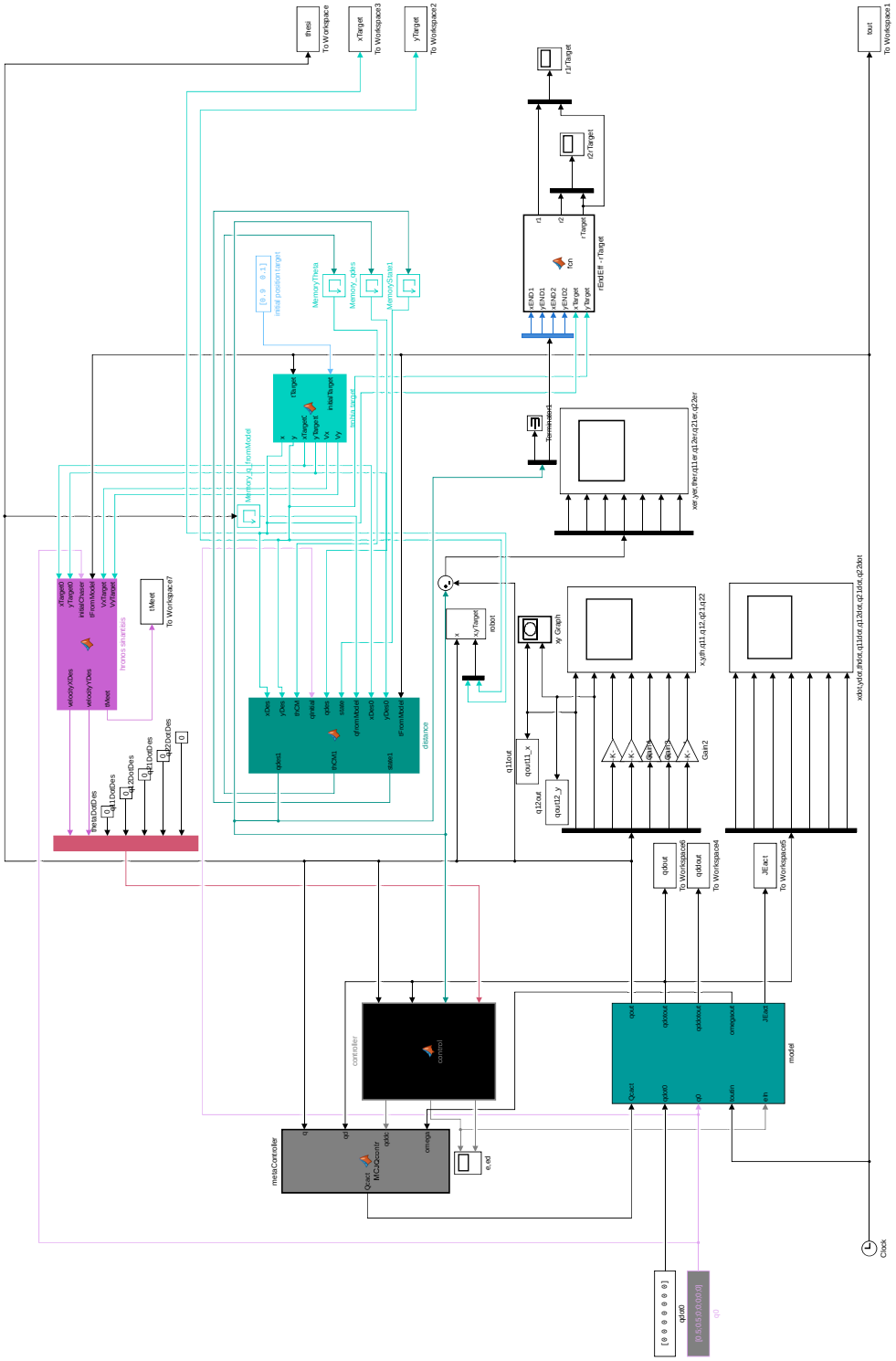


Figure 6.21: The Simulink Model of the Target Chase with controller.

6.2.1 Grasping the target without the use of the manipulators' workspace

This first level of this scenario investigates whether the arms' EE can reach the desired position, using inverse kinematics with the position of the system's COM, without planning (trajectory calculation and WS). The robot is instructed to gain orientation facing the target without planar displacement, then approach the target spacecraft and when the latter is within a certain distance from the manipulators' EE, to grasp the object and continue moving in-orbit, with the same velocity of the target.

The chaser starts with an initial position of (0.5,0.5) while the target is at (0.9,0.1). The target's initial velocity is equal to $V_{t,x}(0) = V_{t,y}(0) = 5 \cdot 10^{-3}$ m/s. The approach criteria is not linked to the robot's EE calculated WS, but equal to 0.02 m. The robot shall start moving towards the target when the desired orientation from the current robot orientation is less than 0.01 rad.

The command is a set-point command and the controller is a model-based PD controller.

Based on experimental observations, the settling time was chosen to be equal to $t_s = 8.5$ s and $\zeta = 1$, as explained in 4.13. Therefore $\omega = 0.7059$, $K_P = 0.4983$ and $K_D = 1.4118$.

After the weights multiplication with K_P and K_D, the position and velocity gains matrices result to:

$$K_P = \begin{bmatrix} 0.5979 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.5979 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 9.9654 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 24.9135 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 24.9135 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 24.9135 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 24.9135 \end{bmatrix}$$

$$K_D = \begin{bmatrix} 2.8235 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2.8235 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 70.5882 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 141.1765 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 141.1765 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 141.1765 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 141.1765 \end{bmatrix}$$

The results of the simulation are shown below.

6.2.1.1 Results

Figure 6.22 shows that the robot's base does not start moving up until it reaches the desired orientation, and then moves towards the position of the target and then follows the target's trajectory.

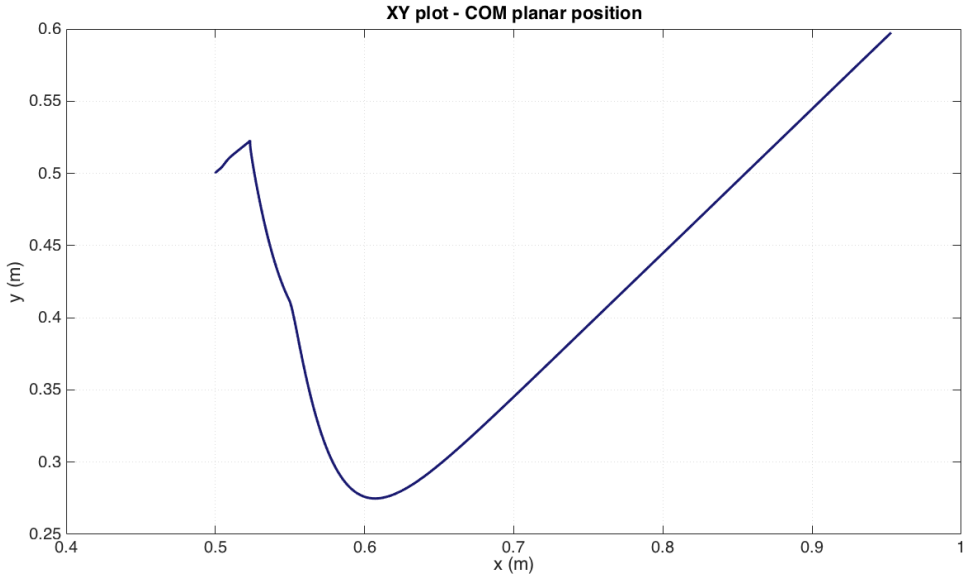
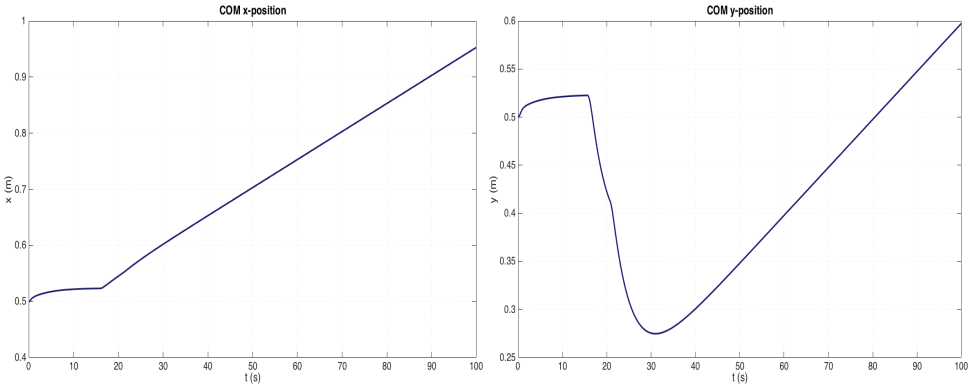


Figure 6.22: The robot's COM planar displacement.

Figure 6.23(a) displays the x-displacement of the robot, which approaches the target and then moves along with the object, after grasping. Figure 6.23(b)

shows more clearly the course of the chaser towards the target and then their common y-trajectory.



(a) The position of the robot's COM on the x-axis.

(b) The position of the robot's COM on the y-axis.

Figure 6.23: The planar position of the robot's COM

In Figure 6.24 we can see the robot's orientation decreasing to -34° in order to gain the initial orientation needed so that when it starts approaching the target, it will need to move on a straight line to grasp the target. Right after the grasping takes place (peak of the graph), the angle starts decreasing until it becomes zero, by the 50^{th} second.

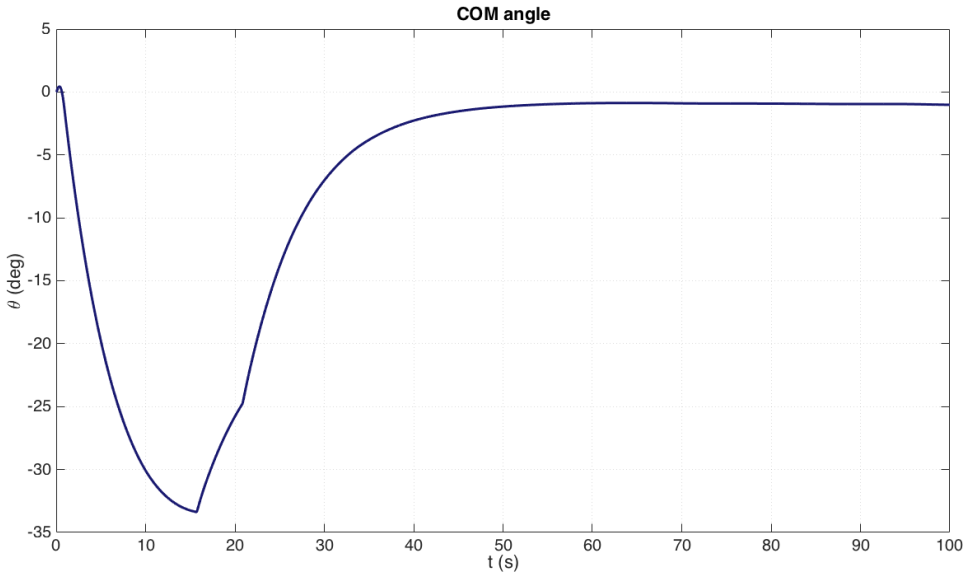


Figure 6.24: The robot's orientation with respect to the absolute coordinate system.

Figure 6.25 shows the displacement of the first link's COM of the left arm, where the initial motion of the robot creates a disturbance, in the first moments of motion, thus the sudden change in the state of the first link. The latter goes back to the initial position, whilst has some last disturbances during the grasping of the target by the second link (as shown by the two peaks in the graph) before the 50^{th} second, when the link reaches its steady-state.

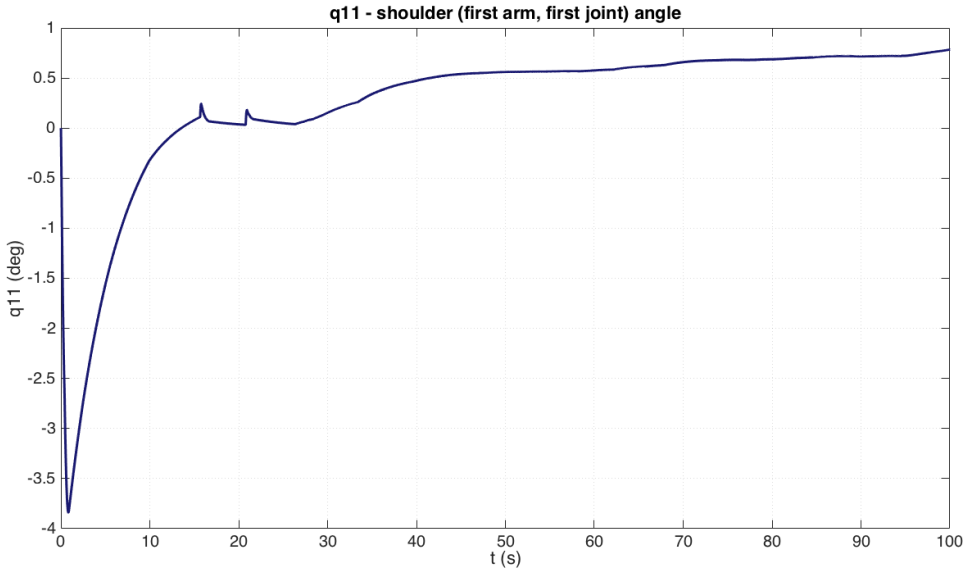


Figure 6.25: The angular displacement of the first joint of the first arm.

Figure 6.26 shows the displacement of the second link of the left arm, which, after a small displacement in the beginning of motion, transferred by the disturbance of the first link, has zero displacement until it attempts to grasp the target and smoothly reaches the desired steady state of $45°$.

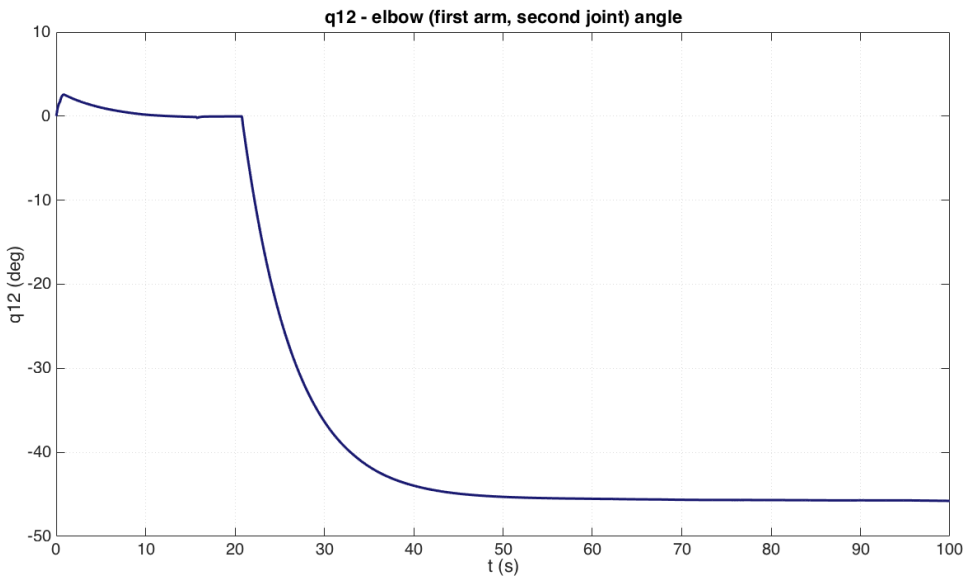


Figure 6.26: The angular displacement of the second joint of the first arm.

Similar behavior can be observed at the right manipulator, which momentarily oscillates in the beginning of motion and then again during the motion of the second link of the arm, which shall move by $45°$ and attempt to grasp the target.

This momentum is transferred to the first link as we see by the small peaks at the graph, before reaching steady state.

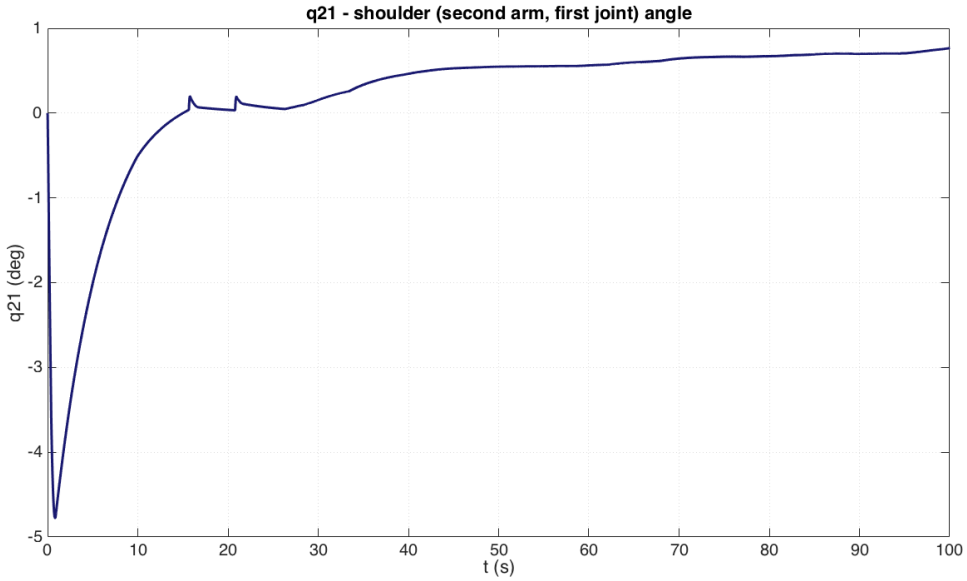


Figure 6.27: The angular displacement of the first joint of the second arm.

The second joint oscillates in the beginning of motion as a result of the motion of the first link, which moves because of the momentum transferred by the motion of the robot. The link then smoothly moves to its steady state, when the control command is given, to grasp the object while having a 45° angle.

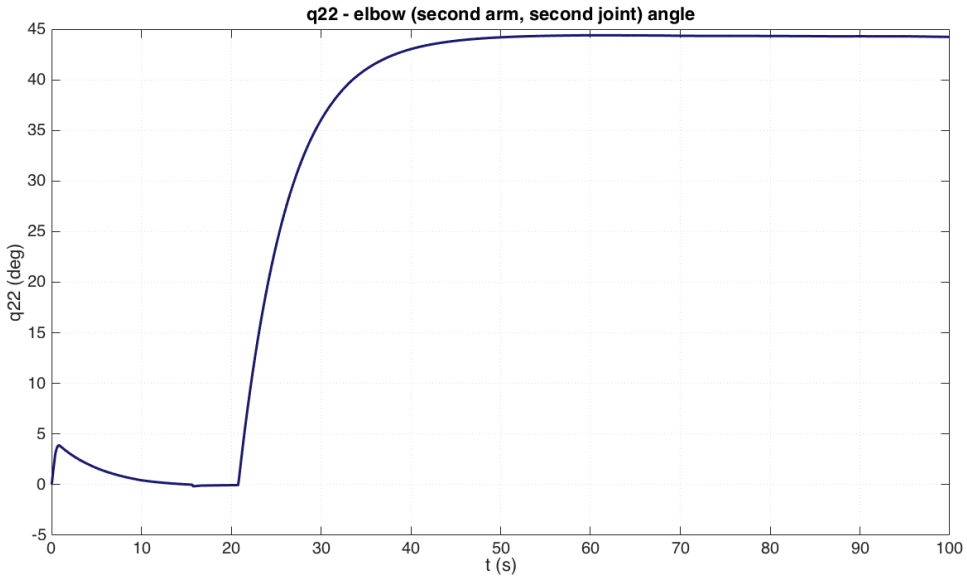


Figure 6.28: The angular displacement of the second joint of the second arm.

