Workspace Control of Free-Floating Space Manipulators for Implementation in a Novel On-Orbit Servicing Mission Architecture

by

Patrick Rousso, **B.Eng.**

A thesis submitted to the
Faculty of Graduate and Postdoctoral Affairs
in partial fulfillment of the requirements for the degree of

Master of Applied Science in Aerospace Engineering

Ottawa-Carleton Institute for Mechanical and Aerospace Engineering
Department of Mechanical and Aerospace Engineering
Carleton University
Ottawa, Ontario
April, 2022

The undersigned hereby recommends to the
Faculty of Graduate and Postdoctoral Affairs
acceptance of the thesis

Workspace Control of Free-Floating Space Manipulators for Implementation in a Novel On-Orbit Servicing Mission Architecture

submitted by **Patrick Rousso**, **B.Eng.**

in partial fulfillment of the requirements for the degree of

Master of Applied Science in Aerospace Engineering

Professor Dan Necsulescu, External Examiner

Professor Alex Ellery, Internal Examiner

Professor Robin Chhabra, Thesis Supervisor

Professor Henry Saari, Chair,
Department of Mechanical and Aerospace Engineering

Ottawa-Carleton Institute for Mechanical and Aerospace Engineering
Department of Mechanical and Aerospace Engineering
Carleton University
January, 2022

Abstract

An industrialized On-Orbit Servicing (OOS) mission architecture is proposed in this thesis which signifies the role of space manipulators in performing OOS missions. This mission architecture summarizes a procedure for OOS mission design based on a series of multi-objective optimization problems. To enhance proximity operations, it is concluded that the existence of an output tracking control system for space manipulators is beneficial. This thesis develops an output tracking controller to control the end-effector pose of an N-link free-floating space manipulator, with non-zero momentum. We employ feedback linearization on the Lie algebra of the Special Euclidean group $\mathrm{SE}(3)$ to remove the nonlinear influence from the system's dynamics and momentum in the controlled end-effector motion. Space manipulators are modelled as open-chain rigid multi-body systems, connected by either single or multi degree of freedom joints. The dynamics and kinematics of free-floating space manipulators are formulated on $\mathrm{SE}(3)$ using the exponential parameterization of screw motions. Multi-degree of freedom joints are modelled as the combination of the joints' individual exponential screw motion mappings to describe their motions as homogeneous transformations. The free-floating systems' equations of motion are obtained through an Euler-Lagrange derivation and decoupled into the base and manipulator motions. The conservation of linear and angular momentum is then considered as an affine nonholonomic constraint to the system, allowing for the dynamic reduction of a space manipulator system with conserved non-zero momentum. Due to the system's free-floating nature, the equations of motion are restricted to the manipulator's joint space which defines the region of controlled states.

Two control cases are considered in this thesis, the first involving control over the output's linear motion and the second considering control over the end-effector pose. The first control case performs feedback linearization on $\mathbb{R}^3$ where the resulting linear end-effector motion is controlled by a classical PID controller defined to impart a critically damped response in the error dynamics. For full pose control, feedback linearization is employed on $\mathfrak{se}(3)$ for subsequent control using a modified feedforward, feedback PID control structure

involving a coordinate-free pose error function. An associated transfer map is used to define the corresponding velocity error on the Lie algebra $\mathfrak{se}(3)$. Using a Lyapunov candidate predicated on the total energy of the error dynamics, the developed full pose output tracking controller is proven to stabilize the end-effector pose to a feasible desired trajectory. The singularity-robust inverse derived from the damped least squares method is implemented in both the linear and full pose workspace controllers to avoid impractical joint torques in the region of kinematic/dynamic singularities.

A free-floating space manipulator simulation platform is developed in Python to analyze the performance of the linear and full pose output tracking controllers. The simulation is carefully structured to maximize computational efficiency such that a fast model propagation is achieved. A validation procedure is performed using an equivalent space manipulator model in Simscape to confirm the accuracy of the implemented kinematics and dynamics models in Python. In both control cases, the output responses are observed under trapezoidal trajectories involving systems containing zero and non-zero momentum. In the case of strictly linear output control, both a circular trajectory and a trajectory approaching a singular configuration are tested to demonstrate the controlled system's dexterity and singularity accommodation ability, respectively. Model uncertainty is also introduced in the space manipulator's mass to assess the controllers' robustness. All testing scenarios demonstrate accurate trajectory followings, and successful mitigation of large control torques in the neighbourhood of singularities.

To my nonna, Elisabetta, and grandfather, Léon, whom I will always remember with love.

Acknowledgments

I first and foremost would like to extend my significant gratitude to my supervisor Dr. Robin Chhabra for providing me with constant support and the wonderful research experience. Aside from the tremendous technical insight and knowledge you have shared with me, you have also been a figure of compassion, perseverance and professionalism for which I very much appreciate. It has been a pleasure to conduct my research under your supervision. I would also like to acknowledge my colleagues in the ASRoM lab who have given their support and feedback in my research and have shared their friendship.

I cannot describe the appreciation and love I have for my mom, Maria, my dad, Gilles, and my brother, Chris. To my parents in particular, thank you for your unconditional love, for your constant support, and for always showing up with more lasagna and cotoletta than I could possibly imagine! Chris, thank you for always being there for me, and knowing the perfect time to make me crack up. You have helped more than you know. To my grandparents, aunts, uncles, and cousins, thank you for your genuine interest in my research and for always bringing me joy when we talk!

Lastly, I would like to thank my friends for their continued support. All of the pub nights, hangouts, debates, laughs, and great memories we shared have always brought a smile to my face when I needed it most! In particular, I would like to thank my roommates for being the best stress relievers I could ever ask for. Knowing you guys always have my back is a great comfort.

Table of Contents

Abstract iii

Acknowledgments vi

Table of Contents vii

List of Tables x

List of Figures xi

1 Introduction 1

1.1 Motivation 1

1.2 Thesis Objectives 2

1.3 Statement of Contributions 3

1.4 Thesis Organization 4

2 Literature Review 5

2.1 On-Orbit Servicing 5

2.2 GNC of Single-Arm Free-Floating Space Manipulators 7

3 Mission Architecture for On-Orbit Servicing 11

3.1 Summary 11

3.2 Proposed Mission Architecture 12

3.3 Workspace Control 15

4 Kinematics and Dynamics of Space Manipulators 17

4.1 Summary 17

4.2 Space Manipulator System Kinematics 17

4.2.1 Multi-DOF Joint Parameterization 18

4.2.2 Forward Kinematics . . . 20
4.2.3 Differential Kinematics . . . 21
4.3 Space Manipulator System Dynamics . . . 24
4.4 Conservation of Momentum . . . 31

5 Output Tracking Control **35**
5.1 Summary . . . 35
5.2 Dynamic Reduction . . . 36
5.3 Feedback Linearization for Linear Output Control . . . 40
5.3.1 Zero Momentum System . . . 40
5.3.2 Non-Zero Momentum System . . . 45
5.4 Feedback Linearization for Full Pose Control . . . 46
5.5 Full Pose Control Structure . . . 49
5.5.1 Error Function . . . 49
5.5.2 Velocity Error . . . 51
5.5.3 Output Tracking Control Design . . . 52
5.6 Output Tracking Controller Block Diagram . . . 56

6 Space Manipulator Simulation Platform **60**
6.1 Summary . . . 60
6.2 Simulation Design . . . 61
6.2.1 Forward Kinematics . . . 64
6.2.2 Differential Kinematics . . . 65
6.2.3 Inertia Matrix . . . 67
6.2.4 Coriolis Matrix . . . 69
6.2.5 Dynamics Reduction . . . 74
6.2.6 Output Tracking Controller . . . 76
6.2.7 Motion Propagation . . . 78
6.3 Multi-Body System Description . . . 80
6.4 Simulation Validation . . . 84
6.4.1 Simscape Model . . . 85
6.4.2 Validation Procedure . . . 91

7 Simulation Results **96**
7.1 Summary . . . 96

7.2 Linear Output Tracking Controller Results 97
7.2.1 Space Manipulator System with Zero Momentum 97
7.2.2 Space Manipulator System with Non-Zero Momentum 107
7.3 Full Pose Output Tracking Controller Results 115
7.3.1 Space Manipulator System with Zero Momentum 115
7.3.2 Space Manipulator System with Non-Zero Momentum 122

8 Conclusion and Future Work **130**

A Mathematical Preliminaries **134**
A.1 Rigid Body Motion in $\mathbb{R}^3$ 134
A.1.1 Rotational and Translational Motion 134
A.1.2 Homogeneous Transformations 136
A.1.3 Rigid Body Velocity 142

B PID Control Design **146**

C Singularity Accommodation **149**

D Homogeneous Transformation Partial Derivative **153**

E Adjoint Operator Partial Derivatives **154**

F Testing Cases in Validation Procedure **156**

G Base Positions for Linear Trajectory Following Scenarios **159**

H Base Positions for Full Pose Trajectory Following Scenarios **164**

List of References **164**

List of Tables

4.1 Lower pair joints 19
6.1 Physical Properties of Simulated System 82
7.1 Mass Values used in the True Space Manipulator Model 106
F.1 Space Manipulator Configurations Tested 156
F.2 System of Generalized Velocities Tested 157
F.3 Force and Torque Vectors Tested 158

List of Figures

3.1 OOS industrialization flow chart. 12
5.1 Output tracking controller block diagram 57
6.1 Space manipulator simulation block diagram 62
6.2 Simulated free-floating space manipulator system 81
6.3 2-Link space manipulator Simscape model. 86
6.4 Base motion subsystem components 87
6.5 Shoulder joint subsystem components 88
6.6 Elbow joint subsystem componentss 88
6.7 Wrist joint subsystem components 89
6.8 Joint actuation example for shoulder joint subsystem 90
6.9 Sensing of the space manipulator system 90
6.10 3-Dimensional space manipulator model in Simscape 91
7.1 Control torques for the linear trapezoidal trajectory applied to a zero momentum system. 98
7.2 Output response to the linear trapezoidal trajectory applied to a zero momentum system. 99
7.3 Control torques for the linear circular trajectory applied to a zero momentum system. 101
7.4 Output response to the linear circular trajectory applied to a zero momentum system. 102
7.5 Damped least squares parameters for SR-inverse in the linear controller with associated control input 104
7.6 Linear output tracking performance in the presence of a singularity 104
7.7 Control torques for the linear trapezoidal trajectory applied to a zero momentum system with uncertainties. 107
7.8 Output response to the linear trapezoidal trajectory applied to a zero momentum system with uncertainties. 108

7.9 Control torques for the linear trapezoidal trajectory applied to a system with non-zero momentum. 109
7.10 Output response to the linear trapezoidal trajectory applied to a system with non-zero momentum. 110
7.11 Control torques for the linear circular trajectory applied to a system with non-zero momentum. 111
7.12 Output response to the linear circular trajectory applied to a system with non-zero momentum. 112
7.13 Control torques for the linear trapezoidal trajectory applied to an uncertain system with non-zero momentum. 113
7.14 Output response to the linear trapezoidal trajectory applied to an uncertain system with non-zero momentum. 114
7.15 Control torques for the full pose trapezoidal trajectory applied to a zero momentum system. 116
7.16 Output linear response to the full pose trapezoidal trajectory applied to a zero momentum system. 117
7.17 Output angular response to the full pose trapezoidal trajectory applied to a zero momentum system. 118
7.18 Control torques for the full pose trapezoidal trajectory applied to a zero momentum system with uncertainties. 119
7.19 Output linear response to the full pose trapezoidal trajectory applied to a zero momentum system with uncertainties. 120
7.20 Output angular response to the full pose trapezoidal trajectory applied to a zero momentum system with uncertainties. 121
7.21 Control torques for the full pose trapezoidal trajectory applied to a system with non-zero momentum. 123
7.22 Output linear response to the full pose trapezoidal trajectory applied to a system with non-zero momentum. 124
7.23 Output angular response to the full pose trapezoidal trajectory applied to an system with non-zero momentum. 125
7.24 Control torques for the full pose trapezoidal trajectory applied to an uncertain system with non-zero momentum. 126
7.25 Output linear response to the full pose trapezoidal trajectory applied to an uncertain non-zero momentum. 127

7.26 Output angular response to the full pose trapezoidal trajectory applied to an uncertain system with non-zero momentum. 128
G.1 Base positions for the linear trapezoidal trajectory applied to a zero momentum system. 160
G.2 Base positions for the linear circular trajectory applied to a zero momentum system. 160
G.3 Base positions for the trajectory in the presence of a singularity. 161
G.4 Base positions for the linear trapezoidal trajectory applied to a zero momentum system with uncertainties. 161
G.5 Base positions for the linear trapezoidal trajectory applied to a non-zero momentum system. 162
G.6 Base positions for the linear circular trajectory applied to a non-zero momentum system. 162
G.7 Base positions for the linear trapezoidal trajectory applied to a non-zero momentum system with uncertainties. 163
H.1 Base positions for the full pose trapezoidal trajectory applied to a zero momentum system. 165
H.2 Base positions for the full pose trapezoidal trajectory applied to a zero momentum system with uncertainties. 165
H.3 Base positions for the full pose trapezoidal trajectory applied to a non-zero momentum system. 166
H.4 Base positions for the full pose trapezoidal trajectory applied to a non-zero momentum system with uncertainties. 166

Chapter 1

Introduction

1.1 Motivation

As society's dependence on space infrastructures continues its increasing trend, extending satellites' lifespans, technological capabilities, and commercial accessibility become increasingly important. Due to the limited feasibility of manned operations in space, On-Orbit Servicing (OOS) plays a vital role in addressing the key problems (e.g., premature satellite failure and orbital debris) faced by the space industry. Some of the main OOS missions include: satellite visual inspection, refuelling, repairing or exchanging damaged components, assisting in orbital manoeuvres, in-orbit assembly of spacecraft, and orbital debris removal [1, 2, 3, 4]; all of which can be performed by single-arm space manipulators. To reduce fuel consumption during such servicing missions, space manipulators may operate in their free-floating regime where the spacecraft's base is uncontrolled (i.e., the base may rotate and translate freely in all directions). Due to the complexities involved with the Guidance, Navigation, and Control (GNC) of free-floating manipulators, enhancing their functionality during the pre-capture phase of close proximity operations is crucial for safe and successful capture and post-capture manoeuvres. Note that considerations on the capture and post-capture operation enhancements are beyond the scope of this research. Machine learning techniques, particularly continuous unsupervised learning strategies, prove beneficial in developing trajectory planners for a free-floating system to capture a moving target satellite [5]. To maximize the performance of the learning algorithms (with regards to the convergence and success rates), as well as improve the versatility and adaptability of the learned policy, implementing and tracking workspace trajectories is favourable. Such trajectories must define both the position and orientation of a servicer's end-effector with respect to the grasping point on the target satellite. As a result, effective

control in the workspace of a space manipulator in its free-floating regime is required to accurately follow such desired trajectories output by the trajectory planner. Control designs which include singularity accommodation and yield sufficiently fast and precise tracking are necessary for achieving a greater performance.

1.2 Thesis Objectives

This work studies the geometric modelling and workspace control of single-arm free-floating space manipulators to complement an industrialized OOS mission architecture. The objectives of this thesis are to:

1. Study an industrialized OOS mission architecture involving multiple servicing satellites (servicers) and multiple target satellites (targets) to identify the mission requirements for space manipulator systems. This investigation signifies the key areas of enhancement essential for servicers' GNC systems used in proximity operations.

2. Derive the kinematics and dynamics of free-floating space manipulators on the Special Euclidean Lie group $\mathrm{SE}(3)$, considering space manipulator designs which include multi-Degree Of Freedom (DOF) joints. The multi-DOF joints are to be defined by the exponential parameterization of screw motions for implementation in the system's kinematics mappings and equations of motion.

3. Perform workspace control on an unperturbed free-floating space manipulator with conserved non-zero momentum. The developed control law should be designed on $\mathrm{SE}(3)$ to stabilize the position and orientation (pose) of an end-effector to a feasible desired trajectory. Viable joint torques must be maintained in the region of singular configurations to ensure structural safety during proximity operations.

4. Develop a high fidelity free-floating space manipulator simulation platform to assess the performance of GNC systems including trajectory planners and control designs. Considerations on computational efficiency should be implemented to achieve a fast model propagation for the simulated space manipulator. The developed simulator should also accommodate machine learning techniques with the objective of improving GNC capabilities.

1.3 Statement of Contributions

The main contributions of the thesis can be summarized in the following.

1. The kinematics and dynamics of free-floating space manipulators are fully formulated on Lie groups with the inclusion of multi-DOF joints. Multi-DOF joints are modelled as the amalgamation of the joint's individual screw motions. The product of exponentials formula, originally proposed for fixed-base manipulators with single-DOF joints, is systematically extended to capture the kinematics and dynamics of free-floating space manipulators with multi-DOF joints. This novel extension is then used to rigorously reduce the equations of motion of space manipulators with conserved non-zero momentum.

    - P. Rousso, R. Chhabra, "Output Tracking Control of Space Manipulators with Non-Zero Momentum with Singularity Accommodation," *Nonlinear Dynamics*, To be submitted in May 2022.

2. A modified feedforward, feedback PID control law predicated on coordinate-free pose and velocity error functions is developed to achieve Lyapunov stable trajectory following in the output pose of an unperturbed free-floating space manipulator with conserved non-zero momentum. Using the reduced dynamics, input-output linearization on the Lie algebra $\mathfrak{se}(3)$ is employed. The developed output tracking control structure implements the singularity-robust inverse method to avoid kinematic/dynamic singularities of the space manipulator system. A conference paper on this research, excluding the singularity avoidance component, has been accepted by the American Control Conference 2022.

    - P. Rousso, R. Chhabra, "Workspace Control of Free-Floating Space Manipulators with Non-Zero Momentum on Lie-Groups," *American Control Conference*, 2022.
    - P. Rousso, R. Chhabra, "Output Tracking Control of Space Manipulators with Non-Zero Momentum with Singularity Accommodation", *Nonlinear Dynamics*, To be submitted in May 2022.

3. An overall mission architecture for performing multiple on-orbit servicing missions in low Earth orbit by a fleet of servicers in the form of free-flying single-arm space manipulators is proposed. The primary mission resources are identified as the fuel

and time consumed during a servicing mission. The proposed architecture considers mission designs which seek the optimal reduction of mission duration and fuel consumption. Servicer satellites are additionally suggested to perform in-orbit machine learning to improve the functionality of their guidance, navigation and control systems in upcoming missions. This research was published in the 2021 IEEE Aerospace Conference.

- P. Rousso, S. Samsam, R. Chhabra, "A Mission Architecture for On-Orbit Servicing Industrialization," *IEEE Aerospace Conference*, 2021.

1.4 Thesis Organization

The following sections of the thesis are organized as follows: chapter 2 provides a literature review on OOS mission scheduling and design, dynamics and kinematics formulations of rigid multi-body systems, and control of free-floating space manipulators.

Chapter 3 proposes an end-to-end mission architecture for OOS industrialization, covering various mission phases of on-orbit servicing and their inter-relations. This chapter suggests a beneficial parking orbit placement, develops mission planning optimization problems, and discusses proximity operation performance enhancements.

In Chapter 4, the kinematics, dynamics and conservation of momentum for free-floating space manipulator systems are derived on Lie groups. Chapter 5 reduces the space manipulator's system dynamics and performs feedback linearization on the end-effector motion. Control over the output's linear motion and the end-effector's pose are considered separately. The developed full pose workspace controller is analyzed for Lyapunov stability, and the singularity-robust inverse method for singularity accommodation is addressed in the controller's design.

Chapter 6 provides details on the structure and functionality of the free-floating space manipulator simulation platform developed in Python. A validation procedure is additionally presented to verify the simulator's performance.

Chapter 7 contains the results of the developed linear and full pose output tracking controllers under various trajectory following tasks. Finally, Chapter 8 provides some concluding remarks and potential future works to complement this research.

Chapter 2

Literature Review

2.1 On-Orbit Servicing

On-orbiting servicing has been an increasingly attractive topic of research with the rise of technological advancements and capabilities of current and prospective satellite missions. OOS missions supply a presently vacant service to the space industry, consequently allowing for in-orbit operations on a selected target spacecrafts to be performed using robotic systems [6, 7]. Some of the key problems faced by the modern space industry include orbital debris, and the significant build and launch costs associated with satellite missions (approximately $100-600 million and $40-$400 million, respectively). There are currently over 17,000 space objects larger than 10 cm, with just over 13,000 of such objects being classified as space debris (non-operational objects in orbit) [8] . If currently left untouched (without adding new satellites or removing debris) the Kessler Syndrome will result in key orbital regions (in particular Low Earth Orbit (LEO)) to become uninhabitable due to the excessive amount of space debris [9, 10, 11]. Given the quantity of orbital debris, NASA predicts that approximately 5 to 10 debris objects must be removed annually to allow for an accessible orbital environment [12]. Space manipulator systems possess great allure for performing servicing missions due to their versatility in safely capturing both operational and non-operational targets. Currently, several OOS servicing missions have been proposed in both Geostationary Earth Orbits (GEO) and LEO, addressing both single-target and multi-target servicing missions. For example, [13] designs a mission for a single servicing satellite to provide GEO satellites with OOS. Included in the mission design are considerations on the mission sequencing and the transfer trajectory for maximizing the revenue from fees and minimizing revenue penalties for late service. Additional servicing mission designs specifically for GEO satellites have been analyzed in [14, 15, 16] due to

the popularity of this orbital region. Moreover, [17] considers a single servicing satellite performing a multi-target refuelling mission in LEO, where all target satellites belong to the same circular orbit. The servicer meets and refuels target satellites at designated service positions within the orbit. Orbital debris removal missions have also been designed for the LEO [18, 19, 20], considering a single servicing satellite de-orbiting multiple debris objects over the course of the mission.

Transfer orbit trajectories between servicing and target satellites are designed in OOS missions to transfer servicers to within a range of approximately 10 km from the targets, in the targets' orbits [21]. This long-range rendezvous phase of OOS missions has been subject to considerable research with regards to optimization. Studies have been conducted to improve OOS mission designs by reducing the fuel consumption and mission time necessary for performing the required orbit transfers. Several survey papers address such transfer trajectory optimization problems from different perspectives [22, 23, 24]. In the latest survey, Shirazi *et al.* reviewed the models, objective functions, approaches, and solutions to the spacecraft trajectory optimization problem, proposed prior to 2018 [22]. Modeling of a spacecraft can be studied based on the transfer type, i.e., impulsive or continuous, and equations of motion, i.e., two-body or N-body problem. The two-impulse Lambert [25, 26] and Hohmann [27] approaches are the traditional transfer trajectory generation methods that were used frequently in the literature. Shape-based approaches are the other popular trajectory design methods that are applicable in both continuous and impulsive transfer types [28]. The popularity of these approaches is because they reduce the computational costs by imposing geometric constraints on the trajectories [29, 30]. The solution to the trajectory optimization problem can be achieved from two different approaches, i.e., analytical or numerical. The analytical approach gives an exact solution for the simple spacecraft trajectory optimization problem, such as in Hohmann and Lambert methods. However, for an advanced transfer trajectory optimization problem with a complex model and objective functions, the numerical solutions, including direct and indirect methods, are promising approaches. Betts [23] and Conway [31] provide an overview of direct and indirect numerical methods in their survey papers.

For OOS missions involving multiple servicing satellites and multiple targets, mission scheduling refers to identifying the sequence in which servicers transfer to target satellites during the servicing mission. Generally speaking, scheduling optimization problems have been well researched in various domains of the literature. From a perspective outside of OOS missions, the dispatch scheduling problem can be treated as a Travelling Salesman

Problem (TSP) [32]. Analogous to the TSP, a servicing mission requires servicers to travel to all target satellites without any repeated visits. Given the similarity between the mission scheduling task and the TSP, Gurtuna and Trepanier have applied TSP algorithms (first used for ground-based vehicle routing optimization) to the scheduling of OOS missions [33]. They showed that the simpler TSP algorithms can be easily adjusted for use in dispatch scheduling. However, for more advanced algorithms (such as tabu search), the solution and implementation are complex, but indicate great potential for use in the mission scheduling optimization problem. Moreover, the studies in [34, 35, 36, 37] all involve the analysis of a single servicing satellite travelling to a set of target satellites, where the optimizations of the servicing sequences are considered. Specifically regarding a mission including multiple servicers, [3] examines the mission scheduling required for 2 servicing satellites and 3 target satellites. Here, the optimal target sequence is predicated on minimizing the time of flight and fuel consumption required for the mission. The particle swarm optimization method is used in conjunction with the Taguchi technique in [3] to achieve a robust solution to the optimization problem. As demonstrated, limited research currently exists in the OOS mission scheduling task including multiple servicers and multiple targets.

2.2 GNC of Single-Arm Free-Floating Space Manipulators

Due to the versatility of space manipulators in performing OOS missions, research on their GNC systems (in both their free-flying and free-floating regimes) has been a highly attractive and motivated topic in space robotics. Free-flying space manipulators involve active control over the system's base throughout manipulation of the robotic arm [38], whereas free-floating manipulators allow the base to move freely during manipulator operations [39]. Operating space manipulators with an unactuated base comes with the advantage of reducing the fuel and energy required during target capture manoeuvres [40], as well as the complexities associated with the GNC designs of under-actuated systems [41]. In this case, the coupled dynamics between the spacecraft and manipulator motions and the mitigation of the uncontrolled base motion are key considerations in the design of free-floating GNC systems [42, 43]. During capture manoeuvres, it is widely suggested for a spacecraft's control system to be removed at the moment of contact between the end-effector and the target [44]. This is necessary for mitigating the potentially damaging behaviours

which may be performed by the controlled servicer. Consequently, free-floating scenarios of space manipulators during their proximity operations must be considered. Despite the GNC difficulties associated with free-floating systems, accurate trajectory planning and stable trajectory following are necessary for in-orbit operations.

Prior to analyzing and developing appropriate GNC strategies for free-floating space manipulators, their kinematics and dynamics models must be established. This thesis considers free-floating space manipulators as open-chain multi-body systems. It is strategic to develop their corresponding dynamics and kinematics models using techniques which are simple and geometrically representative of the physical system. More specifically, in the case of a single-arm space manipulator, these systems are considered to consist of a chain of several rigid links connected by ideal joints. For general rigid multi-body systems, [45] and [46] have taken a geometric approach by developing kinematics and dynamics formulations of multi-body systems based on Lie groups. Lie group formulations do not require the parameterization of multi-body systems in their dynamics and kinematics modellings, and additionally reference the system's physical geometry and structure. The relationship between Lie groups and screw theory is analyzed in [47], where screws are presented as elements of Lie algebras, allowing for a geometric interpretation of rigid body motions. Provided the connection between screw motions and Lie group theory, kinematic mappings from the joint space to the task space of a serial chain manipulator can be formulated [48, 49, 50]. In establishing a space manipulator's equations of motion from a geometric background, Hamiltonian and Lagrangian approaches prove beneficial due to their accommodation to Lie group formulations [50, 51, 52]. Note that due to the low-gravity orbital environment, space manipulators are often considered to possess negligible potential energy. The presence of external disturbances are often additionally presumed to be insignificant during the small time period in which space manipulators perform servicing operations, thus implying an assumption of conserved momentum in the system.

Trajectory designs involved with capture manoeuvres (including either free-flying or free-floating systems) primarily focus on defining an end-effector's linear and angular velocity throughout the pre-capture operation, and at the grasping interface [44]. One of the primary issues in executing trajectories in a free-floating manipulator's workspace is the consequent non-zero motion of the base (due to its uncontrolled nature), causing the base's position and orientation at the grasping point not to be unique (i.e., different workspace trajectories to a target point result in different trajectories of the base) [53]. In analyzing this

issue, the authors in [54, 55] have formulated optimization techniques for designing trajectories which consider external forces and non-zero velocities in the space manipulator's base. Provided an appropriate trajectory for a free-floating system, the control to achieve accurate tracking of the desired path introduces additional complexities in the GNC design. Common control strategies consider reducing the coupled base motion during manipulator operation [42, 56]. Adaptive control techniques are additionally considered, where adaptive controllers allow for their control structures to change during operation in order to minimize error by adjusting the system's response. These control schemes are particularly adequate for handling uncertainties in both the environment and the controlled system's dynamics model. For example, Wang develops a stable generalized parameter adaptation law based on a generalized dynamics regressor to investigate the adaptive inverse dynamics of free-floating space manipulators with parameter uncertainties [57]. Ulrich and Sasiadek demonstrate the efficacy (in regards to the trajectory following performance) of introducing feedforward parameters in an adaptive control structure on space manipulators with model uncertainties [58].

There is an additional category of free-floating space manipulator control strategies, predicated on the Virtual Manipulator (VM) technique introduce by Vafa and Dubowsky in [59]. The VM method interprets free-floating space manipulators by a dynamically equivalent fixed-based manipulator system. The base of the VM (referred to as the Virtual Ground (VG)) is located at the center of mass of the space manipulator system, and is thus fixed provided no external disturbances (zero momentum). In this approach, the dynamics and kinematics of free-floating systems can be interpreted through the behaviour of the more trivial fixed-base VM, for which associated controllers can be developed [60, 61]. In relation to the aforementioned adaptive control methods, Parlaktuna et al. use the notion of the VM to develop a dynamically equivalent model of a free-floating space manipulator to perform adaptive control in the system's joint space [62, 63]. Furthermore, Torres and Dubowsky study path planning and control structures based on the VM approach, which considers minimizing the disturbances on the spacecraft's base. In [64], they develop a method to compute disturbance maps for manipulators with more than 2 degrees of freedom, and additionally demonstrate the use of the manipulator's redundancy in eliminating the disturbances on the base [65]. Dubowsky and Torres introduce the notion of coupling maps in [66] to reduce the effects of the coupled dynamics between the base and manipulator motions, and develop trajectory and control designs (which avoid areas with large disturbance to the base) for the free-floating manipulator system.

A separate foundational approach for controlling free-floating space manipulators is introduced by Yoshida and Umetani in [67], using the notion of the Generalized Jacobian Matrix (GJM) from [68]. In this approach, a differential kinematics mapping from the manipulator joint velocities to the end-effector motion is established, independent of the uncontrolled base motion. The GJM makes use of the conserved momentum in the free-floating space manipulator system to create a relationship between the base and manipulator velocities. Such method allowed for Nenchev and Yoshida to introduce the bias momentum approach in [69] which considers impedance control and the minimization of changes in the base's orientation during the pre-capture of a tumbling target. The Reaction Null-Space (RNS) is developed using the GJM method to identify the motions in the manipulator's joint space which impart the lowest disturbance to the base [70]. Moreover, the GJM approach allows for the implementation of control techniques in the workspace of free-floating space manipulators. This is due to the GJM providing a relationship between the manipulator joints (where control actions are performed) and the end-effector motion (typically where trajectories are designed for target capture missions). Using the GJM concept, feedback linearization techniques allow for accomplishing such control tasks by effectively canceling systems' nonlinearities and enabling the use of linear controllers in the workspace. In [71] a method for feedback linearization of constrained multi-body systems with non-zero momentum is developed. Moreover, [72, 73] feedback linearize space manipulators to perform end-effector position and base attitude control during pre- and post-capture manoeuvers. A tracking controller in the presence of angular momentum is formulated in [74]. From the resulting linearized input-output relation, geometric control techniques can be applied to follow the desired end-effector pose in a stable manner. In [75], Bullo and Murray design a widely versatile tracking control law on Lie groups. The controller consists of a PD feedback action based on the notions of error functions and transport maps, in addition to a feedforward action to aid in the stability of the output.

Chapter 3

Mission Architecture for On-Orbit Servicing

3.1 Summary

In this chapter, an overall mission architecture for performing multiple on-orbit servicing missions by a fleet of servicers in the form of free-flying single-arm space manipulators is proposed. The architecture targets to improve the two key criteria of industrialization: resource and service. The primary resources included in OOS missions are identified to be the fuel and time consumed in completing a mission. Hence, as part of resource management, the presented architecture first recognizes the main contributors to the fuel consumption and mission duration in the far-range rendezvous phase of OOS missions being: (i) the location of the parking orbit, (ii) the type of transfer trajectories, and (iii) the dispatch scheduling. As the result, separate optimization loops are considered in the mission design for minimizing mission costs. To enhance quality of the provided service, the proposed architecture suggests for servicers to perform appropriate in-orbit machine learning that helps improve the functionality of their guidance, navigation and control systems in upcoming missions. Specifically, tasks involving trajectory learning for a servicer's manipulator system in its free-floating regime to reach a simulated moving target while avoiding (virtual) obstacles and compensating for environmental disturbances are suggested. Applying the trajectory learning scenarios to space manipulators possessing workspace control is considered necessary for improving the learning performance, as well as the versatility, and adaptability of learned process.

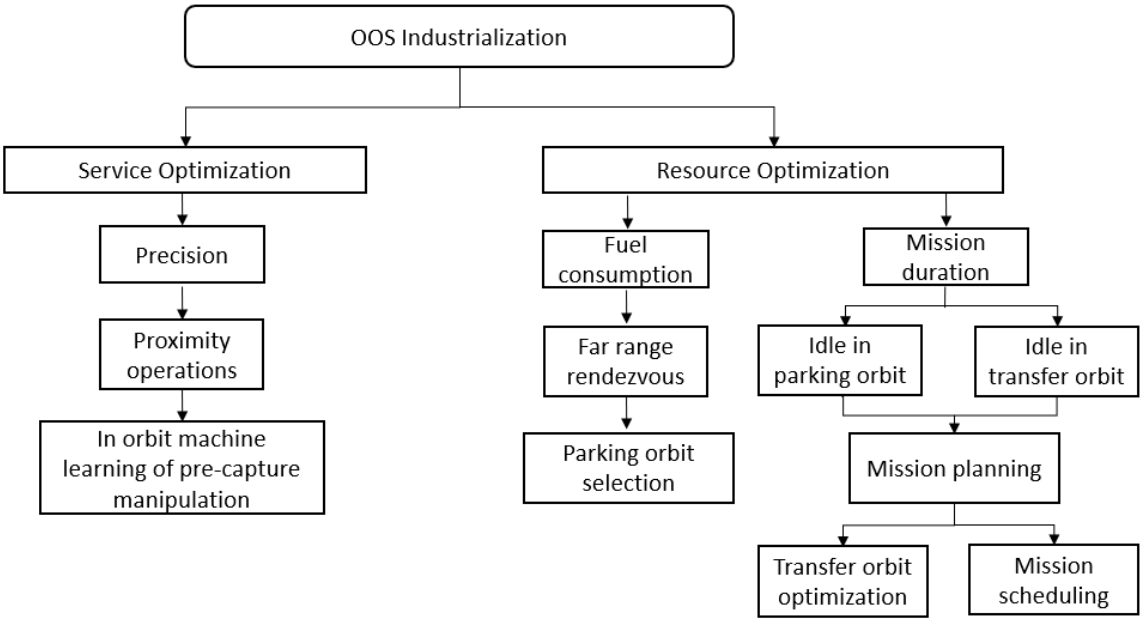


Figure 3.1: OOS industrialization flow chart.

3.2 Proposed Mission Architecture

The OOS mission architecture proposed in this chapter considers multiple servicers and multiple targets (both cooperative operational satellites and non-cooperative orbital debris) in LEO, in the planning. As part of industrializing the OOS, a fleet of approximately 80-100 servicers is suggested to be launched to a single parking orbit in the Sun-synchronous orbital region, forming an equally phased satellite constellation. The size of the servicing fleet is chosen to be of similar magnitude to the LEO communication satellite constellations being on the order of a hundred satellites. An equally phased constellation in the parking orbit is suggested to allow greater service coverage, such that there is always a preferred servicer for rendezvous with a target (with regards to transfer time and fuel consumption). Servicing satellites are assumed in groups of single-arm free-flying space manipulators, with certain servicing capabilities associated with each group. For example, servicers may be designed with a long and flexible manipulator to cater towards in-orbit assembly missions, or designed with stiff links for performing debris removal missions. Multiple servicers may belong to multiple servicing groups, however there are no servicers capable of performing all possible servicing missions.

In the day-to-day activities in the OOS industry, the vast majority of the servicers' time is spent idling in either the parking orbit or a transfer orbit. For example, [76] describes a

servicing approach for GEO spacecraft where it uses a parking orbit as an orbit in which the servicer waits – possibly for several days – until it reaches the same latitude as the client spacecraft before raising itself to GEO. Two possible debris removal mission designs are proposed in [77]: (i) a servicing satellite is launched directly to the orbit of the first debris and subsequently travels to and deorbits all other target debris in the mission, (ii) multiple servicers are launched simultaneously to perform debris removal missions. In either case, [77] states that waiting in a parking orbit is required in order for each servicing satellite to transfer to the orbit of its target debris. The proposed architecture suggests using this waiting time for performing machine learning to improve the functionality of servicers' GNC systems during the pre-capture phase of close proximity servicing operations. We specifically focus on the pre-capture phase, as opposed to other proximity operation phases (e.g., capture and post-capture), due to the importance of designing smooth pre-capture manoeuvres in delivering successful capture and post-capture operations. Furthermore, from the perspective of simulating the learning process in the presence of a virtual target, it is recommended to focus on the pre-capture phase due to the difficulties in imitating the contact forces upon capture and the post-capture operations.

The suggested task involves trajectory learning for a servicer's manipulator system in the free-floating regime to reach a moving target in space while avoiding obstacles and compensating for environmental disturbances. The in-orbit learning process begins by the servicer "visualizing" a virtual target satellite to perform a docking/capture manoeuvre with. The servicer actuates its manipulator, attempting to capture the virtual target, and gathers sensory data (the servicer's base and manipulator states, and the target satellite's position and attitude) during the capture process to be used as the training dataset for learning. This dataset from environmental interactions can then be used to train the servicer's trajectory planning system, as well as perform system identification to enhance the performance of the control system. In defining the trajectory planner learning problem, several challenges may be considered. For example, the trajectory planner could consider the capture of a moving target, primarily for rendezvous with non-cooperative targets (i.e. debris), as well as obstacle avoidance along the path to the capture point on the target body. Obstacles are structural features on the target body (e.g., solar panels, antennas), or parts of the servicer satellite itself (e.g., links of the servicer's manipulator). The learning problem can also impose constraints on the base orientation of the servicing satellite upon successful capture of the target. For communication purposes with a ground station or possibly the Administrator, certain base orientations, identifying antennas' directions, may be desired

once the servicer docks with the target. Similar to every component of the GNC system, it is advantageous for the trajectory planner to be robust against the environmental disturbances (e.g., solar radiation pressure, gravity gradient, atmospheric drag, etc.). Consequently, the learning task accounts for the accumulation of the effect of such disturbances in the relative position and orientation between the servicer and the target over the course of the pre-capture operations. Instantaneously, the disturbance effects will be minimal, however they escalate in time to significantly affect the desired trajectory.

Unsupervised learning (in particular reinforcement learning) is suggested for performing the trajectory learning task in comparison to supervised learning techniques. Firstly, due to the existence of uncertainties in the environmental interactions (by virtue of orbital disturbances) and in the system parameters, an optimal trajectory for the free-floating manipulator to capture a target satellite is not accurately known in advance. Consequently, training datasets are not available to teach the servicing satellite the best path to a capture point on the target, thus rendering supervised learning ineffective. The servicer must perform manipulations in orbit in order to experience the true effects of the orbital environment, as well as its own dynamics. By training with a virtual target satellite, prior to docking with an actual target, the servicer is able to gain experience on its manipulation capabilities, and use reinforcement learning to develop an optimal policy for docking. Secondly, reinforcement learning is more robust to changes in the orbital environment (e.g., increased solar activity), changes to physical properties of the servicer (e.g., mass variations due to fuel consumption), and differences between target satellites (e.g., in geometry or physical properties). A trajectory planning model resulting from supervised learning needs to be retrained with a new training dataset that accounts for such changes in the system and its environment. Conversely, reinforcement learning allows for real time adjustments to the learned policies, greatly reducing the time and computation required to compensate for the aforementioned changes. Moreover, reinforcement learning techniques performed in the continuous domain, such as the Deep Deterministic Policy Gradient (DDPG) algorithm proposed in [78], are particularly desirable due to the difficulties in discretizing the states and actions involved. Further, the manipulation task by a space manipulator system can be considered as a deterministic process, and the DDPG algorithm has already demonstrated an aptitude to be used in space robot applications [5]. Therefore, this algorithm is deemed most suitable for the trajectory learning of a free-floating manipulator system, and it is detailed in the following paragraph.

In applying the DDPG algorithm to the trajectory planning problem, the states space,

action space and reward function must be defined. In this case, the actions determined by the actor neural network define the desired trajectory for which the space manipulator system is to follow. As the overall objective is to move the manipulator's end-effector in time to reach a capture point on the target, it is beneficial (from the perspective of trajectory following performance) for a servicer's GNC system to use an output tracking controller to directly control the end-effector's position and orientation. Moreover, the study conducted in [79] performs an analysis on the action representation for trajectory learning tasks involving manipulators capturing fixed or moving targets, using the DDPG algorithm. The results of the study demonstrate that designing actions in the manipulator's workspace achieve significantly faster convergence to the learned policy and a greater target capture success rate (regarding a fixed target) in comparison to trajectories designed in the manipulator's joint space. The resulting learned policy predicated on workspace trajectories additionally exhibited versatility and adaptability to more complex target capture scenarios involving moving targets and model uncertainties in the manipulator. The authors in [79] show a greater performance when initializing the more complex learning processes with the converged policy output by the trivial stationary target capture scenario, in comparison to learning such new tasks from scratch.

3.3 Workspace Control

As mentioned in the previous subsection, performing output tracking control on free-floating space manipulator systems allows for the greatest performance, versatility, and adaptability resulting from the trajectory learning process. To not stunt the progression of the learning process, and consequently extend the policy's rate of convergence, control designs generating accurate and fast responses in the end-effector's motion must be implemented to achieve sufficient output tracking capabilities. In the early stages of the learning process, highly random actions are performed in order for the actor and critic networks to learn the behaviours of both the agent and the environment. Therefore, the controlled responses of space manipulator systems in following the initially random trajectories are of utmost importance in aiding the performance of the DDPG algorithm. Insufficient workspace control, where the output is incapable of accurately following the desired trajectories, prolongs the convergence to an optimal policy due to the neural networks requiring to learn the controller's behaviour, in order to compensate for the non-negligible tracking errors. As the output trajectory following capabilities increase, the need for the actor and

critic networks to identify the controller's design consequently decreases. Additionally, one of the primary factors influencing the performance of the trajectory learning process is the occurrence of dynamic and kinematic singularities during trajectory following tasks. Often times the regions of singular configurations are not easily known, and therefore the actor network may design trajectories which pass through a singularity, particularly in the early stages of the learning. Consequently, measures for appropriately accommodating and maintaining control during singular regions of operation is crucial for the success of the trajectory learning process.

The remainder of this thesis focuses on developing an output tracking controller (controlling both the position and orientation of the output) for a free-floating space manipulator system with conserved non-zero momentum. This research lays the foundational framework for the trajectory learning task, as well as additional prospective GNC enhancement analyses using reinforcement learning techniques. In designing a workspace control structure with sufficient performance, the following chapter first derives the kinematics, dynamics, and momentum formulations of free-floating space manipulators.

Figure 3.1 portrays the proposed OOS architecture for service industrialization in a flow chart format.

Chapter 4

Kinematics and Dynamics of Free-Floating Space Manipulators

4.1 Summary

This chapter details the kinematic and dynamic modelling of free-floating single-arm space manipulators using the Lie group structure associated with their configuration space. The motion of a single rigid body on the Special Euclidean Lie group $\mathrm{SE}(3)$ is first studied as the foundational framework for modelling a space manipulator system. By considering space manipulators as rigid multi-body systems, the relationships established for a single rigid body are extended to define the full motion of a free-floating space manipulator. The kinematic model begins by parameterizing the six lower pair joints as Lie subgroups of $\mathrm{SE}(3)$, since this thesis considers space manipulators consisting of multi-DOF joints. Using the product of exponentials formula, the forward kinematics mapping of an N-link space manipulator is established to relate the configuration of every body in the system to the spatial coordinate frame. In addition, the matrices of the spatial and body Jacobians, forming the system's differential kinematics, are defined on Lie groups to allow for the derivation of the system's dynamics using the Euler-Lagrange equation. Finally, the conserved total momentum of a free-floating space manipulator system in the spatial coordinate frame is formulated based on the system's inertia matrix.

4.2 Space Manipulator System Kinematics

This thesis considers free-floating space manipulator systems consisting of an N-link manipulator attached to a 6-DOF base spacecraft. All manipulator links are interconnected

through either single or multi-DOF joints, and the motion of the base is captured by 3 prismatic and 3 revolute joints. These six joints are all coincident at the base's center of mass. The space manipulator is modelled as a serial-link open-chain rigid multi-body system with $N+1$ bodies and $N_D = N_{D_M} + 6$ degrees of freedom, where N_{D_M} indicates the number of degrees of freedom included in only the manipulator. All body coordinate frames are attached to the bodies' centers of masses and are assumed to be aligned with the respective principal inertia axes. We number the bodies by $k = 0, \cdots, N$, where body 0 refers to the base. The joint twists capturing the base motion are denoted by $\boldsymbol{\xi}_1, \cdots, \boldsymbol{\xi}_6$, while the twists of the manipulator joints are denoted by $\boldsymbol{\xi}_7, \cdots, \boldsymbol{\xi}_{N_D}$. Let $\boldsymbol{\theta} \in \mathbb{R}^6$ be the collection of the base joint parameters, and $\boldsymbol{q} \in \mathbb{R}^{N_{D_M}}$ be the collection of the joint parameters for the manipulator. The vector of generalized coordinates for a space manipulator system is thus the combination of $\boldsymbol{\theta}$ and $\boldsymbol{q}$, denoted by $\boldsymbol{\Phi} = [\boldsymbol{\theta}^T \ \boldsymbol{q}^T]^T \in \mathbb{R}^{N_D}$.

This section develops a complete kinematic description of free-floating space manipulators on the Lie group $\mathrm{SE}(3)$. To start, the parameterization of various joint types in the language of Lie groups is provided to relate the joint motions of a space manipulator with the theory provided in section A.1, regarding the modelling of rigid body motion on Lie groups. By treating the system joints as subgroups of $\mathrm{SE}(3)$, the resulting forward kinematics model is predicated on the system's geometry and configuration. Finally, this section details the differential kinematics of space manipulators to develop a relationship between the system and end-effector velocities, known as the system Jacobian. The Jacobian is expressed in both the spatial and body coordinate frames to project the joint motions into the end-effector's spatial or body velocity vector.

4.2.1 Multi-DOF Joint Parameterization

As previously mentioned, this thesis involves both single and multi-DOF joints in the modelling of space manipulator systems. The six lower pair joint types are considered, as such joints are most common in robotic systems. Lower pair joints include revolute, prismatic, helical, planar, cylindrical, and spherical joints. Their associated motions can be expressed in subgroups of $\mathrm{SE}(3)$. Table 4.1 demonstrates the classification of each lower pair joint on the $\mathrm{SE}(3)$ Lie group.

Starting with the single-DOF joints located at the bottom row of Table 4.1, the revolute and prismatic joints refer to pure rotational and translational screw motions, respectively. Revolute joints are described by the one-dimensional subgroup of $\mathrm{SO}(3)$, denoted

Table 4.1: Lower pair joints

Dimension	Subgroups of SE(3)		
3	$SE(2) = SO(2) \ltimes \mathbb{R}^2$ planar	$SO(3)$ ball (spherical)	
2	$SO(2) \times \mathbb{R}$ cylindrical		
1	$SO(2)$ revolute	$\mathbb{R}$ prismatic	H_p helical

by $SO(2)$, which specifies rotation about a single axis, and prismatic joints are described by $\mathbb{R}$ which indicates translation in a single direction. The helical joint contains a single independent motion, represented by the subgroup H_p, to express a complete screw motion with a non-zero and finite pitch value. Cylindrical joints combine individual revolute and prismatic motions $SO(2) \times \mathbb{R}$, to both rotate and translate an attached rigid body. The planar and spherical joints in the lower pair group both impart motions with three degrees of freedom. Planar joints posses full planar motion (i.e., translation and rotation in two-dimensional space), combining $\mathbb{R}^2$ and $SO(2)$ to define the Special Euclidean space for two dimensional motion, $SE(2)$. A spherical joint allows for complete rotational motion, described by rotation matrices belonging to $SO(3)$. We may also include the free six-DOF joint to describe unconstrained motion in $SE(3)$. This six-DOF joint proves necessary when describing the uncontrolled base motion of a space manipulator in its free-floating regime.

As mentioned, the motion of the more trivial single-DOF joints are represented in a geometric context by a screw motion (helical joint motion) with either infinite pitch (prismatic joint motion) or zero pitch (revolute joint motion). Subsequently, provided the twist $\boldsymbol{\xi}$ associated with the single-DOF joint motion, the relative motion between a rigid body's initial configuration and its configuration following the joint motion is obtained through the exponential mapping $\exp(\boldsymbol{\xi}\Phi)$. With regards to the multi-DOF joints, this thesis expresses their motion as an amalgamation of individual screw motions, each representing a single degree of freedom in the joint's motion. For example, the full rotational motion of a spherical joint (represented by $SO(3)$) is effectively defined by the combination of three revolute joints placed along each of the primary axes. For a body k with initial configuration $\boldsymbol{g}_{sk}(0)$, and whose motion is defined by a spherical joint, its resulting configuration following the spherical motion $\boldsymbol{g}_{sk}(\Phi_z, \Phi_y, \Phi_x)$ is defined as follows,

$$\boldsymbol{g}_{sk}(\Phi_z, \Phi_y, \Phi_x) = e^{\hat{\boldsymbol{\xi}}_z \Phi_z} e^{\hat{\boldsymbol{\xi}}_y \Phi_y} e^{\hat{\boldsymbol{\xi}}_x \Phi_x} \boldsymbol{g}_{sk}(0). \tag{4.1}$$

The representation of the spherical joint above signifies an Euler angle representation of the joint's motion. The rotation angles Φ_z, Φ_y, and Φ_x refer to the yaw, pitch, and roll of the joint, and $\boldsymbol{\xi}_z, \boldsymbol{\xi}_y, \boldsymbol{\xi}_x$ refer to twists placed along the z, y, x axes. Note that for a planar joint (with motion in SE(2)), the same combination of exponential mappings in equation (4.1) is used, however with the last two exponentials referring to translational motions about the y and x axes. The principle of combining individual revolute and prismatic motions to represent higher DOF joints provides the following general representation of joint motions on Lie groups for the joint m with l_m degrees of freedom,

$$\boldsymbol{g}_m(\Phi_1, \ldots, \Phi_{l_m}) = e^{\hat{\boldsymbol{\xi}}_1 \Phi_1} \ldots e^{\hat{\boldsymbol{\xi}}_{l_m} \Phi_{l_m}} \boldsymbol{g}_m(0). \tag{4.2}$$

The exponential mapping $\exp(\hat{\boldsymbol{\xi}}_i \Phi_i)$ may represent either a pure rotational or translational motion. Based on the Lie group parameterization of the six lower pair joints in Table 4.1, and the motion definition for a single and multi-DOF joint, the system's forward kinematics can be formed. The following subsection describes the method for combining consecutive joint motions to define the forward kinematics mapping from the system configuration to the end-effector pose in SE(3).

