POLITECNICO DI TORINO

FACOLTÀ DI INGEGNERIA

MASTER'S DEGREE

in Mechatronic Engineering

Application of Enhanced Model Reference Adaptive Control (EMRAC) algorithms to the tracking problem of Space Robotic Manipulators

Candidate:
Simone Martini

Supervisor:
Prof. Massimo Violante

Main co-supervisor:
Dott. Umberto Montanaro

Co-supervisor:
Prof. Aldo Sorniotti

a.a 2019/2020

to my family, friends and girlfriend

Abstract

This master thesis proposes the application of the Enhanced Model Reference Adaptive Control (EMRAC) to a space robotic manipulator. The EMRAC aims to be an improvement of the more generic Model Reference Adaptive Control (MRAC) by adding an adaptive integral action and an adaptive switching term to compensate for slow and rapid changing disturbances. Further robustness is achieved by the adoption of the σ-modification strategy that prevents the drifting of the adaptive gains. The robotic arm will replicate a realistic space mission of rescuing a faulty non cooperative micro-satellite. To the best of the author's knowledge, this project application is a novel implementation of the EMRAC controller with σ-modification to a MIMO space robotic system. Simulations show that the EMRAC is capable of controlling highly nonlinear systems affected by parameter uncertainties and unmodelled dynamics, furthermore, the algorithm is robust against disturbances. The thesis presents experimental data from computer simulations of four different implementations of the EMRAC performed using ROS and Gazebo. From simulations data a set of performance indexes are computed, showing that the four EMRAC strategies outperform benchmark controllers equipped with a full-state feedback linearisation control action including two standard controllers (PD and PI strategies) and a robust controller.

Contents

1

Introduction & State of the Art

1.1 Robotic Manipulators in Space Industry

Over the ages, human curiosity and exploration instinct has led us to push the boundaries of what we know. Space exploration is one of the most fascinating and complex challenges for mankind that helps addressing the question of our place in the universe. The growth of the space industry has being vital to advance scientific progress and to boost the development of many areas such as telecommunications, navigation and Earth observation [1].

The increasing number of space missions comes with a growing range of space applications, many of which involve robotics. Space robotics is widely used to assist or replace humans activities, reducing the high risk associated to space tasks [2].
Some of the main tasks for a space manipulator are:

- Debris removal
- Space infrastructure assembly
- On-orbit spacecraft refuelling [3]

All these tasks result in very complex operations due to the harsh challenges of space such as:

- Unpredictable environment
- Unknown and non-cooperative targets
- Microgravity
- Communication delay

To overcome these challenges, high robustness and autonomy of the space manipulator are required.

1.2 Space Manipulator's Control Architectures

The control system is essential for guaranteeing the execution of the command tasks. As today, the state of the art of the control techniques for controlling robotic arms can be summarised by three main categories: Proportional-Integral-Derivative (PID) design, Robust Control and Adaptive algorithm [4].

Even though a lot of advancement and modification have being done from the standard PID, its primary disadvantage is to be a constant gain feedback system without any direct knowledge of the plant, this, combined with its linear behaviour, makes it an unreliable option for controlling a non-linear system affected by noise like a robotic manipulator. More sophisticated control laws to be considered can be found in Robust and non-linear controllers such as Sliding Mode Control (SMC) and Model Predictive Control (MPC), these control strategies are able to ensure stability and performance with bounded uncertainties. Another powerful control approach is the adaptive one, its main benefit is being able to change the control gains to compensate time varying behaviour of the system [4]. Adaptive control is generally divided into three branches: gain scheduling, self tuning and model reference [5]. Adaptive strategies (especially Model Reference adaptive control) are a very worthy candidate to control highly complex and non-linear system because, unlike robust control, they do not require a priori knowledge of the system parameters [4].

The Model Reference Adaptive Control (MRAC) is an adaptive algorithm that allows to impose a reference model dynamics to a system affected by parameter uncertainties, its ability to adapt the control gains allows for achieving good tracking performances and compensating for model mismatches and unknown dynamics [6]. The foundamental property of the MRAC is the possibility to achieve asymptotic tracking of the reference model [7]. Although, in theory, the MRAC algorithm should be able to achieve promising performance, it is very difficult to guarantee stability for highly non-linear system. Usually the MRAC strategy is widely used for controlling quasi-linear system and only few recent articles can be found on the application to robotics. However, as reported in Refs [5] many of this studies showed promising results for the application of the MRAC to robotic manipulators. Many variation of the MRAC have been proposed but in this report, the focus is brought to the Enhanced

Model Reference Adaptive Control (EMRAC).

The EMRAC scheme is an improvement with respect to the standard MRAC thanks to the implementation of:

- An adaptive integral action able to reduce steady states disturbances
- An adaptive switching term able to compensate for the rapid changing behaviour of the disturbances dynamics

The resulting controller is able to achieve higher tracking performances while guaranteeing greater robustness and autonomy [8, 9, 10, 11, 12, 13]. To the best of the author's knowledge, there is no presence in literature of the application of the Enhanced Model Reference Adaptive Control with adaptive integral action and adaptive switching term to the tracking error problem of space robotic manipulator.

1.3 Project's Objective and Thesis Structure

The aim of this project is the design and deployment of the Enhanced Model Reference Adaptive Controller to a space robotic manipulator. The control algorithm will be able to compensate for unmodelled dynamics (such as unknown target mass) and disturbances in the space environment. In particular, the unmodelled disturbance is simulated by an unknown end effector mass representing a non cooperative target. The design has to be scalable to higher degree of freedom manipulators and the real time feasibility of the adaptive controller is shown through the use of an advanced robotic simulation environment.

In the next chapters, first the simulation environment and the adopted workflow for the deployment to a real time system will be presented, second, the description of the space robot manipulator employed for the application will be given. More specifically, Chapter 3 will illustrate the modelling of the robot and the respective model validation. Then, a more in depth analysis of the EMRAC control algorithm will be given, followed by its application. The final chapters will report the data analysis of the simulations' results and final conclusions. Some further development ideas are also introduced at the end of the last chapter.

2

Environment

In order to show the real time feasibility of the adaptive control, the implementation has to be done in an advance robotic simulation environment such as ROS and Gazebo. However, since the EMRAC architecture is not trivial, implementing the controller directly as C++ code would be quite time consuming and inefficient for a preliminary testing phase. For this reason, the controller implementation will be achieved exploiting Simulink graphical interface and, later on, deployed in ROS as C++ code. This Chapter describes the programming environment and the workflow adopted for performing the simulations.

2.1 Programming Environment

2.1.1 ROS

Robot Operating System (ROS) is one of the most popular open source framework for writing robot software. Despite its name, ROS is a meta-operating system that currently runs only on Unix-based platforms such as Ubuntu [14]. Even if, technically, it is not a real time operating system (OS), ROS is provided with its own system clock and communication pipeline [15] and is widely used for testing real time applications thanks to the possibility to integrate it with real time code [14]. ROS provides a set of tools, libraries and conventions that help writing robust robot software [16].

Figure 2.1: ROS logo [17]

2.1.2 Gazebo

Gazebo is a free complex 3D indoor and outdoor multi-robot simulator [18], it comes with robust physics engines such as Open Dynamics Engine (ODE) [19]. Gazebo can easily integrate with ROS through a set of Gazebo plugins that have the same ROS message interface, this allows to highly facilitate the deployment of the physical robot and the application development [18].

Figure 2.2: Gazebo logo [20]

2.1.3 MATLAB & Simulink

MATLAB is a programming platform that can be used for data analysis, algorithm development and model and application creation. It uses the MATLAB matrix-based language, useful for computational mathematics. MATLAB is widely used among engineers for numerous applications, including control system design [21]. Thanks to the integration with Simulink, it is possible to incorporate MATLAB algorithms into models and perform multidomain simulations. Simulink is a block diagram environment that, among its many features, allows for modelling and simulating dynamic systems and for automatic code generation [22]. These two features are very useful for this project' purpose, allowing to design and test the controller in an user friendly environment and, only after, deploy the code in the ROS environment as C++ code.

2.2 Workflow

In this section, a more comprehensive explanation of the adopted workflow is given. In order to integrate MATLAB and Simulink with ROS, the ROS Toolbox add on needs to be installed. This toolbox allows to generate a network of ROS nodes and it is provided with 'Publish' [23] and 'Subscribe' [24] blocks that are used to send and receive messages from ROS topics [25]. The main steps of the workflow are described as follow:

- Tune the Controller in Simulink
- Create an equivalent control system for code generation (using the Subscribe and Publish blocks)
- Automatic code generation using Simulink Coder and Embedded Coder
- Build the code in the ROS environment

After the deployment in ROS, it is necessary to establish a communication between the robot in Gazebo and the generated controller. This communication can be established by creating a C++ controller that subscribes to the deployed controller' topic and uses an interface called 'EffortJointInterface' to send commands to the robot. An example of C++ controller template can be found in [26] and the used 'my_controller.cpp' code is attached in appendix A.4.

3

Case study: 3DOF Manipulator

A robot manipulator can be described as a connection of rigid body called links through a set of joints. It is often referred as a kinematic chain due to the fact that the end effector position and velocity are computed by a combination of the motion of each link with respect to the previous one [27]. Two of the most known space robotic arms are the Canadarm (attached to the space shuttle and returned to earth) and the Canadarm2 (permanently attached to the International Space Station) that are respectively six and seven degrees of freedom manipulators composed by shoulder, elbow and wrist similar to a human arm [28].

Figure 3.1: Astronaut Jerry Ross secured on a foot restraint device connected to the Canadarm aboard the Earth orbiting Atlantis, Nov, 1985

Even though the EMRAC controller can be applied to higher degree of freedom, for this project, a simple three degree of freedom anthropomorphic arm is used. The anthropomorphic arm is composed by two revolute joints in the shoulder and one revolute joint in the elbow. For simplicity, no wrist is attached to the end effector. The robot is modelled by means of the Unified Robotic Description format (URDF) and by means of a set of ordinary differential equation derived from the lagrangian of the system.

3.1 Mathematical Modelling

The following subsections are a synthesis of the "Kinematics", "Differential Kinematics and Statics" and "Dynamics" chapters of [27] adapted to the anthropomorphic space robotic arm used in this project.

3.1.1 Forward Kinematics

The pose of the manipulator is given by its position and orientation with respect to a reference frame. The base of the robotic arm is considered static during the command activity and is chosen as the fixed reference frame. Also, to the end of each link of the arm is associated a frame. Given this structure, the open kinematic chain can be solved recursively as a product between homogeneous transform matrices, each of which is a function of a single joint variable q [27]. To compute the direct kinematic equation, a systematic approach such as the Denavit-Hartenberg convention is used. The following two tables report the Denavit-Hartenberg parameter and the dimension parameters for the three degrees of freedom anthropomorphic manipulator respectively.

Table 3.1: Denavit-Hartenberg parameter for the three degrees of freedom manipulator

Link	a_i	α_i	d_i	θ_i
1	0	$\pi/2$	L_1	q_1
2	L_2	0	0	q_2
3	L_3	0	0	q_3

Table 3.2: Three degrees of freedom manipulator's parameters

Link	Length(m)	Mass(Kg)	Diameter(m)
1	$L_1 = 0.5$	$m_{\ell_1} = 50$	$2r = 0.4$
2	$L_2 = 4$	$m_{\ell_2} = 200$	$2r = 0.4$
3	$L_3 = 4$	$m_{\ell_3} = 200$	$2r = 0.4$

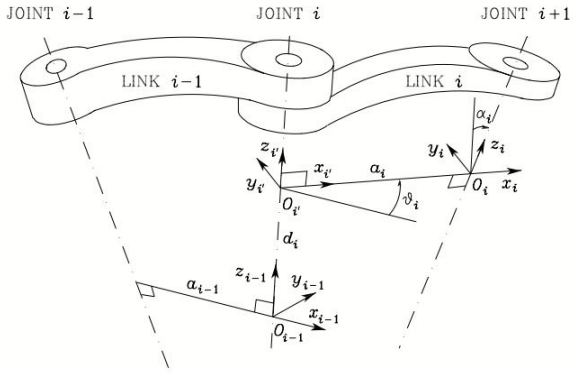


Figure 3.2: Denavit–Hartenberg kinematic parameters [27]

Where:

- a_i is the distance between O_i and $O_{i'}$ along x_i-axis
- d_i is the distance between O_i and $O_{i'}$ along z_i-axis
- θ is the angle between z_i and z_{i-1} about the x_i-axis
- α is the angle between x_i and x_{i-1} about the z_i-axis

The generic homogeneous matrix between two link can be computed as follow:

$$A_i^{i-1}(q_i) = \begin{bmatrix} c_{\theta_i} & -s_{\theta_i}c_{\alpha_i} & s_{\theta_i}s_{\alpha_i} & a_i c_{\theta_i} \\ s_{\theta_i} & c_{\theta_i}c_{\alpha_i} & -c_{\theta_i}s_{\alpha_i} & a_i s_{\theta_i} \\ 0 & s_{\alpha_i} & c_{\alpha_i} & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.1}$$

Considering θ_i as the genetic joint variable q_i it follows that:

$$A_1^0(q_1) = \begin{bmatrix} c_1 & 0 & s_1 & 0 \\ s_1 & 0 & -c_1 & 0 \\ 0 & 1 & 0 & L_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$
$$A_i^{i-1}(q_i) = \begin{bmatrix} c_i & -s_i & 0 & L_i c_i \\ s_i & c_i & 0 & L_i s_i \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad i = 2, 3. \tag{3.2}$$

Finally, the direct forward kinematic function can be computed as:

$$T_3^0(q) = A_1^0 A_2^1 A_3^2 = \begin{bmatrix} c_1 c_{23} & -c_1 s_{23} & s_1 & c_1(L_2 c_2 + L_3 c_{23}) \\ s_1 c_{23} & -s_1 s_{23} & -c_1 & s_1(L_2 c_2 + L_3 c_{23}) \\ s_{23} & c_{23} & 0 & L_1 + L_2 s_2 + L_3 s_{23} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.3}$$

Through the matrix $T_3^0(q)$ it is possible to compute the end effector position as a function of the joint variable [27].

3.1.2 Inverse Kinematics

Since the main goal of the manipulator is to track a desired trajectory, it is necessary to use an inverse kinematic model in order to associate a desired motion of the end effector to the each joint variable' motion. Finding this model is not an easy task due to the inverse kinematic problem. The complexity of the kinematic problem is due to the fact that:

- The equation to solve are usually nonlinear and thus it might not be possible to find the inverse analytically.
- Numerical approach might also fail in case of multiple or infinite solution (the same end effector position might be achievable through different joint configurations)
- there might not be any admissible solution e.g point outside the manipulator dexterous workspace.

By using the differential kinematic equation it is possible to invert the kinematic.

$$\dot{x}_e = \begin{bmatrix} \dot{p}_e \\ \dot{\phi}_e \end{bmatrix} = \begin{bmatrix} J_p(q) \\ J_\phi(q) \end{bmatrix} \dot{q} = J_A(q)\dot{q} \tag{3.4}$$

The motion trajectory in terms of position, velocity and acceleration can be inverted into the equivalent joint position, velocity and acceleration exploiting the Analytical Jacobian J_A. Since the end-effector of the anthropomorphic arm is a point, for simplicity, its orientation is not taken into consideration. The differential kinematic equation to be considered becomes the following

$$\dot{p}_e = \frac{\partial p_e}{\partial q}\dot{q} = J_P(q)\dot{q} \tag{3.5}$$

Where J_P is the $(3 \times n)$ matrix that relate the joint velocities $\dot{q}$ to the end effector linear velocity $\dot{p}_e$

$$J_P = \begin{bmatrix} j_{P_1} & \dots & j_{P_n} \end{bmatrix} \tag{3.6}$$

j_{P_i} is the generic component of J_P and, for revolute joints, can be computed as follow

$$j_{P_i} = z_{i-1} \times (p_e - p_{i-1}) \tag{3.7}$$

Being

- n the degrees of freedom of the manipulator
- z_{i-1} the unit vector of joint $i-1$ axis
- z_{i-1} and z_i are the position vectors of the origins O_{i-1} and O_i
- p_e the position vector of the end effector that for the anthropomorphic manipulator coincides with p_3

The resulting matrix is

$$
\begin{aligned}
J_P &= \begin{bmatrix} z_0 \times (p_3 - p_0) & z_1 \times (p_3 - p_1) & z_2 \times (p_3 - p_2) \end{bmatrix} = \\
&= \begin{bmatrix} -s_1(L_2c_2 + L_3c_{23}) & -c_1(L_2s_2 + L_3s_{23}) & -L_3c_1s_{23} \\ c_1(L_2c_2 + L_3c_{23}) & -s_1(L_2s_2 + L_3s_{23}) & -L_3s_1s_{23} \\ 0 & L_2c_2 + L_3c_{23} & L_3c_{23} \end{bmatrix}
\end{aligned} \tag{3.8}
$$

By doing the time differentiation of the differential kinematic equation, we obtain the operational space acceleration as function of the joint acceleration

$$\ddot{p}_e = J_P(q)\ddot{q} + \dot{J}_P(q,\dot{q})\dot{q} \tag{3.9}$$

The determinant of J_P is

$$|J_P| = -L_2L_3(L_2c_2s_3 - L_3s_2 + L_3s_2c_3^2 + L_3c_2c_3s_3) \tag{3.10}$$

Assuming q_2 and q_3 such that $|J_P| \neq 0$, the matrix J_P is square and non-singular. Therefore, the second order kinematic equation can be inverted.

$$\ddot{q} = J_P^{-1}(q)(\ddot{p}_e - \dot{J}_P(q,\dot{q})\dot{q}) \tag{3.11}$$

Considering the drift due to the integration of 3.5 and 3.11, it is worth introducing the position error and its first and second order time derivative

$$
\begin{aligned}
e &= p_d - p_e \\
\dot{e} &= \dot{p}_d - \dot{p}_e \\
\ddot{e} &= \ddot{p}_d - \ddot{p}_e
\end{aligned} \tag{3.12}
$$

Substituting into 3.9 leads to

$$\ddot{e} = \ddot{p}_d - J_P(q)\ddot{q} - \dot{J}_P(q,\dot{q})\dot{q} \tag{3.13}$$

In order to ensure convergence, of the error it is possible to create a closed-loop inverse kinematic model choosing the joint acceleration vector as

$$\ddot{q} = J_P^{-1}(q)(\ddot{p}_d + K_D\dot{e} + K_Pe - \dot{J}_P(q,\dot{q})\dot{q}) \tag{3.14}$$

Finally, the equation 3.13 becomes

$$\ddot{e} + K_D\dot{e} + K_P e = 0 \tag{3.15}$$

The obtained linear error system is asymptotically stable. K_P and K_D are positive definite diagonal matrix that determine the speed of convergence of the error to zero [27].
The equivalent block scheme of the second order inverse kinematic algorithm is shown in Fig.3.3

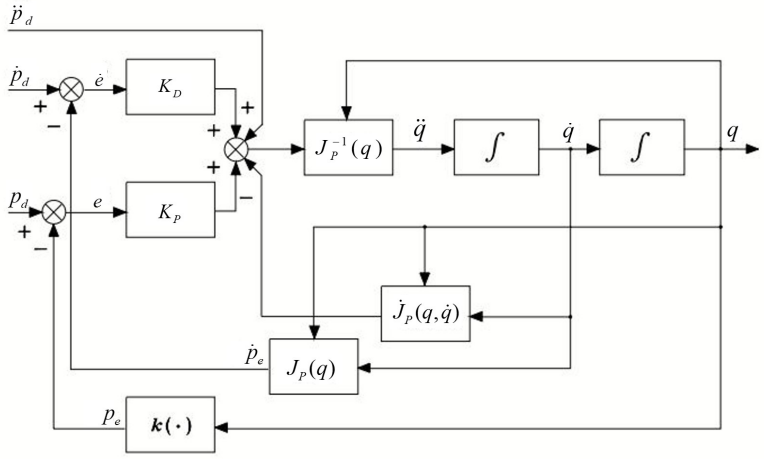


Figure 3.3: Block scheme of the second-order inverse kinematic algorithm [27]

3.1.3 Dynamics

The dynamic model of the manipulator is useful for preliminary simulation of motion and for designing the control algorithm. The joint space dynamic model for a space manipulator can be described by the equation of motion:

$$B(q)\ddot{q} + C(q,\dot{q})\dot{q} + F_v\dot{q} = \tau \tag{3.16}$$

where:

- $B(q) \in \mathbb{R}^{n\times n}$ is the inertia matrix
- $C(q,\dot{q}) \in \mathbb{R}^{n\times n}$ is the Christoffel matrix accounting for centrifugal and Coriolis effect

- $F_v \in \mathbb{R}^{n \times n}$ is the diagonal matrix of viscous friction coefficients
- $\tau \in \mathbb{R}^n$ is the vector of joint torques.

The absence of the gravitational term is due to the fact that the space manipulator is supposed to work in a microgravity environment, therefore, the gravitational acceleration is approximated to be null. For simplicity, also no Coulomb friction torques and end effector' contact forces are considered.

The equation of motion 3.16 can be derived through a systematic method based on the Lagrangian Formulation. The Lagrangian of the mechanical system is:

$$\mathcal{L} = \mathcal{T} - \mathcal{U} \tag{3.17}$$

$\mathcal{T}$ and $\mathcal{U}$ are the kinetic energy and potential energy respectively.
The Lagrange equation is given by

$$\frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{q}}\right)^T - \left(\frac{\partial \mathcal{L}}{\partial q}\right)^T = \xi \tag{3.18}$$

and sets the relationship between the generalized forces applied to the manipulator ξ and the joint positions q, velocities $\dot{q}$ and accelerations $\ddot{q}$ [27].
So, the dynamical model can be derive through the computation of the kinetic energy. Note that, by not considering the gravitational term, the potential mechanical energy is considered to be null as well.

Kinematic Energy

For a robotic arm with n rigid links the total kinetic energy is given by the following equation

$$\mathcal{T} = \sum_{i=1}^{n} (\mathcal{T}_{\ell_i} + \mathcal{T}_{m_i}) \tag{3.19}$$

with $\mathcal{T}_{\ell_i}$ being the contribution of kinetic energy of Link i and $\mathcal{T}_{m_i}$ being the contribution of kinetic energy relative to the motor actuating joint i.
$\mathcal{T}_{\ell_i}$ can be computed by considering the kinetic energy contribution of each link i.
By defining:

- p_{ℓ_i} the position vector of centre of mass of Link i
- p_{j-1} the position vector of the origin of the frame $j-1$
- z_{j-1} the unit vector of axis z of frame $j-1$
- m_{ℓ_i} the mass of the Link i
- $I^i_{\ell_i}$ the constant inertia tensor of Link i

it is possible to apply the Jacobian computation to the intermediate link, leading to

$$\begin{aligned} J_P^{\ell_i} &= \begin{bmatrix} j_{P_1}^{\ell_i} & \dots & j_{P_n}^{\ell_i} & 0 & \dots & 0 \end{bmatrix} \\ J_O^{\ell_i} &= \begin{bmatrix} j_{O_1}^{\ell_i} & \dots & j_{O_n}^{\ell_i} & 0 & \dots & 0 \end{bmatrix} \end{aligned} \tag{3.20}$$

where, for revolute joints

$$\begin{aligned} j_{P_j}^{\ell_i} &= z_{j-1} \times (p_{\ell_i} - p_{j-1}) \\ j_{O_j}^{\ell_i} &= z_{j-1} \end{aligned} \tag{3.21}$$

The kinetic energy contribution of each link i is given by

$$\mathcal{T}_{\ell_i} = \frac{1}{2} m_{\ell_i} \dot{q}^T {J_P^{\ell_i}}^T J_P^{\ell_i} \dot{q} + \frac{1}{2} \dot{q}^T {J_O^{\ell_i}}^T R_i I^i_{\ell_i} R_i^T J_O^{\ell_i} \dot{q} \tag{3.22}$$

A similar procedure can be used for finding $\mathcal{T}_{m_i}$
By defining:

- p_{m_i} the position vector of centre of mass of the rotor moving joint i
- p_{j-1} the position vector of the origin of the frame $j-1$
- z_{j-1} the unit vector of axis z of frame $j-1$
- k_{r_i} the gear reduction ratio
- z_{m_i} unit vector of joint rotation axis
- m_{m_i} the mass of the rotor moving joint i
- $I^{m_i}_{m_i}$ the constant inertia tensor of the rotor moving joint i relative to its centre of mass

The Jacobian to be computed are

$$\begin{aligned} J_P^{m_i} &= \begin{bmatrix} j_{P_1}^{m_i} & \dots & j_{P_n}^{m_i} & 0 & \dots & 0 \end{bmatrix} \\ J_O^{m_i} &= \begin{bmatrix} j_{O_1}^{m_i} & \dots & j_{O_n}^{m_i} & 0 & \dots & 0 \end{bmatrix} \end{aligned} \tag{3.23}$$

where, for revolute joints

$$\begin{aligned} j_{P_j}^{m_i} &= z_{j-1} \times (p_{m_i} - p_{j-1}) \\ j_{O_j}^{m_i} &= \begin{cases} j_{O_j}^{\ell_i} & j = 1, \dots, i-1 \\ k_{r_i} z_{m_i} & j = i \end{cases} \end{aligned} \tag{3.24}$$

The kinetic energy contribution of each Rotor i is given by

$$\mathcal{T}_{m_i} = \frac{1}{2} m_{m_i} \dot{q}^T {J_P^{m_i}}^T J_P^{m_i} \dot{q} + \frac{1}{2} \dot{q}^T {J_O^{m_i}}^T R_{m_i} I_{m_i}^{m_i} R_{m_i}^T J_O^{m_i} \dot{q} \tag{3.25}$$

Having computed the kinematic energy of the manipulator, it is possible to proceed the derivation of the equation of motion by applying 3.18

Inertia Matrix

The inertia matrix represent the contribution of the moment of inertia at each joint axis and the coupling acceleration effect of each joint with respect to each other. From the previous section the kinematic energy can be written in the quadratic form

$$\mathcal{T} = \frac{1}{2} \dot{q}^T B(q) \dot{q} \tag{3.26}$$

Therefore, the inertia matrix can be defined as

$$\begin{aligned} B(q) = \sum_{i=1}^{n} \; & (m_{\ell_i} {J_P^{\ell_i}}^T J_P^{\ell_i} + {J_O^{\ell_i}}^T R_i I_{\ell_i}^i R_i^T J_O^{\ell_i} + \\ & + m_{m_i} {J_P^{m_i}}^T J_P^{m_i} + {J_O^{m_i}}^T R_{m_i} I_{m_i}^{m_i} R_{m_i}^T J_O^{m_i}) \end{aligned} \tag{3.27}$$

leading, for the anthropomorphic space manipulator, to a (3×3) matrix with components:

$$
\begin{aligned}
b_{11} &= I^y_{\ell_1} + I^y_{\ell_2} + I^y_{\ell_3} + I^y_{m_2} + I^y_{m_3} + I^y_{m_1} k^2_{r_1} + m_{\ell_3}(l_3 c_{23} + L_2 c_2)^2 + \\
&\quad + I^x_{\ell_3} s^2_{23} - I^y_{\ell_3} s^2_{23} + I^x_{m_3} s^2_{23} - I^y_{m_3} s^2_{23} + I^x_{\ell_2} s^2_2 - I^y_{\ell_2} s^2_2 + \\
&\quad + I^x_{m_2} s^2_2 - I^y_{m_2} s^2_2 - L^2_2 m_{m_3}(s^2_2 - 1) - l^2_2 m_{\ell_2}(s^2_2 - 1); \\
b_{12} &= 0; \\
b_{13} &= 0; \\
b_{21} &= 0; \\
b_{22} &= I^z_{\ell_2} + I^z_{\ell_3} + I^z_{m_3} + I^z_{m_2} k^2_{r_2} + L^2_2 m_{m_3} + l^2_2 m_{\ell_2} + m_{\ell_3}(L^2_2 + 2 c_3 L_2 l_3 + l^2_3); \\
b_{23} &= I^z_{\ell_3} + I^z_{m_3} k_{r_3} + m_{\ell_3}(l^2_3 + c_3 L_2 l_3); \\
b_{31} &= 0; \\
b_{32} &= I^z_{\ell_3} + I^z_{m_3} k_{r_3} + m_{\ell_3}(l^2_3 + c_3 L_2 l_3); \\
b_{33} &= I^z_{\ell_3} + I^z_{m_3} k^2_{r_3} + m_{\ell_3} l^2_3;
\end{aligned}
\tag{3.28}
$$

Christoffel Matrix

The Christoffel matrix C is represent the contribution of the centrifugal effect and Coriolis effect induced by each joint with respect to the other.
The Christoffel symbols of the first type are defined as

$$
c_{ijk} = \frac{1}{2}\left(\frac{\partial b_{ij}}{\partial q_k} + \frac{\partial b_{ik}}{\partial q_j} - \frac{\partial b_{jk}}{\partial q_i}\right) \tag{3.29}
$$

and allow for the computation of the element of the Christoffel matrix

$$
c_{ij} = \sum_{k=1}^{n} c_{ijk} \dot{q}_k \tag{3.30}
$$

For the case of the robotic arm used in this project, the C-matrix has components

$$
\begin{aligned}
c_{11} &= \dot{q}_2 h_1 + \dot{q}_3 h_2; \\
c_{12} &= \dot{q}_1 h_1; \\
c_{13} &= \dot{q}_1 h_2; \\
c_{21} &= -\dot{q}_1 h_1; \\
c_{22} &= \dot{q}_3 h_3; \\
c_{23} &= (\dot{q}_2 + \dot{q}_3) h_3; \\
c_{31} &= -\dot{q}_1 h_2; \\
c_{32} &= -\dot{q}_2 h_3; \\
c_{33} &= 0;
\end{aligned}
\tag{3.31}
$$

with

$$
\begin{aligned}
h_1 &= (I^x_{\ell_3} - I^y_{\ell_3} + I^x_{m_3} - I^y_{m_3} - l_3^2 m_{\ell_3})\tfrac{\sin(2q_2+2q_3)}{2} + \\
&\quad + (+I^x_{\ell_2} - I^y_{\ell_2} + I^x_{m_2} - I^y_{m_2} - L_2^2 m_{\ell_3} - L_2^2 m_{m_3} - l_2^2 m_{\ell_2})\tfrac{\sin(2q_2)}{2} + \\
&\quad - L_2 l_3 m_{\ell_3} \tfrac{\sin(2q_2+q_3)}{2}; \\
h_2 &= (+I^x_{\ell_3} - I^y_{\ell_3} + I^x_{m_3} - I^y_{m_3} - l_3^2 m_{\ell_3})\tfrac{\sin(2q_2+2q_3)}{2} - L_2 l_3 m_{\ell_3} \left(\tfrac{\sin(q_3)+\sin(2q_2+q_3)}{2}\right); \\
h_3 &= -L_2 l_3 m_{\ell_3} s_3;
\end{aligned}
\tag{3.32}
$$

Dynamic Parameters

In this section the parameters value chosen for the simulation of the space robotic arm will be listed.
The damping coefficient matrix is

$$
F_v = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & f \end{bmatrix} \tag{3.33}
$$

where $f = 2Ns/m$.
For the model simplicity, all rotor mass are considered to be null ($m_{m_i} = 0Kg$) and all gear reduction ratio are chosen unitary ($k_{r_i} = 1$). Therefore, $I^x_{m_i} = I^y_{m_i} = I^z_{m_i} = 0Kgm^2$.
With reference to Tab 3.2, the centre of mass of each link is located at the link's half length ($l_i = L_i/2$).
For Link 1 the inertia tensor is given by the cylinder inertia tensor

$$
I_{\ell_1} = \begin{bmatrix} I^x_{\ell_1} & 0 & 0 \\ 0 & I^y_{\ell_1} & 0 \\ 0 & 0 & I^z_{\ell_1} \end{bmatrix} = \begin{bmatrix} \frac{1}{12} m_{\ell_1}(3r^2 + L_1^2) & 0 & 0 \\ 0 & \frac{1}{12} m_{\ell_1}(3r^2 + L_1^2) & 0 \\ 0 & 0 & \frac{1}{2} m_{\ell_1} r^2 \end{bmatrix} \tag{3.34}
$$

For Link 1 and Link 2, being the longitudinal dimension much bigger with respect to the cylinder radius, the inertia tensor can be approximated to the

rod inertia tensor

$$I_{\ell_i} = \begin{bmatrix} I^x_{\ell_i} & 0 & 0 \\ 0 & I^y_{\ell_i} & 0 \\ 0 & 0 & I^z_{\ell_i} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & \frac{1}{2}m_{\ell_i}L_i^2 & 0 \\ 0 & 0 & \frac{1}{2}m_{\ell_i}L_i^2 \end{bmatrix} \quad i = 2, 3. \tag{3.35}$$

3.2 URDF Modelling

In order to perform the simulation in ROS and Gazebo, the robot has to be modelled in a way that the simulation environment can interpret. The URDF is an XML format for representing a robot model. ROS can parse the URDF through a C++ parser contained in the "urdf" package [29]. In particular, for this project, the format used is the "urdf.xacro", with Xacro being a XML macro language that allow for constructing shorter and more readable XML files through the use of macros [30]. The parameters used for the manipulator dimensions are the same as the previous section, by doing so, the generated URDF file will describe a robot model equivalent to the mathematical one.