Figures 6.29, 6.30 and 6.31 show the tracking errors, which oscillate while the robot is attempting to perform grasping, in two consecutive stages.

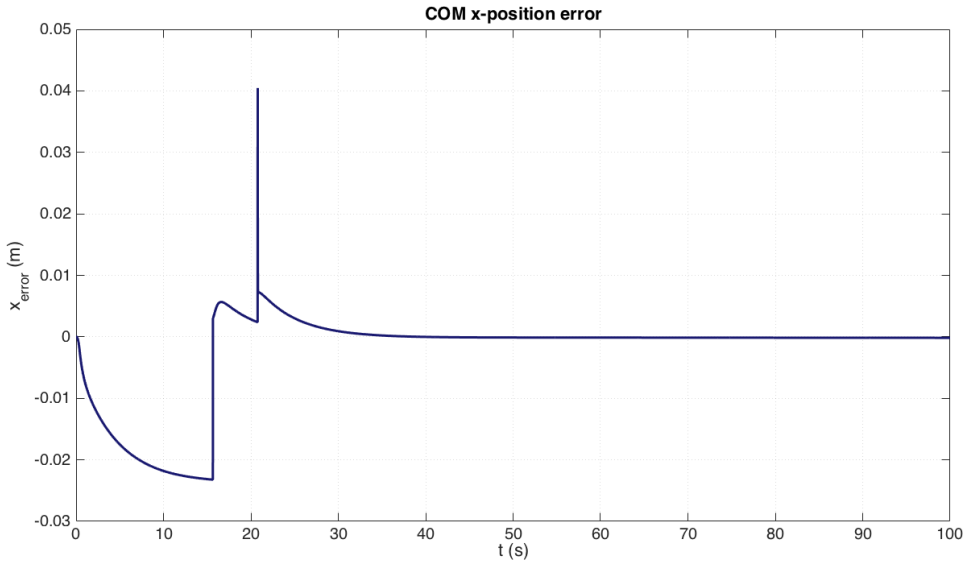


Figure 6.29: The error of the x position of the robot's COM.

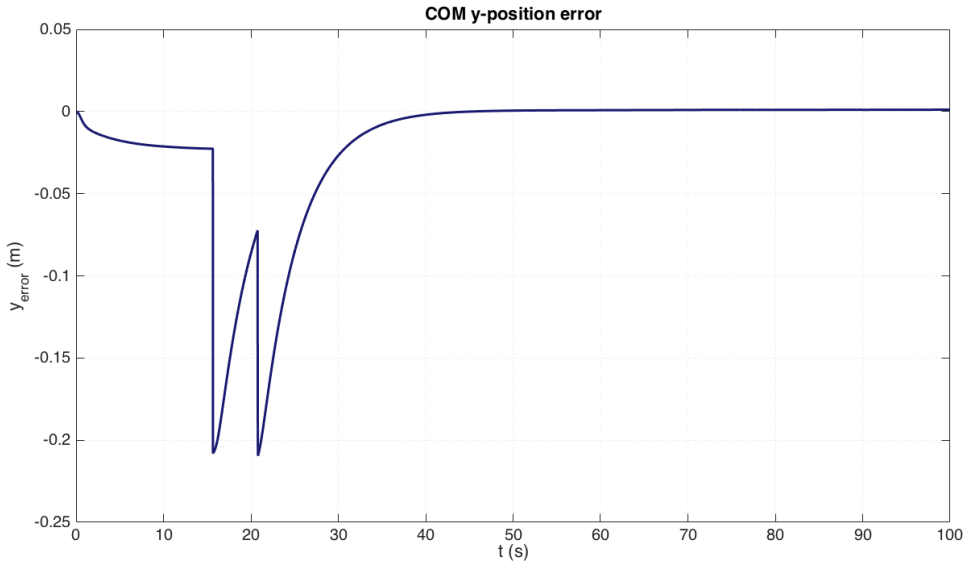


Figure 6.30: The error of the y position of the robot's COM.

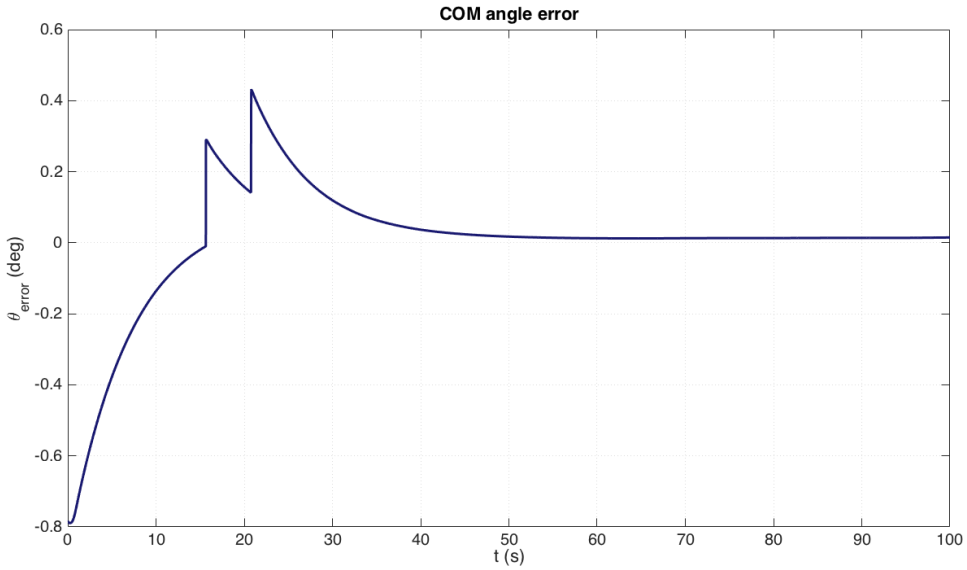


Figure 6.31: The error tracking for the orientation of the robot's base.

We can see the same response by looking at the errors of the manipulators' states, in Figures 6.32, 6.33, 6.34 and 6.35. The first joint of the left arm momentarily has an error of 0.065° at the beginning of motion -vibration caused by the motion of the body of the robot, and then has tiny negative errors during grasping. The second joint of the left arm has a small negative error in the beginning of motion (momentum transfer from the first link) and then at the 20^{th} second has an acute error which goes back to zero right after.

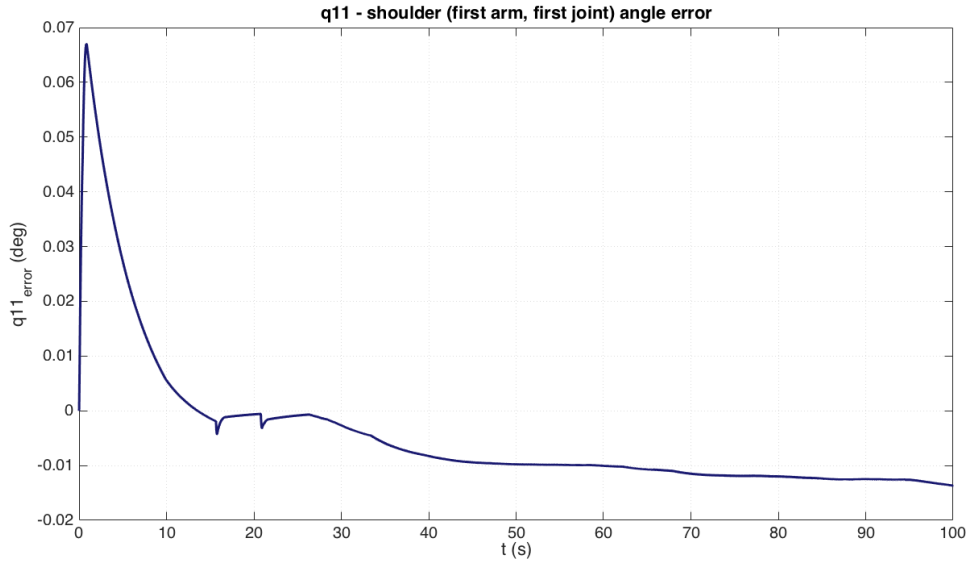


Figure 6.32: The error tracking of the first joint of the left arm.

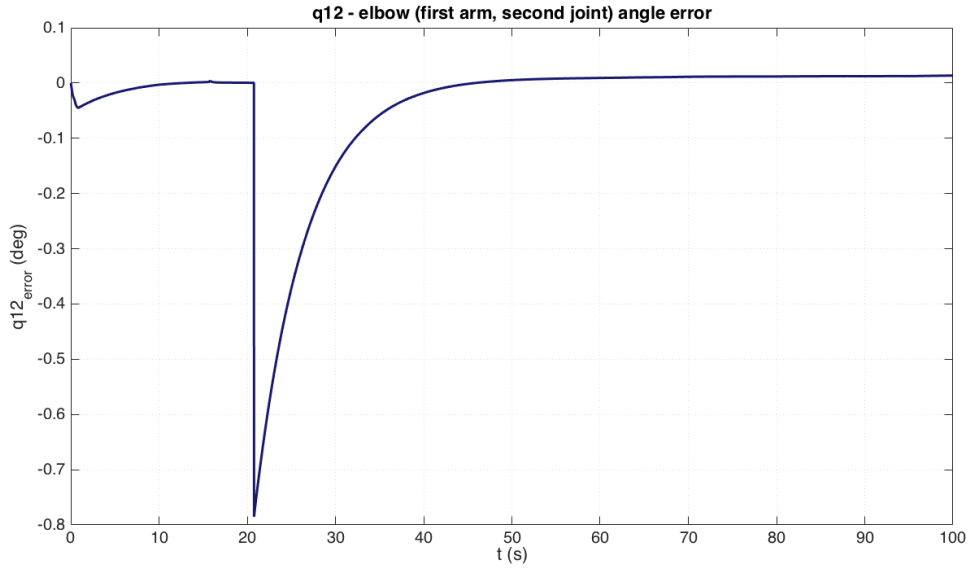


Figure 6.33: The error tracking of the second joint of the left arm.

The same response is observed for the right manipulator, only that, the second link has a negative error of the same magnitude as of the left manipulator, since the right arm is moving in the opposite direction than the left one.

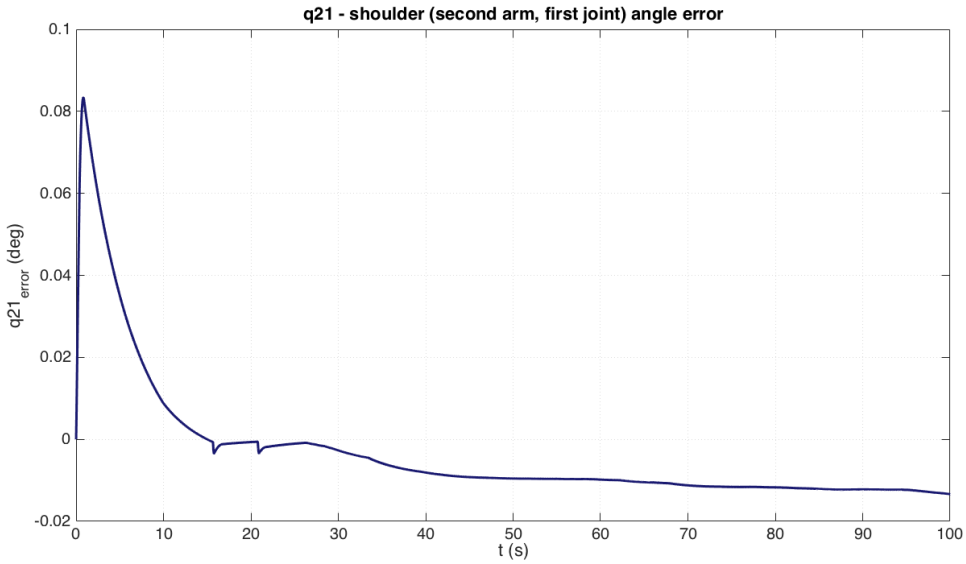


Figure 6.34: The error of the first joint of the right arm.

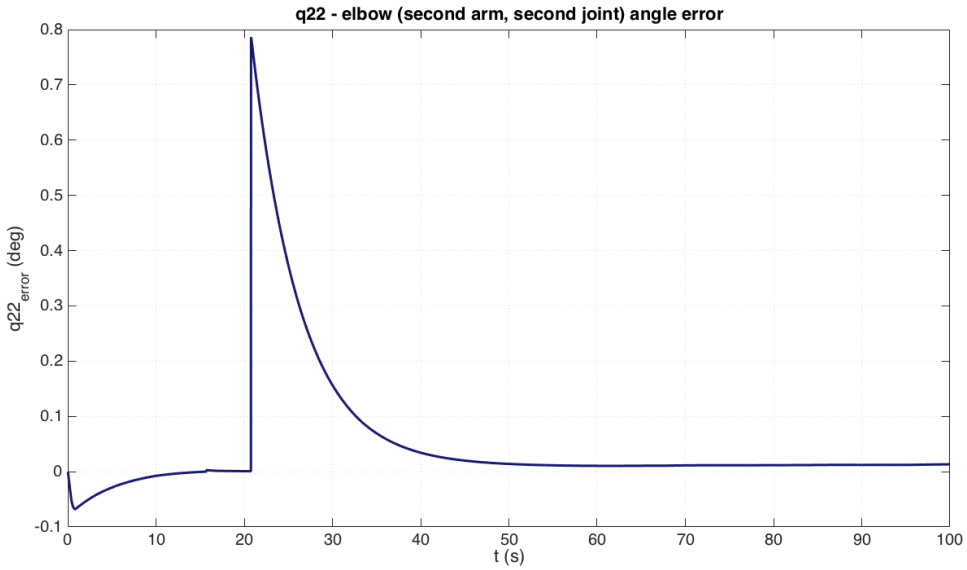


Figure 6.35: The error f the first joint of the right arm.

Figure 6.36 shows the tracking errors compared to their desired position, in the absolute coordinate reference system. We can observe the position errors of the second link of each arm have diametrically opposite errors, since the command for their tracking is received, they have the same position state, and the desired value is the same. The robot's angle state and y-position also show errors of the same magnitude and of the same form, from the 15^{th} to the 20^{th} second, since the robot is moving on the y-axis to approach the robot, while its x-position does not change significantly. All the states have zero residual errors.

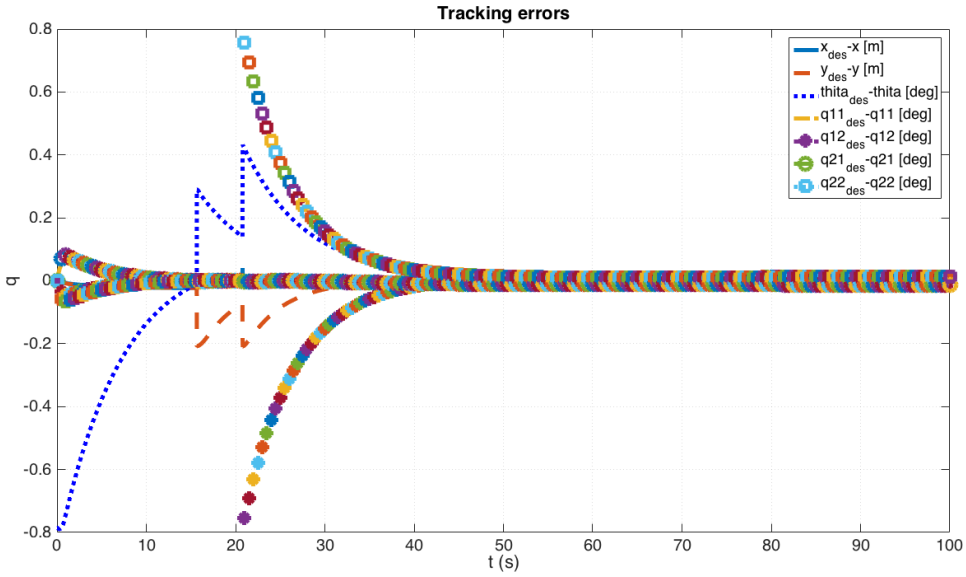


Figure 6.36: The errors tracking of all DOFs for the controller shown in Figure 6.21.

The graph of the errors of the states' velocity has the same form, except that in the beginning of motion the robot's angle and y-position seem to change rapidly, thus the oscillation in the Figure 6.37.

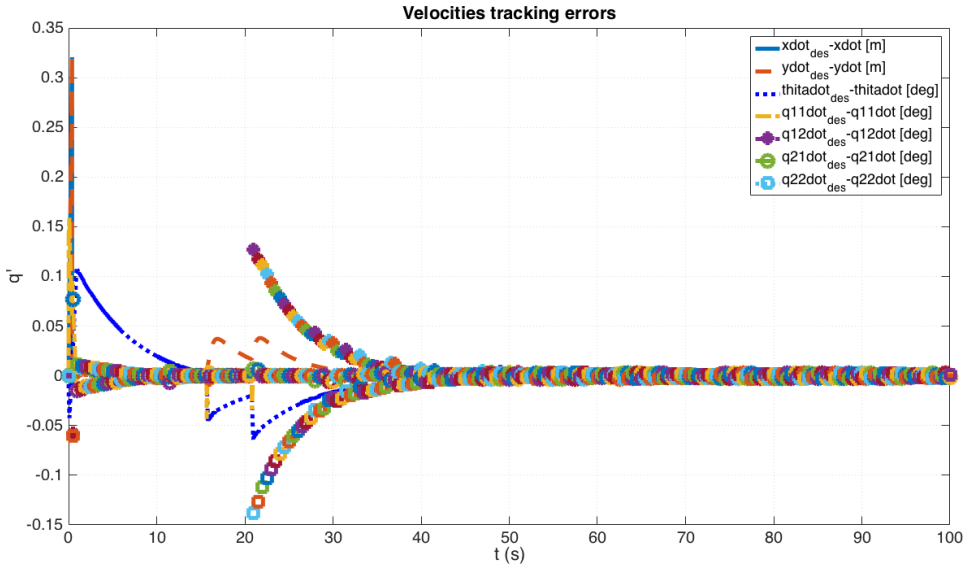


Figure 6.37: The velocity tracking errors of all the variables.