4.2.2 Forward Kinematics

As mentioned, the primary objective of the forward kinematics of a space manipulator system is to determine the position and orientation of end-effector $\boldsymbol{g}_{sN}$ provided the joint configurations in the system. More generally, the forward kinematics model provides a function for computing the configuration of any body k in the system given the joint configurations preceding body k. The region of allowable joint values is referred to as the system's joint space, and the region of achievable end-effector configurations is referred to as either the system's task space or workspace. Following a right-handed coordinate system, a counterclockwise rotation defines the positive direction of rotational motions if viewed along the axis of rotation. For translational motion, the displacement is measured positive in the direction of the corresponding screw axis.

Starting with the homogeneous transformation from the spatial frame to the coordinate frame following the first joint's motion $\boldsymbol{g}_{s0}$, the forward kinematics map sequentially merges individual joint transformations until reaching the end-effector. That is, for a general robotic manipulator with N bodies, and where body k is subsequent to a joint with l_{m_k} degrees of freedom, the forward kinematics to the end-effector is computed as follows,

$$\boldsymbol{g}_{sN}(\boldsymbol{\Phi}) = \boldsymbol{g}_{s0}(\Phi_1,\ldots,\Phi_{l_0})\boldsymbol{g}_{01}(\Phi_{l_0+1},\ldots,\Phi_{l_1})\cdots\boldsymbol{g}_{(N-1)N}(\Phi_{l_{N-1}+1},\ldots,\Phi_{l_N})\boldsymbol{g}_{sN}(0). \tag{4.3}$$

Here, $l_k = \sum_{i=0}^{k} l_{m_i}$ defines the number of degrees of freedom in the system preceding body k. For example, the spacecraft's base has six degrees of freedom, thus defining $l_0 = 6$. The full forward kinematics mapping to the end-effector above can be easily generalized to any body k in the system by simply ending the product of joint transformations at body k. This implies that the combination of joint transformations in equation (4.3) ends at $\boldsymbol{g}_{(k-1)k}$ as opposed to $\boldsymbol{g}_{(N-1)N}$. Note that the joint homogeneous transformation between two connected bodies is defined based on equation (4.2) with $l_m = l_{m_k}$. Implementing equation (4.2) into equation (4.3) demonstrates the general forward kinematic mapping in terms of each individual prismatic or revolute motion included in the system. The notion of compounding individual motions (corresponding to either single or multi-DOF joints) using the exponential joint parameterization in the forward kinematics model is known as the product of exponentials formula. The forward kinematics mapping for body k can thus be expressed by the product of exponentials formula as follows,

$$\boldsymbol{g}_{sk}(\boldsymbol{\Phi}) = e^{\hat{\boldsymbol{\xi}}_1\Phi_1}\cdots e^{\hat{\boldsymbol{\xi}}_{l_k}\Phi_{l_k}}\boldsymbol{g}_{sk}(0). \quad k = 0,\cdots,N \tag{4.4}$$

The product of exponentials formula above signifies the transformation of the k^{th} body's initial configuration into its updated pose based on the current configuration of the space manipulator system.

4.2.3 Differential Kinematics

With the developed forward kinematics model providing a connection from joint positions to the pose of a rigid body k, differentiating this map provides a linear mapping from

joint velocities to the velocity of body k. The differential of the forward kinematics is known as the system's Jacobian $\boldsymbol{J}(\boldsymbol{\Phi}) \in \mathbb{R}^{6\times N_D}$ which can be expressed in either the spatial frame $\boldsymbol{J}^s(\boldsymbol{\Phi})$, or the body frame $\boldsymbol{J}^b(\boldsymbol{\Phi})$, for mapping velocities to the respective frame of reference. For example, the spatial velocity of body k, denoted $\boldsymbol{V}_k^s$, requires body k's spatial Jacobian $\boldsymbol{J}_k^s(\boldsymbol{\Phi})$ and the joint velocities,

$$\boldsymbol{V}_k^s = \boldsymbol{J}_k^s(\boldsymbol{\Phi})\dot{\boldsymbol{\Phi}}. \tag{4.5}$$

As indicated by the nomenclature, the system Jacobian is a function of the generalized coordinates $\boldsymbol{\Phi}$ due to its dependency on the forward kinematics. Recalling the definition of the spatial velocity in equation (A.33), we expand the time derivative of the homogeneous transformation $\boldsymbol{g}_{sk}$

$$\dot{\boldsymbol{g}}_{sk}(\boldsymbol{\Phi}) = \sum_{i=1}^{N_D} \frac{\partial \boldsymbol{g}_{sk}(\boldsymbol{\Phi})}{\partial \Phi_i}\dot{\Phi}_i. \tag{4.6}$$

Based on the definition of the spatial velocity,

$$\hat{\boldsymbol{V}}_k^s = \sum_{i=1}^{N_D} \left(\frac{\partial \boldsymbol{g}_{sk}(\boldsymbol{\Phi})}{\partial \Phi_i}\dot{\Phi}_i \right) \boldsymbol{g}_{sk}^{-1}(\boldsymbol{\Phi}) = \sum_{i=1}^{N_D} \left(\frac{\partial \boldsymbol{g}_{sk}(\boldsymbol{\Phi})}{\partial \Phi_i}\boldsymbol{g}_{sk}^{-1}(\boldsymbol{\Phi}) \right) \dot{\Phi}_i. \tag{4.7}$$

Appendix D provides the derivation of the partial derivative of a homogeneous transformation with respect to a generalized coordinate. The result of this differentiation implies that the argument of the summation above can be replaced by applying the Adjoint of the product of exponentials up to Φ_{i-1} to the i^{th} twist $\boldsymbol{\xi}_i$. That is,

$$\left(\frac{\partial \boldsymbol{g}_{sk}}{\partial \Phi_i}\boldsymbol{g}_{sk}^{-1} \right)^{\vee} = \mathbf{Ad}_{(e^{\hat{\boldsymbol{\xi}}_1\theta_1} \ldots e^{\hat{\boldsymbol{\xi}}_{i-1}\theta_{i-1}})}\boldsymbol{\xi}_i. \tag{4.8}$$

Therefore, the spatial Jacobian is defined by expressing summation in equation (4.7) in matrix form.

$$\boldsymbol{J}_k^s = [\boldsymbol{\eta}_1' \ \cdots \ \boldsymbol{\eta}_{l_k}' \quad \mathbf{0}_{6\times(N_D-l_k)}] \quad \in \mathbb{R}^{6\times N_D} \tag{4.9}$$

$$\boldsymbol{\eta}_i' = (\mathbf{Ad}_1^{i-1})'\boldsymbol{\xi}_i. \quad i = 1,\cdots,l_k, \tag{4.10}$$

where,

$$(\mathbf{Ad}_1^{i-1})' := \mathbf{Ad}_{(e^{\hat{\boldsymbol{\xi}}_1\theta_1} \ \ldots \ e^{\hat{\boldsymbol{\xi}}_{i-1}\theta_{i-1}})}. \tag{4.11}$$

From this equation, the i^{th} column of the spatial Jacobian represents the transformation from the i^{th} twist in the initial configuration to its updated location, given robot's current configuration, with respect to the spatial coordinate frame. This transformation is based on all preceding $i-1$ joint angles, and defined by the Adjoint matrix in equation (4.10). Consequently, the spatial velocity of body k only relies on the preceding joint angles and is independent of the succeeding joint parameters. This is expressed in the Jacobian matrix by all $l_k+1,\cdots,N_D$ columns being zero.

The body Jacobian associated with the k^{th} body provides the linear mapping from the joint velocities to body k's body velocity,

$$\boldsymbol{V}_k^b = \boldsymbol{J}_k^b(\boldsymbol{\Phi})\dot{\boldsymbol{\Phi}}. \tag{4.12}$$

Similarly, based on the definition of $\boldsymbol{V}_k^b$ in equation (A.34) and the derivative of $\boldsymbol{g}_{sk}$, the body Jacobian can be formed as the mapping between $\dot{\boldsymbol{\Phi}}$ and $\boldsymbol{V}_k^b$,

$$\boldsymbol{J}_k^b = [\boldsymbol{\eta}_1 \ \cdots \ \boldsymbol{\eta}_{l_k} \quad \mathbf{0}_{6\times(N_D-l_k)}] \quad \in \mathbb{R}^{6\times N_D} \tag{4.13}$$

$$\boldsymbol{\eta}_i = \mathbf{Ad}_{\boldsymbol{g}_{sk}^{-1}(0)}\mathbf{Ad}_i^{l_k}\boldsymbol{\xi}_i, \quad i = 1,\cdots,l_k \tag{4.14}$$

where for $l > i$

$$\mathbf{Ad}_i^l := \mathbf{Ad}_{(e^{-\hat{\boldsymbol{\xi}}_l\Phi_l} \ \ldots \ e^{-\hat{\boldsymbol{\xi}}_i\Phi_i})}. \tag{4.15}$$

In this body frame representation of the Jacobian matrix, the i^{th} column expresses the updated i^{th} twist with respect to the coordinate frame attached to body k. Again, the transformation updating the twist to the current configuration is predicated on the joint configurations preceding body k, as expressed by the Adjoint multiplication in equation (4.14). Analogous to the connection between the spatial and body velocities, the spatial and body manipulator Jacobians can also be related through the Adjoint operator associated with the relating homogeneous transformation $\boldsymbol{g}_{sk}$,

$$\boldsymbol{J}_k^s = \mathbf{Ad}_{\boldsymbol{g}_{sk}} \boldsymbol{J}_k^b. \tag{4.16}$$

The following section makes use of the differential kinematics mapping presented here in forming the kinetic energy of a space manipulator system. Consequently, the system's Jacobian matrix forms the foundation for deriving the dynamical equations of space manipulators, in the context of Lie group theory.

4.3 Space Manipulator System Dynamics

In this section, the equations of motion for a free-floating space manipulator system are derived based on a Euler-Lagrangian approach. Such approach is known to be computationally inefficient in comparison to the Newton-Euler method; however, its use is necessary for developing model-based control schemes. The equations of motion dictate the acceleration and subsequent motion of a free-floating space manipulator under actuation forces and torques applied to the joints. Deriving the system dynamics based on a Lagrangian approach requires the system's kinetic and potential energies to defined the Lagrangian $\boldsymbol{L}$. Differentiating the Lagrangian with respect to the generalized coordinates and their velocities provides a relationship between the system accelerations and a vector containing the system forces and moments. In the case of a space manipulator system in orbit, the potential energy of the system can be neglected due to the low-gravity environment and the absence of any energy storing elements in the considered system. Consequently, the Lagrangian $\boldsymbol{L}$ is simply equal to the kinetic energy of the space manipulator system $\boldsymbol{K}$. The resulting equations of motion combine the time derivative of the partial derivative of the $\boldsymbol{L}$ with respect to the system's generalized velocities and the partial derivative of the Lagrangian with respect to the generalized coordinates. That is, the equation of motion corresponding to a single degree of freedom i is expressed as follows,

$$\frac{d}{dt}\frac{\partial \boldsymbol{L}}{\partial \dot{\Phi}_i} - \frac{\partial \boldsymbol{L}}{\partial \Phi_i} = T_i \tag{4.17}$$

where T_i represents the generalized force or torque (depending on translational or rotational motion) collocated with the i^{th} generalized coordinate. Based on the Lagrangian definition for space manipulator systems $\boldsymbol{L} = \boldsymbol{K}$, we start by defining the kinetic energy associated with the k^{th} body,

$$\boldsymbol{K}_k(\boldsymbol{\Phi}, \dot{\boldsymbol{\Phi}}) = \frac{1}{2}(\boldsymbol{V}_k^b)^T \boldsymbol{D}_k \boldsymbol{V}_k^b, \tag{4.18}$$

where $\boldsymbol{D}_k$ denotes body k's constant 6×6 inertia matrix with respect to its body frame. As previously mentioned, coordinate frame attached to any body k is located at the body's center of mass and aligned in the direction of its principal axes. This placement and orientation of body coordinate frames results in the following mass matrix (assuming uniform mass distributions),

$$\boldsymbol{D}_k = \begin{bmatrix} m_k & 0 & 0 & 0 & 0 & 0 \\ 0 & m_k & 0 & 0 & 0 & 0 \\ 0 & 0 & m_k & 0 & 0 & 0 \\ 0 & 0 & 0 & I_{xk} & 0 & 0 \\ 0 & 0 & 0 & 0 & I_{yk} & 0 \\ 0 & 0 & 0 & 0 & 0 & I_{zk} \end{bmatrix}. \tag{4.19}$$

Here, m_k refers to the mass of body k, and I_{xk}, I_{yk}, and I_{zk} refer to its principal moments of inertia. The kinetic energy's dependency on $\boldsymbol{\Phi}$ and $\dot{\boldsymbol{\Phi}}$ becomes apparent when introducing body k's Jacobian to express the body velocity vectors in equation (4.18),

$$\boldsymbol{K}_k(\boldsymbol{\Phi}, \dot{\boldsymbol{\Phi}}) = \frac{1}{2}(\boldsymbol{J}_k^b(\boldsymbol{\Phi})\dot{\boldsymbol{\Phi}})^T \boldsymbol{D}_k \boldsymbol{J}_k^b(\boldsymbol{\Phi})\dot{\boldsymbol{\Phi}}, \tag{4.20}$$

which is rewritten as,

$$\boldsymbol{K}_k(\boldsymbol{\Phi},\dot{\boldsymbol{\Phi}}) = \frac{1}{2}\dot{\boldsymbol{\Phi}}^T(\boldsymbol{J}_k^b(\boldsymbol{\Phi}))^T\boldsymbol{D}_k\boldsymbol{J}_k^b(\boldsymbol{\Phi})\dot{\boldsymbol{\Phi}}. \tag{4.21}$$

The resulting expression above signifies that the kinetic energy associated with body k depends on its inertia matrix and body Jacobian. Moreover, the total kinetic energy of the system is the sum of each body's individual kinetic energy,

$$\boldsymbol{K}(\boldsymbol{\Phi},\dot{\boldsymbol{\Phi}}) = \sum_{k=0}^{N}\boldsymbol{K}_k(\boldsymbol{\Phi},\dot{\boldsymbol{\Phi}}), \tag{4.22}$$

The required summation pertaining to the full system kinetic energy thus depends on the matrix products of the body Jacobians and body inertia matrices. This summation can be simplified by introducing the matrix $\bar{\boldsymbol{M}}(\boldsymbol{\Phi}) \in \mathbb{R}^{N_D\times N_D}$. This matrix is known as the inertia matrix of the system and is defined based on

$$\bar{\boldsymbol{M}}(\boldsymbol{\Phi}) := \sum_{k=1}^{N}(\boldsymbol{J}_k^b(\boldsymbol{\Phi}))^T\boldsymbol{D}_k\boldsymbol{J}_k^b(\boldsymbol{\Phi}), \tag{4.23}$$

such that,

$$\boldsymbol{K}(\boldsymbol{\Phi},\dot{\boldsymbol{\Phi}}) = \frac{1}{2}\dot{\boldsymbol{\Phi}}^T\bar{\boldsymbol{M}}(\boldsymbol{\Phi})\dot{\boldsymbol{\Phi}}. \tag{4.24}$$

To efficiently formulate $\bar{\boldsymbol{M}}(\boldsymbol{\Phi})$, the summation in equation (4.23) can be expressed in terms of a single matrix multiplication. In doing so, we define a $6N_D \times N_D$ matrix $\boldsymbol{J}$ which contains all of the body Jacobian matrices in the system,

$$\boldsymbol{J}(\boldsymbol{\Phi}) := \begin{bmatrix} \boldsymbol{J}_0^b(\boldsymbol{\Phi}) \\ \boldsymbol{J}_1^b(\boldsymbol{\Phi}) \\ \vdots \\ \boldsymbol{J}_N^b(\boldsymbol{\Phi}) \end{bmatrix}, \tag{4.25}$$

and a $6N_D \times 6N_D$ block diagonal matrix collecting the individual body inertia matrices,

$$\boldsymbol{D} := \begin{bmatrix} \boldsymbol{D}_0 & 0 & \dots & 0 \\ 0 & \boldsymbol{D}_1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \boldsymbol{D}_N \end{bmatrix}. \tag{4.26}$$

Based on the definition of the matrices $\boldsymbol{J}$ and $\boldsymbol{D}$, the system inertia matrix is expressed as follows,

$$\bar{\boldsymbol{M}}(\boldsymbol{\Phi}) = \boldsymbol{J}^T(\boldsymbol{\Phi})\boldsymbol{D}\boldsymbol{J}(\boldsymbol{\Phi}). \tag{4.27}$$

The inertia matrix can be partitioned by rows and columns into four sub-matrices. These sub-matrices separate the inertial contributions to the system dynamics from the spacecraft base and the manipulator,

$$\bar{\boldsymbol{M}}(\boldsymbol{\Phi}) = \begin{bmatrix} \boldsymbol{H}_o & \boldsymbol{H}_{om} \\ \boldsymbol{H}_{om}^T & \boldsymbol{H}_m \end{bmatrix}. \tag{4.28}$$

Here, $\boldsymbol{H}_o$ is the 6×6 matrix of the base contribution to the system's momentum represented in the base parameterization, $\boldsymbol{H}_{om}$ is the $6 \times N$ matrix of the manipulator contribution to the system's momentum (similarly expressed in the base parameterization), and $\boldsymbol{H}_m$ is the $N \times N$ matrix of the manipulator's inertia.

Given the total system kinetic energy expressed in equation (4.24), the resulting equations of motion of a space manipulator are found using the Euler-Lagrangian equation in (4.17). Since the system inertia matrix is only dependent on the generalized coordinates $\boldsymbol{\Phi}$, the partial derivative of the Lagrangian (equal to the system's total kinetic energy) with respect to the generalized velocities is

$$\frac{\partial \boldsymbol{L}(\boldsymbol{\Phi}, \dot{\boldsymbol{\Phi}})}{\partial \dot{\boldsymbol{\Phi}}} = \bar{\boldsymbol{M}}(\boldsymbol{\Phi})\dot{\boldsymbol{\Phi}}. \tag{4.29}$$

Taking the time derivative of this equation defines the first term in the Euler-Lagrange equation,

$$\frac{d}{dt}\frac{\partial \boldsymbol{L}(\boldsymbol{\Phi},\dot{\boldsymbol{\Phi}})}{\partial \dot{\boldsymbol{\Phi}}} = \bar{\boldsymbol{M}}(\boldsymbol{\Phi})\ddot{\boldsymbol{\Phi}} + \dot{\bar{\boldsymbol{M}}}(\boldsymbol{\Phi})\dot{\boldsymbol{\Phi}}, \tag{4.30}$$

where the time derivative of the system inertia matrix is calculated as follows,

$$\dot{\bar{\boldsymbol{M}}}(\boldsymbol{\Phi}) = \sum_{i=1}^{N_D} \frac{\partial \bar{\boldsymbol{M}}(\boldsymbol{\Phi})}{\partial \Phi_i}\dot{\Phi}_i. \tag{4.31}$$

As the dependency of the system inertia matrix to the generalized coordinates stems from the body Jacobian, the partial derivative of the inertia matrix depends on the partial derivative of the Jacobian matrix with respect to the i^{th} generalized coordinate. That is, by way of the chain rule, the partial derivative of the inertia matrix is calculated as,

$$\frac{\partial \bar{\boldsymbol{M}}(\boldsymbol{\Phi})}{\partial \Phi_i} = \left(\frac{\partial \boldsymbol{J}(\boldsymbol{\Phi})}{\partial \Phi_i}\right)^T \boldsymbol{D}\boldsymbol{J}(\boldsymbol{\Phi}) + \boldsymbol{J}^T(\boldsymbol{\Phi})\boldsymbol{D}\frac{\partial \boldsymbol{J}(\boldsymbol{\Phi})}{\partial \Phi_i}. \quad i = 1,\cdots,N_D \tag{4.32}$$

Given the form of the j^{th} column in body k's body Jacobian in equation (4.14), only the Adjoint operator $\mathbf{Ad}_i^{l_k}$ transforming the twist $\boldsymbol{\xi}_i$ to the current configuration is dependent on $\boldsymbol{\Phi}$. The Adjoint of the initial pose $\boldsymbol{g}_{sk}(0)$ and the twist $\boldsymbol{\xi}_i$ are independent of the system's configuration. Consequently, the derivative of the j^{th} column of body k's body Jacobian with respect to Φ_i becomes

$$\frac{\partial \boldsymbol{J}_{k,j}^b}{\partial \Phi_i} = \mathbf{Ad}_{\boldsymbol{g}_{sk}^{-1}(0)}\frac{\partial \mathbf{Ad}_j^{l_k}}{\partial \Phi_i}\boldsymbol{\xi}_j. \tag{4.33}$$

The expression for the partial derivative of the Adjoint operator is presented below, resulting from the complete derivation provided in Appendix E,

$$\frac{\partial \mathbf{Ad}_j^{l_k}}{\partial \Phi_i} = -\mathbf{Ad}_{i+1}^{l_k}\mathbf{ad}_{\boldsymbol{\xi}_i}\mathbf{Ad}_j^i. \tag{4.34}$$

Implementing the derivative of the Adjoint operator in equation (4.33), the partial derivative of the j^{th} column of the body Jacobian is written as,

$$\frac{\partial \boldsymbol{J}^b_{k,j}}{\partial \Phi_i} = -\mathbf{Ad}_{\boldsymbol{g}_{sk}^{-1}(0)} \mathbf{Ad}^{l_k}_{i+1} \mathbf{ad}_{\boldsymbol{\xi}_i} \mathbf{Ad}^{i}_{j} \boldsymbol{\xi}_j. \quad i \leq j \tag{4.35}$$

For $i > j$ this partial derivative is identically equal to zero. The equations of motion can now be demonstrated in matrix form provided the expression for the partial derivative of the inertia matrix,

$$\bar{\boldsymbol{M}}(\boldsymbol{\Phi})\ddot{\boldsymbol{\Phi}} + \dot{\bar{\boldsymbol{M}}}(\boldsymbol{\Phi})\dot{\boldsymbol{\Phi}} - \frac{1}{2}\dot{\boldsymbol{\Phi}}^T \frac{\partial \bar{\boldsymbol{M}}(\boldsymbol{\Phi})}{\partial \boldsymbol{\Phi}}\dot{\boldsymbol{\Phi}} = \boldsymbol{\tau}. \tag{4.36}$$

The vector $\boldsymbol{\tau} \in \mathbb{R}^{N_D}$ contains the forces and torques collocated with the generalized coordinates in the system. Due to the free-floating nature of the spacecraft's base, the first 6 elements of $\boldsymbol{\tau}$ (corresponding to the base motion) are equal to zero. That is, $\boldsymbol{\tau} = [\mathbf{0}_{1\times 6}\ \boldsymbol{\tau}_m^T]^T$, where $\boldsymbol{\tau}_m \in \mathbb{R}^N$ is the vector of torques applied to the manipulator joints. The generalized coordinate velocity vector $\dot{\boldsymbol{\Phi}}$ can additionally be factored out from the second and third term in equation (4.36) to yield

$$\bar{\boldsymbol{M}}(\boldsymbol{\Phi})\ddot{\boldsymbol{\Phi}} + \left(\sum_{i=1}^{N_D} \frac{\partial \bar{\boldsymbol{M}}(\boldsymbol{\Phi})}{\partial \Phi_i}\dot{\Phi}_i - \frac{1}{2}\begin{bmatrix} \dot{\boldsymbol{\Phi}}^T \frac{\partial \bar{\boldsymbol{M}}(\boldsymbol{\Phi})}{\partial \Phi_1} \\ \vdots \\ \dot{\boldsymbol{\Phi}}^T \frac{\partial \bar{\boldsymbol{M}}(\boldsymbol{\Phi})}{\partial \Phi_{N_D}}. \end{bmatrix} \right) \dot{\boldsymbol{\Phi}} = \boldsymbol{\tau}. \tag{4.37}$$

The resulting term which multiplies $\dot{\boldsymbol{\Phi}}$ represents the system Coriolis matrix, denoted by $\bar{\boldsymbol{C}}(\boldsymbol{\Phi}, \dot{\boldsymbol{\Phi}}) \in \mathbb{R}^{N_D \times N_D}$. This matrix contains the space manipulator's Coriolis and centrifugal effects, i.e.,

$$\bar{\boldsymbol{C}}(\boldsymbol{\Phi}, \dot{\boldsymbol{\Phi}}) = \sum_{i=1}^{N_D} \frac{\partial \bar{\boldsymbol{M}}(\boldsymbol{\Phi})}{\partial \Phi_i}\dot{\Phi}_i - \frac{1}{2}\begin{bmatrix} \dot{\boldsymbol{\Phi}}^T \frac{\partial \bar{\boldsymbol{M}}(\boldsymbol{\Phi})}{\partial \Phi_1} \\ \vdots \\ \dot{\boldsymbol{\Phi}}^T \frac{\partial \bar{\boldsymbol{M}}(\boldsymbol{\Phi})}{\partial \Phi_{N_D}}. \end{bmatrix}. \tag{4.38}$$

In a similar fashion to the inertia matrix partitioning, the system Coriolis matrix can likewise be broken down into its matrix elements. In this case, these sub-matrices demonstrate the Coriolis and centrifugal effects stemming from the motion of the base and the manipulator,

$$\bar{\boldsymbol{C}}(\boldsymbol{\Phi},\dot{\boldsymbol{\Phi}}) = \begin{bmatrix} \boldsymbol{C}_o & \boldsymbol{C}_{om} \\ \boldsymbol{C}_{mo} & \boldsymbol{C}_m \end{bmatrix}. \tag{4.39}$$

The 6×6 and $6 \times N$ matrices $\boldsymbol{C}_o$ and $\boldsymbol{C}_{om}$ represent the contribution of the Coriolis forces on the total system momentum expressed in the parameterization of the base, resulting from base and manipulator motions, respectively. Additionally, $\boldsymbol{C}_{mo}$ and $\boldsymbol{C}_m$ are $N \times 6$ and $N \times N$ matrices summarizing those contributions in the Coriolis forces on the manipulator. Making use of the system inertia and Coriolis matrices, the equations of motion for a space manipulator system are written in matrix form as follows,

$$\bar{\boldsymbol{M}}(\boldsymbol{\Phi})\ddot{\boldsymbol{\Phi}} + \bar{\boldsymbol{C}}(\boldsymbol{\Phi},\dot{\boldsymbol{\Phi}})\dot{\boldsymbol{\Phi}} = \boldsymbol{\tau}, \tag{4.40}$$

$$\begin{bmatrix} \boldsymbol{H}_o & \boldsymbol{H}_{om} \\ \boldsymbol{H}_{om}^T & \boldsymbol{H}_m \end{bmatrix} \begin{bmatrix} \ddot{\boldsymbol{\theta}} \\ \ddot{\boldsymbol{q}} \end{bmatrix} + \begin{bmatrix} \boldsymbol{C}_o & \boldsymbol{C}_{om} \\ \boldsymbol{C}_{mo} & \boldsymbol{C}_m \end{bmatrix} \begin{bmatrix} \dot{\boldsymbol{\theta}} \\ \dot{\boldsymbol{q}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{0}_{6\times 1} \\ \boldsymbol{\tau}_m \end{bmatrix}. \tag{4.41}$$

Given these matrix partitionings, the dynamics of a free-floating space manipulator system can be divided into the equations of motion pertaining to the base (i.e., the first six rows of equation (4.40)) and and those concerning the manipulator (i.e., the last N rows of equation (4.40)) separately,

$$\boldsymbol{H}_o\ddot{\boldsymbol{\theta}} + \boldsymbol{H}_{om}\ddot{\boldsymbol{q}} + \boldsymbol{C}_o\dot{\boldsymbol{\theta}} + \boldsymbol{C}_{om}\dot{\boldsymbol{q}} = \boldsymbol{0}_{6\times 1} \tag{4.42}$$

$$\boldsymbol{H}_{om}^T\ddot{\boldsymbol{\theta}} + \boldsymbol{H}_m\ddot{\boldsymbol{q}} + \boldsymbol{C}_{mo}\dot{\boldsymbol{\theta}} + \boldsymbol{C}_m\dot{\boldsymbol{q}} = \boldsymbol{\tau}_m. \tag{4.43}$$

Separating the equations of motion demonstrates the coupling between the spacecraft base and the manipulator. Equation (4.43) emphasizes the dynamic influence that an actuation torque applied to the manipulator joints imposes on the base motion. Due to the application of $\boldsymbol{\tau}_m$, the spacecraft base's acceleration is proportional to the coupling inertia

sub-matrix $\boldsymbol{H}_{om}^{T}$ and with velocity related by the coupling component of the Coriolis matrix $\boldsymbol{C}_{mo}$. Thus, having no control of the spacecraft base in the free-floating regime imposes difficulties in controlling the entire space manipulator system due to the coupled motion between the base and manipulator.

4.4 Conservation of Momentum

As previously mentioned, the motion of a free-floating space manipulator is exclusively controlled by actuating the spacecraft's manipulator. As a result, in order for the system to achieve some form of trajectory following, the affect of the manipulator motion on the motion of the entire system is required. The influence of the base motion to space manipulator system becomes irrelevant from a control perspective during the free-floating regime due to the base being unactuated. Consequently, the goal of this section is to obtain a relationship between the base and manipulator motions emanating from the system's conserved momentum. We start by defining the generalized system momentum of a space manipulator to be the matrix product of the system's inertia and its associated generalized coordinate velocities,

$$\boldsymbol{P} := \bar{\boldsymbol{M}}(\boldsymbol{\Phi})\dot{\boldsymbol{\Phi}}. \tag{4.44}$$

Recalling equation (4.27) describing the matrix form of the system inertia matrix, the general momentum can be expressed in terms of the matrices $\boldsymbol{D}$ and $\boldsymbol{J}$ which collect bodies' inertia matrices and body Jacobians. Therefore,

$$\boldsymbol{P} = \boldsymbol{J}^{T}\boldsymbol{D}\boldsymbol{J}\dot{\boldsymbol{\Phi}}. \tag{4.45}$$

By expanding the expression of the generalized system momentum above, the momentum associated with each body can be explicitly shown with reference to its associated body Jacobian and individual inertia matrix,

$$\boldsymbol{P} = \begin{bmatrix} (\boldsymbol{J}_0^b)^T & (\boldsymbol{J}_1^b)^T & \dots & (\boldsymbol{J}_N^b)^T \end{bmatrix} \begin{bmatrix} \boldsymbol{D}_0 & 0 & \dots & 0 \\ 0 & \boldsymbol{D}_1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \boldsymbol{D}_N \end{bmatrix} \begin{bmatrix} \boldsymbol{J}_0^b \\ \boldsymbol{J}_1^b \\ \vdots \\ \boldsymbol{J}_N^b \end{bmatrix} \dot{\boldsymbol{\Phi}}, \tag{4.46}$$

which is simplified to

$$\boldsymbol{P} = \begin{bmatrix} (\boldsymbol{J}_0^b)^T & \dots & (\boldsymbol{J}_N^b)^T \end{bmatrix} \begin{bmatrix} \boldsymbol{D}_0 \boldsymbol{J}_0^b \dot{\boldsymbol{\Phi}} \\ \boldsymbol{D}_1 \boldsymbol{J}_1^b \dot{\boldsymbol{\Phi}} \\ \vdots \\ \boldsymbol{D}_N \boldsymbol{J}_N^b \dot{\boldsymbol{\Phi}} \end{bmatrix} = \begin{bmatrix} (\boldsymbol{J}_0^b)^T & \dots & (\boldsymbol{J}_N^b)^T \end{bmatrix} \begin{bmatrix} \boldsymbol{D}_0 \boldsymbol{V}_0^b \\ \boldsymbol{D}_1 \boldsymbol{V}_1^b \\ \vdots \\ \boldsymbol{D}_N \boldsymbol{V}_N^b \end{bmatrix}. \tag{4.47}$$

The vector $\boldsymbol{D}_k \boldsymbol{V}_k^b \in \mathbb{R}^6$ represents the body momentum of the k^{th} body, denoted by $\boldsymbol{P}_k^b$, about body k's center of mass and expressed in the body coordinate frame. The total momentum of the space manipulator system is simply the summation of the momenta. However, this summation is only valid provided that the momenta are expressed with respect to the spatial coordinate frame. That is,

$$\boldsymbol{P}^s = \sum_{k=0}^{N} \boldsymbol{P}_k^s \tag{4.48}$$

indicates the total spatial momentum of the system $\boldsymbol{P}^s$, when the body momenta expressed in equation (4.47) are transformed to the spatial coordinate frame. For body k this spatial momentum is denoted by $\boldsymbol{P}_k^s$ and it is related to the body momentum by

$$\boldsymbol{P}_k^s = \mathbf{Ad}_{g_{sk}}^{-T} \boldsymbol{P}_k^b. \tag{4.49}$$

As previously discussed in section 4.3, the first 6 rows of the inertia matrix correspond to the contributions of the base and manipulator motions on the system momentum expressed in the parameterization of the base. As this section aims to develop a relationship between the base and manipulator motions, only the first 6 rows of the inertia matrix are of

concern in the calculation of the total momentum. Based on the inertia matrix definition, the first 6 rows of $\bar{\boldsymbol{M}}$ correspond to the first 6 rows of the $\boldsymbol{J}^T$ matrix, and thus the first 6 columns of $\boldsymbol{J}_0^b$ (i.e. the first 6 rows of $(\boldsymbol{J}_0^b)^T$). Note that the first 6 columns of $\boldsymbol{J}_0^b$ are the only non-zero elements in the base's Jacobian since the base's motion is defined by six degrees of freedom ($l_0 = 6$). We denote the collection of the nonzero columns of the base's body Jacobian by the matrix $\bar{\boldsymbol{J}}_0^b$. That is,

$$\bar{\boldsymbol{J}}_0^b = \mathbf{Ad}_{\boldsymbol{g}_{s0}^{-1}(0)}[\mathbf{Ad}_1^6\boldsymbol{\xi}_1 \quad \mathbf{Ad}_2^6\boldsymbol{\xi}_2 \quad \dots \quad \mathbf{Ad}_6^6\boldsymbol{\xi}_6] \in \mathbb{R}^{6\times 6}. \tag{4.50}$$

Factoring out the Adjoint operator for the product of exponentials going from the first generalized coordinate to the sixth $\mathbf{Ad}_1^6$ and recalling the definition of the spatial Jacobian in (4.9),

$$\bar{\boldsymbol{J}}_0^b = \mathbf{Ad}_{\boldsymbol{g}_{s0}^{-1}(0)}\mathbf{Ad}_1^6[\boldsymbol{\xi}_1 \quad \mathbf{Ad}_{e^{\hat{\xi}_1\theta_l}}\boldsymbol{\xi}_2 \quad \dots \quad \mathbf{Ad}_{e^{\hat{\xi}_1\theta_l}\dots e^{\hat{\xi}_5\theta_5}}\boldsymbol{\xi}_6] =: \mathbf{Ad}_{\boldsymbol{g}_{s0}^{-1}}\bar{\boldsymbol{J}}_0^s \in \mathbb{R}^{6\times 6}. \tag{4.51}$$

Here, $\bar{\boldsymbol{J}}_0^s$ is the first 6 columns of the body 0's spatial Jacobian. Since for every body k in the chain the first 6 columns of the body Jacobian $\boldsymbol{J}_k^b$ correspond to the same screws but in body k's coordinate frame, the truncated Jacobian

$$\bar{\boldsymbol{J}}_k^b := \mathbf{Ad}_{\boldsymbol{g}_{sk}^{-1}}\bar{\boldsymbol{J}}_0^s \in \mathbb{R}^{6\times 6}, \qquad k = 1, \cdots, N \tag{4.52}$$

is forming the first 6 columns of $\boldsymbol{J}_k^b$. Hence, the first 6 rows of (4.47) reads

$$\begin{aligned}\bar{\boldsymbol{P}} :&= \begin{bmatrix}(\bar{\boldsymbol{J}}_0^s)^T\mathbf{Ad}_{\boldsymbol{g}_{s0}^{-1}}^T & (\bar{\boldsymbol{J}}_0^s)^T\mathbf{Ad}_{\boldsymbol{g}_{s1}^{-1}}^T & \dots & (\bar{\boldsymbol{J}}_0^s)^T\mathbf{Ad}_{\boldsymbol{g}_{sN}^{-1}}^T\end{bmatrix}\begin{bmatrix}\boldsymbol{P}_0^b\\ \boldsymbol{P}_1^b\\ \vdots\\ \boldsymbol{P}_N^b\end{bmatrix}\\ &= (\bar{\boldsymbol{J}}_0^s)^T(\mathbf{Ad}_{\boldsymbol{g}_{s0}^{-1}}^T\boldsymbol{P}_0^b + \mathbf{Ad}_{\boldsymbol{g}_{s1}^{-1}}^T\boldsymbol{P}_1^b + \dots + \mathbf{Ad}_{\boldsymbol{g}_{sN}^{-1}}^T\boldsymbol{P}_N^b) = (\bar{\boldsymbol{J}}_0^s)^T\sum_{k=0}^{N}\boldsymbol{P}_k^s = (\bar{\boldsymbol{J}}_0^s)^T\boldsymbol{P}^s.\end{aligned} \tag{4.53}$$

Referring back to the partitioning of the inertia matrix $\bar{\boldsymbol{M}}(\boldsymbol{\Phi})$ in (4.28), the total spatial momentum of the system is obtained by

$$\boldsymbol{P}^s(\boldsymbol{\Phi},\dot{\boldsymbol{\Phi}}) = (\bar{\boldsymbol{J}}_0^s)^{-T}\bar{\boldsymbol{P}} = (\bar{\boldsymbol{J}}_0^s)^{-T}[\boldsymbol{H}_o \quad \boldsymbol{H}_{om}] \begin{bmatrix} \dot{\boldsymbol{\theta}} \\ \dot{\boldsymbol{q}} \end{bmatrix}. \tag{4.54}$$

Under the aforementioned assumption that the space manipulator system is in an undisturbed environment, the total momentum of the system in the spatial coordinate frame is assumed constant and equal to $\mu \in \mathbb{R}^6$. Therefore, the conservation of momentum equation yields

$$\boldsymbol{H}_o\dot{\boldsymbol{\theta}} + \boldsymbol{H}_{om}\dot{\boldsymbol{q}} = (\bar{\boldsymbol{J}}_0^s)^T\boldsymbol{\mu}. \tag{4.55}$$

This equation may be considered as an affine nonholonomic constraint on the space manipulator motion. We denote the submanifold of the tangent bundle of the configuration manifold that is defined via this equation by

$$\mathscr{M} := (\boldsymbol{P}^s)^{-1}(\boldsymbol{\mu}). \tag{4.56}$$

This defines all possible states of the system in which the system can evolve in, when the total momentum of the system is conserved and equal to $\boldsymbol{\mu}$. Note that the case of $\boldsymbol{\mu} = 0$ has been extensively studied in the literature [68]. However, the system may not be in rest (either deliberately or non-deliberately) when beginning the capture process of a target object; thus signifying the importance of studying control scenarios involving systems with conserved non-zero momentum.

Chapter 5

Output Tracking Control of Free-Floating Space Manipulators

5.1 Summary

This chapter develops an output tracking controller to stabilize feasible end-effector pose trajectories of a free-floating space manipulator in an undisturbed environment. Two control cases are considered: the first and more trivial task involves performing output tracking control on only the linear end-effector motion, while the second task expands upon this to address the full pose control of the end-effector on $\mathrm{SE}(3)$. In both output tracking control strategies, feedback linearization is employed in the manipulator joint space to linearize the non-linear properties of the end-effector motion by defining an appropriate feedback transformation. This control approach evidently allows for linear control (e.g., PID control) implementation in the workspace of the space manipulator. Free-floating space manipulators are inherently underactuated with available control only in the joint space of the manipulator, and they posses conserved total spatial momentum. Taking advantage of the conserved quantities, a systematic procedure is developed to reduce the system dynamics at a non-zero momentum and rigorously express the equations of motion independent of the uncontrolled base motion by restricting the space of allowable velocities. Feedback linearization applied to the reduced dynamics results in a linear relation relating the input control torques to the end-effector motion that solely depends on the controlled states of the system (i.e., the manipulator joint motions). A classical PID controller is implemented to control the system's linear output, and a modified feedforward, feedback PID control law is used to stabilize any feasible trajectory (including both position and orientation) of the end-effector on $\mathrm{SE}(3)$. The following subsections carefully reduce the dynamics of the

space manipulator system with a non-zero conserved momentum, and subsequently define control laws derived from the input-output linearization on $\mathbb{R}^3$ and on the Lie algebra $\mathfrak{se}(3)$. Singularity accommodation in the designed controller is introduced using the singularity robust inverse method.

5.2 Dynamic Reduction

Considering a space manipulator in its free-floating regime, this section aims to reduce the system's dynamics by restricting the equations of motion to the space of controlled states. This implies developing a dynamic relationship between the manipulator torque $\boldsymbol{\tau}_m \in \mathbb{R}^N$ and only the motion of the manipulator joints. In Section 4.3, (4.43) summarizes the equations of motion involving both the base and manipulator under the influence of a manipulator torque. The following dynamic reduction process relates the motion of the base with that of the manipulator using the system's momentum. Substituting this relationship into equation (4.43) allows for an expression of the dynamical equations as only a function of the manipulator's motion. The dynamic reduction process is predicated on the assumption of an undisturbed environment, signifying that the system momentum is constant, however not necessarily zero. Consequently, we begin the reduction process by using the conserved linear and angular momentum presented in Section 4.4 to restrict the system velocities. From this conserved quantity, a non-linear relationship between the base and manipulator motions arises. Recalling the conservation of total momentum in equation (4.55), we can obtain this relation by isolating for the base velocity as follows,

$$\dot{\boldsymbol{\theta}} = \boldsymbol{H}_o^{-1}(\bar{\boldsymbol{J}}_0^s)^T\boldsymbol{\mu} - \boldsymbol{H}_o^{-1}\boldsymbol{H}_{om}\dot{\boldsymbol{q}} = \boldsymbol{B} - \boldsymbol{S}\dot{\boldsymbol{q}}, \tag{5.1}$$

where we define

$$\boldsymbol{S}(\boldsymbol{\Phi}) := \boldsymbol{H}_o^{-1}\boldsymbol{H}_{om}, \tag{5.2}$$

$$\boldsymbol{B}(\boldsymbol{\Phi};\boldsymbol{\mu}) := \boldsymbol{H}_o^{-1}(\bar{\boldsymbol{J}}_0^s)^T\boldsymbol{\mu}. \tag{5.3}$$

This relationship provides an expression for the velocity of the base in terms of the manipulator motion.

With the objective of the dynamic reduction being to remove all dependency on the base motion in the dynamic equations, the base acceleration (appearing in equation (4.43))

must also be expressed in terms of the manipulator joint accelerations. This implies taking the time derivative of the relationship between the base and manipulator joint velocities in equation (5.1). This time derivative expresses the base acceleration in the following form,

$$\ddot{\boldsymbol{\theta}} = \dot{\boldsymbol{B}} - \dot{\boldsymbol{S}}\dot{\boldsymbol{q}} - \boldsymbol{S}\ddot{\boldsymbol{q}}. \tag{5.4}$$

where the time derivatives $\dot{\boldsymbol{S}}$ and $\dot{\boldsymbol{B}}$ are defined as,

$$\dot{\boldsymbol{S}}(\boldsymbol{\Phi}, \dot{\boldsymbol{\Phi}}) = \boldsymbol{H}_o^{-1}\dot{\boldsymbol{H}}_{om} - \boldsymbol{H}_o^{-1}\dot{\boldsymbol{H}}_o\boldsymbol{H}_o^{-1}\boldsymbol{H}_{om}, \tag{5.5}$$

$$\dot{\boldsymbol{B}}(\boldsymbol{\Phi}, \dot{\boldsymbol{\Phi}}; \boldsymbol{\mu}) = -\boldsymbol{H}_o^{-1}\dot{\boldsymbol{H}}_o\boldsymbol{H}_o^{-1}(\bar{\boldsymbol{J}}_0^s)^T\boldsymbol{\mu} + \boldsymbol{H}_o^{-1}(\dot{\bar{\boldsymbol{J}}}_0^s)^T\boldsymbol{\mu}. \tag{5.6}$$

Similar to the partitioning introduced in equation (4.28), the time derivative of the inertia matrix components $\dot{\boldsymbol{H}}_{om}$ and $\dot{\boldsymbol{H}}_o$ are defined by partitioning the time derivative of the system inertia matrix $\dot{\bar{\boldsymbol{M}}}(\boldsymbol{\Phi}, \dot{\boldsymbol{\Phi}})$,

$$\dot{\bar{\boldsymbol{M}}}(\boldsymbol{\Phi}, \dot{\boldsymbol{\Phi}}) = \begin{bmatrix} \dot{\boldsymbol{H}}_o & \dot{\boldsymbol{H}}_{om} \\ \dot{\boldsymbol{H}}_{om}^T & \dot{\boldsymbol{H}}_m \end{bmatrix}. \tag{5.7}$$

The matrix $\dot{\bar{\boldsymbol{M}}}(\boldsymbol{\Phi}, \dot{\boldsymbol{\Phi}})$ is calculated based on (4.31), as detailed in Section 4.3.

Referring back to the matrix $\dot{\boldsymbol{B}}$, the time derivative of the truncated spatial Jacobian for the base ought to be determined. A similar computation presented in equation (4.35) must be performed to calculate $\dot{\bar{\boldsymbol{J}}}_0^s$, however for a Jacobian matrix in the spatial frame. Based on the definition of the spatial Jacobian in equation (4.9), the partial derivative of the j^{th} column of $\bar{\boldsymbol{J}}_0^s$ with respect to the i^{th} generalized coordinate Φ_i is calculated by the following relation,

$$\frac{\partial \bar{\boldsymbol{J}}_{0,j}^s}{\partial \Phi_i} = \frac{\partial (\mathbf{Ad}_1^{j-1})'}{\partial \Phi_i}\boldsymbol{\xi}_j. \tag{5.8}$$

Appendix E provides the derivation for the partial derivative of the Adjoint operator $(\mathbf{Ad}_1^{j-1})'$ in the expression above. The resulting partial derivative for the Adjoint operator above is as follows,

$$\frac{\partial(\mathbf{Ad}_1^{j-1})'}{\partial\Phi_i} = (\mathbf{Ad}_1^i)'\mathbf{ad}_{\boldsymbol{\xi}_i}(\mathbf{Ad}_{i+1}^{j-1})'. \quad 1 \leq i \leq (j-1) \tag{5.9}$$

For any $i > (j-1)$ this derivative is identically equal to zero. Consequently, the expression for the partial derivative of the column $\bar{\boldsymbol{J}}_{0,j}^s$ with respect to Φ_i becomes,

$$\frac{\partial \bar{\boldsymbol{J}}_{0,j}^s}{\partial\Phi_i} = (\mathbf{Ad}_1^i)'\mathbf{ad}_{\boldsymbol{\xi}_i}((\mathbf{Ad}_{i+1}^{j-1})'\boldsymbol{\xi}_j). \tag{5.10}$$

Since the first column of the spatial Jacobian expresses the unchanged twist $\boldsymbol{\xi}_1$ (including no Adjoint operator), and the spacecraft base contains 6 degrees of freedom, equation (5.10) applies to all columns $2 \leq j \leq 6$. Using equation (5.10), the time derivative $\dot{\bar{\boldsymbol{J}}}_0^s$ is computed as follows,

$$\dot{\bar{\boldsymbol{J}}}_0^s(\boldsymbol{\Phi}, \dot{\boldsymbol{\Phi}}) = \sum_{i=2}^{6} \frac{\partial \bar{\boldsymbol{J}}_0^s}{\partial\Phi_i}\dot{\Phi}_i. \tag{5.11}$$

Based on the matrices $\dot{\bar{\boldsymbol{M}}}(\boldsymbol{\Phi}, \dot{\boldsymbol{\Phi}})$ and $\dot{\bar{\boldsymbol{J}}}_0^s(\boldsymbol{\Phi}, \dot{\boldsymbol{\Phi}})$, a dependency on the base velocities is introduced in the relationship between the base and manipulator accelerations. This dependency on $\dot{\boldsymbol{\theta}}$ is removed through restricting to $\mathscr{M}$. This restriction implements equation (5.1) to express the base velocity with that of the manipulator in the $\dot{\boldsymbol{\Phi}}$ matrix. The same dependency on $\dot{\boldsymbol{\theta}}$ in system Coriolis matrix can analogously be removed by restricting to $\mathscr{M}$. That is, we define the following matrices to be independent of the base velocity,

$$\mathbb{S}(\boldsymbol{\Phi}, \dot{\boldsymbol{q}}; \mu) := \dot{\boldsymbol{S}}(\boldsymbol{\Phi}, [(\boldsymbol{B} - \boldsymbol{S}\dot{\boldsymbol{q}})^T \ \dot{\boldsymbol{q}}^T]^T), \tag{5.12}$$

$$\mathbb{B}(\boldsymbol{\Phi}, \dot{\boldsymbol{q}}; \mu) := \dot{\boldsymbol{B}}(\boldsymbol{\Phi}, [(\boldsymbol{B} - \boldsymbol{S}\dot{\boldsymbol{q}})^T \ \dot{\boldsymbol{q}}^T]^T), \tag{5.13}$$

$$\mathbb{C}(\boldsymbol{\Phi}, \dot{\boldsymbol{q}}; \mu) := \bar{\boldsymbol{C}}(\boldsymbol{\Phi}, [(\boldsymbol{B} - \boldsymbol{S}\dot{\boldsymbol{q}})^T \ \dot{\boldsymbol{q}}^T]^T), \tag{5.14}$$

where the matrix $\mathbb{C}(\boldsymbol{\Phi}, \dot{\boldsymbol{q}}; \mu)$ is partitioned as follows,

$$\mathbb{C}(\boldsymbol{\Phi}, \dot{\boldsymbol{q}}; \mu) = \begin{bmatrix} \mathbb{C}_o & \mathbb{C}_{om} \\ \mathbb{C}_{mo} & \mathbb{C}_m \end{bmatrix}. \tag{5.15}$$

Note that the equations of motion pertaining to the base in equation (4.42) are equivalent to the derived relationship between the base and manipulator accelerations stemming from the conservation of momentum. Hence by restricting the dynamics to $\mathscr{M}$, we can omit these equations from the set of dynamical equations of the space manipulator. The set of dynamical equations governing the motion of the manipulator can then be obtained by restricting equation (4.43) to $\mathscr{M}$. Substituting the relationships from the conservation of momentum (5.1) and (5.4) into equation (4.43) yields the following reduced set of dynamical equations independent of the base motion:

$$\boldsymbol{M}\ddot{\boldsymbol{q}} + \boldsymbol{C}\dot{\boldsymbol{q}} + \boldsymbol{E} = \boldsymbol{\tau}_m, \tag{5.16}$$

where,

$$\boldsymbol{M}(\boldsymbol{\Phi}) := \boldsymbol{H}_m - \boldsymbol{H}_{om}^T \boldsymbol{S}, \tag{5.17}$$

$$\boldsymbol{C}(\boldsymbol{\Phi}, \dot{\boldsymbol{q}}; \mu) := \mathbb{C}_m - \boldsymbol{H}_{om}^T \mathbb{S} - \mathbb{C}_{mo} \boldsymbol{S}, \tag{5.18}$$

$$\boldsymbol{E}(\boldsymbol{\Phi}, \dot{\boldsymbol{q}}; \mu) := \boldsymbol{H}_{om}^T \mathbb{B} + \mathbb{C}_{mo} \boldsymbol{B}. \tag{5.19}$$

The reduced equations of motion in equation (5.16) express the relationship between internal torques applied to the manipulator's joints and the resulting joint motions considering the contributions of the base motion. As mentioned, the resulting equations are independent of the base motion (i.e., $\dot{\boldsymbol{\theta}}$ and $\ddot{\boldsymbol{\theta}}$), however still depend on the base configuration $\boldsymbol{\theta}$. The base configuration is a measurable quantity of a space manipulator system and therefore does not need to be expressed in terms of the position of the manipulator joints. Following this reduction of the system dynamics, the resulting control law to linearize the double derivative of the system's output will solely depend on the controlled states of the system (i.e., the manipulator joint motions). The following subsections perform feedback linearization for the two output tracking control cases to derive an appropriate control torque which linearizes the desired end-effector motion.