3.2.1 Code Explanation

In order for the URDF file to be parse correctly, a namespace must be specified [31]. For a xacro file the following lines are needed

```
<?xml version="1.0"?>
<robot name="space_arm"
    xmlns:xacro="http://www.ros.org/wiki/xacro">
```

The space robot model is built connecting a series of link to each other using joints. URDF allows to use a series of tag to describe these objects. First, since the robot will have a fixed base while performing the simulations, the robot needs to be permanently attached to the world frame. To achieve this, the base link is attached to a reference link called 'world' by means of a 'fixed' joint type [32]. The 'joint' and 'link' tags allow to specify the properties of the robot such as mass and spawn position of each link. The base is a box shape fixed link with high mass and inertia, representing the spacecraft to which the manipulator is attached. Since the mathematical model doesn't have a base link, in order to have coherence within the two robots, the base link is spawned underneath the ground level. In this way, the robotic arm has the first link starting at the origin of the world frame. The resulting lines of code are

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

<joint name="world_joint" type="fixed">
<parent link="world"/>
<child link="base_link"/>
<origin rpy="0 0 0" xyz="0 0 0"/>
</joint>

<link name="base_link">
<visual>
<origin rpy="0 0 0" xyz="0 0 -0.5"/>
<geometry>
<box size="1 1 1"/>
</geometry>
</visual>
<collision>
<origin rpy="0 0 0" xyz="0 0 -0.5"/>
<geometry>
<box size="1 1 1"/>
</geometry>
</collision>
<inertial>
<origin rpy="0 0 0" xyz="0 0 -0.5"/>
<mass value="300"/>
<inertia ixx="300" ixy="0.0" ixz="0.0"
         iyy="300" iyz="0.0" izz="300"/>
</inertial>
</link>
```

The rest of the robotic arm in modelled by connecting a series of links with continuous type joint (Full code in appendix A.1)[33]. The 'continuous' joint is the equivalent of a revolute joint without speed and angle limitations. Since the geometry tag 'cylinder' creates a cylinder which has the longitudinal dimension on the z-axis, it cannot be used for the second and third link of the manipulator which, from the mathematical model, require the longitudinal dimension on the x-axis. For this reason, the second and third link will be modelled as a rod-

like hyperrectangle using the 'box' tag that allows to choose the the length for each axes (Fig 3.4). By using xacro, it is possible to include other files inside our URDF Xacro file. With the following line of code

```
<xacro:include filename=
    "$(find surrey_space_arm_description)/
    urdf/surrey_3dof_space_arm.gazebo"/>
```

a '.gazebo' file is included. This file (Full code in appendix A.2) add some graphical customization and the 'gazebo_ros_control' plugin that will be used to simulate the robot's controller in gazebo.
A 'transmission' block with 'hardware_interface::EffortJointInterface' ros_control interface needs to be added for each joint that the controller has to actuate [34]. This segment of the code also allows to set the mechanical reduction ratio of the motor equal to the mathematical model ($k_{r_i} = 1$).

```
<transmission name="tran1">
<type>transmission_interface/SimpleTransmission</type>
<joint name="joint_1">
<hardwareInterface>hardware_interface/EffortJointInterface
</hardwareInterface>
</joint>
<actuator name="motor1">
<hardwareInterface>hardware_interface/EffortJointInterface
</hardwareInterface>
<mechanicalReduction>1</mechanicalReduction>
</actuator>
</transmission>
```

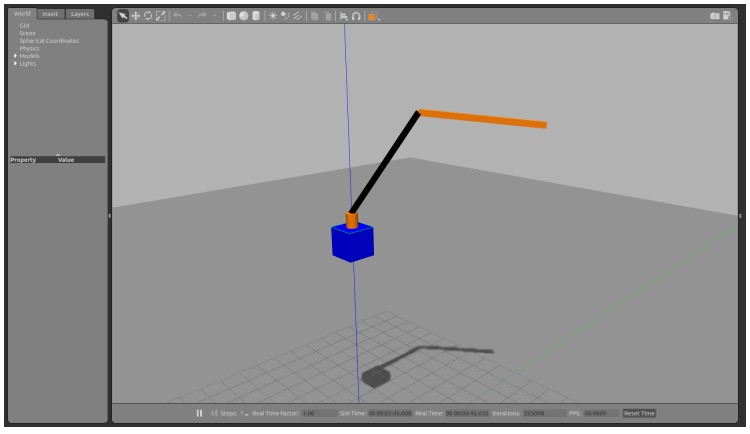


Figure 3.4: URDF file opened in Gazebo

3.2.2 Non Cooperative Target Disturbance

A second version of the URDF file with an end-effector mass will be needed for simulating the response of the controller to a non cooperative target attached to the end-effector the manipulator. The goal is to replicate a realistic scenario of faulty micro-satellite rescued by the manipulator. The satellite thrusters try to counteract the trajectory imposed by the space robotic arm. The micro-satellite body is modelled as an additional link with a $1m$ radius spherical shape. The disturbance is applied to the body through the 'apply_body_wrench' service [35] with the help of a 'IMU' plugin sensor [36].

```
<link name="imu_link">
<visual>
<origin rpy="0 0 0" xyz="0 0 0"/>
<geometry>
<sphere radius="1.0"/>
</geometry>
</visual>
<inertial>
<origin rpy="0 0 0" xyz="0 0 0"/>
<mass value="100"/>
<inertia ixx="40" ixy="0.0" ixz="0.0"
         iyy="40" iyz="0.0" izz="40"/>
</inertial>
</link>

<!-- IMU Plug-in -->
<gazebo reference="imu_link">
<gravity>true</gravity>
<sensor name="imu_sensor" type="imu">
<always_on>true</always_on>
<update_rate>100</update_rate>
<visualize>true</visualize>
<topic>__default_topic__</topic>
<plugin filename="libgazebo_ros_imu_sensor.so"
name="imu_plugin">
<topicName>imu</topicName>
<bodyName>imu_link</bodyName>
```

```
<updateRateHZ>100.0</updateRateHZ>
<gaussianNoise>0.0</gaussianNoise>
<xyzOffset>0 0 0</xyzOffset>
<rpyOffset>0 0 0</rpyOffset>
<frameName>imu_link</frameName>
<initialOrientationAsReference>false
</initialOrientationAsReference>
</plugin>
<pose>0 0 0 0 0 0</pose>
</sensor>
</gazebo>
```

The disturbance model used to simulate the thruster force to oppose the movement is

$$\eta = -ka \tag{3.36}$$

where k is the proportional thruster gain and a is the linear acceleration of the end-effector mass. The IMU sensor sends the acceleration data to a python script that communicate the computed force to the 'apply_body_wrench' service.

3.3 Model Validation

Before using the mathematical model of the robot for control purposes, the model has to be validated to ensure that the dynamic behaviour is comparable to the one of the URDF model used for simulations. In order to compare the two models, they need to be simulated in the same environment.

3.3.1 MATLAB Simulink URDF Import

Exploiting the MathWorks Simscape Multibody environment, it is possible to import the URDF file using the command