Figure 6.38 shows the forces and torques command values, as calculated by the controller, to be sent to the robot's actuators, in order for all the states to reach their desired position, at each given moment. We can see that the first thruster produces 27N and the second thruster -18N, while the third 5N; the reaction wheel applies 12Nm, at the beginning of the move. The combination of the first and second thruster's forces shall move the robot to the desired orientation, with the use of the RW. During grasping, the second and third thruster, together with the RW are producing significant thrust/torque.

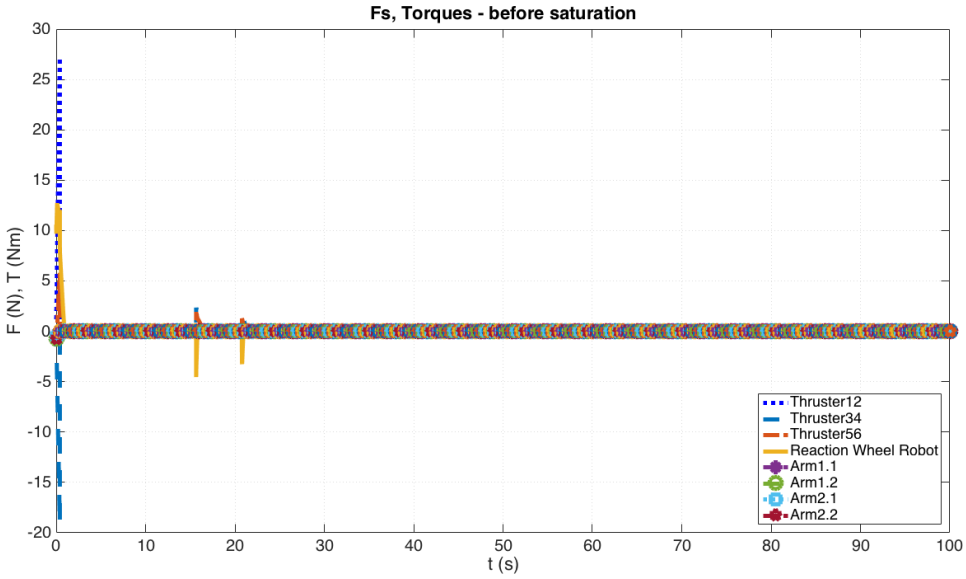


Figure 6.38: The actuators and torques, as calculated by the controller, to perform the desired move for all the seven DOFs.

However the actuators cannot deliver the thrust/torque as calculated by the controller. The forces/torques actually delivered to the system, are shown in Figure 6.39. Here we can see the effect of the controller more clearly, since the thrusters are to operate momentarily, with a pulse of 0.7/-0.7 N, while the RW is operating from the beginning of the motion until the end, within the motor's saturation limits.

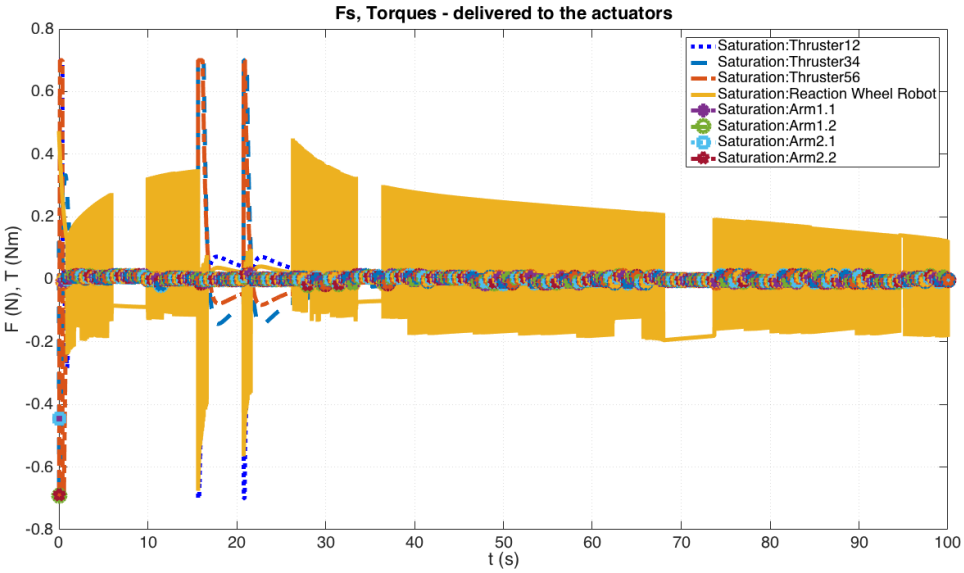


Figure 6.39: The forces and torques actually delivered by the actuators, bounded by the physical constraints of the system -saturation of the motors and thrusters.

6.2.1.2 Dsicussion

The system reached its desired state with no residual error. The saturation has a significant effect on the torques and forces, especially on the RW torque, which then produces torque in more continuous way. The first link of each arm has significant oscillation, especially at the beginning of the move, since they are mounted on the robot's frame -which causes the vibration.

6.2.2 Target Chase with Trajectory Planning

This model constitutes an advanced version of the controller presented in the previous section. The robot's EE is to be driven to a specific location, in order to catch moving object. When near the object, the arms are commanded to grasp the moving object with the optimal configuration:

$$q_{12} = q_{22} = 45^{\circ} \tag{6.2}$$

and follow its orbit.

The planner is making use of the robot's EE path independent reachable WS to calculate when to initiate the grasping stage. The latter is combined with the calculation of the t_{meet}, when the robot should have grasped the object and start its on-orbit motion, along with the target.

The robot's COM is located at (0.3,0.3) initially, while the target starts from (0.6,0.8), with a velocity of $V_t(0) = 5^{-3}m/s$. Figure 6.40 presents the Simulink representation of the full control system, including the planner, the controller, the meta-controller and the virtual representation dynamic model of the physical system.

The control command requires that the base moves by 0.2m on the X and Y axis respectively. The robot's COM initial position is equal to:
$(x_{COM_{init}}, y_{COM_{init}}) = (0.8, 1.0)$
and is commanded to reach the position:
$(x_{COM_{final}}, y_{COM_{final}}) = (1, 1.2)$.

The command is a set-point command and the controller is a model-based PD controller.

Based on experimental observations, the settling time was chosen to be equal to $t_s = 8.5$ s and $\zeta = 1$, as explained in 4.13. Therefore $\omega = 0.7059$, $K_P = 0.4983$ and $K_D = 1.4118$.

After the weights multiplication with K_P and K_D, the position and velocity gains matrices result to:

$$K_P = \begin{bmatrix} 0.5979 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.5979 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 9.9654 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 24.9135 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 24.9135 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 24.9135 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 24.9135 \end{bmatrix}$$

$$K_D = \begin{bmatrix} 2.8235 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2.8235 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 70.5882 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 141.1765 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 141.1765 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 141.1765 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 141.1765 \end{bmatrix}$$

The graphs of the model's response at the simulation are included below.

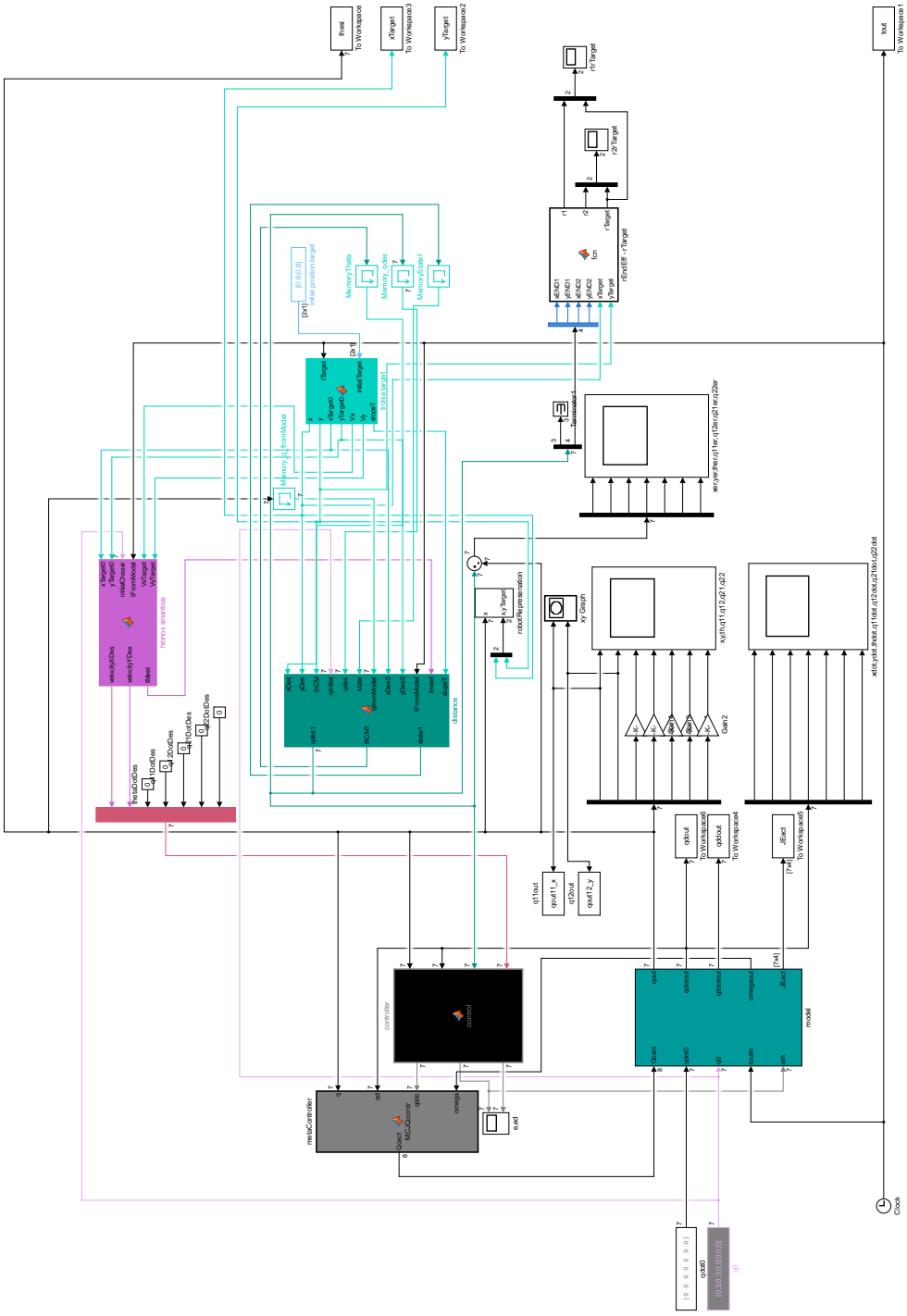


Figure 6.40: The Simulink Model of the Target Chaser with controller and trajectory planning.

6.2.2.1 Results

Figure 6.41 shows that there is no planar displacement in the beginning of the move, since the robot was only turning towards the object, and then started moving rapidly towards it until (0.4,1), when the grasping took place and continued moving smoothly with the target object.

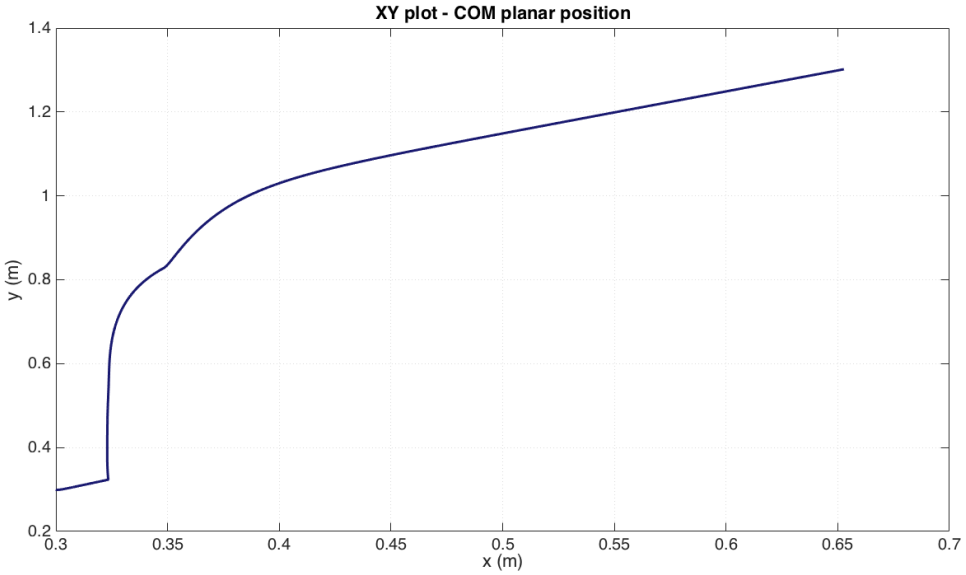
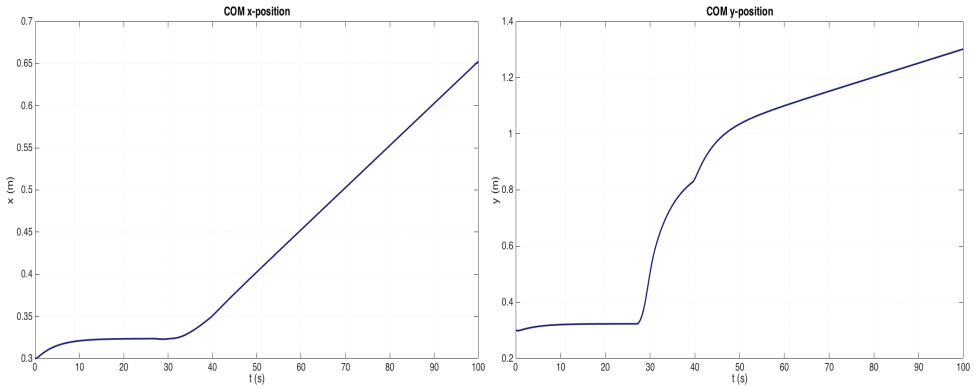


Figure 6.41: The robot's COM planar motion in respect to time.

In Figure 6.42(a) we can see the robot's motion along the x-axis, which is similar to its motion along the y-axis; however the vertical motion is mode rapid and less smooth than at the x-axis. This is because the robot needs to traverse a larger distance (0.6m) on the y-direction at the same time it needs to traverse half the distance (0.3m) to the target on the x-axis.



(a) (b)

Figure 6.42: The planar position of the robot's COM. (a) The position of the robot's COM on the x-axis. (b) The position of the robot's COM on the y-axis.

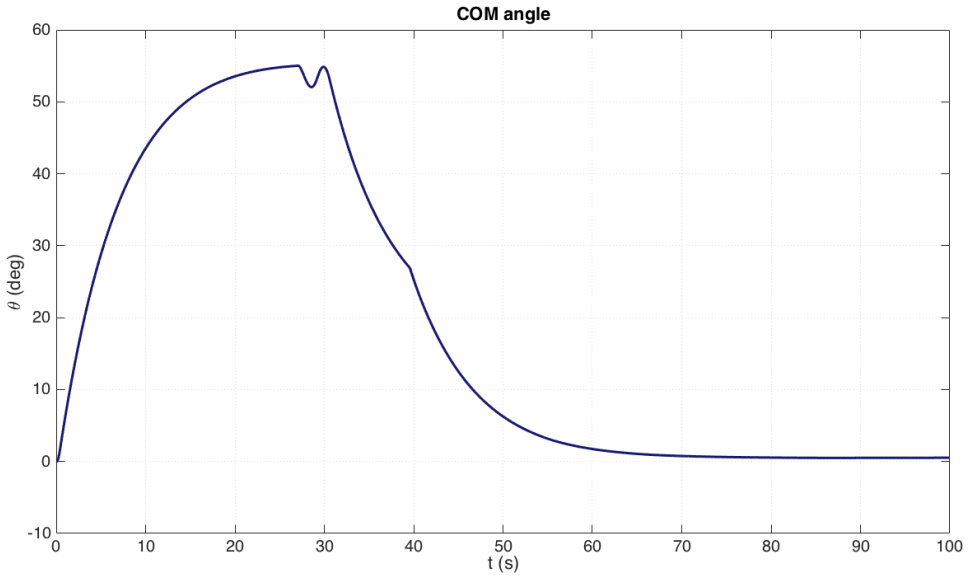


Figure 6.43: The robot's COM angle in the absolute coordinate system, in respect to time.

Figure 6.44 shows the angle displacement of the first link of the left arm, which reaches the value of 2° at the beginning of the motion, because of momentum transfer from the robot's body to the joint to which the first link is attached to, and mounted on the robot's body frame. It then goes slowly back to zero, until

the grasping command is given, when the planner realises the target is within the manipulators' EE WS and attempts to approach the target and initiate the grasping stage.

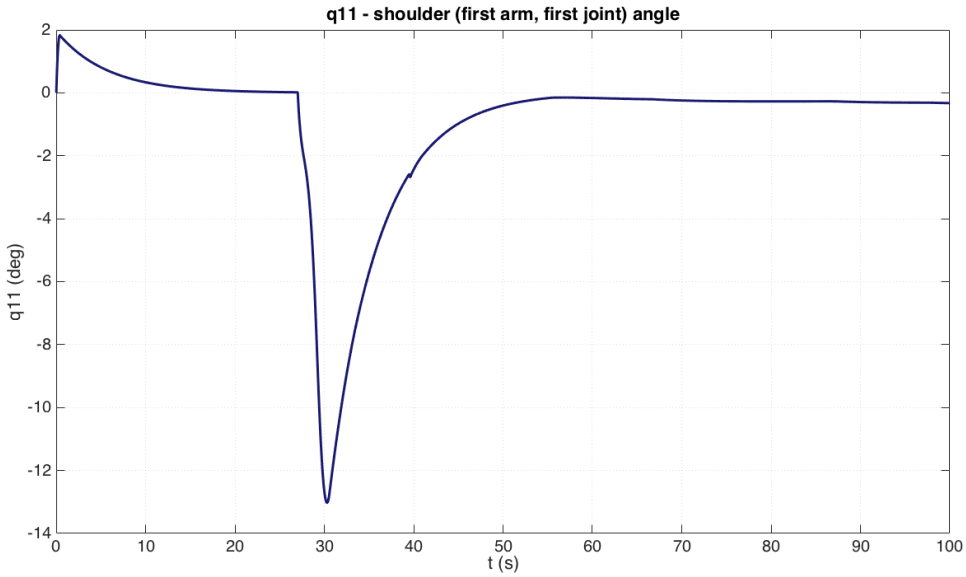


Figure 6.44: The angular position of the COM of the first link of the left manipulator, in respect to time.

Figure **??** shows that the second link of the left arm is affected by the disturbance of the first link in the beginning of motion, and has a small negative displacement (opposite to the first link's) and then is affected by the robot's sudden motion on the 30^{th} second, to start going towards its steady-state (-45°) and the grasping of the object from the 40^{th} until the 70^{th} second of motion.