5.3 Feedback Linearization for Linear Output Control

5.3.1 Zero Momentum System

Starting with the most trivial of control tasks, this subsection performs linear control of the end-effector motion for a free-floating space manipulator with constant zero momentum. The objective of performing feedback linearization in this instance is to obtain an input-output connection between the actuation torque applied to the manipulator $\boldsymbol{\tau}_m$, and the time derivatives of the end-effector's position. This control objective leads to the following problem statement for linear output tracking control:

Problem 1. *Consider the free-floating space manipulator reduced dynamics in* (5.16)*, and the end-effector position* $\boldsymbol{p}_N^s$ *as the output of the system. Given a twice differentiable desired trajectory* $\bar{\boldsymbol{p}}_N^s(t) \in \mathbb{R}^3$ *for the output, find* $\boldsymbol{\tau}_m$ *to make the desired output trajectory exponentially stable.*

We begin deriving the relationship between the manipulator torques and end-effector motion by establishing the body velocity of the end-effector based on the system's body Jacobian matrix,

$$\boldsymbol{V}_N^b = \boldsymbol{J}_N^b \begin{bmatrix} \dot{\boldsymbol{\theta}} \\ \dot{\boldsymbol{q}} \end{bmatrix} \in \mathbb{R}^6. \tag{5.20}$$

The differential kinematic relationship above may be partitioned by rows to demonstrate the Jacobian components pertaining to the linear and angular velocities ($\boldsymbol{v}_N^b$ and $\boldsymbol{\omega}_N^b$), as well as by columns to separate the influence from the base and manipulator configurations on the end-effector's velocity. Based on these partitionings, the differential kinematics mapping is expanded as follows,

$$\begin{bmatrix} \boldsymbol{v}_N^b \\ \boldsymbol{\omega}_N^b \end{bmatrix} = \begin{bmatrix} \boldsymbol{J}_{v_o}^b & \boldsymbol{J}_{v_m}^b \\ \boldsymbol{J}_{\omega_o}^b & \boldsymbol{J}_{\omega_m}^b \end{bmatrix} \begin{bmatrix} \dot{\boldsymbol{\theta}} \\ \dot{\boldsymbol{q}} \end{bmatrix}. \tag{5.21}$$

The 3×6 and $3 \times N$ matrices $\boldsymbol{J}_{v_o}^b$ and $\boldsymbol{J}_{v_m}^b$ represent the base and manipulator contributions (denoted by the subscripts o and m respectively) of the system Jacobian on the end-effector's linear velocity (denoted by the subscript v). In keeping with this subscript

notation, the remaining sub-matrices with the subscript ω form the contributions from the base and manipulator configurations on the angular end-effector motion in the body frame. Since the control task covered in this section focuses on controlling the linear portion of the end-effector's trajectory, only the first three rows of equation (5.21) are considered,

$$\boldsymbol{v}_N^b = [\boldsymbol{J}_{v_o}^b \quad \boldsymbol{J}_{v_m}^b] \begin{bmatrix} \dot{\boldsymbol{\theta}} \\ \dot{\boldsymbol{q}} \end{bmatrix} \in \mathbb{R}^3. \tag{5.22}$$

To express the end-effector's linear velocity in terms of only the controlled states, we restrict the system velocity vector to $\mathscr{M}$ and remove the dependency on the base motion. For the case of a zero momentum system ($\boldsymbol{\mu} = \mathbf{0}_{6\times 1}$), the relationship between the base and manipulator velocities in (5.1) reduces to,

$$\dot{\boldsymbol{\theta}} = -\boldsymbol{S}\dot{\boldsymbol{q}} \in \mathbb{R}^6, \tag{5.23}$$

where the matrix $\boldsymbol{S}$ is still defined based on the first six rows of the inertia matrix as shown in equation (5.2). Consequently, the output velocity in terms of only the manipulator motion is expressed as follows,

$$\boldsymbol{v}_N^b = [\boldsymbol{J}_{v_o}^b \quad \boldsymbol{J}_{v_m}^b] \begin{bmatrix} -\boldsymbol{S}\dot{\boldsymbol{q}} \\ \dot{\boldsymbol{q}} \end{bmatrix}. \tag{5.24}$$

Performing the matrix multiplication above results in the definition of the $3 \times N$ generalized Jacobian matrix, denoted by $\boldsymbol{J}_v^*$:

$$\boldsymbol{v}_N^b = (\boldsymbol{J}_{v_m}^b - \boldsymbol{J}_{v_o}^b \boldsymbol{S})\dot{\boldsymbol{q}} \in \mathbb{R}^3, \tag{5.25}$$

$$\boldsymbol{J}_v^* := \boldsymbol{J}_{v_m}^b - \boldsymbol{J}_{v_o}^b \boldsymbol{S} \in \mathbb{R}^{3\times N_{D_M}}. \tag{5.26}$$

The generalized Jacobian matrix provides a relationship depicting the influence of the manipulator joints' motion on the end-effector's linear velocity in the body frame. Note that the subscript v is again applied to the GJM to signify that we are only considering the linear portion of this relationship (i.e., the first three rows of $\boldsymbol{J}^*$). Note that the GJM

expressed in equation 5.50 is an extension of the original GJM definition in [68] (which includes a general Jacobian matrix mapping from $\mathbb{R}^n \to \mathbb{R}^n$) to provide a mapping from the manipulator's joint space in $\mathbb{R}^{N_{D_M}}$ to the actual output of the system in $\mathbb{R}^3$.

As explained in Section A.1.3, the resulting body frame velocity vector represents the velocity of the coordinate frame attached to the end-effector relative to the spatial coordinate frame, from the perspective of the body frame. In performing output tracking control, the goal is for the end-effector to follow a desired trajectory defined with respect to the spatial coordinate frame. By viewing the trajectory from the spatial frame, the associated tracking error has a physical significance with regards to the control objective. For the controlled system output and the desired trajectory to share a consistent frame of reference, the linear body velocity $\boldsymbol{v}_N^b$ must be transformed to the spatial frame. This transformation requires the rotation matrix from the spatial frame to the end-effector coordinate frame $\boldsymbol{R}_{sN}$ coming from the forward kinematics such that,

$$\boldsymbol{v}_N^s = \dot{\boldsymbol{p}}_{sN} = \boldsymbol{R}_{sN}\boldsymbol{v}_N^b = \boldsymbol{R}_{sN}\boldsymbol{J}_v^*\dot{\boldsymbol{q}}. \tag{5.27}$$

Note that we first define the end-effector's body velocity for subsequent transformation to the spatial frame (as opposed to initially using the spatial Jacobian) due to the fact that the velocity vector resulting from the spatial Jacobian does not express the correct output of the system. Recall that the spatial velocity provides the velocity of a point attached to the end-effector that is passing through the origin of the spatial frame. This spatial velocity is inconsistent with the initial definition of the system output (i.e., the motion of the coordinate frame attached to the end-effector). Therefore we must first define the end-effector velocity using the body Jacobian matrix (yielding the correct output of the system), which is then transformed to the spatial frame for consistency with the desired trajectory.

Continuing with the feedback linearization process, taking the time derivative of equation (5.27) provides a relationship between the linear acceleration of the end-effector and the manipulator joint accelerations,

$$\dot{\boldsymbol{v}}_N^s = \dot{\boldsymbol{R}}_{sN}\boldsymbol{J}_v^*\dot{\boldsymbol{q}} + \boldsymbol{R}_{sN}\dot{\boldsymbol{J}}_v^*\dot{\boldsymbol{q}} + \boldsymbol{R}_{sN}\boldsymbol{J}_v^*\ddot{\boldsymbol{q}}, \tag{5.28}$$

where the time derivatives of the GJM and rotation matrix are defined as follows,

$$\dot{\boldsymbol{J}}_v^* = \dot{\boldsymbol{J}}_{v_m}^b - \dot{\boldsymbol{J}}_{v_o}^b \boldsymbol{S} - \boldsymbol{J}_{v_o}^b \mathbb{S}, \tag{5.29}$$

$$\dot{\boldsymbol{R}}_{sN} = \boldsymbol{R}_{sN} \tilde{\boldsymbol{\omega}}_N^b = \boldsymbol{R}_{sN} \left((\boldsymbol{J}_{\omega_m} - \boldsymbol{J}_{\omega_o} \boldsymbol{S}) \dot{\boldsymbol{q}} \right)^{\sim}. \tag{5.30}$$

Note that the time derivatives of the body Jacobian sub-matrices are defined by $\dot{\boldsymbol{J}}_N^b$ based on (4.35) as follows,

$$\dot{\boldsymbol{J}}_N^b(\boldsymbol{\Phi}, \dot{\boldsymbol{\Phi}}) = \sum_{i=1}^{N_D} \frac{\partial \boldsymbol{J}_{k,j}^b}{\partial \Phi_i} \dot{\Phi}_i = \begin{bmatrix} \dot{\boldsymbol{J}}_{v_o}^b & \dot{\boldsymbol{J}}_{v_m}^b \\ \dot{\boldsymbol{J}}_{\omega_o}^b & \dot{\boldsymbol{J}}_{\omega_m}^b \end{bmatrix}. \tag{5.31}$$

The dependency on the base velocity in the derivative of the generalized Jacobian matrix (by way of $\dot{\boldsymbol{J}}_{v_o}$ and $\dot{\boldsymbol{J}}_{v_m}$) is again removed by restricting to $\mathscr{M}$. That is,

$$\mathbb{J}_{v_o}(\boldsymbol{\Phi}, \dot{\boldsymbol{q}}) := \dot{\boldsymbol{J}}_{v_o}^b(\boldsymbol{\Phi}, [(\boldsymbol{S}\dot{\boldsymbol{q}})^T \ \dot{\boldsymbol{q}}^T]^T) \in \mathbb{R}^{3\times 6}, \tag{5.32}$$

$$\mathbb{J}_{v_m}(\boldsymbol{\Phi}, \dot{\boldsymbol{q}}) := \dot{\boldsymbol{J}}_{v_m}^b(\boldsymbol{\Phi}, [(\boldsymbol{S}\dot{\boldsymbol{q}})^T \ \dot{\boldsymbol{q}}^T]^T) \in \mathbb{R}^{3\times N_{D_M}}, \tag{5.33}$$

such that,

$$\mathbb{J}_v^*(\boldsymbol{\Phi}, \dot{\boldsymbol{q}}) := \dot{\boldsymbol{J}}_v^*(\boldsymbol{\Phi}, [(\boldsymbol{S}\dot{\boldsymbol{q}})^T \ \dot{\boldsymbol{q}}^T]^T) = \mathbb{J}_{v_m} - \mathbb{J}_{v_o} \boldsymbol{S} - \boldsymbol{J}_{v_o}^b \mathbb{S} \in \mathbb{R}^{3\times N_{D_M}}. \tag{5.34}$$

The twice differentiated system output in equation (5.28) now presents an opportunity to obtain a relationship between the manipulator torque (i.e., the control input) and the end-effector's motion. Isolating for the joint accelerations in the reduced dynamic equations (equation (5.16)) for a system with zero momentum provides the following relationship,

$$\ddot{\boldsymbol{q}} = \boldsymbol{M}^{-1}(\boldsymbol{\tau}_m - \boldsymbol{C}\dot{\boldsymbol{q}}). \tag{5.35}$$

Substituting for the joint accelerations in equation (5.28) delivers the desired relationship between the linear end-effector motion and joint torques,

$$\dot{\boldsymbol{v}}_N^s = \boldsymbol{R}_{sN}\left((\boldsymbol{J}_{\omega_m} - \boldsymbol{J}_{\omega_o}\boldsymbol{S})\dot{\boldsymbol{q}}\right)^{\sim}\boldsymbol{J}_v^*\dot{\boldsymbol{q}} + \boldsymbol{R}_{sN}\mathbb{J}_v^*\dot{\boldsymbol{q}} + \boldsymbol{R}_{sN}\boldsymbol{J}_v^*(\boldsymbol{M}^{-1}(\boldsymbol{\tau}_m - \boldsymbol{C}\dot{\boldsymbol{q}})). \tag{5.36}$$

Note that only $\boldsymbol{\Phi}$ and $\dot{\boldsymbol{q}}$ appear in (5.36) as these are known quantities of a space manipulator system through sensor measurements. Feedback linearization can now be implemented to linearize the system's output by defining an appropriate input torque vector $\boldsymbol{\tau}_m$ to cancel the non-linearities in equation (5.36). Note that it is assumed that there is sufficient dexterity in the space manipulator to fully actuate the system in the context of linear output control. That is:

Assumption 1. *The manipulator system contains actuators imparting at least 3 independent motions.*

Given the relationship between $\boldsymbol{\tau}_m$ and $\dot{\boldsymbol{v}}_N^s$, and under Assumption 1, the following feedback transformation law removes all non-linearities in the system output:

$$\boldsymbol{\tau}_m = \boldsymbol{M}\boldsymbol{J}_v^{*\diamond}\boldsymbol{R}_{sN}^T(\boldsymbol{U}_v - \boldsymbol{R}_{sN}\left((\boldsymbol{J}_{\omega_m} - \boldsymbol{J}_{\omega_o}\boldsymbol{S})\dot{\boldsymbol{q}}\right)^{\sim}\boldsymbol{J}_v^*\dot{\boldsymbol{q}} - \boldsymbol{R}_{sN}\mathbb{J}_v^*\dot{\boldsymbol{q}}) + \boldsymbol{C}\dot{\boldsymbol{q}}, \tag{5.37}$$

such that the input-output relationship can be controlled using standard control techniques for linear systems,

$$\dot{\boldsymbol{v}}_N^s = \ddot{\boldsymbol{p}}_{sN} = \boldsymbol{U}_v. \tag{5.38}$$

Here, $\diamond$ represents the Singularity-Robust (SR) inverse of a matrix, defined for non-square matrices. The SR-inverse is implemented as a general case of Assumption 1, pertaining to redundant manipulator designs. For the case of a manipulator with exactly three independent motions, $\boldsymbol{J}_v^*$ becomes a 3×3 square matrix and equation (5.37) includes $\boldsymbol{J}_v^{*-1}$ in place of $\boldsymbol{J}_v^{*\diamond}$. Details regarding the formulation of the SR-inverse are included in Appendix C. Moreover, $\boldsymbol{U}_v \in \mathbb{R}^3$ is the vector of control inputs to the resulting closed-loop control system. For the case of linear control of the end-effector, $\boldsymbol{U}_v$ defines the output of a PID controller whose design is described in Appendix B.

5.3.2 Non-Zero Momentum System

Building upon the linear control case for a zero momentum system, this subsection performs the feedback linearization on a system with non-zero momentum. The primary differences in the derivation of the control torque stem from the addition of the matrix $\boldsymbol{B}$ when restricting the linearization process to $\mathscr{M}$. Again starting from the first three rows of the differential kinematics, and expressing the system velocity vector $\dot{\boldsymbol{\Phi}}$ only in terms of $\dot{\boldsymbol{q}}$, the linear body velocity of the end-effector is defined as follows,

$$\boldsymbol{v}_N^b = [\boldsymbol{J}_{v_o}^b \quad \boldsymbol{J}_{v_m}^b] \begin{bmatrix} \boldsymbol{B} - \boldsymbol{S}\dot{\boldsymbol{q}} \\ \dot{\boldsymbol{q}} \end{bmatrix}. \tag{5.39}$$

Performing the matrix multiplication yields the following expression of the linear portion of the body velocity for a system with non-zero momentum,

$$\boldsymbol{v}_N^b = (\boldsymbol{J}_{v_m}^b - \boldsymbol{J}_{v_o}^b \boldsymbol{S})\dot{\boldsymbol{q}} + \boldsymbol{J}_{v_o}^b \boldsymbol{B} \in \mathbb{R}^3, \tag{5.40}$$

$$\boldsymbol{J}_v^* := \boldsymbol{J}_{v_m}^b - \boldsymbol{J}_{v_o}^b \boldsymbol{S} \in \mathbb{R}^{3 \times N_{D_M}}. \tag{5.41}$$

The definition of the generalized Jacobian matrix is consistent with that for the zero momentum system case, however $\boldsymbol{v}_N^b$ has the additional dependency on the system's momentum by way of the matrix $\boldsymbol{B}$. The influence of the system's momentum is mapped to the motion of the end-effector through the base's configuration in the body Jacobian. For the reasons previously mentioned, the linear body velocity is subsequently rotated into the spatial reference frame using the $\boldsymbol{R}_{sN}$ rotation matrix:

$$\boldsymbol{v}_N^s = \boldsymbol{R}_{sN}(\boldsymbol{J}_v^* \dot{\boldsymbol{q}} + \boldsymbol{J}_{v_o}^b \boldsymbol{B}). \tag{5.42}$$

Differentiating the linear portion of the velocity above defines the linear acceleration of the end-effector with respect to the spatial frame. Again, the expression for $\dot{\boldsymbol{v}}_N^s$ introduces a relationship between the system's output and the manipulator joint accelerations. In the case of the space manipulator containing conserved non-zero momentum, the linear acceleration of the end-effector now includes additional terms predicated on the system's total momentum $\boldsymbol{\mu}$ (in comparison to equation (5.36)) via the matrices $\dot{\boldsymbol{E}}$, $\dot{\boldsymbol{B}}$, and $\mathbb{B}$,

$$\dot{\boldsymbol{v}}_N^s = \boldsymbol{R}_{sN}\boldsymbol{J}_v^*\ddot{\boldsymbol{q}} + (\dot{\boldsymbol{R}}_{sN}\boldsymbol{J}_v^* + \boldsymbol{R}_{sN}\mathbb{J}_v^*)\dot{\boldsymbol{q}} + (\dot{\boldsymbol{R}}_{sN}\boldsymbol{J}_{v_o}^b + \boldsymbol{R}_{sN}\mathbb{J}_{v_o})\boldsymbol{B} + \boldsymbol{R}_{sN}\boldsymbol{J}_{v_o}^b\mathbb{B}, \tag{5.43}$$

where $\dot{\boldsymbol{R}}_{sN}$ is substituted based on

$$\dot{\boldsymbol{R}}_{sN} = \boldsymbol{R}_{sN}\tilde{\boldsymbol{\omega}}_N^b = \boldsymbol{R}_{sN}\left((\boldsymbol{J}_{\omega_m} - \boldsymbol{J}_{\omega_o}\boldsymbol{S})\dot{\boldsymbol{q}} + \boldsymbol{J}_{\omega_o}\boldsymbol{B}\right)\tilde{}. \tag{5.44}$$

Substituting $\ddot{\boldsymbol{q}}$ from the reduced equations of motion in (5.16), including the matrix $\boldsymbol{E}$ (a function of the system momentum), into (5.43),

$$\begin{aligned}\dot{\boldsymbol{v}}_N^s = {} & \boldsymbol{R}_{sN}\boldsymbol{J}_v^*(\boldsymbol{M}^{-1}(\boldsymbol{\tau}_m - \boldsymbol{C}\dot{\boldsymbol{q}} - \boldsymbol{E})) + (\dot{\boldsymbol{R}}_{sN}\boldsymbol{J}_v^* + \boldsymbol{R}_{sN}\mathbb{J}_v^*)\dot{\boldsymbol{q}} + (\dot{\boldsymbol{R}}_{sN}\boldsymbol{J}_{v_o}^b + \boldsymbol{R}_{sN}\mathbb{J}_{v_o})\boldsymbol{B} \\ & + \boldsymbol{R}_{sN}\boldsymbol{J}_{v_o}^b\mathbb{B}.\end{aligned} \tag{5.45}$$

Based on the equation above and Assumption 1, the following manipulator torque along with (5.44) achieves feedback linearization,

$$\begin{aligned}\boldsymbol{\tau}_m = {} & \boldsymbol{M}\boldsymbol{J}^{*\diamond}\boldsymbol{R}_{sN}^T(\boldsymbol{U}_v - (\dot{\boldsymbol{R}}_{sN}\boldsymbol{J}_v^* + \boldsymbol{R}_{sN}\mathbb{J}_v^*)\dot{\boldsymbol{q}} - (\dot{\boldsymbol{R}}_{sN}\boldsymbol{J}_{v_o}^b + \boldsymbol{R}_{sN}\mathbb{J}_{v_o})\boldsymbol{B} - \boldsymbol{R}_{sN}\boldsymbol{J}_{v_o}^b\mathbb{B}) \\ & + \boldsymbol{C}\dot{\boldsymbol{q}} + \boldsymbol{E},\end{aligned} \tag{5.46}$$

resulting in the same linear input-output relationship as in the zero momentum case,

$$\dot{\boldsymbol{v}}_N^s = \ddot{\boldsymbol{p}}_{sN} = \boldsymbol{U}_v. \tag{5.47}$$

As previously mentioned, Appendix B details the design of the PID controller $\boldsymbol{U}_v$ defining the controlled behaviour of the linear end-effector motion. This includes the choice of the PID gains dictating the response of a third order linear system for which the end-effector motion shall follow.

5.4 Feedback Linearization for Full Pose Control

This section builds upon the linear control task by considering control over both the linear and angular motion of the end-effector. In this section, the derivation of the feedback

linearization control law is performed for the general case of a space manipulator system with constant non-zero momentum. For this control task, the space manipulator's output is evidently defined as the pose of the end-effector $\boldsymbol{g}_{sN}$. This culminates into the following problem statement summarizing the section's objective:

Problem 2. *Consider the free-floating space manipulator dynamics in* (4.42) *and* (4.43)*, and the position and orientation of the end-effector, i.e.,* $\boldsymbol{g}_{sN}$ *defined in* (4.4)*, as the output of the system. Given a twice differentiable desired trajectory* $\bar{\boldsymbol{g}}_{sN}(t) \in SE(3)$ *for the output, find* $\boldsymbol{\tau}_m$ *to make the desired output trajectory exponentially stable.*

The general method for performing feedback linearization outlined previously remains applicable when considering the full pose of the output. A relationship between the manipulator torque $\boldsymbol{\tau}_m$ and end-effector motion is again the fundamental objective for input-output linearization. Equivalent to section 5.3, this relationship is derived starting from the differential kinematics mapping to define end-effector's body velocity. All six rows of the system body Jacobian are now considered to include both the linear and angular velocity components of the output. Consequently, $\boldsymbol{J}_N^b$ is only partitioned by columns to portray the 6×6 and $6 \times N$ sub-matrices $\boldsymbol{J}_o^b$ and $\boldsymbol{J}_m^b$, representing the contributions from the base and manipulator configurations, respectively. We again restrict the system velocity vector to $\mathscr{M}$, leaving the end-effector's body velocity vector to be defined as follows,

$$\boldsymbol{V}_N^b = [\boldsymbol{J}_o^b \quad \boldsymbol{J}_m^b] \begin{bmatrix} \boldsymbol{B} - \boldsymbol{S}\dot{\boldsymbol{q}} \\ \dot{\boldsymbol{q}} \end{bmatrix}. \tag{5.48}$$

Performing the matrix multiplication above provides a similar definition of the generalized Jacobian matrix to that seen in equation (5.41), however with an additional three rows which consider the angular motion of the end-effector. From the following relationship for the end-effector's body velocity corresponding to a system with non-zero momentum,

$$\boldsymbol{V}_N^b = (\boldsymbol{J}_m^b - \boldsymbol{J}_o^b \boldsymbol{S})\dot{\boldsymbol{q}} + \boldsymbol{J}_o^b \boldsymbol{B}, \tag{5.49}$$

the $6 \times N$ generalized Jacobian matrix, denoted by $\boldsymbol{J}^*$, is defined as follows,

$$\boldsymbol{J}^* = \boldsymbol{J}_m^b - \boldsymbol{J}_o^b \boldsymbol{S}. \tag{5.50}$$

The GJM above again provides a mapping from the manipulator's joint space, however to the system's workspace in SE(3), in this case, without parameterizing the output at any point. Differentiating the end-effector's body velocity vector with respect to time allows for a relationship between the end-effector's motion and the joint accelerations,

$$\dot{\boldsymbol{V}}_N^b = \boldsymbol{J}^* \ddot{\boldsymbol{q}} + \mathbb{J}^* \dot{\boldsymbol{q}} + \mathbb{J}_o \boldsymbol{B} + \boldsymbol{J}_o \mathbb{B}, \tag{5.51}$$

where $\mathbb{J}^*(\boldsymbol{\Phi}, \dot{\boldsymbol{q}}; \mu)$ and $\mathbb{J}_o(\boldsymbol{\Phi}, \dot{\boldsymbol{q}}; \mu)$ are defined in the same manner as (5.32). Substituting $\ddot{\boldsymbol{q}}$ from the system's reduced dynamics in equation (5.16), and restricting to $\mathscr{M}$, yields the following relationship between the end-effector's motion and the joint torques,

$$\dot{\boldsymbol{V}}_N^b = \boldsymbol{J}^* \boldsymbol{M}^{-1} (\boldsymbol{\tau}_m - \boldsymbol{C}\dot{\boldsymbol{q}} - \boldsymbol{E}) + \mathbb{J}^* \dot{\boldsymbol{q}} + \mathbb{J}_o \boldsymbol{B} + \boldsymbol{J}_o \mathbb{B}. \tag{5.52}$$

Feedback linearization can now be implemented to linearize the system's output by defining an appropriate input torque $\boldsymbol{\tau}_m$ which cancels the non-linearities in equation (5.52). Analogous to Assumption 1, we again assume that the manipulator contains adequate actuation for controlling both the position and orientation of the end-effector. Controlling the six independent motions of the end-effector leads to the following assumption:

Assumption 2. *The manipulator system contains actuators imparting at least six independent motions.*

Given the relationship between $\boldsymbol{\tau}_m$ and $\dot{\boldsymbol{V}}_N^b$ in equation (5.52), and under Assumption 2, the following feedback control law:

$$\boldsymbol{\tau}_m = \boldsymbol{M}\boldsymbol{J}^{*\dagger} (\boldsymbol{U} - \mathbb{J}^* \dot{\boldsymbol{q}} - \mathbb{J}_o \boldsymbol{B} - \boldsymbol{J}_o \mathbb{B}) + \boldsymbol{C}\dot{\boldsymbol{q}} + \boldsymbol{E}, \tag{5.53}$$

linearizes the input-output relationship between the linear and angular end-effector motion and the output tracking controller for full pose control $\boldsymbol{U}$,

$$\dot{\boldsymbol{V}}_N^b = \boldsymbol{U}. \tag{5.54}$$

Here, $\boldsymbol{U} \in \mathbb{R}^6$ is the vector of control input in the resulting closed-loop control system, outlined in the following section. Moreover, the Moore-Penrose inverse is applied to the GJM as a general case of Assumption 2 for when the space robot's manipulator contains more than six degrees of freedom (thus introducing redundancy in the controlling the end-effector pose). For a manipulator designed with six degrees of freedom, the $\boldsymbol{J}^*$ matrix becomes square (6×6) and the natural matrix inverse can be performed. The following subsection describes the design of the full pose controller $\boldsymbol{U}$, predicated on a modification to the classical PID controller outlined in section B.

5.5 Full Pose Control Structure

In this section, a control law on the Lie group $\mathrm{SE}(3)$ is proposed which stabilizes any twice differentiable feasible trajectory $\bar{\boldsymbol{g}}_{sN}(t) \in \mathrm{SE}(3)$ of the space manipulator's end-effector. This output tracking controller builds upon a PID control design by implementing an additional feed-forward component to achieve Lyapunov stability. This section additionally defines the configuration error function and the velocity error corresponding to the system's output, necessary for developing the modified PID control structure. Finally, the proposed control law is proven to stabilize the full end-effector pose toward a feasible trajectory using a Lyapunov function based on the total energy of the error dynamics.

As mentioned previously, the output of the system is defined as the configuration of the end-effector, $\boldsymbol{g}_{sN} \in \mathrm{SE}(3)$ with associated body velocity expressed in the end-effector frame $\boldsymbol{V}_{sN}^b \in \mathfrak{se}(3)$. In this section, the dynamics of the output is considered following its linearization outlined in the previous section. Consequently, the motion of the end-effector is fully described by the control forces and torques imparted by $\boldsymbol{U}$. Thus, the relationship in equation (5.54) expresses the progression of the end-effector's body velocity in time.

5.5.1 Error Function

This subsection defines a positive definite error function to establish the configuration difference between the desired and actual end-effector trajectories. In this case, the error function is thus referred to as a configuration error function. This thesis considers a quadratic

error function going from $\mathrm{SE}(3) \to \mathbb{R}_{\geq 0}$, a positive scalar. Using group operation on the desired and actual end-effector configurations in $\mathrm{SE}(3)$, the output pose error $\boldsymbol{g}_e$ is defined as follows,

$$\boldsymbol{g}_e = \bar{\boldsymbol{g}}_{sN}^{-1}\boldsymbol{g}_{sN} := \begin{bmatrix} \boldsymbol{R}_e & \boldsymbol{p}_e \\ \boldsymbol{0} & 1 \end{bmatrix} \in \mathrm{SE}(3). \tag{5.55}$$

This representation of the error expresses the relative configuration of the actual end-effector with respect to the desired trajectory at an instance in time. Note that in practice, the end-effector's actual pose is estimated using the forward kinematics mapping presented in section 4.2 provided sensor measurements of the system's generalized coordinates. As a member of $\mathrm{SE}(3)$, the pose error contains components $(\boldsymbol{R}_e, \boldsymbol{p}_e)$ indicating the orientation and position errors respectively. The corresponding orientation error is defined from group operation on the desired and actual end-effector orientations ($\bar{\boldsymbol{R}}_{sN}$ and $\boldsymbol{R}_{sN}$) and the position error is simply the Euclidean distance between the desired and actual end-effector positions ($\bar{\boldsymbol{p}}_{sN}$ and $\boldsymbol{p}_{sN}$),

$$\boldsymbol{R}_e = \bar{\boldsymbol{R}}_{sN}^T\boldsymbol{R}_{sN} \in \mathrm{SO}(3), \tag{5.56}$$

$$\boldsymbol{p}_e = \bar{\boldsymbol{R}}_{sN}^T(\boldsymbol{p}_{sN} - \bar{\boldsymbol{p}}_{sN}) \in \mathbb{R}^3. \tag{5.57}$$

As mentioned, the configuration error function maps the group error $\boldsymbol{g}_e$ to a positive scalar. Let $\psi\colon \mathrm{SE}(3) \to \mathbb{R}_{\geq 0}$ be the error function which considers both orientation and position errors. We define the function $\psi(\boldsymbol{g}_e)$ as

$$\psi(\boldsymbol{g}_e) = \psi_1(\boldsymbol{p}_e) + \psi_2(\boldsymbol{R}_e), \tag{5.58}$$

based on the two positive definite error functions $\psi_1\colon \mathbb{R}^3 \to \mathbb{R}_{\geq 0}$ and $\psi_2\colon \mathrm{SO}(3) \to \mathbb{R}_{\geq 0}$ that are specified in the following. The positive definite error function associated with the position error is trivially defined as the quadratic norm of $\boldsymbol{p}_e$,

$$\psi_1(\boldsymbol{p}_e) = \frac{1}{2}||\boldsymbol{p}_e||^2 = \frac{1}{2}||\boldsymbol{p}_{sN} - \bar{\boldsymbol{p}}_{sN}||^2, \tag{5.59}$$

and the positive definite error function on the orientation is defined by Koditschek in [80] as follows,

$$\psi_2(\boldsymbol{R}_e) = \frac{1}{2}\text{tr}(\mathbf{1}_{3\times 3} - \boldsymbol{R}_e), \tag{5.60}$$

where tr($\boldsymbol{A}$) refers to the trace of a square matrix $\boldsymbol{A}$. Combining the position and orientation error functions in equations (5.59) and (5.60) yields the following positive definite error function associated with the group element $\boldsymbol{g}_e$ considered in this thesis,

$$\psi(\boldsymbol{g}_e) = \frac{1}{2}\text{tr}(\mathbf{1}_{3\times 3} - \boldsymbol{R}_e) + \frac{1}{2}||\boldsymbol{p}_e||^2. \tag{5.61}$$

This configuration error becomes crucial in designing the proportional and integral control actions in the output tracking control structure. In the following subsection, the end-effector's velocity error is defined for the subsequent formation of the controller's derivative action. Note that for simplicity in notation, we refer to the error function on $\boldsymbol{g}_e$ as just ψ.

5.5.2 Velocity Error

We now begin defining a compatible velocity error by defining the body velocity associated with the desired trajectory $\bar{\boldsymbol{V}}^b_{sN}$. From the kinematics of the end-effector, we define the reference body velocity of the output as $\hat{\bar{\boldsymbol{V}}}^b_{sN} = \bar{\boldsymbol{g}}^{-1}_{sN}\dot{\bar{\boldsymbol{g}}}_{sN} \in \mathfrak{se}(3)$, which expresses the desired body velocity in the coordinate frame of the desired trajectory. In comparing this desired end-effector velocity with the true motion of the end-effector, the quantities $\bar{\boldsymbol{V}}^b_{sN}$ and $\boldsymbol{V}^b_{sN}$ must be expressed in a consistent frame of reference to quantify the error. This introduces a transport map in the definition of the velocity error to project the desired body velocity into the coordinate frame attached to the end-effector.

Recall that the Adjoint operator transforms elements of $\mathfrak{se}(3)$ between two coordinate frames provided their relative homogeneous transformation, as demonstrated in equation (A.36). Thus, the Adjoint operator provides the necessary transport map to convert $\bar{\boldsymbol{V}}^b_{sN}$ into the frame of reference of the end-effector. Since the group error $\boldsymbol{g}_e$ represents the relative motion from the end-effector's frame to the desired trajectory frame, taking its inverse defines the required transformation from the desired trajectory to the body frame. Taking the Adjoint of $\boldsymbol{g}_e^{-1} = \boldsymbol{g}^{-1}_{sN}\bar{\boldsymbol{g}}_{sN}$ and pre-multiplying with $\bar{\boldsymbol{V}}^b_{sN}$ now expresses the actual

and desired body velocities in a shared reference frame. The velocity error $\boldsymbol{V}_e$ compatible with the group error $\boldsymbol{g}_e$ can now be defined as follows,

$$\boldsymbol{V}_e^b = \boldsymbol{V}_{sN}^b - \mathbf{Ad}_{\boldsymbol{g}_e^{-1}} \bar{\boldsymbol{V}}_{sN}^b \in \mathbb{R}^6. \tag{5.62}$$

Evidently, the body velocity error is similarly expressed with respect to the end-effector's body coordinate frame. With the definitions of the configuration error function in (5.61) and the velocity error associated with the output above, the following subsection introduces the output tracking control structure to drive $\boldsymbol{g}_e$ to the identity and $\boldsymbol{V}_e^b$ to zero.

5.5.3 Output Tracking Control Design

In this subsection, an output tracking controller is designed with reference to Problem 2. The developed control law is predicated on the combination of feedback terms and a feedforward term to satisfy the stability in the output tracking control task requested in Problem 2. The feedback terms involve proportional, integral, and derivative control actions which make use of the coordinate-free error function and velocity error definitions in the previous subsections. The modified feedforward, feedback PID control law thus contains the following actions to control the complete end-effector motion described in (5.54),

$$\boldsymbol{U} = \boldsymbol{U}_{pi} + \boldsymbol{U}_d + \boldsymbol{U}_{ff}. \tag{5.63}$$

Given a 6×6 symmetric positive definite matrix $\boldsymbol{K}_d$ expressing the derivative gain, the control action to dissipate the velocity error to zero is defined as follows,

$$\boldsymbol{U}_d = -\boldsymbol{K}_d \boldsymbol{V}_e^b = -\boldsymbol{K}_d (\boldsymbol{V}_{sN}^b - \mathbf{Ad}_{\boldsymbol{g}_e^{-1}} \bar{\boldsymbol{V}}_{sN}^b). \tag{5.64}$$

The 6×6 symmetric positive definite proportional and integral gains $\boldsymbol{K}_p$ and $\boldsymbol{K}_i$ are applied to drive the output's configuration error function ψ to zero, and accordingly $\boldsymbol{g}_e$ to the identity. In defining the proportional and integral control actions, the gradient of the error function in (5.61) must first be defined. This gradient $\nabla\psi$ becomes apparent from the time derivative of the error function which can be computed as follows,

$$\dot{\psi} = (\nabla\psi)^T \boldsymbol{V}_e^b := \begin{bmatrix} \boldsymbol{p}_e \\ \text{skew}(\boldsymbol{R}_e)^\vee \end{bmatrix}^T \boldsymbol{V}_e^b, \tag{5.65}$$

where $\text{skew}(\boldsymbol{A}) = \frac{1}{2}(\boldsymbol{A} - \boldsymbol{A}^T)$ for all 3×3 matrices $\boldsymbol{A}$, and the operator $\vee$ converts skew symmetric matrices to their vector representation (i.e., $\vee : \mathfrak{so}(3) \to \mathbb{R}^3$). The definition of the error function's gradient is demonstrated in the expression for $\dot{\psi}$ above, using the gradient definition on Riemannian manifolds [81]. The proportional and integral control actions are predicated on the gradient of the error function as follows,

$$\boldsymbol{U}_{pi} = -\boldsymbol{K}_p \nabla\psi - \boldsymbol{K}_i \boldsymbol{F}_i \tag{5.66}$$

$$\dot{\boldsymbol{F}}_i = \boldsymbol{K}_p \nabla\psi + \boldsymbol{K}_d \boldsymbol{V}_e^b. \tag{5.67}$$

As indicated by equation (5.67), $\boldsymbol{F}_i$ refers to the integral of the proportional and derivative actions. The feedforward component of the output tracking control law is designed in the manner which aids in the stability of the closed loop system. The impact of the feedforward will become apparent in the proof of Theorem 1. Based on the configuration and velocity error definitions previously defined, an appropriate structure for the output tracking feedforward action is as follows,

$$\boldsymbol{U}_{ff} = \mathbf{ad}_{\boldsymbol{V}_{sN}^b} \mathbf{Ad}_{\boldsymbol{g}_e^{-1}} \bar{\boldsymbol{V}}_{sN}^b + \mathbf{Ad}_{\boldsymbol{g}_e^{-1}} \dot{\bar{\boldsymbol{V}}}_{sN}^b. \tag{5.68}$$

The feedforward control action is necessary for performing trajectory following tasks, and defines the desired acceleration associated with the reference trajectory, expressed with respect to the body coordinate frame attached to the end-effector. Note that a stability analysis on the internal dynamics of free-floating space manipulator systems considered here is outside the scope of this thesis, leading to the following assumption:

Assumption 3. *The internal dynamics of the free-floating space manipulator system remains bounded if the output of the system $\boldsymbol{g}_{sN}$ is stable.*

Theorem 1. *Consider the dynamics of a free-floating space manipulator's end-effector in* (5.54) *and the position and orientation of the end-effector, i.e., $\boldsymbol{g}_{sN} \in SE(3)$, in* (4.4) *as the*

output of the system. Assume that the system is externally unperturbed and its total linear and angular momentum is conserved and equal to $\boldsymbol{\mu} \in \mathbb{R}^6$. *Let* $\bar{\boldsymbol{g}}_{sN}(t) \in SE(3)$ *be a twice differentiable feasible trajectory of the end-effector of the space manipulator.*

Provided the modified feedforward, feedback PID control law defined in equations (5.64)-(5.68) *in addition to the feedback linearization law in* (5.53)*, and under Assumptions 2 and 3, the desired trajectory* $\bar{\boldsymbol{g}}_{sN}(t)$ *is Lyapunov stable provided the following Lyapunov function* $V_L : \mathbb{R} \to \mathbb{R}_{\geq 0}$ *based on the total energy of the error dynamics:*

$$V_L(t) = \psi + \frac{1}{2}||\boldsymbol{V}_e^b||^2_{\boldsymbol{K}_p^{-1}} + \frac{1}{2}||\boldsymbol{F}_i + \boldsymbol{V}_e^b||^2_{\boldsymbol{K}_p^{-1}}, \tag{5.69}$$

where the quadratic norm of a vector $\boldsymbol{V} \in \mathbb{R}^n$ *subject to the metric* $\boldsymbol{K} \in \mathbb{R}^{n \times n}$ *is defined as,*

$$||\boldsymbol{V}||^2_{\boldsymbol{K}} = \boldsymbol{V}^T \boldsymbol{K} \boldsymbol{V}. \tag{5.70}$$

Proof. We prove the Lyapunov stability of the proposed output tracking control law by demonstrating that the time derivative of the Lyapunov candidate (5.69) along the trajectories of the system is negative semidefinite. In this process, the time derivatives of the three terms in (5.69) are analyzed separately. Recall that the time derivative of the error function has been previously established in equation (5.65) to produce an expression for $\boldsymbol{\psi}$. Thus, we begin the derivation process by taking the time derivative of the second term in (5.69) which expresses the kinetic energy of the error dynamics. Based on the vector norm definition in (5.70) and symmetry,

$$\frac{d}{dt}\left(\frac{1}{2}||\boldsymbol{V}_e^b||^2_{\boldsymbol{K}_p^{-1}}\right) = \boldsymbol{V}_e^{bT} \boldsymbol{K}_p^{-1} \dot{\boldsymbol{V}}_e^b. \tag{5.71}$$

The time derivative of the velocity error makes use of the linearized end-effector dynamics in (5.54), and demonstrates the significance of the feedforward control action's structure. We perform the derivative of the velocity error in (5.62) as follows,

$$\begin{aligned}\dot{\boldsymbol{V}}_e &= \frac{d}{dt}\left(\boldsymbol{V}_{sN}^b - \mathbf{Ad}_{\boldsymbol{g}_e^{-1}}\bar{\boldsymbol{V}}_{sN}^b\right)\\ &= \dot{\boldsymbol{V}}_{sN}^b - \frac{d}{dt}\left(\mathbf{Ad}_{\boldsymbol{g}_e^{-1}}\right)\bar{\boldsymbol{V}}_{sN}^b - \mathbf{Ad}_{\boldsymbol{g}_e^{-1}}\dot{\bar{\boldsymbol{V}}}_{sN}^b\\ &= (\boldsymbol{U}_{ff}+\boldsymbol{U}_{pi}+\boldsymbol{U}_d) - \mathbf{ad}_{\boldsymbol{V}_{sN}^b}\mathbf{Ad}_{\boldsymbol{g}_e^{-1}}\bar{\boldsymbol{V}}_{sN}^b - \mathbf{Ad}_{\boldsymbol{g}_e^{-1}}\dot{\bar{\boldsymbol{V}}}_{sN}^b\\ &= \boldsymbol{U}_{pi}+\boldsymbol{U}_d. \end{aligned} \tag{5.72}$$

Moreover, the time derivative of the third term in (5.69) is computed by virtue of the chain rule as follows,

$$\begin{aligned}\frac{d}{dt}\left(\frac{1}{2}||\boldsymbol{F}_i+\boldsymbol{V}_e^b||^2_{\boldsymbol{K}_p^{-1}}\right) &= (\boldsymbol{F}_i+\boldsymbol{V}_e^b)^T\boldsymbol{K}_p^{-1}(\dot{\boldsymbol{F}}_i+\dot{\boldsymbol{V}}_e^b)\\ &= \boldsymbol{F}_i^T\boldsymbol{K}_p^{-1}\dot{\boldsymbol{F}}_i+\boldsymbol{F}_i^T\boldsymbol{K}_p^{-1}\dot{\boldsymbol{V}}_e^b+(\boldsymbol{V}_e^b)^T\boldsymbol{K}_p^{-1}\dot{\boldsymbol{F}}_i+(\boldsymbol{V}_e^b)^T\boldsymbol{K}_p^{-1}\dot{\boldsymbol{V}}_e^b,\end{aligned} \tag{5.73}$$

which can be simplified by including the expressions for $\dot{\boldsymbol{F}}_i$ and $\dot{\boldsymbol{V}}_e^b$ from equations (5.67) and (5.72). We analyze each of the four terms above separately in the following:

$$\begin{aligned}\boldsymbol{F}_i^T\boldsymbol{K}_p^{-1}\dot{\boldsymbol{F}}_i &= \boldsymbol{F}_i^T\boldsymbol{K}_p^{-1}(\boldsymbol{K}_p\nabla\psi+\boldsymbol{K}_d\boldsymbol{V}_e^b)\\ &= \boldsymbol{F}_i\nabla\psi+\boldsymbol{F}_i^T\boldsymbol{K}_p^{-1}\boldsymbol{K}_d\boldsymbol{V}_e^b,\end{aligned} \tag{5.74}$$

$$\begin{aligned}\boldsymbol{F}_i^T\boldsymbol{K}_p^{-1}\dot{\boldsymbol{V}}_e^b &= \boldsymbol{F}_i^T\boldsymbol{K}_p^{-1}(\boldsymbol{U}_{pi}+\boldsymbol{U}_d)\\ &= \boldsymbol{F}_i^T\boldsymbol{K}_p^{-1}(-\boldsymbol{K}_p\nabla\psi-\boldsymbol{K}_i\boldsymbol{F}_i-\boldsymbol{K}_d\boldsymbol{V}_e^b)\\ &= -\boldsymbol{F}_i^T\nabla\psi-\boldsymbol{F}_i^T\boldsymbol{K}_p^{-1}\boldsymbol{K}_i\boldsymbol{F}_i-\boldsymbol{F}_i^T\boldsymbol{K}_p^{-1}\boldsymbol{K}_d\boldsymbol{V}_e^b,\end{aligned} \tag{5.75}$$

$$(\boldsymbol{V}_e^b)^T\boldsymbol{K}_p^{-1}\dot{\boldsymbol{F}}_i = (\boldsymbol{V}_e^b)^T\nabla\psi+(\boldsymbol{V}_e^b)^T\boldsymbol{K}_p^{-1}\boldsymbol{K}_d\boldsymbol{V}_e^b, \tag{5.76}$$

$$(\boldsymbol{V}_e^b)^T\boldsymbol{K}_p^{-1}\dot{\boldsymbol{V}}_e^b = -(\boldsymbol{V}_e^b)^T\nabla\psi-(\boldsymbol{V}_e^b)^T\boldsymbol{K}_p^{-1}\boldsymbol{K}_i\boldsymbol{F}_i-(\boldsymbol{V}_e^b)^T\boldsymbol{K}_p^{-1}\boldsymbol{K}_d\boldsymbol{V}_e^b. \tag{5.77}$$

Implementing the expanded interpretations above for each term in (5.73) yields,

$$\frac{d}{dt}\left(\frac{1}{2}||\boldsymbol{F}_i+\boldsymbol{V}_e^b||^2_{\boldsymbol{K}_p^{-1}}\right) = -\boldsymbol{F}_i^T\boldsymbol{K}_p^{-1}\boldsymbol{K}_i\boldsymbol{F}_i-(\boldsymbol{V}_e^b)^T\boldsymbol{K}_p^{-1}\boldsymbol{K}_i\boldsymbol{F}_i \tag{5.78}$$

Combining the derivatives in equations (5.65), (5.71) and (5.78), we derive the expression for $\dot{V}_L$ as follows,

$$\begin{aligned}\dot{V}_L &= (\nabla\psi)^T\boldsymbol{V}_e^b+(\boldsymbol{V}_e^b)^T\boldsymbol{K}_p^{-1}(\boldsymbol{U}_{pi}+\boldsymbol{U}_b)+(-\boldsymbol{F}_i^T\boldsymbol{K}_p^{-1}\boldsymbol{K}_i\boldsymbol{F}_i-(\boldsymbol{V}_e^b)^T\boldsymbol{K}_p^{-1}\boldsymbol{K}_i\boldsymbol{F}_i)\\ &= (\nabla\psi)^T\boldsymbol{V}_e^b+(\boldsymbol{V}_e^b)^T\boldsymbol{K}_p^{-1}(-\boldsymbol{K}_p\nabla\psi-\boldsymbol{K}_i\boldsymbol{F}_i-\boldsymbol{K}_d\boldsymbol{V}_e^b)-\boldsymbol{F}_i^T\boldsymbol{K}_p^{-1}\boldsymbol{K}_i\boldsymbol{F}_i-(\boldsymbol{V}_e^b)^T\boldsymbol{K}_p^{-1}\boldsymbol{K}_i\boldsymbol{F}_i\\ &= -(\boldsymbol{V}_e^b)^T\boldsymbol{K}_p^{-1}\boldsymbol{K}_i\boldsymbol{F}_i-(\boldsymbol{V}_e^b)^T\boldsymbol{K}_p^{-1}\boldsymbol{K}_d\boldsymbol{V}_e^b-\boldsymbol{F}_i^T\boldsymbol{K}_p^{-1}\boldsymbol{K}_i\boldsymbol{F}_i-(\boldsymbol{V}_e^b)^T\boldsymbol{K}_p^{-1}\boldsymbol{K}_i\boldsymbol{F}_i\\ &= -(\boldsymbol{V}_e^b)^T\boldsymbol{K}_p^{-1}\boldsymbol{K}_d\boldsymbol{V}_e^b-2(\boldsymbol{V}_e^b)^T\boldsymbol{K}_p^{-1}\boldsymbol{K}_i\boldsymbol{F}_i-\boldsymbol{F}_i^T\boldsymbol{K}_p^{-1}\boldsymbol{K}_i\boldsymbol{F}_i,\end{aligned} \tag{5.79}$$

which can be written in matrix form:

$$\dot{V}_L = \begin{bmatrix}\boldsymbol{F}_i^T & \boldsymbol{V}_e^T\end{bmatrix}\begin{bmatrix}\boldsymbol{K}_p^{-1}\boldsymbol{K}_i & \boldsymbol{K}_p^{-1}\boldsymbol{K}_i\\ \boldsymbol{K}_p^{-1}\boldsymbol{K}_i & \boldsymbol{K}_p^{-1}\boldsymbol{K}_d\end{bmatrix}\begin{bmatrix}\boldsymbol{F}_i\\ \boldsymbol{V}_e\end{bmatrix}. \tag{5.80}$$

Provided that $\boldsymbol{K}_d > \boldsymbol{K}_i$, and due to the fact that control gain matrices are positive definite, the time derivative of the Lyapunov function is negative semidefinite.