```
smimport('model.urdf')
```

in MATLAB command prompt. This command maps the URDF 'robot' element into a Simscape Multibody model [37]. The Robot model placed inside a Simulink file and can be used in simulation through the 'Simulink-PS Converter' and 'PS-Simulink Converter' blocks. The 'gravity' option in the

'Mechanism Configuration' block has to be set null in order to match the absence of the gravitational term of the mathematical model. Fig 3.5 displays the resulting Simulink block scheme of the space manipulaotr.

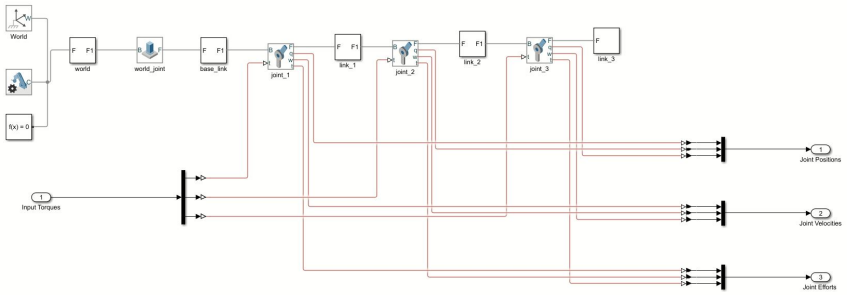


Figure 3.5: Imported URDF Robot in MATLAB Simulink.

3.3.2 Mathematical Simulink Model

The Simulink Mathematical model derivation is a straight forward implementation of the equation of motion (3.16) into a Simulink block scheme like shown in Fig 3.6. The 'State-Space' block, associated with the proper matrices, function as a double integrator.

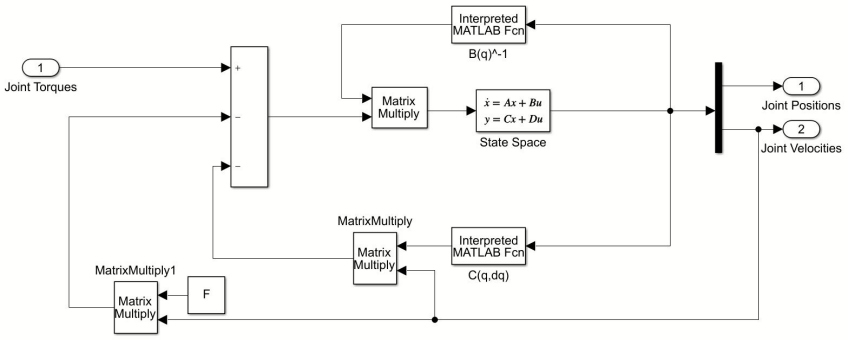


Figure 3.6: MATLAB Simulink equivalent mathematical model.

3.3.3 Model Comparison

Having both of the models into the same environment, the model validation is done providing them with the same open loop input torque and comparing the respective joint positions and velocities. The torque inputs used for the model validation are sinusoidal signals with characteristic described in Tab 3.3. With the naming 'Normal' and 'Strong' referring to a real-like scenario and a higher,

more conservative torque amplitude respectively.

Table 3.3: Sinusoidal open loop input torques

	Amplitude(Nm)	Frequency(rad/s)	Phase(rad)
Normal	$100 \begin{bmatrix} 1 \\ 0.7 \\ 0.4 \end{bmatrix}$	$2\pi/20$	0
Strong	$500 \begin{bmatrix} 1 \\ 0.7 \\ 0.4 \end{bmatrix}$	$2\pi/20$	0

Three different simulation over a time period of $40s$ are performed:

- Mathematical model vs URDF imported model with 'Normal' Input
- Mathematical model vs URDF imported model with 'Strong' Input
- Mathematical model vs URDF with a $100Kg$ end-effector mass imported model with 'Normal' Input

The simulation results are displayed in Fig 3.7, 3.8, 3.9 and the resulting data are analysed using the following performance indexes

- The maximum absolute (ME) value of error

$$ME = \max\left(|r_{ref}(t) - r(t)|\right) \tag{3.37}$$

  where $r_{ref}(t)$ and $r(t)$ are the reference and actual variable respectively.

- The root mean square error ($RMSE$) value

$$RMSE = \sqrt{\frac{1}{t_f - t_i}\int_{t_i}^{t_f} \left(r_{ref}(t) - r(t)\right)^2 dt} \tag{3.38}$$

  where t_f and t_i are the initial and final times of the simulation [38].

Even with relatively high input torque, the dynamics of the URDF robot are well captured by the model derived from the equation of motion 3.16. As expected, the behaviour of the robot with a $100Kg$ end effector mass is fairly different from the mathematical model.

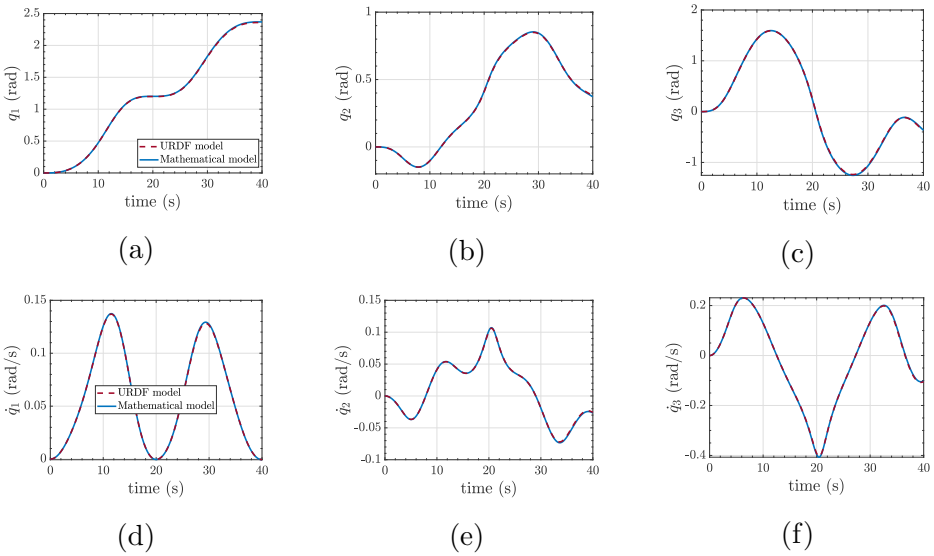


Figure 3.7: Mathematical model vs URDF imported model with 'Normal' Input

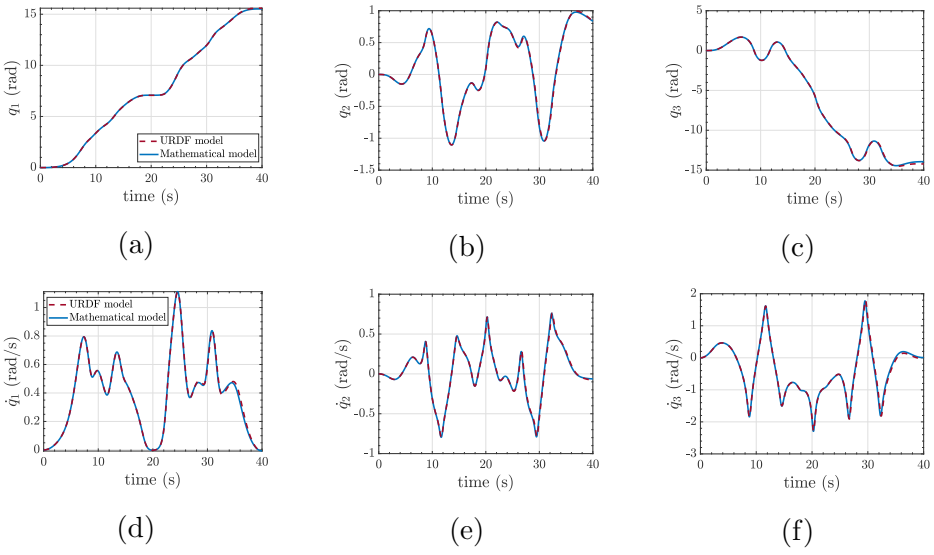


Figure 3.8: Mathematical model vs URDF imported model with 'Strong' Input

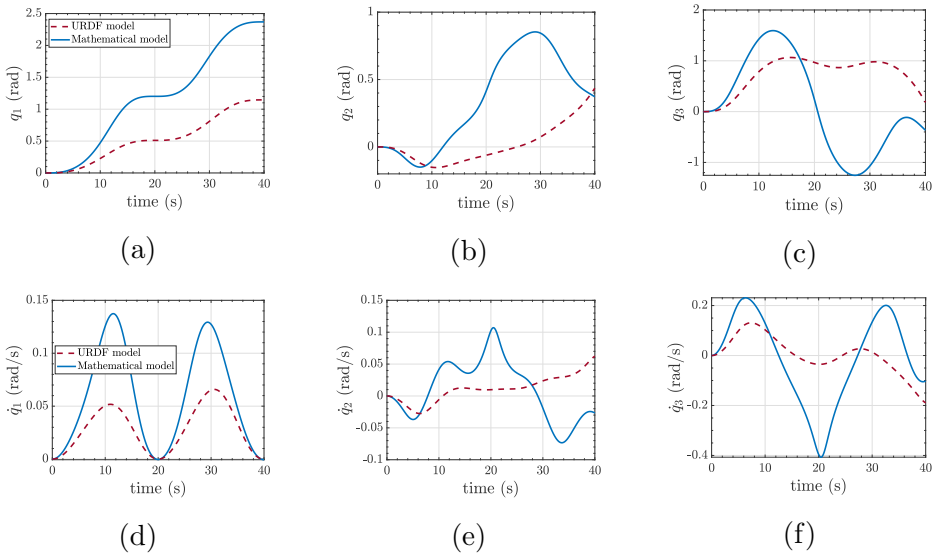


Figure 3.9: Mathematical model vs URDF with a $100Kg$ end-effector mass imported model with 'Normal' Input

Table 3.4: Maximum Joint Position Error

	q_1	q_2	q_3
Normal	0,010105521	0,007430467	0,016122063
Strong	0,075311286	0,060438488	0,292418505
$100Kg$	1,222905338	0,812475789	2,171963564

Table 3.5: Maximum Joint Velocity Error

	$\dot{q_1}$	$\dot{q_2}$	$\dot{q_3}$
Normal	0,001194227	0,001762023	0,004694169
Strong	0,037854994	0,062042062	0,177007495
$100Kg$	0,086172991	0,102883352	0,371941761

Table 3.6: Maximum Joint Position and Velocity Norms Error

	$\|q\|$	$\|\dot{q}\|$
Normal	0,018625014	0,00501444
Strong	0,301846216	0,188075226
$100Kg$	2,484489971	0,384370474

Table 3.7: RMSE: Joint Position

	q_1	q_2	q_3
Normal	0,005570571	0,002907044	0,009255633
Strong	0,022069766	0,016381556	0,085754679
$100Kg$	0,766167075	0,448607565	1,154543109

Table 3.8: RMSE: Joint Velocity

	$\dot{q_1}$	$\dot{q_2}$	$\dot{q_3}$
Normal	0,00042585	0,000698071	0,001938762
Strong	0,010412544	0,016592983	0,052484388
$100Kg$	0,040995032	0,0544968	0,157085593

Table 3.9: RMSE: Joint Position and Velocity Norms

	$\|q\|$	$\|\dot{q}\|$
Normal	0,011186998	0,002104151
Strong	0,090051624	0,05602106
$100Kg$	1,45644448	0,171249459

4

The Enhanced Model Reference Adaptive Control (EMRAC)

Adaptive controller, such as the MRAC, are widely researched especially for system with intense dynamics and parameter uncertainties. In this chapter, an in-depth description of the Enhanced Model Reference Adaptive Control (EMRAC) used for this Project will be presented.

4.1 The EMRAC Algorithm

Model Reference Adaptive Control (MRAC) aims at imposing the dynamic of a reference model to a plant. It has a strong theoretical foundation and has proven to be a worthy solution for controlling system affected by unmodelled dynamics and unknown real time parameters . To further improve the MRAC performance when facing intense nonlinearities and rapid changing disturbances, an adaptive integral action and an adaptive switching term are added to the standard control law like in [8]. The resulting controller is known in literature as the Enhanced MRAC (EMRAC). The EMRAC has been previously used for Single Input Single Output (SISO) systems affected by uncertainties and disturbances showing promising results [8, 10, 11]. Robustness of the EMRAC can also be enhanced by using methods such as, parameter projection and σ-modification, for stopping the unbounded drifting of the adaptive gains [12]. In this thesis, a Multi Input Multi Output (MIMO) extension of the EMRAC (control scheme in Fig 4.1) is adopted for controlling the space robotic manipulator. In order to prevent the drifting of the control gains and guaranteeing ultimate boundedness of the closed loop system, a σ-modification framework is exploited. To the best of the author's knowledge,

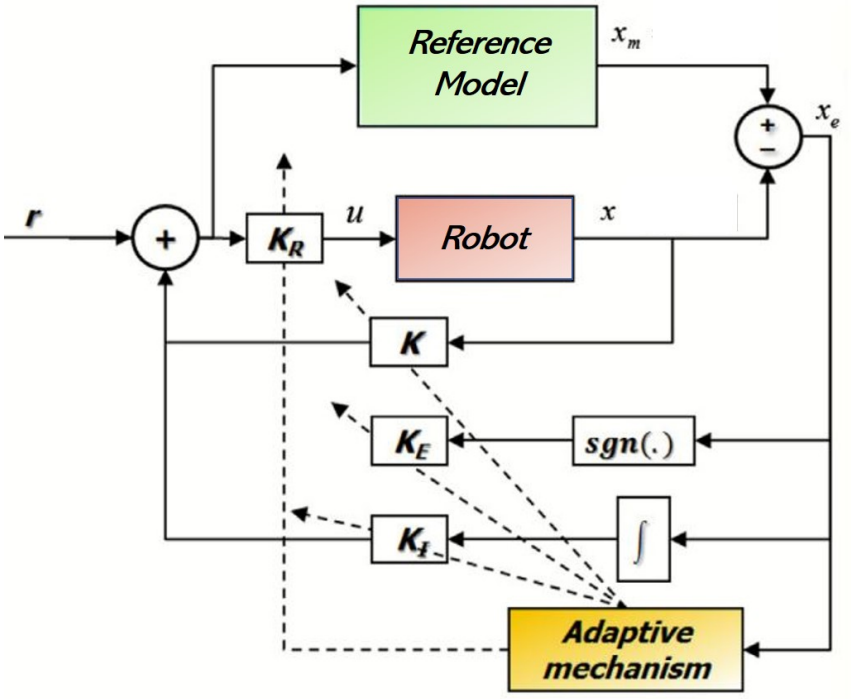


Figure 4.1: EMRAC Control Scheme, modified from [9].

this project application is a novel implementation of the EMRAC controller with σ-modification to a MIMO space robotic system. More specifically, in the following sections, two different extension of the EMRAC algorithm to the MIMO case will be introduced.

4.2 Problem Formulation

The problem formulation presented in this section is a MIMO extension of the EMRAC algorithm proposed in [8], referring to the notation adopted in [12] Considering a plant with n-degrees of freedom of order l in the following form:

$$\dot{x} = Ax + Bu + B_e\delta, \tag{4.1}$$

where $x \in \mathbb{R}^{nl}$ and $u \in \mathbb{R}^n$ are the state vector and the system input respectively, $A \in \mathbb{R}^{nl\times nl}$ and $B \in \mathbb{R}^{nl\times n}$ are the system matrices defined as:

$$\begin{aligned} A &= \begin{bmatrix} 0_{(l-1)n,n} & & I_{(l-1)n} \\ -A_1 & \cdots & -A_l \end{bmatrix}, \\ B_e &= \begin{bmatrix} 0_{(l-1)n,n} \\ I_n \end{bmatrix}, \qquad B = B_e B_c G, \end{aligned} \tag{4.2}$$

with A_i being the matrix coefficients of the dynamic matrix, B_c and G are a positive matrix and a positive diagonal matrix respectively. $t_0 \in \mathbb{R}$ is the initial time instant. $\delta = \delta(t)$ is the scalar disturbance acting on the plant dynamics and belongs either to $L_2 \cap L_\infty$ or $C \cap L_\infty$, where C is the set of continuous scalar functions. While the definite positiveness of the matrices B_c and G is assumed, all the entries of the plant are supposed to be unknown.

The control objective of the EMRAC algorithm consists in imposing the dynamics of a reference model to system 4.1 while keeping all the closed-loop signals bounded. The reference model is a LTI system with asymptotic stability properties

$$\dot{x}_m = A_m x_m + B_m r \tag{4.3}$$

$x_m \in \mathbb{R}^{nl}$ is the reference model state. The reference input $r \in \mathbb{R}^n$ is bounded and the matrices $A_m \in \mathbb{R}^{nl \times nl}$ and $B_m \in \mathbb{R}^{nl \times n}$ are of the same form of the plant matrices

$$A_m = \begin{bmatrix} 0_{(l-1)n,n} & & I_{(l-1)n} \\ -A_{m1} & \cdots & -A_{ml} \end{bmatrix}, \quad B_m = B_e B_{mc}, \tag{4.4}$$

$B_{mc} \in \mathbb{R}^{n \times n}$ is a given positive matrix and $A_{mi} \in \mathbb{R}^{n \times n}$ are the matrix coefficients giving A_m Hurwitz.
The control action of the EMRAC is given by

$$u(t) = u_{MCS}(t) + u_I(t) + u_N(t), \tag{4.5}$$

with

$$u_{MCS}(t) = K_X(t)x(t) + K_R(t)r(t), \tag{4.6a}$$

$$u_I(t) = K_I(t)x_I(t), \tag{4.6b}$$

where $\dot{x}_I = x_e + f_e$, giving x_i as the integral of the racking error: $x_e = x_m - x$. The adaptive gains are composed by a proportional part and an integral adaptive contribution and are computed as

$$K_X = \phi_X^T + y_e x^T \beta_X \quad \textit{and} \quad \dot{\phi}_X^T = y_e x^T \alpha_X + f_X^T, \tag{4.7a}$$

$$K_R = \phi_R^T + y_e r^T \beta_R \quad \textit{and} \quad \dot{\phi}_R^T = y_e r^T \alpha_R + f_R^T, \tag{4.7b}$$

$$K_I = \phi_I^T + y_e x_I^T \beta_I \quad \textit{and} \quad \dot{\phi}_I^T = y_e x_I^T \alpha_I + f_I^T, \tag{4.7c}$$

$\alpha_X, \alpha_R, \alpha_I, \beta_X, \beta_R$ and β_I are strictly positive constant diagonal matrices that determine the speed of the adaptation, while $y_e = C_e x_e$, with $C_e = B_e^T P$, and $PA_m + A_m^T P = -Q$, $Q = Q^T > 0$.
where $Q \in \mathbb{R}^{nl \times nl}, P \in \mathbb{R}^{nl \times nl}, B_e^T \in \mathbb{R}^{n \times nl}, C_e^T \in \mathbb{R}^{nl \times nl}$, and $y_e \in \mathbb{R}^n$. y_e represent the projection of the tracking error and has to be properly tuned for guaranteeing closed loop stability of the system
Having assumed A_m to be Hurwitz, it exists a solution of the Lyapunov Equation P. The functions f_e, f_X, f_R and f_I are used in the σ-modification strategy for preventing the drifting of the adaptive gains in presence of disturbances.

Having extent the EMRAC controller to a multi input multi output system, two different approaches for defining the adaptive switching term are used, a first one where the components of y_e are considered separately leading to an element-wise tuning of the switching action, and a second one where the tuning is performed directly on the unit vector of y_e. From now on, these approaches will be referred as EMRAC-EW and EMRAC-UV respectively.

4.2.1 EMRAC-EW

For this control strategy, the switching term of control action is defined as

$$u_N(t) = K_N(t)\Psi(y_e(t)), \tag{4.8}$$

where $\Psi(\cdot)$ is the function defined as

$$\Psi(y_e) = [\text{sgn}(y_{e,1}), \text{sgn}(y_{e,2}), ..., \text{sgn}(y_{e,n})]. \tag{4.9}$$

The related adaptive gain is computed as

$$K_N = \phi_N \quad and \quad \dot{\phi}_N = \gamma diag(\bar{y}_e) + f_N, \tag{4.10}$$

with $\bar{y}_e = [|y_{e,1}|, |y_{e,2}|, ..., |y_{e,n}|]$, $\gamma \in \mathbb{R}^{n \times n}$ is a constant positive diagonal matrix and f_N is the σ-modification term related to the switching action

4.2.2 EMRAC-UV

For this control strategy, the switching term of control action is defined as

$$u_N(t) = K_N(t)\frac{y_e(t)}{\|y_e(t)\|}, \tag{4.11}$$

The related adaptive gain is computed as

$$K_N = \phi_N \quad and \quad \dot{\phi}_N = \gamma \|y_e\|_H + f_N \tag{4.12}$$

with $\|y_e\|_H = \sqrt{h_1 y_{e,1}^2 + h_1 y_{e,2}^2 + h_1 y_{e,3}^2}$ being the weighted norm of y_e with the generic weights h_i, $\gamma \in \mathbb{R}^{n\times n}$ is a constant positive scalar and f_N is the σ-modification term related to the switching action

4.2.3 σ-Modification

For both of the control architecture, a σ-modification strategy is used to prevent the drifting of the adaptive gains.
Three diagonal matrices are defined for collecting the adaptive gains

$$\Gamma_\alpha = diag(\hat{\alpha}_1, \hat{\alpha}_2, ..., \hat{\alpha}_{2nl+n}), \tag{4.13a}$$

$$\Gamma_\beta = diag(\hat{\beta}_1, \hat{\beta}_2, ..., \hat{\beta}_{2nl+n}), \tag{4.13b}$$

$$\Gamma_\gamma = diag(\hat{\gamma}_1, \hat{\gamma}_2, ..., \hat{\gamma}_n). \tag{4.13c}$$

with $(\hat{\alpha}_i, \hat{\beta}_i) = (\alpha_X, \beta_X)$, $i = 1, ..., nl$, $(\hat{\alpha}_i, \hat{\beta}_i) = (\alpha_R, \beta_R)$, $i = 1, ..., n$, $(\hat{\alpha}_i, \hat{\beta}_i) = (\alpha_I, \beta_I)$, $i = 1, ..., nl$, $(\hat{\gamma}_i) = (\gamma_i)$, $i = 1, ..., n$. Also, the integral adaptive part of the adaptive gains is stacked in the vector ϕ

$$\begin{aligned} \phi^T &= \begin{bmatrix} \phi_1 & \phi_2 & \dots & \phi_{2nl+n} \end{bmatrix} = \\ &= \begin{bmatrix} \phi_X^T & \vdots & \phi_R^T & \vdots & \phi_I^T \end{bmatrix} \end{aligned} \tag{4.14}$$

and the elements f_X,f_R and f_I are collected in the vector f

$$\begin{aligned} f^T &= \begin{bmatrix} f_1 & f_2 & \dots & f_{2nl+n} \end{bmatrix} = \\ &= \begin{bmatrix} f_X^T & \vdots & f_R^T & \vdots & f_I^T \end{bmatrix}. \end{aligned} \tag{4.15}$$

with the σ-modification strategy, the adaptive gains of the EMRAC are computed defining f as

$$f_X^T = -\rho_X \sigma_\phi(||\phi||)\phi_X^T, \tag{4.16a}$$

$$f_R^T = -\rho_R \sigma_\phi(||\phi||)\phi_R^T, \tag{4.16b}$$

$$f_I^T = -\rho_I \sigma_\phi(||\phi||)\phi_I^T. \tag{4.16c}$$

with ρ_X,ρ_R and ρ_I positive constants. The σ-function is

$$\sigma_\phi(||\phi||) = \begin{cases} 0 & if ||\phi|| \leq \hat{\mathrm{M}}_\phi, \\ \eta_\phi(\frac{||\phi||}{\hat{\mathrm{M}}_\phi} - 1) & if \hat{\mathrm{M}}_\phi < ||\phi|| \leq 2\hat{\mathrm{M}}_\phi, \\ \eta_\phi & if ||\phi|| > 2\hat{\mathrm{M}}_\phi, \end{cases} \tag{4.17}$$

where

$$\hat{\mathrm{M}}_{\phi} = \sqrt{\frac{\lambda_{\max}(\Gamma_{\alpha}^{-1}\Gamma_{\rho})}{\lambda_{\min}(\Gamma_{\alpha}^{-1}\Gamma_{\rho})}\mathrm{M}} \tag{4.18}$$

with $\Gamma_{\rho} = diag(\hat{\rho}_1, \hat{\rho}_2, ..., \hat{\rho}_{2nl+n})$, $\hat{\rho}_i = \rho_X$, $i = 1, ..., nl$, $\hat{\rho}_i = \rho_R$, $i = 1, ..., n$, $\hat{\rho}_i = \rho_I$, $i = 1, ..., nl$ $\lambda_{\min}(\mathrm{H})$ and $\lambda_{\max}(\mathrm{H})$ represent the minimum and maximum eigenvalue of H matrix respectively.

Finally, η_{ϕ} is a positive constant that follow the relation

$$b\eta_{\phi}\lambda_{\min}(\Gamma_{\alpha}^{-1}\Gamma_{\rho}) > \frac{3}{4}\lambda_{\min}(Q) \tag{4.19}$$

σ_{ϕ} is a continuous, non-negative function that activates whenever $||\phi||$ exceeds a pre-set threshold.

An analogue approach is used for defining the gain K_N and x_I with the σ-modification (only the main equation are shown).

$$f_N = -\rho_N \sigma_{\phi_N}(||\phi_N||)\phi_N \tag{4.20}$$

For EMRAC-EW

$$\sigma_{\phi_{N_i}}(||\phi_{N_i}||) = \begin{cases} 0 & if||\phi_{N_i}|| \leq \hat{\mathrm{M}}_{\phi_{N_i}}, \\ \eta_{\phi_{N_i}}(\frac{||\phi_{N_i}||}{\hat{\mathrm{M}}_{\phi_{N_i}}} - 1) & if\hat{\mathrm{M}}_{\phi_{N_i}} < ||\phi_{N_i}|| \leq 2\hat{\mathrm{M}}_{\phi_{N_i}}, \\ \eta_{\phi_{N_i}} & if||\phi_{N_i}|| > 2\hat{\mathrm{M}}_{\phi_{N_i}}, \end{cases} \tag{4.21}$$

For EMRAC-UV

$$\sigma_{\phi_N}(||\phi_N||) = \begin{cases} 0 & if||\phi_N|| \leq \hat{\mathrm{M}}_{\phi_N}, \\ \eta_{\phi_N}(\frac{||\phi_N||}{\hat{\mathrm{M}}_{\phi_N}} - 1) & if\hat{\mathrm{M}}_{\phi_N} < ||\phi_N|| \leq 2\hat{\mathrm{M}}_{\phi_N}, \\ \eta_{\phi_N} & if||\phi_N|| > 2\hat{\mathrm{M}}_{\phi_N}, \end{cases} \tag{4.22}$$

$$\hat{\mathrm{M}}_{\phi_N} = \sqrt{\frac{\lambda_{\max}(\Gamma_{\gamma}^{-1}\Gamma_{\rho_N})}{\lambda_{\min}(\Gamma_{\gamma}^{-1}\Gamma_{\rho_N})}\mathrm{M}} \tag{4.23}$$

$$f_e^T = -\rho_e \sigma_I(||x_I||)x_I^T \tag{4.24}$$

$$\sigma_I(||x_I||) = \begin{cases} 0 & if||x_I|| \leq \hat{\mathrm{M}}_I, \\ \eta_I(\frac{||x_I||}{\hat{\mathrm{M}}_I} - 1) & if\hat{\mathrm{M}}_I < ||x_I|| \leq 2\hat{\mathrm{M}}_I, \\ \eta_I & if||x_I|| > 2\hat{\mathrm{M}}_I, \end{cases} \tag{4.25}$$

5

Application of the EMRAC to Space Manipulators: Four Approaches

In this chapter the problem of motion control of the manipulator is discussed. The control of a robotic arm is usually achieved by controlling the actuators generalised forces i.e the control action is performed in the joint space. As a consequence many control approaches can be utilised for controlling a manipulator, in particular, two main categories can be distinguished, joint space control and operational space control. When the position of the end effector is known in the operational space (for example with optical sensor), the operational space control is a better choice, even though, it requires higher complexity due to the fact that the inverse kinematic has to be embedded in the control algorithm [27]. Since this project involves a space robotic arm, the end effector operational position usually cannot be exactly determined online due to the lack of environment references in space. Therefore, a joint space control algorithm is adopted and the trajectory in the operational space is converted into the joint space trajectory offline through the inverse kinematic algorithm.

5.1 Inverse Dynamics

As already discussed in in chapter 3, a robotic manipulator is a highly nonlinear system and the dynamic model is described by equation 3.16. For simplicity

all the terms apart from $B(q)\ddot{q}$ are collected into $n(q, \dot{q})$ giving

$$B(q)\ddot{q} + n(q, \dot{q}) = u \tag{5.1}$$

Exploiting the nonlinear state feedback, it is possible to perform an exact linearization of the system dynamics. The possibility is granted by the fact that $B(q)\ddot{q}$ is invertible, as discussed in 3.
By defining the inverse dynamics control action u as

$$u = B(q)y + n(q, \dot{q}) \tag{5.2}$$

leading to

$$\ddot{q} = y \tag{5.3}$$

the resulting system (block scheme in Fig. 5.1) is linear and decoupled, meaning that the generic component of the new input y_i influences only the generic joint variable q_i, without being affected by induced dynamics of other joints [27]. Furthermore, the relationship of the system 5.3 is a double integrator, meaning that the linear system is unstable. Therefore, an additional control action has to be employed to control the decoupled robotic manipulator.

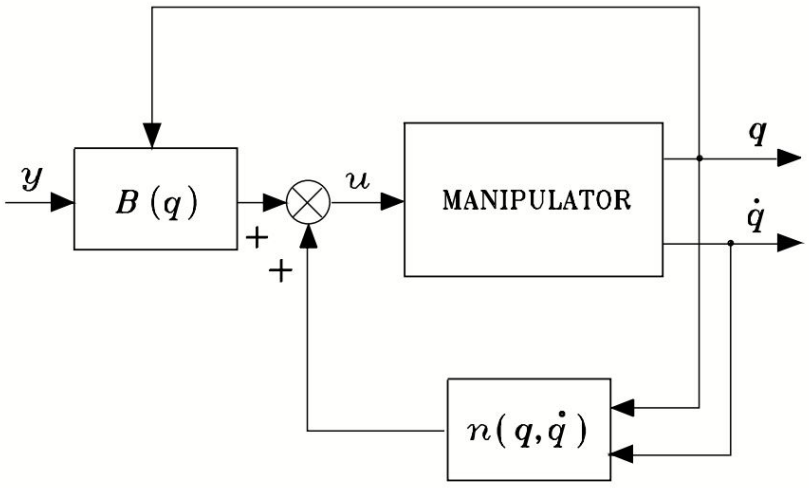


Figure 5.1: Inverse Dynamics Control Scheme [27].

5.2 EMRAC Controllers

Even if the EMRAC controller is able to compensate for system nonlinearities, having a feedback control law, could ease the control effort and allow for a