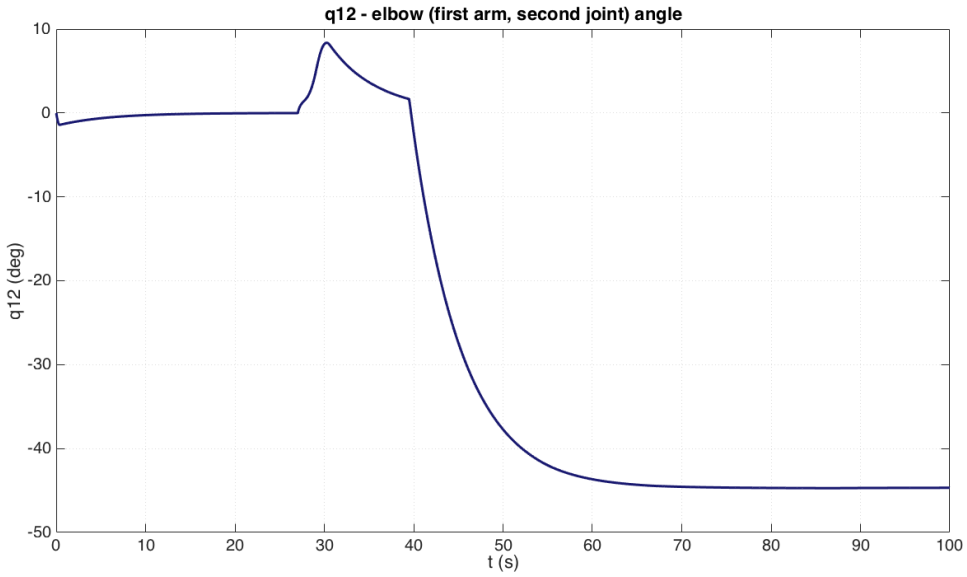


Figure 6.45: The angular position of the COM of the second link of the left manipulator, in respect to time.

The graph of the angular motion of the first link of the right arm (Figure6.46) shows the same response as the first link of the left arm, as shown in Figure 6.46.

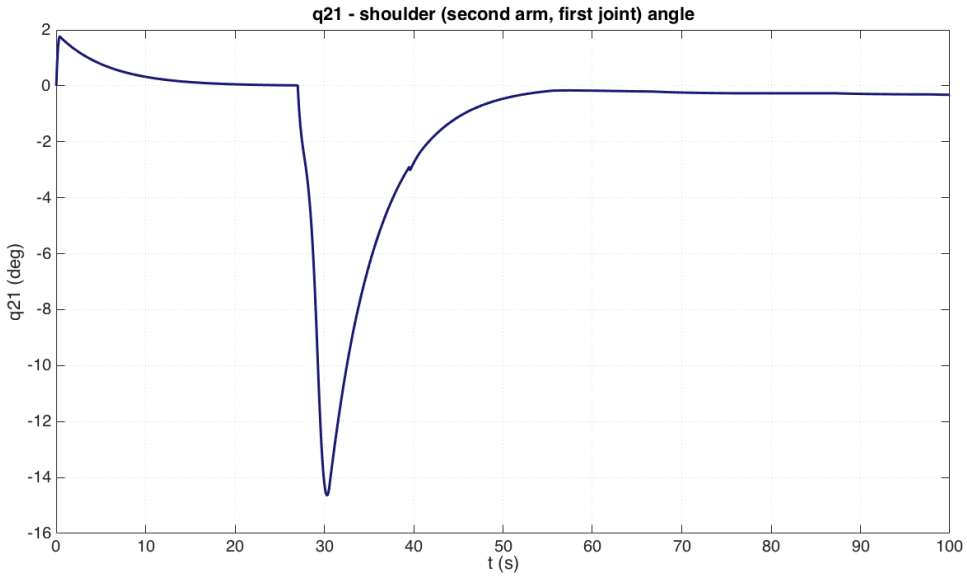


Figure 6.46: The response of the angular position of the first link of the right manipulator.

Likewise, the second link of the right arm has the same response to the control commands and the motion of other parts of the robot thus the peaks in Figure 6.47. The link however moves to the opposite side as the one of the left arm, so it reaches 45° by the 70^{th} second as well. The two joints are expected to move to opposite directions, since they are to grasp the object located between the EE of the two links.

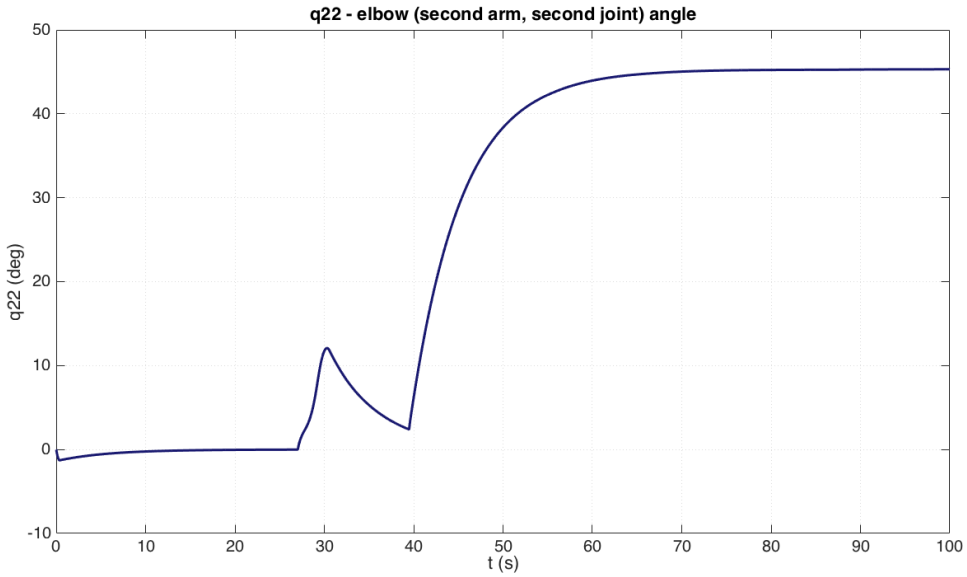


Figure 6.47: The response of the angular position of the second link of the right manipulator.

The errors in the x-position of the COM of the system indicate what was described above, by observing the responses of the system's states. The robot seems to be moving towards the target, and as soon as a control command is given, the robot seems to have a disturbance of -0.04 to 0.04 m, which is quickly eliminated by the 40^{th} second of motion.

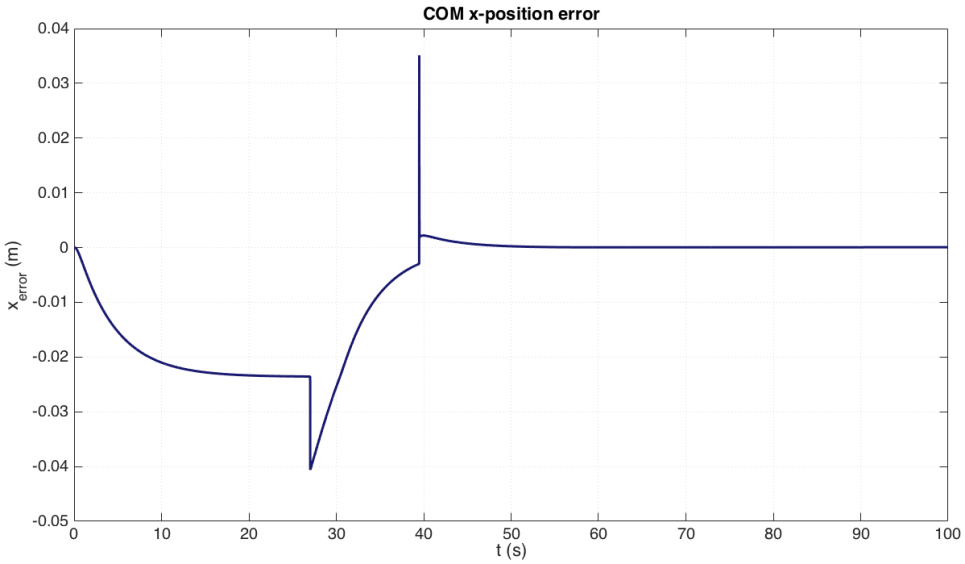


Figure 6.48: The errors tracking of the x-position of the robot base.

The same same can be observed in Figures 6.49 and 6.50, whereas the first links of the arms have a similar behavior, as shown in Figures 6.51 and 6.52.

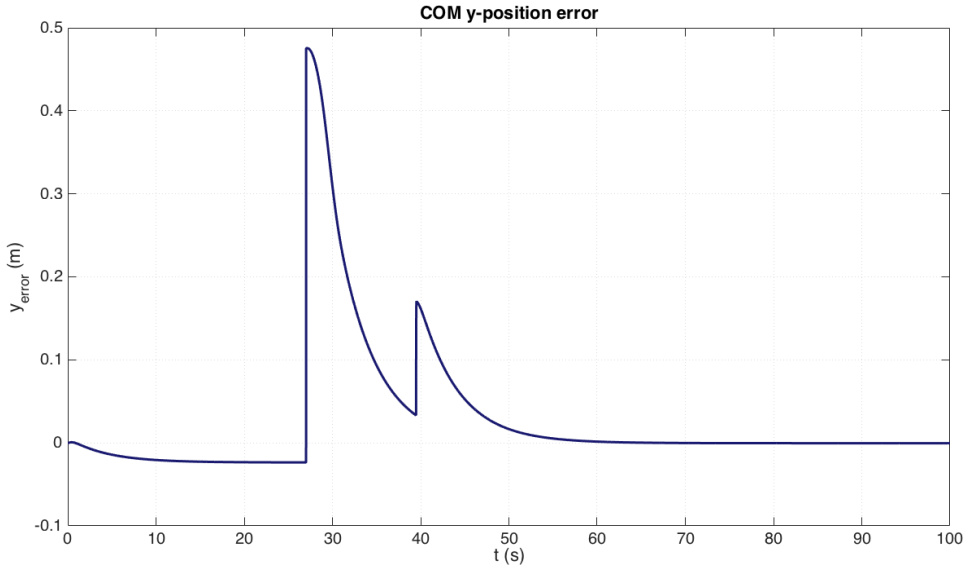


Figure 6.49: The errors tracking of the y-position of the robot base.

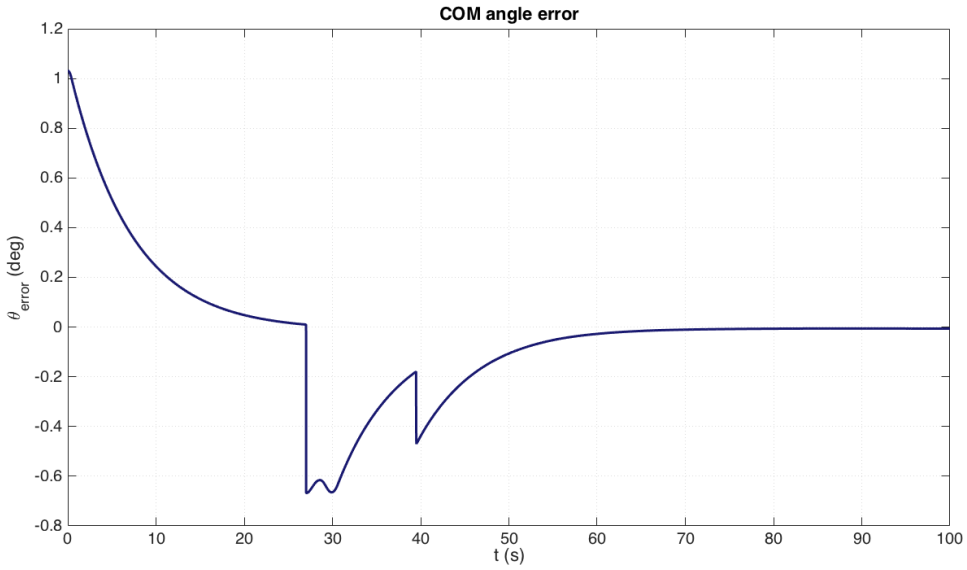


Figure 6.50: The errors tracking for the orientation of the robot.

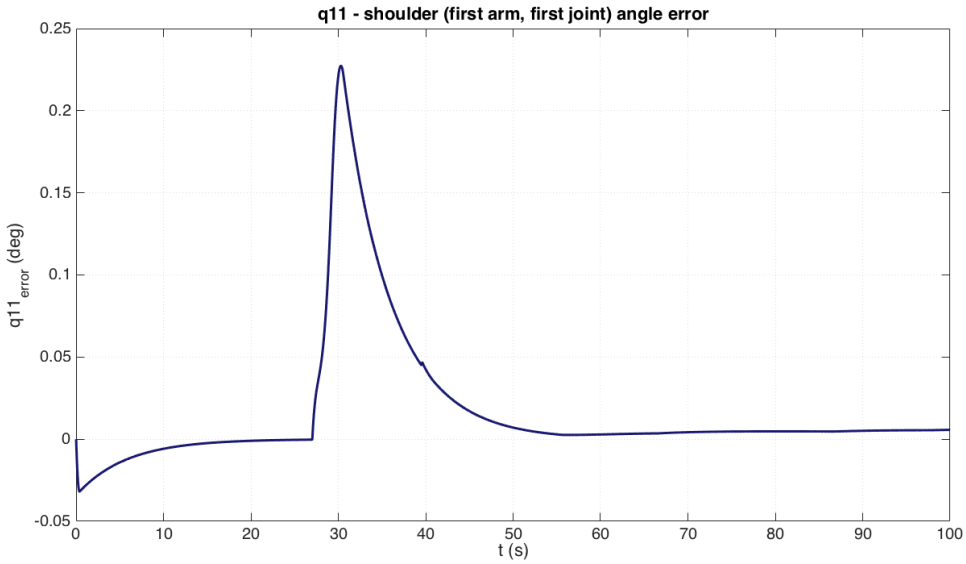


Figure 6.51: The errors tracking of the first link of the left arm.

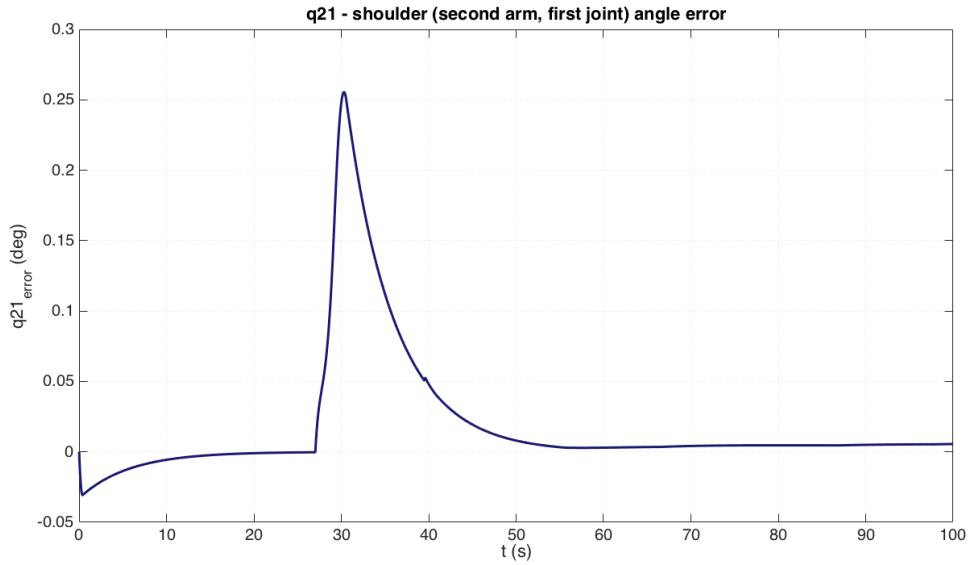


Figure 6.52: The errors tracking of the first link of the right arm.

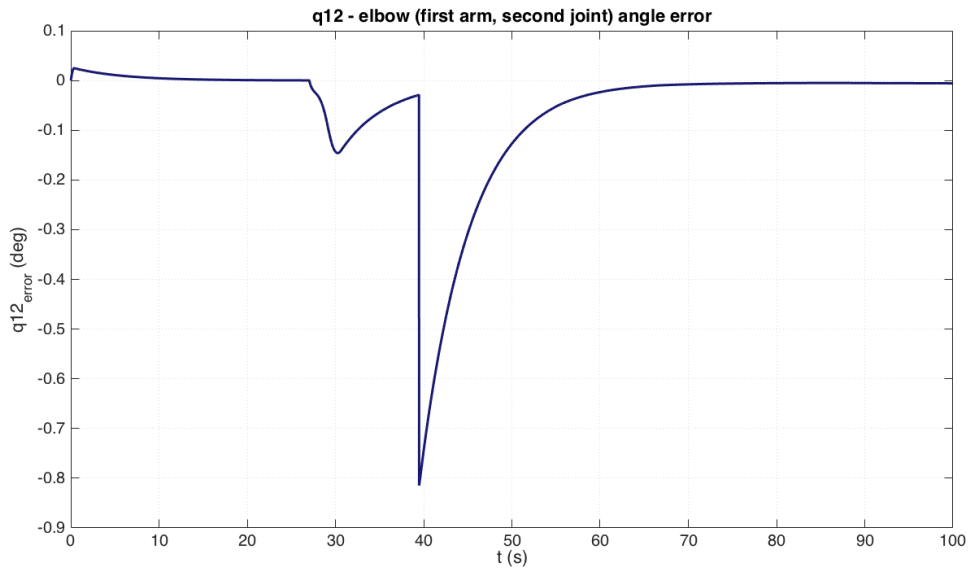


Figure 6.53: The errors tracking of the second link of the left arm.

Figures 6.53 and 6.54 show the disturbances transferred to the second links of the left arm and the second arm, which have a peak of $-0.8°$ and $0.8°$, respectively; the error however still moves to zero with no residual error.