□

Remark 1. *The proposed output-tracking control law in* (5.5.3) *and* (5.53) *is only defined for an open neighborhood of the identity of* $SE(3)$*, where the exponential mapping is a diffeomorphism.*

5.6 Output Tracking Controller Block Diagram

In this section, the dynamics reduction, feedback linearization, and output tracking control structures detailed in this chapter are implemented in a block diagram to illustrate the interconnections of these processes. Outlining the complete structure of the output tracking controller in a trajectory following task is beneficial when simulating such scenarios, as will be discussed in the following chapter.The block diagram in Figure 5.1 depicts the

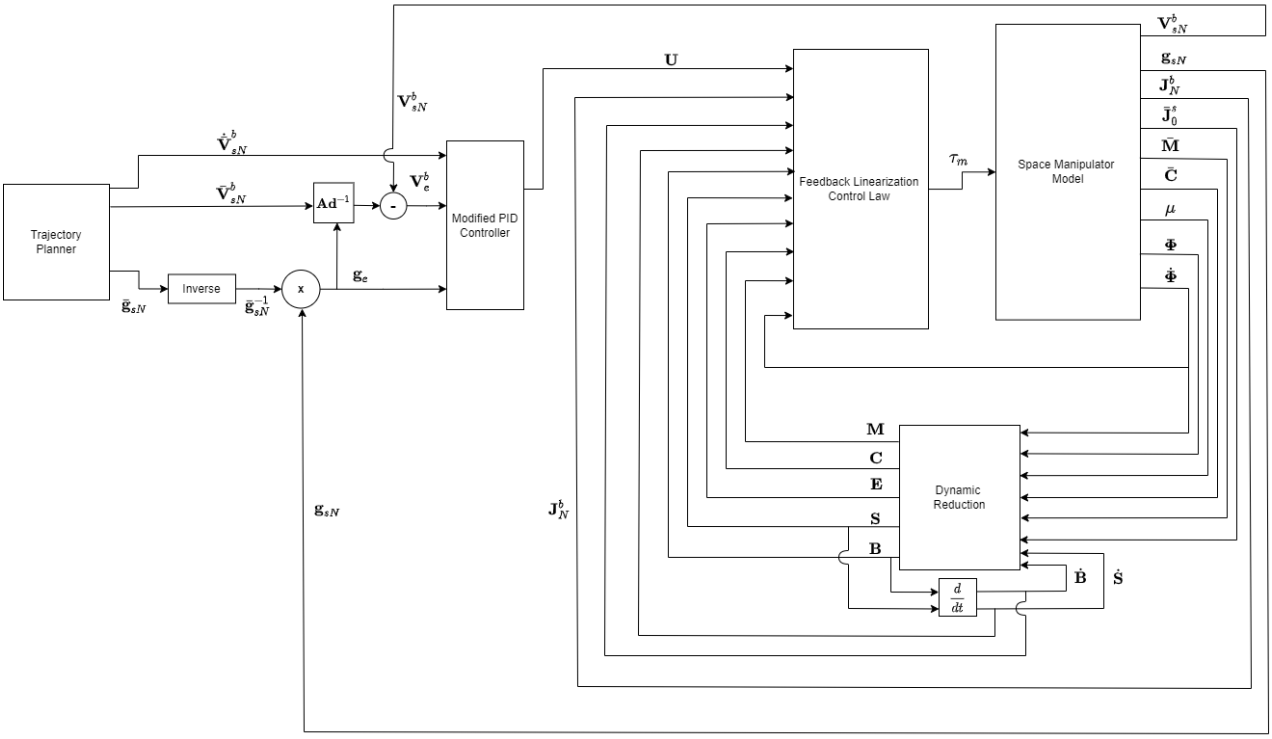


Figure 5.1: Output tracking controller block diagram

block structure of an output trajectory following task involving the modified feedforward, feedback PID full pose controller. The same block diagram structure remains valid for the linear end-effector control task, however with slight adjustments to the inputs and outputs of certain blocks. The path following process begins with a feasible desired trajectory for the output, generated by a trajectory planner. As mentioned in Problem 2, the trajectory planner provides the desired end-effector pose $\bar{\boldsymbol{g}}_{sN} \in \mathrm{SE}(3)$, body velocity vector with respect to the body frame $\bar{\boldsymbol{V}}^b_{sN} \in \mathbb{R}^6$, and corresponding body acceleration vector $\dot{\bar{\boldsymbol{V}}}^b_{sN} \in \mathbb{R}^6$. In the case of linear output control, these signals simply change to the desired linear position, velocity, and acceleration of the output.

To determine a control action which guides the system's output along the desired path, the configuration and velocity errors defined in section 5.5 must be established. The controller block diagram defines the group error signal $\boldsymbol{g}_e$ based on equation (5.55) using the inverse of the desired pose from the trajectory planner and the end-effector pose from the system's kinematics. The group error signal is subsequently input directly into the *Modified PID Controller* block for computing the configuration error function in equation (5.61). Regarding the output's velocity error, Figure 5.1 depicts the definition in equation (5.62)

through multiplication of desired body velocity signal with that output by the $\mathbf{Ad}^{-1}$ block. With the input to the Adjoint inverse block being the group error, the resulting output signal represents the transport element $\mathbf{Ad}_{\boldsymbol{g}_e^{-1}}$ to convert the desired body velocity into the end-effector frame. Subtracting the product of the aforementioned signals with the actual end-effector velocity output by the space manipulator model consequently defines the velocity error input to the controller.

Provided the full pose output tracking control law defined in subsection 5.5.3, the *Modified PID Controller* block additionally requires the desired body acceleration vector as an input. This block defines the output tracking control law using equations (5.64)-(5.68) to obtain a control action $\boldsymbol{U}$. Note for linear output control, the *Modified PID Controller* is replaced with the classical PID control structure detailed in section B. In this case, the inputs to the controller block are simply the linear tracking and velocity errors defined by equations (B.3) and (B.4), with the subsequent output control action $\boldsymbol{U}_v$ defined by equation (B.5).

The trajectory following process subsequently linearizes the end-effector's motion using the feedback linearization method outlined in section 5.4. The resulting feedback linearizing control law is predicated on the space manipulator's reduced equations of motion, as indicated by the inputs to the *Feedback Lineararization Control Law* block in Figure 5.1. Based on the system's dynamics and kinematics output by the *Space Manipulator Model* block, the system's equations of motion are restricted to the manipulator joint space in the *Dynamic Reduction* block. In this process, the matrices $\boldsymbol{B}$ and $\boldsymbol{S}$ resulting from the restriction to $\mathscr{M}$ in equation (5.1) require feedback of the system's inertia matrix $\bar{M}$, momentum $\boldsymbol{\mu}$, and truncated base Jacobian $\bar{\boldsymbol{J}}_0^b$. The *Dynamic Reduction* block additionally requires feedback of the system's states ($\boldsymbol{\Phi}$ and $\dot{\boldsymbol{\Phi}}$) and Coriolis matrix $\bar{\boldsymbol{C}}$, as well as the time derivatives $\dot{\boldsymbol{B}}$ and $\dot{\boldsymbol{S}}$ to compute the matrices $\boldsymbol{M}$, $\boldsymbol{C}$, and $\boldsymbol{E}$ in the reduced dynamics (equations (5.17)-(5.19)). Note that this dynamic reduction process is independent of the output tracking control task and hence applies to both the linear and full pose cases.

The remaining dependency in the feedback linearization control law comes from the generalized Jacobian matrix for which the system's body Jacobian matrix $\boldsymbol{J}_N^b$ is required as an input. The *Feedback Linearization* block thus computes the GJM using equation (5.50), and subsequently tests the system's manipulability using the metric in presented in equation (C.5). The inverse of the GJM is then performed according to the SR-inverse described in section C. Finally, the manipulator torque $\boldsymbol{\tau}_m$ in computed via equation (5.53) as the sole output of the block. This control action is applied to the space manipulator system which

subsequently moves accordingly. The following chapter, in conjunction with the principles established in chapter 4, details the methods involved in replicating the motion of a free-floating space manipulator system in a simulation setting.

Chapter 6

Space Manipulator Simulation Platform

6.1 Summary

This chapter details the free-floating space manipulator simulation platform constructed in Python 3, developed for simulating the behaviour of free-floating multi-body systems. The simulation implements the dynamic and kinematic models on Lie groups presented in Chapter 4, as well as the output tracking controllers based on feedback linearization derived in Chapter 5. The overall structure of the simulation is first described to demonstrate the specific functions and signal connections necessary for simulating the motion of a free-floating system in orbit. The specific algorithms associated with the dynamic and kinematic modellings are provided and discussed, followed by the algorithmic implementation of the proposed output tracking control structures. These algorithms are carefully developed for optimizing the computational efficiency of the space manipulator's kinematics, dynamics and control to achieve a near real-time simulation. Subsequently, the specific multi-body system (predicated on the design of space manipulators proposed in Chapter 3) simulated in this thesis is fully described. The kinematic and dynamic models implemented in Python are verified by comparing the simulated motion with an equivalent space manipulator system constructed in the Simscape platform provided by MATLAB & Simulink. This chapter describes the validation process and provides the corresponding validation results which confirm the forward kinematics, differential kinematics, and dynamical equations of motion included in the Python simulator. The future directions of the Python simulator in the context of implementing machine learning techniques are finally discussed.

6.2 Simulation Design

This section presents the overall simulation structure for modelling the motion of a free-floating space manipulator in the orbital environment. The high level objective of the simulation platform is to replicate a docking manoeuvre between a single-arm servicing satellite (in its free-floating regime) and a target object (either cooperative or non-cooperative) in space. The simulator provides a testing platform for evaluating the efficacy of any developed guidance, navigation, and control technology for a mission executed by a space manipulator system to capture both fixed and moving targets. The simulation consists of five main sections following the initialization to reflect the simulated task. These five sections correspond to the trajectory planning, output tracking controller, kinematics, dynamics, and numerical integration. Figure 6.1 conveys the inputs, outputs, and interconnections of these simulation components by illustrating the overall block diagram of the simulator. Note that in this case, the output tracking controller block contains the block diagram corresponding to the proposed feedback linearization control design in Figure 5.1. That is, combining Figure 6.1 with the in-depth block diagram for the output tracking controller in Figure 5.1 provides the complete end to end structure of the simulator.

As previously mentioned, an initialization procedure is required prior to starting the trajectory following process depicted in Figure 6.1. As part of the initialization, the design of the manipulator (i.e., the number of links N and joint types), the physical properties of the whole space manipulator system (e.g., mass and geometric design of each body in the system), the locations of all body coordinate frames with respect to the spatial coordinate frame (i.e., all $\boldsymbol{g}_{sk}(0)$'s), the initial system configuration, and initial system velocity are all established. The initialization process subsequently calculates each body's individual inertia matrix to define the system's $\boldsymbol{D}$ matrix, as well as every twist in the system, since these are constant throughout the simulation. Additionally, the Adjoint matrices of each initial transformation $\mathbf{Ad}_{\boldsymbol{g}_{sk}(0)}$ and the adjoint of each system twist $\mathbf{ad}_{\boldsymbol{\xi}_i}$ are likewise constant and therefore computed and stored in the initialization. Lastly, the system's momentum is defined in the initialization as it is a conserved quantity throughout the simulation.

Following the initialization, the simulation scenario runs for a user-specified duration following a user-specified timestep. Referring back to the simulation block diagram, the process is shown to start from the trajectory planner. Note that the output signals depicted in Figure 6.1 specifically correspond to the case of full end-effector pose control. In this case, the desired end-effector's pose $\bar{\boldsymbol{g}}_{sN}$ and its corresponding body velocity $\bar{\boldsymbol{V}}_N^b$ and body

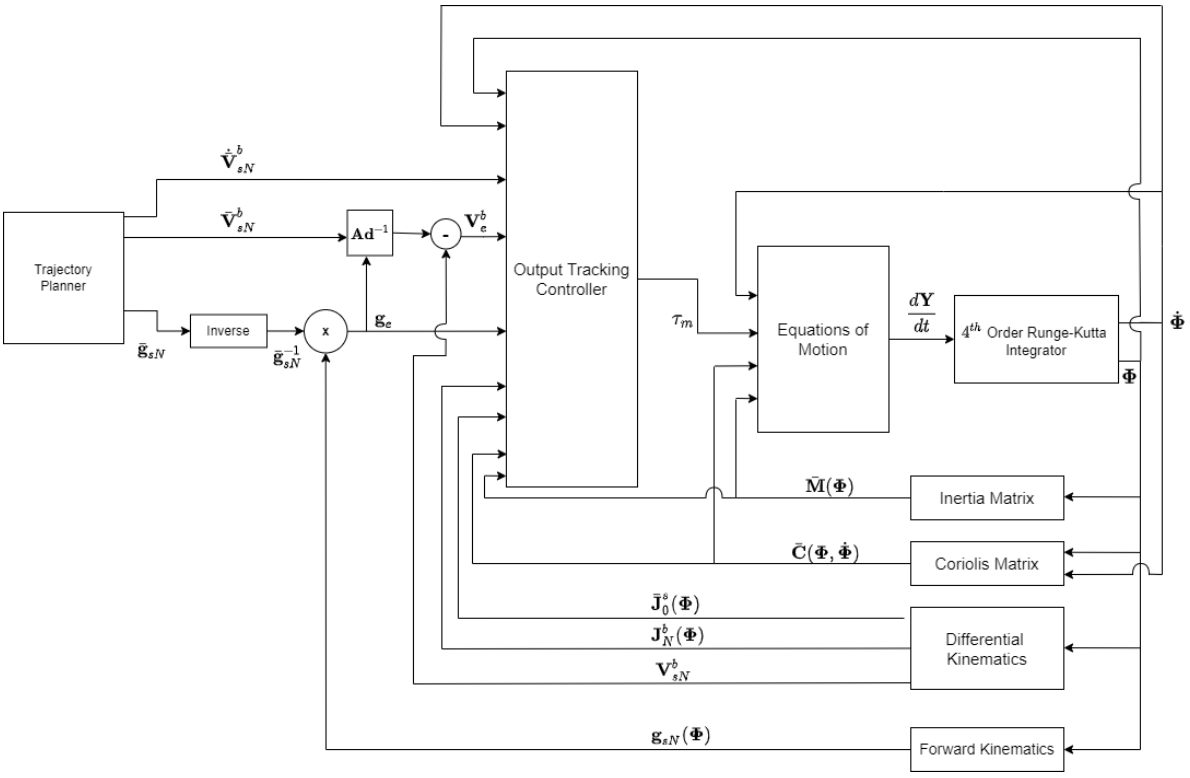


Figure 6.1: Space manipulator simulation block diagram

acceleration $\dot{\boldsymbol{V}}_N^b$ are output by the trajectory planner (analogous to the controller diagram in Figure 5.1). For the case of linear output control, a desired linear end-effector motion in the spatial frame, i.e., $\boldsymbol{p}_{sN}$, $\bar{\boldsymbol{v}}_N^s$, and $\dot{\bar{\boldsymbol{v}}}_N^s$, are output. Combining the inverse of the desired pose with the actual end-effector pose output by the system's forward kinematics generates the pose error for the output tracking controller. Referring back to the control structure in Figure 5.1, the output tracking control block additionally depends on the desired end-effector acceleration and velocity, the actual end-effector body velocity, and the end-effector and truncated base Jacobian matrices to generate a control torque $\boldsymbol{\tau}_m$.

This chapter now focuses on the simulation of the dynamics and kinematics of the space manipulator system. Under the influence of the manipulator control torque, the subsequent simulation blocks propagate the corresponding motion of the dynamic system. Evidently, the motion propagation requires the Euler-Lagrange equations of motion presented in (4.40), forming the basis of the *Equations of Motion* block in Figure 6.1. As such, the system inertia and Coriolis matrices, and the velocity of the generalized coordinates are necessary inputs. The resulting output defines a system of ordinary differential equations (ODEs) to be numerically integrated for $\boldsymbol{Y}$, including the system's degrees of freedom and their velocities,

$$\boldsymbol{Y} = \begin{bmatrix} \dot{\boldsymbol{\Phi}} \\ \boldsymbol{\Phi} \end{bmatrix} \in \mathbb{R}^{2N_D \times 1}, \tag{6.1}$$

such that,

$$\frac{d\boldsymbol{Y}}{dt} = \begin{bmatrix} \ddot{\boldsymbol{\Phi}} \\ \dot{\boldsymbol{\Phi}} \end{bmatrix} = \begin{bmatrix} \bar{\boldsymbol{M}}(\boldsymbol{\Phi})^{-1}(\boldsymbol{\tau} - \bar{\boldsymbol{C}}(\boldsymbol{\Phi}, \dot{\boldsymbol{\Phi}})\dot{\boldsymbol{\Phi}}) \\ \dot{\boldsymbol{\Phi}} \end{bmatrix} \in \mathbb{R}^{2N_D \times 1}. \tag{6.2}$$

The vector $\boldsymbol{Y}$ is obtained by numerically integrating this set of ODEs using the 4^{th} order Runge-Kutta method, as indicated by Figure 6.1. Based on the updated configuration and velocity, the system's dynamics and kinematics are likewise updated to capture the system's motion following the torque application. These updates are performed by the inertia matrix, Coriolis matrix, differential kinematics, and forward kinematics blocks shown in Figure 6.1. The resulting differential kinematics, and dynamic-related matrices $\bar{\boldsymbol{M}}(\boldsymbol{\Phi})$ and

$\bar{\boldsymbol{C}}(\boldsymbol{\Phi}, \dot{\boldsymbol{\Phi}})$, are then used in the following timestep's dynamic reduction process to define a new control torque for achieving output tracking.

The following subsections provide detailed descriptions of the algorithms associated with each block in the simulation block diagram. In keeping with the flow of content presented in this thesis, the implementation of the kinematics models are first discussed, followed by the algorithms for simulating the system's dynamics, numeric integration, and workspace controller.

6.2.1 Forward Kinematics

As outlined in section 4.2.2, computing the forward kinematics to any member in a multi-body system requires the product of exponentials formulation. In performing the POE, equations (A.13) and (A.14) are implemented in Python to compute the exponential map of a twist and associated twist angle. The Rodrigues' formula in (A.4) is also coded to compute the exponential of the $\mathfrak{so}(3)$ elements. As previously mentioned, the initialization process in the Python simulator defines all twists contained in the system. Therefore, given the system's configuration $\boldsymbol{\Phi}$, the exponential of each twist and twist angle can be computed. The forward kinematics function implemented in Python computes the homogeneous transformation matrix from any body frame k_1 to any subsequent body in the system k_2 (i.e., $\boldsymbol{g}_{k_1 k_2}$). Note that given the initial placement of the spatial coordinate frame in Figure 6.2, $k_1 = 0$ defines the homogeneous transformation from the spatial frame to any body k_2 in the system (i.e., $\boldsymbol{g}_{sk_2}$).

Algorithm 1 describes the procedure of the forward kinematics function in the Python simulator. The function's inputs include the desired system bodies k_1 and k_2 for which the transformation matrix is computed between, and the initial transformation between these two bodies $\boldsymbol{g}_{k_1 k_2}(0)$ (as required by the product of exponentials formula in equation (4.4)). Note that the vector of generalized coordinate positions is not required as an input as it is defined to be an attribute of the multi-body system, and is thus accessible in every function. The majority of the function's computation stems from the single for loop to calculate the required product of exponentials. This calculation involves implementing equation (4.4) starting from the index l_{k_1} (defining the number of degrees of freedom preceding k_1) and ending at l_{k_2}. In the loop, the screw motions between the two bodies are combined through post-multiplication to form the POE matrix $\boldsymbol{G}_{k_1 k_2}$. This matrix $\boldsymbol{G}_{k_1 k_2}$ is lastly pre-multiplied to the initial transformation between the two bodies to obtain the kinematic mapping from body k_1 to body k_2.

Algorithm 1 Forward Kinematics Function from Body k_1 to Body k_2

Input: 1) Bodies k_1 and k_2
2) Initial transformation $\boldsymbol{g}_{k_1k_2}(0)$

Output: 1) Homogeneous transformation from body k_1 to body k_2: $\boldsymbol{g}_{k_1k_2}$
2) Product of exponentials from body k_1 to body k_2: $\boldsymbol{G}_{k_1k_2}$

function FORWARD KINEMATICS(k_1, k_2, $\boldsymbol{g}_{k_1k_2}(0)$)
Define number of DOFs preceding bodies k_1 and k_2: l_{k_1}, l_{k_2}
Initialize $\boldsymbol{G}_{k_1k_2}$ as the identity $\mathbf{1}_{4\times4}$
for twist $i = l_{k_1}, l_{k_2}$ **do**
Extract position Φ_i from the system position vector
Extract twist $\boldsymbol{\xi}_i$ from the twist repository
Compute the exponential map $e^{\boldsymbol{\xi}_i\Phi_i}$
Update $\boldsymbol{G}_{k_1k_2}$ by post multiplying the exponential map $e^{\boldsymbol{\xi}_i\Phi_i}$:
$\boldsymbol{G}_{k_1k_2} \leftarrow \boldsymbol{G}_{k_1k_2}e^{\boldsymbol{\xi}_i\Phi_i}$
end for
Post multiply the initial transformation $\boldsymbol{g}_{k_1k_2}(0)$ with $\boldsymbol{G}_{k_1k_2}$:
$\boldsymbol{g}_{k_1k_2} \leftarrow \boldsymbol{G}_{k_1k_2}\boldsymbol{g}_{k_1k_2}(0)$
end function

6.2.2 Differential Kinematics

The differential kinematics aspect of the multi-body simulator refers to the functions which compute the body and spatial Jacobian matrices. It is important to design these functions in a versatile manner such that the Jacobian corresponding to any body k in the system can be determined by simply changing the input. The benefit of such versatility will become apparent in the computations of the system's dynamics, and the implementation of the output tracking controller. In this simulation, two independent functions are created for separately calculating the spatial and body Jacobian matrices associated with body k. As expected, these functions are predicated on the definitions of the columns of the spatial and body Jacobians from equations (4.9) and (4.13), respectively.

Firstly, Algorithm 2 outlines the computation of the spatial Jacobian, where the desired body k is the only required input. The main loop of the function defines each column of the spatial Jacobian based on equation (4.9). This involves transforming the initial twists based on the system's current configuration. This transformation relies on the Adjoint of the

product of exponentials from the spatial frame to the screw motion associated with column $i-1$. Consequently, a matrix $\boldsymbol{G}_{si}$ is initialized as the identity to then define the product of exponentials up to column $i-1$ in the loop. Additionally, these product of exponentials matrices to each twist i are stored for use in subsequent functions to increase efficiency and not to repeat identical computations. The single for loop in Algorithm 2 runs through all of the non-zero columns of the spatial Jacobian which is defined by the number of screw motions in the system preceding body k (i.e., defined by the parameter l_k).

Algorithm 2 Spatial Jacobian for Body k

Input: Body k

Output: Spatial Jacobian to body k: $\boldsymbol{J}_k^s$

function SPATIAL JACOBIAN(k)
- Define number of DOFs preceding body k: l_k
- Initialize $\boldsymbol{J}_k^s$ as a zero matrix $\mathbf{0}_{6\times N_D}$
- Initialize matrix $\boldsymbol{G}_{si}$ as the identity $\mathbf{1}_{4\times 4}$
- **for** column $i = 1, l_k$ **do**
  - Extract position Φ_i from the system position vector
  - Extract twist $\boldsymbol{\xi}_i$ from the twist repository
  - Compute the exponential map $e^{\boldsymbol{\xi}_i\Phi_i}$
  - Post multiply the exponential map $e^{\boldsymbol{\xi}_i\Phi_i}$ with $\boldsymbol{G}_{si}$ and store $\boldsymbol{G}_{si}$:
    $$\boldsymbol{G}_{si} \leftarrow \boldsymbol{G}_{si}e^{\boldsymbol{\xi}_i\Phi_i}$$
  - Define $\mathbf{Ad}_{\boldsymbol{G}_{si}}$ and compute column $\boldsymbol{\eta}_i^{'}$ using (4.9)
- **end for**

end function

The spatial Jacobian for the base body ($k = 0$) resulting from Algorithm 2 can subsequently be truncated to define $\bar{\boldsymbol{J}}_0^s$ for use in the system's dynamic reduction process.

Using the same general methodology as the spatial Jacobian algorithm, Algorithm 3 demonstrates the implementation of the body Jacobian to any body k. Again, only a single input is required to specify the body k, and a single for loop is used to define the non-zero columns of the body Jacobian. In this case, equation (4.13) is required for defining column $\boldsymbol{\eta}_i$. Algorithm 1 is required here to establish the product of exponentials for the Adjoint operator $\mathbf{Ad}_i^{l_k}$. Recall that the Adjoint $\mathbf{Ad}_i^{l_k}$ indicates the inverse product of exponentials formula to that performed in the forward kinematics map. Hence, the inverse of the output

of the POE function in Algorithm 1 is taken prior to applying the Adjoint operator.

Algorithm 3 Body Jacobian for Body k

Input: 1) Body k
2) Initial transformation $\boldsymbol{g}_{sk}(0)$
Output: Body Jacobian to body k: $\boldsymbol{J}_k^b$

function BODY JACOBIAN(k)
Define number of DOFs preceding body k: l_k
Initialize $\boldsymbol{J}_k^b$ as zero matrix $\mathbf{0}_{6\times N_D}$
Define the inverse Adjoint of the initial transformation: $\mathbf{Ad}_{\boldsymbol{g}_{sk(0)}^{-1}}$
for column $i = 1, l_k$ **do**
Extract twist $\boldsymbol{\xi}_i$ from the twist repository
Let $\boldsymbol{G}_{il_k} \leftarrow \text{POE}(i, l_k)$ and invert
Define $\mathbf{Ad}_i^{l_k}$ using $\boldsymbol{G}_{il_k}^{-1}$ and compute column $\boldsymbol{\eta}_i$ using (4.13)
end for
end function

This completes the kinematics modelling aspect of the multi-body simulator. We now progress to the implementation of the Euler-Lagrange equations of motion derived in section 4.3. To start, the following subsection details the function developed for computing the system inertia matrix $\bar{\boldsymbol{M}}(\boldsymbol{\Phi})$.

6.2.3 Inertia Matrix

As defined in equation (4.27), the inertia matrix is predicated on the body Jacobian matrices of each body in the system, as well as each member's individual inertia matrix. The individual body inertia matrices are already computed and stored in the simulation's initialization, and are simply called upon in the system inertia matrix function. As will be seen in the motion propagation algorithm, the inertia matrix computation is the first instance in which the body Jacobians for each body are calculated. To optimize the computational efficiency of the simulator, only a single spatial Jacobian matrix to the end-effector $\boldsymbol{J}_N^s$ is computed since the columns of each preceding body's spatial Jacobian are included in $\boldsymbol{J}_N^s$. Consequently, the non-zero columns of every matrix $\boldsymbol{J}_k^s$ can be defined by partitioning $\boldsymbol{J}_N^s$ up to column l_k. Moreover, the corresponding body Jacobian matrices $\boldsymbol{J}_k^s$ can be easily

obtained via the Adjoint operator, as shown in equation (4.16). This method for defining each body Jacobian removes the need for calling the body Jacobian function $N+1$ number of times, thus reducing the computational load of the inertia matrix algorithm.

Algorithm 4 demonstrates the methodology for calculating the system inertia matrix $\bar{\boldsymbol{M}}$. Since the inertia matrix is only predicated on the system's configuration and physical properties (both attributes of the multi-body system in the simulation), the inertia matrix function requires no explicit inputs. The main loop of this function uses $\boldsymbol{J}_N^s$, as previously mentioned, to define the matrix $\boldsymbol{J}$ containing the body Jacobians of each body in the system for subsequent use in equation (4.27). This matrix $\boldsymbol{J}$ is additionally beneficial for acting as a repository for all $\boldsymbol{J}_k^b$'s throughout the simulation. This further increases the efficiency of the entire simulation as the matrix $\boldsymbol{J}$ can be referenced throughout the remaining components of the simulator (e.g., Coriolis matrix, output tracking controller) to avoid duplicate body Jacobian calculations. As a result, the system inertia matrix function outputs both the $\bar{\boldsymbol{M}}$ matrix, the repository of body Jacobians $\boldsymbol{J}$, and the truncates spatial Jacobian for the base $\bar{\boldsymbol{J}}_0^s$ (required in the dynamics reduction process).

Algorithm 4 System Inertia Matrix

Output: 1) System inertia matrix: $\bar{\boldsymbol{M}}$
2) Matrix of body Jacobians: $\boldsymbol{J}$
3) Truncated spatial Jacobian for the base: $\bar{\boldsymbol{J}}_0^s$

function INERTIA MATRIX()
Call $\boldsymbol{D}$ from initialization
Let $\boldsymbol{J}_N^s \leftarrow$ SPATIALJACOBIAN$(k = N)$
Initialize $\boldsymbol{J}$ as zero matrix $\mathbf{0}_{6(N+1)\times N_D}$
for body k = 0, N **do**
Define number of DOFs preceding body k: l_k
Call $\boldsymbol{g}_{sk}(0)$ from repository of initial configurations for each body
Call $\boldsymbol{G}_{sk}$ from repository of product of exponentials to each body
Let $\boldsymbol{g}_{sk} \leftarrow \boldsymbol{G}_{sk}\, \boldsymbol{g}_{sk}(0)$
Extract columns 1 to l_k in $\boldsymbol{J}_N^s$ to define non-zero columns of $\boldsymbol{J}_k^s$
if k = 0 **then**
Define the truncated spatial Jacobian for the base $\bar{\boldsymbol{J}}_0^s$
end if
Convert $\boldsymbol{J}_k^s$ to $\boldsymbol{J}_k^b$ using $\mathbf{Ad}_{\boldsymbol{g}_{sk}^{-1}}$ and (4.16)
Fill $\boldsymbol{J}$ with non-zero columns of $\boldsymbol{J}_k^b$
end for
Compute $\bar{\boldsymbol{M}}$ using (4.27)
end function

6.2.4 Coriolis Matrix

In calculating the system Coriolis matrix, the derivative of the inertia matrix with respect to each generalized coordinate is required. Recalling the definition of $\bar{\boldsymbol{C}}$ in equations (4.38) and (4.32), the Coriolis matrix is heavily predicated on the partial derivatives of body Jacobians with respect to each generalized coordinate. As such, the majority of the computations in the Coriolis matrix process are performed using the Jacobian derivative function outlined in Algorithm 5 below. This function takes the matrix $\boldsymbol{J}$ as an input, and returns a repository of the partial derivatives for all $\boldsymbol{J}_k^b$'s with respect to each position Φ (i.e., $\frac{\partial \boldsymbol{J}}{\partial \boldsymbol{\Phi}}$). Additionally, a repository of the partial derivatives of $\bar{\boldsymbol{J}}_0^s$ with respect to each Φ is created

and filled. Note that the former output is only required for the Coriolis matrix computation, and the partial derivatives of $\bar{\boldsymbol{J}}_0^s$ will become an important output for computing $\dot{\bar{\boldsymbol{J}}}_0^s$ in the output tracking controller section of the simulation (necessary for feedback linearization).

The Jacobian partial derivative function starts by pre-defining the partial derivative repositories for both $\boldsymbol{J}$ and $\bar{\boldsymbol{J}}_0^s$. Subsequently, the main loops of the function involve defining all of the Adjoint operators required for the partial derivative of a body Jacobian matrix shown in equation (4.35) and for the partial of the truncated spatial Jacobian in equation (5.10). As such, the first for loop runs through each body k in the system to set the values for l_k and $\mathbf{Ad}_{g_{sk}^{-1}(0)}$, followed by a for loop which runs through every column j of the body Jacobian. The third and final for loop takes the partial derivative of the j^{th} column of body k's body frame Jacobian with respect to each generalized coordinate i. Recall that the body Jacobian partial derivative has non-zero elements for $i \leq j$, thus limiting the indices of the for loops and increasing computational efficiency (only the non-zero columns of the body Jacobian partial derivative are considered in the loops).

Algorithm 5 Partial Derivative of Body and Truncated Spatial Jacobian Matrices

Input: Matrix of body Jacobians: $\boldsymbol{J}$

Output: 1) Repository $\frac{\partial \boldsymbol{J}}{\partial \boldsymbol{\Phi}}$
2) Repository $\frac{\partial \bar{\boldsymbol{J}}_0^s}{\partial \boldsymbol{\Phi}}$

function JACOBIAN DERIVATIVES($\boldsymbol{J}$)
Initialize repositories $\frac{\partial \boldsymbol{J}}{\partial \boldsymbol{\Phi}}$, $\frac{\partial \bar{\boldsymbol{J}}_0^s}{\partial \boldsymbol{\Phi}}$
Initialize dictionary for storing all unique Adjoint matrices
for body $k = 0, N$ **do**
Define number of DOFs preceding body k: l_k
Call $\mathbf{Ad}_{g_{sk}^{-1}(0)}$ from the initialization
for column $j = 1, l_k$ **do**
Extract twist $\boldsymbol{\xi}_j$ from the twist repository
for DOF $i = j, l_k$ **do**
if $\mathbf{Ad}_j^i$ is not in the Adjoint dictionary **then**
Let $\boldsymbol{G}_{ji} \leftarrow \text{POE}(j, i)$ and invert
Define $\mathbf{Ad}_j^i$ using $\boldsymbol{G}_{ji}^{-1}$ and update the Adjoint dictionary
end if
if $\mathbf{Ad}_{i+1}^{l_k}$ is not in the Adjoint dictionary **then**
Let $\boldsymbol{G}_{(i+1)l_k} \leftarrow \text{POE}(i+1, l_k)$ and invert
Define $\mathbf{Ad}_{i+1}^{l_k}$ using $\boldsymbol{G}_{(i+1)l_k}^{-1}$ and update the Adjoint dictionary
end if
Call $\mathbf{ad}_{\xi_i}$ from the initialization
Calculate $\frac{\partial \boldsymbol{J}_{k,j}^b}{\partial \Phi_i}$ using equation (4.35) and update $\frac{\partial \boldsymbol{J}}{\partial \boldsymbol{\Phi}}$
end for
if body $k = 0$ and $(2 \leq j \leq 6)$ **then**
for DOF $i = 1, j-1$ **do**
if $(\mathbf{Ad}_1^i)'^{-1}$ is not in the Adjoint dictionary **then**
Let $\boldsymbol{G}_{1i} \leftarrow \text{POE}(1, i)$
Define $(\mathbf{Ad}_1^i)'$ using $\boldsymbol{G}_{1i}$ and update the Adjoint
end if
if $(\mathbf{Ad}_{i+1}^{j-1})'^{-1}$ is not in the Adjoint dictionary **then**
Let $\boldsymbol{G}_{1+1,j-1} \leftarrow \text{POE}(i+1, j-1)$
Define $(\mathbf{Ad}_{i+1}^{j-1})'$ using $\boldsymbol{G}_{1+1,j-1}$ and update the Adjoint dictionary
end if

Calculate $\frac{\partial \bar{\boldsymbol{J}}_0^s}{\partial \Phi_i}$ using equation (5.11) and update $\frac{\partial \bar{\boldsymbol{J}}_0^s}{\partial \boldsymbol{\Phi}}$
end for
end if
end for
end for
end function

For the case of $k = 0$, referring to the spacecraft's base, the function adds a single for loop to iterate through the first six columns of the base's spatial Jacobian. Recall again that for the truncated spatial Jacobian of the base, columns $2 \leq j \leq 6$ are non-zero in the partial derivative matrix. Additionally, dictionaries for storing all unique Adjoint matrices are constantly updated and called upon in the function such that Adjoint matrix calculations are not duplicated. This dictionary greatly reduces the function's computational load as shared Adjoint matrices are used in several instances in both the body and spatial Jacobian partial derivatives. Note that the upper and lower indices in the Adjoint dictionary reflect the Adjoint operator definition in equation (4.15), corresponding to the body Jacobian. The Adjoint operators used in the spatial Jacobian are thus the inverse of those stored in the dictionary, as defined in equation (4.11), consequently implying that the Adjoint operator $(\mathbf{Ad}_i^j)'$ corresponds to the inverse of the matrices in the dictionary.

Following the definition of the partial derivatives of each body Jacobian matrix with respect to all generalized coordinates, the residual computation of the Coriolis matrix remains trivial. The Coriolis matrix function is left to calculate the partial derivative of the system inertia matrix with respect to each system position, using equation (4.32). This computation is included in a single for loop iterating through the generalized coordinates of the system. This single for loop additionally implements the Coriolis matrix definition in equation (4.38) by performing the first term summation, and defining the second term matrix. Algorithm 6 demonstrates the Coriolis matrix function in the Python simulation. As mentioned, this function relies on the partial derivatives of the body Jacobian matrices from the previous algorithm, as well as the generalized coordinate velocities and the matrix of body Jacobians as inputs. In addition to the system Coriolis matrix, a repository storing the partial derivatives of the inertia matrix with respect to the generalized coordinates is also output.

Algorithm 6 System Coriolis Matrix

Input: 1) System velocities: $\dot{\boldsymbol{\Phi}}$
2) Matrix of body Jacobians: $\boldsymbol{J}$
3) Repository of partial derivatives for body Jacobians: $\frac{\partial \boldsymbol{J}}{\partial \boldsymbol{\Phi}}$
Output: 1) System Coriolis matrix: $\bar{\boldsymbol{C}}$
2) Repository of inertia matrix partial derivatives: $\frac{\partial \bar{\boldsymbol{M}}}{\partial \boldsymbol{\Phi}}$

function CORIOLIS MATRIX($\dot{\boldsymbol{\Phi}}$, $\boldsymbol{J}$, $\frac{\partial \boldsymbol{J}}{\partial \boldsymbol{\Phi}}$)
Initialize repository $\frac{\partial \bar{\boldsymbol{M}}}{\partial \boldsymbol{\Phi}}$
Initialize the first term summation $\sum_{i=1}^{N_D} \frac{\partial \bar{\boldsymbol{M}}(\boldsymbol{\Phi})}{\partial \Phi_i} \dot{\Phi}_i$
for column $i = 1, N_D$ **do**
Extract $\frac{\partial \boldsymbol{J}}{\partial \Phi_i}$ from $\frac{\partial \boldsymbol{J}}{\partial \boldsymbol{\Phi}}$
$\frac{\partial \bar{\boldsymbol{M}}}{\partial \Phi_i} \leftarrow \left(\frac{\partial \boldsymbol{J}}{\partial \Phi_i}\right)^T \boldsymbol{D}\boldsymbol{J} + \boldsymbol{J}^T \boldsymbol{D} \frac{\partial \boldsymbol{J}}{\partial \Phi_i}$
Update $\frac{\partial \bar{\boldsymbol{M}}}{\partial \Phi_i}$ repository
Add $\frac{\partial \bar{\boldsymbol{M}}(\boldsymbol{\Phi})}{\partial \Phi_i} \dot{\Phi}_i$ to the first term summation

end for
Create matrix $\left[\left(\dot{\boldsymbol{\Phi}}^T \frac{\partial \bar{\boldsymbol{M}}(\boldsymbol{\Phi})}{\partial \Phi_1}\right)^T \quad \dots \quad \left(\dot{\boldsymbol{\Phi}}^T \frac{\partial \bar{\boldsymbol{M}}(\boldsymbol{\Phi})}{\partial \Phi_{N_D}}\right)^T\right]^T$
Calculate system Coriolis matrix $\bar{\boldsymbol{C}}$ using equation (4.38)
end function

With the implementation of the system Coriolis matrix, the plant dynamics and kinematics of a space manipulator are completely modelled in the Python simulator. The following subsections now detail the dynamics reduction, feedback linearization, and output tracking control law structures outlined in chapter 5, as well as the process for updating the space manipulator's motion (in response to the system's initial momentum and control actions). Note that the controller portion of the simulation may be replaced by other control designs which correspond to the rigid multi-body system being simulated. For the output tracking controller based on feedback linearization, we begin by detailing the implementation of the dynamic reduction.

6.2.5 Dynamics Reduction

In this subsection, the simulation of the *Dynamic Reduction* block in the output tracking controller block diagram (Figure 5.1) is described. As shown by the block's output signals, the primary functions of this process are to define the matrices $\boldsymbol{B}$ and $\boldsymbol{S}$ resulting from the conservation of momentum constraint, as well as the matrices from the reduced equations of motion ($\boldsymbol{M}$, $\boldsymbol{C}$, and $\boldsymbol{E}$). The dynamic reduction function additionally covers the time derivative aspect of the controller block diagram to define the matrix derivatives necessary for the feedback linearization control law: $\dot{\boldsymbol{B}}$, and $\dot{\boldsymbol{S}}$. Note that the time derivatives of these matrices (along with the system Coriolis matrix $\bar{\boldsymbol{C}}$) are restricted to $\mathscr{M}$ using the change of variable for the base motion coming from the momentum constraint.

Algorithm 7 demonstrates the dynamic reduction function for defining the outputs of the *Dynamic Reduction* block, and the time derivative signals in Figure 5.1. The function inputs the generalized coordinate velocities, the truncated spatial Jacobian for the base and its partial derivative with respect to the system positions, and the outputs to both the inertia and Coriolis matrix functions. The matrices $\boldsymbol{B}$ and $\boldsymbol{S}$ from the conservation of momentum constraint are first defined based on equations (5.3) and (5.2), followed by a for loop running through all generalized velocities to calculate the time derivatives of $\bar{\boldsymbol{M}}$ and $\bar{\boldsymbol{J}}_0^s$ (both restricted to $\mathscr{M}$). Subsequently, the time derivatives $\dot{\boldsymbol{S}}$ and $\dot{\boldsymbol{B}}$ are defined based on equations (5.5) and (5.6), using the derivatives $\dot{\bar{\boldsymbol{M}}}$ and $\dot{\bar{\boldsymbol{J}}}_0^s$ which are predicated only on $\dot{\boldsymbol{q}}$ (i.e., $\mathbb{S}$ and $\mathbb{B}$ are obtained). Finally, the matrices $\boldsymbol{M}$, $\boldsymbol{C}$, and $\boldsymbol{E}$ of the system dynamics constrained to the manipulator motion are calculated using equations (5.17)-(5.19).

Algorithm 7 Dynamics Reduction Process

Input: 1) System velocities: $\dot{\boldsymbol{\Phi}}$
2) Truncated spatial Jacobian for base: $\bar{\boldsymbol{J}}_0^s$
3) System inertia matrix: $\bar{\boldsymbol{M}}$
4) System Coriolis matrix: $\bar{\boldsymbol{C}}$
5) Repository of inertia matrix partial derivatives: $\frac{\partial \bar{\boldsymbol{M}}}{\partial \boldsymbol{\Phi}}$
6) Repository $\frac{\partial \bar{\boldsymbol{J}}_0^s}{\partial \boldsymbol{\Phi}}$

Output: 1) Matrices from reduced dynamics: $\boldsymbol{M}, \boldsymbol{C}, \boldsymbol{E}$
2) Matrices from momentum constraint: $\boldsymbol{B}$, $\boldsymbol{S}$
3) Restricted time derivatives: $\mathbb{B}$, $\mathbb{S}$
3) Restricted velocity vector: $\dot{\boldsymbol{\Phi}}(\boldsymbol{\Phi}, \dot{\boldsymbol{q}}, \boldsymbol{\mu})$

function DYNAMICS REDUCTION($\dot{\boldsymbol{\Phi}}$, $\bar{\boldsymbol{J}}_0^s$, $\bar{\boldsymbol{M}}$, $\bar{\boldsymbol{C}}$, $\frac{\partial \bar{\boldsymbol{M}}}{\partial \boldsymbol{\Phi}}$, $\frac{\partial \bar{\boldsymbol{J}}_0^s}{\partial \boldsymbol{\Phi}}$)
Call system momentum $\boldsymbol{\mu}$ from initialization
Calculate $\boldsymbol{B}$ and $\boldsymbol{S}$ using equations (5.3) and (5.2)
Restrict $\bar{\boldsymbol{C}}$ to $\mathscr{M}$ to obtain $\mathbb{C}(\boldsymbol{\Phi}, \dot{\boldsymbol{q}}; \boldsymbol{\mu})$
Initialize time derivative matrices for $\dot{\bar{\boldsymbol{M}}}$ and $\dot{\bar{\boldsymbol{J}}}_0^s$
Restrict system velocities to $\mathscr{M}$: $\dot{\boldsymbol{\Phi}}(\boldsymbol{\Phi}, \dot{\boldsymbol{q}}, \boldsymbol{\mu}) \leftarrow [(\boldsymbol{B} - \boldsymbol{S}\dot{\boldsymbol{q}})^T \ \dot{\boldsymbol{q}}^T]^T$
for i = 1, N_D **do**
$\dot{\bar{\boldsymbol{M}}} \leftarrow \dot{\bar{\boldsymbol{M}}} + \frac{\partial \bar{\boldsymbol{M}}}{\partial \Phi_i} \dot{\Phi}_i(\boldsymbol{\Phi}, \dot{\boldsymbol{q}}, \boldsymbol{\mu})$
$\dot{\bar{\boldsymbol{J}}}_0^s \leftarrow \dot{\bar{\boldsymbol{J}}}_0^s + \frac{\partial \bar{\boldsymbol{J}}_0^s}{\partial \Phi_i} \dot{\Phi}_i(\boldsymbol{\Phi}, \dot{\boldsymbol{q}}, \boldsymbol{\mu})$
end for
Calculate $\mathbb{S}$ by equation (5.5) using restricted $\dot{\bar{\boldsymbol{M}}}$
Calculate $\mathbb{B}$ by equation (5.6) using restricted $\dot{\bar{\boldsymbol{M}}}$ and $\dot{\bar{\boldsymbol{J}}}_0^s$
Calculate reduced inertia and Coriolis matrices $\boldsymbol{M}$ and $\boldsymbol{C}$ by (5.17) and (5.18)
Calculate $\boldsymbol{E}$ from reduced dynamics by (5.19)
end function

As demonstrated in Figure 5.1, the dynamic reduction process then leads into the feedback linearization section of the simulation. Recalling Chapter 5, two output tracking control cases are considered (e.g., linear output control and full pose control of the output). To address the more complete and complex full pose control of the end-effector, the following subsection outlines the feedback linearization process for this control case, as well as the implementation of the modified PID controller presented in section 5.5.

6.2.6 Output Tracking Controller

The implementation of the full pose output tracking controller is separated into two processes: the first to define the control actions resulting from the modified feedforward, feedback PID control law, and the second to perform the feedback linearization on the end-effector motion. Regarding the full pose control structure, the group error $\boldsymbol{g}_e$, gradient of the error function $\nabla\psi$, and body velocity error $\boldsymbol{V}_e^b$ must be first defined to establish the proportional, integral, derivative, and feedforward control actions. To start, the group error requires the actual and desired end-effector poses, where $\boldsymbol{g}_{sN}$ is computed by the forward kinematics function in Algorithm 1 and $\bar{\boldsymbol{g}}_{sN}$ is defined by a trajectory planner. Following the group error, $\boldsymbol{V}_e^b$ and $\nabla\psi$ are defined based on equations (5.62) and (5.65), allowing for the definitions of $\boldsymbol{U}_{pi}$ and $\boldsymbol{U}_d$. Note that $\boldsymbol{V}_e^b$ requires the desired end-effector body velocity which again comes from an external trajectory planner. Lastly, the feedforward gain $\boldsymbol{U}_{ff}$ makes use of the output's desired body acceleration (again an input to the controller) to complete the full pose output tracking control law. Using the resulting control action $\boldsymbol{U}$, in addition to the matrices output by the dynamic reduction process in Algorithm 7, the majority of the feedback linearization law from equation (5.53) is established. The only supplementary computations required for the linearization process are to define the GJM $\bar{\boldsymbol{J}}^*$, its derivative restricted to $\mathscr{M}$, and its SR-inverse $\bar{\boldsymbol{J}}^{*\diamond}$.

Algorithm 8 outlines the function for performing full pose output tracking control based on the controller and feedback linearization processes mentioned above. As mentioned, the necessary inputs to the controller are the desired end-effector trajectory ($\bar{\boldsymbol{g}}_{sN}$, $\bar{\boldsymbol{V}}_{sN}^b$, and $\dot{\bar{\boldsymbol{V}}}_{sN}^b$), the matrices $\boldsymbol{M}$, $\boldsymbol{C}$, and $\boldsymbol{E}$ from the reduced dynamics, the matrix $\boldsymbol{B}$ from the momentum constraint and its restricted derivative $\mathbb{B}$, and the system's body Jacobian for defining the GJM. Additionally, the restricted generalized velocities and the repository of partial derivatives of the body Jacobians are required for computing the derivative of the GJM. This function yields a single output defining the control actuation to the manipulator $\boldsymbol{\tau}_m$ for achieving feedback linearization and output tracking control.