smaller tracking error. For this reason, for the application of the EMRAC-EW and the EMRAC-UV, two different control architectures have been tested. A first one, where the EMRAC control action is augmented through the feedback linearization of the nonlinear terms of the robot, as shown in Fig 5.2. A second approach, where the EMRAC is applied directly on the system with no feedback linearization illustrated in Fig 5.3. These two control architectures will be denoted with the extension '-FL' and '-NFL' respectively.

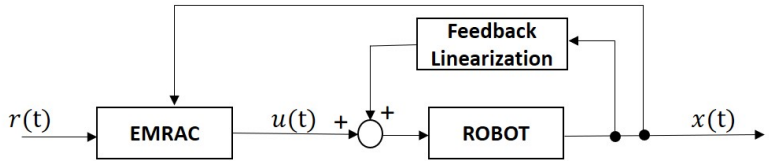


Figure 5.2: EMRAC Controller with Feedback Linearization.

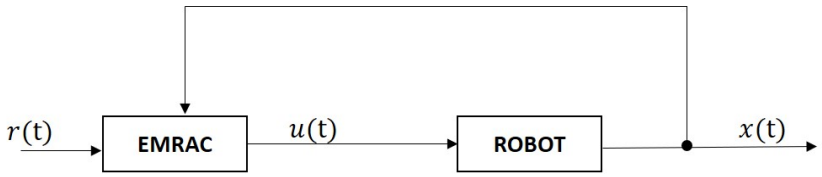


Figure 5.3: EMRAC Controller with No Feedback Linearization.

This brings the total number of EMRAC control strategies to be tested to 4:

- EMRAC-EW-NFL
- EMRAC-EW-FL
- EMRAC-UV-NFL
- EMRAC-UV-FL

6

Simulations & Analysis

Showing the real time feasibility of a control algorithm helps to validate its potential for real-life applications. This chapter reports the simulation data of the space robotic manipulator controlled by the EMRAC controllers, presented in chapter 5, preforming a desired trajectory in a real time environment (see chapter 2). While tracking the trajectory, the robotic arm will hold an end effector mass of $100Kg$ affected by a disturbance force (see chapter 3). This simulation tries to replicate a real-like scenario where a space robotic manipulator is acting on a non-cooperative micro-satellite. The simulation will be replicated using well-known control architectures for performance comparison.

6.1 Simulation

6.1.1 Trajectory Generation

The chosen joint space trajectory is obtained through the interpolation of two polynomials of order 5 and the application of a inverse kinematic algorithm, similarly to the approach used in [27]. The resulting trajectory last $250s$ and consist of two laps performed on an inclined circumference preceded by a first stroke to reach the starting position. Within the trajectory, some steady state interval are present. The time-line of the simulation can be resumed as follow:

- $t_0 = 0s$ start of the simulation (first steady state interval)
- $t_1 = 5s$ start of the first segment
- $t_2 = 55s$ end of the first stroke (second steady state interval)
- $t_{c_0} = 60s$ start of the first circumference lap

- $t_{c_1} = 150s$ start of the second circumference lap
- $t_{c_2} = 240s$ end of the second circumference lap (third steady state interval)
- $t_f in = 250s$ end of the simulation

Fig 6.1 and Fig 6.2 show the trajectory in the operational space and in the joint space, respectively.

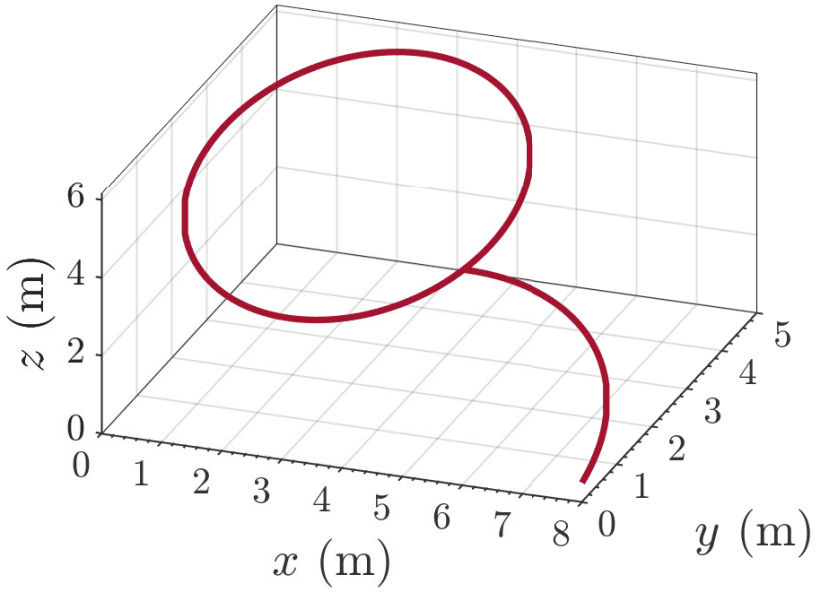
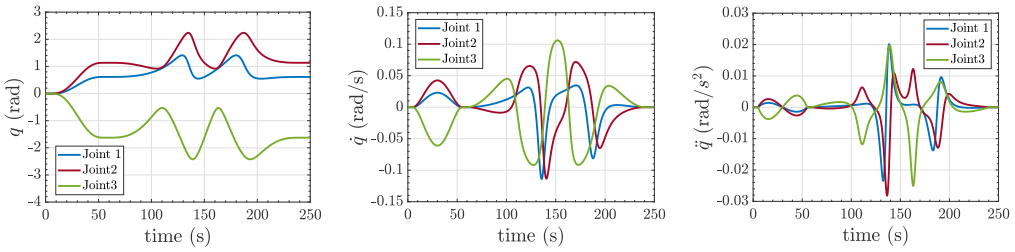


Figure 6.1: Operational Space Trajectory



(a) (b) (c)

Figure 6.2: Joint Space Trajectory

Fig 6.3 shows that $\det J \neq 0$ during the inclined circumference stroke, implying that the trajectory is non-singular within that interval.

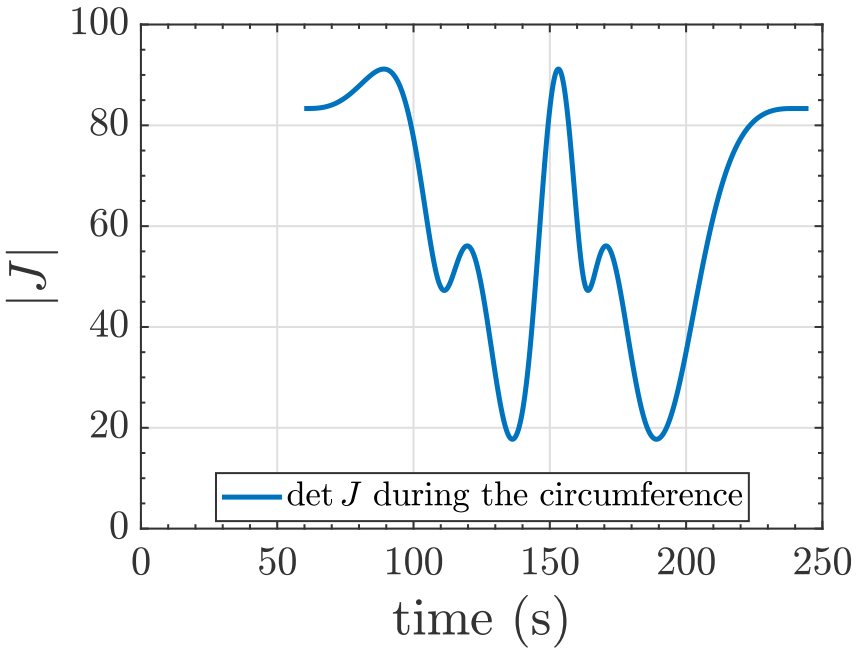


Figure 6.3: Determinant of the Jacobian during the inclined circumference stroke

6.1.2 Benchmarks Controllers

In this section are presented the benchmarks controllers used for comparing the obtained results. Some well known centralized control strategy for robot are discussed in [27]. As a comparison for the EMRAC controller, a proportional derivative (PD) controller with feedback nonlinear compensation and a robust Controller are picked from [27], and also a Proportional Integral (PI) strategy is used.

PD Control

This PD controller consist of a linear controller applied to the linearized system discussed in chapter 5. The feedback control block scheme is illustrated in Fig 6.4

Robust Control

Due to an imperfect modelling of the mathematical model, it is possible to have an imperfect compensation. The robust control of Fig 6.5 solves this problem

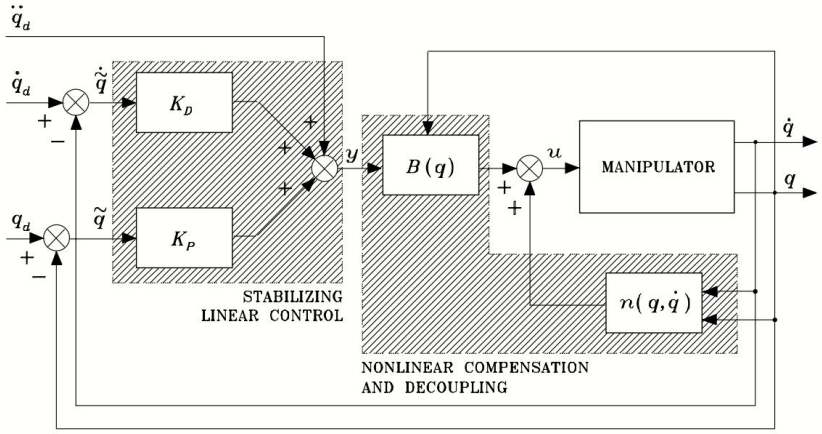


Figure 6.4: PD Controller [27]

by adding a robust contribution to the PD controller for counteracting the uncertainty related to the nonlinear terms computation.

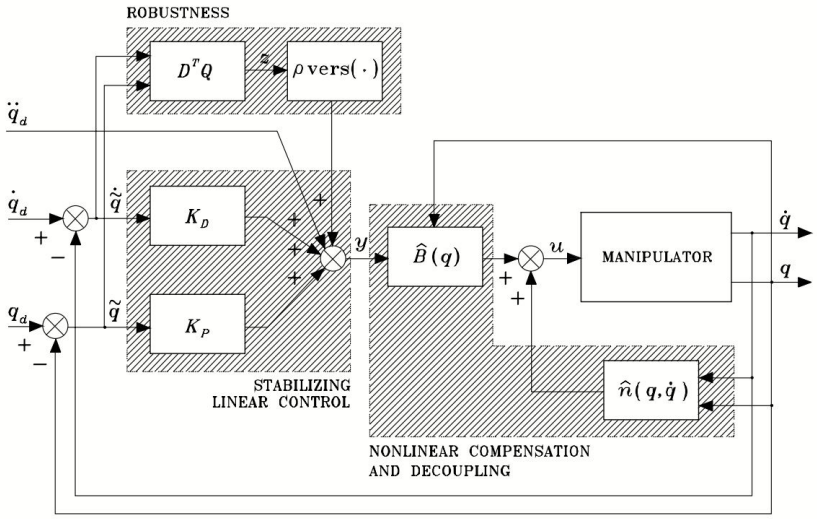


Figure 6.5: PD Robust Controller [27]

PI Control

The PI control scheme adopted is shown in Fig 6.6

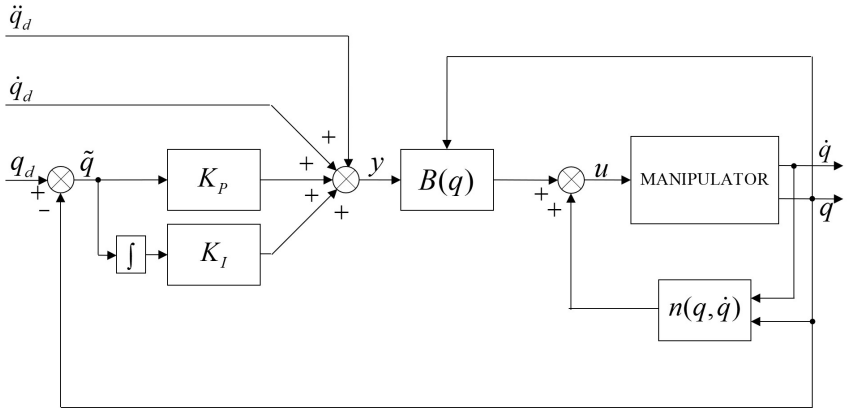


Figure 6.6: PI Controller [27]

6.1.3 Tuning of the EMRAC Controllers

The tuning of the EMRAC controller is performed in MATLAB on the imported URDF model of the robot considering an end effector mass of $10Kg$, and then, the tuned controller is deployed in ROS where the simulation are performed setting the unknown target mass to $100Kg$. The EMRAC controller will run with sampling time at $5ms$ and the control parameters are tuned with a trial and error procedure to find a good trade off between tracking performance and command effort.

In Tab 6.1 the tuned control parameters are reported. Note that, in order to have a fair comparison, same K_P and K_D (6.1) gain matrices were used for all controllers.

$$\begin{aligned} K_P &= 10^{-3} \begin{bmatrix} 6.7 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \\ K_D &= 10^{-2} \begin{bmatrix} 16.67 & 0 & 0 \\ 0 & 9 & 0 \\ 0 & 0 & 6.19 \end{bmatrix} \end{aligned} \tag{6.1}$$

Considering $n = 3$ and $l = 2$, I_{nl} and I_n are defined as a (6×6) and a (3×3) identity matrix, respectively. B_e and P are computed as discussed in chapter 4.

Since the simulations in Simulink are not in real time, the performance will be different from the simulations in Gazebo. For compensating this discrepancy, a more conservative tuning of the controller has to be performed, otherwise the real time simulation might require too much computational power and crash. This behaviour is not predictable because it varies on the computer used for

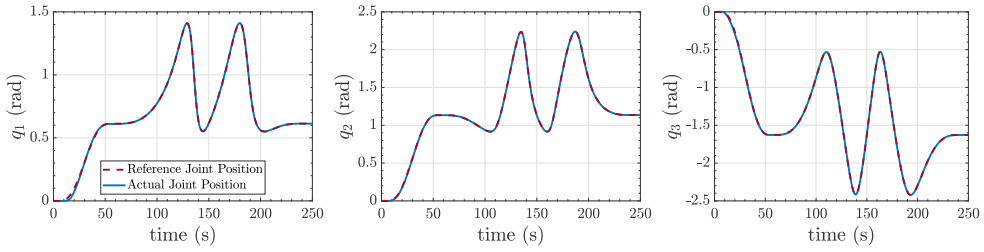
Table 6.1: Tuned Control Parameters

	EMRAC-EW-NFL	EMRAC-EW-FL	EMRAC-UV-NFL	EMRAC-UV-FL
α_X	$10^{-2} I_{nl}$	$10^{-4} I_{nl}$	$10^{-2} I_{nl}$	$10^{-4} I_{nl}$
α_R	I_n	$10^{-4} I_n$	I_n	$10^{-4} I_n$
α_I	$10^{-3} I_{nl}$	$10^{-4} I_{nl}$	$10^{-3} I_{nl}$	$10^{-4} I_{nl}$
β_X	$\alpha_X/20$	$\alpha_X/20$	$\alpha_X/20$	$\alpha_X/20$
β_R	$\alpha_R/20$	$\alpha_R/20$	$\alpha_R/20$	$\alpha_R/20$
β_I	$\alpha_I/20$	$\alpha_I/20$	$\alpha_I/20$	$\alpha_I/20$
γ	$\begin{bmatrix} 0.2 & 0 & 0 \\ 0 & 0.2 & 0 \\ 0 & 0 & 0.05 \end{bmatrix}$	$5 \times 10^{-4} I_n$	10^{-1}	5×10^{-4}
C_e	$10 B_e^T P$	$B_e^T P$	$10 B_e^T P$	$B_e^T P$
ε	$10^3 \begin{bmatrix} 3 \\ 10 \\ 10^3 \end{bmatrix}$	$15 \begin{bmatrix} 3 \\ 10 \\ 10^2 \end{bmatrix}$	$10^3 \begin{bmatrix} 3 \\ 10 \\ 10^3 \end{bmatrix}$	$15 \begin{bmatrix} 3 \\ 10 \\ 10^2 \end{bmatrix}$

simulation and is due to the fact that ROS cannot guarantee a constant publishing rate of $5ms$. Having a variable publishing rate could cause an incorrect behaviour of the discrete time EMRAC controller leading to unbounded gains and a consequent failure of the control system.

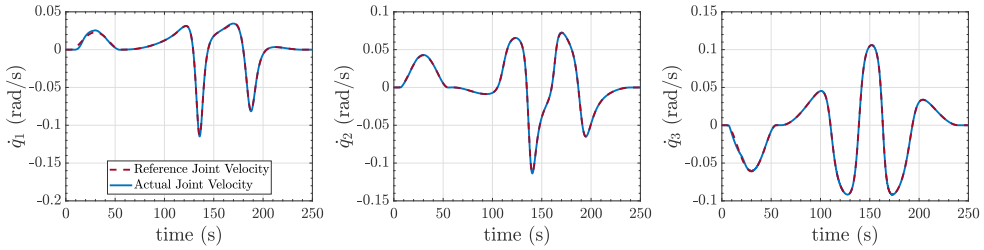
6.2 Simulation Figures

In this section, the figures of the simulations are reported. A more comprehensive illustration is reserved for the EMRAC-EW-NFL while only some of the figures are reported for the other simulations (the remaining ones can be found in appendix C).



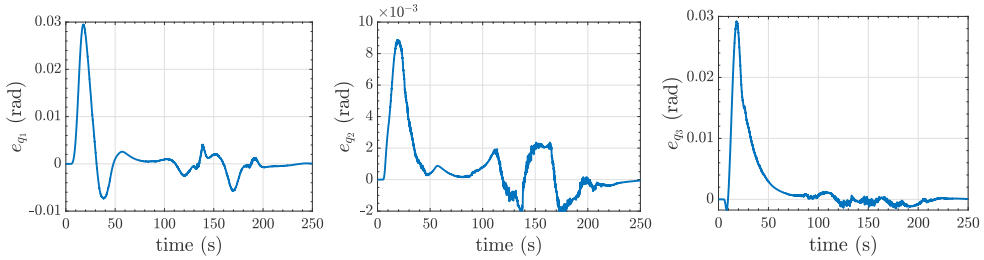
(a) (b) (c)

Figure 6.7: EMRAC-EW-NFL Joint Position



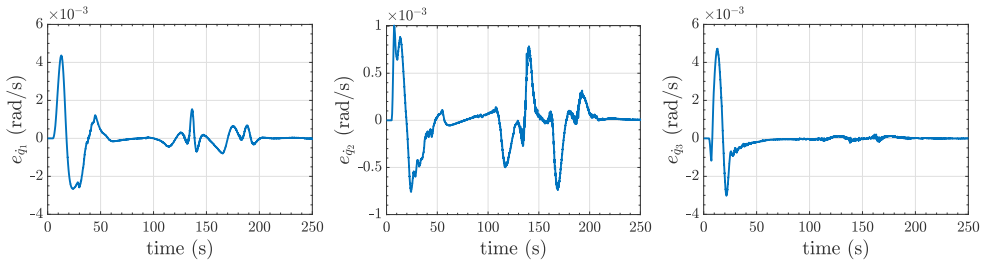
(a) (b) (c)

Figure 6.8: EMRAC-EW-NFL-NOFL Joint Velocity



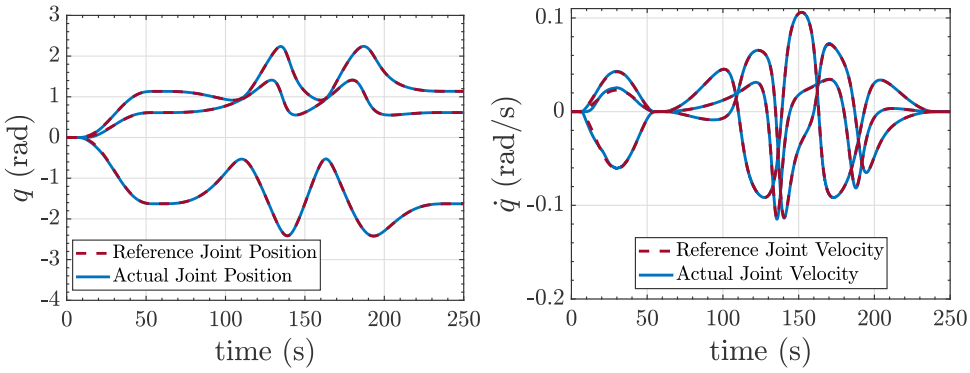
(a) (b) (c)

Figure 6.9: EMRAC-EW-NFL Joint Position Error



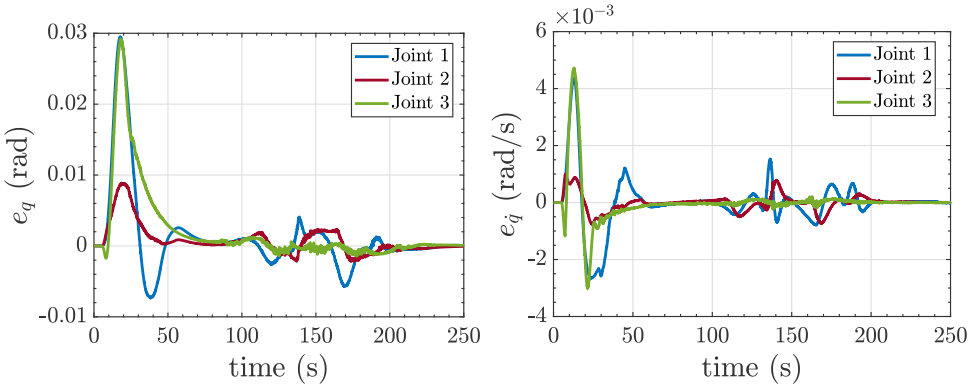
(a) (b) (c)

Figure 6.10: EMRAC-EW-NFL Joint Velocity Error



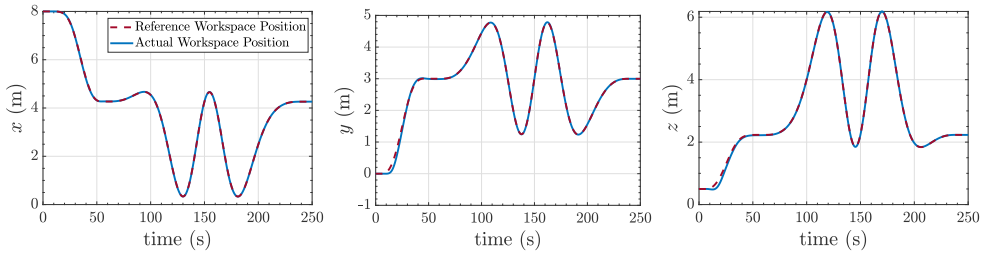
(a) (b)

Figure 6.11: EMRAC-EW-NFL Joint Position and Velocity



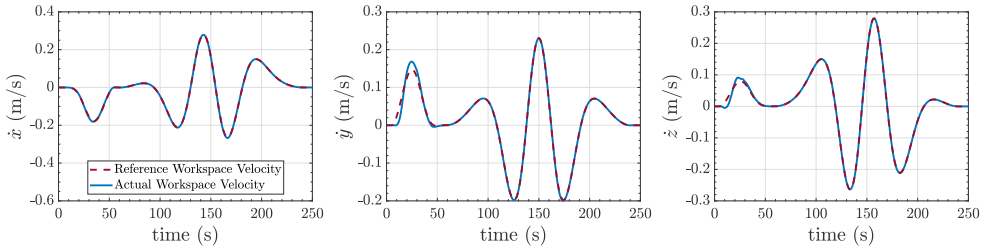
(a) (b)

Figure 6.12: EMRAC-EW-NFL Joint Position and Velocity Error



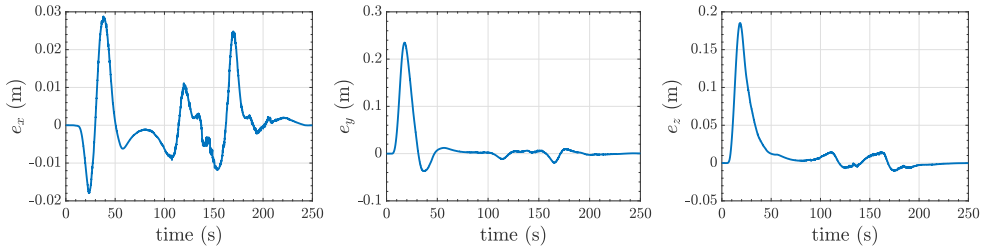
(a) (b) (c)

Figure 6.13: EMRAC-EW-NFL Operational Space Position



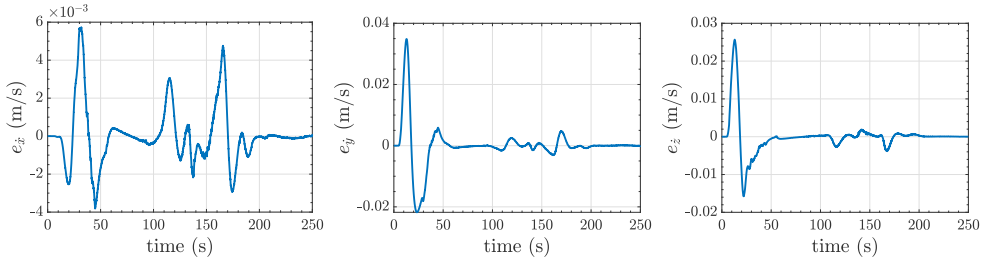
(a) (b) (c)

Figure 6.14: EMRAC-EW-NFL Operational Space Velocity



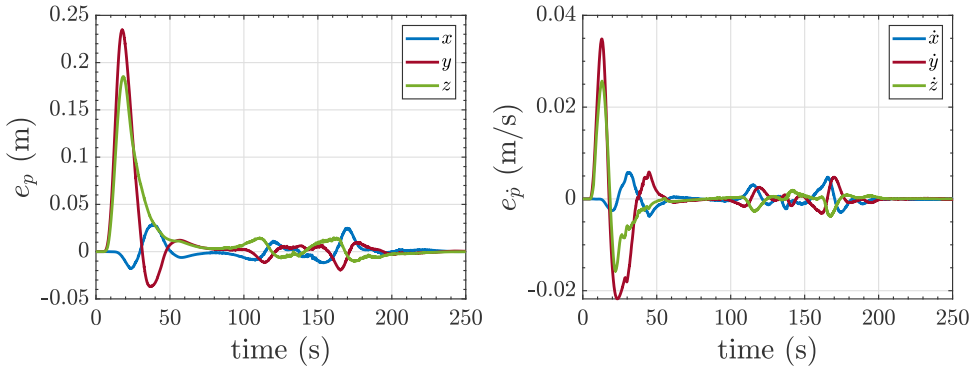
(a) (b) (c)

Figure 6.15: EMRAC-EW-NFL Operational Space Position Error



(a) (b) (c)

Figure 6.16: EMRAC-EW-NFL Operational Space Velocity Error



(a) (b)

Figure 6.17: EMRAC-EW-NFL Operational Space Position and Velocity Error

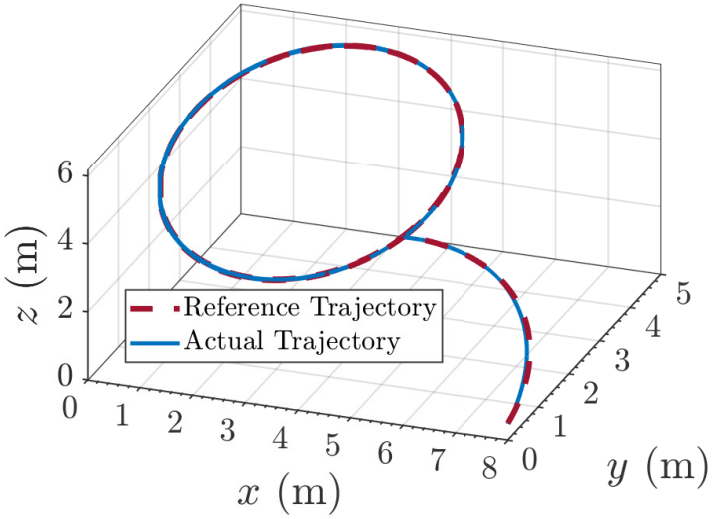
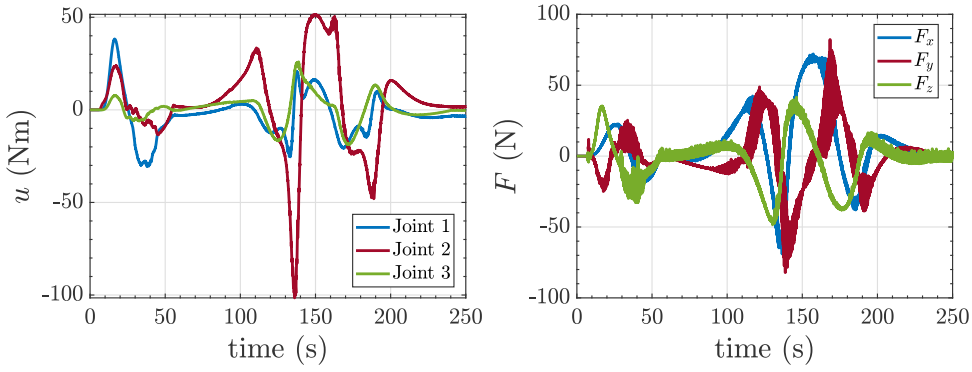
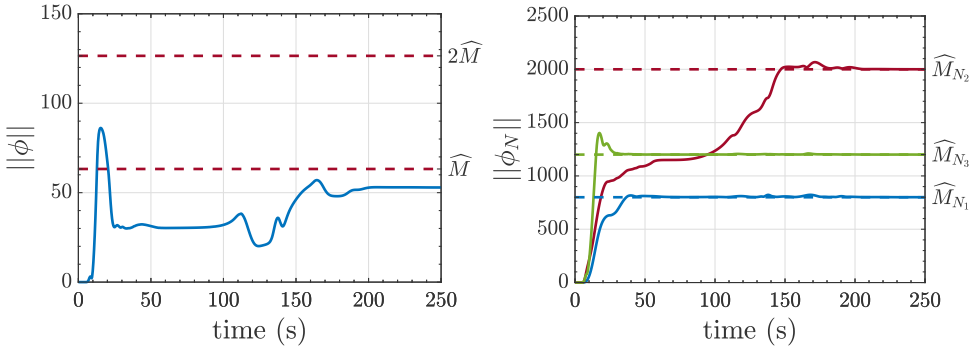


Figure 6.18: EMRAC-EW-NFL Operational Space Trajectory



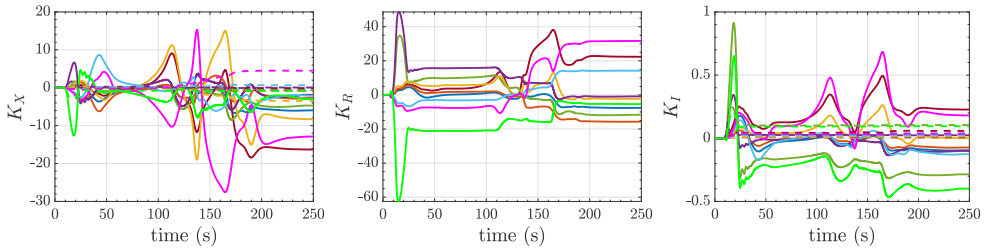
(a) (b)

Figure 6.19: EMRAC-EW-NFL Control Actions and Disturbance Force



(a) (b)

Figure 6.20: EMRAC-EW-NFL $||\phi||$ and $||\phi_N||$



(a) (b) (c)

Figure 6.21: EMRAC-EW-NFL Control Gains

From Fig 6.12 and Fig 6.17 it can be noticed how even a relatively small error in the joint space results in a results in a relatively large error in the operational space. For this reason a precise joint tracking performance (i.e Fig 6.11) are needed to ensure an acceptable tracking of the trajectory in the operational space (Fig 6.18).
Considering the gain evolution, the boundedness of the gains can be verified from Fig 6.21 and also from the boundedness of $||\phi||$ and $||\phi_N||$ (Fig 6.20) aided by the σ-modification strategy.

By comparing Fig 6.19a , Fig 6.22b , Fig 6.23b and Fig 6.24b , it noticeable how the joint command action presents slight signs of chattering in the case of EMRAC with feedback compensation. This slightly difference does not seems to impact the overall good performances of all four EMRAC controllers.

From Fig 6.25 it is evident that the PD controller is not able to perform the complete trajectory. After $\sim$ 102s the control action diverges and the following data are to be considered meaningless. The divergence occur during the inclined circumference stroke so, due to the considerations from Fig 6.3, it is not caused by the presence of singularity.

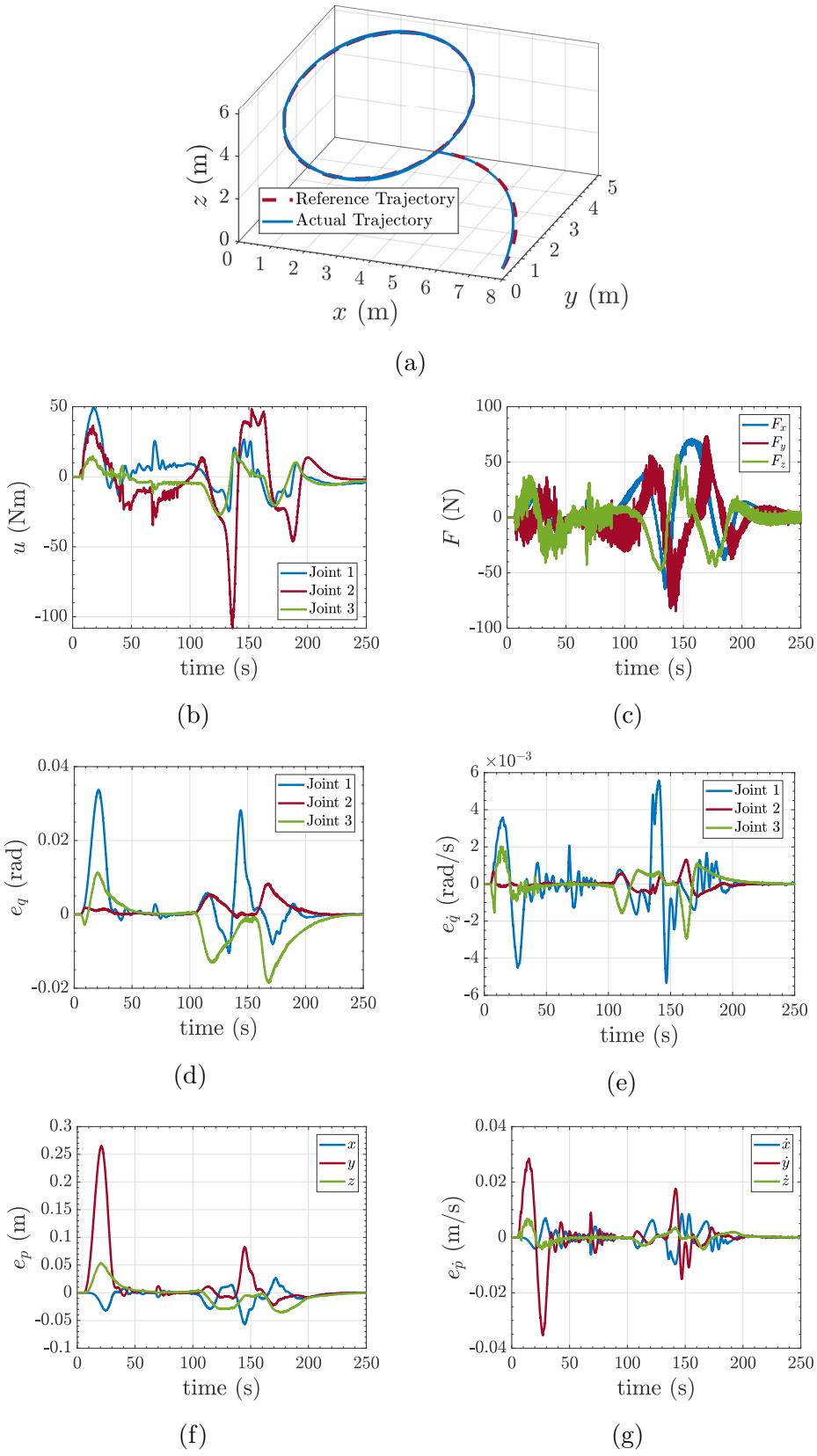


Figure 6.22: EMRAC-EW-FL

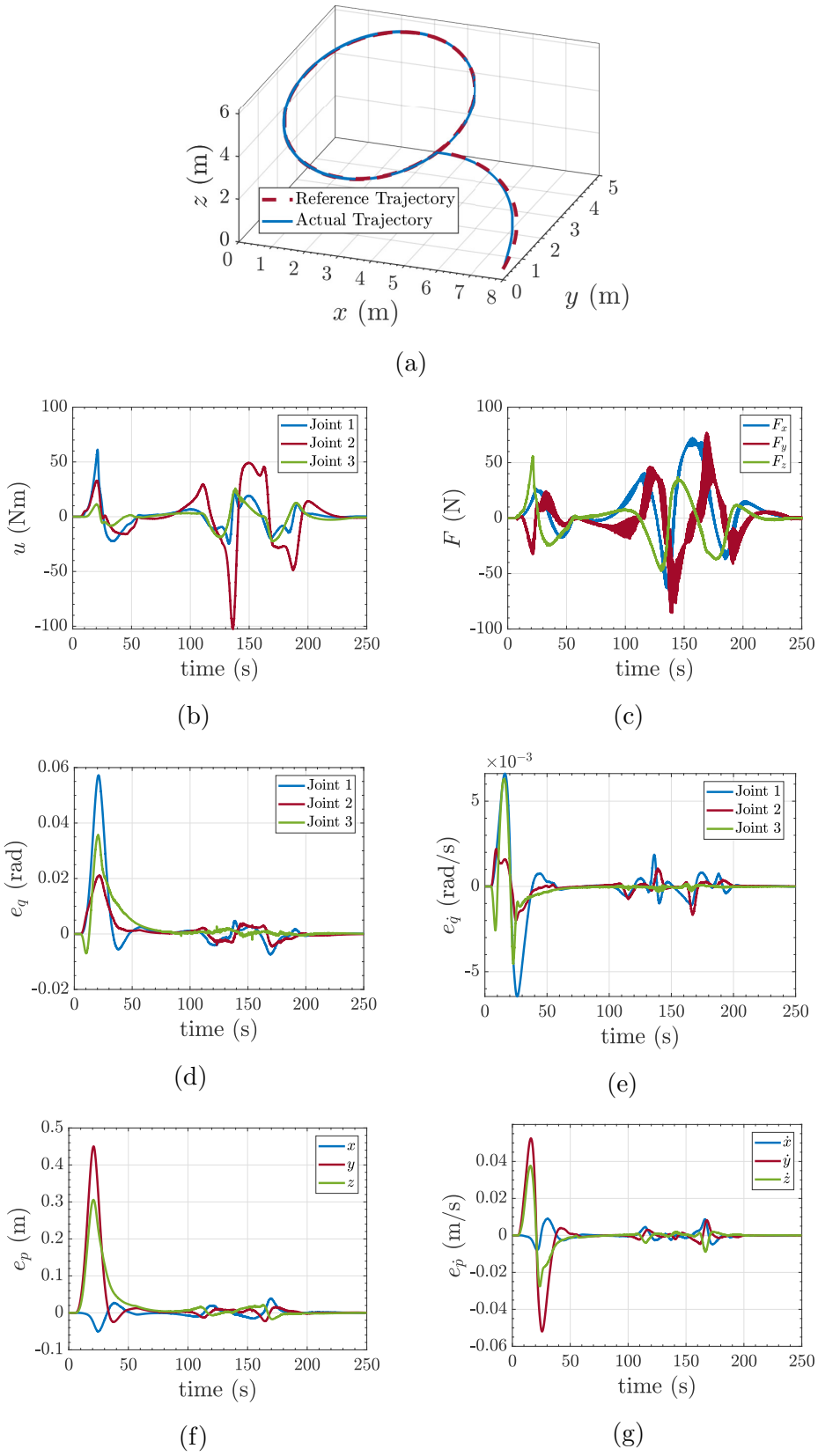


Figure 6.23: EMRAC-UV-NFL

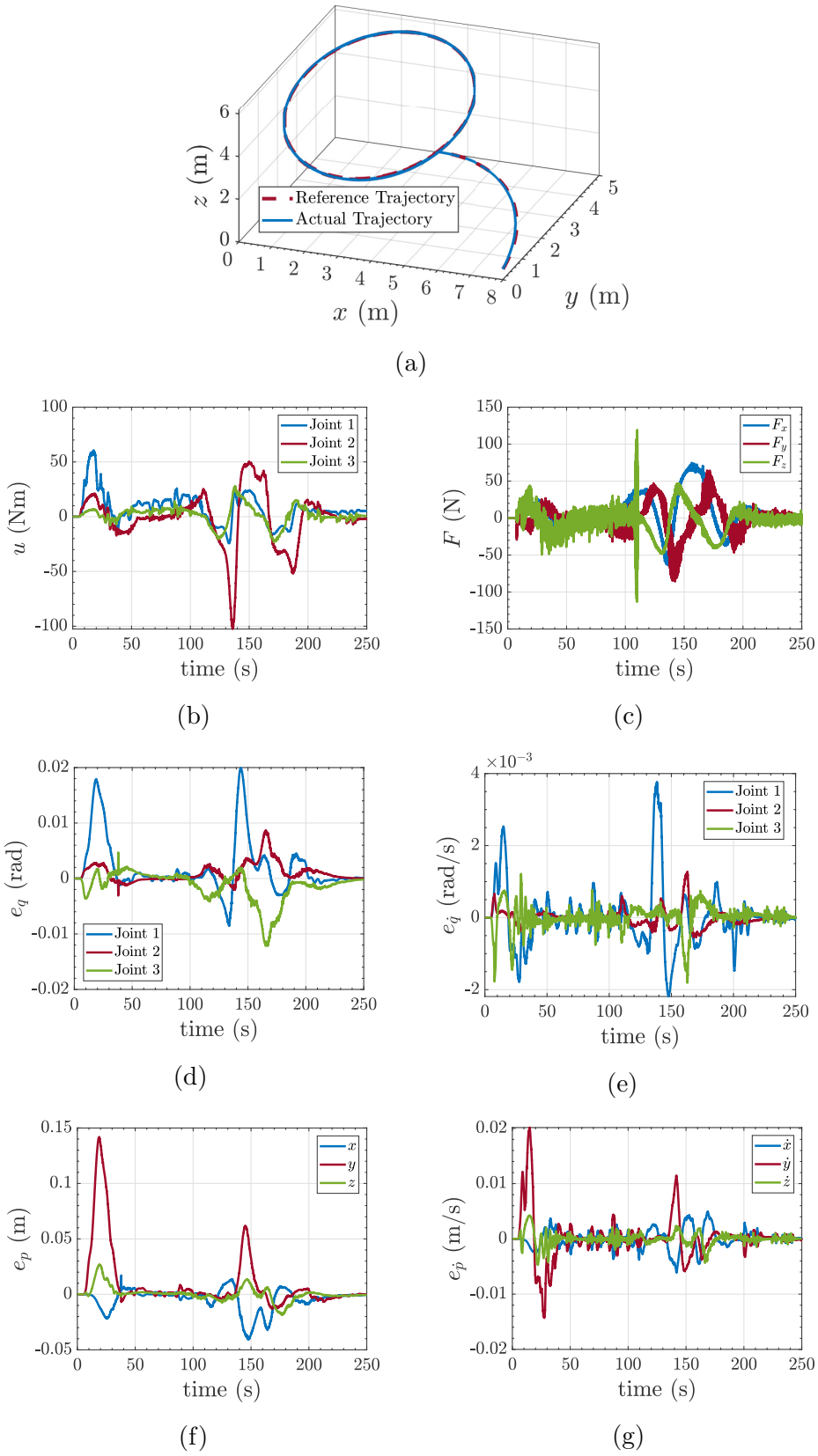


Figure 6.24: EMRAC-UV-FL

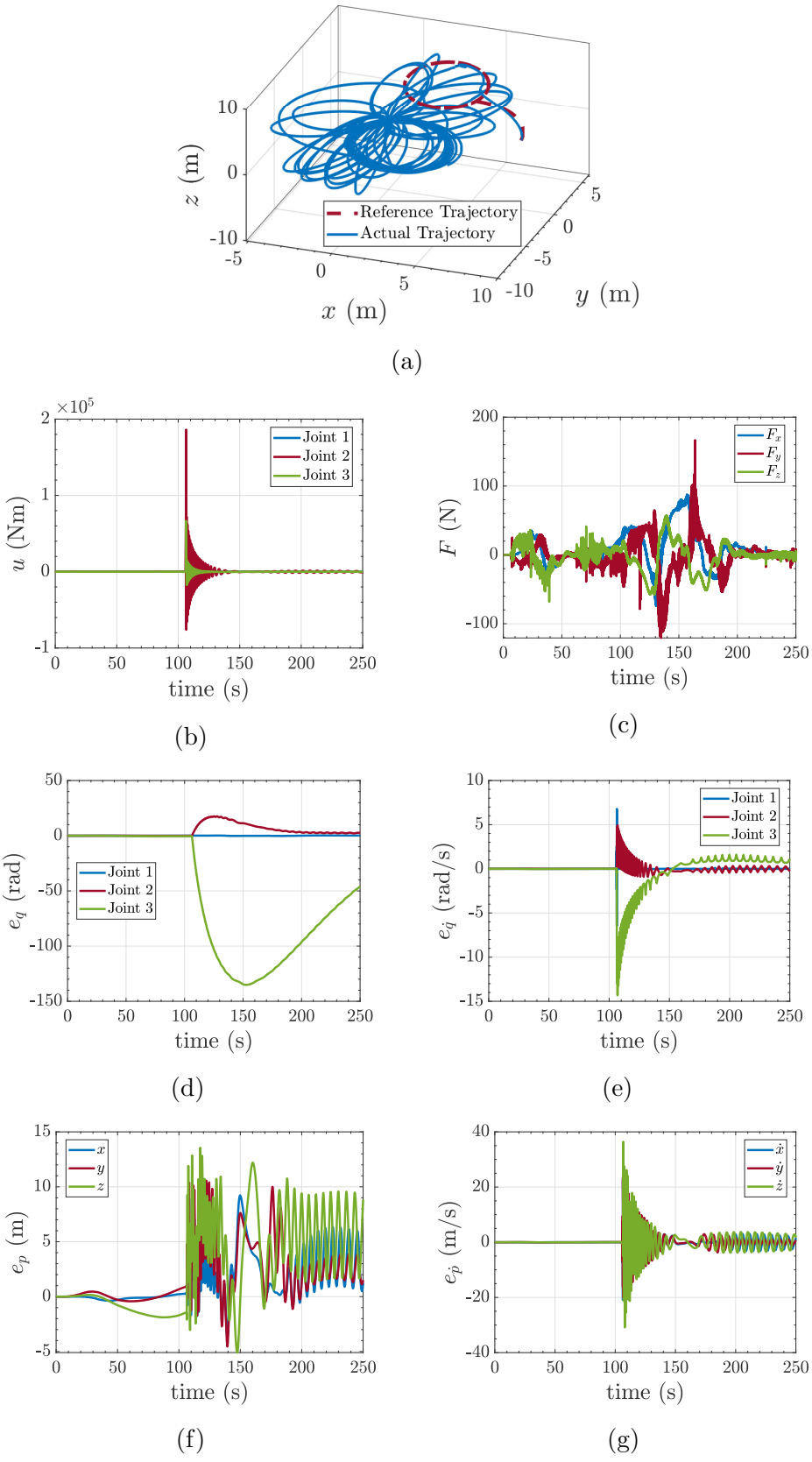


Figure 6.25: PD

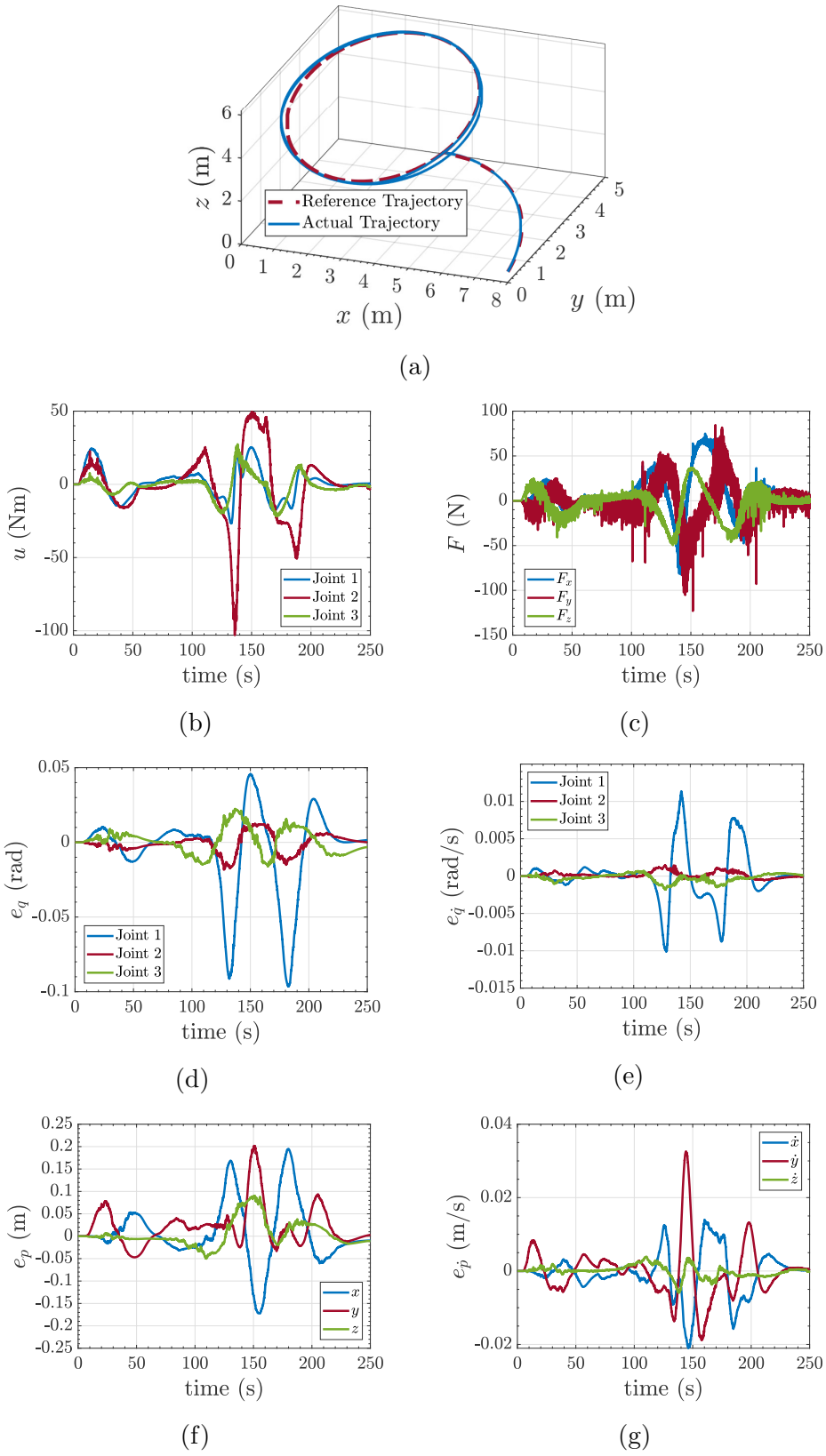


Figure 6.26: ROBUST

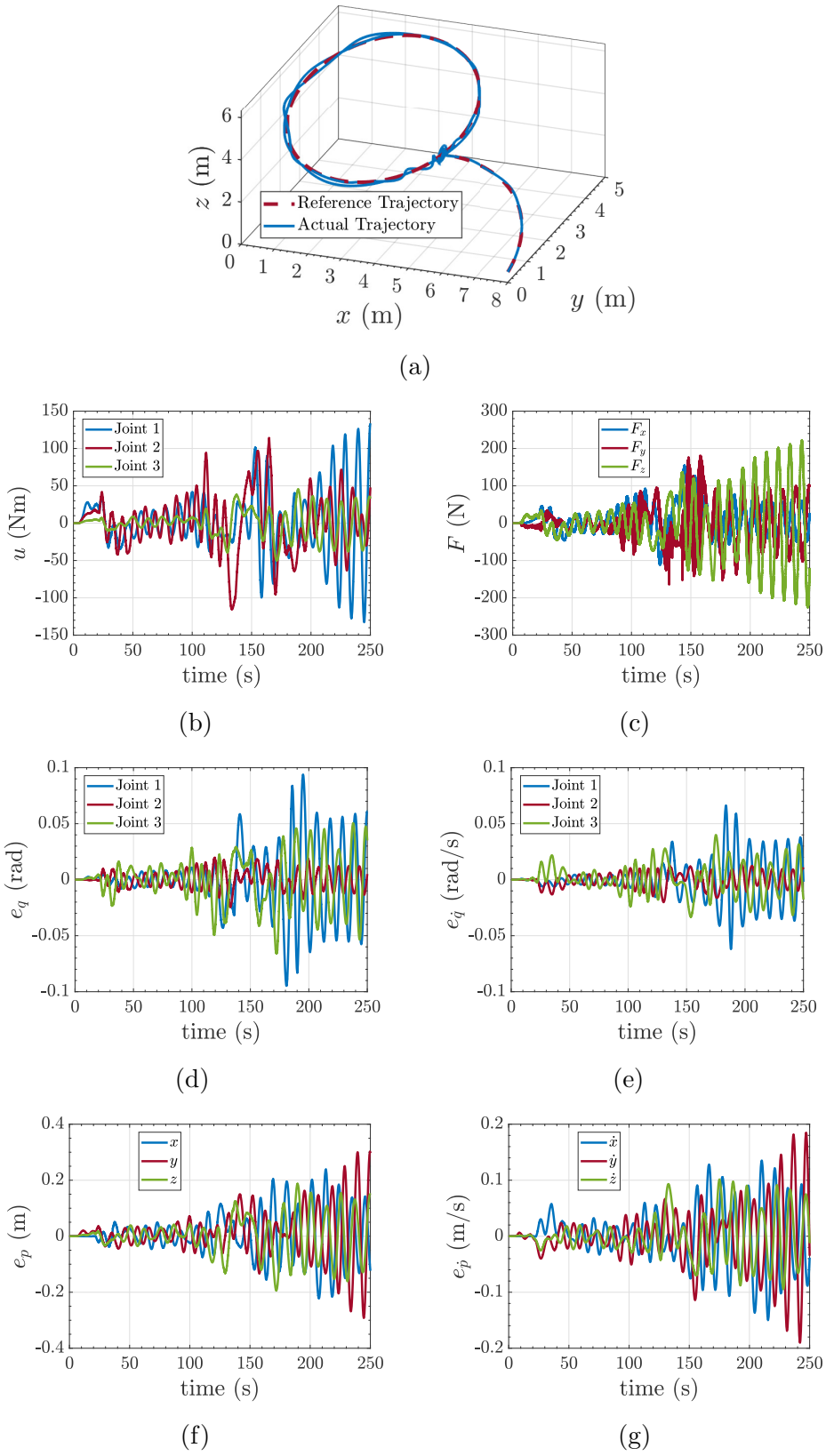


Figure 6.27: PI

Finally, from all the joint error figures above, we can notice that the EMRAC controllers tend to have a slightly worse behaviour in the first $100s$ of the trajectory with respect to the robust and PI controllers. However, for the rest of the trajectory the EMRACs outperform the benchmarks controllers. Is good to notice that the disturbances in Fig 6.19b , Fig 6.22c , Fig 6.23c , Fig 6.24c , Fig 6.25c , Fig 6.26c and Fig 6.27c are not the same signal due to the fact that it depends on the linear acceleration of the robot's end effector (that might be different in each simulation).

6.3 Data Analysis

Through the simulations' figures, it is noticeable that the EMRAC controller presents good tracking performance, also, it can be seen that the PD controller without the robust contribution crashes in the middle of the simulation due to a divergence of the control gains. A more in-depth analysis can be performed by examining the data collected from the simulations. In this view, three performance indexes are exploited and presented in the following section.

6.3.1 Performance Indexes

Two of the performance indexes adopted for data analysis have already been introduced in chapter 3 and allow for a study on the tracking error, while the third one is used to measure the control effort [38]. These performance indexes were computed for different time intervals using the time-line presented in section 6.1.1 and are defined as:

- The maximum absolute value of error (ME)

$$ME = \max\left(|r_{ref}(t) - r(t)|\right) \tag{6.2}$$

  where $r_{ref}(t)$ and $r(t)$ are the reference and actual variable respectively.

- The root mean square error ($RMSE$) value

$$RMSE = \sqrt{\frac{1}{t_f - t_i}\int_{t_i}^{t_f} \left(r_{ref}(t) - r(t)\right)^2 dt} \tag{6.3}$$

  where t_f and t_i are the initial and final times tested time interval.

- The integral of the absolute value of the control action normalised with time ($IACA$)

$$IACA = \frac{1}{t_f - t_i} \int_{t_i}^{t_f} |u(t)|\, dt \tag{6.4}$$

where $|u(t)|$ is the $L2$ norm of the command effort [38].

6.3.2 Results

In this section, some of the performance indexes computed for the EMRAC and benchmark controller simulations are reported. The complete indexes tables of all time intervals are listed in appendix B.

Table 6.2: $t_0 - t_{fin}$ Maximum Joint Position and Velocity Norms Error

	$\|q\|$	$\|\dot{q}\|$
EMRAC-EW-FL	0,035496	0,005621
EMRAC-EW-NFL	0,042292	0,00648
EMRAC-UV-FL	0,020052	0,003781
EMRAC-UV-NFL	0,070448	0,009261
ROBUST	0,097596	0,01138
PI	0,099321	0,067176
PD[1]	0,391446	0,01786

Table 6.3: $t_0 - t_{fin}$ RMSE: Joint Position and Velocity Norms

	$\|q\|$	$\|\dot{q}\|$
EMRAC-EW-FL	0,010634	0,00163
EMRAC-EW-NFL	0,008832	0,00124
EMRAC-UV-FL	0,006445	0,000959
EMRAC-UV-NFL	0,013967	0,002009
ROBUST	0,03073	0,003446
PI	0,036952	0,023243
PD[1]	0,261434	0,008198

[1]Performance index computed before the controller crash $\sim 102s$

Table 6.4: $t_0 - t_{fin}$ IACA: Command Activity Norm

	$\|u\|$
EMRAC-EW-FL	23,33846
EMRAC-EW-NFL	20,816
EMRAC-UV-FL	23,80465
EMRAC-UV-NFL	20,4665
ROBUST	19,15357
PI	49,02533
PD[1]	8.2979

7

Conclusion and future work

7.1 Conclusion

From the performance indexes computed in section 6.3, it can be inferred that the EMRAC-EW-NFL and, more in general, all the EMRAC controllers outperform, for the most part, the benchmark control architectures, proving the feasibility of adopting these control strategies on real time nonlinear systems affected by parameter uncertainty and unmodelled dynamics such as space robotic manipulators. The behaviour of each controller can be summarized as:

- The PI controller is not able to ensure zero steady state tracking error and an overall acceptable tracking of the trajectory, also, the IACA value shows a relatively high command effort related to the controller's attempt to compensate for the oscillation induced by the disturbance.

- The PD controller crashes after around $102s$ of performing the trajectory due to a divergence of the control gains. Considering the performance indexes before the crash, the control effort and the tracking performances are the highest and worse, respectively, out of all controllers.

- The robust is the only one, out of the benchmark controllers, that provides acceptable tracking performance performance but, even if it requires a lowest command effort, the tracking performances tend to degrade throughout the simulation ending with a relatively high steady state tracking error (especially on the third joint).

- The EMRAC controllers trade a reasonably small worsening of the command effort with a fairly overall improvement of the tracking performances. In particular, is quite evident the adapting behaviour of the

EMRAC controller by the fact that, after the first segment of the trajectory ($t_1 - t_2$) where the tracking is worse than the Robust controller (see appendix B), the control gains are able to adapt and improve the tracking performance finishing with ~ 0 steady state error.

7.1.1 Consideration on the EMRAC

The application of the EMRAC has shown promising result for controlling a space robotic manipulator. Simulations show that, even without nonlinear compensation, the EMRAC controller is able to compensate the nonlinearities of the robot, furthermore, the feedback linearization lead to an increase of the command effort. In fact, even though the EMRAC-UV-FL has shown the best results from a tracking performance point of view, it is the most demanding in terms of command effort. The EMRAC-EW-NFL on the other hand, seems to be the best trade off between command activity and tracking performance. Finally, it is possible that with a even more refined tuning of the controllers, the behaviour could further improve.

7.1.2 Decentralised EMRAC

All the controller tested until now follow a centralized control approach. It would be interesting to test an additional decentralised version of the EMRAC for comparison. Since this is outside of the scope of the project, only a preliminary testing has been done, without fine tuning of the controller. The following figures and performance indexes show the feasibility of this approach and promising results in terms of performance, similar to the centralised approach.

Table 7.1: $t_0 - t_{fin}$ Performance Indexes

	ME		$RMSE$		$IACA$
	$\|q\|$	$\|\dot{q}\|$	$\|q\|$	$\|\dot{q}\|$	$\|u\|$
EMRAC-DEC	0,0253	0,0051	0,0050	0,0010	22,5061

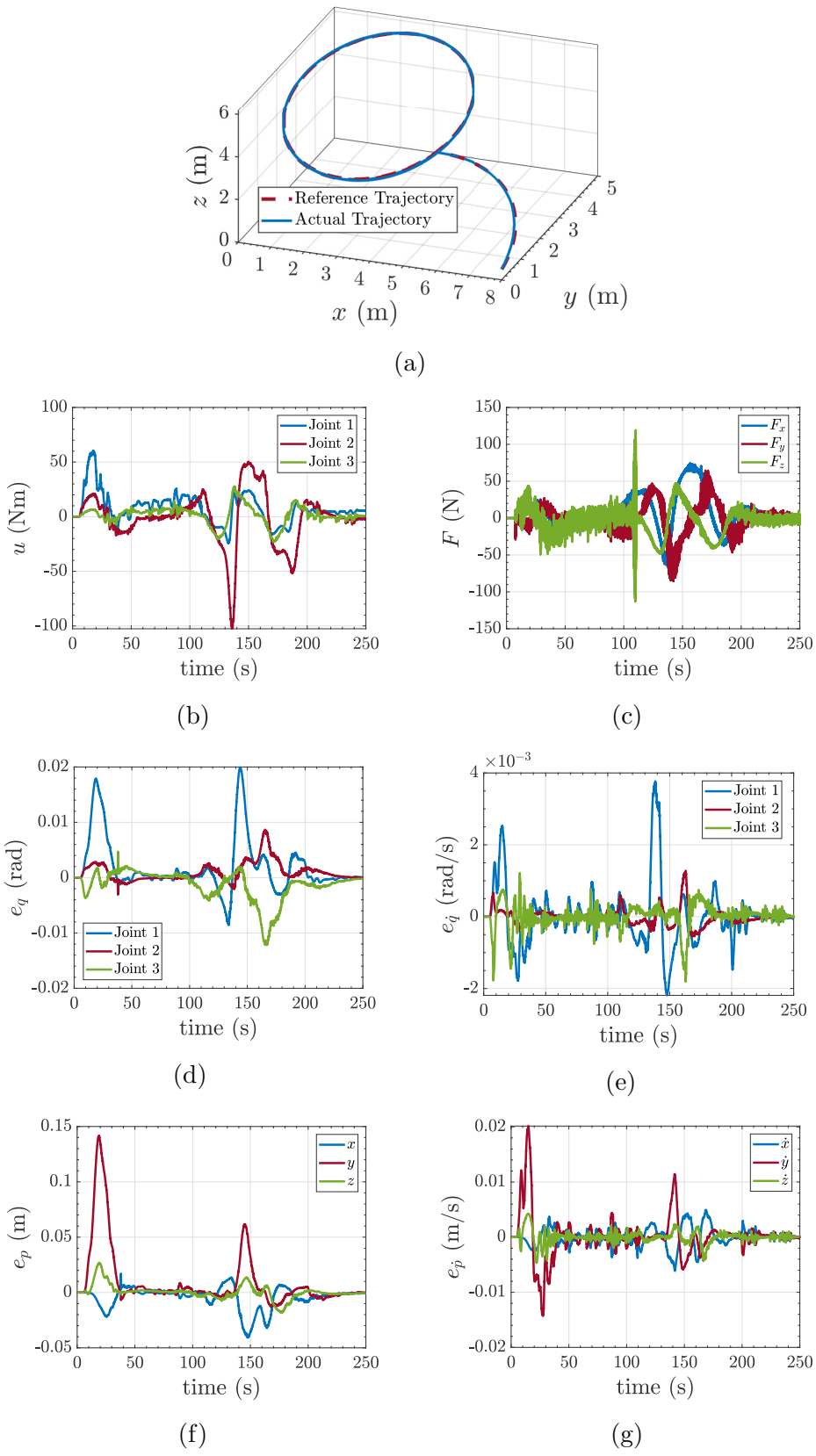


Figure 7.1: EMRAC-DEC

7.2 Future Work

Out of this project, many improvement and future application idea have emerged.

7.2.1 Improvements

A first improvement could be the introduction of more disturbances and elements related to the space environment such as solar radiation, aerodynamic drag and so on. A more extensive parametric analysis with higher unknown target masses could also show even further the capability of the EMRAC and enlarge the range of applications. Since out of the benchmark control strategies the only acceptable behaviour was the one of the robust control, it could be useful to have a comparison with a more comprehensive set of benchmark controllers including, for instance, the ROS standard PID. Another improvement that would validate real life application even further would be the study of the computational complexity that characterise the EMRAC with respect to the benchmark controllers. Finally, a more in-depth analysis of the decentralized approach could be held.

7.2.2 Future Projects

An interesting future project could be the application of the EMRAC for stabilizing the base of a space robotic manipulator in a free flying case (base-link is not fixed and has no restriction) using a similar approach of [39].

Appendix A

Code

A.1 surrey_3dof_space_arm_efmass.urdf.xacro