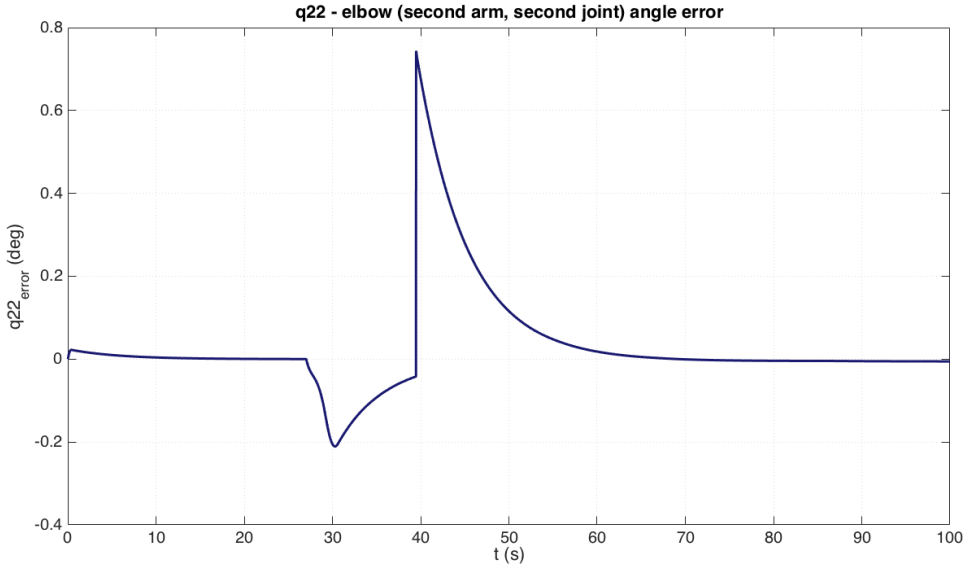


Figure 6.54: The errors tracking of the second link of the right arm.

As shown in Figure 6.55, no state error overpasses the value of 1, with the arms' links and the robot angle having the largest disturbance throughout time.

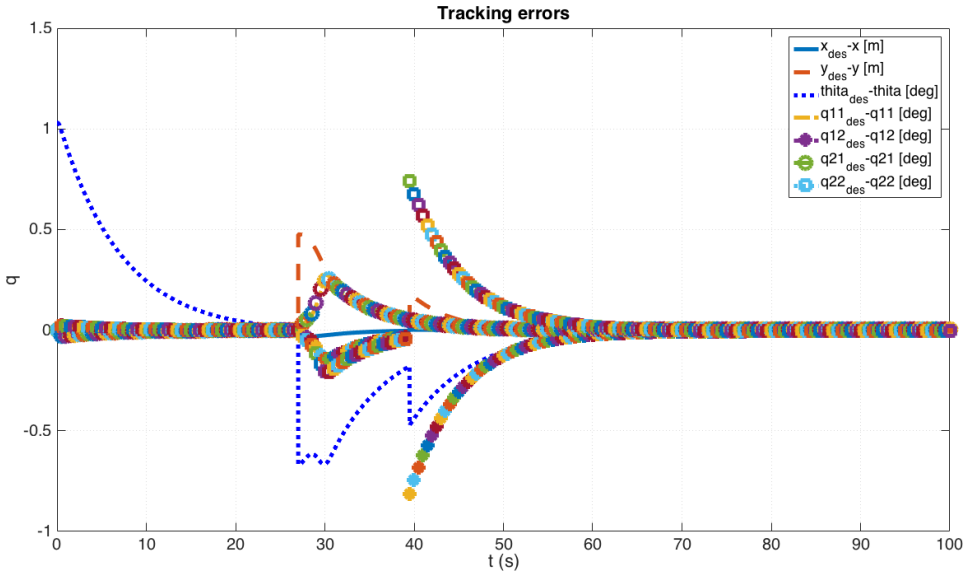


Figure 6.55: The errors tracking of all DOFs for the controller shown in Figure 6.40.

The errors in velocity show that the velocity might has a more unstable profile, nevertheless the magnitude of the errors is lower than the errors in position, as shown in Figure 6.56.

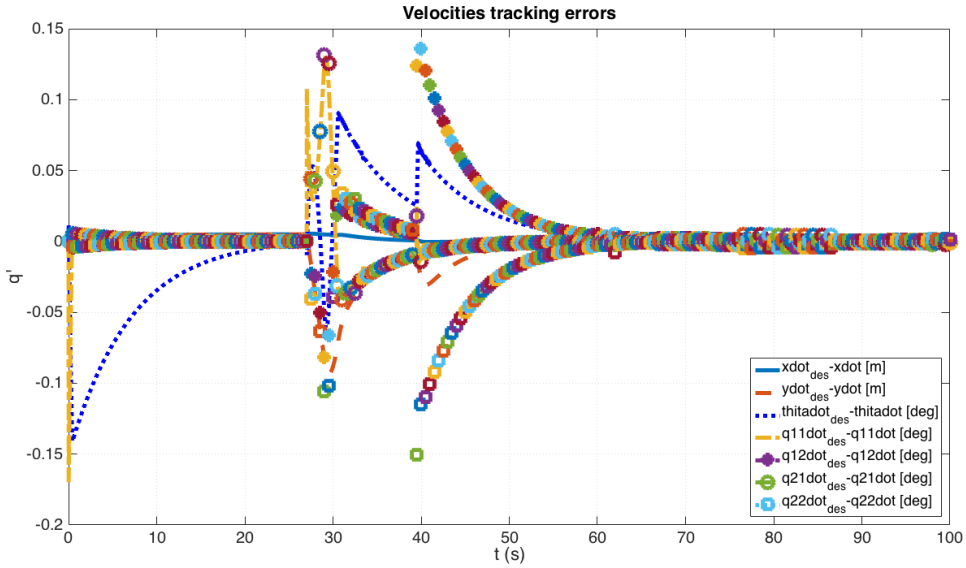


Figure 6.56: The errors tracking of the velocity of all DOFs for the controller shown in Figure 6.40.

Figure 6.57 shows that the RW is asked to deliver the biggest value of torque to the system, over 10Nm; the second set of thrusters follows, with the need to produce -6N, during the approach phase.

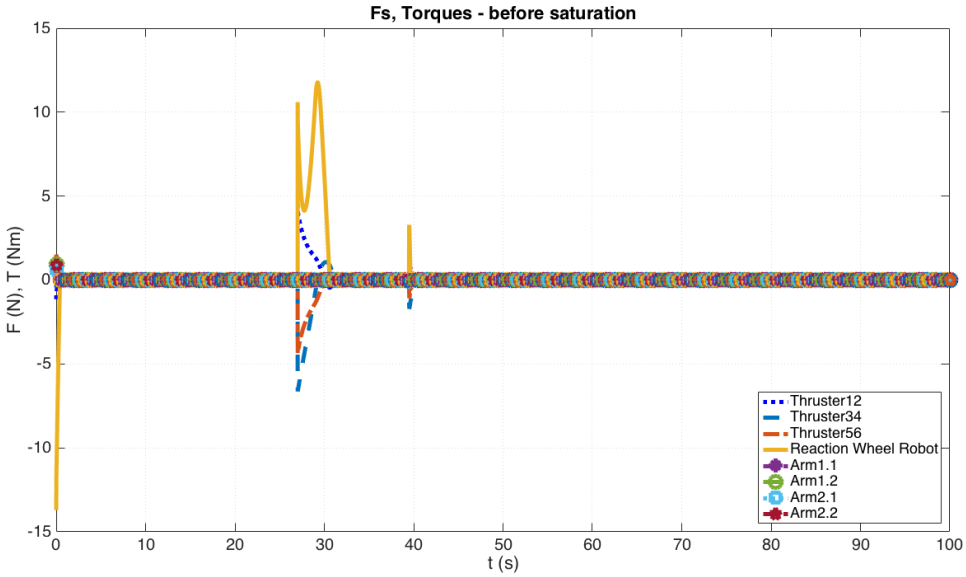


Figure 6.57: The values of forces and torques to be produced by the actuators, as calculated by the controller.

Figure 6.58 shows the redistribution of forces and torques to the system, as they are finally delivered by the actuators. The thrusters operate for longer period, as well as the RW, which is the only actuator operating until the end of the simulation, when the robot is moving with zero acceleration, following the target's trajectory.

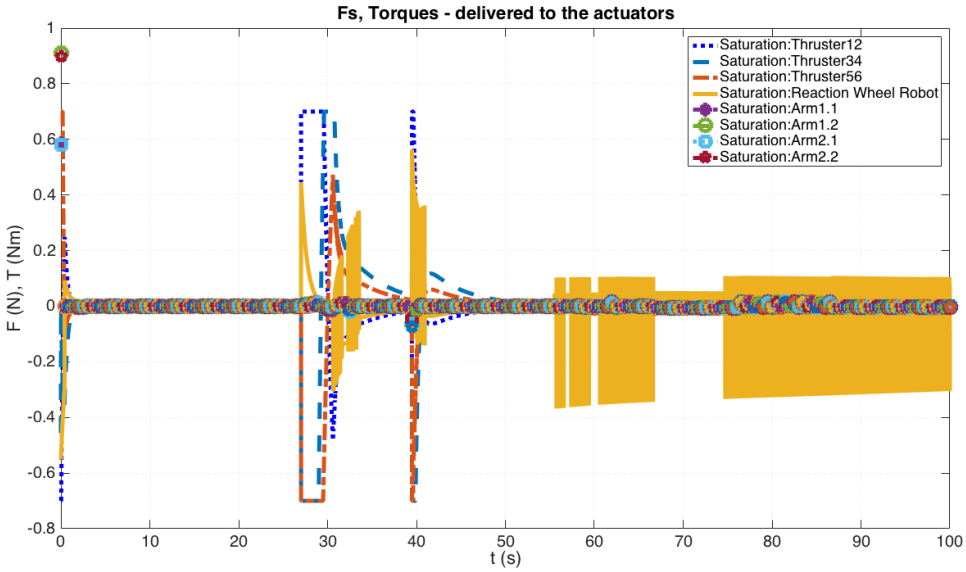


Figure 6.58: The values of forces and torques that are actually produced by the actuators, and sent to the system.

Figure 6.59 demonstrates the robot's configuration, orientation and position at the end of the simulation.

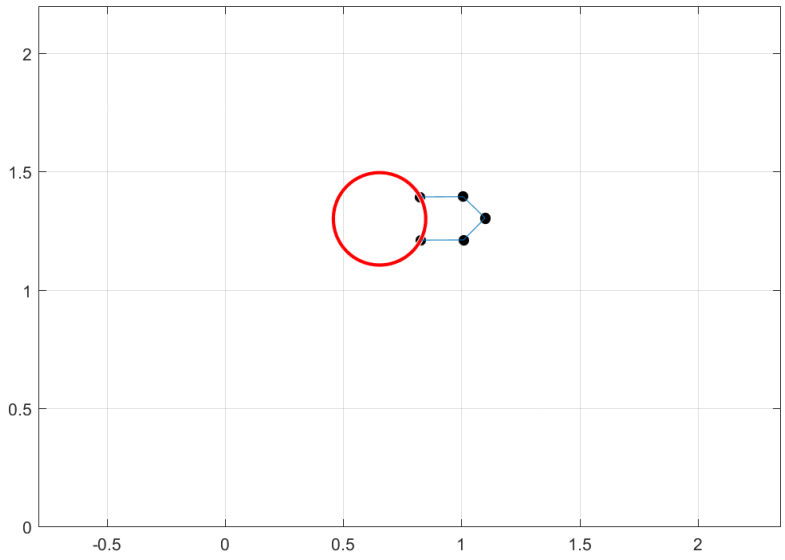


Figure 6.59: The robot at the end of the simulation - the target is located between the EEs of the manipulators.

6.2.2.2 Discussion

The residual errors are minor yet insignificant. The robot appears to perform its task and all the errors are zeroed out for 10s ($t = 15s - 25s$) and then has troubles following the object's orbit unfailingly. Regardless of the oscillations, Figures 6.42(a), 6.42(b), 6.43, 6.44, 6.45, 6.46, 6.47, show that all the states follow the control command when the system is at steady-state.

Most importantly, Figure 6.58 shows that the optimisation criterion with the objective of minimum fuel consumption is successfully effective, since the thrusters seem to be operating shortly (delivering their maximum thrust, as in the actual physical system), whereas the RW motor is taking all the load until the end of the motion. The latter is especially important since the stages of approaching and grasping shall take place in short time whilst the rectilinear motion carrying the target shall last for longer time.

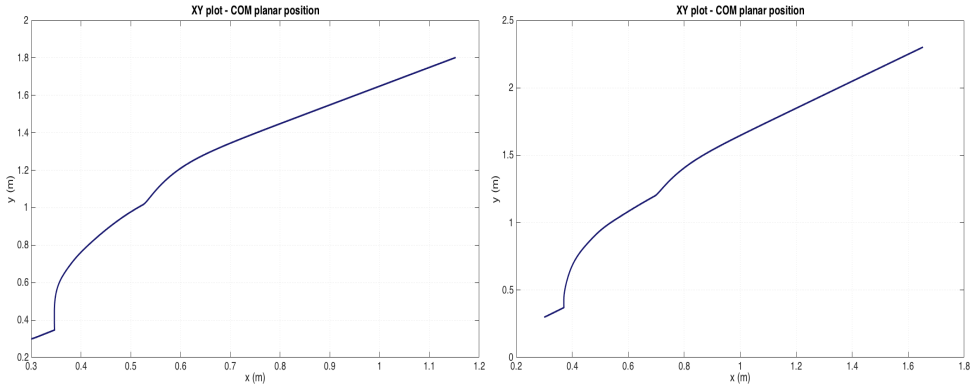
6.2.2.3 Change in the target's velocity

In order to confirm the validity of the model, other cases of a target's were included, where the target is moving with double and triple velocity than before. Therefore the target now has respectively, a velocity equal to:

- case 1 - $V_t = 10^{-3} m/s$
- case 2 - $V_t = 15^{-3} m/s$.

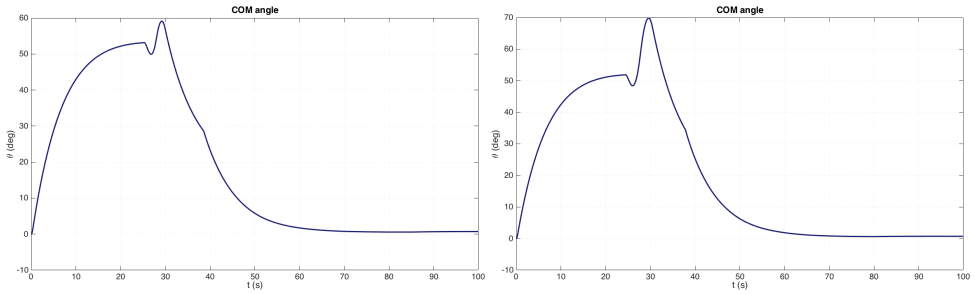
The results of the simulation show the same behavior of the chaser as in the above simulation, with some differences in the magnitude of the values of some of the states' position and velocity, and of the actuators' delivered force and torque.

The Figures are included below.



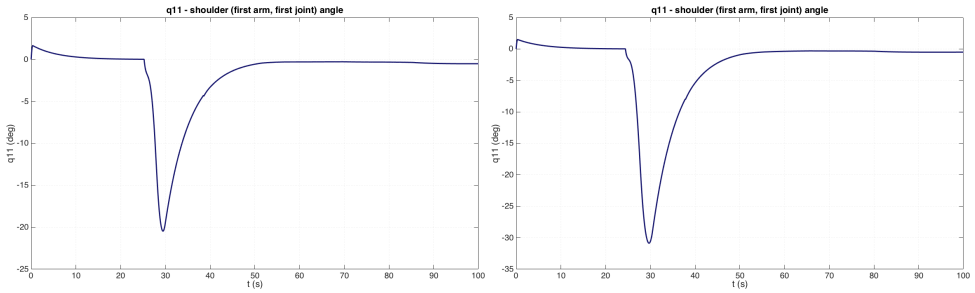
(a) (b)

Figure 6.60: The robot's COM planar displacement - (a) case 1, (b) case 2.



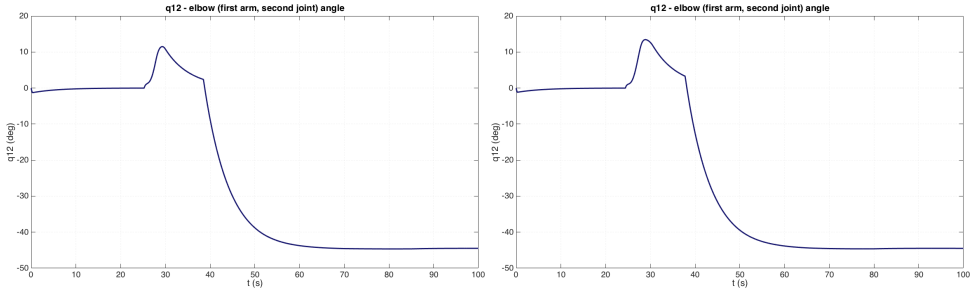
(a) (b)

Figure 6.61: The robot's orientation in respect to the absolute coordinate system - (a) case 1, (b) case 2.



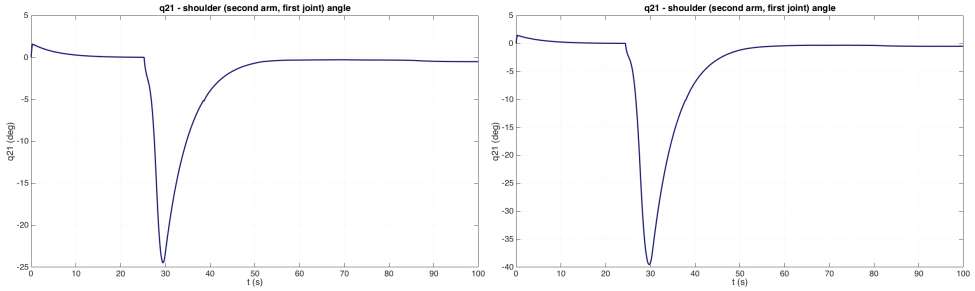
(a) (b)

Figure 6.62: The angular displacement of the first link of the first arm - (a) case 1, (b) case 2.



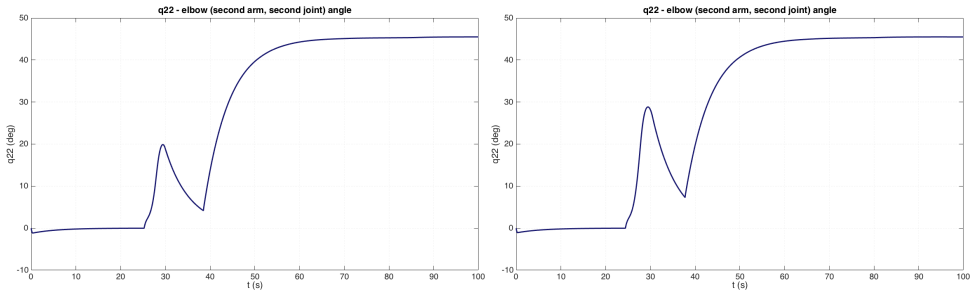
(a) (b)

Figure 6.63: The angular displacement of the second link of the first arm - (a) case 1, (b) case 2.