Algorithm 8 Output Tracking Controller

Input: 1) Restricted system velocities: $\dot{\boldsymbol{\Phi}}(\boldsymbol{\Phi}, \dot{\boldsymbol{q}}, \boldsymbol{\mu})$
2) Desired trajectory: $\bar{\boldsymbol{g}}_{sN}$, $\bar{\boldsymbol{V}}^b_{sN}$, $\dot{\bar{\boldsymbol{V}}}^b_{sN}$
3) Matrices from reduced dynamics: $\boldsymbol{M}$, $\boldsymbol{C}$, $\boldsymbol{E}$
4) Matrices $\boldsymbol{B}$ and $\mathbb{B}$
5) System body Jacobian: $\boldsymbol{J}^b_N$
6) Repository $\frac{\partial \boldsymbol{J}}{\partial \boldsymbol{\Phi}}$

Output: Manipulator control torque: $\boldsymbol{\tau}_m$

function OUTPUT TRACKING CONTROLLER($\dot{\boldsymbol{\Phi}}(\boldsymbol{\Phi}, \dot{\boldsymbol{q}}, \boldsymbol{\mu})$, $\bar{\boldsymbol{g}}_{sN}$, $\bar{\boldsymbol{V}}^b_{sN}$, $\dot{\bar{\boldsymbol{V}}}^b_{sN}$, $\boldsymbol{M}$, $\boldsymbol{C}$, $\boldsymbol{E}$, $\boldsymbol{B}$, $\mathbb{B}$, $\boldsymbol{J}^b_N$)

Define PID gains: $\boldsymbol{K}_p$, $\boldsymbol{K}_i$, $\boldsymbol{K}_d$

$\boldsymbol{g}_{sN} \leftarrow$ FORWARD KINEMATICS$(0, N, \boldsymbol{g}_{sN}(0))$

$\boldsymbol{g}_e \leftarrow \bar{\boldsymbol{g}}^{-1}_{sN} \boldsymbol{g}_{sN}$

Define $\nabla \psi$ and $\boldsymbol{V}^b_e$ by equations (5.65) and (5.62)

Define $\boldsymbol{U}_{pi}$, $\boldsymbol{U}_d$, $\boldsymbol{U}_{ff}$ by equations (5.66), (5.64), (5.68)

Partition $\boldsymbol{J}^b_N$ into its base and manipulator portions: $\boldsymbol{J}^b_o$ and $\boldsymbol{J}^b_m$

Define GJM $\boldsymbol{J}^*$ by equation (5.50)

Initialize the restricted time derivative matrix $\mathbb{J}^b_N$

for i = 1, N_D **do**

$\mathbb{J}^b_N \leftarrow \mathbb{J}^b_N + \frac{\partial \boldsymbol{J}^b_N}{\partial \Phi_i} \dot{\Phi}_i(\boldsymbol{\Phi}, \dot{\boldsymbol{q}}, \boldsymbol{\mu})$

end for

Partition $\mathbb{J}^b_N$ into its base and manipulator portions: $\mathbb{J}^b_o$ and $\mathbb{J}^b_m$

Define GJM derivative $\mathbb{J}^*$ by using $\mathbb{J}^b_o$ and $\mathbb{J}^b_m$

Define the system's manipulability w using equation (C.5)

if $w < w_t$ **then**

$\lambda \leftarrow \lambda_0 \left(1 - \frac{w}{w_t}\right)^2$

else

$\lambda \leftarrow 0$

end if

Calculate the SR-Inverse of the GJM $\boldsymbol{J}^{*\diamond}$ by equation (C.4)

Calculate $\boldsymbol{\tau}_m$ by equation (5.53)

end function

As depicted by the simulation block diagram in Figure 6.1, the control torque is subsequently applied to the space manipulator system's dynamics in order to propagate the generalized coordinates and velocities. The following subsection outlines the propagation process which establishes the set of differential equations for numeric integration.

6.2.7 Motion Propagation

As mentioned above, the motion propagation process starts by applying the control torque to the multi-body system's equations of motion. This aspect of the simulator is consistent for any simulation scenario. Regardless of the controller design, this process simply replicates the expected motion of a rigid multi-body system under the presence of some torque or force vector. The system's equations of motion stem from the Euler-Lagrange formulation shown in section 4.3, and are thus defined using the system's inertia and Coriolis matrices (as indicated by the input signals to the *Equations of Motion* block in Figure 6.1). Using the Euler-Lagrange equations of motion, and provided a system actuation vector $\boldsymbol{\tau}$, the simulator then defines the resulting generalized accelerations to be integrated. Double integrating such generalized accelerations evidently provides the system's velocities and configuration. Thus, the first step in the motion propagation method defines the matrix of ODEs containing the generalized accelerations and velocities, as defined in (6.2). Following integration, the desired matrix $\boldsymbol{Y}$ of generalized velocities and positions (equation (6.1)) is obtained.

As previously mentioned, the simulator's numeric integration is performed using the fourth order Runge-Kutta method. This integration technique requires a function f which defines the set of ODEs to be solved. In performing the numeric integration for a single timestep, this function f is only predicated on the vector $\boldsymbol{Y}$ which defines the set of integrated values. The fourth order Runge-Kutta operator combines the outputs of $f(\boldsymbol{Y})$ evaluated at four different inputs of $\boldsymbol{Y}$ to obtain the integrated values for the following time step. That is, for timestep n, the desired vector $\boldsymbol{Y}_{n+1}$ resulting from the fourth order Runge-Kutta integration is calculated as follows,

$$\boldsymbol{Y}_{n+1} = \boldsymbol{Y}_n + \frac{1}{6}(\boldsymbol{K}_1 + 2\boldsymbol{K}_2 + 2\boldsymbol{K}_3 + \boldsymbol{K}_4) \tag{6.3}$$

where,

$$\boldsymbol{K}_1 = f(\boldsymbol{Y}_n)dt \tag{6.4}$$

$$\boldsymbol{K}_2 = f\left(\boldsymbol{Y}_n + \frac{\boldsymbol{K}_1}{2}\right)dt \tag{6.5}$$

$$\boldsymbol{K}_3 = f\left(\boldsymbol{Y}_n + \frac{\boldsymbol{K}_2}{2}\right)dt \tag{6.6}$$

$$\boldsymbol{K}_4 = f(\boldsymbol{Y}_n + \boldsymbol{K}_3)dt. \tag{6.7}$$

Algorithm 10 outlines the process of performing the fourth order Runge-Kutta numeric integration on the Euler-Lagrange equations of motion. Note that the simulator (and in turn the simulation block diagram) may be adjusted to include additional variables in the $\boldsymbol{Y}$ matrix for integration. Provided the desired end-effector trajectory, and the initial conditions for integration (i.e., the system's initial configuration and velocity), the motion propagation function calls the output tracking controller to obtain a control action. Based on the resulting actuation torque, the system of ODEs is formed. Algorithm 9 outlines the function which performs the trajectory following process to define $\frac{d\boldsymbol{Y}}{dt}$.

Algorithm 9 Definition of ODEs for a Trajectory Following Task

Input: 1) Desired trajectory: $\bar{\boldsymbol{g}}_{sN}$, $\bar{\boldsymbol{V}}^b_{sN}$, $\dot{\bar{\boldsymbol{V}}}^b_{sN}$
2) Current system states: $\boldsymbol{Y}_n$
Output: System of ODEs: $\frac{d\boldsymbol{Y}}{dt}$

function ODEs($\bar{\boldsymbol{g}}_{sN}$, $\bar{\boldsymbol{V}}^b_{sN}$, $\dot{\bar{\boldsymbol{V}}}^b_{sN}$, $\boldsymbol{Y}_n$)
Extract and define $\dot{\boldsymbol{\Phi}}_n$ from $\boldsymbol{Y}_n$
Extract and globally define $\boldsymbol{\Phi}_n$ from $\boldsymbol{Y}_n$
$\bar{\boldsymbol{M}}, \boldsymbol{J}, \bar{\boldsymbol{J}}^s_0 \leftarrow$ INERTIA MATRIX()
$\frac{\partial \boldsymbol{J}}{\partial \boldsymbol{\Phi}}, \frac{\partial \bar{\boldsymbol{J}}^s_0}{\partial \boldsymbol{\Phi}} \leftarrow$ JACOBIAN DERIVATIVES($\boldsymbol{J}$)
$\bar{\boldsymbol{C}}, \frac{\partial \bar{\boldsymbol{M}}}{\partial \boldsymbol{\Phi}} \leftarrow$ CORIOLIS MATRIX($\dot{\boldsymbol{\Phi}}_n, \boldsymbol{J}, \frac{\partial \boldsymbol{J}}{\partial \boldsymbol{\Phi}}$)
$\boldsymbol{M}, \boldsymbol{C}, \boldsymbol{E}, \boldsymbol{B}, \boldsymbol{S}, \mathbb{B}, \mathbb{S}, \dot{\boldsymbol{\Phi}}(\boldsymbol{\Phi}, \dot{\boldsymbol{q}}, \boldsymbol{\mu}) \leftarrow$ DYNAMICS REDUCTION($\dot{\boldsymbol{\Phi}}_n, \bar{\boldsymbol{J}}^s_0, \bar{\boldsymbol{M}}, \bar{\boldsymbol{C}}, \frac{\partial \bar{\boldsymbol{M}}}{\partial \boldsymbol{\Phi}}, \frac{\partial \bar{\boldsymbol{J}}^s_0}{\partial \boldsymbol{\Phi}}$)
$\boldsymbol{\tau}_m \leftarrow$ OUTPUT TRACKING CONTROLLER($\dot{\boldsymbol{\Phi}}(\boldsymbol{\Phi}, \dot{\boldsymbol{q}}, \boldsymbol{\mu}), \bar{\boldsymbol{g}}_{sN}, \bar{\boldsymbol{V}}^b_{sN}, \dot{\bar{\boldsymbol{V}}}^b_{sN}, \boldsymbol{M}, \boldsymbol{C}, \boldsymbol{E}, \boldsymbol{B}, \mathbb{B}, \boldsymbol{J}^b_N$)
Define the system of ODEs $\frac{d\boldsymbol{Y}}{dt}$ from equation (6.2) using $\boldsymbol{\tau}_m$
end function

Since all of the aforementioned functions for the dynamics, kinematics, dynamic reduction, and output tracking controller depend on the matrix $\boldsymbol{Y}_n$, the numeric integrator must run the entire control process four times to estimate $\boldsymbol{Y}_{n+1}$. Evidently, the output of the motion propagation algorithm in this case is the matrix $\boldsymbol{Y}_{n+1}$ for use in the subsequent timestep.

Algorithm 10 Motion Propagation Algorithm

Input: 1) Current system states: $\boldsymbol{Y}_n$
2) Desired trajectory: $\bar{\boldsymbol{g}}_{sN}$, $\bar{\boldsymbol{V}}^b_{sN}$, $\dot{\bar{\boldsymbol{V}}}^b_{sN}$

Output: Integrated system states for next timestep: $\boldsymbol{Y}_{n+1}$

function MOTION PROPAGATION($\boldsymbol{Y}_n$, $\bar{\boldsymbol{g}}_{sN}$, $\bar{\boldsymbol{V}}^b_{sN}$, $\dot{\bar{\boldsymbol{V}}}^b_{sN}$)

$\boldsymbol{K}_1 \leftarrow \text{ODEs}(\bar{\boldsymbol{g}}_{sN}, \bar{\boldsymbol{V}}^b_{sN}, \dot{\bar{\boldsymbol{V}}}^b_{sN}, \boldsymbol{Y}_n)\ dt$

$\boldsymbol{K}_2 \leftarrow \text{ODEs}(\bar{\boldsymbol{g}}_{sN}, \bar{\boldsymbol{V}}^b_{sN}, \dot{\bar{\boldsymbol{V}}}^b_{sN}, (\boldsymbol{Y}_n + \frac{\boldsymbol{K}_1}{2})\ dt$

$\boldsymbol{K}_3 \leftarrow \text{ODEs}(\bar{\boldsymbol{g}}_{sN}, \bar{\boldsymbol{V}}^b_{sN}, \dot{\bar{\boldsymbol{V}}}^b_{sN}, (\boldsymbol{Y}_n + \frac{\boldsymbol{K}_2}{2})\ dt$

$\boldsymbol{K}_4 \leftarrow \text{ODEs}(\bar{\boldsymbol{g}}_{sN}, \bar{\boldsymbol{V}}^b_{sN}, \dot{\bar{\boldsymbol{V}}}^b_{sN}, (\boldsymbol{Y}_n + \boldsymbol{K}_3))\ dt$

$\boldsymbol{Y}_{n+1} \leftarrow \boldsymbol{Y}_n + \frac{1}{6}(\boldsymbol{K}_1 + 2\boldsymbol{K}_2 + 2\boldsymbol{K}_3 + \boldsymbol{K}_4)$

end function

6.3 Multi-Body System Description

Referring back to the proposed mission architecture for OOS industrialization outlined in Chapter 3, the preferred space manipulator design contains a single N-link manipulator attached to the spacecraft's base. Consequently, this simulation models a 2-link, 7-DOF shoulder-elbow-wrist space manipulator system illustrated in Figure 6.2. As shown, the full space manipulator system contains 13 twists: 3 translational and 3 rotational twists for the base motion, and 7 rotational twists to define the motion of the manipulator. The variables Φ_j appearing in Figure 6.2 indicate the direction of positive rotation or translation associated with the j^{th} degree of freedom in the system. For simulation purposes, the two spherical joints (located at the manipulator's shoulder and wrist) are modeled as three individual revolute joints, separated by small masses. Note that such design of spherical joints in the simulation are still considered multi-DOF joints as they define the motion in $\text{SO}(3)$ between two rigid bodies possessing comparatively large masses. Provided this

modelling of the spherical joints, the simulation replicates a space manipulator system with 10 bodies ($N = 9$) and 13 degrees of freedom ($N_D = 7$).

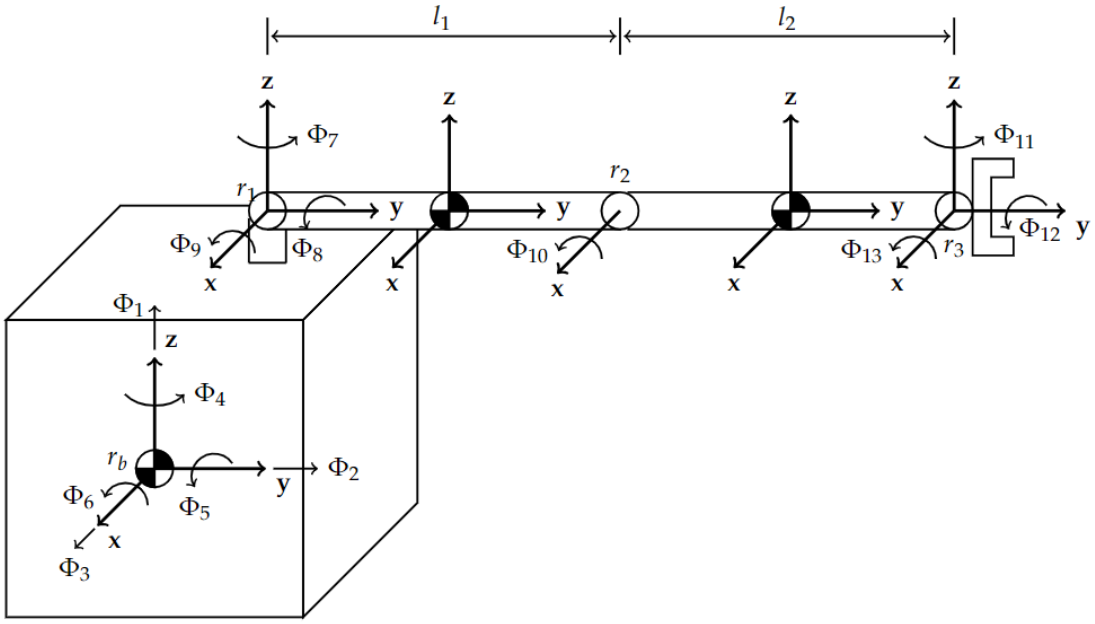


Figure 6.2: Simulated free-floating space manipulator system

Table 6.1 contains the physical properties of each body in the simulated system. Note that l, w, h refer to the body's length, width, and height dimensions measured along the y, x, and z axes of the body coordinate frames, respectively. The moments of inertia for each body are computed based on the geometries and masses provided in Table 6.1 with the assumption of constant densities. The densities of the components in the simulated system differ

For body k, the corresponding inertia tensor $\boldsymbol{I}_k$ is calculated as follows,

$$\boldsymbol{I}_k = \begin{bmatrix} \frac{m_k}{12}(l^2+h^2) & 0 & 0 \\ 0 & \frac{m_k}{12}(w^2+h^2) & 0 \\ 0 & 0 & \frac{m_k}{12}(l^2+w^2) \end{bmatrix} \tag{6.8}$$

where m_k is the mass of body k. Additionally, the center of masses of each body are assumed to be located at their geometric centers, as specified by the centroids illustrated in

Table 6.1: Physical Properties of Simulated System

Body	**Index** (k)	**Dimension** [m] (l, w, h)	**Mass** [Kg]
Base	0	(1, 1, 1)	200
Shoulder bodies	1, 2, 3	(0.05, 0.1, 0.1)	0.5
Link 1	4	(2, 0.1, 0.1)	2
Link 2	5	(2, 0.1, 0.1)	2
Wrist bodies	6, 7, 8	(0.04, 0.04, 0.04)	0.1
End-effector	9	(0.05, 0.05, 0.05)	0.5

Fig. 6.2. The spatial coordinate frame is initially located at the center of mass of the base body, and the initial spatial positions of the joints and end-effector in metres are as follows:

$$\begin{aligned} \boldsymbol{p}_{shoulder} &= [0.1 \;\; 0.5 \;\; 0.6]^T \\ \boldsymbol{p}_{elbow} &= [0.1 \;\; 2.5 \;\; 0.6]^T \\ \boldsymbol{p}_{wrist} &= [0.14 \;\; 4.5 \;\; 0.64]^T \\ \boldsymbol{p}_{end-effector} = \boldsymbol{p}_N &= [0.14 \;\; 4.55 \;\; 0.64]^T. \end{aligned}$$

Note that $\boldsymbol{p}_{shoulder}$ and $\boldsymbol{p}_{wrist}$ refer to the locations of the first members in the shoulder and wrist clusters, corresponding to the motions about the z-axis Φ_7 and Φ_{11}, respectively. The centers of the subsequent joint bodies are located at a width's distance away in the x-direction (defining the axes of rotation for Φ_8 and Φ_{12}), followed by an additional offset distance equal to the respective joint bodies' height in the z-direction (defining the axes of rotation for Φ_9 and Φ_{13}) to complete the spherical joint clusters. Link 1 and the end-effector are attached to the last joint members (i.e., the revolute joints with motion about the x-axes) in the shoulder and wrist joints, respectively.

Based on the body coordinate frames for each member in the system and the axes of motion illustrated in Fig. 6.2, the corresponding joint twists are determined using the twist definition described in equations A.13 and A.14. First, for the three translational motions of the spacecraft base (Φ_1, Φ_2, Φ_3), the following defines the unit vectors in the directions

of translation,

$$\boldsymbol{v}_1 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \qquad \boldsymbol{v}_2 = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \qquad \boldsymbol{v}_3 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \tag{6.9}$$

resulting in the translational twists for the base motion to be defined as,

$$\boldsymbol{\xi}_1 = \begin{bmatrix} \boldsymbol{v}_1 \\ \mathbf{0}_{3\times 1} \end{bmatrix}, \qquad \boldsymbol{\xi}_2 = \begin{bmatrix} \boldsymbol{v}_2 \\ \mathbf{0}_{3\times 1} \end{bmatrix}, \qquad \boldsymbol{\xi}_3 = \begin{bmatrix} \boldsymbol{v}_3 \\ \mathbf{0}_{3\times 1} \end{bmatrix}. \tag{6.10}$$

Regarding the remaining revolute motions for the spacecraft base and manipulator, the twist axes for each rotational motion are defined along the following axes,

$$\boldsymbol{w}_4 = \boldsymbol{w}_7 = \boldsymbol{w}_{11} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{6.11}$$

$$\boldsymbol{w}_5 = \boldsymbol{w}_8 = \boldsymbol{w}_{12} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \tag{6.12}$$

$$\boldsymbol{w}_6 = \boldsymbol{w}_9 = \boldsymbol{w}_{10} = \boldsymbol{w}_{13} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}. \tag{6.13}$$

To define the linear velocity component of the rotational twists, the locations of the points along each twist axis with respect to the spatial coordinate frame are as follows,

$$\boldsymbol{r}_b = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \tag{6.14}$$

$$\boldsymbol{r}_7 = \begin{bmatrix} 0 \\ 0.5 \\ 0.5 \end{bmatrix} \quad \boldsymbol{r}_8 = \begin{bmatrix} 0.1 \\ 0.5 \\ 0.5 \end{bmatrix} \quad \boldsymbol{r}_9 = \begin{bmatrix} 0.1 \\ 0.5 \\ 0.6 \end{bmatrix}, \tag{6.15}$$

$$\boldsymbol{r}_{10} = \begin{bmatrix} 0.1 \\ 2.5 \\ 0.6 \end{bmatrix}, \tag{6.16}$$

$$\boldsymbol{r}_{11} = \begin{bmatrix} 0.1 \\ 4.5 \\ 0.6 \end{bmatrix} \quad \boldsymbol{r}_{12} = \begin{bmatrix} 0.14 \\ 4.5 \\ 0.6 \end{bmatrix} \quad \boldsymbol{r}_{13} = \begin{bmatrix} 0.14 \\ 4.5 \\ 0.64 \end{bmatrix}, \tag{6.17}$$

where the coordinates are a consequence of the dimensions provided in Table 6.1. The initial homogeneous transformation matrices between the spatial and body coordinate frames $\boldsymbol{g}_{sk}(0)$ make use of the spatial position vectors above, and the assumption that all body coordinate frames are aligned with the spatial frame. This leads to the following general definition for $\boldsymbol{g}_{sk}(0)$:

$$\boldsymbol{g}_{sk}(0) = \begin{bmatrix} \mathbf{1}_{3\times 3} & \boldsymbol{r}_k \\ \mathbf{0} & 1 \end{bmatrix}. \tag{6.18}$$

6.4 Simulation Validation

To ensure that the motion of a free-floating space manipulator is being accurately replicated in the developed Python simulator, the validity of the dynamic and kinematic algorithms must be analyzed. The validity test conducted in this section develops an equivalent space

manipulator model, as that described in the previous section, to compare the system's motion under different dynamic and kinematic scenarios. A reference 2-link shoulder-elbow-wrist space manipulator system with identical physical parameters to those in Table 6.1 is thus modelled using the Simscape Multibody package within MATLAB/Simulink. Several different initial conditions were provided to both the Python simulator and the Simscape model to confirm the functionality and implementation of the forward kinematics, differential kinematics, and dynamics algorithms. In this section, the specific procedures for testing such algorithms are described, and the corresponding validation results are discussed. To start, the following subsection details the construction of the reference space manipulator system in Simscape.

6.4.1 Simscape Model

Simscape Multibody is a modelling tool, provided within the Simulink environment, for replicating the 3-dimensional motion of multi-body systems. The Simscape Multibody package provides blocks which model single rigid bodies (known as solid blocks), single and multi-DOF joints, and sensors to measure the positions, velocities, and accelerations of both rigid bodies and individual joint motions. Connecting rigid body blocks with an appropriate joint allows for the modelling of any mechanical multi-body system. For example, modelling an elbow joint in a 2-link manipulator requires coordinate frames at the ends of the two links (each modelled by a solid block) such that they can be connected to a revolute joint block. The revolute joint block attaches these two coordinate frames at a coincident point to replicate the physical attachment. All joint blocks in Simscape allow for actuation by an external force or moment, in turn moving the entire multi-body system accordingly. This resulting motion stems from an internal derivation of the system's equations of motion, based on the physical and geometrical properties of the designed multi-body system. The physical properties of a rigid body, such as geometry and mass, are specified within Simscape's solid block, which then automatically computes a corresponding inertia matrix if not otherwise specified.

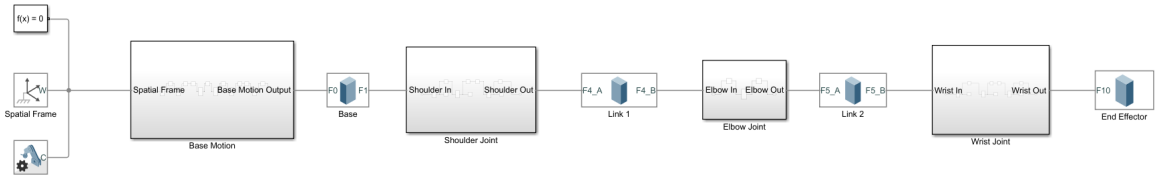


Figure 6.3: 2-Link space manipulator Simscape model.

Figure 6.3 displays the high level Simscape model of the equivalent 2-link space manipulator system presented in section 6.3. As shown, the Simscape model begins with a spatial coordinate frame, defined to be equivalently aligned with the Python model, and includes subdivisions which define each joint's motion in the system. Following the product of exponentials formulation, the 6-DOF motion of the base is modelled as consecutive prismatic and revolute joints, all coincident at the base's center of mass and connected in the same order as indicated by Figure 6.2 (where the system's first motion corresponds to the z translation of the base).

Figure 6.4 shows the internal components of the *Base Motion* subsystem to demonstrate the modelling of the free-floating base in Simscape. Here, the joints are connected from left to right, where the first three motions of the base are prismatic along the z, y,and x axes (corresponding to Φ_1, Φ_2, and Φ_3 respectively), followed by three revolute motions Φ_4, Φ_5, and Φ_6 about the z, y,and x axes respectively. Note that the prismatic and revolute joint blocks in Simscape are designed to only translate and rotate about the z-axis of the input coordinate frame (i.e., the coordinate frame connected to the B input port on the joint block). In this case, the spatial coordinate frame is input to the *Base Motion* subsystem with the intended prismatic motion Φ_1 already being aligned along the z-axis of this coordinate frame. For the subsequent linear motion along the y-axis, a coordinate transformation is required to align the z-axis of the input coordinate with the y-axis of the spatial frame (i.e., in the direction of the intended motion). This coordinate transformation is performed using Simscape's rigid transformation block, indicated in Figure 6.4 by the label $Z-Y$. This label specifies that the coordinate transformation is rotating the z-axis of the coordinate frame output by the Φ_1 prismatic joint to be aligned with the y-axis of the spatial frame. The coordinate transformation process is repeated for the prismatic joint along the x-axis, and subsequently for the revolute joints describing the base's rotation about the y and x axes of its body frame. The output of the *Base Motion* subsystem is the coordinate frame following the combination of the three translational and three rotational motions associated with the base. This final coordinate frame is then input into the solid block defining the spacecraft's base, as shown by Figure 6.3. The input $F0$ signifies the signal connection

between a coordinate frame $F0$ located at the center of mass of the base and the coordinate frame output by the *Base Motion* subsystem.

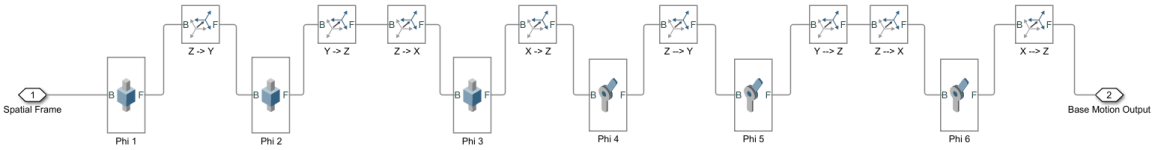


Figure 6.4: Base motion subsystem components

Equivalent to the simulated system in Python, spherical joints are simulated in Simscape as the amalgamation of three individual revolute joints, each connected by small masses to replicate physical actuators. Figure 6.5 expands the *Shoulder Joint* subsystem to demonstrate the small mass cluster forming this spherical joint. As shown in the overall Simscape model, the input to the shoulder joint subsystem is the $F1$ coordinate frame output by the base block. The origin of this $F1$ coordinate frame specifies the placement of the shoulder joint on the base body, and thus shares the coordinates of the $\boldsymbol{r}_7$ axis point defined in section 6.3. This coordinate frame similarly defines the geometric center of the actuator body corresponding to the z rotation of the shoulder joint (coordinate frame R associated with the *Body 1* solid block). To simulate the shoulder's rotational motion about the y-axis, a transformation block is again required to rotate the coordinate frame attached to the geometric center of body 1 by -90° about its x-axis. This coordinate transformation aligns the axis of motion of the Φ_8 joint with the y-axis of the input coordinate frame. The shoulder's joint motion about the y axis is also located at an offset distance along the x-axis of the $F1$ frame, as described in section 6.3. The origin of resulting coordinate frame is now coincident with the point $\boldsymbol{r}_8$, defining the geometric center of body 2. The previous process is repeated for the implementation of the shoulder joint's revolute motion about the x-axis. Between the last revolute motion and the shoulder joint's output coordinate frame, an additional transformation is required to reset output coordinate frame into its proper orientation (undoing the transformation required for simulating Φ_9 in the correct direction). The output coordinate frame defines the location in which the manipulator's first link attaches to the shoulder joint.

Referring back to the overall Simscape model in Figure 6.3, the input port to the *Link 1* solid block, labeled $F4_A$, refers to a coordinate frame whose origin is placed at the link's geometric center with respect to the $x-z$ plane, and at one end of the link. As previously

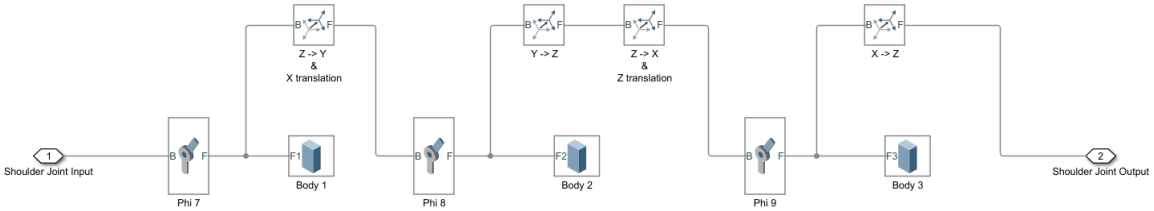


Figure 6.5: Shoulder joint subsystem components

mentioned, this $F4_A$ coordinate frame is connected to the output of the shoulder joint subsystem in order to define the spherical motion of link 1. A coordinate frame denoted $F4_B$ is placed at the opposite end of the link (i.e., at the link's length away along the y-axis) for subsequent connection with the elbow joint. The elbow joint subsystem, shown in Figure 6.6 simply contains a single coordinate transformation required to align the elbow rotation with the x-axis of the spatial frame, in the system's initial configuration. Analogous to shoulder joint, an equivalently opposite transformation is required to reset the output coordinate frame to the correct orientation. This output coordinate frame attaches one end of the second manipulator link (specified by the $F5_A$ coordinate frame) to the elbow joint. Note that there is no actuator body associated with the elbow joint, signifying that the origins of the $F4_B$ and $F5_A$ coordinate frames are coincident.

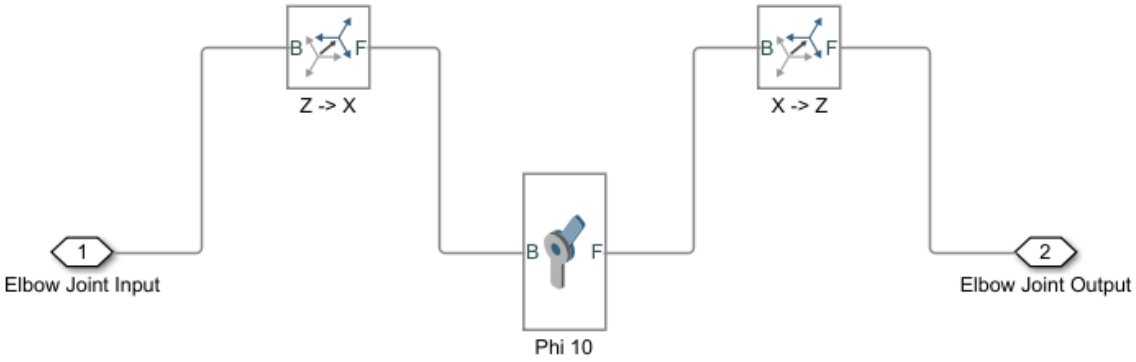


Figure 6.6: Elbow joint subsystem componentss

Finally, the output coordinate frame $F5_B$ of the *Link 2* block (located at the opposite end of the second link along the y-axis) is input to the wrist joint subsystem for connection with body 6 in the system (associated with the z rotational motion of the wrist joint). The wrist subsystem is designed in the exact same manner as the shoulder joint, as demonstrated in Figure 6.7 showing the expanded wrist subsystem. The output of the wrist subsystem defines a coordinate frame, following the three offset revolute motions of the wrist, located

at the end of the last actuator body (along the y-axis), in the geometric center of the $x-z$ plane. The output of the wrist joint is then attached to the $F12$ coordinate frame, denoting a frame located on the leftmost (with respect to the y-axis) outer face of the end-effector, and in this face's geometric center.

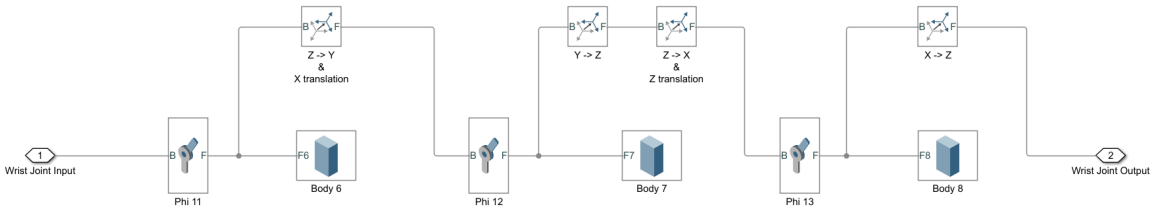


Figure 6.7: Wrist joint subsystem components

This space manipulator model in Simscape can now be manipulated by defining various initial conditions to the joint blocks, as well as external actuation torques. To start, each revolute and prismatic joint block in Simscape allows for their initial position and velocity to be internally defined. As a result, the initial conditions $\mathbf{\Phi}_{initial}$ and $\dot{\mathbf{\Phi}}_{initial}$ can be input to the Simscape model, and the system's motion will propagate accordingly. Additionally, the joint blocks include an actuation property where either a torque or a force may be provided as an input signal to the block. The input actuation may be in the form of a timeseries defined in a MATLAB script connected to the Simscape model. The timeseries specifies the torque or force value provided to the revolute or prismatic joint at each timestep in the simulation. As an example of how to actuate joints in a specified manner, Figure 6.8 depicts the introduction of torque signals to the revolute joints in the shoulder subsystem of the Simscape model. As shown, the revolute joints have an additional input port labeled t for which the torques T_7, T_8, T_9 are input as timeseries sent from a MATLAB script. Following the actuation of the joint blocks, the system's motion is automatically computed by Simscape.

Joint blocks in Simscape additionally provide sensing capabilities, useful for comparing the resulting motion of the generalized coordinates with the Python model. A joint's position, velocity, and acceleration can be set as output signals to the block, and subsequently sent to the MATLAB workspace for subsequent post processing. This sensing feature in the joints allows for the joint variables $\mathbf{\Phi}$, $\dot{\mathbf{\Phi}}$ and $\ddot{\mathbf{\Phi}}$ to be compared at each time step in the simulation. Moreover, Simscape's transform sensor block provides measurements of

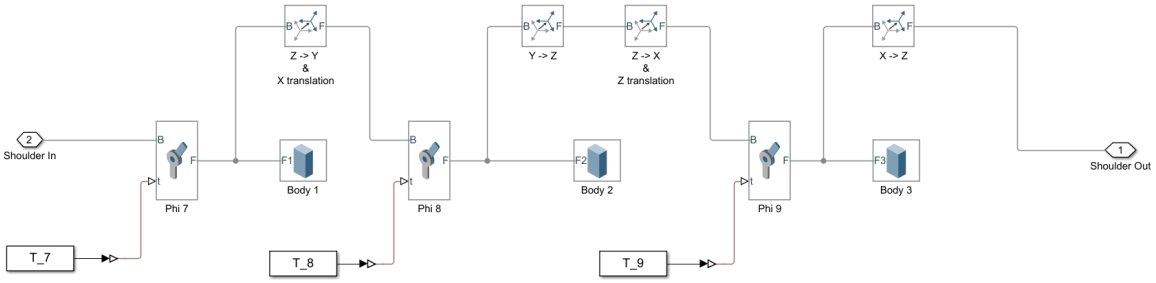


Figure 6.8: Joint actuation example for shoulder joint subsystem

each body's pose and velocity vector. In this model, these sensing blocks input the spatial coordinate frame as a reference, and attach to the desired body coordinate frame to be measured. The measured output signals include a quaternion defining the frame's orientation, the frame's angular velocity about each axis, and the linear position and velocity of the attached coordinate frame. These sensors measure and output the body velocities expressed in the spatial reference frame. To obtain the body velocity vectors expressed in the body coordinate frame (for comparison with the velocities output by the Python simulator's differential kinematics), the output quaternion is first converted to a rotation matrix in MATLAB using the built-in *quat2rotm* function. Pre-multiplying this rotation matrix $\boldsymbol{R}_{ks}$ with the measured body angular and linear velocities in Simscape provides their representations in the body frame, as desired. Figure 6.9 illustrates the introduction of the sensor blocks in the Simscape model.

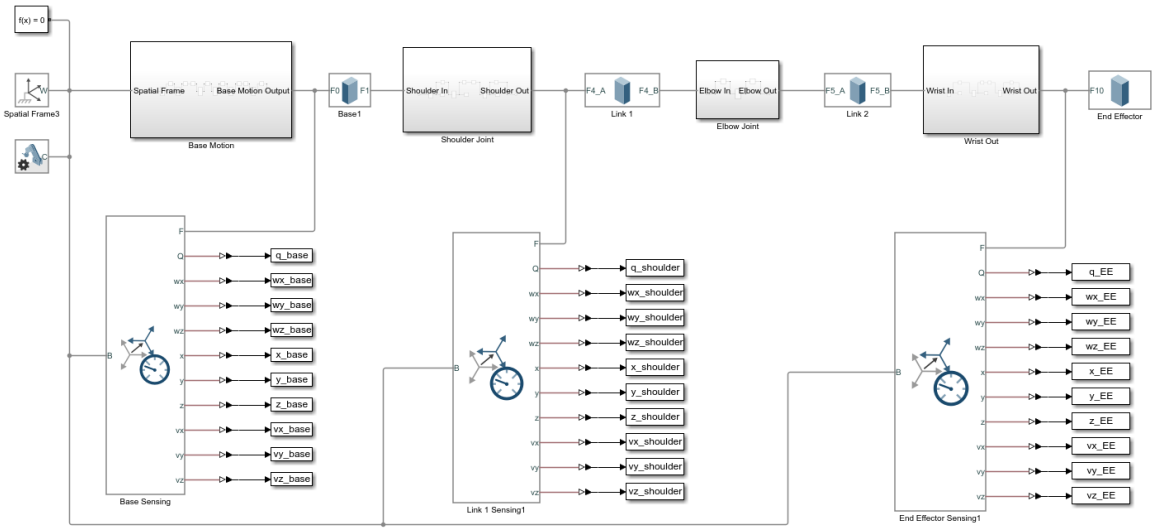


Figure 6.9: Sensing of the space manipulator system

The resulting 3D realization of the fully connected Simscape model is illustrated in Figure 6.10, depicting the space manipulator system in the configuration of Φ_9 = 45° and Φ_{10} = -90°. Note that the blue, red, and green bodies at the shoulder and wrist joints represent the actuator bodies associated with the *z, y* and *x* rotations of the spherical joints, respectively. Additionally note that there is no actuator member at the elbow joint, as previously mentioned.

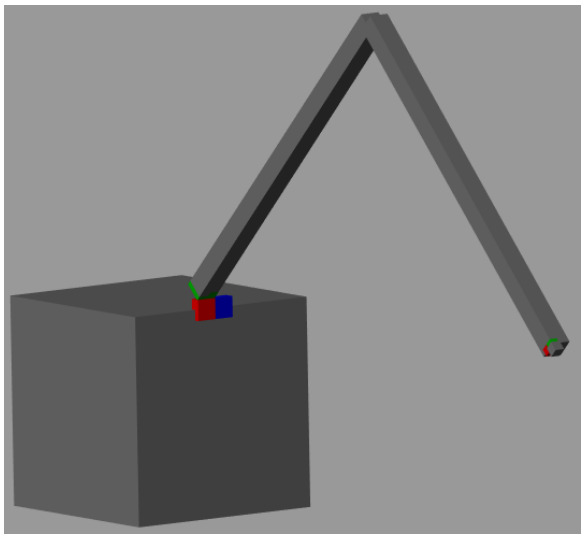


Figure 6.10: 3-Dimensional space manipulator model in Simscape

6.4.2 Validation Procedure

The methods for testing and confirming the dynamics and kinematics algorithms implemented in the Python simulator are discussed in this subsection. As previously mentioned, the purpose for constructing an equivalent Simscape model is to produce reference values for which the dynamic and kinematic results in Python can be compared against. The testing sequence begins by validating the forward kinematics model, as this function is required in both the differential kinematics and dynamics processes. Once confirmed accurate, the computation for each body's Jacobian matrix is tested, prior to verifying the dynamics model of the system. In testing the dynamics, the calculation of the system inertia matrix is first individually validated in order for the inertia component of the dynamic behaviour to be confirmed accurate before introducing the Coriolis and centrifugal effects. The following sections discuss the procedures for performing each validation test, and subsequently discuss the findings from the Python and Simscape comparisons.

Forward Kinematics Validation

The testing procedure for confirming the forward kinematics involves comparing the position vectors of various coordinate frames in the system. In this case, the pose of the base, shoulder, elbow, and wrist coordinate frames are compared. Recall that the shoulder and wrist coordinate frames refer to the frames located at the end of the last actuator body in the respective joint cluster. Examining the position and orientation of each joint coordinate frame ensures that the product of exponentials function to compute the forward kinematics in Python is being computed correctly, provided generalized coordinate positions.

Several different initial configurations are specified for the space manipulator system in both the Python and Simscape simulations. Provided a configuration vector $\mathbf{\Phi}$, the corresponding homogeneous transformations to each joint coordinate frame are computed (given the initial homogeneous transformations defined in section 6.2) in Python, and compared with those measured in Simscape. As mentioned, Simscape measures orientations in terms of quaternions, thus requiring conversions to rotation matrices in the post processing. The x, y, and z components of each joint frame's position vector are directly output from the Simscape sensing block, allowing for immediate comparisons.

Fully assessing the forward kinematics method in Python involves testing multiple configurations $\mathbf{\Phi}$ against the Simscape model in order to cover a wide range of the space manipulator's configuration space. In purely isolating the system's kinematics, the system's generalized coordinate velocities $\dot{\mathbf{\Phi}}$ are set to zero, and there is no torque actuation on the manipulator joints. In total, 10 different configurations are tested, all of which yielding matches between the joint poses computed in Python and Simscape up to 10 decimal places. Table F.1 in Appendix F contains the 10 different configurations tested in the forward kinematics validation process.

Differential Kinematics Validation

Validating the differential kinematics model in Python implies confirming the accuracy of the calculated Jacobian matrices for each body in the system. Recall the dependency on the product of exponentials formula in the Jacobian matrix definition, and note that the implementation of this formula is confirmed correct following the validation of the forward kinematics model. Therefore, any errors in the differential kinematics output are a direct result of errors in the implementation of the differential kinematics model. Comparing body

velocity vectors expressed in the body frame provides a methodology for assessing the Jacobian matrix computation in Python. More specifically, the velocity vectors corresponding to the base, shoulder, elbow, and wrist coordinate frames are chosen for comparison. It is important to test the differential kinematics model at each joint to ensure functionality throughout the system, as is necessary for the inertia and Coriolis matrix computations.

As previously mentioned, the sensing blocks in Simscape measure the attached coordinate frame's body velocity components expressed in the spatial coordinate frame. As part of the post-processing in MATLAB, the body velocity components output by Simscape are rotated into the body frame using the rotation matrix between the spatial frame and coordinate frame being measured. This post-processing rotation is required to compare the velocities output by Simscape with those calculated in Python (using the body Jacobian matrix and the vector of generalized coordinates $\dot{\mathbf{\Phi}}$). Equivalent initial generalized velocity vectors are provided and both simulations are run for a single time step. This ensures that the resulting joints' body velocities output by both simulations correspond to the same $\dot{\mathbf{\Phi}}$.

Since the Jacobian matrix is a function of the generalized coordinate positions, testing multiple configurations which cover different areas of the configuration space is again necessary to fully validate the differential kinematics. Additionally, various system velocity vectors are tested for each different configuration in order to analyze a body Jacobian matrix under numerous joint velocity conditions. The same 10 configurations used in the forward kinematics testing are applied in the differential kinematics procedure. For each configuration, 8 different combinations of generalized coordinate velocities are tested. Each testing case resulted in a match of up to 10 decimal places between the body velocities measured in Simscape and those calculated in Python. Table F.2 includes the 8 velocity vectors tested for each of the 10 configurations in Table F.1.

Inertia Matrix Validation

Progressing to the assessment of the Python simulator's dynamics model, we begin by analyzing the computation of the system inertia matrix.Isolating the inertia matrix contributions in the space manipulator's dynamics firstly requires setting all generalized coordinate velocities to zero. Recalling the Euler-Lagrange equations of motion in equation (4.40), a system with zero velocity removes all effects from Coriolis and centrifugal forces in the dynamics. Dynamically exciting a system (through external forces and torques applied to both the base and manipulator) which is initially at rest implies that the joint accelerations are influenced by the system's inertia until the system's generalized velocities become

non-negligible.

The validation testing procedure for the inertia matrix applies forces and torques to the space manipulators, and compares the subsequent generalized coordinate accelerations $\ddot{\mathbf{\Phi}}$. In the Python simulator, the vector $\ddot{\mathbf{\Phi}}$ is computed by the Euler-Lagrange equations of motion. Again, in order to neglect the Coriolis and centrifugal affects in the dynamic response, the system accelerations are only compared after the first timestep when the system velocities remain negligible. As the inertia matrix is a function of the system configuration, the inertia matrix for multiple orientations of the space manipulator are tested (the same orientations used in the forward and differential kinematics validation procedures) to ensure the inertia matrix's veracity throughout the system's workspace. For each configuration, 8 different combinations of base and manipulator joint forces and torques are applied in order to manipulate the space manipulator system in several different ways. Table F.3 demonstrates the torque vectors applied to each configuration in Table F.1 during the inertia matrix validation testing. The resulting generalized accelerations for every test case again produced matches of up to 10 decimal places between the two simulations, thus confirming the veracity of the inertia matrix implementation in Python.

Coriolis Matrix Validation

Finally, the implementation of the space manipulator's complete dynamics are validated by testing the Coriolis matrix computation in Python. Again, the testing procedure involves comparing the system accelerations output in Python against those from the Simscape simulation following the application of external forces and torques. As previously mentioned, the Coriolis and centrifugal influences become prominent when sufficient generalized velocities are reached. As such, in analyzing the Coriolis matrix, the space manipulator system is initialized with a non-zero velocity vector. The resulting dynamic response evidently contains influences from both the inertia of the system and the Coriolis and centrifugal forces. Since the inertia matrix method has already been confirmed accurate, any discrepancies in the generalized coordinate accelerations is a result of errors with the Coriolis matrix computation in Python.

A similar testing procedure to that for the inertia matrix is implemented for validating the Coriolis matrix. Since the Coriolis matrix is a function of both the system position and velocity, several configurations paired with several different generalized coordinate velocities are tested (similar to the Jacobian matrix validation procedure). For each configuration and system velocity pairing, the same torque vectors used in the inertia matrix validation

are applied here to obtain multiple different dynamic responses from the system. As the space manipulator system is already initialized with non-zero velocity, the accelerations output after a single time step are used for comparison between the Python and Simscape simulations. The specific testing conditions performed are again contained in Appendix F, where tables F.1, F.2, and F.3 are all used in the testing scenarios. The resulting accelerations are shown to match up to 10 decimal places, consequently confirming the Coriolis matrix computation and thus the complete dynamical behaviour simulated free-floating space manipulator in Python.

With the confirmation of the Coriolis matrix implementation in Python, the veracity of the dynamics and kinematics models in the simulator are fully validated. The following chapter makes use of the replicated space manipulator system in the developed Python simulator to test the performance of the linear and full pose output tracking controllers structured in chapter 5. Such performance testing involves analyzing the controlled system's response under various trajectory following tasks.

Chapter 7

Simulation Results

7.1 Summary

In this section, the efficacy of the proposed linear and full pose output tracking controllers presented in chapter 5 are evaluated under various trajectory following scenarios. These scenarios are performed using the Python simulation platform discussed in the previous chapter. To start, control over the linear end-effector motion is considered for the free-floating space manipulator system described in Figure 6.2 with zero momentum. The end-effector's linear trajectory following capabilities are tested for a trivial trapezoidal trajectory, followed by a more complex circular path in the spatial $x-y$ plane. The same trajectories are applied to the system possessing non-zero momentum to demonstrate the controller's ability to dissipate an initial motion in the output and achieve path following. To assess the robustness of the linear controller, uncertainties are introduced in the system's mass. Regarding the full pose controller, the system is again initially analyzed under zero momentum, however for a trapezoidal trajectory in both the angular and linear components of the end-effector. The same trajectory is then tested for the system being initially in motion, where the end-effector remains capable of reaching the target position within the same time period. Finally, the singularity accommodation methodology is tested for a trajectory which forces the space manipulator to pass in proximity to a singular configuration. The progression of the system's manipulability and the damping factor in the SR-inverse method are observed to ensure safe joint actuation while operating in the singular region.