```
<?xml version="1.0"?>

<!-- Simone Martini, supervisor: Dr Umberto Montanaro.
     Martini's Master Thesis Project: Design & Application of the Enhanced
     Model Reference Adaptive Control to Robotic Manipulator
     Surrey University, Department of Mechanical Engineering Sciences,
     May 2020 -->

<robot xmlns:xacro="http://www.ros.org/wiki/xacro" name="space_arm">

<!-- Import all Gazebo-customization elements, including Gazebo colors -->
<xacro:include filename="$(find surrey_space_arm_description)/urdf/
surrey_3dof_space_arm_efmass.gazebo"/>


<link name="world"/>

<joint name="world_joint" type="fixed">
  <parent link="world"/>
  <child link="base_link"/>
  <origin rpy="0 0 0" xyz="0 0 0"/>
</joint>

<link name="base_link">
  <visual>
    <origin rpy="0 0 0" xyz="0 0 -0.5"/>
    <geometry>
      <box size="1 1 1"/>
    </geometry>
```

```
  </visual>
  <collision>
    <origin rpy="0 0 0" xyz="0 0 -0.5"/>
    <geometry>
      <box size="1 1 1"/>
    </geometry>
  </collision>
  <inertial>
    <origin rpy="0 0 0" xyz="0 0 -0.5"/>
    <mass value="300"/>
    <inertia ixx="300" ixy="0.0" ixz="0.0" iyy="300" iyz="0.0" izz="300"/>
  </inertial>
</link>

<joint name="joint_1" type="continuous">
  <axis xyz="0 0 1" />
  <origin rpy="0 0 0" xyz="0 0 0"/>
  <parent link="base_link"/>
  <child link="link_1"/>
  <dynamics damping="2"/>
</joint>

<link name="link_1">
  <visual>
    <origin rpy="0 0 0" xyz="0 0 0.25"/>
    <geometry>
      <cylinder radius="0.2" length="0.5"/>
    </geometry>
  </visual>
  <collision>
    <origin rpy="0 0 0" xyz="0 0 0.25"/>
    <geometry>
      <cylinder radius="0.2" length="0.5"/>
    </geometry>
  </collision>
  <inertial>
    <origin rpy="0 0 0" xyz="0 0 0.25"/>
    <mass value="50"/>
    <inertia ixx="1.5417" ixy="0.0" ixz="0.0" iyy="1.5417" iyz="0.0"
    izz="1"/>
  </inertial>
</link>

<joint name="joint_2" type="continuous">
  <axis xyz="0 0 1" />
```

```
  <origin rpy="1.57 0 0" xyz="0 0 0.5"/>
  <parent link="link_1"/>
  <child link="link_2"/>
  <dynamics damping="2"/>
</joint>

<link name="link_2">
  <visual>
    <origin rpy="0 0 0" xyz="2 0 0"/>
    <geometry>
      <box size="4 0.2 0.2"/>
    </geometry>
  </visual>
  <collision>
    <origin rpy="0 0 0" xyz="2 0 0"/>
    <geometry>
      <box size="4 0.2 0.2"/>
    </geometry>
  </collision>
  <inertial>
    <origin rpy="0 0 0" xyz="2 0 0"/>
    <mass value="200"/>
    <inertia ixx="0.1" ixy="0.0" ixz="0.0" iyy="268.667" iyz="0.0"
    izz="268.667"/>
  </inertial>
</link>

<joint name="joint_3" type="continuous">
  <axis xyz="0 0 1" />
  <origin rpy="0 0 0" xyz="4 0 0"/>
  <parent link="link_2"/>
  <child link="link_3"/>
  <dynamics damping="2"/>
</joint>

<link name="link_3">
  <visual>
    <origin rpy="0 0 0" xyz="2 0 0"/>
    <geometry>
      <box size="4 0.2 0.2"/>
    </geometry>
  </visual>
  <collision>
    <origin rpy="0 0 0" xyz="2 0 0"/>
    <geometry>
```

```
      <box size="4 0.2 0.2"/>
    </geometry>
  </collision>
  <inertial>
    <origin rpy="0 0 0" xyz="2 0 0"/>
    <mass value="200"/>
    <inertia ixx="0.1" ixy="0.0" ixz="0.0" iyy="268.667" iyz="0.0"
    izz="268.667"/>
  </inertial>
</link>

<joint name="joint_4" type="revolute">
  <axis xyz="0 0 1" />
  <limit effort="10" lower="0" upper="0" velocity="3" />
  <origin rpy="0 0 0" xyz="4 0 0"/>
  <parent link="link_3"/>
  <child link="imu_link"/>
</joint>

<link name="imu_link">
  <visual>
   <origin rpy="0 0 0" xyz="0 0 0"/>
   <geometry>
     <sphere radius="1.0"/>
   </geometry>
  </visual>
  <inertial>
    <origin rpy="0 0 0" xyz="0 0 0"/>
    <mass value="200"/>
    <inertia ixx="80" ixy="0.0" ixz="0.0" iyy="80" iyz="0.0" izz="80"/>
  </inertial>
</link>


<!-- IMU Plug-in -->

<gazebo reference="imu_link">
  <gravity>true</gravity>
  <sensor name="imu_sensor" type="imu">
    <always_on>true</always_on>
    <update_rate>100</update_rate>
    <visualize>true</visualize>
    <topic>__default_topic__</topic>
    <plugin filename="libgazebo_ros_imu_sensor.so" name="imu_plugin">
      <topicName>imu</topicName>
```

```
      <bodyName>imu_link</bodyName>
      <updateRateHZ>100.0</updateRateHZ>
      <gaussianNoise>0.0</gaussianNoise>
      <xyzOffset>0 0 0</xyzOffset>
      <rpyOffset>0 0 0</rpyOffset>
      <frameName>imu_link</frameName>
      <initialOrientationAsReference>false</initialOrientationAsReference>
    </plugin>
    <pose>0 0 0 0 0 0</pose>
  </sensor>
</gazebo>


    <!-- Transmissions -->

<transmission name="tran1">
  <type>transmission_interface/SimpleTransmission</type>
  <joint name="joint_1">
    <hardwareInterface>hardware_interface/EffortJointInterface
    </hardwareInterface>
  </joint>
  <actuator name="motor1">
    <hardwareInterface>hardware_interface/EffortJointInterface
    </hardwareInterface>
    <mechanicalReduction>1</mechanicalReduction>
  </actuator>
</transmission>

<transmission name="tran2">
  <type>transmission_interface/SimpleTransmission</type>
  <joint name="joint_2">
    <hardwareInterface>hardware_interface/EffortJointInterface
    </hardwareInterface>
  </joint>
  <actuator name="motor2">
    <hardwareInterface>hardware_interface/EffortJointInterface
    </hardwareInterface>
    <mechanicalReduction>1</mechanicalReduction>
  </actuator>
</transmission>

<transmission name="tran3">
  <type>transmission_interface/SimpleTransmission</type>
  <joint name="joint_3">
    <hardwareInterface>hardware_interface/EffortJointInterface
```

```
    </hardwareInterface>
  </joint>
  <actuator name="motor3">
    <hardwareInterface>hardware_interface/EffortJointInterface
    </hardwareInterface>
    <mechanicalReduction>1</mechanicalReduction>
  </actuator>
</transmission>

</robot>
```

A.2 surrey_3dof_space_arm_efmass.gazebo

```
<!-- Simone Martini, supervisor: Dr Umberto Montanaro.
     Martini's Master Thesis Project: Design & Application of the Enhanced
     Model Reference Adaptive Control to Robotic Manipulator
     Surrey University, Department of Mechanical Engineering Sciences,
     May 2020 -->

<robot>

<!-- ros_control plugin -->
  <gazebo>
    <plugin name="gazebo_ros_control" filename="libgazebo_ros_control.so">
      <robotNamespace>/space_arm</robotNamespace>
      <robotSimType>gazebo_ros_control/DefaultRobotHWSim</robotSimType>
      <legacyModeNS>true</legacyModeNS>
    </plugin>
  </gazebo>

<!-- base_link -->
  <gazebo reference="base_link">
    <material>Gazebo/Orange</material>
    <selfCollide>false</selfCollide>
  </gazebo>

<!-- link_1 -->
  <gazebo reference="link_1">
    <material>Gazebo/Orange</material>
    <selfCollide>false</selfCollide>
  </gazebo>

<!-- link_2 -->
  <gazebo reference="link_2">
```

```
    <material>Gazebo/Black</material>
    <selfCollide>false</selfCollide>
  </gazebo>

<!-- link_3 -->
  <gazebo reference="link_3">
    <material>Gazebo/Orange</material>
    <selfCollide>false</selfCollide>
  </gazebo>

    <!-- link_4 -->
  <gazebo reference="imu_link">
    <material>Gazebo/Blue</material>
    <selfCollide>false</selfCollide>
  </gazebo>