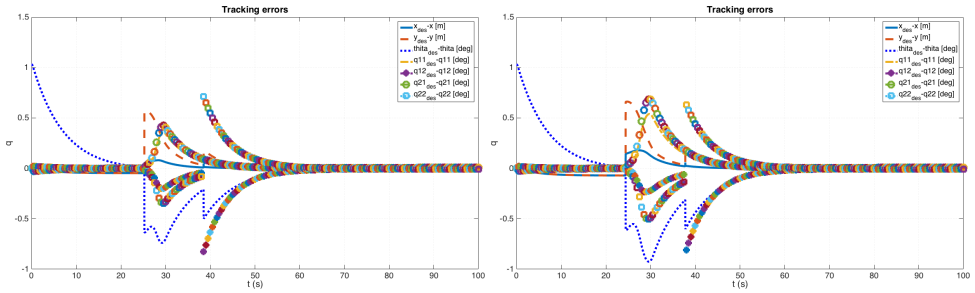
(a) (b)

Figure 6.64: The angular displacement of the first link of the second arm - (a) case 1, (b) case 2.



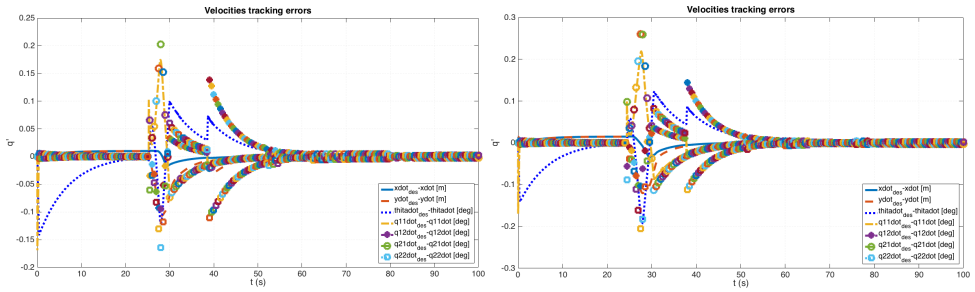
(a) (b)

Figure 6.65: The angular displacement of the second link of the second arm - (a) case 1, (b) case 2.



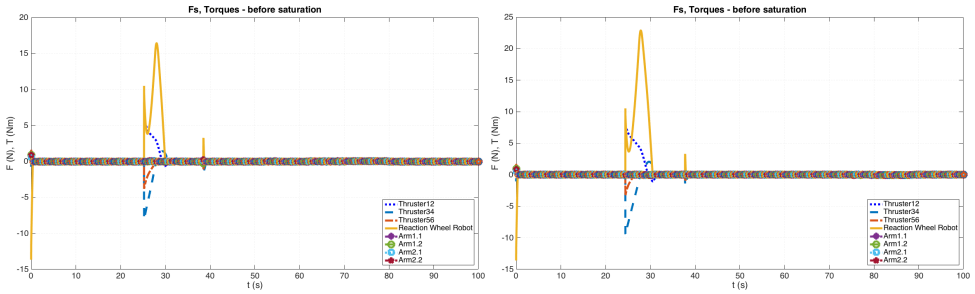
(a) (b)

Figure 6.66: The errors tracking of all DOFs - (a) case 1, (b) case 2.



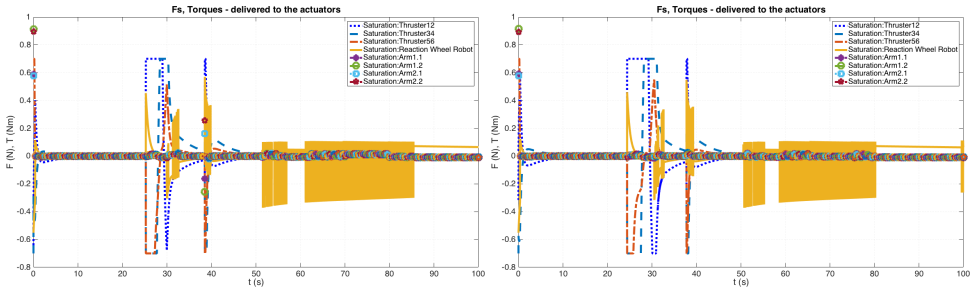
(a) (b)

Figure 6.67: The errors tracking of the velocity of all DOFs - (a) case 1, (b) case 2.



(a) (b)

Figure 6.68: The forces and torques to be delivered by the actuators, as calculated by the controller - (a) case 1, (b) case 2.



(a) (b)

Figure 6.69: The forces and torques eventually delivered to the system by the actuators - (a) case 1, (b) case 2.

6.2.2.3.1 Discussion The model responses well to a command for quicker response, successfully grasps the target and reaches the latter's speed on time. This means that the planner is doing a proper calculation of the time and point of meet and moves from the one stage to the other (orientation - approach - grasping - stable velocity & trajectory calculation) without any problems.

We can also observe the effect of the saturation on the system's actuators, which redistributes the load to the thusters sets and the reaction wheel's motor. The arms' motors seem to be have the same response in both arms.

Chapter 7

Simulink - ROS / Gazebo

In order to evaluate the vadility of the Planner and Controller, it is necessary to test the response of the system in another platform, other than Simulink, where the model is independent. Therefore, before proceeding to real time experiments, it was decided that a dynamic and visual representation of the robot emulator has to take place in a simulation environment, in which the testing the response of the system to the control commands would be possible.

It was decided that Gazebo should be the Simulation software where the testing would take place, which is receiving commands from Simulink through ROS. The model-based controller is still in Simulink, since such a complex controller is hard to code anew in ROS. However ROS's compatibility with Gazebo and Simulink should make the communication between the two platforms easy, while one will have the ability to view and evaluate the responce of the system in a graphical environment, equipt with physics engines, able to simulate the dynamic behaviour of the robot emulator.

Firstly, a representation of the robot emulator, which consists of the robot base and manipulators, including the actuators (thustres, reaction wheel and manipulators' servo motors) has to be done in Gazebo, in URDF format. The result of the code generated in the ROS environment provided in the Gazebo platform, is shown in Figure 7.1.

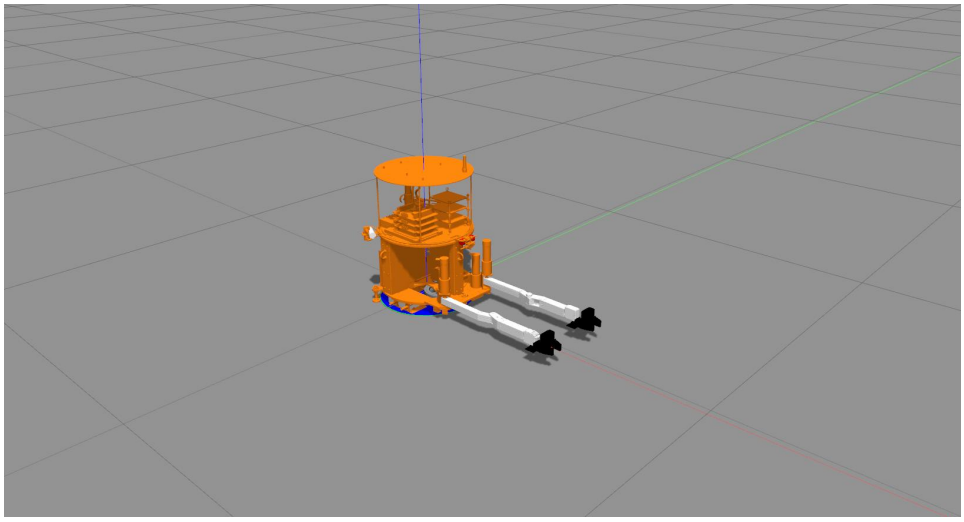


Figure 7.1: The Cepheus robot modeling representation in the Gazebo environment.

7.1 The Simulink Model

Then the file in Simulink had to be modified, in order to allow communication with the ROS modules and the custom-made ROS modules which provide the controller feedback of the position and velocity of the generalized coordinates and to allow sending and receiving commands to the latter, as well to the thrusters' sets and the RW. The model representation in Simulink has been removed, since the control commands are sent directly to the model system in Gazebo. The Simulink Planner and Controller, as used in a reali-time experiment with Gazebo, through ROS, is shown in Figure 7.2.

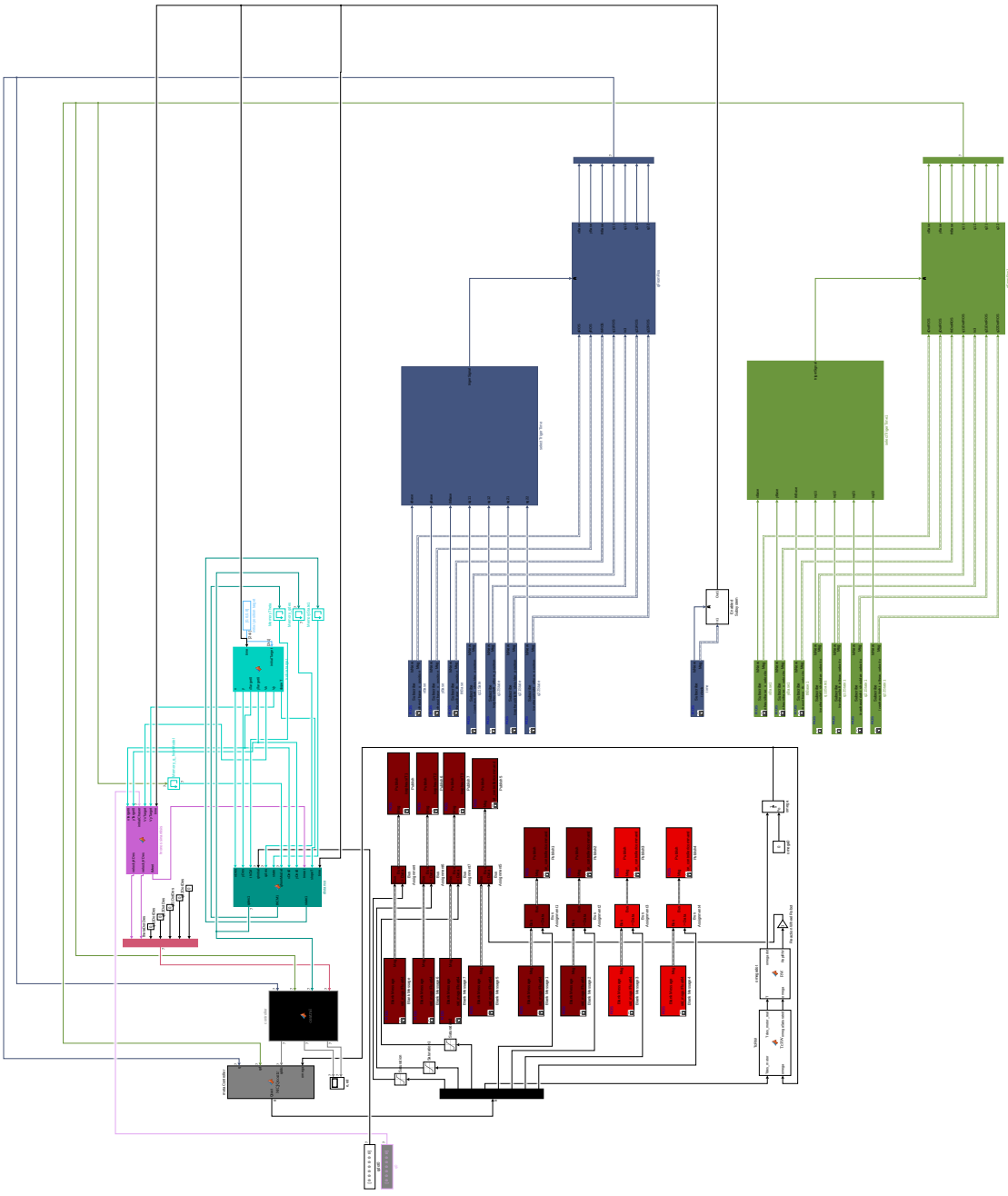


Figure 7.2: The Simulink Model-Based Controller intergrated with the Gazebo-ROS modules, during a real time experiment.

The robot is given the command to approach and grasp a moving target, which is given as a known trajectory in the planar space, as shown in Figure 7.3.

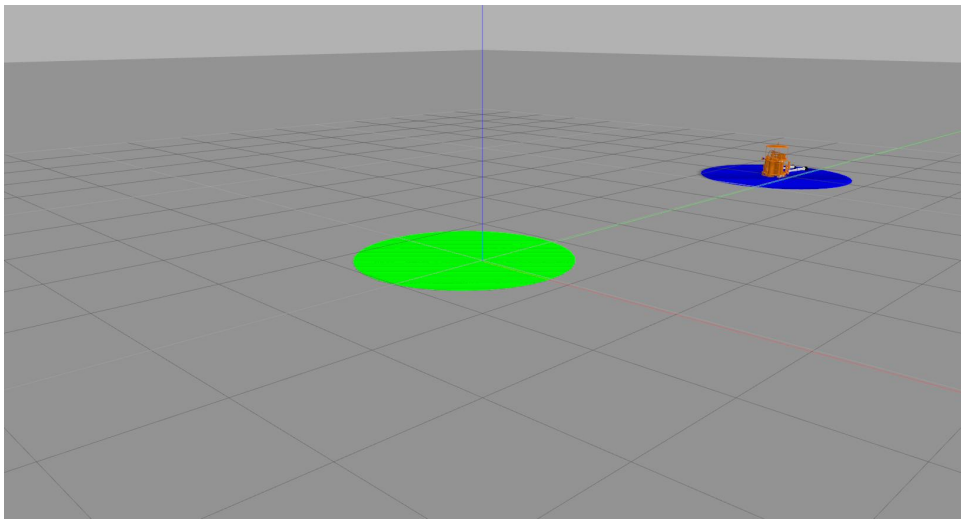


Figure 7.3: Frame of the real time experiment in Gazebo where the Cepehus robot approaching the target.

Since its planar movement was not explicitely easy to simulate, two prismatic joints were created, in the bottom of the robot base, in order to allow the planar movement. Another joint was placed on top of the prismatic ones, in order to simulate its angular displacement (revolute). At the moment, the grippers are not considered active components, while the thrusters are simulated as prismatic joints, the RW as a revolute joint, as well as the manipulators' motors.

7.2 Real-time experiments in Gazebo

In the experiments ran, the robot seemed to follow the expected trajectory, however synchronisation issues did not allow the proper testing of the communication of Simulink and Gazebo. Figure 7.4 shows the robot's displacement during an experiment execution.

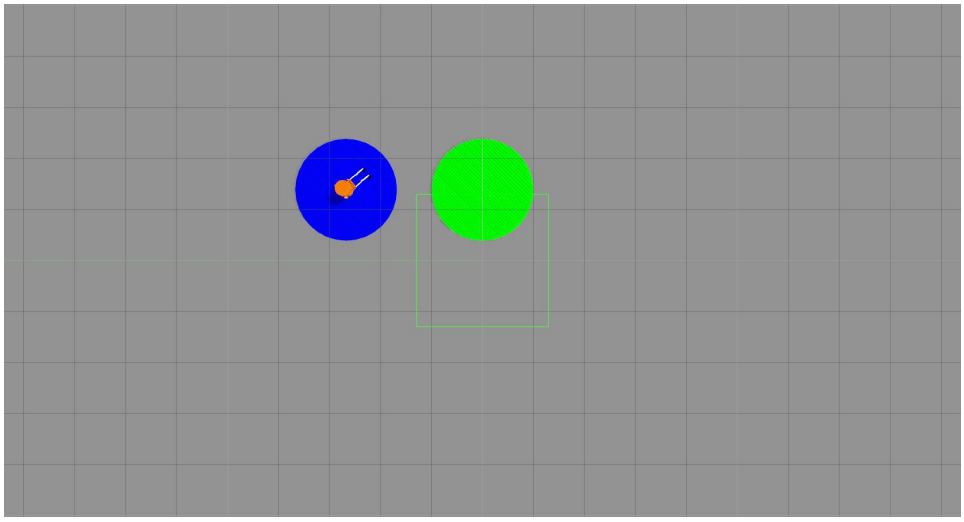


Figure 7.4: Top view of the approach phase during a real time experiment in the gazebo environment.

During the experiments the thrusters performance was recorded, and is shown in Figures 7.5, 7.6 and 7.7.

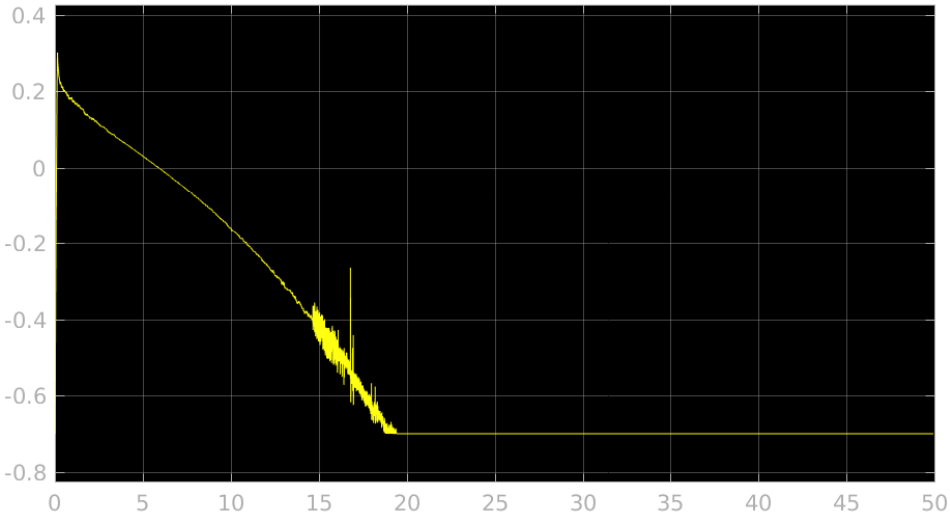


Figure 7.5: Thruster T_{12} during the target chase real-time experiment in Gazebo -commands sent from Simulink, through ROS modules.

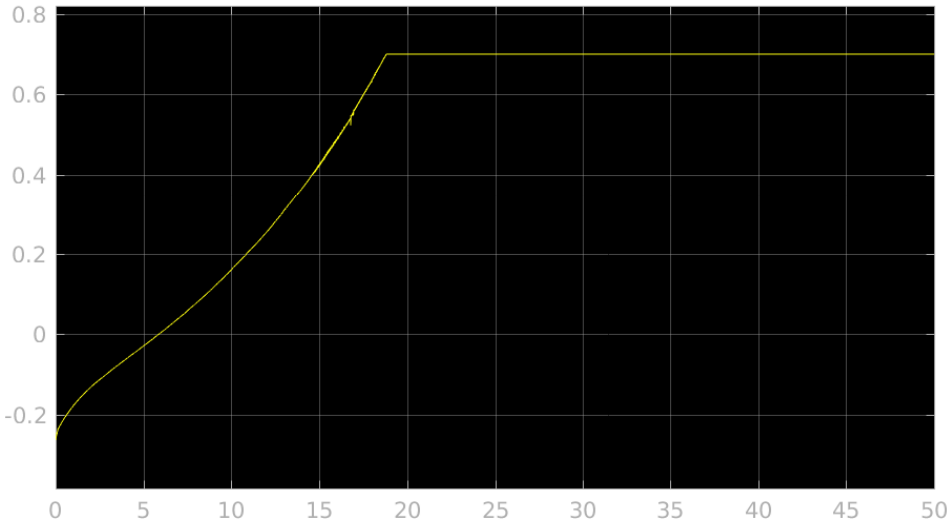


Figure 7.6: Thruster T_{34} during the target chase real-time experiment in Gazebo -commands sent from Simulink, through ROS modules.