7.2 Linear Output Tracking Controller Results

We begin, in this section, by demonstrating the efficacy of the linear output tracking controller developed in section 5.3. Starting with the more trivial control case, this section first assesses the linear controller's prowess for controlling a space manipulator system initially at rest (i.e. $\boldsymbol{\Phi} = 0$ and thus $\boldsymbol{\mu} = 0$). Subsequently, the same system is tested under non-zero momentum conditions to demonstrate the control law's ability to dissipate the initial motion in the end-effector, and stabilize the output to the desired path. The linear output tracking controller is applied to the single-arm 13-DOF space manipulator described in section 6.3. For all simulated control scenarios presented in this section, the Python simulator detailed in the previous section is run for a time step of $dt = 0.01$ s and for PID gains corresponding to an a value of 4:

$$\boldsymbol{K}_p = \begin{bmatrix} 48 & 0 & 0 \\ 0 & 48 & 0 \\ 0 & 0 & 48 \end{bmatrix}, \quad \boldsymbol{K}_d = \begin{bmatrix} 12 & 0 & 0 \\ 0 & 12 & 0 \\ 0 & 0 & 12 \end{bmatrix}, \quad \boldsymbol{K}_i = \begin{bmatrix} 64 & 0 & 0 \\ 0 & 64 & 0 \\ 0 & 0 & 64 \end{bmatrix}. \tag{7.1}$$

For each testing scenario of the linear output tracking controller (unless otherwise specified), the space manipulator system is initialized with the following non-zero joint positions at the shoulder and elbow joints: $\Phi_{9,ini} = 45^\circ$ and $\Phi_{10,ini} = -90^\circ$, producing an initial end-effector position of $\boldsymbol{p}_{N,ini} = [0.14 \;\; 3.36 \;\; 0.63]^T$ m. This choice of $\boldsymbol{\Phi}_{ini}$ ensures that the space manipulator is starting from a well defined system, away from the region of singular configurations. In the following subsections, the controlled space manipulator is tested under various trajectory following scenarios which approximate the pre-capture proximity operations involved during in-orbit docking procedures. Note that results of the control input and resulting output trajectory tracking are exclusively provided, however the base spacecraft is in motion throughout all trajectory following scenarios. Note that all figures regarding the trajectories of the base positions associated with each trajectory following task in this section are included in Appendix G.

7.2.1 Space Manipulator System with Zero Momentum

Regarding OOS missions involving cooperative targets, the capture points on the targets' structures can be controlled to a fixed position in space by way of the target satellites'

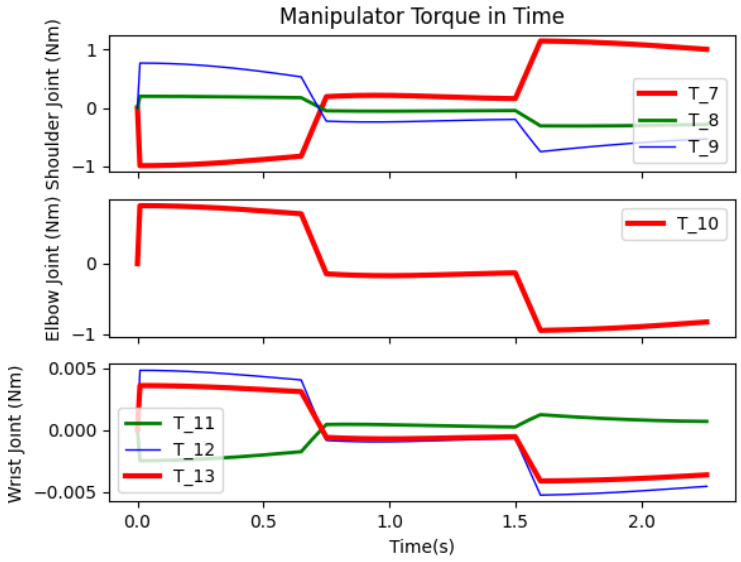
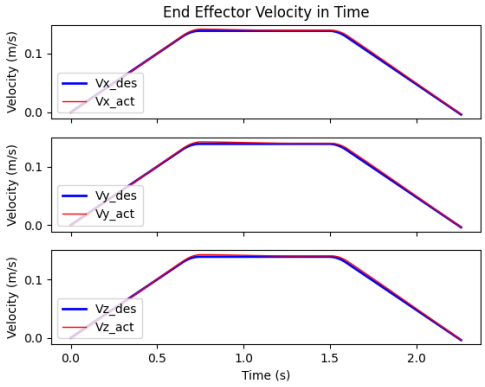


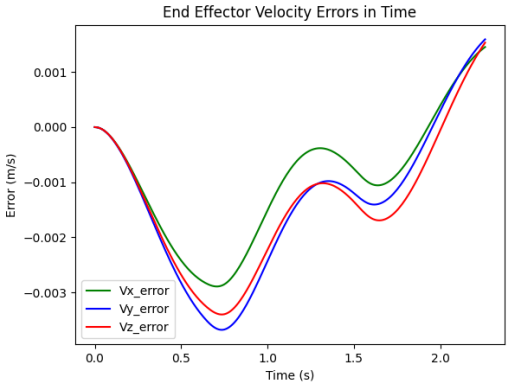
Figure 7.1: Control torques for the linear trapezoidal trajectory applied to a zero momentum system.

GNC systems. These missions significantly simplify the design of the desired trajectory, simply requiring a standard trajectory from the initial end-effector location to the fixed capture point. This subsection tests docking scenarios where the simulated space manipulator contains conserved zero momentum. All testing scenarios assume an obstacle-free path between the system's end-effector and the capture point throughout the simulation.

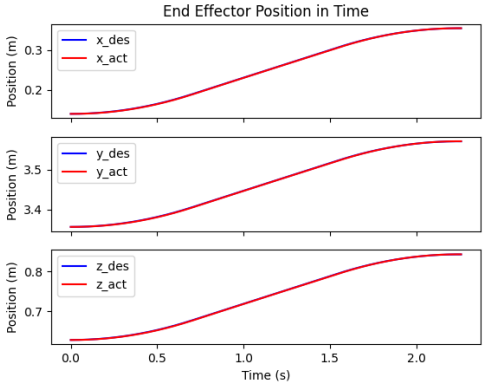
We start by considering a desired trajectory which connects the space manipulator's initial end-effector position with a target position of $\bar{\boldsymbol{p}}_{N,final} = [0.35 \;\; 3.57 \;\; 0.84]^T$ m. These two points can simply be connected by a line defined by an appropriate trapezoidal trajectory. For this capture scenario, the trapezoidal trajectory commands a constant acceleration of 0.2 m/s^2 for 0.65 s, followed by a smooth deceleration for 0.1 seconds to reach constant velocity. This deceleration period avoids sharp changes in the desired velocity, in turn requiring a smoother control action during the switch from constant acceleration to constant velocity in the output. The end-effector moves at constant velocity for 0.75 s followed by a constant deceleration of 0.2 m/s^2 for 0.65 s to reach the target position with no linear velocity in the output. The same smoothing period of gradual deceleration from constant velocity to constant deceleration is again implemented. The consequent time to reach the



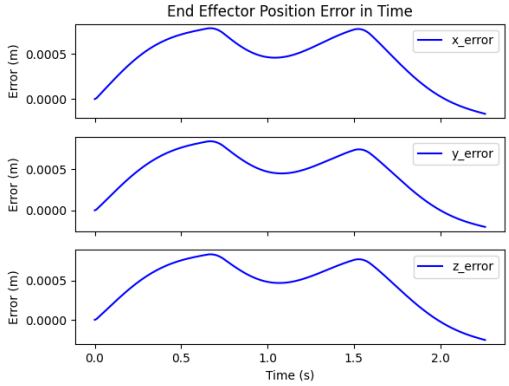
((a)) Desired and actual end-effector velocity



((b)) Velocity error



((c)) Desired and actual end-effector position



((d)) Tracking error

Figure 7.2: Output response to the linear trapezoidal trajectory applied to a zero momentum system.

target point approximately 36 cm away is 2.25 s. Note that the described trapezoidal trajectory is applied to all directions of the output's linear motion.

Figure 7.1 demonstrates the control torque resulting from the linear output tracking controller over the course of the trajectory following task. It is shown that the initial torque values required to accelerate the end-effector 0.2 m/s^2 from rest are all within a magnitude of 1 Nm of torque at each manipulator joint. As indicated by Figure 7.1, a greater control action is needed about the x and z axes as actuating in these directions controls the linear position of the output. Neglecting the insignificant actuation at the wrist joint, Figure 7.1 shows only around 0.2 Nm of torque about the y-axis of the shoulder joint (T_8), compared to magnitudes of around 1 Nm (T_7) and 1.5 Nm ($T_9 + T_{10}$) to move the end-effector along the z and x axes, respectively. The resulting end-effector motion is demonstrated in Figure 7.2. Accurate trajectory followings are achieved in both the end-effector's velocity and position trajectories, as conveyed by Figures 7.2(a) and 7.2(c) which show the desired and actual end-effector velocity and position trajectories along each axis in time, respectively. To better illustrate the tracking accuracy, Figures 7.2(b) and 7.2(d) demonstrate the velocity and position errors along each axis throughout the trajectory. As expected, the largest velocity errors (across all axes) occur just following the transition from constant acceleration to constant velocity; with the largest error being of magnitude 3.7 mm/s in the y-direction. The tracking errors are similarly greatest at the transition points between acceleration and constant velocity. As shown by Figure 7.2(d), the controller is able to track the desired trajectory within less than 1 mm along all axes. Therefore, for a relatively fast trajectory of travelling 36 cm in 2.25 s, the linear output tracking controller is able to achieve highly accurate path following, with minimal torque loads at the joints.

In this work, we are limited in testing docking scenarios which involve the capture of moving objects (typically non-cooperative targets) due to the complexities in designing trajectories for free-floating robots and moving targets (as discussed in section **??**). Research regarding complex trajectory designs is beyond the scope of this project, however we demonstrate the dexterity of the controlled space manipulator (using the linear output tracking controller) by commanding a circular trajectory in the $x-y$ plane of the spatial coordinate frame. In the simulation, a desired circular trajectory of radius $r = 0.2$ m is output by the trajectory planner. The start of the circular path, i.e., when $\theta_{circ} = 0$, corresponds to the initial position of the end-effector, with the initial linear motions being in the negative x and positive y directions. The desired motion commands a constant angular velocity of $\omega_{circ} = 1$ rad/s, resulting in the following desired circular trajectory in time,

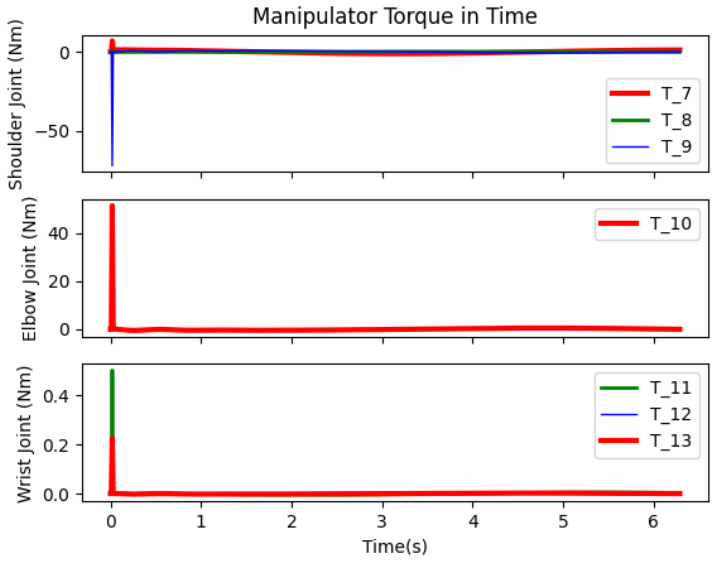
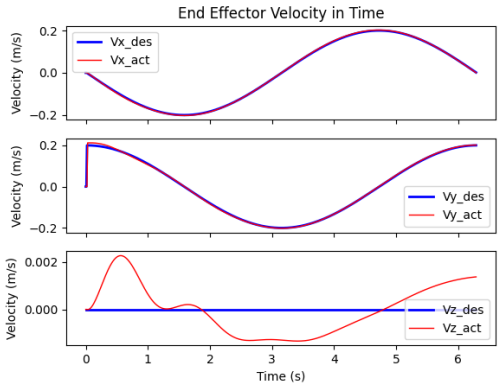


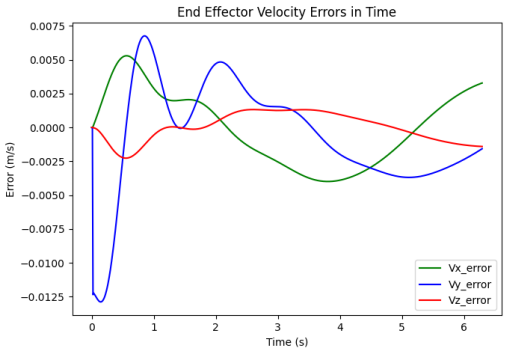
Figure 7.3: Control torques for the linear circular trajectory applied to a zero momentum system.

$$\bar{\boldsymbol{p}}_N = \begin{bmatrix} r\cos(\omega_{circ}t) \\ r\sin(\omega_{circ}t) \\ 0 \end{bmatrix} + \begin{bmatrix} x_{N,ini} - r \\ y_{N,ini} \\ z_{N,ini} \end{bmatrix}. \tag{7.2}$$

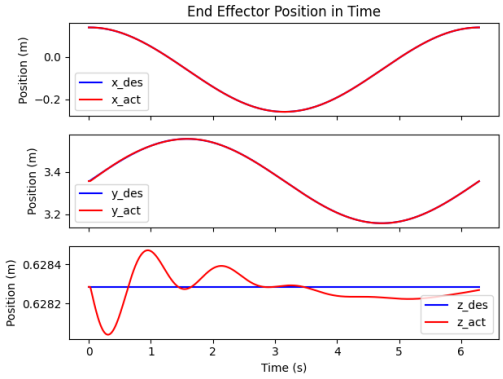
Figure 7.3 demonstrates the applied control torque on the manipulator joints during the circular trajectory following scenario. As shown, a significant initial torque of around 62 Nm about the x axis of the shoulder joint and approximately 50 Nm at the elbow joint are required for the end-effector to immediately follow the desired circular trajectory. This sudden torque input is a result of a large jump in the end-effector's velocity, as shown by Figure 7.4(a) which depicts the output's actual and desired velocities in each axis, due to the end-effector starting from rest. The circular trajectory requires an immediate velocity increase of 0.2 m/s along the y-axis, hence the large actuation about the x-axes of the manipulator joints. Following the initial jumps in actuation at the shoulder and elbow joints, the torque values at the shoulder vary within 2.5 Nm (in magnitude), and within 1 Nm at the elbow joint for the remainder of the simulation. An undershoot of around 1.3 cm/s occurs



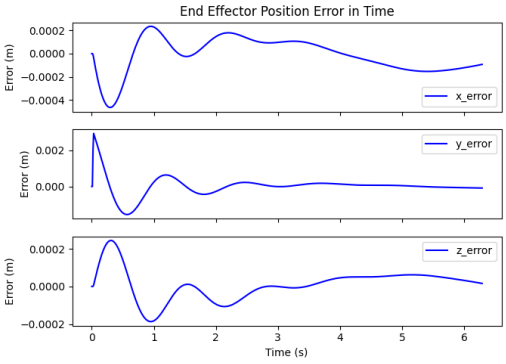
((a)) Desired and actual end-effector velocity



((b)) Velocity error



((c)) Desired and actual end-effector position



((d)) Tracking error

Figure 7.4: Output response to the linear circular trajectory applied to a zero momentum system.

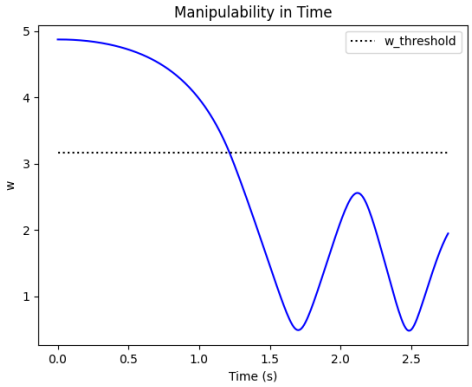
when performing this velocity jump, as displayed by Figure 7.4(b). This velocity error is shown to diminished to under 2 mm/s within 1.2 s. Figures 7.4(a) and 7.4(b) also indicate minimal residual motions in the z-direction of the output where no motion is expected (under 3 mm/s), and along the x-axis where the velocity error is held to within 5 mm/s at its peak. Moreover, Figures 7.4(c) and 7.4(d), showing the actual and desired trajectories and the corresponding tracking error along each axis of the output, demonstrate accurate output tracking along the circular path. As expected, the greatest position error occurs initially along the y-axis, however the controller is still able to achieve a path following of within 2 mm in this direction. Effectively perfect trajectory following is demonstrated in the x and z directions as tracking errors of under 1 mm are achieved.

The previous control tests demonstrated the tracking capabilities of the linear output tracking controller with the highest possible accuracy. As discussed in section C, the most accurate tracking performance corresponds to a singularity-free configuration, resulting in a well-defined system. We now wish to progress from the emphasis on accuracy and exhibit the proficiency of the singularity accommodation feature in the developed control structure. This evidently involves designing a desired trajectory which requires the space manipulator to pass near a known singular configuration. Starting from the following initial configuration:

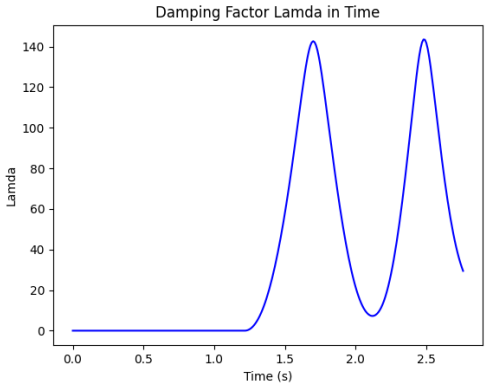
$$\mathbf{\Phi}_{ini} = [0,\ 0,\ 0,\ -28,\ 20,\ 15,\ 90,\ 20,\ -25,\ -60,\ 0,\ 0,\ 0]^T \text{deg}, \tag{7.3}$$

a trapezoidal trajectory which decelerates by a magnitude of 0.4 m/s^2 for 1.25 s, travels at constant velocity for 0.25 s, and accelerates by 0.4 m/s^2 for 1.25 s along all axes is provided. Figures 7.5(a) and 7.5(b) display the progression of the space manipulator's manipulability and damping factor throughout this trajectory following task. As indicated by the plot of manipulability, the system starts in a well-defined configuration with sufficient manipulability, and as the simulation progresses, the space manipulator becomes increasingly ill-conditioned.

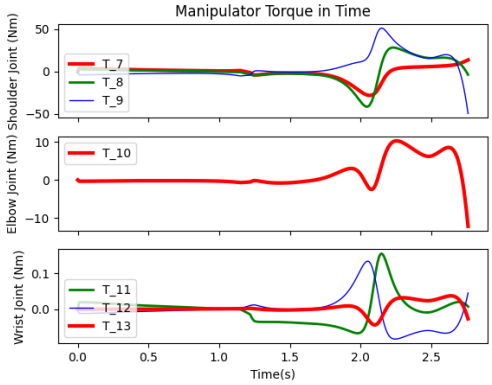
Figure 7.5(a) shows the system's manipulability to fall below the threshold value of $\sqrt{10}$ after about 1.2 s, consequently inducing a shift from the pseudo-inverse solution to the SR-inverse in the feedback linearization law. This shift is demonstrated by the introduction



((a)) Manipulability

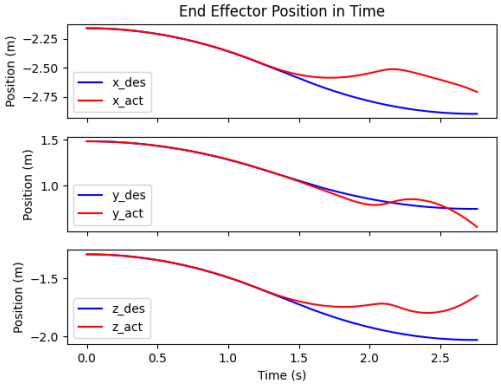


((b)) Damping factor

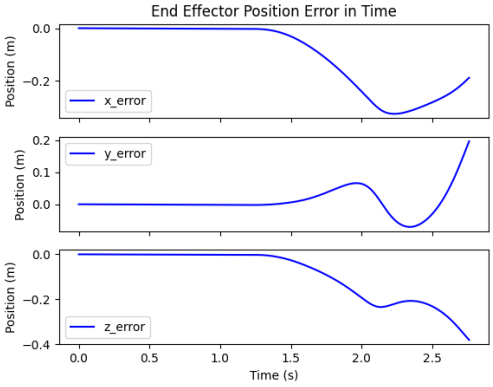


((c)) Control torque

Figure 7.5: Damped least squares parameters for SR-inverse in the linear controller with associated control input



((a)) Desired and actual end-effector position



((b)) Tracking error

Figure 7.6: Linear output tracking performance in the presence of a singularity

of a non-zero damping factor once the system enters the region of insufficient manipulability, as shown in Figure 7.5(b). The damping factor is shown to quickly increase as the system progresses towards the singularity. At the system's closest proximity to the singular configuration, occurring at 1.7 s, a manipulability of 0.5 and a maximum damping factor of 142 (where $\lambda_0 = 200$, as mentioned in section C) are reached. The increase in the damping factor is shown to effectively limit the actuation torques to within 10 Nm at the shoulder and elbow joints at the time of lowest manipulability, as per Figure 7.5(c) (demonstrating the control input in time). Figures 7.6(a) and 7.6(b) show the end-effector's linear position and the associated tracking error, respectively, to exhibit the resulting trajectory following capabilities of the space manipulator system with insufficient manipulability. Highly accurate trajectory following is achieved (within 2 mm) until around 1.4 s, afterwhich the end-effector is shown to deviate from the desired path by about 1 cm in all axes. This tracking error continues to increase by an order of magnitude as the damping factor similarly increases, and less emphasis is placed on the tracking accuracy. After 1.7 s, the space manipulator is shown in Figure 7.5(a) to become more well conditioned, in turn causing the damping factor to decrease. In this case, the controller attempts to regain trajectory following in the output, consequently allowing for larger control torques to be performed, as shown by Figure 7.5(c). Note that the torque levels at this point (still below the manipulability threshold) are within an acceptable range of 50 Nm at the shoulder and 10 Nm at the elbow. In trying to re-follow the desired path, the system again falls towards the singularity, causing the damping to increase and the manipulator joint torques to again be effectively maintained within a safe limit.

The final test of the linear output tracking controller (applied to a system with zero momentum) assesses the controller's robustness by introducing model uncertainties. Specifically, uncertainties are added to the masses of each body in the true space manipulator system (i.e., the model associated with the plant dynamics). Consequently, the inertia and Coriolis matrices for the actual system use masses which have been varied by a random amount. In the simulation, this random variation is determined by a random number generator with upper and lower limits of 20 and 10 percent of the associated mass's estimated value. Such variations are randomly chosen to be either added or subtracted to the estimated masses values. The physical properties included in Table 6.1 are consequently treated as the estimated values, and are used in the dynamics reduction and feedback linearization

Table 7.1: Mass Values used in the True Space Manipulator Model

Body Index	**True Mass** (kg)	**Mass Variation** (kg)	**Percent Variation**
0	228.91	28.91	14.45 %
1	0.442	-0.0579	11.59 %
2	0.408	-0.0921	18.41 %
3	0.443	-0.0572	11.44 %
4	2.215	0.215	10.76 %
5	2.367	0.367	18.34 %
6	0.0881	-0.0119	11.89 %
7	0.119	0.0193	19.31 %
8	0.0891	-0.0109	10.92 %
9	0.585	0.0852	17.04 %

processes. The uncertain system is tested using the same trapezoidal trajectory as in Figure 7.2, and the mass values for defining the true system's inertia matrix displayed in Table 7.1. Note that the column titled "Mass Variation" refers to the mass quantity added or subtracted to the body's corresponding value in Table 6.1, and the column "Percent Variation" refers to the percentage of the estimated mass value associated with the variation.

Figure 7.7 demonstrates the control torques involved in performing the trajectory following task for an uncertain system. Equivalent control actions are observed when comparing the manipulator joint torques from Figure 7.7 with those implemented on the system with no uncertainties in Figure 7.1. The resulting end-effector linear motion is conveyed in Figure 7.8. Analyzing the the end-effector's linear velocity components in Figure 7.8(a) shows some discrepancies in the trapezoidal trajectories along the y and z axes. These discrepancies are emphasized by Figure 7.8(b) which depicts the progression of the velocity error. The maximum velocity errors in the y and z directions (at the transition from constant acceleration to constant velocity) are shown to be approximately 6 mm/s, which is nearly double the corresponding errors shown for the case of no model uncertainties. Note that the differences observed in the velocity errors along the x-axis remain negligible (around 3 mm/s in both cases). Figure 7.8(c) exhibits the trajectory following in the end-effector's

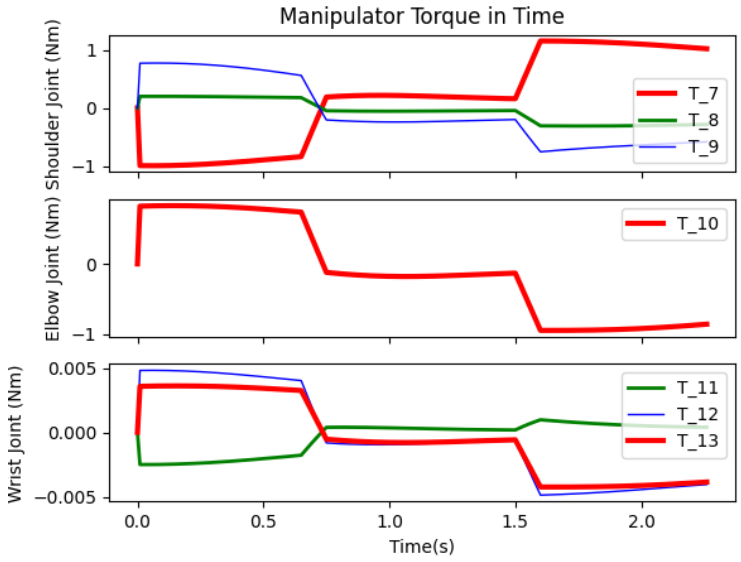
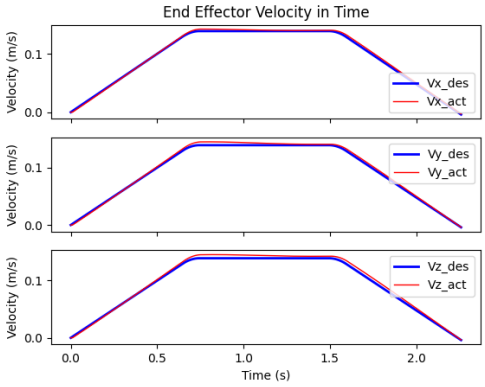


Figure 7.7: Control torques for the linear trapezoidal trajectory applied to a zero momentum system with uncertainties.

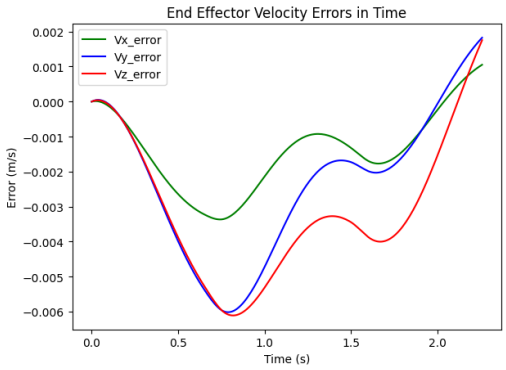
linear position, where maximum tracking errors on the order of millimeters about all axes are shown in Figure 7.8(d) (as was observed in Figure 7.2(d) for the system including no uncertainties). Thus, accurate trajectory following is still achieved for a system containing uncertainties in the space manipulator's inertia matrix.

7.2.2 Space Manipulator System with Non-Zero Momentum

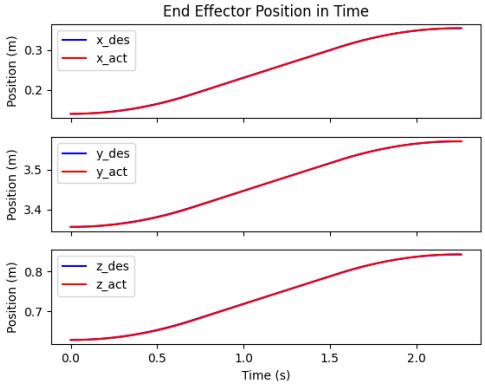
We now move on to analyzing the prowess of the linear output tracking controller when applied to a system with conserved non-zero momentum. In performing this analysis, the same trajectory following scenarios from the previous subsection are repeated here; starting with the trapezoidal trajectory performed in Figure 7.2. An initial angular motion of 0.2 rad/s is now added about the z-axis of the spacecraft's base (i.e., $\dot{\Phi}_{4,ini} = 0.2$ rad/s) to introduce a conserved non-zero momentum to the system. The initial base rotation in turn introduces an equivalent angular rotation to the end-effector, as viewed from the body coordinate frame attached to the base. Consequently, the system's output starts the trajectory following task with a non-zero velocity along the x-axis of the base coordinate frame (initially coincident with spatial coordinate frame). Figure 7.10(a), illustrating the actual and desired trapezoidal velocity trajectories, demonstrates this initial motion in the negative x-direction of the output. Note that the simulation is allowed to run for 1.2 seconds longer



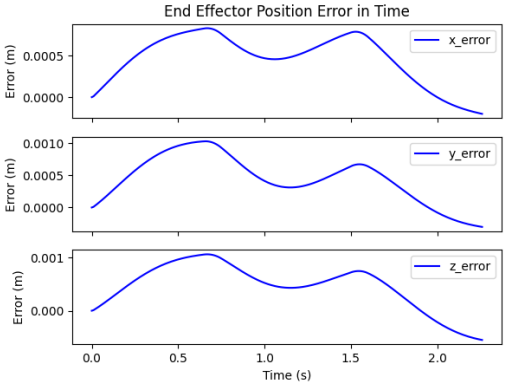
((a)) Desired and actual end-effector velocity



((b)) Velocity error



((c)) Desired and actual end-effector position



((d)) Tracking error

Figure 7.8: Output response to the linear trapezoidal trajectory applied to a zero momentum system with uncertainties.

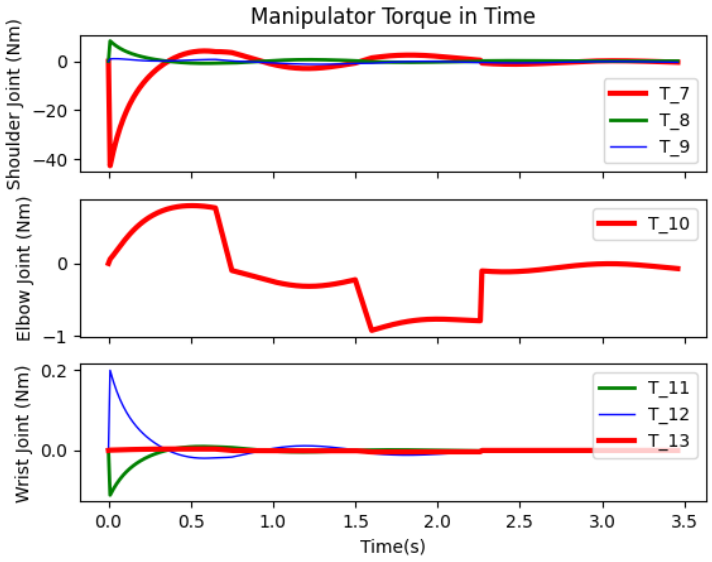
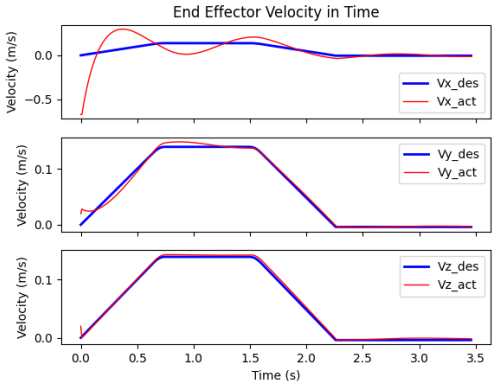


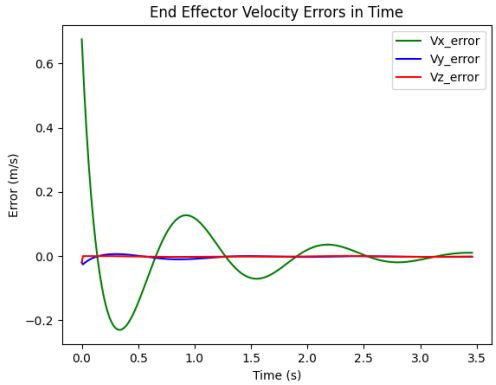
Figure 7.9: Control torques for the linear trapezoidal trajectory applied to a system with non-zero momentum.

than that for the zero momentum case to permit the tracking and velocity errors to settle.

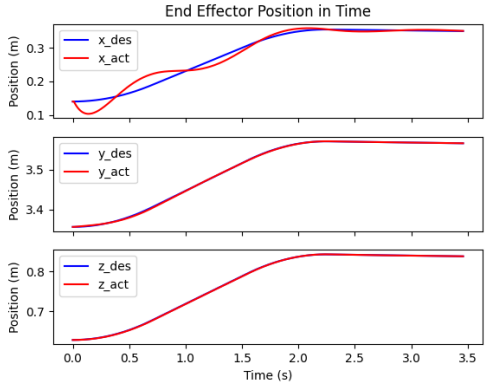
Figure 7.9 shows the manipulator control torques performed by the linear output tracking controller. As expected, to dissipate the initial motion in the x-axis of the output, the greatest amount of torque is initially required about the z-axis of the shoulder joint. Figure 7.9 shows a maximum torque of T_7 = -42 Nm which is significantly larger than the maximum value of 1 Nm required in zero momentum case. Figure 7.10 exhibits the resulting end-effector position and velocity trajectories with their corresponding errors. The effects of the initial velocity error of 0.67 m/s in the x-direction, shown in Figure 7.10(b), are apparent in the output's position along the x-axis, shown in Figure 7.10(c). An oscillatory response is observed in stabilizing the end-effector's position in the x-direction to the desired trajectory. Based on Figure 7.10(d) which displays the output's position error in time, a peak tracking error of 3.8 cm is reached in the x-direction, occurring at around 15 ms into the trajectory. Such oscillations in the x-axis reduce to within 8 mm after 1.9 seconds (while the end-effector is still in motion), and eventually settling at the target position to within 1 mm after 3.4 seconds. Additionally, due to the minimal residual motion in the y and z directions of the output, the output tracking controller is able to follow the desired trajectory along these axes within 3 mm throughout the trajectory following scenario.



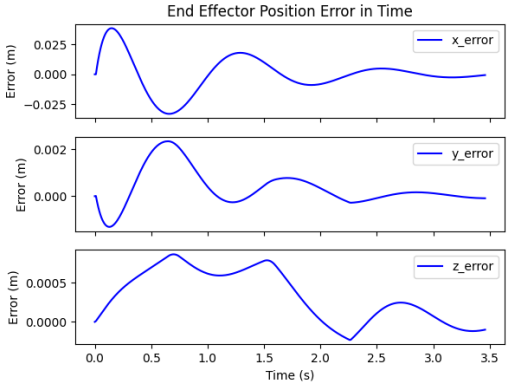
((a)) Desired and actual end-effector velocity



((b)) Velocity error



((c)) Desired and actual end-effector position



((d)) Tracking error

Figure 7.10: Output response to the linear trapezoidal trajectory applied to a system with non-zero momentum.

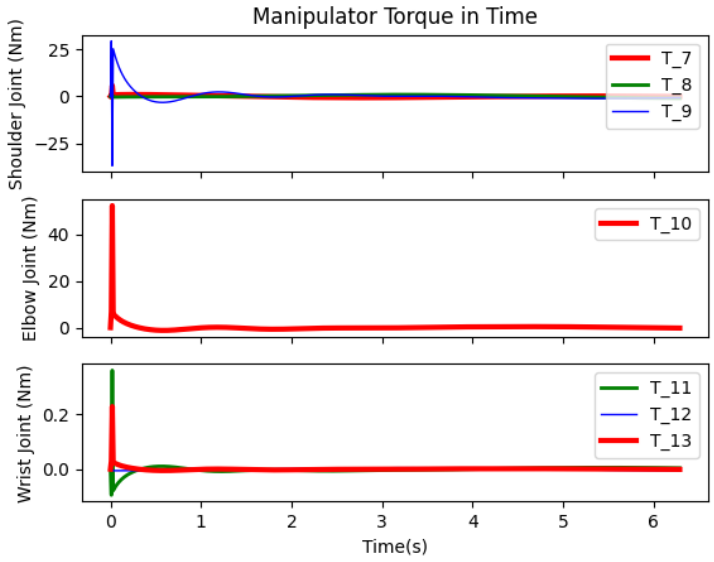
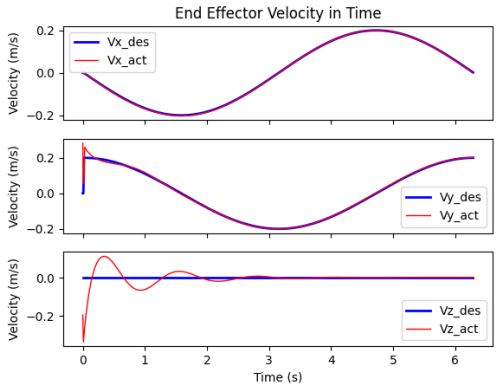


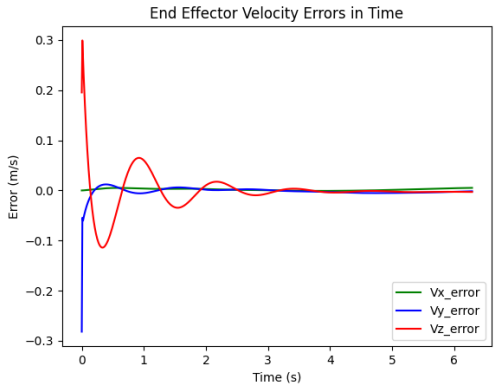
Figure 7.11: Control torques for the linear circular trajectory applied to a system with non-zero momentum.

Moreover, the linear output tracking controller is again tested for the more complex circular trajectory shown in Figure 7.4, however for a system initially in motion. In this case, the spacecraft's base includes an initial angular velocity of -0.1 rad/s about the x-axis of its body coordinate frame (i.e., $\dot{\Phi}_{6,ini}$ = -0.1 rad/s). This in turn induces an initial velocity in the negative z direction of the output (i.e., out of the plane of the desired circular trajectory).

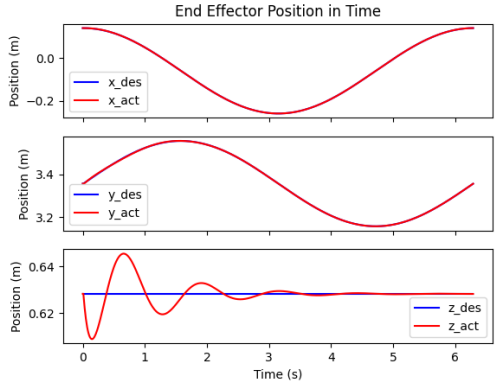
Figure 7.11 demonstrates the commanded torque from the linear output tracking controller to mitigate the end-effector's initial motion in the z direction, and follow the circular trajectory. Comparing the control torques in Figure 7.11 with those shown in Figure 7.3, less actuation in magnitude at the shoulder joint is demonstrated. Figure 7.11 shows the application of T_9 to remain in the negative direction, however with a magnitude of 36 Nm. This additional actuation in the positive direction is required to compensate for the initial end-effector motion occurring in the negative z direction. Figure 7.12 demonstrates the resulting controlled response of the output along the circular trajectory. Figure 7.12(a) shows an initial end-effector velocity in the z-direction of -0.2 m/s which is dissipated after about 1.8 seconds to achieve an associated velocity error of less than 1 cm/s, as conveyed by Figure 7.12(b). The influence of the initial end-effector motion is apparent in Figures



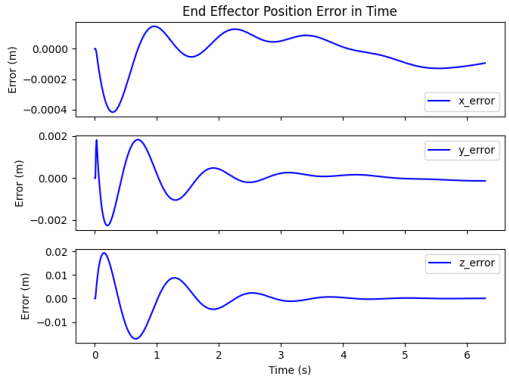
((a)) Desired and actual end-effector velocity



((b)) Velocity error



((c)) Desired and actual end-effector position



((d)) Tracking error

Figure 7.12: Output response to the linear circular trajectory applied to a system with non-zero momentum.

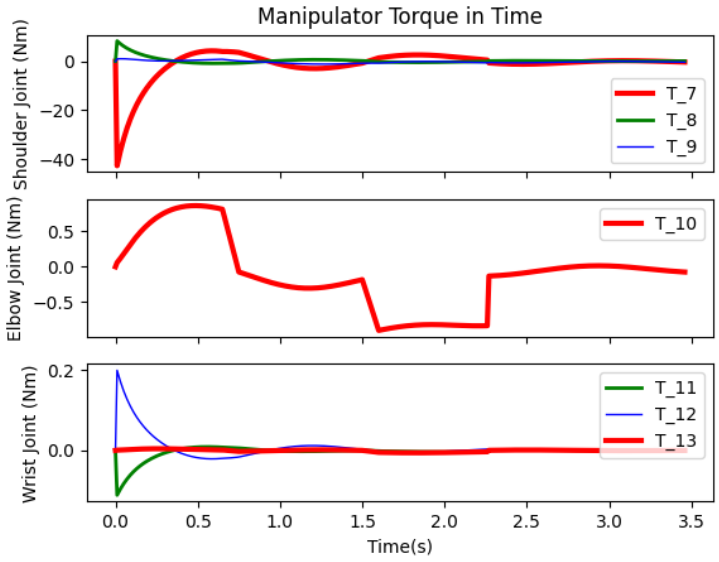
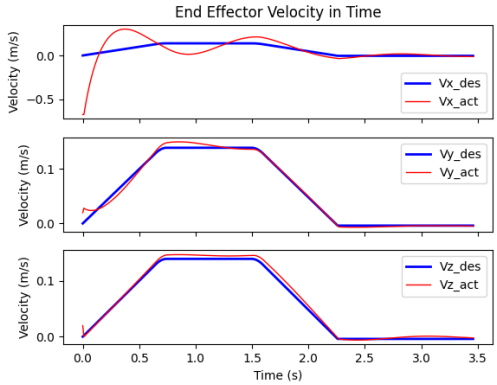


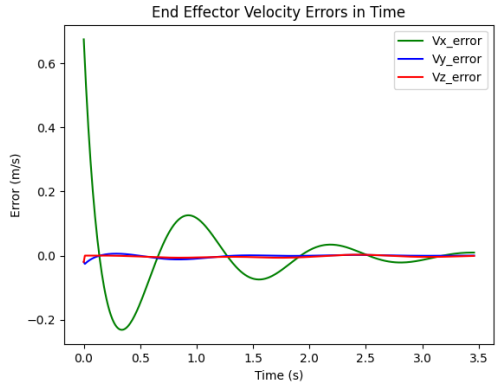
Figure 7.13: Control torques for the linear trapezoidal trajectory applied to an uncertain system with non-zero momentum.

7.12(c) and 7.12(d) which exhibit the end-effector's position and corresponding tracking error in time, respectively. Equivalent tracking responses in the x and y axes between the zero and non-zero momentum cases are shown, however a maximum tracking error of approximately 1.8 cm after 15 ms occurs in the z-direction due to the initial motion in the base. The output's tracking error in this direction is shown to converge to within 5 mm of the desired trajectory after about 2 seconds.

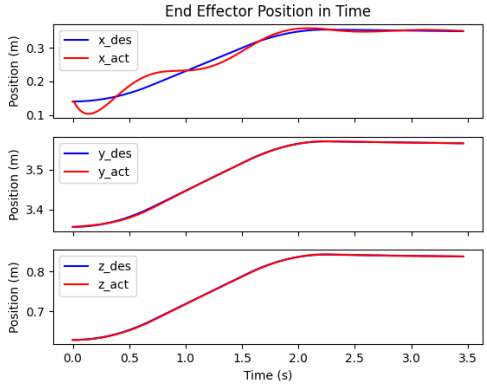
We move on to conducting the final test of the linear output tracking controller; evaluating the controller's robustness to model uncertainties when applied to a system with non-zero momentum. An equivalent testing scenario to that used in the zero momentum test is used in this analysis (implementing the same uncertain values shown in Table 7.1), however with an added initial angular motion of 0.2 rad/s about the z-axis of the base body ($\dot{\Phi}_{4,ini}$ = 0.2 rad/s). Figure 7.13 exhibits the control torques at the manipulator joints during the trajectory following task. Comparing with the control actions included in the same test performed on a system involving no uncertainties (Figure 7.9), no large differences in the applied torques are observed. The resulting velocity and position trajectories, and their associated errors, are shown in Figure 7.14. Comparing with the output's linear response observed in the test involving no uncertainties (Figure 7.10), negligible differences in the



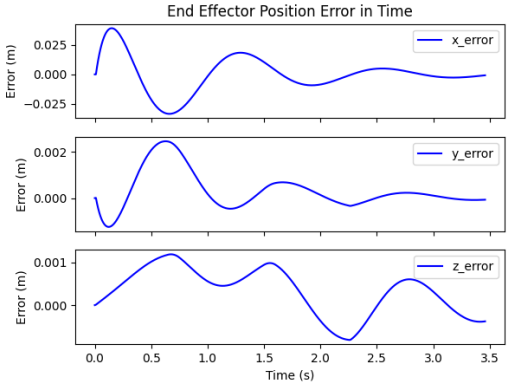
((a)) Desired and actual end-effector velocity



((b)) Velocity error



((c)) Desired and actual end-effector position



((d)) Tracking error

Figure 7.14: Output response to the linear trapezoidal trajectory applied to an uncertain system with non-zero momentum.

behaviours of the tracking and velocity errors along all axes are shown. This similarity in the outputs' response signifies the linear output tracking controller's ability to maintain high trajectory following accuracy for a system with an uncertain inertia matrix.

7.3 Full Pose Output Tracking Controller Results

In this section, the same linear trajectories of the output tested in the previous section are implemented here, however with an additional desired trajectory for the end-effector's orientation to assess of the full pose controller. This section is again broken up by applying the full pose controller to the space manipulator system with conserved zero and non-zero momenta. The following controller gains are used in the simulation of the full pose controller to satisfy the Lyapunov stability condition discussed in Theorem 1: $\boldsymbol{K}_p = \text{diag}\{60,60,60,60,60,60\}$, $\boldsymbol{K}_d = \text{diag}\{15,15,15,15,15,15\}$, $\boldsymbol{K}_i = \text{diag}\{10,10,10,10,10,10\}$. Note that the diagonal values of the PID gains are empirically established by observing the error response resulting from the modified feedforward, feedback PID control law in equations (5.64), (5.66), and (5.68). Moreover, all output tracking testing scenarios presented in this section start with an initial configuration of $\Phi_{9,ini} = 60^\circ$ and $\Phi_{10,ini} = -90^\circ$ (all other generalized coordinates are initialized to zero). This initial configuration corresponds to an initial end-effector pose of:

$$\boldsymbol{g}_{sN}(0) = \begin{bmatrix} 1 & 0 & 0 & 0.14 \\ 0 & 0.866 & 0.5 & 3.25 \\ 0 & -0.5 & 0.866 & 1.37 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{7.4}$$

where the end-effector's position is measured in meters. The figures regarding the trajectories of the base positions associated with each trajectory following task in this section are included in Appendix H, for reference.

7.3.1 Space Manipulator System with Zero Momentum

We start by testing the full pose controller under a trapezoidal trajectory in both the linear and angular components of the end-effector's motion. The same trajectory for the linear

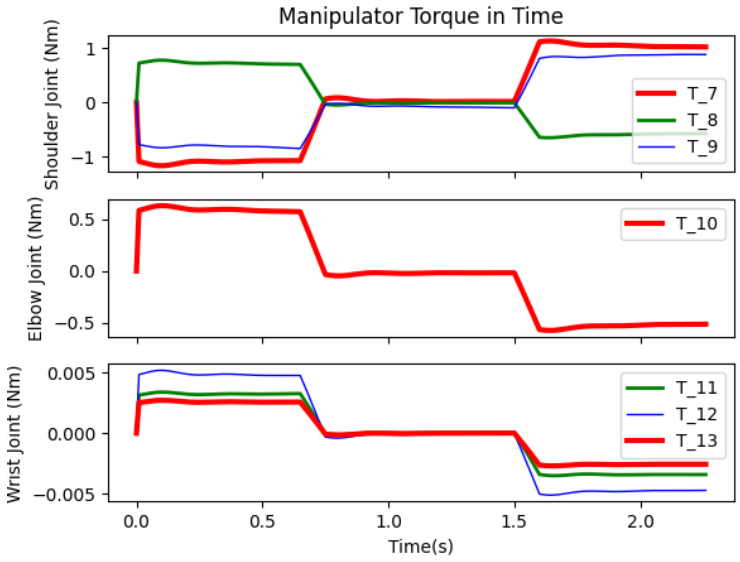
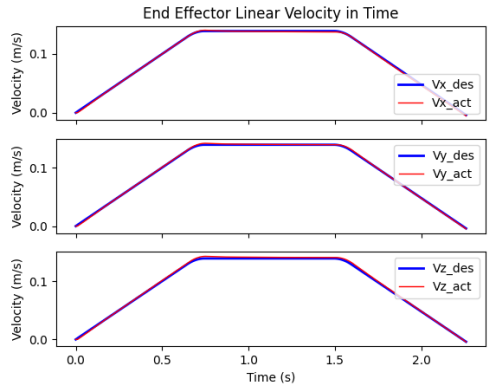


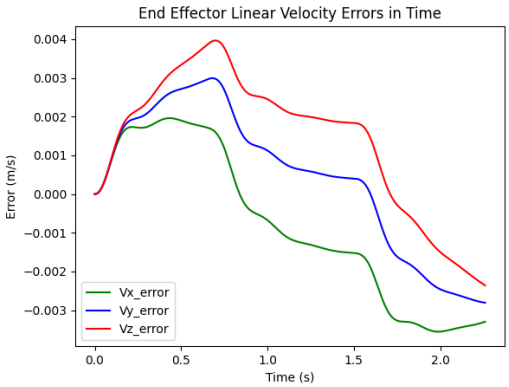
Figure 7.15: Control torques for the full pose trapezoidal trajectory applied to a zero momentum system.

end-effector motion performed in the previous section, and displayed in Figure 7.2, is again implemented here. The trajectory over the output's angular motion involves the same trajectory structure as for the linear motion (i.e., an initial constant acceleration for 0.65 s, followed by a smooth deceleration to reach and travel at a constant velocity for 0.75 s, and ending with a constant deceleration of 0.65 s), however using an angular acceleration magnitude of 0.09 rad/s^2 about all primary axes. The angular trajectory results in angular rotations of approximately 0.1 rad about all axes of the output. Recall that the full pose control law includes desired body velocities and accelerations expressed in the end-effector's body frame, and controls end-effector's pose with respect to the spatial frame.