</robot>
```

A.3 empty_world.world

```
<?xml version="1.0" ?>

<!-- Simone Martini, supervisor: Dr Umberto Montanaro.
     Martini's Master Thesis Project: Design & Application of the Enhanced
     Model Reference Adaptive Control to Robotic Manipulator
     Surrey University, Department of Mechanical Engineering Sciences,
     May 2020 -->

<sdf version="1.4">
  <world name="default">
    <include>
      <uri>model://ground_plane</uri>
    </include>
    <include>
      <uri>model://sun</uri>
    </include>
    <gravity>0 0 0</gravity>
  </world>
</sdf>
```

A.4 my_controller.cpp

```
//  Simone Martini, supervisor: Dr Umberto Montanaro.
//  Martini's Master Thesis Project: Design & Application of the Enhanced
//  Model Reference Adaptive Control to Robotic Manipulator
//  Surrey University, Department of Mechanical Engineering Sciences,
//  May 2020

#include <controller_interface/controller.h>
#include <hardware_interface/joint_command_interface.h>
#include <pluginlib/class_list_macros.h>
#include <std_msgs/Float64.h>
namespace my_controller_ns
{
 class MyPositionController : public
 controller_interface::Controller<hardware_interface::EffortJointInterface>
 {
  public:
  bool init(hardware_interface::EffortJointInterface* hw,
  ros::NodeHandle &n)
  {
   // get joint name from the parameter server
   std::string my_joint;
   if (!n.getParam("joint", my_joint))
   {
    ROS_ERROR("Could not find joint name");
    return false;
   }
   // get the joint object to use in the realtime loop
   joint_ = hw->getHandle(my_joint);  // throws on failure
   command_ = joint_.getEffort();  // set the current joint goal
   po_ = joint_.getPosition();

   // Start command subscriber
   sub_command_ = n.subscribe<std_msgs::Float64>("command", 1,
   &MyPositionController::setCommandCB, this);
   pub_po_ = n.advertise<std_msgs::Float64>("po",1);

   return true;
  }

  void update(const ros::Time& time, const ros::Duration& period)
  {
   joint_.setCommand(command_);
```

```
  }

   void setCommandCB(const std_msgs::Float64ConstPtr& msg)
   {
    command_ = msg->data;
   }

   void starting(const ros::Time& time) { }
   void stopping(const ros::Time& time) { }

   private:
     hardware_interface::JointHandle joint_;
     double command_;
     double po_;
     ros::Subscriber sub_command_;
     ros::Publisher pub_po_;
 };

 PLUGINLIB_EXPORT_CLASS(my_controller_ns::MyPositionController,
 controller_interface::ControllerBase);