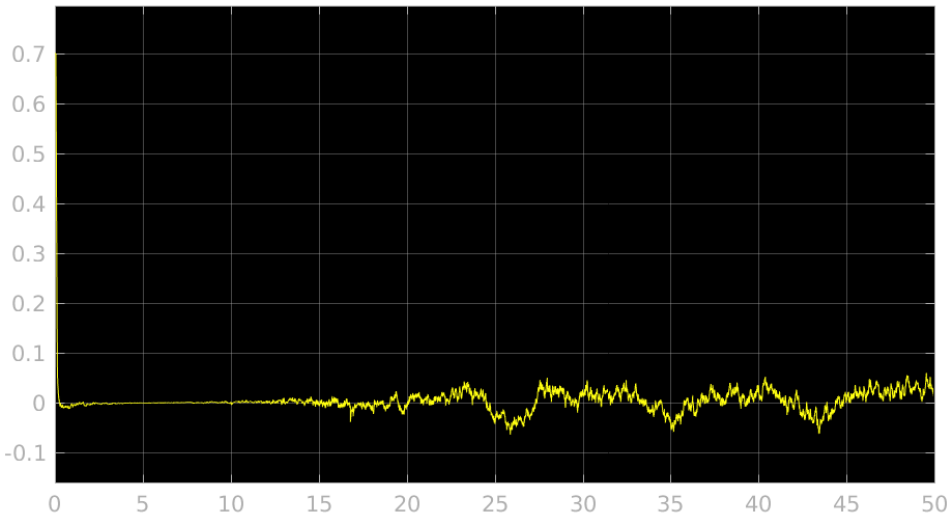


Figure 7.7: Thruster T_{56} during the target chase real-time experiment in Gazebo -commands sent from Simulink, through ROS modules.

7.3 Outcomes

The robot base seemed to follow the expected trajectory, however communication issues prevented the proper evaluation of the Controller and Planner. Thrusters T2 and T3 seem to be in continous operation, while the 3rd set of thrusters seems to have a very low value, always close to zero. In order for the system to reach steady-state, there should be no residual forces acting on the robot body, therefore further investigation should take place, on the feedback the controller is recieving, and the commands it's sending to the actuators.

Chapter 8

Conclusions - Future Work

This paper presents the full dynamic modelling of a 7 DOF space robot emulator, with two two-link manipulators. In the effort of autonomously controlling the robot to perform grasping a moving target on-orbit, a model-based controller and a planner -objective minimum fuel consuption- were developed, and coded in Simulink, Matlab. The dynamic representation of the model was also developed in the same environment. Later, the representation of the model of the robot system was developed in Gazebo, in URDF, and through ROS, commands were sent from Simulink , to the simulation environment of Gazebo, in order to check the validity of the controller and planner.

The robot seems to behave as expected, and reaches the final objective, which is to approach a target on-orbit and when it is within its manipulators' WS, to attempt to grasp the target, with a predefined manipulators' configuration. The results show an initial overshoot in the beginning of the move (when receiving the control command), while the RW to have a residual value when the systems reaches its steady-state. A modification in the control gains could remove the overshoot, as well as the control command sent to the actuators.

Concerning the Gazebo representation of the model, proper synchronisation between Simulink, ROS and Gazebo, should allow the real-time experiments testing and evalutation of the SImulink code, before testing it on the actual robot. The final step will be rea-time testing of the Simulink controller in the Cepheus robot. Moreover, attempting to trasfer the controller in ROS completely, should remove the synchronisation issues, and make the testing of new modules easier, while contributing to the sustainability of the controller.

Bibliography

[1] Hubble telescope. http://hubblesite.org/the_telescope/hubble_essentials/. Accessed: June 2016.

[2] Robots in space. http://www.universetoday.com/43750/robots-in-space/. Accessed: June 2016.

[3] Phoenix lander. http://phoenix.lpl.arizona.edu/index.php. Accessed: June 2016.

[4] Exomars mission. http://exploration.esa.int/mars/. Accessed: October 2016.

[5] Esa exploration. http://exploration.esa.int/mars/48088-mission-overview/. Accessed: June 2016.

[6] Iss. http://www.daviddarling.info/encyclopedia/I/ISS.html. Accessed: June 2016.

[7] Canadarm. http://www.asc-csa.gc.ca/eng/canadarm/. Accessed: June 2016.

[8] Canadarm2. http://www.asc-csa.gc.ca/eng/iss/canadarm2/. Accessed: June 2016.

[9] Dextre. http://www.asc-csa.gc.ca/eng/iss/dextre/. Accessed: June 2016.

[10] In-space robotic servicing. https://gameon.nasa.gov/projects-2/archived-projects-2/robotic-satellite-servicing/. Accessed: February 2017.

[11] In-space maintenance. http://www.h2020-peraspera.eu/?page_id=363. Accessed: February 2017.

[12] In-space assembly. http://www.h2020-peraspera.eu/?page_id=365. Accessed: February 2017.

[13] Clean space. http://www.esa.int/Our_Activities/Space_Engineering_Technology/Clean_Space/How_to_catch_a_satellite. Accessed: February 2017.

[14] Newway air bearings. http://www.newwayairbearings.com/. Accessed: October 2016.

[15] Nanosatellites. http://www.nanosats.eu/. Accessed: June 2016.

[16] Ioannis K. Kaliakatsos. *Design of ropulsion system with ressurised C02 and motion control of a space robot emulator, Diploma Thesis, in Greek*. NTUA, 2006.

[17] Reaction wheel. https://en.wikipedia.org/wiki/Reaction_wheel. Accessed: June 2016.

[18] Demetrios G. Psarros. *Analysis, design and evaluation of reaction wheels for a planar space robot emulator, MSc Thesis, in Greek*. NTUA, 2006.

[19] Lipo batteries. http://www.rchelicopterfun.com/rc-lipo-batteries.html. Accessed: June 2016.

[20] Aruco: a minimal library for augmented reality applications based on opencv. http://www.uco.es/investiga/grupos/ava/node/26. Accessed: February 2017.

[21] Claudio Melchiorri. *Design of Motion Control Systems*. Universita di Bologna.

[22] Patrick M Knupp. Achieving finite element mesh quality via optimization of the jacobian matrix norm and associated quantities. part i—a framework for surface mesh optimization. *International Journal for Numerical Methods in Engineering*, 48(3):401–420, 2000.

[23] A quick tutorial on multibody dynamics - school of interactive computing, georgia institute of technology. http://www.cc.gatech.edu/~karenliu/Home_files/dynamics_1.pdf. Accessed: July 2016.

[24] Claudio Melchiorri. *Control of robot manipulators, Inverse dynamics control*. Universita di Bologna.

[25] Bruno Siciliano, Lorenzo Sciavicco, Luigi Villani, and Giuseppe Oriolo. *Robotics: modelling, planning and control*. Springer Science & Business Media, 2010.

[26] Papadopoulos E. and Kyriakopoulos K. *Introduction to Robotics, in Greek*. NTUA Press, 2004.

[27] Giorgos Papastergiou. *Design, Implementation and Control of robotic arms for a robot space emulator, Diploma Thesis, in Greek*. NTUA, 2015.

[28] Evangelos Papadopoulos, Ioannis K Kaliakatsos, and Dimitrios Psarros. Minimum fuel techniques for a space robot simulator with a reaction wheel and pwm thrusters. In *Control Conference (ECC), 2007 European*, pages 3391–3398. IEEE, 2007.

[29] Chandeok Park and Daniel J Scheeres. Solutions of the optimal feedback control problem using hamiltonian dynamics and generating functions. In *Decision and Control, 2003. Proceedings. 42nd IEEE Conference on*, volume 2, pages 1222–1227. IEEE, 2003.

[30] Evangelos Papadopoulos. *On the Dynamics and Control of Space Manipulators, Doctor of Philosophy*. Massachusetts Institute of Technology, 1990.

Appendix A

M & C Matrices

M Matrix

$M_{11} = 2 * m1 + 3 * m2 + mb$
$M_{12} = 0$
$M_{13} = -m2 * r1 * cos(dba - th) - ab * m2 * cos(db + th) - a1 * m2 * cos(q21 + th) - ac2 * m2 * cos(da2 + q21 + q22 + th) + m1 * r1 * sin(dba - th) + m2 * r1 * sin(dba - th) + ac1 * m1 * sin(da1 - q11 - th) + ac2 * m2 * sin(da2 - q11 - q12 - th) + 2 * ab * m1 * sin(db + th) + 2 * ab * m2 * sin(db + th) - m1 * r1 * sin(dba + th) - m2 * r1 * sin(dba + th) - a1 * m2 * sin(q11 + th) - a1 * m2 * sin(q21 + th) - ac1 * m1 * sin(da1 + q21 + th) - ac2 * m2 * sin(da2 + q21 + q22 + th)$
$M_{14} = ac1 * m1 * sin(da1 - q11 - th) + ac2 * m2 * sin(da2 - q11 - q12 - th) - a1 * m2 * sin(q11 + th)$
$M_{15} = ac2 * m2 * sin(da2 - q11 - q12 - th)$
$M_{16} = -ac1 * m1 * sin(da1 + q21 + th) - m2 * (a1 * (cos(q21 + th) + sin(q21 + th)) + ac2 * (cos(da2 + q21 + q22 + th) + sin(da2 + q21 + q22 + th)))$
$M_{17} = -ac2 * m2 * (cos(da2 + q21 + q22 + th) + sin(da2 + q21 + q22 + th))$
$M_{21} = 0$
$M_{22} = 2 * m1 + m2 + mb$
$M_{23} = -m1 * r1 * cos(dba - th) + ac1 * m1 * cos(da1 - q11 - th) + m2 * (ac2 * cos(da2 - q11 - q12 - th) - ab * cos(db + th) + r1 * cos(dba + th) + a1 * cos(q11 + th)) + m1 * ((-2) * ab * cos(db + th) + r1 * cos(dba + th) - ac1 * cos(da1 + q21 + th))$
$M_{24} = ac1 * m1 * cos(da1 - q11 - th) + ac2 * m2 * cos(da2 - q11 - q12 - th) + a1 * m2 * cos(q11 + th)$
$M_{25} = ac2 * m2 * cos(da2 - q11 - q12 - th)$
$M_{26} = -ac1 * m1 * cos(da1 + q21 + th); I1 + I2 + ac1^2 * m1 + (a1^2 + ac2^2) *$

$m2 + a1 * m2 * r1 * cos(dba + q21) + ac1 * m1 * r1 * cos(da1 + dba + q21) + 2 * a1 * ac2 * m2 * cos(da2 + q22) + ac2 * m2 * r1 * cos(da2 + dba + q21 + q22) + a1 * ab * m2 * cos(db + q21 + 2 * th) + ab * ac1 * m1 * cos(da1 + db + q21 + 2 * th) + ab * ac2 * m2 * cos(da2 + db + q21 + q22 + 2 * th)$
$M_{27} = 0$
$M_{31} = -m2 * r1 * cos(dba - th) - ab * m2 * cos(db + th) - a1 * m2 * cos(q21 + th) - ac2 * m2 * cos(da2 + q21 + q22 + th) + m1 * r1 * sin(dba - th) + m2 * r1 * sin(dba - th) + ac1 * m1 * sin(da1 - q11 - th) + ac2 * m2 * sin(da2 - q11 - q12 - th) + 2 * ab * m1 * sin(db + th) + 2 * ab * m2 * sin(db + th) - m1 * r1 * sin(dba + th) - m2 * r1 * sin(dba + th) - a1 * m2 * sin(q11 + th) - a1 * m2 * sin(q21 + th) - ac1 * m1 * sin(da1 + q21 + th) - ac2 * m2 * sin(da2 + q21 + q22 + th)$
$M_{32} = -m1 * r1 * cos(dba - th) + ac1 * m1 * cos(da1 - q11 - th) + m2 * (ac2 * cos(da2 - q11 - q12 - th) - ab * cos(db + th) + r1 * cos(dba + th) + a1 * cos(q11 + th)) + m1 * ((-2) * ab * cos(db + th) + r1 * cos(dba + th) - ac1 * cos(da1 + q21 + th))$
$M_{33} = 2 * I1 + 2 * I2 + Ib + 2 * (ac1^2 * m1 + (a1^2 + ac2^2) * m2 + ab^2 * (m1 + m2) + (m1 + m2) * r1^2) - 2 * ab * (m1 + m2) * r1 * cos(db - dba) + 2 * (-a1 * ab * m2 * cos(db - q11) - ab * ac1 * m1 * cos(da1 + db - q11) + a1 * m2 * r1 * cos(dba - q11) + ac1 * m1 * r1 * cos(da1 + dba - q11) + a1 * ac2 * m2 * cos(da2 - q12) - ab * ac2 * m2 * cos(da2 + db - q11 - q12) + ac2 * m2 * r1 * cos(da2 + dba - q11 - q12) + a1 * m2 * r1 * cos(dba + q21) + ac1 * m1 * r1 * cos(da1 + dba + q21) + a1 * ac2 * m2 * cos(da2 + q22) + ac2 * m2 * r1 * cos(da2 + dba + q21 + q22) + ab * m1 * r1 * cos(db - dba + 2 * th) + ab * m2 * r1 * cos(db - dba + 2 * th) + a1 * ab * m2 * cos(db + q21 + 2 * th) + ab * ac1 * m1 * cos(da1 + db + q21 + 2 * th) + ab * ac2 * m2 * cos(da2 + db + q21 + q22 + 2 * th))$
$M_{34} = I1 + I2 + ac1^2 * m1 + (a1^2 + ac2^2) * m2 - a1 * ab * m2 * cos(db - q11) - ab * ac1 * m1 * cos(da1 + db - q11) + a1 * m2 * r1 * cos(dba - q11) + ac1 * m1 * r1 * cos(da1 + dba - q11) + 2 * a1 * ac2 * m2 * cos(da2 - q12) - ab * ac2 * m2 * cos(da2 + db - q11 - q12) + ac2 * m2 * r1 * cos(da2 + dba - q11 - q12)$
$M_{35} = I2 + ac2^2 * m2 + ac2 * m2 * (a1 * cos(da2 - q12) - ab * cos(da2 + db - q11 - q12) + r1 * cos(da2 + dba - q11 - q12))$
$M_{36} = 0$
$M_{37} = I2 + ac2^2 * m2 + ac2 * m2 * (a1 * cos(da2 + q22) + r1 * cos(da2 + dba + q21 + q22) + ab * cos(da2 + db + q21 + q22 + 2 * th))$
$M_{41} = ac1 * m1 * sin(da1 - q11 - th) + ac2 * m2 * sin(da2 - q11 - q12 - th) - a1 * m2 * sin(q11 + th)$
$M_{42} = ac1 * m1 * cos(da1 - q11 - th) + ac2 * m2 * cos(da2 - q11 - q12 - th) + a1 * m2 * cos(q11 + th)$
$M_{43} = I1 + I2 + ac1^2 * m1 + (a1^2 + ac2^2) * m2 - a1 * ab * m2 * cos(db - q11) - ab * ac1 * m1 * cos(da1 + db - q11) + a1 * m2 * r1 * cos(dba - q11) + ac1 * m1 * r1 * cos(da1 + dba - q11) + 2 * a1 * ac2 * m2 * cos(da2 - q12) - ab * ac2 * m2 * cos(da2 + db - q11 - q12) + ac2 * m2 * r1 * cos(da2 + dba - q11 - q12)$

$M_{44} = I1 + I2 + It + ac1^2 * m1 + (a1^2 + ac2^2) * m2 + 2 * Ima * n^2 + 2 * a1 * ac2 * m2 * cos(da2 - q12)$
$M_{45} = I2 + It + ac2^2 * m2 + Ima * n^2 + a1 * ac2 * m2 * cos(da2 - q12)$
$M_{46} = 0$
$M_{47} = 0$
$M_{51} = ac2 * m2 * sin(da2 - q11 - q12 - th)$
$M_{52} = ac2 * m2 * cos(da2 - q11 - q12 - th)$
$M_{53} = I2 + ac2^2 * m2 + ac2 * m2 * (a1 * cos(da2 - q12) - ab * cos(da2 + db - q11 - q12) + r1 * cos(da2 + dba - q11 - q12))$
$M_{54} = I2 + It + ac2^2 * m2 + Ima * n^2 + a1 * ac2 * m2 * cos(da2 - q12)$
$M_{55} = I2 + It + ac2^2 * m2 + Ima * n^2$
$M_{56} = 0$
$M_{57} = 0$
$M_{61} = -ac1 * m1 * sin(da1 + q21 + th) - m2 * (a1 * (cos(q21 + th) + sin(q21 + th)) + ac2 * (cos(da2 + q21 + q22 + th) + sin(da2 + q21 + q22 + th)))$
$M_{62} = -ac1 * m1 * cos(da1 + q21 + th)$
$M_{63} = I1 + I2 + ac1^2 * m1 + (a1^2 + ac2^2) * m2 + a1 * m2 * r1 * cos(dba + q21) + ac1 * m1 * r1 * cos(da1 + dba + q21) + 2 * a1 * ac2 * m2 * cos(da2 + q22) + ac2 * m2 * r1 * cos(da2 + dba + q21 + q22) + a1 * ab * m2 * cos(db + q21 + 2 * th) + ab * ac1 * m1 * cos(da1 + db + q21 + 2 * th) + ab * ac2 * m2 * cos(da2 + db + q21 + q22 + 2 * th)$
$M_{64} = 0$
$M_{65} = 0$
$M_{66} = I1 + I2 + It + ac1^2 * m1 + (a1^2 + ac2^2) * m2 + 2 * Ima * n^2 + 2 * a1 * ac2 * m2 * cos(da2 + q22)$
$M_{67} = I2 + It + ac2^2 * m2 + Ima * n^2 + a1 * ac2 * m2 * cos(da2 + q22)$
$M_{71} = -ac2 * m2 * (cos(da2 + q21 + q22 + th) + sin(da2 + q21 + q22 + th))$
$M_{72} = 0$
$M_{73} = I2 + ac2^2 * m2 + ac2 * m2 * (a1 * cos(da2 + q22) + r1 * cos(da2 + dba + q21 + q22) + ab * cos(da2 + db + q21 + q22 + 2 * th))$
$M_{74} = 0$
$M_{75} = 0$
$M_{76} = I2 + It + ac2^2 * m2 + Ima * n^2 + a1 * ac2 * m2 * cos(da2 + q22)$
$M_{77} = I2 + It + ac2^2 * m2 + Ima * n^2$