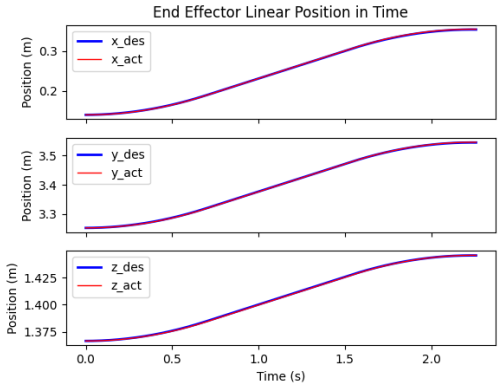
Figure 7.15 shows the control torques performed in the full pose trapezoidal trajectory following task implemented on the space manipulator with zero momentum. Minimal control actions are demonstrated to complete the aforementioned trajectory following task, where all manipulator joint actuations are shown to be within about 1 Nm of torque. Figures 7.16 and 7.17 exhibit the resulting linear and angular components of the end-effector's motion, respectively. Firstly analyzing the output's linear trajectory, the velocities along each axis follow their respective desired trajectories with less than 5 mm/s of error. The



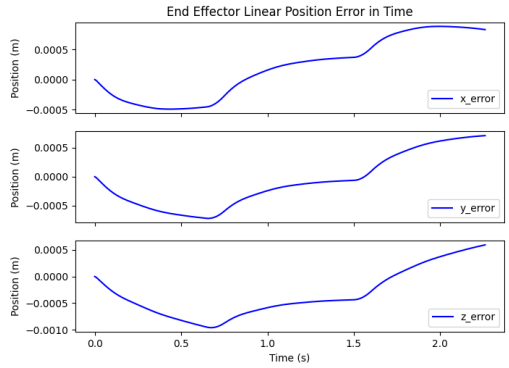
((a)) Desired and actual end-effector linear velocity



((b)) Linear velocity error

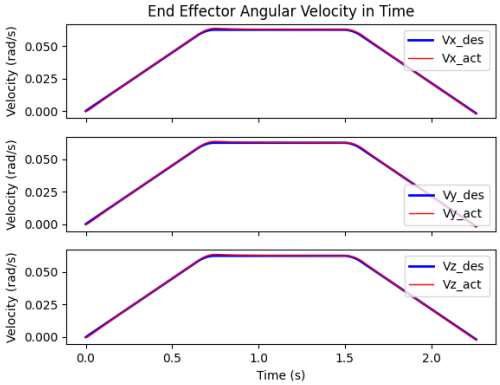


((c)) Desired and actual end-effector linear position

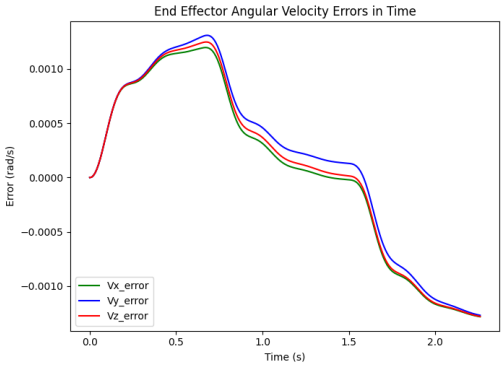


((d)) Linear tracking error

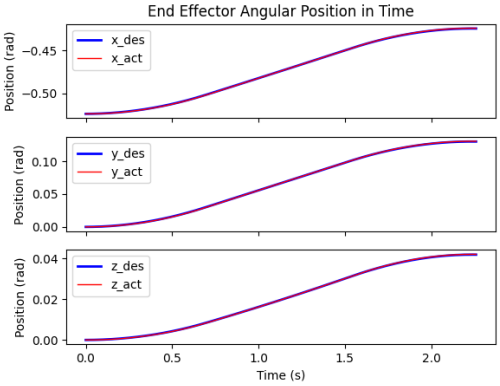
Figure 7.16: Output linear response to the full pose trapezoidal trajectory applied to a zero momentum system.



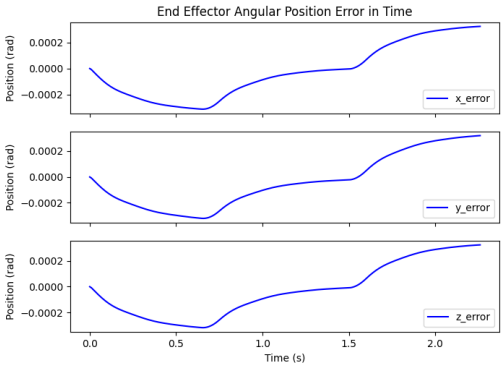
((a)) Desired and actual end-effector angular velocity



((b)) Angular velocity error



((c)) Desired and actual end-effector angular position



((d)) Angular tracking error

Figure 7.17: Output angular response to the full pose trapezoidal trajectory applied to a zero momentum system.

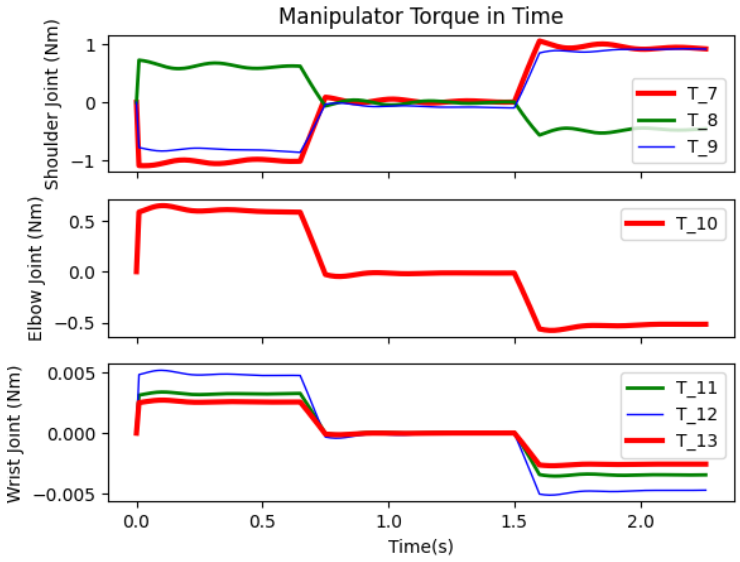
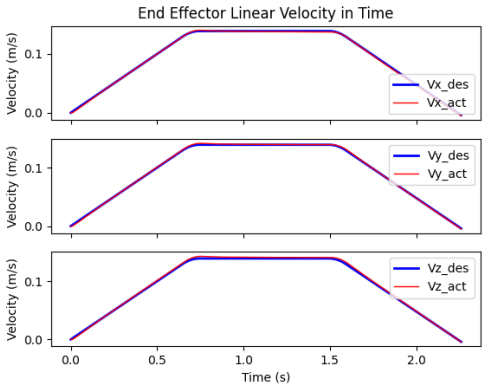


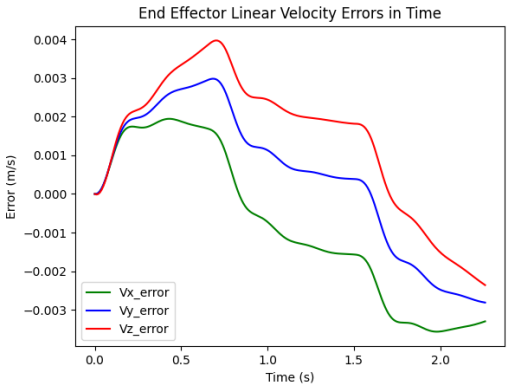
Figure 7.18: Control torques for the full pose trapezoidal trajectory applied to a zero momentum system with uncertainties.

progression of the velocity error in Figure 7.16(b) again displays the largest errors occurring at the transition from constant acceleration to constant velocity, reaching a maximum value of approximately 4 mm/s in the z-direction. The associated tracking error depicted in Figure 7.16(d) demonstrates highly accurate trajectory following capabilities, with the largest deviation shown to be around 1 mm along the z-axis. Moreover, Figure 7.17(b) exhibits the velocity error associated with the end-effector's angular motion, showing similar behaviours about all axes. In this case, the largest velocity error again occurs at the same location as for the linear motion, reaching maximum velocity differences of approximately 0.0012-0.0013 rad/s. Such velocity errors correspond to maximum angular tracking errors well within a single degree of rotation (values of around 0.0003 rad in magnitude), as demonstrated by the progression of the angular position errors in Figure 7.17(d).

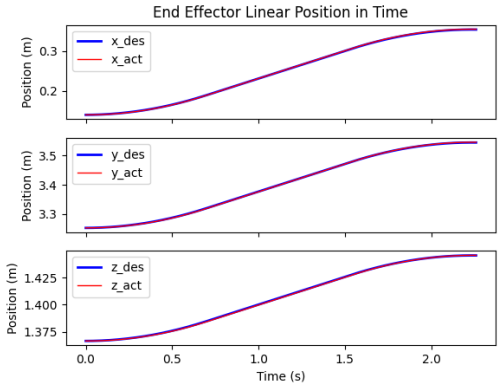
Model uncertainties are again introduced in the system's masses to analyze the controller's robustness to uncertainties in the inertia matrix. For consistency, and to allow for direct comparisons in the results, the uncertain mass values contained in Table 7.1 are again implemented for the case of full pose control. The same linear and angular trajectories as those shown in Figures 7.16 and 7.16 are used in this test such that the influences



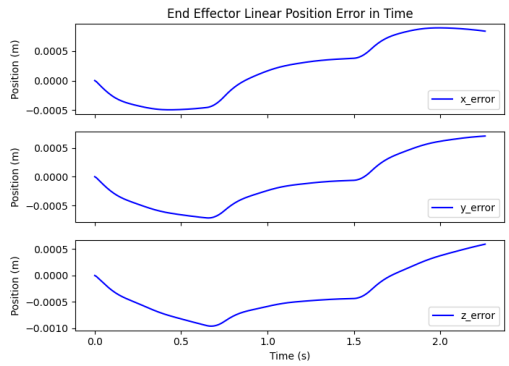
((a)) Desired and actual end-effector linear velocity



((b)) Linear velocity error

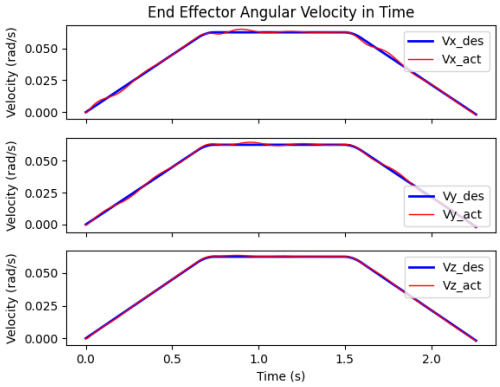


((c)) Desired and actual end-effector linear position

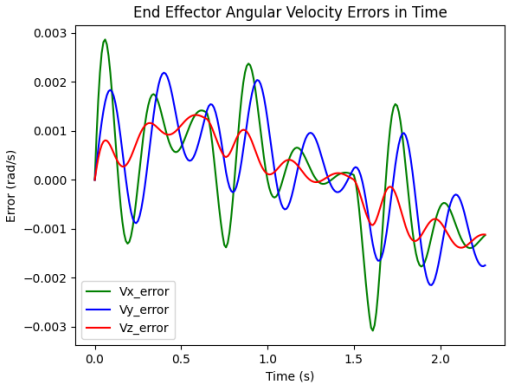


((d)) Linear tracking error

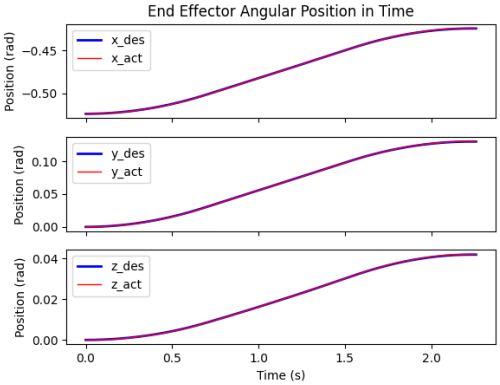
Figure 7.19: Output linear response to the full pose trapezoidal trajectory applied to a zero momentum system with uncertainties.



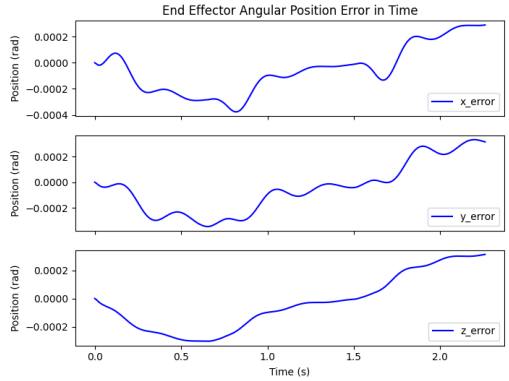
((a)) Desired and actual end-effector angular velocity



((b)) Angular velocity error



((c)) Desired and actual end-effector angular position



((d)) Angular tracking error

Figure 7.20: Output angular response to the full pose trapezoidal trajectory applied to a zero momentum system with uncertainties.

from model uncertainties can be observed. The resulting control actions from the trajectory following task with model uncertainty are shown in Figure 7.18. The manipulator joint actuations are again contained within 1 Nm of torque, however the control actions slightly waver from the values shown in Figure 7.15. Such wavering is evidently a result of the system's estimated dynamic behaviour used in the dynamic reduction and feedback linearization processes.

The trajectories of the linear and angular positions of the end-effector are demonstrated in Figures 7.19 and 7.20, respectively. Regarding the tracking performance over the linear end-effector motion, a maximum velocity error of 4 mm/s in the z-direction is exhibited in Figure 7.19(b), and all linear positions are able to follow the desired trajectory within 1 mm, as demonstrated by Figure 7.19(d) (an equivalent response in the linear motion as shown for the case of no model uncertainties). Accurate trajectory following is similarly shown for the end-effector's orientation, where the angular positions shown in Figure 7.20(d) demonstrate tracking abilities within 0.0003 rad of the desired angular trajectory. Comparing with the angular trajectory results demonstrated for the system including no model uncertainties in Figure 7.17, the model uncertainty is shown to induce oscillations in the behaviour of the angular velocity error. These oscillations are of greatest amplitude about the x-axis, reaching a maximum angular velocity error of 0.003 rad/s. Overall, highly accurate output trajectory followings are achieved in both the linear and angular motions, despite the output tracking control law involving model uncertainties.

7.3.2 Space Manipulator System with Non-Zero Momentum

In this subsection, the tests involving the trapezoidal trajectories over both the linear and angular end-effector motions applied to systems with and without model uncertainties are repeated, however with the spacecraft containing an initial motion of $\dot{\Phi}_4 = 0.2$ rad/s. We begin with Figure 7.21 which demonstrates the control actions involved in the full pose trajectory following for a system with non-zero momentum and no model uncertainties. As in the linear control case, a large initial torque of about 70 Nm in magnitude about the z-axis of the shoulder joint is required to dissipate the end-effector's initial motion about the x-axis, with respect to the spatial frame. The actuation about the elbow joint remains within 1 Nm of torque as this joint induces no motion about the x-axis of the output's spatial position. Figure 7.22 depicts the corresponding linear velocity and position responses of the

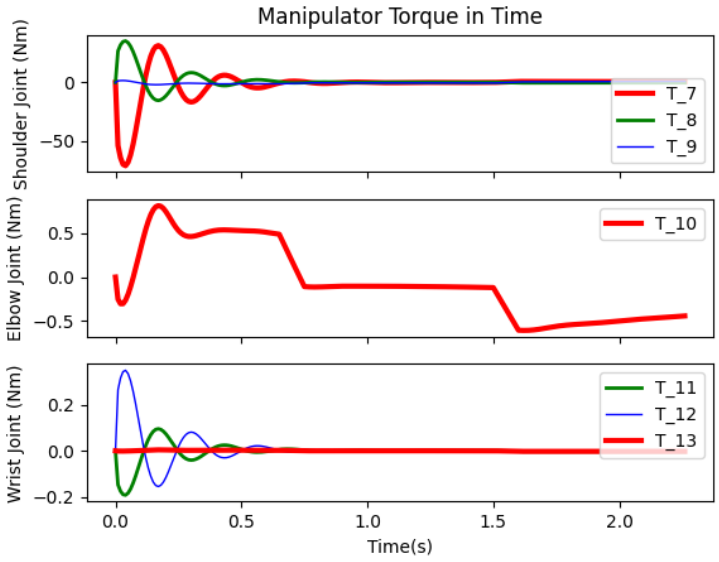
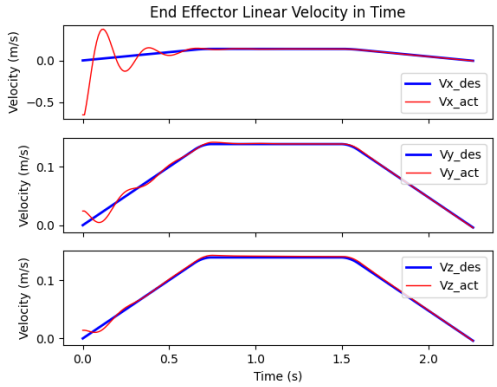


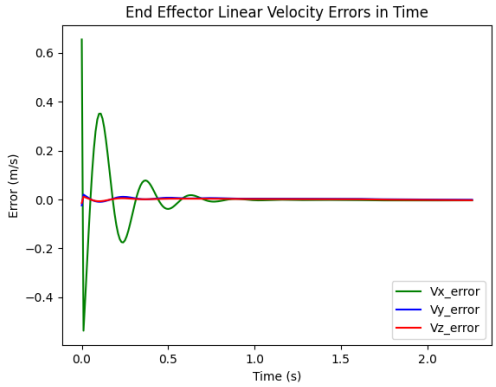
Figure 7.21: Control torques for the full pose trapezoidal trajectory applied to a system with non-zero momentum.

end-effector. Recall that the velocities shown in Figure 7.23(a) are with respect to the end-effector's body coordinate frame, and the linear positions in Figure 7.23(a) express the end-effector's position with respect to the spatial frame. The full pose output tracking controller is shown to effectively diminish the initial velocity in the x-axis due to the base's rotational velocity. Based on the velocity error displayed in Figure 7.22(b), a maximum overshoot of magnitude of 0.53 m/s along the x-axis occurs, eventually settling after around 0.56 s to within 1 cm/s. The initial oscillation about the desired trajectory in the linear position along the x-axis, shown in Figures 7.22(c) and 7.22(d), reaches a maximum undershoot of 1.68 cm which reduces to within 1 cm after approximately 0.22 s.

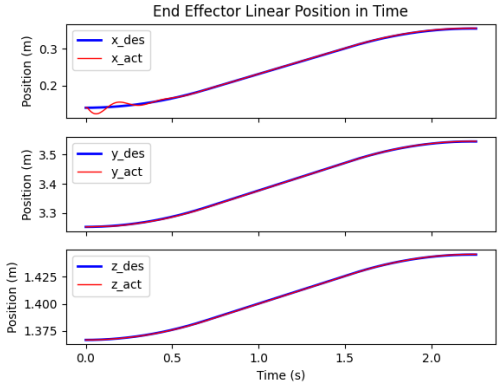
Moreover, the end-effector's angular trajectory is exhibited in Figure 7.23, where Figure 7.23(a) demonstrates the output's rotational velocity with respect to the body frame. Initial motions of -0.1 rad/s and 0.14 rad/s about the y and z axes are induced by the initial motion of the base. In dissipating such rotational motions, a maximum overshoot of 0.054 rad/s about the y-axis and a maximum undershoot of 0.092 rad/s about the z-axis are shown in Figure 7.23(b). Such velocity errors are decreased to within 0.02 rad/s after around 0.37 s. Note that the base's initial angular rotation about the z-axis in turn induces an equivalent angular rotation of 0.2 rad/s in the end-effector's angular velocity with respect to the spatial frame. This angular motion in the output is apparent from the oscillations



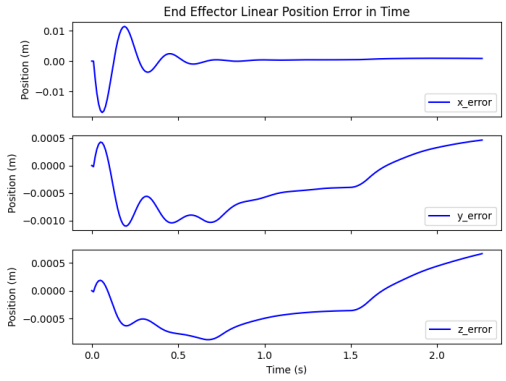
((a)) Desired and actual end-effector linear velocity



((b)) Linear velocity error

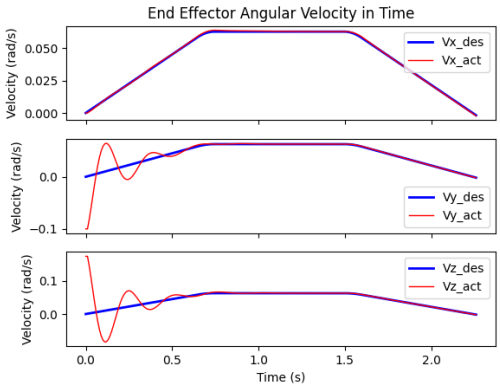


((c)) Desired and actual end-effector linear position

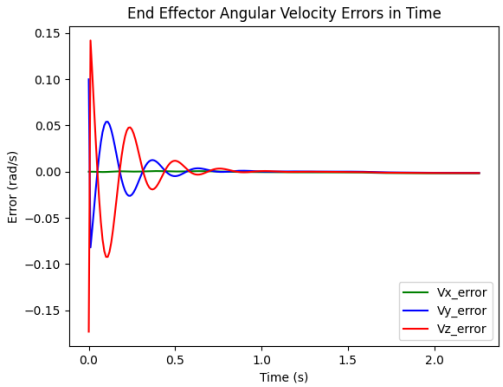


((d)) Linear tracking error

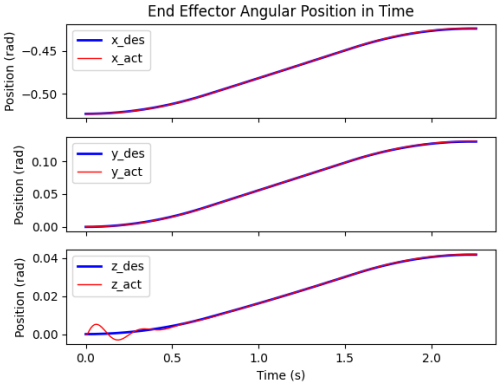
Figure 7.22: Output linear response to the full pose trapezoidal trajectory applied to a system with non-zero momentum.



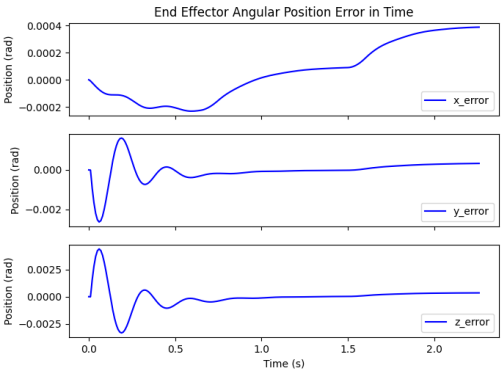
((a)) Desired and actual end-effector angular velocity



((b)) Angular velocity error



((c)) Desired and actual end-effector angular position



((d)) Angular tracking error

Figure 7.23: Output angular response to the full pose trapezoidal trajectory applied to an system with non-zero momentum.

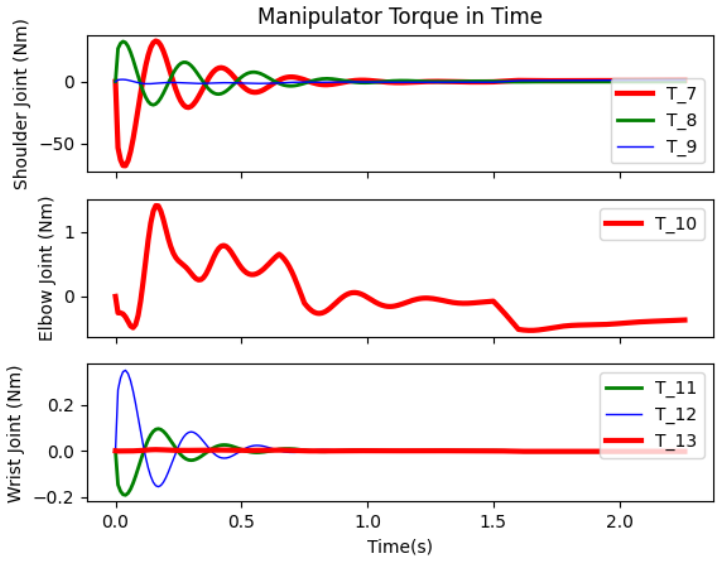
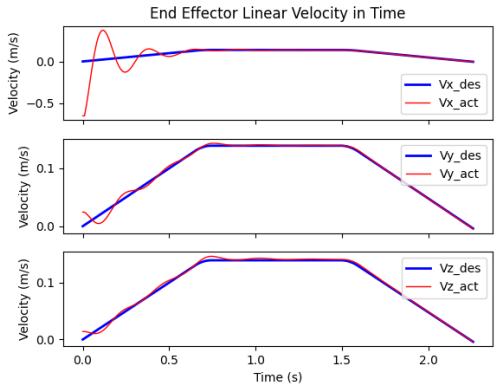


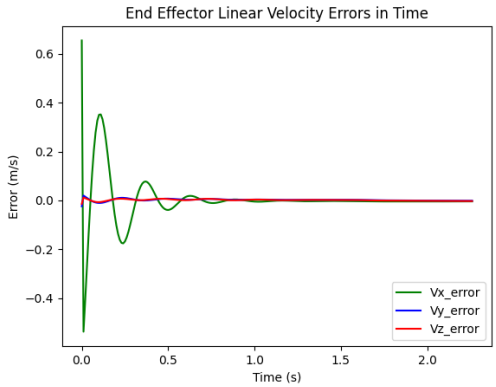
Figure 7.24: Control torques for the full pose trapezoidal trajectory applied to an uncertain system with non-zero momentum.

which occur in the z-angular position of the end-effector, as shown in Figure 7.23(c). The associated tracking error in Figure 7.23(d) demonstrates a maximum error of 0.0044 rad about this axis. Thus, the full pose output tracking controller is able to achieve highly accurate trajectory followings for a system with non-zero momentum.

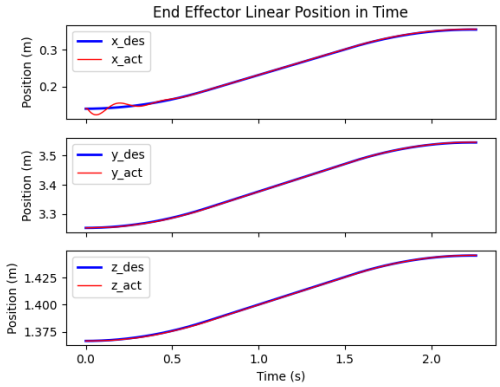
Finally, Figure 7.24 demonstrates the manipulator joint torques for controlling the space manipulator system with non-zero momentum and model uncertainties. Similar torque values at the shoulder and wrist joints to those performed in Figure 7.21 are displayed, however approximately 1 Nm of additional torque is actuated at the elbow joint in the case of an uncertain system. Regarding the end-effector's response in the linear motion, shown in Figure 7.25, virtually equivalent behaviours in both the velocity and position are exhibited when compared with the response in the case of no model uncertainty (Figure 7.22). Consequently, the full pose output tracking controller remains robust to the uncertainties in the system's inertia matrix when applied to a system with non-zero momentum. Figure 7.26 exhibits the resulting angular velocity and position of the end-effector where discrepancies with respect to the response shown in Figure 7.23 begin to appear. Increased oscillatory



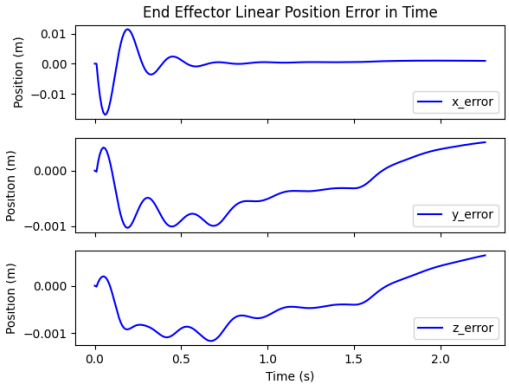
((a)) Desired and actual end-effector linear velocity



((b)) Linear velocity error

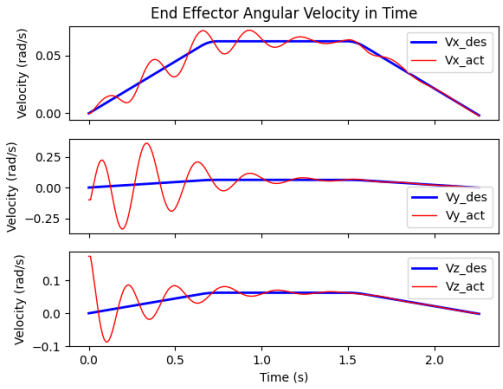


((c)) Desired and actual end-effector linear position

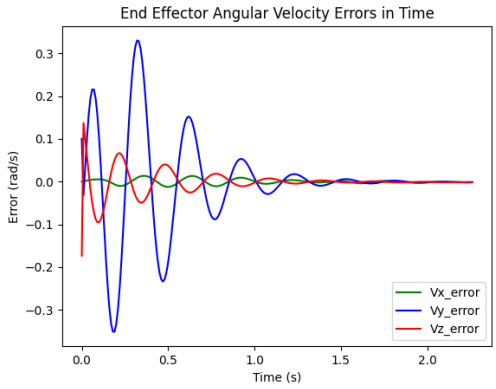


((d)) Linear tracking error

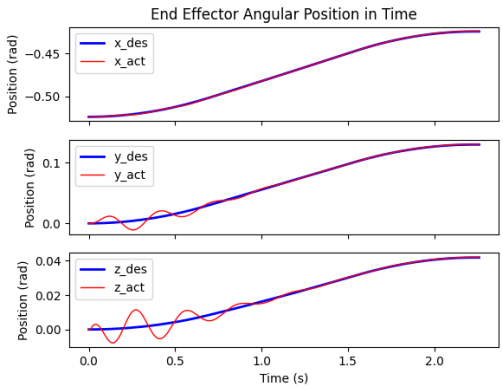
Figure 7.25: Output linear response to the full pose trapezoidal trajectory applied to an uncertain non-zero momentum.



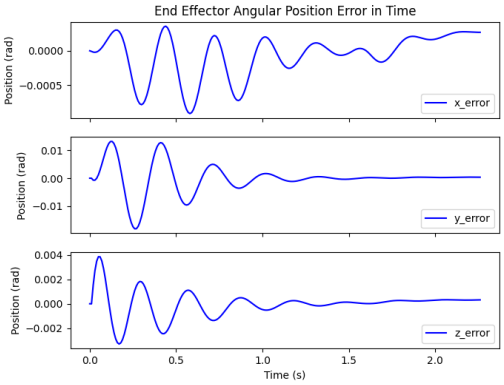
((a)) Desired and actual end-effector angular velocity



((b)) Angular velocity error



((c)) Desired and actual end-effector angular position



((d)) Angular tracking error

Figure 7.26: Output angular response to the full pose trapezoidal trajectory applied to an uncertain system with non-zero momentum.

behaviours about all axes of the angular velocity and the y and z axes of the angular position are illustrated in Figure 7.26(a) and Figure 7.26(c), respectively. Hence, the presence of model uncertainties is shown to negatively influence the response in the output's angular motion. However, the full pose output tracking controller remains robust to such uncertainties as the controller is capable of dissipating the initial motion in the end-effector to achieve an angular tracking error (about all axes) within 0.001 rad after 0.55 s.

Chapter 8

Conclusion and Future Work

In this thesis, we developed two output tracking controllers to control both the linear position and pose of a free-floating space manipulator's end-effector with conserved non-zero momentum. The developed control laws complement the industrialized OOS mission architecture proposed in chapter 3, to address the requirement concerning service precision during proximity operations. The proposed architecture additionally addressed the optimal minimization of mission duration and fuel consumption by carefully suggesting an appropriate parking orbit for the fleet of servicers in the SSO region of LEO, and by formulating the multi-objective optimization problems involved in transfer trajectory planning and multi-target mission scheduling. Using geometric mechanics, the kinematics and dynamics models of a single-arm free-floating space manipulator servicer were derived. The product of expnentials formula was formulated for space manipulators considered as multi-body systems with multi-DOF joints to form the forward and differential kinematics mappings. For implementation in the product of exponentials, multi-DOF joints were newly modelled as the amalgamated exponential mappings of individual screw motions along a single axis. The dynamics of a free-floating space manipulator was derived using the Euler-Lagrange equation, and subsequently decoupled into the base and manipulator motions. Using the conserved linear and angular momentum as affine nonholonomic constraints, the decoupled equations of motion were reduced manipulator joints space.

Output tracking controllers were developed to address the two previously mentioned control problems presented in this thesis. Both workspace control strategies were predicated on feedback linearization in the manipulator joint space such that the end-effector could be controlled using linear control approaches. In the cases of controlling the linear position and pose of the output, feedback linearization was employed on $\mathbb{R}^3$ and $\mathfrak{se}(3)$,

respectively, to eliminate the influences from the system's reduced dynamics on the end-effector motion. A classical PID controller was designed to force the output's linear motion to behave as a critically damped system. The proposed modified feedforward, feedback PID controller was structured to stabilized the end-effector's pose along a feasible desired trajectory, in the sense of Lyapunov. The full pose output tracking control law was shown to act on the gradient of the coordinate free error function applied to the pose group error, and the associated velocity error in the body frame. Singularity accommodation using the SR-inverse technique was implemented in the feedback linearization laws to mitigate the risk of commanding unachievable actuations at the manipulator joints.

The proposed output tracking control laws were tested on a developed free-floating space manipulator simulation platform in Python. The details of the simulation structure were provided, and the considerations for computational efficiency, particularly in the computation of the Coriolis matrix, were addressed. The modelling of an equivalent space manipulator system in Simscape for validation of the Python simulation was described, and the testing procedures for confirming the accuracy of the kinematics and dynamics in Python were discussed.

The Python simulator was subsequently used to observe the controlled end-effector motions resulting from the linear and full pose output tracking controllers. In the linear case, a trapezoidal trajectory was provided to move the end-effector approximately 36 cm in 2.25 s. The resulting controlled response of the output demonstrated highly accurate followings of within 1 mm along all axes of the desired trajectory. When model uncertainties were introduced in the same trajectory following task via discrepancies in the system's mass values, the linear controller was able to maintain the tracking errors to within approximately 1 mm. This exhibited the developed linear controller's robustness to uncertainties in the system's inertia matrix. Placing the controlled space manipulator in a more complex trajectory following scenario demonstrated the controller's ability to follow a circular path of radius 0.2 m, restricted in a single plane of motion. The associated control actions were maintained within an acceptable torque range (around 50-60 Nm at the shoulder and elbow joints), resulting in a maximum initial tracking error of 2 mm occurring in the y-direction, subsequently reduced to within 1 mm after 1.5 s. Adding an initial motion to the spacecraft's base introduced non-zero momentum to the system and initial unwanted motions in the output. Regarding the trapezoidal trajectory, a rotational velocity of $\dot{\Phi}_4 = 0.2$ rad/s was implemented, causing an initial velocity of 0.67 m/s in x-direction of the output, with respect to the spatial frame. The linear output tracking controller was shown to dissipate

such motion in the output to within a velocity error of 1 cm/s after 1.8 s, and achieve associated tracking errors within 8 mm along all axes. Performing the same test on the uncertain space manipulator system demonstrated an essentially equivalent response in the controlled output which emphasized the linear controller's robustness to inertia matrix uncertainties. Successful mitigation of an initial out-of-plane motion of 0.2 m/s in the circular trajectory was additionally shown, where strictly in-plane motion (within 5 mm) was regained after about 2 s. Moreover, when the space manipulator system was forced into a region of low manipulability (i.e., below the manipulability threshold), the damping factor in the SR-inverse was shown to increase, resulting in the control actions to be held within an acceptable boundary, at the expense of tracking accuracy.

In the analysis of the full pose output tracking controller, the same trapezoidal trajectory as in the linear case was implemented for the linear motion of the output, and an additional trapezoidal trajectory for end-effector's angular motion generated a net rotation of 0.1 rad about all axes. The modified feedforward, feedback PID controller was shown to limit the output's pose to within 1 mm and 0.0003 rad of error in the linear and angular motions, respectively. When applied to the same uncertain space manipulator system as in the linear case, the full pose controller displayed robustness by maintaining a linear tracking error of within 1 mm. An increased oscillatory behaviour was demonstrated in the end-effector's angular velocity, however the angular tracking error was still held to within 0.0003 rad about all axes. An initial base rotation of $\dot{\Phi}_4 = 0.2$ rad/s was again introduced and the same trapezoidal trajectory was repeated. In dissipating the initial motion in the output, the control actions were maintained within an acceptable region (a maximum torque of around 70 Nm about the z-axis of the shoulder joint), and the full pose controller was shown to reduce the initial linear velocity along the x-direction to within 1 cm/s after 0.56 s (after reaching a maximum overshoot of magnitude of 0.53 m/s). Consequently, the linear tracking error was diminished to within 1 cm after around 0.22 s from an overshoot value of 1.68 cm. Moreover, the end-effector's initial angular velocity of 0.2 rad/s in the z-direction, with respect to the spatial frame, was successfully dissipated to yield angular tracking errors of within 0.0044 rad about all axes. When introducing the uncertain space manipulator model, the output's linear behaviour remained consistent with response involving a certain system model. In the angular motion, the model uncertainties induced oscillations about the y and z axes of the velocity, however the full pose controller demonstrated its robustness by reducing all angular tracking errors to within 0.001 rad after 0.55 s.

Future directions for this research primarily involve the incorporation of machine learning techniques to further improve the GNC performance of space manipulator systems. Reinforcement learning is a particularly attractive and promising method for tasks concerning the non-trivial trajectory planning of free-floating space manipulators, as this is a model-free learning process. In-depth research regarding the structure of the learning operation (i.e., the definition of states, actions, and reward function) is necessary for obtaining a desired trajectory planning model, following the learning process's convergence. Reinforcement learning approaches possess large potential for extending the trajectory planning capabilities to include capture of moving targets, obstacle avoidance, and disturbance mitigation. Moreover, reinforcement learning may also be implemented as a system identification method to improve the accuracy of a space manipulator's inertia matrix. To complete such learning tasks, further research on singularity avoidance techniques and the methods of accurate control near a singular configuration are necessary. This involves examining the internal dynamics of a free-floating space manipulator system and performing a stability analysis on the system's states. Additionally, the developed simulation platform in Python is currently being improved with the implementation of a 3-dimensional rendering of the free-floating space manipulator and a target satellite.

Appendix A

Mathematical Preliminaries

A.1 Rigid Body Motion in $\mathbb{R}^3$

In this section, the motion of a single rigid body in the Euclidean 3-dimensional space $\mathbb{R}^3$ is reviewed. The motion of a rigid body is described by a transformation matrix defining the relative position and orientation between two coordinate frames at a single instance in time. This thesis considers rigid body transformations, signifying that such transformations establish a mapping between right-handed, orthonormal coordinate frames. Generally speaking, all rigid body motions involve some combination of a rotation and a translation. In the following sections the geometric realization of pure rotational and translational motions are defined based on Lie group theory, and later combined to specify a complete rigid body motion. Finally, the time derivative of a relative configuration in $\mathrm{SE}(3)$ is analyzed for interpreting the angular and linear velocity of a rigid body in both the spatial and body coordinate frames.

A.1.1 Rotational and Translational Motion

Let us define a right-handed coordinate frame attached to a body in motion, as well as a coordinate frame fixed in space. These frames are referred to as the body-coordinate frame denoted by B and the spatial coordinate frame denoted by S. Analyzing the pure rotational motion of a rigid body in $\mathbb{R}^3$, the moving body induces a continuously changing relative orientation between the body and spatial coordinate frames, however the origin of the body coordinate frame remains fixed with respect to the spatial frame in time.

The relative orientation between the two coordinate frames defines the orientation of the rigid body at a given instance in time, t. In this work, rotation matrices $\boldsymbol{R} \in \mathbb{R}^{3\times 3}$ are used

as the representation of a body's orientation in $\mathbb{R}^3$. The group of all 3×3 rotation matrices is a 3-dimensional space referred to as the Special Orthogonal group $\mathrm{SO}(3)$. Members of the $\mathrm{SO}(3)$ satisfy the following properties: $\boldsymbol{R}\boldsymbol{R}^T = \mathbf{1}_{3\times 3}$ and $\det \boldsymbol{R} = 1$. That is,

$$\mathrm{SO}(3) := \{\boldsymbol{R} \in \mathbb{R}^{3\times 3} : \boldsymbol{R}\boldsymbol{R}^T = \mathbf{1}_{3\times 3},\ \det(\boldsymbol{R}) = 1\}, \tag{A.1}$$

where $\det(\boldsymbol{R})$ refers to the determinant of the matrix $\boldsymbol{R}$. The orientation of any rigid body can be described by a unique member of the rotation group $\mathrm{SO}(3)$, and the trajectory of a rotating body can be expressed as the curve $\boldsymbol{R}(t) \in \mathrm{SO}(3)$ indicating change in the body's orientation in time. The rotation of a single rigid body can be described as the rotation about a single axis $\boldsymbol{w} \in \mathbb{S}^2 \subset \mathbb{R}^3$ on the unit 2-sphere by a rotation angle $\Phi \in \mathbb{R}$. The corresponding net rotation $\boldsymbol{R}$ associated with the coordinates of rotation ($\boldsymbol{w}$, Φ) can be described using the exponential mapping of the $\mathrm{SO}(3)$,

$$\boldsymbol{R}(\boldsymbol{w}, \Phi) = e^{\tilde{\boldsymbol{w}}\Phi} \in \mathrm{SO}(3). \tag{A.2}$$

The tilde operator indicates the vector space isomorphism between the Lie algebra of the $\mathrm{SO}(3)$ Lie group, denoted by $\mathfrak{so}(3)$, and $\mathbb{R}^3$. That is, for a vector $\boldsymbol{u} = [u_1\ u_2\ u_3]^T \in \mathbb{R}^3$, the tilde operator performs the mapping to the following 3×3 skew-symmetric matrix,

$$\tilde{\boldsymbol{u}} = \begin{bmatrix} 0 & -u_3 & u_2 \\ u_3 & 0 & -u_1 \\ -u_2 & u_1 & 0 \end{bmatrix} \in \mathfrak{so}(3). \tag{A.3}$$

Note that the scalar multiple of any member of $\mathfrak{so}(3)$ is also an element of $\mathfrak{so}(3)$. For a unit skew-symmetric matrix $\tilde{\boldsymbol{w}}$ with magnitude $||\boldsymbol{w}|| = 1$, Rodrigues' formula provides a closed form solution for the exponential mapping between the $\mathrm{SO}(3)$ Lie group and its Lie algebra $\mathfrak{so}(3)$ in equation (A.2). Expanding on Rodrigues' formula for the general case where $||\boldsymbol{w}|| \neq 1$, presents the following forms for the mapping $\exp(\tilde{\boldsymbol{w}}\Phi)$ [45],

$$e^{\tilde{\boldsymbol{w}}\Phi} = \begin{cases} \boldsymbol{I}_{3\times 3} + \tilde{\boldsymbol{w}}\sin\Phi + \tilde{\boldsymbol{w}}^2(1-\cos\Phi), & ||\boldsymbol{w}|| = 1 \\ \boldsymbol{I}_{3\times 3} + \frac{\tilde{\boldsymbol{w}}}{||\boldsymbol{w}||}\sin(||\boldsymbol{w}||\Phi) + \frac{\tilde{\boldsymbol{w}}^2}{||\boldsymbol{w}||^2}\left(1-\cos(||\boldsymbol{w}||\Phi)\right). & ||\boldsymbol{w}|| \neq 1 \end{cases} \tag{A.4}$$

Progressing to the more trivial realization of purely translational rigid body motion, let $\boldsymbol{p} \in \mathbb{R}^3$ denote the position of a point on the rigid body with respect to the spatial coordinate frame. The spatial change in the position $\boldsymbol{p}$ in time defines the trajectory of the rigid body under pure translational motion. This trajectory is represented by the curve $\boldsymbol{p}(t) \in \mathbb{R}^3$.

A.1.2 Homogeneous Transformations

The full motion of a rigid body, involving both rotational and translational components, is expressed by combining the individual representations of the two motions studied in the previous section. Let $\boldsymbol{g} = (\boldsymbol{p}, \boldsymbol{R})$ denote such a combination for a translation $\boldsymbol{p}$ and a rotation $\boldsymbol{R}$ that is called a rigid body transformation. The Special Euclidean Lie group SE(3) is the set of all such transformations, defining the six dimensional configuration space of a rigid body in $\mathbb{R}^3$,

$$\mathrm{SE}(3) := \{\boldsymbol{g} = (\boldsymbol{p},\, \boldsymbol{R}) \,:\, \boldsymbol{p} \in \mathbb{R}^3,\, \boldsymbol{R} \in \mathrm{SO}(3)\}. \tag{A.5}$$

The transformations $\boldsymbol{g} \in \mathrm{SE}(3)$ not only identify the position and orientation (pose) of a rigid body, but can additionally be used to transform the coordinate expression of a point $\boldsymbol{q} \in \mathbb{R}^3$ from one reference frame to another. Let $\boldsymbol{p}_{sb} \in \mathbb{R}^3$ and $\boldsymbol{R}_{sb} \in \mathrm{SO}(3)$ denote the relative position and orientation of a rigid body with respect to the spatial coordinate frame, respectively. Accordingly, the pose of the rigid body with respect to the spatial frame is the transformation $\boldsymbol{g}_{sb} = (\boldsymbol{p}_{sb}, \boldsymbol{R}_{sb}) \in \mathrm{SE}(3)$. If the point $\boldsymbol{q}$ has known coordinates with respect to the body coordinate frame $\boldsymbol{q}_b$, the corresponding realization of $\boldsymbol{q}$ in the spatial coordinate frame $\boldsymbol{q}_s$ is obtained through the affine transformation

$$\boldsymbol{q}_s = \boldsymbol{g}_{sb}(\boldsymbol{q}_b) = \boldsymbol{p}_{sb} + \boldsymbol{R}_{sb}\boldsymbol{q}_b. \tag{A.6}$$

Rigid transformations can be employed in a straightforward manner, using the notion of homogeneous coordinates for points and vectors in 3-dimensional space. The homogeneous representation considers points and vectors in $\mathbb{R}^4$ on which 4×4 linear transformation matrices can be applied. The homogeneous representation of a point

$\boldsymbol{q} = [q_1,\ q_2,\ q_3]^T \in \mathbb{R}^3$ is the vector $\bar{\boldsymbol{q}} \in \mathbb{R}^4$ containing the coordinates of $\boldsymbol{q}$ and a scaling factor equal to 1 as the last element,

$$\bar{\boldsymbol{q}} = \begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ 1 \end{bmatrix}. \tag{A.7}$$

Consequently, a vector $\boldsymbol{v} \in \mathbb{R}^3$ which is the difference of two points has the following homogeneous coordinates,

$$\bar{\boldsymbol{v}} = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ 0 \end{bmatrix}. \tag{A.8}$$

Provided the linear and angular attributes of a rigid body transformation $\boldsymbol{p}$ and $\boldsymbol{R}$, the associated 4×4 homogeneous transformation is of the following form,

$$\bar{\boldsymbol{g}} = \begin{bmatrix} \boldsymbol{R} & \boldsymbol{p} \\ \boldsymbol{0}_{1\times 3} & 1 \end{bmatrix} \in \mathrm{SE}(3). \tag{A.9}$$

Therefore, the affine transformation of $\boldsymbol{q}_b$ into its spatial representation can be fully described as a linear transformation using the homogeneous representation of the point $\boldsymbol{q} \in \mathbb{R}^3$ and the homogeneous transformation $\bar{\boldsymbol{g}}_{sb} \in \mathrm{SE}(3)$,

$$\bar{\boldsymbol{q}}_s = \bar{\boldsymbol{g}}_{sb}\bar{\boldsymbol{q}}_b = \begin{bmatrix} \boldsymbol{R}_{sb} & \boldsymbol{p}_{sb} \\ \boldsymbol{0}_{1\times 3} & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{q}_b \\ 1 \end{bmatrix}. \tag{A.10}$$

Additionally, homogeneous transformations can be combined to compose a sequence of rotational and translational motions relative to some reference frame by multiplying sequential rigid body transformation matrices. This combination of successive transformations proves useful in describing the motion of multi-body systems. The configuration of a single body with respect to the spatial coordinate frame can be expressed as the composition of the individual relative motions of preceding bodies in the system. For example, knowing the relative pose between an inertial reference frame and a body-fixed coordinate frame attached to body 1, $\bar{\boldsymbol{g}}_{s1}$, and the relative pose between body 1 and body 2, $\bar{\boldsymbol{g}}_{12}$, the configuration of the second body with respect to the spatial frame $\bar{\boldsymbol{g}}_{s2}$ is determined as,

$$\bar{\boldsymbol{g}}_{s2} = \bar{\boldsymbol{g}}_{s1}\bar{\boldsymbol{g}}_{12} \in \mathrm{SE}(3). \tag{A.11}$$

Remark 2. *Hereinafter, for simplicity we drop the bar associated with the homogeneous representation of rigid transformations from our notation.*

Analogous to the exponential mapping of SO(3), the exponential map of SE(3) can be defined for the members of its Lie algebra, denoted by $\mathfrak{se}(3)$. Elements of $\mathfrak{se}(3)$ are referred to as twists and parameterized by $\boldsymbol{\xi} = [\boldsymbol{v}^T \ \boldsymbol{w}^T]^T \in \mathbb{R}^6$, where the twist coordinates are $\boldsymbol{w} \in \mathbb{R}^3$ and $\boldsymbol{v} \in \mathbb{R}^3$. The vector $\boldsymbol{w}$ refers to the infinitesimal rotation associated with the rigid body movement and the vector $\boldsymbol{v}$ corresponds to the infinitesimal translation associated with the motion. According to the twist coordinates, a member of the Lie algebra $\mathfrak{se}(3)$, $\hat{\boldsymbol{\xi}} \in \mathbb{R}^{4\times4}$, takes the following form,

$$\hat{\boldsymbol{\xi}} = \begin{bmatrix} \tilde{\boldsymbol{w}} & \boldsymbol{v} \\ \boldsymbol{0}_{1\times3} & 0 \end{bmatrix} \in \mathfrak{se}(3). \tag{A.12}$$

Hence, the hat operator acts as a vector space isomorphism between $\mathbb{R}^6$ and $\mathfrak{se}(3)$. A twist $\hat{\boldsymbol{\xi}} \in \mathfrak{se}(3)$ is a unit twist provided that $||\boldsymbol{w}|| = 1$. Similarly to SO(3), taking the exponential of a twist $\hat{\boldsymbol{\xi}}\Phi \in \mathfrak{se}(3)$, with $\hat{\boldsymbol{\xi}}$ being unit twist and $\Phi \in \mathbb{R}$, provides a mapping between $\mathfrak{se}(3)$ and its Lie group SE(3). For the case where a twist is defined by twist coordinates $(\boldsymbol{v}, \boldsymbol{0}_{3\times1})$, the following relation defines the exponential mapping,

$$e^{\hat{\boldsymbol{\xi}}\Phi} = \begin{bmatrix} \boldsymbol{1}_{3\times3} & \boldsymbol{v}\Phi \\ \boldsymbol{0}_{1\times3} & 1 \end{bmatrix} \in \mathrm{SE}(3). \tag{A.13}$$

For the case where $\boldsymbol{w} \neq 0$ (and assuming $||\boldsymbol{w}|| = 1$), the exponential of a twist is defined by

$$e^{\hat{\boldsymbol{\xi}}\Phi} = \begin{bmatrix} e^{\tilde{\boldsymbol{w}}\Phi} & (\mathbf{1}_{3\times 3} - e^{\tilde{\boldsymbol{w}}\Phi})(\tilde{\boldsymbol{w}}\boldsymbol{v}) + \boldsymbol{w}\boldsymbol{w}^T\boldsymbol{v}\Phi \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix} \in \mathrm{SE}(3). \tag{A.14}$$

The notion of screw provides a geometric interpretation of twist elements and their exponential. Screws define a particular class of rigid body motions which consist of a rotation about and a translation along a single axis. As indicated by the name, this class of motion is known as screw motion defined by an axis $\boldsymbol{l} \subset \mathbb{R}^3$ which is a line specified by a point and a direction, a pitch h, and a magnitude M. To express a screw motion as an element of $\mathrm{SE}(3)$, consider the transformation of a point $\boldsymbol{p} \in \mathbb{R}^3$ (with respect to the spatial frame) from its initial position $\boldsymbol{p}_i$ to its position following the screw motion $\boldsymbol{p}_f$ in homogeneous coordinates,

$$\begin{bmatrix} \boldsymbol{p}_f \\ 1 \end{bmatrix} = \begin{bmatrix} e^{\tilde{\boldsymbol{w}}\Phi} & (\mathbf{1}_{3\times 3} - e^{\tilde{\boldsymbol{w}}\Phi})\boldsymbol{q} + h\Phi\boldsymbol{w} \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{p}_i \\ 1 \end{bmatrix}. \tag{A.15}$$

This relationship follows the form $\boldsymbol{p}_f = \boldsymbol{g}\boldsymbol{p}_i$ where $\boldsymbol{g}$ denotes the 4×4 rigid body transformation for the screw motion, defined above. This homogeneous transformation for a screw resembles the exponential mapping of a twist to $\mathrm{SE}(3)$ for the case of $\Phi \neq 0$ defined by equation (A.14).