}
```

Appendix B

Performance Indexes Tables

B.1 Maximum Error Performance Index

Table B.1: $t_0 - t_{fin}$ Maximum Joint Position and Velocity Error

	q_1	q_2	q_2	$\dot{q_1}$	$\dot{q_2}$	$\dot{q_3}$
EMRAC-EW-FL	0,033726	0,008375	0,018488	0,005575	0,00131	0,002947
EMRAC-EW-NFL	0,029516	0,008867	0,029216	0,004365	0,001006	0,004724
EMRAC-UV-FL	0,019915	0,008673	0,012171	0,00377	0,001277	0,00181
EMRAC-UV-NFL	0,057193	0,021079	0,035668	0,006618	0,002203	0,00632
EMRAC-DEC	0,020669	0,013846	0,008348	0,003826	0,002927	0,001844
ROBUST	0,096551	0,01837	0,022452	0,011369	0,001453	0,001818
PI	0,094752	0,024412	0,066065	0,066299	0,020104	0,03996
PD[1]	0,067411	0,06312	0,386688	0,005919	0,004797	0,017309

Table B.2: $t_0 - t_{fin}$ Maximum Joint Position and Velocity Norm Error

	$\|q\|$	$\|\dot{q}\|$
EMRAC-EW-FL	0,035496	0,005621
EMRAC-EW-NFL	0,042292	0,00648
EMRAC-UV-FL	0,020052	0,003781
EMRAC-UV-NFL	0,070448	0,009261
EMRAC-DEC	0,025315	0,005057
ROBUST	0,097596	0,01138
PI	0,099321	0,067176
PD[1]	0,391446	0,01786

[1]Performance index computed before the controller crash $\sim 102s$

Table B.3: $t_1 - t_2$ Maximum Joint Position and Velocity Error

	q_1	q_2	q_2	$\dot{q_1}$	$\dot{q_2}$	$\dot{q_3}$
EMRAC-EW-FL	0,033726	0,001728	0,011326	0,00452	0,000668	0,002026
EMRAC-EW-NFL	0,029516	0,008867	0,029216	0,004365	0,001006	0,004724
EMRAC-UV-FL	0,017938	0,003087	0,004705	0,002534	0,000668	0,001775
EMRAC-UV-NFL	0,057193	0,021079	0,035668	0,006618	0,002203	0,00632
EMRAC-DEC	0,020669	0,013846	0,005156	0,003826	0,002927	0,001844
ROBUST	0,012925	0,005826	0,009103	0,001181	0,000758	0,00105
PI	0,007735	0,010845	0,023428	0,006273	0,013247	0,021817
PD	0,067411	0,039314	0,269348	0,005919	0,004797	0,017309

Table B.4: $t_1 - t_2$ Maximum Joint Position and Velocity Norm Error

	$\|q\|$	$\|\dot{q}\|$
EMRAC-EW-FL	0,035496	0,004573
EMRAC-EW-NFL	0,042292	0,00648
EMRAC-UV-FL	0,018195	0,002643
EMRAC-UV-NFL	0,070448	0,009261
EMRAC-DEC	0,025315	0,005057
ROBUST	0,014573	0,001415
PI	0,02524	0,025869
PD	0,27171	0,01786

Table B.5: $t_2 - t_{c_0}$ Maximum Joint Position and Velocity Error

	q_1	q_2	q_2	$\dot{q_1}$	$\dot{q_2}$	$\dot{q_3}$
EMRAC-EW-FL	0,001006	0,000322	0,000584	0,000422	3,53E-05	7,51E-05
EMRAC-EW-NFL	0,00257	0,00087	0,002044	0,000168	0,000108	0,00016
EMRAC-UV-FL	0,001046	0,000305	0,001016	0,000306	8,52E-05	0,000134
EMRAC-UV-NFL	0,002495	0,001389	0,003495	0,000197	0,000113	0,00023
EMRAC-DEC	0,000429	0,001273	0,000135	0,000126	0,000415	0,000437
ROBUST	0,00966	0,003319	0,003256	0,001193	0,000234	0,000319
PI	0,007411	0,006274	0,007188	0,004464	0,004776	0,003255

Table B.6: $t_2 - t_{c_0}$ Maximum Joint Position and Velocity Norm Error

	$\|q\|$	$\|\dot{q}\|$
EMRAC-EW-FL	0,001175	0,000424
EMRAC-EW-NFL	0,003287	0,000255
EMRAC-UV-FL	0,00142	0,000311
EMRAC-UV-NFL	0,004353	0,00032
EMRAC-DEC	0,001338	0,000543
ROBUST	0,010721	0,001243
PI	0,009286	0,005992

Table B.7: $t_{c_0} - t_{c_1}$ Maximum Joint Position and Velocity Error

	q_1	q_2	q_2	$\dot{q_1}$	$\dot{q_2}$	$\dot{q_3}$
EMRAC-EW-FL	0,028218	0,005387	0,013094	0,005575	0,000587	0,001591
EMRAC-EW-NFL	0,004094	0,002194	0,001498	0,001527	0,000787	0,000242
EMRAC-UV-FL	0,019915	0,003688	0,004167	0,00377	0,000598	0,000774
EMRAC-UV-NFL	0,004755	0,003924	0,002731	0,001866	0,001061	0,000333
EMRAC-DEC	0,003239	0,010675	0,006673	0,000806	0,002485	0,000919
ROBUST	0,091367	0,01837	0,022452	0,011369	0,001453	0,001818
PI	0,058569	0,024412	0,049398	0,033929	0,020104	0,027831

Table B.8: $t_{c_0} - t_{c_1}$ Maximum Joint Position and Velocity Norm Error

	$\|q\|$	$\|\dot{q}\|$
EMRAC-EW-FL	0,028316	0,005621
EMRAC-EW-NFL	0,004105	0,001528
EMRAC-UV-FL	0,020052	0,003781
EMRAC-UV-NFL	0,006319	0,001876
EMRAC-DEC	0,010779	0,002599
ROBUST	0,093783	0,01138
PI	0,061955	0,035148

Table B.9: $t_{c_1} - t_{c_2}$ Maximum Joint Position and Velocity Error

	q_1	q_2	q_2	$\dot{q_1}$	$\dot{q_2}$	$\dot{q_3}$
EMRAC-EW-FL	0,009304	0,008375	0,018488	0,002448	0,00131	0,002947
EMRAC-EW-NFL	0,005731	0,00236	0,001443	0,000789	0,000739	0,000207
EMRAC-UV-FL	0,009026	0,008673	0,012171	0,001641	0,001277	0,00181
EMRAC-UV-NFL	0,007419	0,004593	0,002102	0,001099	0,001667	0,000276
EMRAC-DEC	0,003628	0,002195	0,008348	0,000562	0,000522	0,001379
ROBUST	0,096551	0,015084	0,01618	0,008768	0,001141	0,001489
PI	0,094752	0,018918	0,066065	0,066299	0,01645	0,03996

Table B.10: $t_{c_1} - t_{c_2}$ Maximum Joint Position and Velocity Norm Error

	$\|q\|$	$\|\dot{q}\|$
EMRAC-EW-FL	0,020756	0,003271
EMRAC-EW-NFL	0,005796	0,000938
EMRAC-UV-FL	0,015415	0,002234
EMRAC-UV-NFL	0,008568	0,001934
EMRAC-DEC	0,008352	0,001435
ROBUST	0,097596	0,008872
PI	0,099321	0,067176

Table B.11: $t_{c_2} - t_{fin}$ Maximum Joint Position and Velocity Error

	q_1	q_2	q_2	$\dot{q_1}$	$\dot{q_2}$	$\dot{q_3}$
EMRAC-EW-FL	0,000118	5,63E-05	0,00018	5,05E-05	5,97E-06	2,8E-05
EMRAC-EW-NFL	0,000103	0,000121	0,000128	3,02E-05	9,18E-06	1,48E-05
EMRAC-UV-FL	0,00019	7,09E-05	0,00043	0,000173	1,63E-05	0,000289
EMRAC-UV-NFL	0,000144	0,000116	0,000167	1,08E-05	1,02E-05	1,57E-05
EMRAC-DEC	3,95E-05	1,28E-05	1,65E-05	1,39E-05	1,17E-05	1,9E-05
ROBUST	0,001139	0,000923	0,00529	0,000114	0,000116	0,000294
PI	0,057954	0,012057	0,053494	0,036705	0,009599	0,031788

Table B.12: $t_{c_2} - t_{fin}$ Maximum Joint Position and Velocity Norm Error

	$\|q\|$	$\|\dot{q}\|$
EMRAC-EW-FL	0,000215	5,07E-05
EMRAC-EW-NFL	0,000185	3,09E-05
EMRAC-UV-FL	0,000449	0,0003
EMRAC-UV-NFL	0,000241	1,92E-05
EMRAC-DEC	4,22E-05	2,29E-05
ROBUST	0,005403	0,000335
PI	0,072244	0,044997

B.2 RMSE Performance Index

Table B.13: $t_0 - t_{fin}$ RMSE: Joint Position and Velocity

	q_1	q_2	q_2	$\dot{q_1}$	$\dot{q_2}$	$\dot{q_3}$
EMRAC-EW-FL	0,008354	0,002264	0,006178	0,001472	0,000274	0,000643
EMRAC-EW-NFL	0,006108	0,00224	0,005973	0,000915	0,000282	0,000788
EMRAC-UV-FL	0,005325	0,002	0,003029	0,000862	0,000229	0,000354
EMRAC-UV-NFL	0,010966	0,004889	0,007136	0,001595	0,000552	0,00109
EMRAC-DEC	0,003486	0,002798	0,002311	0,000688	0,000623	0,000442
ROBUST	0,028849	0,006266	0,008534	0,003372	0,00044	0,000557
PI	0,028928	0,0081	0,021518	0,017905	0,006682	0,013228
PD[1]	0,037583	0,035796	0,25623	0,003269	0,002288	0,007161

Table B.14: $t_0 - t_{fin}$ RMSE: Joint Position and Velocity Norms

	$\|q\|$	$\|\dot{q}\|$
EMRAC-EW-FL	0,010634	0,00163
EMRAC-EW-NFL	0,008832	0,00124
EMRAC-UV-FL	0,006445	0,000959
EMRAC-UV-NFL	0,013967	0,002009
EMRAC-DEC	0,005032	0,001028
ROBUST	0,03073	0,003446
PI	0,036952	0,023243
PD[1]	0,261434	0,008198

Table B.15: $t_1 - t_2$ RMSE: Joint Position and Velocity

	q_1	q_2	q_2	$\dot{q_1}$	$\dot{q_2}$	$\dot{q_3}$
EMRAC-EW-FL	0,014875	0,000946	0,005273	0,002053	0,000163	0,00076
EMRAC-EW-NFL	0,013135	0,004501	0,013152	0,001936	0,000473	0,001741
EMRAC-UV-FL	0,008058	0,001561	0,001571	0,001023	0,000176	0,000539
EMRAC-UV-NFL	0,023945	0,010306	0,015698	0,003449	0,001076	0,002409
EMRAC-DEC	0,007535	0,005277	0,001963	0,001485	0,001064	0,00069
ROBUST	0,007766	0,002935	0,00393	0,000678	0,000266	0,000418
PI	0,003487	0,004702	0,008273	0,002903	0,005146	0,007529
PD	0,039569	0,020231	0,123968	0,003692	0,00267	0,0096

Table B.16: $t_1 - t_2$ RMSE: Joint Position and Velocity Norms

	$\|q\|$	$\|\dot{q}\|$
EMRAC-EW-FL	0,01581	0,002195
EMRAC-EW-NFL	0,019125	0,002647
EMRAC-UV-FL	0,008357	0,00117
EMRAC-UV-NFL	0,03043	0,004342
EMRAC-DEC	0,009406	0,001952
ROBUST	0,009185	0,00084
PI	0,010134	0,00957
PD	0,131693	0,010626

Table B.17: $t_2 - t_{c_0}$ RMSE: Joint Position and Velocity

	q_1	q_2	q_2	$\dot{q_1}$	$\dot{q_2}$	$\dot{q_3}$
EMRAC-EW-FL	0,000717	0,000286	0,000497	0,000303	2,1E-05	3,42E-05
EMRAC-EW-NFL	0,002499	0,000835	0,001764	8,97E-05	4,9E-05	0,000111
EMRAC-UV-FL	0,000836	0,000223	0,000846	0,000183	3,12E-05	6,72E-05
EMRAC-UV-NFL	0,002393	0,001275	0,003016	0,000127	8,67E-05	0,000188
EMRAC-DEC	0,000318	0,000703	4,74E-05	7,67E-05	0,000247	0,000153
ROBUST	0,006851	0,002787	0,00255	0,001116	0,000194	0,000265
PI	0,005444	0,004644	0,00476	0,003058	0,003209	0,002297

Table B.18: $t_2 - t_{c_0}$ RMSE: Joint Position and Velocity Norms

	$\|q\|$	$\|\dot{q}\|$
EMRAC-EW-FL	0,000918	0,000305
EMRAC-EW-NFL	0,003171	0,000151
EMRAC-UV-FL	0,00121	0,000198
EMRAC-UV-NFL	0,004055	0,000243
EMRAC-DEC	0,000772	0,000301
ROBUST	0,007824	0,001163
PI	0,008595	0,004992

Table B.19: $t_{c_0} - t_{c_1}$ RMSE: Joint Position and Velocity

	q_1	q_2	q_2	$\dot{q_1}$	$\dot{q_2}$	$\dot{q_3}$
EMRAC-EW-FL	0,007712	0,002073	0,00565	0,001768	0,000248	0,0005
EMRAC-EW-NFL	0,001378	0,000984	0,000723	0,000313	0,000228	6,15E-05
EMRAC-UV-FL	0,005918	0,001157	0,001569	0,001097	0,000174	0,000194
EMRAC-UV-NFL	0,00181	0,001684	0,00104	0,000402	0,000268	7,44E-05
EMRAC-DEC	0,000783	0,002333	0,002086	0,000191	0,000635	0,000294
ROBUST	0,030004	0,006647	0,010673	0,003965	0,000521	0,00065
PI	0,017125	0,008669	0,017447	0,00905	0,006873	0,011038

Table B.20: $t_{c_0} - t_{c_1}$ RMSE: Joint Position and Velocity Norms

	$\|q\|$	$\|\dot{q}\|$
EMRAC-EW-FL	0,009782	0,001854
EMRAC-EW-NFL	0,001842	0,000392
EMRAC-UV-FL	0,00623	0,001127
EMRAC-UV-NFL	0,002682	0,000488
EMRAC-DEC	0,003226	0,000726
ROBUST	0,032532	0,004052
PI	0,025938	0,015842

Table B.21: $t_{c_1} - t_{c_2}$ RMSE: Joint Position and Velocity

	q_1	q_2	q_2	$\dot{q_1}$	$\dot{q_2}$	$\dot{q_3}$
EMRAC-EW-FL	0,002747	0,003025	0,007518	0,000653	0,000358	0,000745
EMRAC-EW-NFL	0,001856	0,001181	0,000582	0,000312	0,000198	4,82E-05
EMRAC-UV-FL	0,002464	0,002862	0,004594	0,000485	0,000309	0,000376
EMRAC-UV-NFL	0,002282	0,001771	0,000362	0,000394	0,000339	4,49E-05
EMRAC-DEC	0,000958	0,000612	0,002837	0,000163	0,000141	0,000425
ROBUST	0,036458	0,007579	0,008617	0,003861	0,000462	0,000563
PI	0,043423	0,009288	0,028902	0,02733	0,007509	0,017173

Table B.22: $t_{c_1} - t_{c_2}$ RMSE: Joint Position and Velocity Norms

	$\|q\|$	$\|\dot{q}\|$
EMRAC-EW-FL	0,008556	0,001054
EMRAC-EW-NFL	0,002276	0,000373
EMRAC-UV-FL	0,005947	0,000687
EMRAC-UV-NFL	0,002912	0,000522
EMRAC-DEC	0,003056	0,000477
ROBUST	0,038221	0,003929
PI	0,052983	0,033139

Table B.23: $t_{c_2} - t_{fin}$ RMSE: Joint Position and Velocity

	q_1	q_2	q_2	$\dot{q_1}$	$\dot{q_2}$	$\dot{q_3}$
EMRAC-EW-FL	8,15E-05	4,82E-05	0,000154	2,9E-05	3,86E-06	1,26E-05
EMRAC-EW-NFL	9,29E-05	0,000105	0,00011	1,44E-05	6,75E-06	7,42E-06
EMRAC-UV-FL	0,00014	6,14E-05	0,000297	9,16E-05	6,77E-06	0,000107
EMRAC-UV-NFL	0,000138	9,81E-05	0,000144	7,42E-06	7,91E-06	9,23E-06
EMRAC-DEC	3,11E-05	7,66E-06	7,18E-06	5,96E-06	4,79E-06	6,93E-06
ROBUST	0,000895	0,000678	0,004606	0,000102	9,32E-05	0,000253
PI	0,040441	0,00818	0,038288	0,026396	0,00688	0,02239

Table B.24: $t_{c_2} - t_{fin}$ RMSE: Joint Position and Velocity Norms

	$\|q\|$	$\|\dot{q}\|$
EMRAC-EW-FL	0,000181	3,18E-05
EMRAC-EW-NFL	0,000178	1,75E-05
EMRAC-UV-FL	0,000334	0,000141
EMRAC-UV-NFL	0,000222	1,42E-05
EMRAC-DEC	3,28E-05	1,03E-05
ROBUST	0,004741	0,000289
PI	0,056288	0,03529

B.3 IACA Performance Index

Table B.25: $t_0 - t_{fin}$ IACA: Command Activity

	u_1	u_1	u_1	$\|u\|$
EMRAC-EW-FL	10,0083	17,58968	6,718147	23,33846
EMRAC-EW-NFL	8,313878	15,98405	5,140425	20,816
EMRAC-UV-FL	11,92473	16,15269	6,226673	23,80465
EMRAC-UV-NFL	8,003725	16,23979	5,383781	20,4665
EMRAC-DEC	7,791614	17,07115	9,066284	22,5061
ROBUST	7,795469	15,04397	5,138355	19,15357
PI	29,2915	28,96	13,16318	49,02533
PD[1]	5.9278	5.2993	1.2778	8.2979

Table B.26: $t_1 - t_2$ IACA: Command Activity

	u_1	u_1	u_1	$\|u\|$
EMRAC-EW-FL	16,00021	16,15562	4,553587	24,56047
EMRAC-EW-NFL	16,41405	8,219845	3,01922	19,42327
EMRAC-UV-FL	17,81794	11,15986	3,719923	23,37016
EMRAC-UV-NFL	14,86408	11,4975	3,723625	20,01349
EMRAC-DEC	11,89894	17,69691	8,029068	24,13037
ROBUST	12,13427	10,21595	2,952094	16,7074
PI	17,52426	16,37346	4,401358	26,40749
PD	9,510323	8,058526	1,803333	12,79238

Table B.27: $t_2 - t_{c_0}$ IACA: Command Activity

	u_1	u_1	u_1	$\|u\|$
EMRAC-EW-FL	7,229367	10,07138	4,109146	13,12047
EMRAC-EW-NFL	2,937161	1,887366	0,239152	3,507339
EMRAC-UV-FL	12,25873	1,631375	3,811622	12,95363
EMRAC-UV-NFL	1,243814	1,101323	0,54009	1,793705
EMRAC-DEC	0,586401	13,00954	8,829047	15,75658
ROBUST	0,544812	2,32941	1,554067	2,865435
PI	10,79698	16,84366	4,484654	21,53736

Table B.28: $t_{c_0} - t_{c_1}$ IACA: Command Activity

	u_1	u_1	u_1	$\|u\|$
EMRAC-EW-FL	10,39469	21,17269	9,259708	27,12243
EMRAC-EW-NFL	6,35315	19,84855	6,972342	23,29346
EMRAC-UV-FL	13,56824	18,40942	8,742023	27,87334
EMRAC-UV-NFL	7,336347	18,94393	6,895071	22,73575
EMRAC-DEC	7,570509	20,88341	10,70157	26,54072
ROBUST	7,561509	17,32165	6,393706	21,37318
PI	16,56648	31,23956	11,02125	40,65419

Table B.29: $t_{c_1} - t_{c_2}$ IACA: Command Activity