C Matrix

$C_{11} = -ac1 * m1 * q11dot^2 * cos(da1 - q11 - th) - ac2 * m2 * q11dot^2 * cos(da2 - q11 - q12 - th) - 2 * ac2 * m2 * q11dot * q12dot * cos(da2 - q11 - q12 - th) - ac2 * m2 * q12dot^2 * cos(da2 - q11 - q12 - th) - a1 * m2 * q11dot^2 * cos(q11 + th) - a1 * m2 * q21dot^2 * cos(q21 + th) - ac1 * m1 * q21dot^2 * cos(da1 + q21 + th) -$

$ac2*m2*q21dot^2*cos(da2+q21+q22+th)-2*ac2*m2*q21dot*q22dot*cos(da2+q21+q22+th)-ac2*m2*q22dot^2*cos(da2+q21+q22+th)+a1*m2*q21dot^2*sin(q21+th)-thdot^2*((m1+m2)*r1*cos(dba-th)+ac1*m1*cos(da1-q11-th)+m1*((-2)*ab*cos(db+th)+r1*cos(dba+th)+ac1*cos(da1+q21+th))+m2*(r1*(cos(dba+th)+sin(dba-th))-ab*(2*cos(db+th)+sin(db+th))+a1*(cos(q11+th)+cos(q21+th)-sin(q21+th))+ac2*(cos(da2-q11-q12-th)+cos(da2+q21+q22+th)-sin(da2+q21+q22+th))))+ac2*m2*q21dot^2*sin(da2+q21+q22+th)+2*ac2*m2*q21dot*q22dot*sin(da2+q21+q22+th)+ac2*m2*q22dot^2*sin(da2+q21+q22+th)-2*thdot*(ac2*m2*q12dot*cos(da2-q11-q12-th)+q11dot*(ac1*m1*cos(da1-q11-th)+ac2*m2*cos(da2-q11-q12-th)+a1*m2*cos(q11+th))+ac2*m2*q22dot*(cos(da2+q21+q22+th)-sin(da2+q21+q22+th))+q21dot*(ac1*m1*cos(da1+q21+th)+m2*(a1*cos(q21+th)+ac2*cos(da2+q21+q22+th)-a1*sin(q21+th)-ac2*sin(da2+q21+q22+th))))$

$C_{21} = ac2*m2*q12dot^2*sin(da2-q11-q12-th)+q11dot^2*(ac1*m1*sin(da1-q11-th)+ac2*m2*sin(da2-q11-q12-th)-a1*m2*sin(q11+th))+2*q11dot*(ac2*m2*q12dot*sin(da2-q11-q12-th)+thdot*(ac1*m1*sin(da1-q11-th)+ac2*m2*sin(da2-q11-q12-th)-a1*m2*sin(q11+th)))+ac1*m1*q21dot^2*sin(da1+q21+th)+2*thdot*(ac2*m2*q12dot*sin(da2-q11-q12-th)+ac1*m1*q21dot*sin(da1+q21+th))+thdot^2*(-m1*r1*sin(dba-th)+ac1*m1*sin(da1-q11-th)+m2*(ac2*sin(da2-q11-q12-th)+ab*sin(db+th)-r1*sin(dba+th)-a1*sin(q11+th))+m1*(2*ab*sin(db+th)-r1*sin(dba+th)+ac1*sin(da1+q21+th)))$

$C_{31} = 2*ac2*m2*q11dot*q12dot*(a1*sin(da2-q12)-ab*sin(da2+db-q11-q12)+r1*sin(da2+dba-q11-q12))+ac2*m2*q12dot^2*(a1*sin(da2-q12)-ab*sin(da2+db-q11-q12)+r1*sin(da2+dba-q11-q12))+q11dot^2*(-a1*ab*m2*sin(db-q11)-ab*ac1*m1*sin(da1+db-q11)+a1*m2*r1*sin(dba-q11)+ac1*m1*r1*sin(da1+dba-q11)-ab*ac2*m2*sin(da2+db-q11-q12)+ac2*m2*r1*sin(da2+dba-q11-q12))-a1*m2*q21dot^2*r1*sin(dba+q21)-ac1*m1*q21dot^2*r1*sin(da1+dba+q21)-2*a1*ac2*m2*q21dot*q22dot*sin(da2+q22)-a1*ac2*m2*q22dot^2*sin(da2+q22)-ac2*m2*q21dot^2*r1*sin(da2+dba+q21+q22)-2*ac2*m2*q21dot*q22dot*r1*sin(da2+dba+q21+q22)-ac2*m2*q22dot^2*r1*sin(da2+dba+q21+q22)-a1*ab*m2*q21dot^2*sin(db+q21+2*th)-ab*ac1*m1*q21dot^2*sin(da1+db+q21+2*th)-ab*ac2*m2*q21dot^2*sin(da2+db+q21+q22+2*th)-2*ab*ac2*m2*q21dot*q22dot*sin(da2+db+q21+q22+2*th)-ab*ac2*m2*q22dot^2*sin(da2+db+q21+q22+2*th)-2*ab*thdot^2*((m1+m2)*r1*sin(db-dba+2*th)+a1*m2*sin(db+q21+2*th)+ac1*m1*sin(da1+db+q21+2*th)+ac2*m2*sin(da2+db+q21+q22+2*th))-2*thdot*(ac2*m2*q12dot*(-a1*sin(da2-q12)+ab*sin(da2+db-q11-$

$q12) - r1 * sin(da2 + dba - q11 - q12)) + q11dot * (a1 * ab * m2 * sin(db - q11) + ab * ac1 * m1 * sin(da1 + db - q11) - a1 * m2 * r1 * sin(dba - q11) + ab * ac2 * m2 * sin(da2 + db - q11 - q12) - r1 * (ac1 * m1 * sin(da1 + dba - q11) + ac2 * m2 * sin(da2 + dba - q11 - q12))) + ac2 * m2 * q22dot * (a1 * sin(da2 + q22) + r1 * sin(da2 + dba + q21 + q22) + ab * sin(da2 + db + q21 + q22 + 2 * th)) + q21dot * (a1 * m2 * r1 * sin(dba + q21) + ac1 * m1 * r1 * sin(da1 + dba + q21) + ac2 * m2 * r1 * sin(da2 + dba + q21 + q22) + a1 * ab * m2 * sin(db + q21 + 2 * th) + ab * ac1 * m1 * sin(da1 + db + q21 + 2 * th) + ab * ac2 * m2 * sin(da2 + db + q21 + q22 + 2 * th)))$

$C_{41} = a1 * ac2 * m2 * q12dot * (2 * q11dot + q12dot) * sin(da2 - q12) + 2 * a1 * ac2 * m2 * q12dot * thdot * sin(da2 - q12) + thdot^2 * (a1 * ab * m2 * sin(db - q11) + ab * ac1 * m1 * sin(da1 + db - q11) - a1 * m2 * r1 * sin(dba - q11) + ab * ac2 * m2 * sin(da2 + db - q11 - q12) - r1 * (ac1 * m1 * sin(da1 + dba - q11) + ac2 * m2 * sin(da2 + dba - q11 - q12)))$

$C_{51} = -ac2 * m2 * (a1 * q11dot^2 * sin(da2 - q12) + 2 * a1 * q11dot * thdot * sin(da2 - q12) + thdot^2 * (a1 * sin(da2 - q12) - ab * sin(da2 + db - q11 - q12) + r1 * sin(da2 + dba - q11 - q12)))$

$C_{61} = -a1 * ac2 * m2 * q22dot * (2 * q21dot + q22dot) * sin(da2 + q22) - 2 * a1 * ac2 * m2 * q22dot * thdot * sin(da2 + q22) + thdot^2 * (a1 * m2 * r1 * sin(dba + q21) + ac1 * m1 * r1 * sin(da1 + dba + q21) + ac2 * m2 * r1 * sin(da2 + dba + q21 + q22) - a1 * ab * m2 * sin(db + q21 + 2 * th) - ab * ac1 * m1 * sin(da1 + db + q21 + 2 * th) - ab * ac2 * m2 * sin(da2 + db + q21 + q22 + 2 * th))$

$C_{71} = ac2 * m2 * (a1 * q21dot^2 * sin(da2 + q22) + 2 * a1 * q21dot * thdot * sin(da2 + q22) + thdot^2 * (a1 * sin(da2 + q22) + r1 * sin(da2 + dba + q21 + q22) - ab * sin(da2 + db + q21 + q22 + 2 * th)))]$

Appendix B

The $J_{c_{act}}$ matrix

J acting Matrix

Where the angles:
$th1 = pi()/6$
$th2 = -pi()/2$
$th3 = -pi()/6$

$J_{11} = cos(th1)$
$J_{12} = cos(th2)$
$J_{13} = cos(th3) J_{14} = 0$
$J_{15} = 0$
$J_{16} = 0$
$J_{17} = 0$
$J_{18} = 0$
$J_{21} = sin(th1)$
$J_{22} = sin(th2)$
$J_{23} = sin(th3)$
$J_{24} = 0$
$J_{25} = 0 J_{26} = 0$
$J_{27} = 0$
$J_{28} = 0$
$J_{31} = -((r1 * cos(th1) + ab * cos(db + th)) * sin(th1) + (r1 * sin(th1) + ab * sin(db + th)) * cos(th1))$
$J_{32} = -((r1 * cos(th2) + ab * cos(db + th)) * sin(th2) + (r1 * sin(th2) + ab * sin(db + th)) * cos(th2))$

$J_{33} = ((r1 * cos(th3) + ab * cos(db + th)) * cos(th3) + (r1 * sin(th3) + ab * sin(db + th)) * sin(th3))$
$J_{34} = -n$
$J_{35} = 0$
$J_{36} = 0$
$J_{37} = 0$
$J_{38} = 0$
$J_{41} = 0$
$J_{42} = 0$
$J_{43} = 0$
$J_{44} = 0$
$J_{45} = ita * n$
$J_{46} = ita * n$
$J_{47} = 0$
$J_{48} = 0$
$J_{51} = 0$
$J_{52} = 0$
$J_{53} = 0$
$J_{54} = 0$
$J_{55} = 0$
$J_{56} = ita * n$
$J_{57} = 0$
$J_{58} = 0$
$J_{61} = 0$
$J_{62} = 0$
$J_{63} = 0$
$J_{64} = 0$
$J_{65} = 0$
$J_{66} = 0$
$J_{67} = ita * n$
$J_{68} = ita * n$
$J_{71} = 0$
$J_{72} = 0$
$J_{73} = 0$
$J_{74} = 0$
$J_{75} = 0$
$J_{76} = 0$
$J_{77} = 0$
$J_{78} = ita * n$

Appendix C

Validation of the Reaction Wheel Parameters Experiment

C.1 Matlab Code

All the measurements analysed

Mean value of torques for speeds: -30, -40, -50, -100, -200, -300

Extract ROS data into Matlab Dataseries

$bag = rosbag('2016-06-22-15-53-45.bag');$
$bagselect1 = select(bag,'Topic','/reaction_wheel_velocity_controller/command');$
$bagselect2 = select(bag,'Topic','/reaction_wheel_velocity_controller/state');$
$ts2 = timeseries(bagselect2,'Command');$
$plot(ts2.data);$

Torque vector
$force = zeros(6,1);$

Experimentally shown that 0.012Nm is the Torque that needs to be overcomed for the RW to start - gain velocity => Static Friction Force

$force(1) = 0.012; force(2) = mean(ts2.data(1249 : 1466));$
$force(3) = mean(ts2.data(1031 : 1237));$
$force(4) = mean(ts2.data(866 : 1024));$
$force(5) = mean(ts2.data(710 : 850));$
$force(6) = mean(ts2.data(1956 : 2236));$
$force(7) = mean(ts2.data(1720 : 1917));$

$velocity = [0; 30; 40; 50; 100; 200; 300];$

$plot(velocity, force, '*');$
$axis([-10350 - 0.10.5]);$

$p = polyfit(velocity, force, 1)$

$forceLinear = polyval(p, velocity);$

figure
plot(velocity,force,'o',velocity,forceLinear,'r–')
title('Linearisation')

Measurements analysed: before RW Saturation
Mean value of torques for speeds: -30, -40, -50, -100, -200, -300

Extract ROS data into Matlab Dataseries

$bag = rosbag('2016 - 06 - 22 - 15 - 53 - 45.bag');$
$bagselect1 = select(bag,'Topic','/reaction_wheel_velocity_controller/command');$
$bagselect2 = select(bag,'Topic','/reaction_wheel_velocity_controller/state');$
$ts2 = timeseries(bagselect2,'Command');$
$plot(ts2.data);$

Torque vector
$force = zeros(4, 1);$

Experimentally shown that 0.012Nm is the Torque that needs to be overcomed for the RW to start - gain velocity => Static Friction Force

$force(1) = 0.012; force(2) = mean(ts2.data(1249 : 1466));$
$force(3) = mean(ts2.data(1031 : 1237));$
$force(4) = mean(ts2.data(866 : 1024));$

$velocity = [0; 30; 40; 50];$

```
plot(velocity,force);
Axis([-10 350 -0.1 0.5]);
```

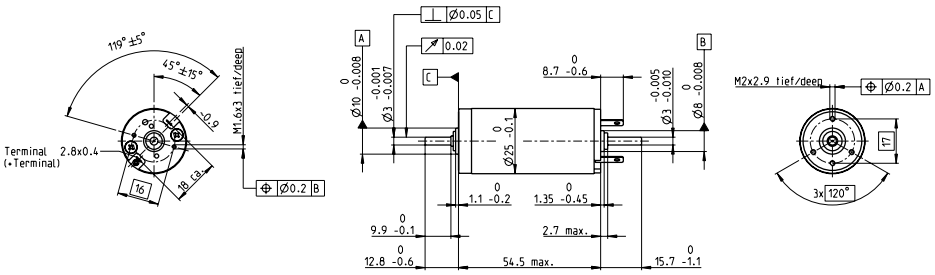
$p = polyfit(velocity, force, 1)$
$forceLinear = polyval(p, velocity);$

```
figure
plot(velocity,force,'o',velocity,forceLinear,'r–')
title('Linearization')
```

Appendix D

Manuals

RE 25 ∅25 mm, Precious Metal Brushes CLL, 10 Watt



M 1:2

Stock program
Standard program
Special program (on request)

Part Numbers

	118740	118741	118742	118743	118744	118745	118746	118747	118748
Motor Data									
Values at nominal voltage									
1 Nominal voltage V	4.5	8	9	12	15	18	24	32	48
2 No load speed rpm	5360	5320	5230	4850	4980	4790	5190	5510	5070
3 No load current mA	79.7	44.4	38.7	26.3	21.8	9.88	14.4	11.7	6.96
4 Nominal speed rpm	4980	4520	4220	3800	3920	3710	4130	4450	4000
5 Nominal torque (max. continuous torque) mNm	11.4	20.9	23.9	28.6	28.2	28.7	28	27.9	27.9
6 Nominal current (max. continuous current) A	1.5	1.5	1.5	1.24	1.01	0.811	0.652	0.516	0.317
7 Stall torque mNm	131	132	119	129	131	126	136	144	132
8 Starting current A	16.5	9.23	7.31	5.5	4.57	3.52	3.1	2.61	1.47
9 Max. efficiency %	87	87	86	87	87	90	87	87	87
Characteristics									
10 Terminal resistance Ω	0.273	0.867	1.23	2.18	3.28	5.11	7.73	12.3	32.6
11 Terminal inductance mH	0.0275	0.0882	0.115	0.238	0.353	0.551	0.832	1.31	3.48
12 Torque constant mNm/A	7.99	14.3	16.3	23.5	28.6	35.8	43.9	55.2	89.9
13 Speed constant rpm/V	1200	668	584	406	334	267	217	173	106
14 Speed / torque gradient rpm/mNm	40.9	40.5	44	37.7	38.3	38.2	38.3	38.5	38.6
15 Mechanical time constant ms	4.99	4.4	4.37	4.25	4.23	4.22	4.22	4.22	4.23
16 Rotor inertia gcm²	11.7	10.4	9.49	10.8	10.6	10.6	10.5	10.5	10.5

Specifications

Thermal data

17	Thermal resistance housing-ambient	14 K/W
18	Thermal resistance winding-housing	3.1 K/W
19	Thermal time constant winding s	12.5 s
20	Thermal time constant motor	612 s
21	Ambient temperature	-20…+85°C
22	Max. permissible winding temperature	+100°C

Mechanical data (ball bearings)

23	Max. permissible speed	5500 rpm
24	Axial play	0.05 - 0.15 mm
25	Radial play	0.025 mm
26	Max. axial load (dynamic)	3.2 N
27	Max. force for press fits (static)	64 N
	(static, shaft supported)	800 N
28	Max. radial loading, 5 mm from flange	16 N

Other specifications

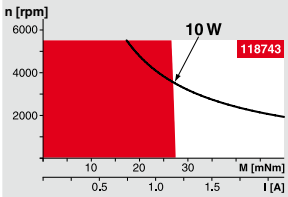
29	Number of pole pairs	1
30	Number of commutator segments	11
31	Weight of motor	130 g

CLL = Capacitor Long Life

Values listed in the table are nominal.
Explanation of the figures on page 79.

Option
Preloaded ball bearings

Operating Range



Comments

Continuous operation
In observation of above listed thermal resistance (lines 17 and 18) the maximum permissible winding temperature will be reached during continuous operation at 25°C ambient.
= Thermal limit.

Short term operation
The motor may be briefly overloaded (recurring).

Assigned power rating

maxon Modular System

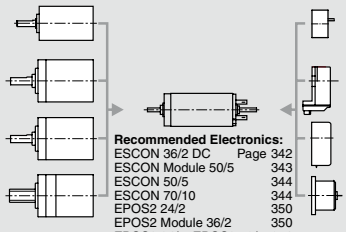
Overview on page 20–25

Planetary Gearhead
∅26 mm
0.75 - 4.5 Nm
Page 270

Planetary Gearhead
∅32 mm
0.75 - 6.0 Nm
Page 272/273/276

Koaxdrive
∅32 mm
1.0 - 4.5 Nm
Page 281

Spindle Drive
∅32 mm
Page 301–303



Recommended Electronics:

ESCON 36/2 DC	Page 342
ESCON Module 50/5	343
ESCON 50/5	344
ESCON 70/10	344
EPOS2 24/2	350
EPOS2 Module 36/2	350
EPOS2 24/5, EPOS2 50/5	351
EPOS2 P 24/5	354
EPOS3 70/10 EtherCAT	357
MAXPOS 50/5	360
Notes	**22**

Encoder MR
128 - 1000 CPT,
3 channels
Page 319

Encoder Enc
22 mm
100 CPT, 2 channels
Page 324

Encoder HED_ 5540
500 CPT,
3 channels
Page 325/327

DC-Tacho DCT
∅22 mm
0.52 V
Page 336