	u_1	u_1	u_1	$\|u\|$
EMRAC-EW-FL	7,306096	17,06251	6,087292	21,70388
EMRAC-EW-NFL	6,814355	18,90287	5,312098	22,18262
EMRAC-UV-FL	8,02212	19,15212	5,860326	22,88905
EMRAC-UV-NFL	6,107811	18,75688	5,628555	21,69057
EMRAC-DEC	6,835852	14,64653	8,829952	19,98113
ROBUST	6,859662	17,76757	5,839068	21,2547
PI	48,18928	37,12627	20,76485	71,90708

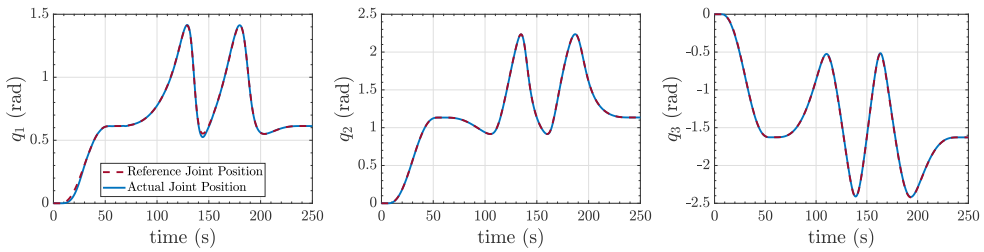
Table B.30: $t_{c_2} - t_{fin}$ IACA: Command Activity

	u_1	u_1	u_1	$\|u\|$
EMRAC-EW-FL	4,561011	2,0331	3,298205	5,985726
EMRAC-EW-NFL	3,287261	1,606843	0,32956	3,674158
EMRAC-UV-FL	5,246983	2,143993	1,253816	6,04961
EMRAC-UV-NFL	0,303041	1,05895	0,603691	1,256167
EMRAC-DEC	2,898763	6,96869	3,56074	8,345409
ROBUST	0,509391	1,059388	0,514441	1,283499
PI	83,63999	7,876947	24,34779	90,52605

Appendix C

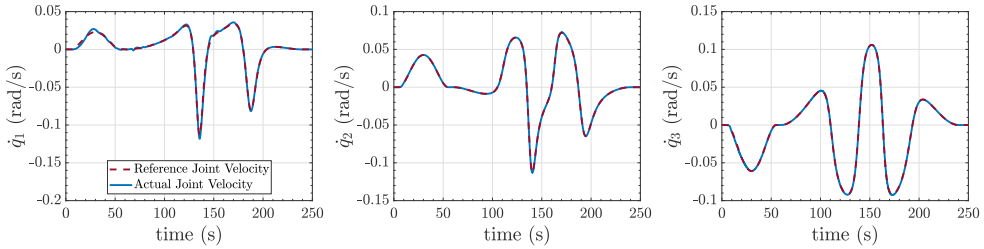
Simulations Figures

C.1 EMRAC-EW-FL



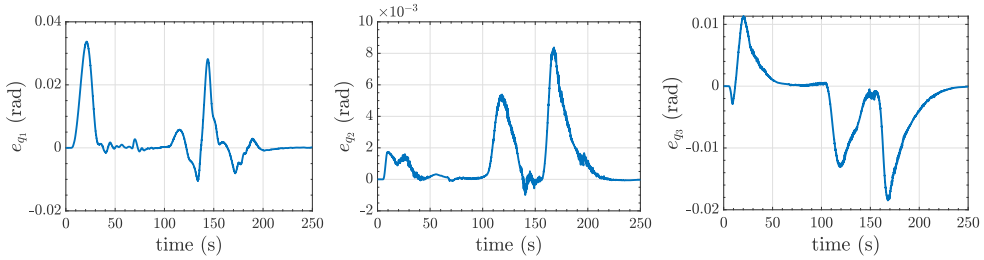
(a) (b) (c)

Figure C.1: EMRAC-EW-FL Joint Position



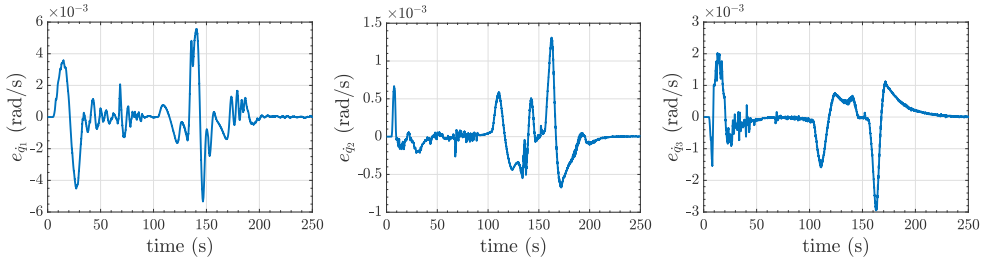
(a) (b) (c)

Figure C.2: EMRAC-EW-FL Joint Velocity



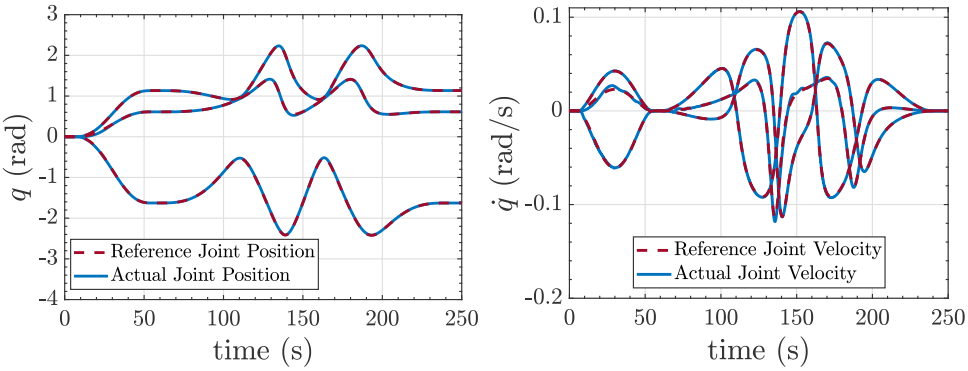
(a) (b) (c)

Figure C.3: EMRAC-EW-FL Joint Position Error



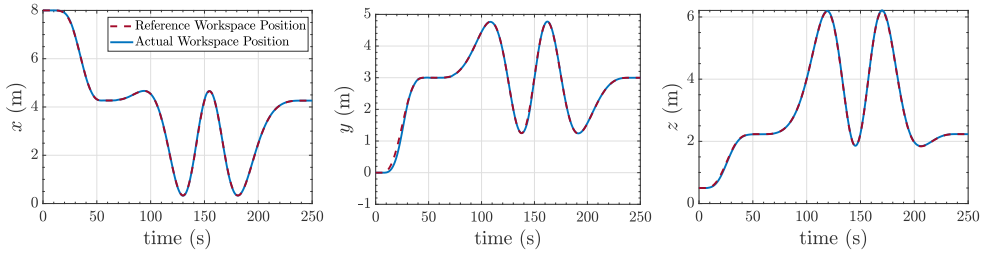
(a) (b) (c)

Figure C.4: EMRAC-EW-FL Joint Velocity Error



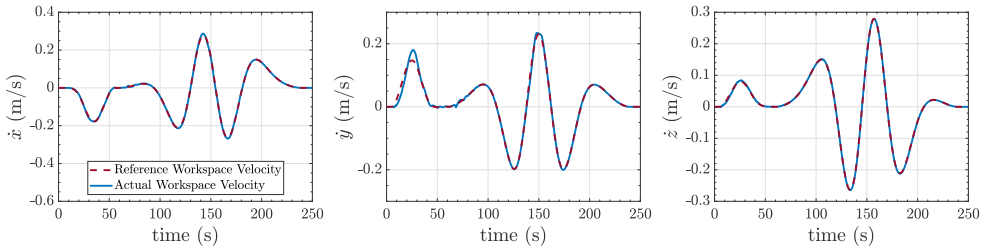
(a) (b)

Figure C.5: EMRAC-EW-FL Joint Position and Velocity



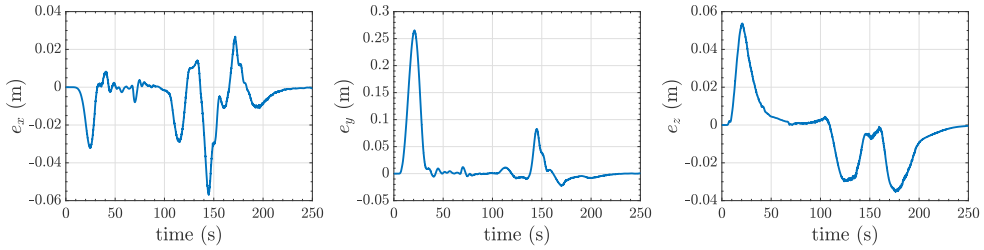
(a) (b) (c)

Figure C.6: EMRAC-EW-FL Operational Space Position



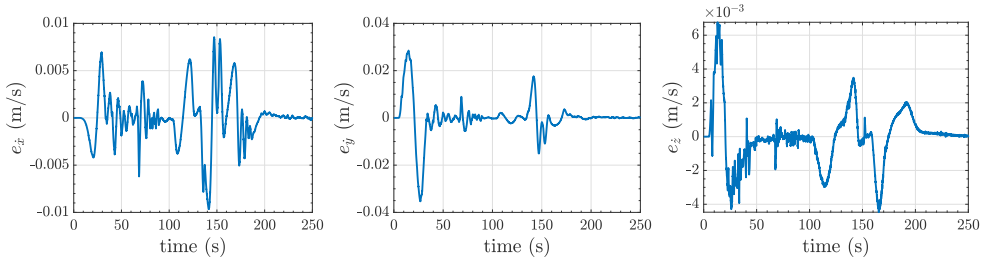
(a) (b) (c)

Figure C.7: EMRAC-EW-FL Operational Space Velocity



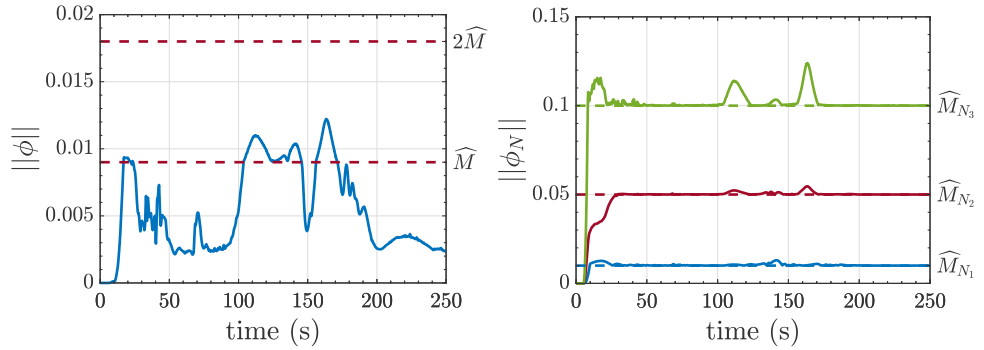
(a) (b) (c)

Figure C.8: EMRAC-EW-FL Operational Space Position Error



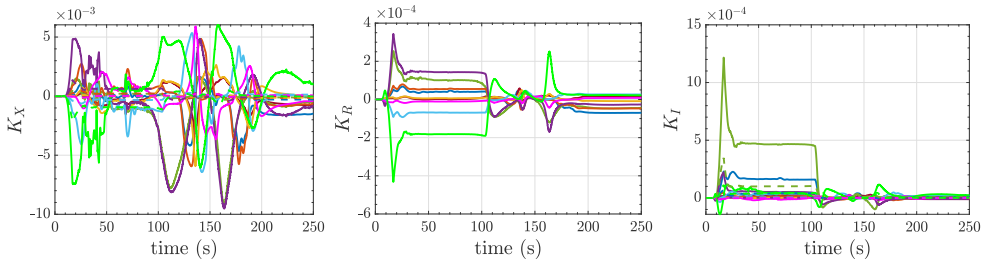
(a) (b) (c)

Figure C.9: EMRAC-EW-FL Operational Space Velocity Error



(a) (b)

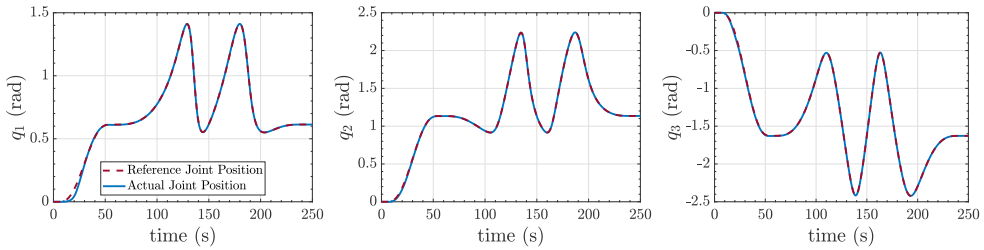
Figure C.10: EMRAC-EW-FL $||\phi||$ and $||\phi_N||$



(a) (b) (c)

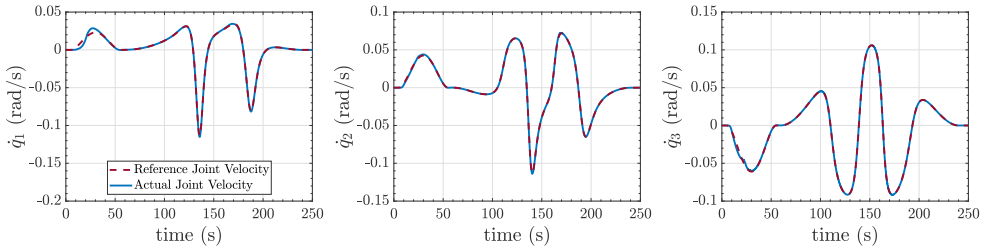
Figure C.11: EMRAC-EW-FL Control Gains

C.2 EMRAC-UV-NFL



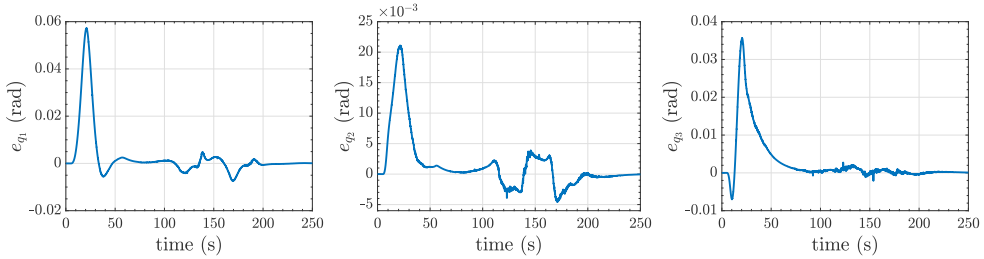
(a) (b) (c)

Figure C.12: EMRAC-UV-NFL Joint Position



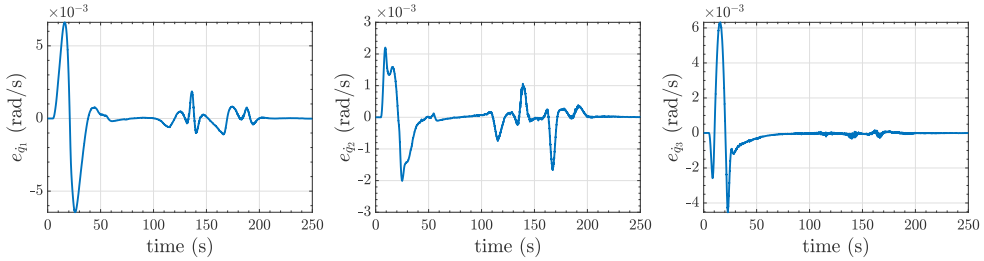
(a) (b) (c)

Figure C.13: EMRAC-UV-NFL Joint Velocity



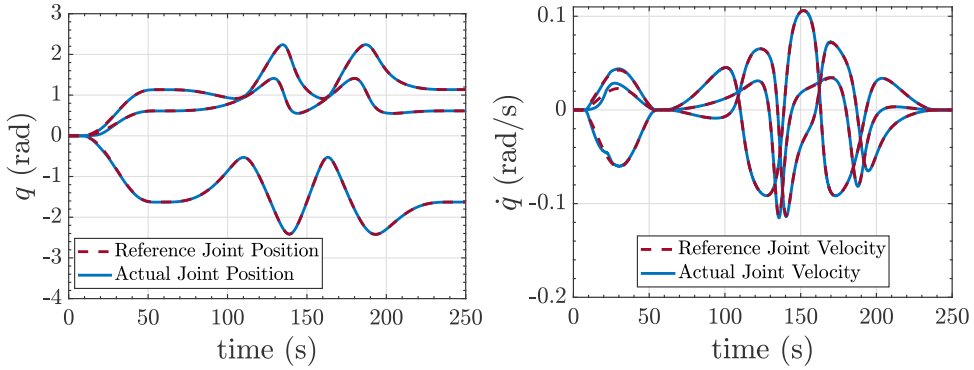
(a) (b) (c)

Figure C.14: EMRAC-UV-NFL Joint Position Error



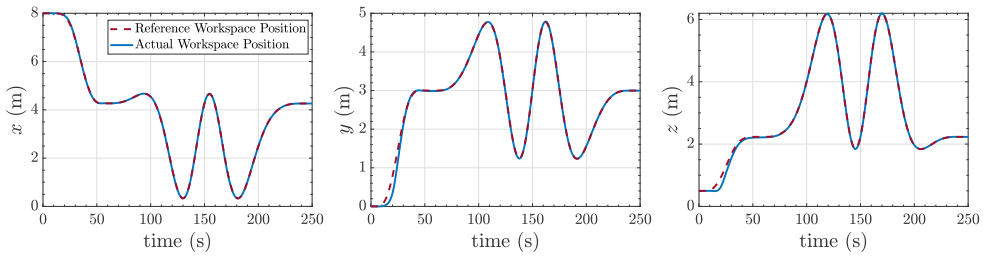
(a) (b) (c)

Figure C.15: EMRAC-UV-NFL Joint Velocity Error



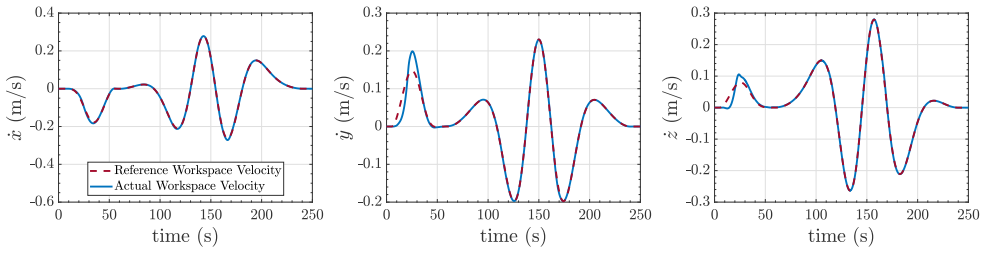
(a) (b)

Figure C.16: EMRAC-UV-NFL Joint Position and Velocity



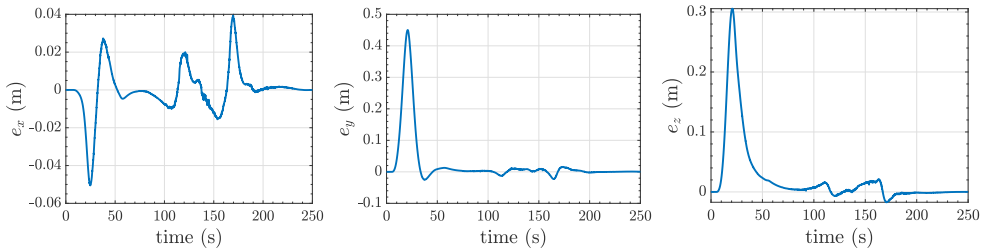
(a) (b) (c)

Figure C.17: EMRAC-UV-NFL Operational Space Position



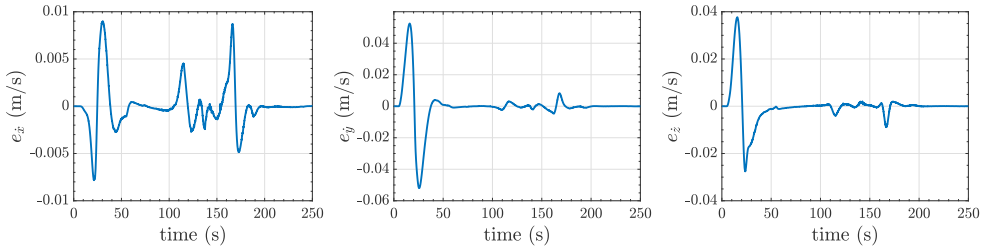
(a) (b) (c)

Figure C.18: EMRAC-UV-NFL Operational Space Velocity



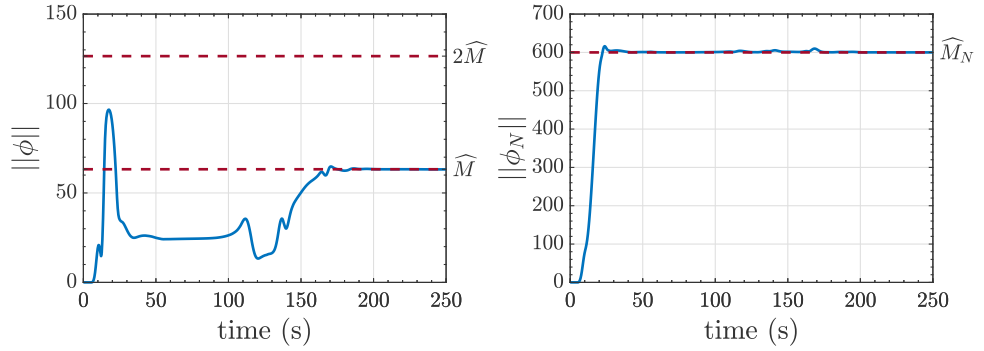
(a) (b) (c)

Figure C.19: EMRAC-UV-NFL Operational Space Position Error



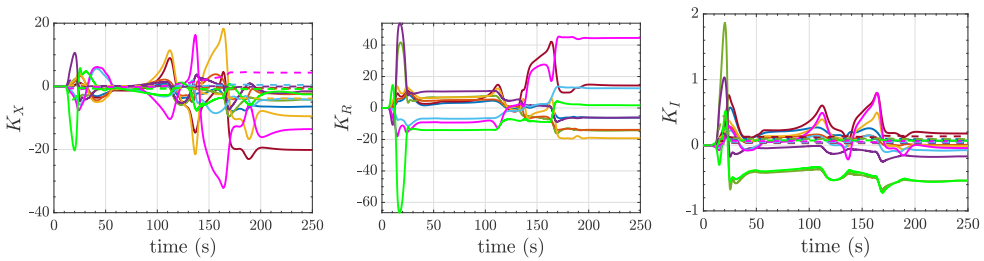
(a) (b) (c)

Figure C.20: EMRAC-UV-NFL Operational Space Velocity Error



(a) (b)

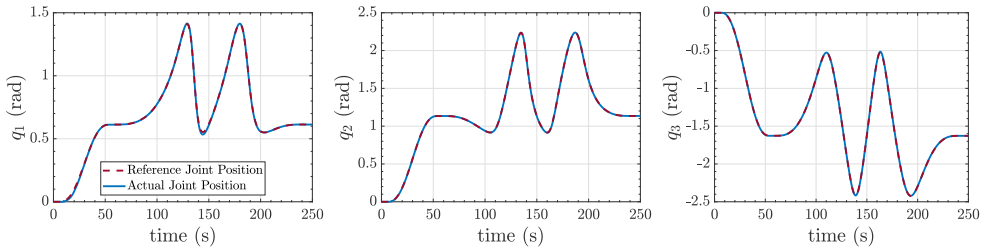
Figure C.21: EMRAC-UV-NFL $||\phi||$ and $||\phi_N||$



(a) (b) (c)

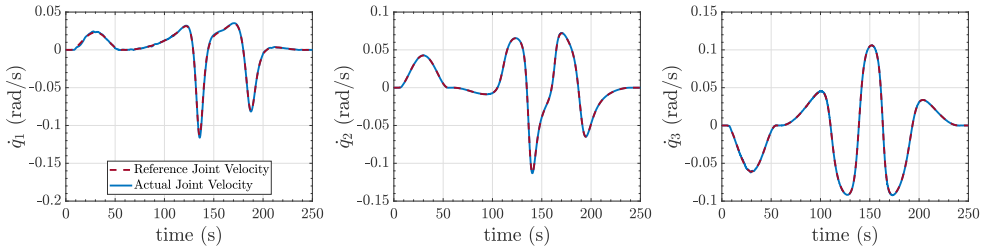
Figure C.22: EMRAC-UV-NFL Control Gains

C.3 EMRAC-UV-FL



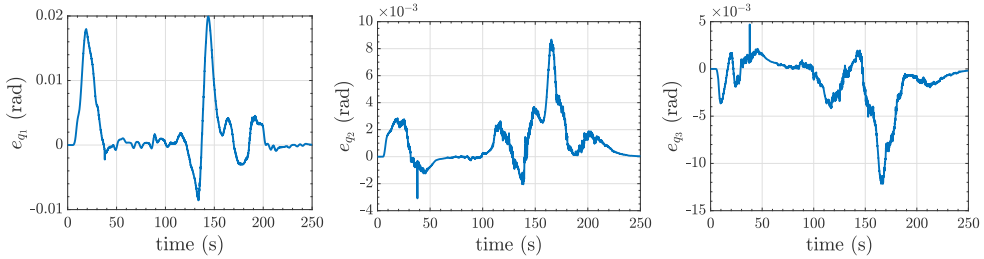
(a) (b) (c)

Figure C.23: EMRAC-UV-FL Joint Position



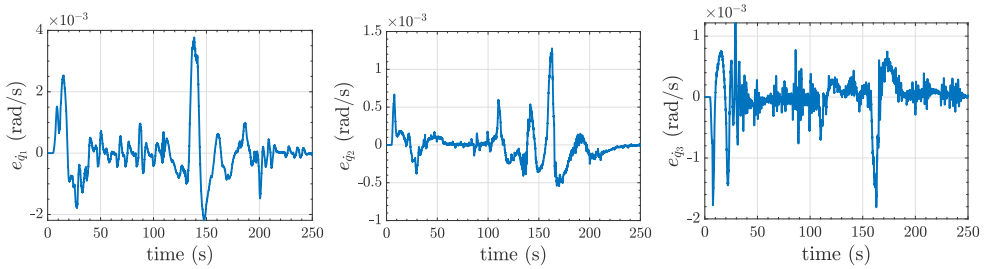
(a) (b) (c)

Figure C.24: EMRAC-UV-FL Joint Velocity



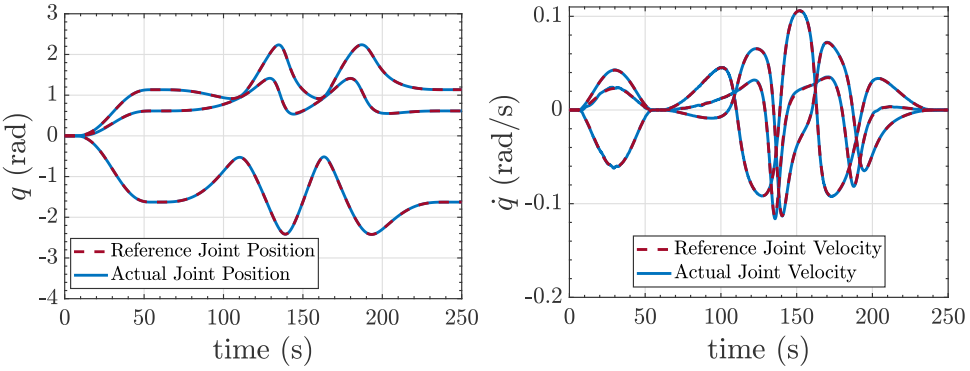
(a) (b) (c)

Figure C.25: EMRAC-UV-FL Joint Position Error



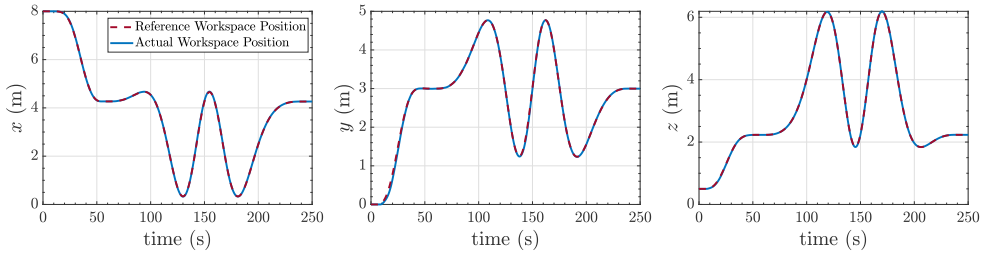
(a) (b) (c)

Figure C.26: EMRAC-UV-FL Joint Velocity Error



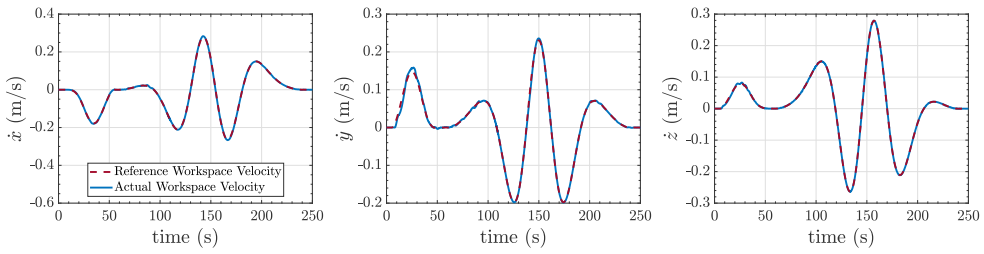
(a) (b)

Figure C.27: EMRAC-UV-FL Joint Position and Velocity



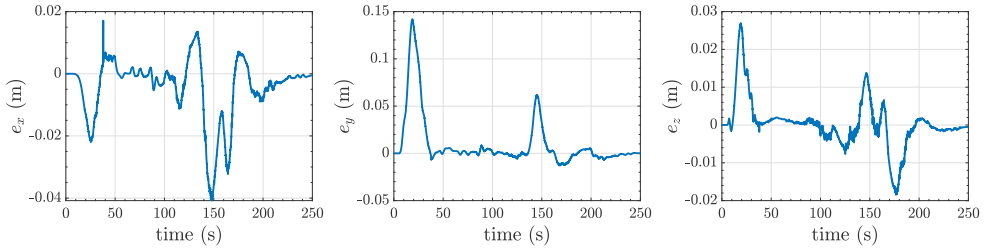
(a) (b) (c)

Figure C.28: EMRAC-UV-FL Operational Space Position



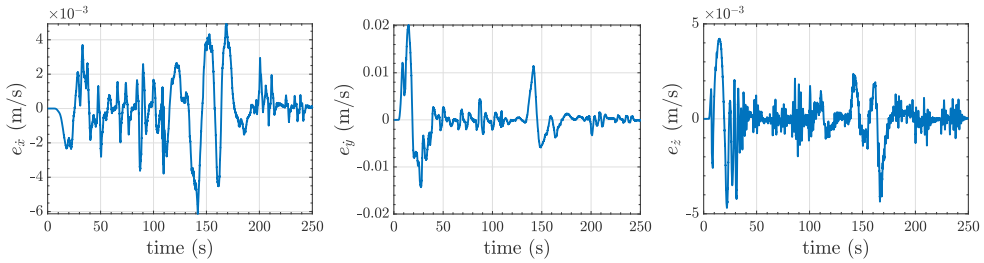
(a) (b) (c)

Figure C.29: EMRAC-UV-FL Operational Space Velocity



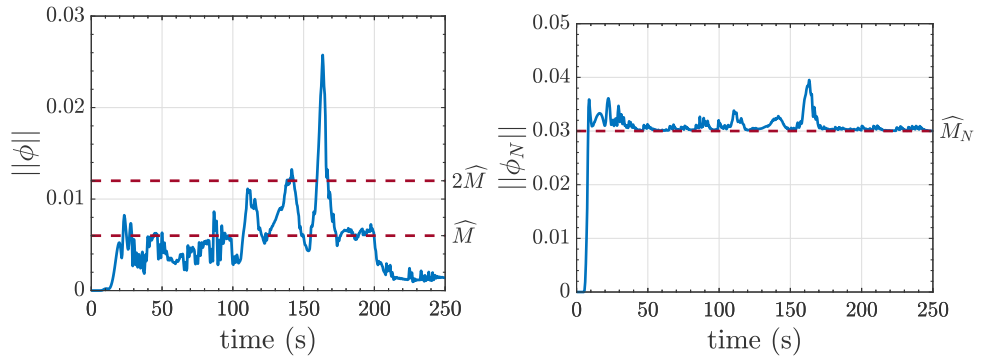
(a) (b) (c)

Figure C.30: EMRAC-UV-FL Operational Space Position Error



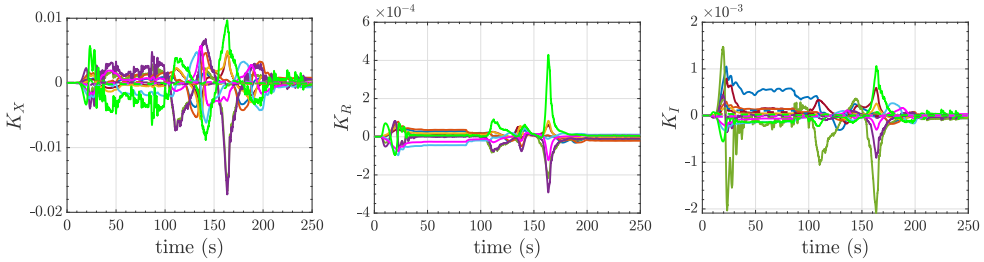
(a) (b) (c)

Figure C.31: EMRAC-UV-FL Operational Space Velocity Error



(a) (b)

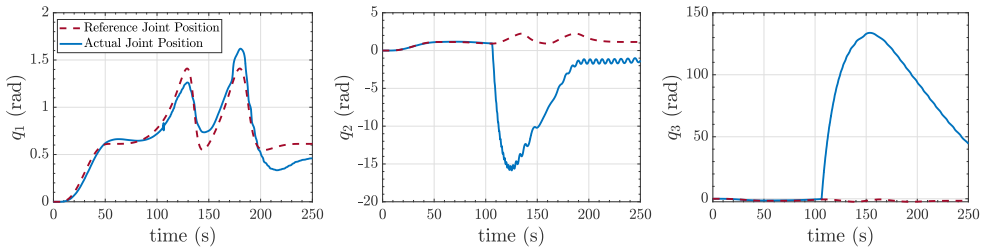
Figure C.32: EMRAC-UV-FL $||\phi||$ and $||\phi_N||$



(a) (b) (c)

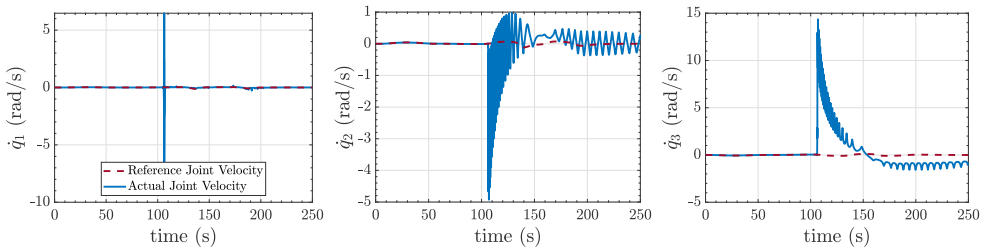
Figure C.33: EMRAC-UV-FL Control Gains

C.4 PD



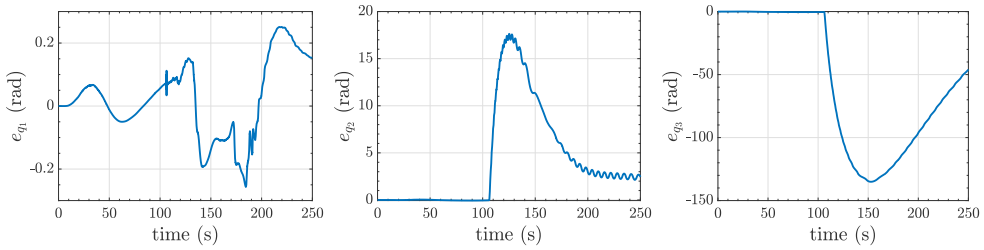
(a) (b) (c)

Figure C.34: PD Joint Position



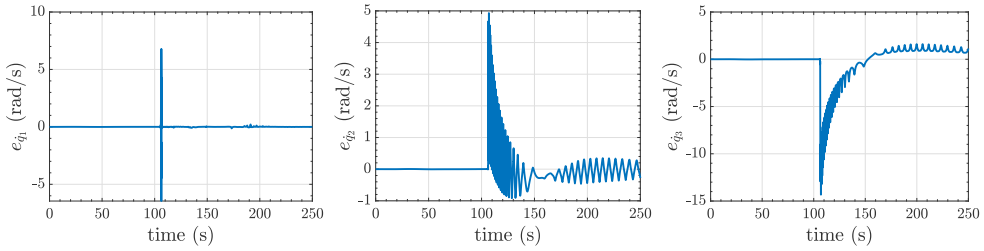
(a) (b) (c)

Figure C.35: PD Joint Velocity



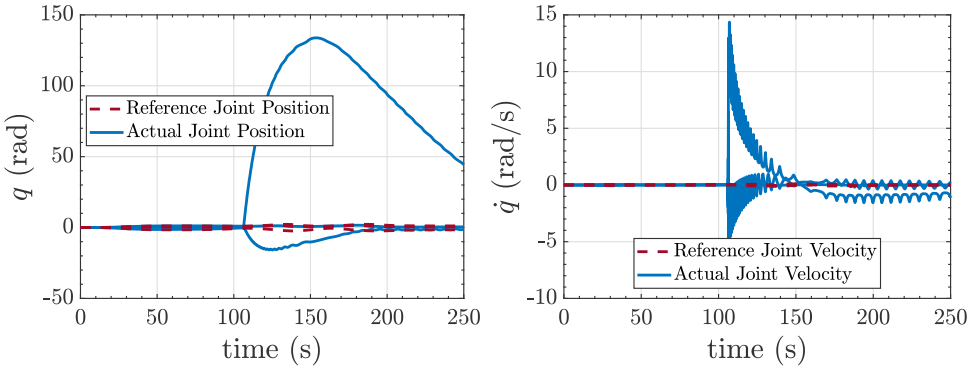
(a) (b) (c)

Figure C.36: PD Joint Position Error



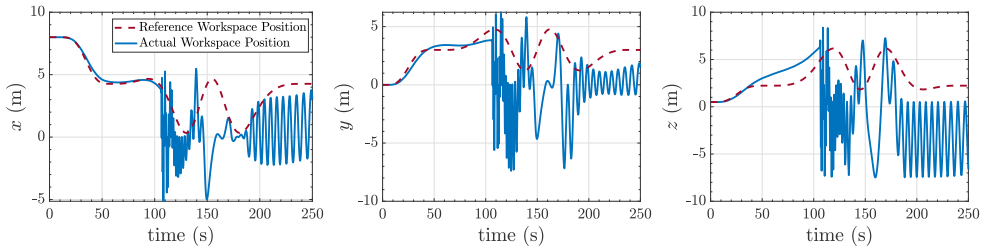
(a) (b) (c)

Figure C.37: PD Joint Velocity Error



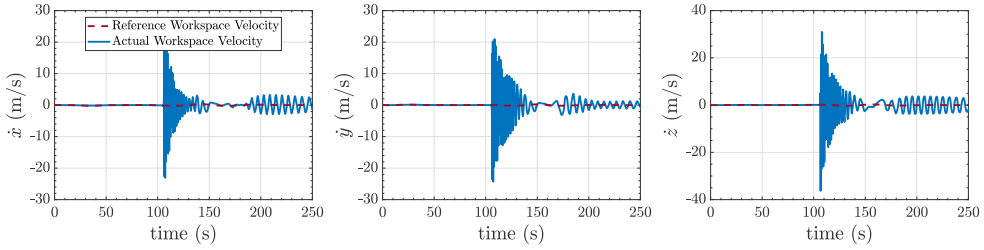
(a) (b)

Figure C.38: PD Joint Position and Velocity



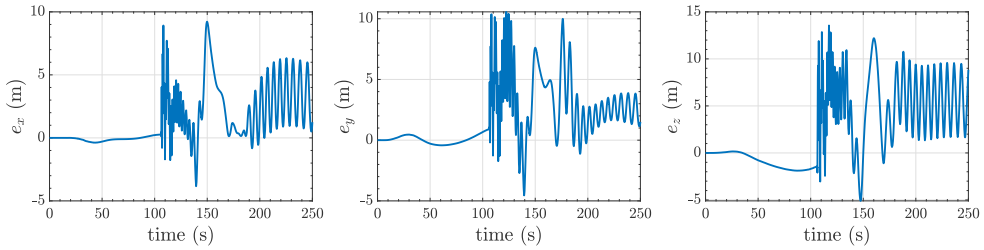
(a) (b) (c)

Figure C.39: PD Operational Space Position



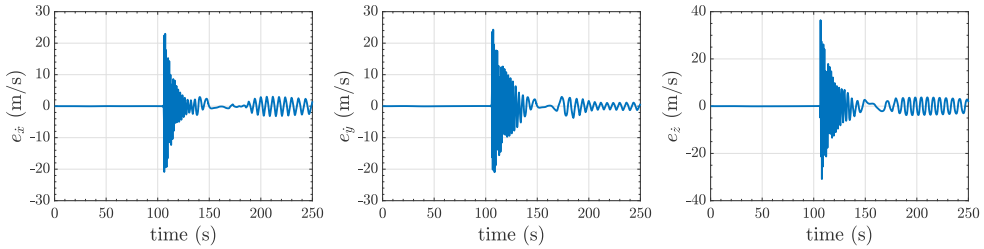
(a) (b) (c)

Figure C.40: PD Operational Space Velocity



(a) (b) (c)

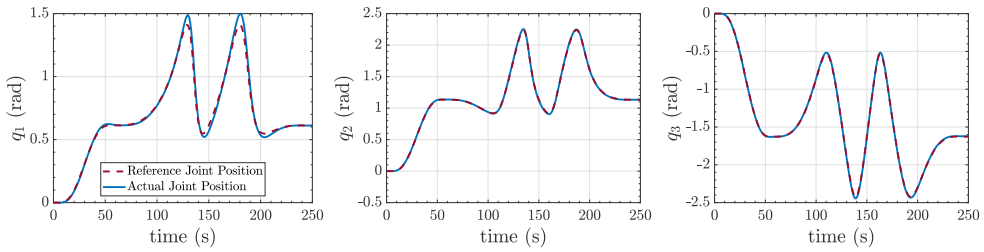
Figure C.41: PD Operational Space Position Error



(a) (b) (c)

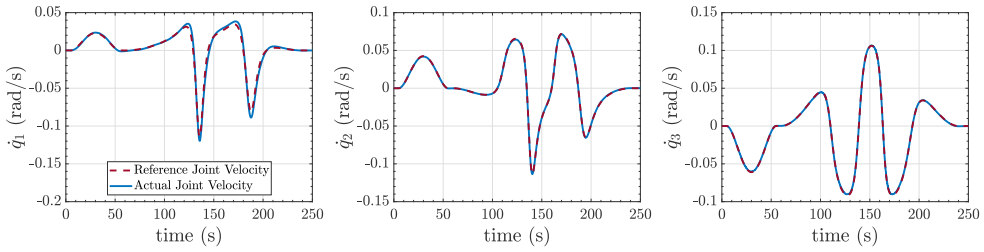
Figure C.42: PD Operational Space Velocity Error

C.5 Robust



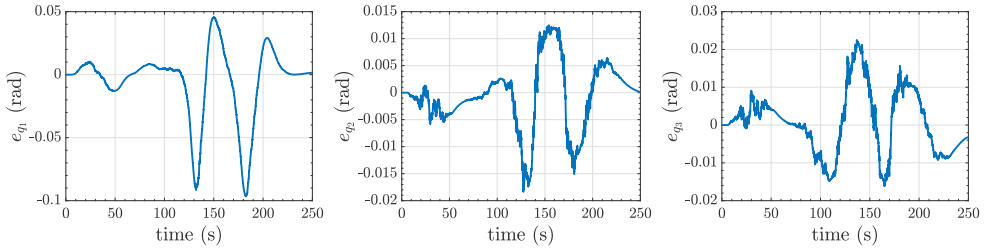
(a) (b) (c)

Figure C.43: Robust Joint Position



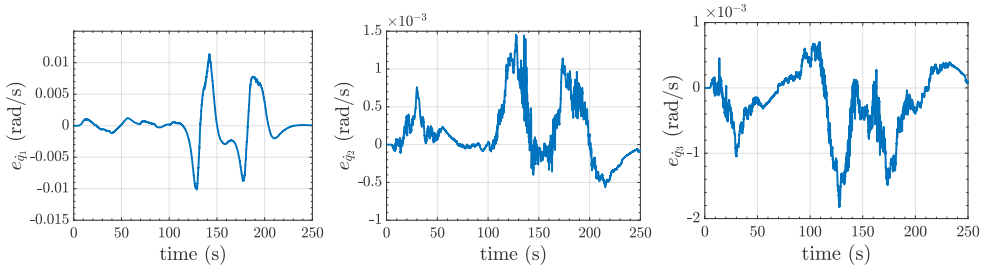
(a) (b) (c)

Figure C.44: Robust Joint Velocity



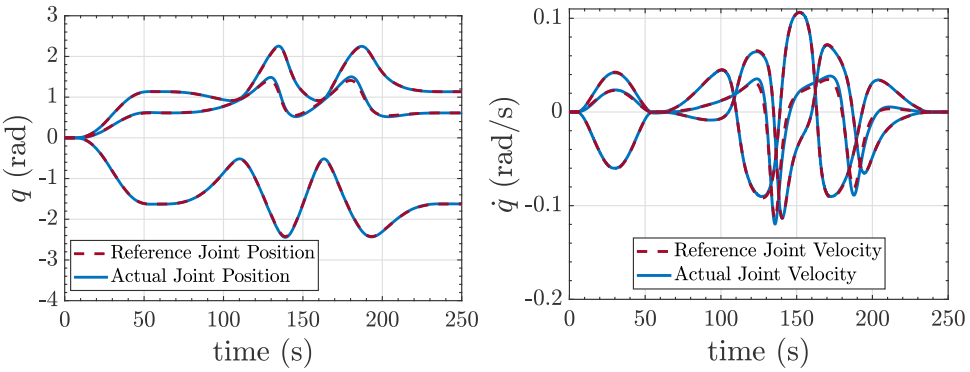
(a) (b) (c)

Figure C.45: Robust Joint Position Error



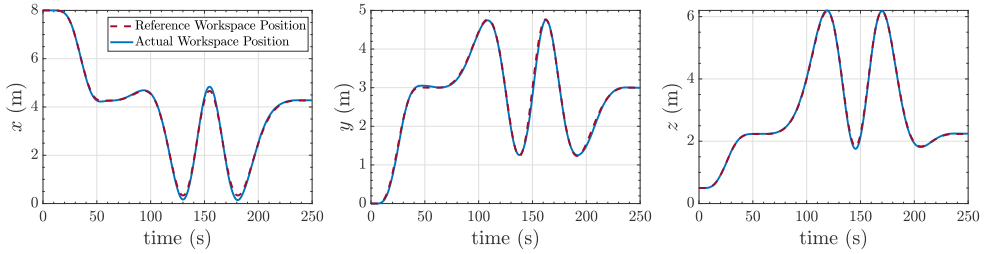
(a) (b) (c)

Figure C.46: Robust Joint Velocity Error



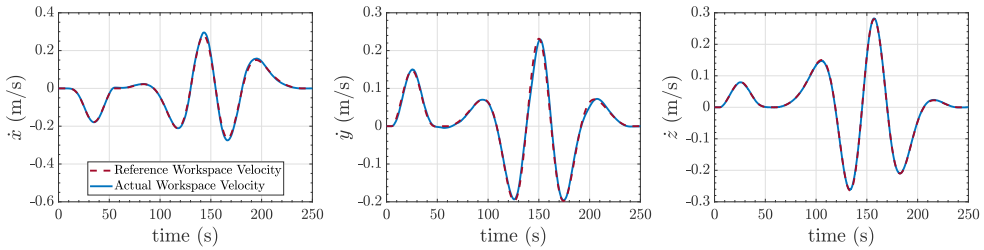
(a) (b)

Figure C.47: Robust Joint Position and Velocity



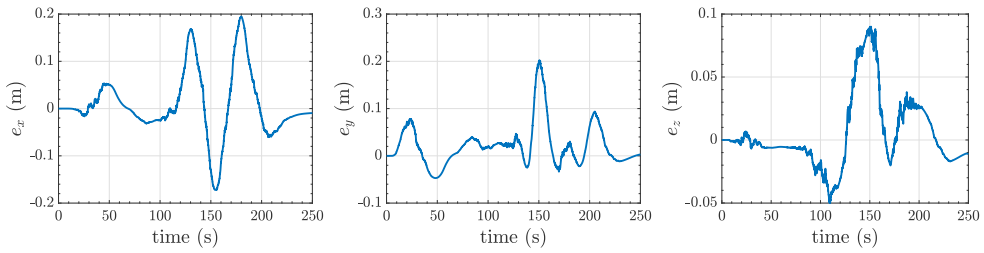
(a) (b) (c)

Figure C.48: Robust Operational Space Position



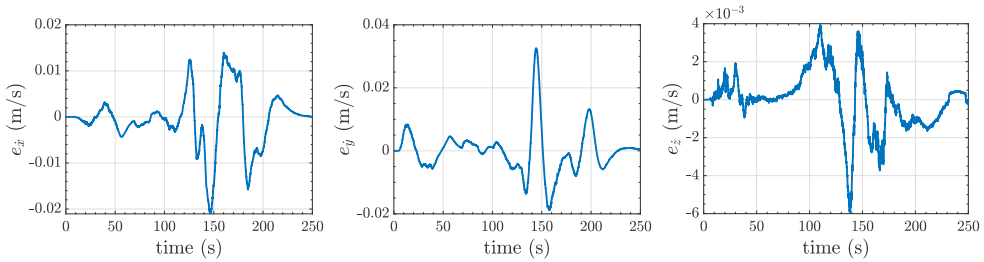
(a) (b) (c)

Figure C.49: Robust Operational Space Velocity



(a) (b) (c)

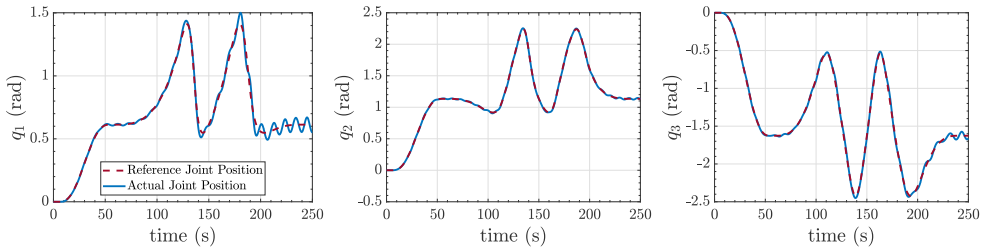
Figure C.50: Robust Operational Space Position Error



(a) (b) (c)

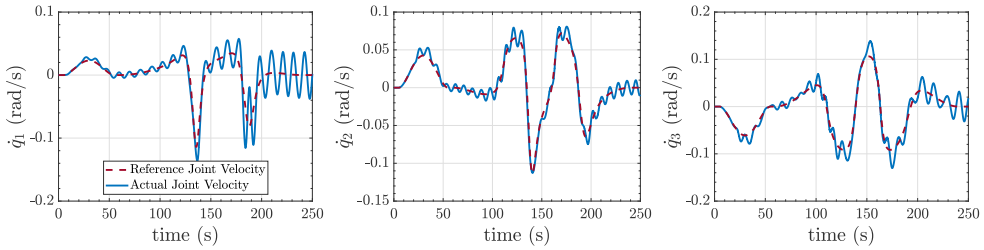
Figure C.51: Robust Operational Space Velocity Error

C.6 PI



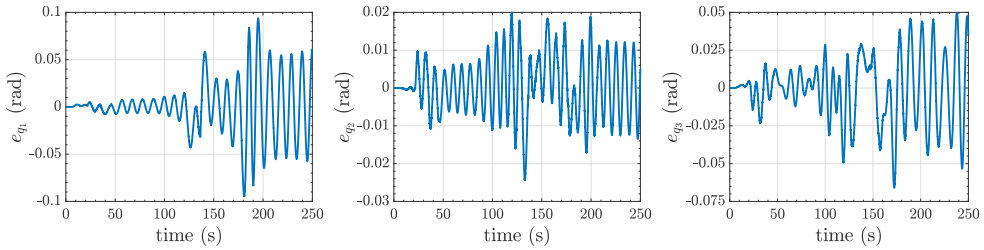
(a) (b) (c)

Figure C.52: PI Joint Position



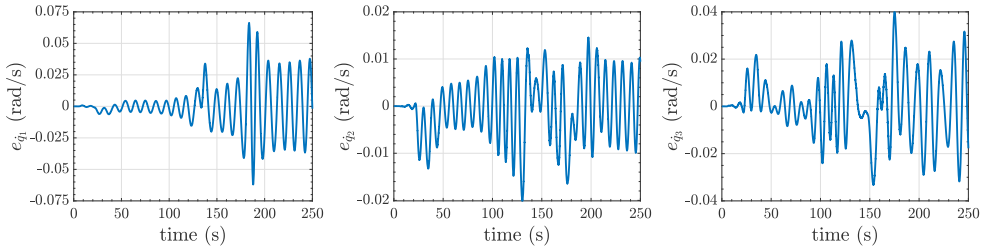
(a) (b) (c)

Figure C.53: PI Joint Velocity



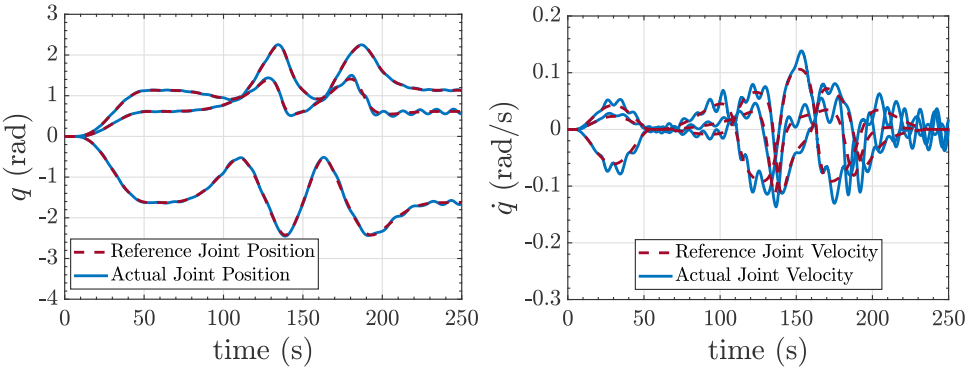
(a) (b) (c)

Figure C.54: PI Joint Position Error



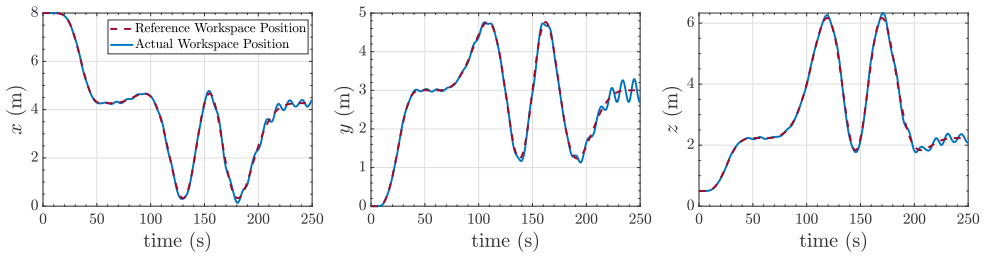
(a) (b) (c)

Figure C.55: PI Joint Velocity Error



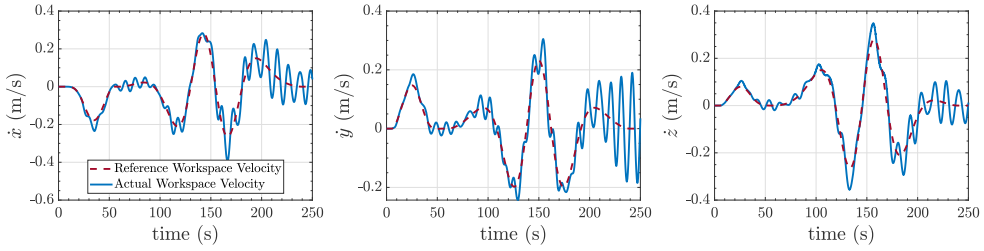
(a) (b)

Figure C.56: PI Joint Position and Velocity



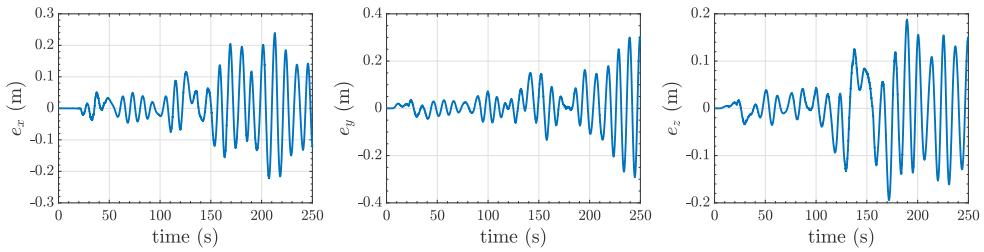
(a) (b) (c)

Figure C.57: PI Operational Space Position



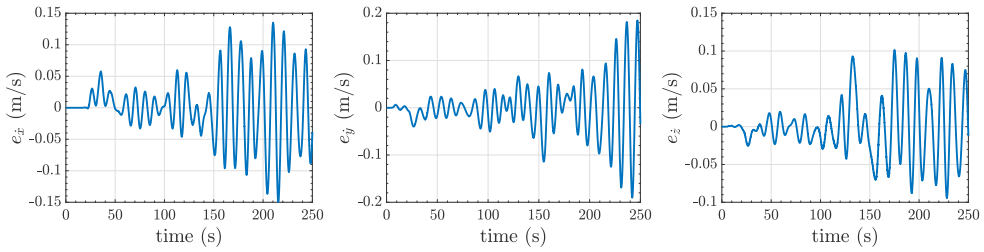
(a) (b) (c)

Figure C.58: PI Operational Space Velocity



(a) (b) (c)

Figure C.59: PI Operational Space Position Error



(a) (b) (c)

Figure C.60: PI Operational Space Velocity Error

Bibliography

[1] European Commission. *Communication From the Commission to the European Parliament, the Council, the European Economic and Social Committee and the Committee of the Regions Eu Space Industrial Policy Releasing the Potential for Economic Growth in the Space Sector*. 0108 final. COM, Brussels, 2013.

[2] Kazuya Yoshida, Brian Wilcox, Gerd Hirzinger, and Roberto Lampariello. Space robotics. In *Springer Handbook of Robotics*, pages 1423–1462. Springer, 2016.

[3] Yang Gao and Steve Chien. Review on space robotics: Toward top-level science through space exploration. *Science Robotics*, 2(7), 2017.

[4] Syed Ali Ajwad, Jamshed Iqbal, Muhammad Imran Ullah, and Adeel Mehmood. A systematic review of current and emergent manipulator control approaches. *Frontiers of mechanical engineering*, 10(2):198–210, 2015.

[5] Dan Zhang and Bin Wei. A review on model reference adaptive control of robotic manipulators. *Annual Reviews in Control*, 43:188–198, 2017.

[6] Umberto Montanaro. Multi-input model reference adaptive control via the feedback of the reference state and minimal control synthesis. *Automation*, Under Revision.

[7] Nhan T Nguyen. Model-reference adaptive control. In *Model-Reference Adaptive Control*, pages 83–123. Springer, 2018.

[8] Mario Di Bernardo, Alessandro Di Gaeta, Umberto Montanaro, and Stefania Santini. Synthesis and experimental validation of the novel lq-nemcsi adaptive strategy on an electronic throttle valve. *IEEE Transactions on Control Systems Technology*, 18(6):1325–1337, 2010.

[9] Annamaria Buonomano, Umberto Montanaro, Adolfo Palombo, and Stefania Santini. Building temperature control using an enhanced mrac approach. In *2015 European Control Conference (ECC)*, pages 3629–3634. IEEE, 2015.

[10] Umberto Montanaro, Alessandro di Gaeta, and Veniero Giglio. An mrac approach for tracking and ripple attenuation of the common rail pressure for gdi engines. In *Proceedings of the 18th IFAC World Congress*, pages 4173–4180, 2011.

[11] Annamaria Buonomano, Umberto Montanaro, Adolfo Palombo, and Stefania Santini. Dynamic building energy performance analysis: A new adaptive control strategy for stringent thermohygrometric indoor air requirements. *Applied Energy*, 163:361–386, 2016.

[12] Umberto Montanaro and Josep M Olm. Integral mrac with minimal controller synthesis and bounded adaptive gains: The continuous-time case. *Journal of the Franklin Institute*, 353(18):5040–5067, 2016.

[13] Annamaria Buonomano, Umberto Montanaro, Adolfo Palombo, and Stefania Santini. Temperature and humidity adaptive control in multi-enclosed thermal zones under unexpected external disturbances. *Energy and Buildings*, 135:263–285, 2017.

[14] ROS.org. *Introduction*, 2018. `http://wiki.ros.org/ROS/Introduction`.

[15] ROS.org. *Core Components*. `https://www.ros.org/core-components/`.

[16] ROS.org. *About ROS*. `https://www.ros.org/about-ros/`.

[17] ROS.org. *ROS Brand Guidelines*. `https://www.ros.org/press-kit/`.

[18] ROS.org. *Integration with Other Libraries*. `https://www.ros.org/integration/`.

[19] gazebosim.org. *Why Gazebo?* `http://gazebosim.org/`.

[20] gazebosim.org. *Logos*, 2014. `http://gazebosim.org/media#logos`.

[21] MathWorks. *What is MATLAB?* `https://it.mathworks.com/discovery/what-is-matlab.html`.

[22] MathWorks. *Simulink*. `https://it.mathworks.com/help/simulink/`.

[23] MathWorks. *Publish*. https://www.mathworks.com/help/ros/ref/publish.html.

[24] MathWorks. *Subscribe*. https://www.mathworks.com/help/ros/ref/subscribe.html.

[25] ROS.org. *Understanding ROS Topics*, 2019. http://wiki.ros.org/ROS/Tutorials/UnderstandingTopics.

[26] bduffany. *Writing a new controller*, 2014. https://github.com/ros-controls/ros_control/wiki/controller_interface.

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

[28] Canadian Space Agency. *Canadarm, Canadarm2, and Canadarm3 – A comparative table*, 2020. https://asc-csa.gc.ca/eng/iss/canadarm2/canadarm-canadarm2-canadarm3-comparative-table.asp.

[29] ROS.org. *urdf*, 2019. http://wiki.ros.org/urdf.

[30] ROS.org. *xacro*, 2020. http://wiki.ros.org/xacro.

[31] ROS.org. *Using Xacro to Clean Up a URDF File*, 2018. http://wiki.ros.org/urdf/Tutorials/Using%20Xacro%20to%20Clean%20Up%20a%20URDF%20File.

[32] gazebosim.org. *Tutorial: Using a URDF in Gazebo*, 2014. http://gazebosim.org/tutorials?tut=ros_urdf.

[33] ROS.org. *Building a Visual Robot Model with URDF from Scratch*, 2019. http://wiki.ros.org/urdf/Tutorials/Building%20a%20Visual%20Robot%20Model%20with%20URDF%20from%20Scratch.

[34] gazebosim.org. *Tutorial: ROS Control*, 2014. http://gazebosim.org/tutorials/?tut=ros_control.

[35] gazebosim.org. *Tutorial: ROS Communication*, 2014. http://gazebosim.org/tutorials/?tut=ros_comm.

[36] gazebosim.org. *Tutorial: Using Gazebo plugins with ROS*, 2014. http://gazebosim.org/tutorials?tut=ros_gzplugins.

[37] MathWorks. *URDF Import*, 2020. https://it.mathworks.com/help/physmod/sm/ug/urdf-import.html#:~:text=Imported%20CAD%20models%20have%20their,a%20separate%20MATLAB%20data%20file.

[38] Zhengyuan Wang, Umberto Montanaro, Saber Fallah, Aldo Sorniotti, and Basilio Lenzo. A gain scheduled robust linear quadratic regulator for vehicle direct yaw moment control. *Mechatronics*, 51:31–45, 2018.

[39] Yoji Umetani, Kazuya Yoshida, et al. Resolved motion rate control of space manipulators with generalized jacobian matrix. *IEEE Transactions on robotics and automation*, 5(3):303–314, 1989.