Given a unit twist $\boldsymbol{\xi} = [\boldsymbol{v}^T \ \boldsymbol{w}^T]^T \in \mathbb{R}^6$, an angle Φ, and the associated homogeneous transformation in (A.14), a screw's pitch that conveys the ratio of translational motion to rotational motion is found by

$$h = \frac{d}{\Phi} = \boldsymbol{w}^T\boldsymbol{v}, \tag{A.16}$$

where d is the travelled distance. In the case of pure translation, the pitch h is infinity. The product of the pitch and the net rotation, defined by the magnitude $M = \Phi$, provides the amount of translation associated with the screw motion. For a general screw motion, where $\Phi \neq 0$, the axis $\boldsymbol{l}$ is the line aligned with the unit vector $\boldsymbol{w} \in \mathbb{R}^3$, specified by

$$\boldsymbol{l} = \{\boldsymbol{w} \times \boldsymbol{v} + \lambda \boldsymbol{w} \,:\, \lambda \in \mathbb{R}\}. \tag{A.17}$$

When dealing with pure translations, i.e., $\boldsymbol{w} = 0$, the screw axis $\boldsymbol{l}$ is sought through

$$\boldsymbol{l} = \{\lambda \boldsymbol{v} \,:\, \lambda \in \mathbb{R}\}. \tag{A.18}$$

Conversely, given a screw with the homogeneous transformation in (A.15), one can find a twist

$$\boldsymbol{\xi} = \begin{bmatrix} -\boldsymbol{w} \times \boldsymbol{q} + h\boldsymbol{w} \\ \boldsymbol{w} \end{bmatrix} \tag{A.19}$$

and an angle $\Phi = M$ (assuming $\|\omega\| = 1, M \neq 0$), such that $\boldsymbol{g} = \exp(\hat{\boldsymbol{\xi}}\Phi)$. Specifically, the twist corresponding to a screw motion with zero pitch (pure rotation) is obtained by

$$\boldsymbol{\xi} = \begin{bmatrix} -\boldsymbol{w} \times \boldsymbol{q} \\ \boldsymbol{w} \end{bmatrix}, \tag{A.20}$$

and for pitch $h = \infty$ (pure translation), the corresponding twist is

$$\boldsymbol{\xi} = \begin{bmatrix} \boldsymbol{v} \\ \mathbf{0}_{3\times 1} \end{bmatrix}. \tag{A.21}$$

Based on the above characterization, twists are shown to represent screw motions and their associated homogeneous transformations can be expressed by the exponential map. That is, if $\boldsymbol{p}(0)$ denotes the initial position of a point $\boldsymbol{p}$ and $\boldsymbol{p}(\Phi)$ indicates the point's position after the screw motion, then

$$\boldsymbol{p}(\Phi) = e^{\hat{\boldsymbol{\xi}}\Phi}\boldsymbol{p}(0). \tag{A.22}$$

Similarly for the elements of SE(3), exponential of a twist provides a transformation from the initial pose of a body $\boldsymbol{g}_{sb}(0)$ to the rigid body's pose following a screw motion, i.e.,

$$\boldsymbol{g}_{sb}(\Phi) = e^{\hat{\boldsymbol{\xi}}\Phi}\boldsymbol{g}_{sb}(0). \tag{A.23}$$

The inverse of the exponential map can also be performed to extract the twist $\hat{\boldsymbol{\xi}}\Phi$ corresponding to a screw motion leading to a rigid homogeneous transformation $\boldsymbol{g}$. This inverse mapping is referred to as the log function on SE(3).

$$\hat{\boldsymbol{\xi}}\Phi = \log(\boldsymbol{g}) = \log\left(\begin{bmatrix} \boldsymbol{R} & \boldsymbol{p} \\ \boldsymbol{0}_{1\times 3} & 1 \end{bmatrix}\right) := \begin{bmatrix} \Phi\tilde{\boldsymbol{w}} & \boldsymbol{A}^{-1}\boldsymbol{p} \\ \boldsymbol{0}_{1\times 3} & 0 \end{bmatrix} \in \mathfrak{se}(3), \tag{A.24}$$

where the the 3 x 3 matrix $\boldsymbol{A}$ is equal to identity in the case of pure translational motion. For unit twists including a rotational element (i.e., $\boldsymbol{w} \neq \boldsymbol{0}$), this matrix is defined by

$$\boldsymbol{A} := (\boldsymbol{1}_{3\times 3} - \boldsymbol{R})\tilde{\boldsymbol{w}} + \boldsymbol{w}\boldsymbol{w}^T\Phi. \tag{A.25}$$

In a similar manner, the element of $\mathfrak{so}(3)$ in the definition above (i.e., $\tilde{\boldsymbol{w}}\Phi$) is obtained through the log function on the SO(3) Lie group. In this case, the inverse of the exponential mapping $\exp(\tilde{\boldsymbol{w}}\Phi)$ provides the skew-symmetric matrix of the axis of rotation $\tilde{\boldsymbol{w}}$, and the corresponding rotation angle Φ. The log function of a rotation matrix $\boldsymbol{R} \neq \boldsymbol{1}_{3\times 3} \in \mathrm{SO}(3)$ is defined through the following relationships,

$$\Phi = \cos^{-1}\left(\frac{\mathrm{trace}(\boldsymbol{R}) - 1}{2}\right), \tag{A.26}$$

$$\tilde{\boldsymbol{w}} = \frac{1}{2\sin\Phi}\left(\boldsymbol{R} - \boldsymbol{R}^T\right). \tag{A.27}$$

Observe that the axis of rotation $\boldsymbol{w}$ is not unique.

A.1.3 Rigid Body Velocity

In this subsection, we define the spatial and body velocity vectors associated with rigid body motions on the SE(3). We start by studying the strictly rotational case and subsequently we extend it to general rigid body motions. As demonstrated previously, a rotational motion is described by a time varying orientation of the body coordinate frame with respect to the spatial frame, $\boldsymbol{R}_{sb}(t) \in \mathrm{SO}(3)$. Based on this rotational trajectory, the motion of a point $\boldsymbol{q} \in \mathbb{R}^3$ (located on the rigid body) in the spatial frame is defined by,

$$\boldsymbol{q}_s(t) = \boldsymbol{R}_{sb}(t)\boldsymbol{q}_b. \tag{A.28}$$

The rate of change of the position of $\boldsymbol{q}$ defines its velocity $\boldsymbol{v} \in \mathbb{R}^3$ at a given instance. The position of $\boldsymbol{q}$ remains fixed from the perspective of the body coordinate frame (i.e., $\dot{\boldsymbol{q}}_b = 0$), however moves according to equation (A.28) when viewed in the spatial frame. Consequently, taking the derivative of equation (A.28) with respect to time defines the velocity of the trajectory $\boldsymbol{q}(t)$ as viewed from the spatial reference frame,

$$\boldsymbol{v}_s(t) = \dot{\boldsymbol{q}}_s = \frac{d}{dt}\left(\boldsymbol{R}_{sb}\boldsymbol{q}_b\right) = \dot{\boldsymbol{R}}_{sb}\boldsymbol{q}_b = \dot{\boldsymbol{R}}_{sb}\left(\boldsymbol{R}_{sb}^T\boldsymbol{R}_{sb}\right)\boldsymbol{q}_b = \left(\dot{\boldsymbol{R}}_{sb}\boldsymbol{R}_{sb}^T\right)\boldsymbol{q}_s \in \mathbb{R}^3. \tag{A.29}$$

Note that the matrix $\dot{\boldsymbol{R}}(t)\boldsymbol{R}(t)^{-1} \in \mathbb{R}^{3\times 3}$ is a skew-symmetric matrix and it belongs to $\mathfrak{so}(3)$. Let the vector $\boldsymbol{\omega}_{sb}^s \in \mathbb{R}^3$ be the relative angular velocity between the body and spatial coordinate frames, as seen from the spatial frame at a given moment in time. Then, we have

$$\tilde{\boldsymbol{\omega}}_{sb}^s = \dot{\boldsymbol{R}}_{sb}\boldsymbol{R}_{sb}^{-1} \in \mathfrak{so}(3), \tag{A.30}$$

which maps the spatial coordinates of a point to its spatial velocity based on (A.29), i.e., $\boldsymbol{v}_s = \tilde{\boldsymbol{\omega}}_{sb}^s\boldsymbol{q}_s = \boldsymbol{\omega}_{sb}^s \times \boldsymbol{q}_s$. Similarly, the body angular velocity vector $\boldsymbol{\omega}_{sb}^b \in \mathbb{R}^3$ (from the perspective of the body frame at a given instance in time) makes use of the fact that the matrix product $\boldsymbol{R}(t)^{-1}\dot{\boldsymbol{R}}(t)$ is a skew-symmetric matrix. Consequently, the body angular velocity is defined by

$$\tilde{\boldsymbol{\omega}}_{sb}^{b} = \boldsymbol{R}_{sb}^{-1}\dot{\boldsymbol{R}}_{sb} \in \mathfrak{so}(3). \tag{A.31}$$

Additionally, the spatial and body angular velocity vectors $\boldsymbol{\omega}_{sb}^{s}$ and $\boldsymbol{\omega}_{sb}^{b}$ are related through the relative rotation matrix between the spatial and body coordinate frames,

$$\boldsymbol{\omega}_{sb}^{s} = \boldsymbol{R}_{sb}\boldsymbol{\omega}_{sb}^{b} \in \mathbb{R}^3. \tag{A.32}$$

For more general rigid body motions (i.e., including translational motions), let $\boldsymbol{g}_{sb}(t) \in \mathrm{SE}(3)$ indicate a trajectory of the pose of the body coordinate frame with respect to the spatial frame. Analogous to the rotational motion, the matrices $\dot{\boldsymbol{g}}_{sb}\boldsymbol{g}_{sb}^{-1}$ and $\boldsymbol{g}_{sb}^{-1}\dot{\boldsymbol{g}}_{sb}$ belong to the Lie algebra $\mathfrak{se}(3)$, and they respectively define the rigid body's instantaneous spatial velocity $\boldsymbol{V}_{sb}^{s} \in \mathbb{R}^6$ and body velocity $\boldsymbol{V}_{sb}^{b} \in \mathbb{R}^6$ with respect to the spatial frame. The spatial velocity in twist coordinates reads

$$\hat{\boldsymbol{V}}_{sb}^{s} = \dot{\boldsymbol{g}}_{sb}\boldsymbol{g}_{sb}^{-1} = \begin{bmatrix} \boldsymbol{v}_{sb}^{s} \\ \boldsymbol{\omega}_{sb}^{s} \end{bmatrix}^{\wedge} \in \mathfrak{se}(3). \tag{A.33}$$

The first three components of the spatial velocity vector $\boldsymbol{v}_{sb}^{s} \in \mathbb{R}^3$ correspond to the linear velocity with respect to the spatial coordinate frame. Physically speaking, this linear velocity represents the velocity of a particle in the rigid body which travels through the origin of the spatial frame at the given moment in time. Furthermore, the last three components of the spatial velocity $\boldsymbol{\omega}_{sb}^{s} \in \mathbb{R}^3$ specify the spatial angular velocity of the body. As previously mentioned, this angular velocity simply represents the rigid body's rate of rotation from the perspective of the spatial frame.

The rigid body velocity vector can also be expressed in the body coordinate frame, by extending the definition of the body angular velocity in (A.31) to motions in $\mathrm{SE}(3)$. The body velocity in twist coordinates is defined by

$$\hat{\boldsymbol{V}}_{sb}^{b} = \boldsymbol{g}_{sb}^{-1}\dot{\boldsymbol{g}}_{sb} = \begin{bmatrix} \boldsymbol{v}_{sb}^{b} \\ \boldsymbol{\omega}_{sb}^{b} \end{bmatrix}^{\wedge} \in \mathfrak{se}(3). \tag{A.34}$$

Here, the linear component of the body velocity $\boldsymbol{v}_{sb}^{b} \in \mathbb{R}^3$ is physically interpreted as the relative velocity between the origins of the body and spatial coordinate frames viewed from the perspective of the body frame. The angular velocity $\boldsymbol{\omega}_{sb}^{b} \in \mathbb{R}^3$ again represents the rotational velocity of the rigid body as viewed in the body frame.

Relating the spatial and body velocity vectors requires some form of a transformation between the two coordinate frames. The Adjoint operator is a linear automorphism of the Lie algebra $\mathfrak{se}(3)$ which converts the expression of a twist from one coordinate frame to another. Provided a relative homogeneous transformation between two coordinate frames $\boldsymbol{g} \in \mathrm{SE}(3)$ with a rotation $\boldsymbol{R} \in \mathrm{SO}(3)$ and a position $\boldsymbol{p} \in \mathbb{R}^3$, the associated Adjoint mapping denoted $\mathbf{Ad}_{\boldsymbol{g}}$ is defined by the following 6×6 matrix,

$$\mathbf{Ad}_{\boldsymbol{g}} = \begin{bmatrix} \boldsymbol{R} & \tilde{\boldsymbol{p}}\boldsymbol{R} \\ \mathbf{0}_{3\times 3} & \boldsymbol{R} \end{bmatrix}. \tag{A.35}$$

Taking the Adjoint of the homogeneous transformation from the body coordinate frame to the spatial coordinate frame $\boldsymbol{g}_{sb}$ provides the following relationship which converts the body velocities to the spatial velocities,

$$\boldsymbol{V}^{s} = \mathbf{Ad}_{\boldsymbol{g}_{sb}}\boldsymbol{V}^{b}. \tag{A.36}$$

The inverse transformation converts the spatial velocities to the body velocities:

$$\boldsymbol{V}^{b} = \mathbf{Ad}_{\boldsymbol{g}_{sb}}^{-1}\boldsymbol{V}^{s}, \tag{A.37}$$

where the closed form definition for the inverse of the Adjoint operator is

$$\mathbf{Ad}_{\boldsymbol{g}}^{-1} = \mathbf{Ad}_{\boldsymbol{g}^{-1}} = \begin{bmatrix} \boldsymbol{R}^T & -\boldsymbol{R}^T \tilde{\boldsymbol{p}} \\ \mathbf{0}_{3\times 3} & \boldsymbol{R}^T \end{bmatrix}. \tag{A.38}$$

Similar to the Adjoint mapping, the Lie bracket operator on the Lie algebra $\mathfrak{se}(3)$, introduces a linear mapping between elements of $\mathfrak{se}(3)$ that is denoted by $\mathbf{ad}_{\boldsymbol{\xi}}$ for a twist $\boldsymbol{\xi} = [\boldsymbol{v}^T \ \boldsymbol{w}^T]^T \in \mathbb{R}^6$. This mapping is called the adjoint operator and defined by

$$\mathbf{ad}_{\boldsymbol{\xi}} = \begin{bmatrix} \tilde{\boldsymbol{w}} & \tilde{\boldsymbol{v}} \\ \mathbf{0}_{3\times 3} & \tilde{\boldsymbol{w}} \end{bmatrix}. \tag{A.39}$$

Appendix B

PID Control Design

This section describes the design of a PID controller leading to exponential stability of the end-effector linear motion to a desired trajectory. The tracking error $\boldsymbol{e}(t) \in \mathbb{R}^3$ is defined as the difference between the desired end-effector trajectory $\bar{\boldsymbol{p}}_{sN}(t) \in \mathbb{R}^3$ and the actual trajectory of the output $\boldsymbol{p}_{sN}(t) \in \mathbb{R}^3$. A PID control law is predicated on the combination of three terms: an amplified tracking error signal, an amplified signal of the velocity error (i.e., the time derivative of the tracking error), and an amplified signal for the integral of the tracking error. The gains for amplifying each respective signal are known as the proportional, derivative, and integral gains indicated by the matrices $\boldsymbol{K}_p$, $\boldsymbol{K}_d$, and $\boldsymbol{K}_i$. For the case of controlling the end-effector position in $\mathbb{R}^3$, the PID gains are selected to be 3×3 diagonal matrices with strictly positive diagonal elements. The diagonal elements signify control over the x, y, and z coordinates of the output:

$$\boldsymbol{K}_p = \begin{bmatrix} K_{p_1} & 0 & 0 \\ 0 & K_{p_2} & 0 \\ 0 & 0 & K_{p_3} \end{bmatrix}, \quad \boldsymbol{K}_d = \begin{bmatrix} K_{d_1} & 0 & 0 \\ 0 & K_{d_2} & 0 \\ 0 & 0 & K_{d_3} \end{bmatrix}, \quad \boldsymbol{K}_i = \begin{bmatrix} K_{i_1} & 0 & 0 \\ 0 & K_{i_2} & 0 \\ 0 & 0 & K_{i_3} \end{bmatrix}. \tag{B.1}$$

As previously mentioned, the output of the PID control law $\boldsymbol{U}_v$ controls the behaviour of the tracking error dynamics. The goal of this control output is to force the linear motion of the end-effector $\boldsymbol{p}_{sN}(t)$ to match the desired trajectory $\bar{\boldsymbol{p}}_{sN}(t)$ provided by a trajectory planner. The PID controller is defined as follows,

$$\boldsymbol{U}_v = \dot{\bar{\boldsymbol{v}}}_N^s + \boldsymbol{K}_d\dot{\boldsymbol{e}} + \boldsymbol{K}_p\boldsymbol{e} + \boldsymbol{K}_i\int \boldsymbol{e}\,dt, \tag{B.2}$$

where as described,

$$\boldsymbol{e}(t) = \bar{\boldsymbol{p}}_{sN}(t) - \boldsymbol{p}_{sN}(t), \tag{B.3}$$

$$\dot{\boldsymbol{e}}(t) = \bar{\boldsymbol{v}}_N^s(t) - \boldsymbol{v}_N^s(t). \tag{B.4}$$

To design a desired behaviour for the tracking error, the time response of the closed-loop error dynamics must be analyzed,

$$\ddot{\boldsymbol{e}} + \boldsymbol{K}_d\dot{\boldsymbol{e}} + \boldsymbol{K}_p\boldsymbol{e} + \boldsymbol{K}_i\int \boldsymbol{e}\,dt = \boldsymbol{0}_{3\times 1}. \tag{B.5}$$

Due to the choice of diagonal gain matrices, in each direction of motion the error dynamics is fully decoupled and provides a linear third order differential equation. That is, if we define $\boldsymbol{E} := \int \boldsymbol{e}\,dt$, for $j = 1,2,3$ we have

$$\dddot{E}_j + K_{d_j}\ddot{E}_j + K_{p_j}\dot{E}_j + K_{i_j}E_j = 0. \tag{B.6}$$

Taking the Laplace transform of the governing equations, we form the characteristic equations

$$s^3 + K_{d_j}s^2 + K_{p_j}s + K_{i_j} = 0. \tag{B.7}$$

Given strictly positive gains and based on the Routh-Hurwitz stability criterion, it is easy to check that the output tracking error is globally exponentially stable at $E = 0$, if for every $j = 1,2,3$,

$$K_{i_j} < K_{p_j}K_{d_j}. \tag{B.8}$$

Hence, under this condition exponential linear trajectory tracking of the end-effector can be achieved. Evidently, the PID gains are used to alter the root locations of the characteristic equation such that a desired behaviour in the error dynamics is achieved.

In this thesis, the characteristic equation for the error behaviour in j^{th} direction is designed to contain three repeated negative real roots whose location is denoted by $-a_j$. This control design signifies a critically damped behaviour in the time response of the tracking error integral. Therefore,

$$s^3 + K_{d_j}s^2 + K_{p_j}s + K_{i_j} = (s+a_j)^3 = s^3 + 3a_j s^2 + 3a_j^2 s + a_j^3. \tag{B.9}$$

This results in the following definitions for the diagonal elements of the proportional, derivative, and integral gains,

$$K_{p_j} = 3a_j^2, \quad K_{d_j} = 3a_j, \quad K_{i_j} = a_j^3. \tag{B.10}$$

Based on the designed gains for the behaviour of the tracking error in the Laplace domain, taking the inverse Laplace transform provides the time response $E_j(t)$ for a step input. In this domain, the critically damped nature of the tracking error becomes apparent from the time response being independent of any sinusoidal functions. The following equation is the time response of the integral of the error along a single axis,

$$E_j(t) = C_{1j}e^{-a_j t} + C_{2j}te^{-a_j t} + C_{3j}t^2e^{-a_j t}, \tag{B.11}$$

where C_{1j}, C_{2j}, and C_{3j} are defined by the initial conditions to the tracking error, its integral, and the velocity error. Therefore the time response of the tracking error is

$$e_j(t) = \left(C_{2j} - aC_{1j}\right)e^{-a_j t} + \left(2C_{3j} - aC_{2j}\right)te^{-a_j t} - aC_{3j}t^2e^{-a_j t}. \tag{B.12}$$

Appendix C

Singularity Accommodation

Within a trajectory following task, it is possible for the desired trajectory to force the free-floating space manipulator system into a singular configuration at some point along the requested path. This singular configuration stems from the internal dynamics of the system and causes the system's Jacobian matrix to contain linear dependency and consequently lose rank. Referring back to the definition of the GJM in (5.50), the system Jacobian losing rank in turn causes $\boldsymbol{J}^*$ to become ill conditioned. Taking the pseudo-inverse of the GJM (in the case of a redundant manipulator) when in the neighbourhood of a singular configuration, as required in the feedback linearization control law in equation (5.53), consequently commands unattainable actuation torques even for minimal movement in the output (in the singular directions). Managing the output tracking controller to operate within a safe region in the manipulator's joint space (i.e., a region which prevents excessive and potentially damaging joint velocities) requires the introduction of some method in the control design to push the system away from singular configurations.

This thesis implements the damped least-squares method separately proposed in 1986 by [82] and [83] to address the singularity avoidance problem in the output tracking control task. The damped-least squares method adds a damping factor to the inverse differential kinematics which reduces the trajectory following capabilities, however maintains the joints within a tolerable motion threshold. Consequently, singularity accommodation through the damped least-squares method is a constant exchange, quantified by the damping factor, between performance and practical control actions at the joints. First, let us explicitly define the formulation of the pseudo-inverse in a redundant manipulator's inverse kinematics with some Jacobian matrix $\boldsymbol{J}$:

$$\dot{\boldsymbol{\Phi}} = \boldsymbol{J}^{\dagger}\boldsymbol{V}_N = \boldsymbol{J}^T(\boldsymbol{J}\boldsymbol{J}^T)^{-1}\boldsymbol{V}_N. \tag{C.1}$$

The pseudo-inverse shown in equation (C.1) satisfies a solution for the inverse kinematics problem which minimizes the least-squares norm. That is, equation (C.1) satisfies:

$$\min(||\boldsymbol{V}_N - \boldsymbol{J}\dot{\boldsymbol{\Phi}}||). \tag{C.2}$$

Evidently, the solution associated with the minimum norm provides a set of joint velocities (for which infinitely many solutions exist for redundant manipulators) which are most accurate in generating the end-effector velocity vector $\boldsymbol{V}_N$. As previously mentioned, strictly choosing the solution yielding the minimum norm is troublesome when the system is in the region of singular configurations. This solution no longer becomes feasible in its practical implementation due to the large joint velocities required to achieve the corresponding end-effector motion in the singular direction.

The damped least-squares method alters the problem for which the inverse kinematics is solved in order to consider the practical feasibility of the solution as well. In addition to considering the accuracy of the solution through the least-squares minimum norm problem in equation (C.2), the damped least-squares method additionally considers the norm of the joint velocities. Now, the solution to the inverse kinematics problem aims to minimize the weighted sum of the joint velocities accuracy and feasibility provided positive definite weighting factors $\boldsymbol{W}_1$ and $\boldsymbol{W}_2$. That is, the damped least squares solution satisfies the following problem [82],

$$\min(||\boldsymbol{V}_N - \boldsymbol{J}\dot{\boldsymbol{\Phi}}||^2_{\boldsymbol{W}_1} + ||\dot{\boldsymbol{\Phi}}||^2_{\boldsymbol{W}_2}), \tag{C.3}$$

where the vector norms above are defined equivalently to equation (5.70). As indicated by (C.3), the weight $\boldsymbol{W}_1$ signifies the emphasis placed on the accuracy of the mapping between joint and end-effector velocities, whereas the weight $\boldsymbol{W}_2$ denotes the importance of feasible joint velocities in the inverse kinematics solution. In [82], the authors present the solution to the damped least squares problem in (C.3) known as the Singularity Robust inverse (SR-inverse). As mentioned in [84], a considerable amount of the literature studies the case

involving no task priority (i.e., $\boldsymbol{W}_1 = \mathbf{1}$) and the damping of joint velocities through setting $\boldsymbol{W}_2 = \lambda\mathbf{1}$. Since the weights are defined as positive definite matrices, the damping factor $\lambda \geq 0$. Using these weights, the SR-inverse of $\boldsymbol{J}$, denoted $\boldsymbol{J}^{\diamond}$, is defined as follows,

$$\boldsymbol{J}^{\diamond} = \boldsymbol{J}^T(\boldsymbol{J}\boldsymbol{J}^T - \lambda\mathbf{1})^{-1}. \tag{C.4}$$

Note that the SR-inverse solution above reduces to the Moores-Penrose pseudo-inverse (equation (C.1)) in the case of no damping on the joint velocities (i.e. $\lambda = 0$). Such inverse method is thus implemented on the GJM matrices ($\boldsymbol{J}_v^*$ and J^*) included in the feedback laws for both control cases. In the context of a trajectory following task, increasing the damping factor reduces the performance of the output's tracking capabilities. Consequently, it is most beneficial during a trajectory following scenario to only implement a joint velocity damping effect once the system's Jacobian matrix begins to lose rank close to a singular configuration. Outside of this region, when the Jacobian is well defined, the Moores-Penrose pseudo-inverse solution is preferred to achieve the greatest performance of the output tracking control law. Taking the determinant of the matrix $\boldsymbol{J}\boldsymbol{J}^T$ (i.e., the matrix being inverted in the pseudo-inverse) provides a metric for identifying ill conditioned systems. When $\det(\boldsymbol{J}\boldsymbol{J}^T) = 0$, the system reaches a singular configuration and the matrix $\boldsymbol{J}\boldsymbol{J}^T$ loses rank. Using the definition of a system's manipulability w presented by Yoshikawa in [85] as,

$$w := \sqrt{\det(\boldsymbol{J}\boldsymbol{J}^T)}, \tag{C.5}$$

to quantify the proximity to a singularity, the authors in [82] scale the damping factor based on w and a manipulability threshold value w_t. Above this threshold, the system is deemed sufficiently well defined to strictly consider accuracy in the inverse kinematics solution using (C.1). Once below w_t, the system is in the neighbourhood of a singular configuration, thus requiring a compromise in accuracy to provide damping to the joint velocities. Provided a maximum damping value λ_0 for when the system reaches singularity and $w = 0$, the damping factor scales as follows,

$$\lambda = \begin{cases} \lambda_0 \left(1 - \frac{w}{w_t}\right)^2, & w < w_t, \\ 0, & w \geq w_t. \end{cases} \tag{C.6}$$

Note that additional methods for defining the damping factor have been studied in the literature, involving the rate of change of the manipulatbility factor, the tracking error, the minimum singular value of $\boldsymbol{J}$, and the condition number of $\boldsymbol{J}\boldsymbol{J}^T + \lambda \mathbf{1}$, for example [84, 86, 87, 88, 89]. Based on an empirical analysis, the manipulability threshold w_t is set to $\sqrt{10}$ and the maximum damping λ_0 is set to 200 in this thesis.

Appendix D

Homogeneous Transformation Partial Derivative

The following provides the derivation of the partial derivative of a homogeneous transformation between the spatial frame and a coordinate frame attached to body k, $\boldsymbol{g}_{sk} \in \mathrm{SE}(3)$, with respect to the i^{th} generalized coordinate Φ_i [90]:

$$\frac{\partial \boldsymbol{g}_{sk}}{\partial \Phi_i} = e^{\hat{\boldsymbol{\xi}}_1 \Phi_1} \dots e^{\hat{\boldsymbol{\xi}}_{i-1} \Phi_{i-1}} \frac{\partial}{\partial \Phi_i} \left(e^{\hat{\boldsymbol{\xi}}_i \Phi_i} \right) e^{\hat{\boldsymbol{\xi}}_{i+1} \Phi_{i+1}} \dots e^{\hat{\boldsymbol{\xi}}_k \Phi_k} \boldsymbol{g}_{sk}(0) \tag{D.1}$$

$$= e^{\hat{\boldsymbol{\xi}}_1 \Phi_1} \dots e^{\hat{\boldsymbol{\xi}}_{i-1} \Phi_{i-1}} \left(\hat{\boldsymbol{\xi}}_i \right) e^{\hat{\boldsymbol{\xi}}_i \Phi_i} \dots e^{\hat{\boldsymbol{\xi}}_k \Phi_k} \boldsymbol{g}_{sk}(0), \tag{D.2}$$

which when multiplied with $\boldsymbol{g}_{sk}^{-1}$ yields the following,

$$\left(\frac{\partial \boldsymbol{g}_{sk}}{\partial \Phi_i} \right) \boldsymbol{g}_{sk}^{-1} = e^{\hat{\boldsymbol{\xi}}_1 \Phi_1} \dots e^{\hat{\boldsymbol{\xi}}_{i-1} \Phi_{i-1}} \left(\hat{\boldsymbol{\xi}}_i \right) e^{-\hat{\boldsymbol{\xi}}_{i-1} \Phi_{i-1}} \dots e^{-\hat{\boldsymbol{\xi}}_1 \Phi_1}. \tag{D.3}$$

Appendix E

Adjoint Operator Partial Derivatives

In the following, the derivation of the partial derivative of the Adjoint operator $\mathbf{Ad}_j^l$ being applied to an element $\hat{\boldsymbol{A}} \in \mathfrak{se}(3)$, with respect to a generalized coordinate Φ_i is provided:

$$\begin{aligned}
\frac{\partial \mathbf{Ad}_j^l \boldsymbol{A}}{\partial \Phi_i} &= e^{-\hat{\boldsymbol{\xi}}_l \Phi_l} \dots e^{-\hat{\boldsymbol{\xi}}_{i+1} \Phi_{i+1}} e^{-\hat{\boldsymbol{\xi}}_i \Phi_i} \left(-\hat{\boldsymbol{\xi}}_i\right) \dots e^{-\hat{\boldsymbol{\xi}}_k \Phi_k} \left(\hat{\boldsymbol{A}}\right) e^{\hat{\boldsymbol{\xi}}_k \Phi_k} \dots e^{\hat{\boldsymbol{\xi}}_l \Phi_l} \\
&\quad + e^{-\hat{\boldsymbol{\xi}}_l \Phi_l} \dots e^{-\hat{\boldsymbol{\xi}}_k \Phi_k} \left(\hat{\boldsymbol{A}}\right) e^{\hat{\boldsymbol{\xi}}_k \Phi_k} \dots e^{\hat{\boldsymbol{\xi}}_i \Phi_i} \left(\hat{\boldsymbol{\xi}}_i\right) e^{\hat{\boldsymbol{\xi}}_{i+1} \Phi_{i+1}} \dots e^{\hat{\boldsymbol{\xi}}_l \Phi_l} && \text{(E.1)} \\
&= -e^{-\hat{\boldsymbol{\xi}}_l \Phi_l} \dots e^{-\hat{\boldsymbol{\xi}}_{i+1} \Phi_{i+1}} \left(\hat{\boldsymbol{\xi}}_i\right) \left(\mathbf{Ad}_j^i \boldsymbol{A}\right)^\wedge e^{\hat{\boldsymbol{\xi}}_{i+1} \Phi_{i+1}} \dots e^{\hat{\boldsymbol{\xi}}_l \Phi_l} \\
&\quad + e^{-\hat{\boldsymbol{\xi}}_l \Phi_l} \dots e^{-\hat{\boldsymbol{\xi}}_{i+1} \Phi_{i+1}} \left(\mathbf{Ad}_j^i \boldsymbol{A}\right)^\wedge \left(\hat{\boldsymbol{\xi}}_i\right) e^{\hat{\boldsymbol{\xi}}_{i+1} \Phi_{i+1}} \dots e^{\hat{\boldsymbol{\xi}}_l \Phi_l} && \text{(E.2)} \\
&= -e^{-\hat{\boldsymbol{\xi}}_l \Phi_l} \dots e^{-\hat{\boldsymbol{\xi}}_{i+1} \Phi_{i+1}} \left[\hat{\boldsymbol{\xi}}_i \left(\mathbf{Ad}_j^i \boldsymbol{A}\right)^\wedge + \left(\mathbf{Ad}_j^i \boldsymbol{A}\right)^\wedge \hat{\boldsymbol{\xi}}_i\right] e^{\hat{\boldsymbol{\xi}}_l \Phi_l} \dots e^{\hat{\boldsymbol{\xi}}_{i+1} \Phi_{i+1}} && \text{(E.3)} \\
&= -\mathbf{Ad}_{i+1}^l \mathbf{ad}_{\hat{\xi}_i} \mathbf{Ad}_j^i (\boldsymbol{A}) && \text{(E.4)}
\end{aligned}$$

where between (E.1) and (E.2) we note that $e^{\hat{\boldsymbol{\xi}}_i \Phi_i} \hat{\boldsymbol{\xi}}_i = \hat{\boldsymbol{\xi}}_i e^{\hat{\boldsymbol{\xi}}_i \Phi_i}$.

We now present the derivation of the partial derivative $\frac{\partial (\mathbf{Ad}_1^{j-1})'}{\partial \Phi_i}$ required for equation (5.8). We again demonstrate the derivation for a general twist $\hat{\boldsymbol{A}} \in \mathfrak{se}(3)$:

$$\frac{\partial(\mathbf{Ad}_1^{j-1})'\boldsymbol{A}}{\partial\Phi_i} = e^{\hat{\boldsymbol{\xi}}_1\Phi_1}\dots e^{\hat{\boldsymbol{\xi}}_i\Phi_i}\left(\hat{\boldsymbol{\xi}}_i\right)e^{\hat{\boldsymbol{\xi}}_{i+1}\Phi_{i+1}}\dots e^{\hat{\boldsymbol{\xi}}_{j-1}\Phi_{j-1}}\left(\hat{\boldsymbol{A}}\right)e^{-\hat{\boldsymbol{\xi}}_{j-1}\Phi_{j-1}}\dots e^{-\hat{\boldsymbol{\xi}}_1\Phi_1}$$
$$- e^{\hat{\boldsymbol{\xi}}_1\Phi_1}\dots e^{\hat{\boldsymbol{\xi}}_{j-1}\Phi_{j-1}}\left(\hat{\boldsymbol{A}}\right)e^{-\hat{\boldsymbol{\xi}}_{j-1}\Phi_{j-1}}\dots e^{-\hat{\boldsymbol{\xi}}_{i+1}\Phi_{i+1}}e^{-\hat{\boldsymbol{\xi}}_i\Phi_i}\left(\hat{\boldsymbol{\xi}}_i\right)\dots e^{-\hat{\boldsymbol{\xi}}_1\Phi_1} \tag{E.5}$$

$$= e^{\hat{\boldsymbol{\xi}}_1\Phi_1}\dots e^{\hat{\boldsymbol{\xi}}_i\Phi_i}\left[\hat{\boldsymbol{\xi}}_i\left((\mathbf{Ad}_{i+1}^{j-1})'\boldsymbol{A}\right)^{\wedge} - \left((\mathbf{Ad}_{i+1}^{j-1})'\boldsymbol{A}\right)^{\wedge}\hat{\boldsymbol{\xi}}_i\right]e^{-\hat{\boldsymbol{\xi}}_i\Phi_i}\dots e^{\hat{\boldsymbol{\xi}}_1\Phi_1} \tag{E.6}$$

$$= (\mathbf{Ad}_1^i)'\mathbf{ad}_{\boldsymbol{\xi}_i}(\mathbf{Ad}_{i+1}^{j-1})'(\boldsymbol{A}). \tag{E.7}$$

Appendix F

Testing Cases in Validation Procedure

Table F.1: Space Manipulator Configurations Tested

Position										
Φ_1 [m]	0	0	0	0	0	0	0	0	0	1
Φ_2 [m]	0	0	0	0	0	0	0	0	0	3.2
Φ_3 [m]	0	0	0	0	0	0	0	0	0	1.5
Φ_4 [deg]	0	60	0	90	0	45	-75	54	50	18
Φ_5 [deg]	0	0	90	0	45	65	-35	82	-20	74
Φ_6 [deg]	0	30	0	0	60	0	80	-26	63	-35
Φ_7 [deg]	45	0	30	0	-45	0	-50	114	135	70
Φ_8 [deg]	30	45	0	90	30	-50	35	79	26	20
Φ_9 [deg]	-60	0	70	0	0	40	45	-5	55	60
Φ_{10} [deg]	40	90	-80	-45	70	0	70	-26	-25	-45
Φ_{11} [deg]	0	0	0	0	0	65	-45	31	80	-30
Φ_{12} [deg]	0	90	0	0	50	0	80	59	-49	-60
Φ_{13} [deg]	0	0	0	0	0	0	-60	36	64	20

Table F.2: System of Generalized Velocities Tested

Velocity								
$\dot{\Phi}_1$ [m/s]	0	0	0	2	1	0	0	2.3
$\dot{\Phi}_2$ [m/s]	0	0	0	2	0	2	0	1.9
$\dot{\Phi}_3$ [m/s]	0	0	0	1	1	0	0	2.5
$\dot{\Phi}_4$ [rad/s]	0	2	1.5	0.8	0	1	0.9	2.3
$\dot{\Phi}_5$ [rad/s]	0	1	1.8	1.5	0	2	3	2.2
$\dot{\Phi}_6$ [rad/s]	0	0	2	3	2	0	1.5	2
$\dot{\Phi}_7$ [rad/s]	1.2	2	0	0	1.5	1.2	1	1.5
$\dot{\Phi}_8$ [rad/s]	0.5	0	0	0	0.5	0.8	2	0.2
$\dot{\Phi}_9$ [rad/s]	0	1.5	0	2	2	0	1.3	2
$\dot{\Phi}_{10}$ [rad/s]	1.1	3	2	0	1.6	0	2	1.8
$\dot{\Phi}_{11}$ [rad/s]	0	0.05	0.09	0.1	0	0.08	0.03	0.1
$\dot{\Phi}_{12}$ [rad/s]	0	0	0.03	0.09	0	0.2	0.1	0.05
$\dot{\Phi}_{13}$ [rad/s]	0	0.08	0.06	0.1	0	0.15	0.07	0.08

Table F.3: Force and Torque Vectors Tested

Force/Torque								
F_1 [N]	0	0	0	2	0	4.5	0	9
F_2 [N]	0	0	0	4	8.5	0	0	1
F_3 [N]	0	0	0	5.5	0	1.5	0	5
T_4 [Nm]	0	4.5	0	0	2	0	7.4	3
T_5 [Nm]	0	6.5	7	0	6.5	0	1.2	4.5
T_6 [Nm]	0	3	2.5	8	0	0	5.6	4.5
T_7 [Nm]	0	0	4	1	6	0	2.2	8
T_8 [Nm]	0	9	3.5	0	0.5	2.5	3	1.6
T_9 [Nm]	0	0	5	3.5	0	2	9.1	2.8
T_{10} [Nm]	6	8	0	0	4.5	0	2.2	9
T_{11} [Nm]	0.005	0	0	0.003	0	0.004	0.009	0.003
T_{12} [Nm]	0.008	0	0	0.003	0	0.005	0.009	0.005
T_{13} [Nm]	0.002	0	0	0.009	0.005	0	0.002	0.007

Appendix G

Base Positions for Linear Trajectory Following Scenarios

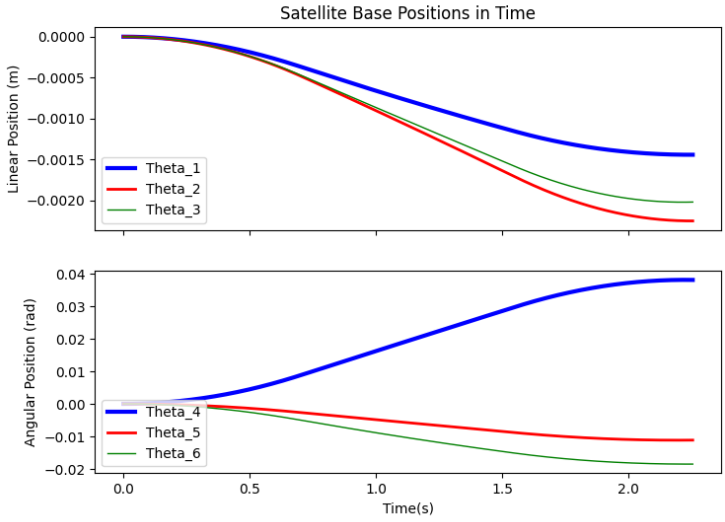


Figure G.1: Base positions for the linear trapezoidal trajectory applied to a zero momentum system.

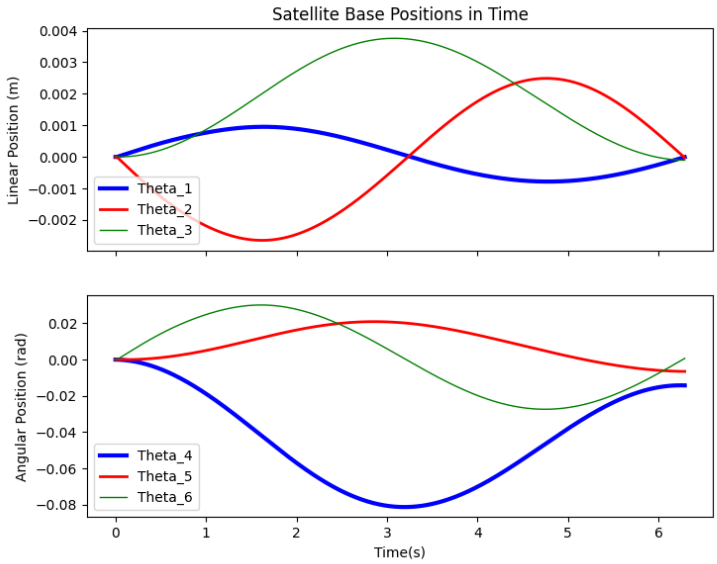


Figure G.2: Base positions for the linear circular trajectory applied to a zero momentum system.

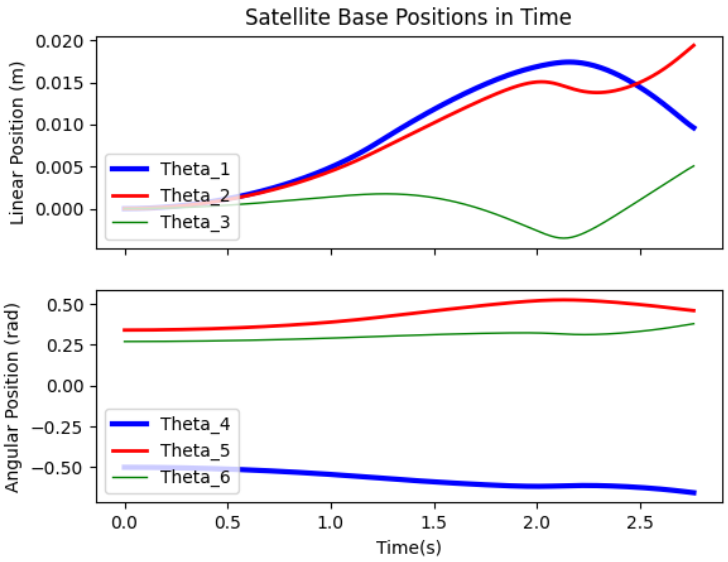


Figure G.3: Base positions for the trajectory in the presence of a singularity.

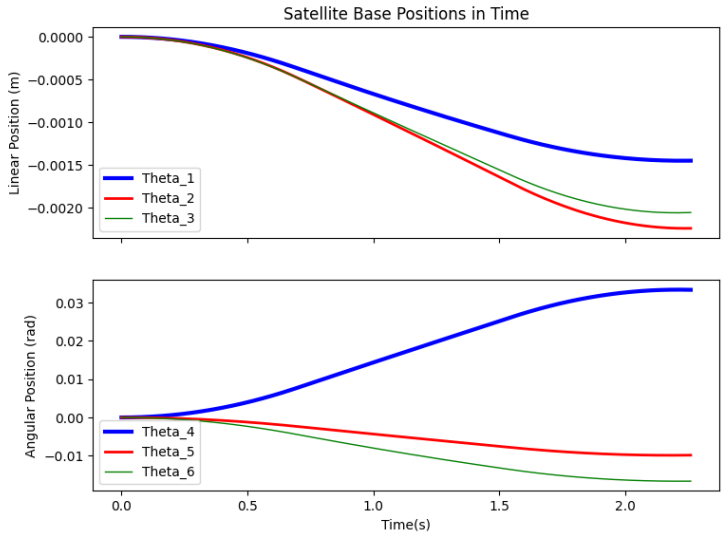


Figure G.4: Base positions for the linear trapezoidal trajectory applied to a zero momentum system with uncertainties.

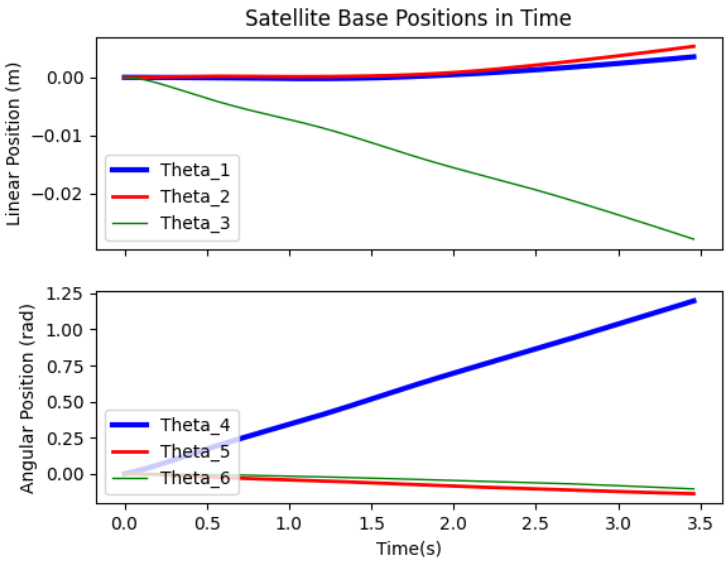


Figure G.5: Base positions for the linear trapezoidal trajectory applied to a non-zero momentum system.

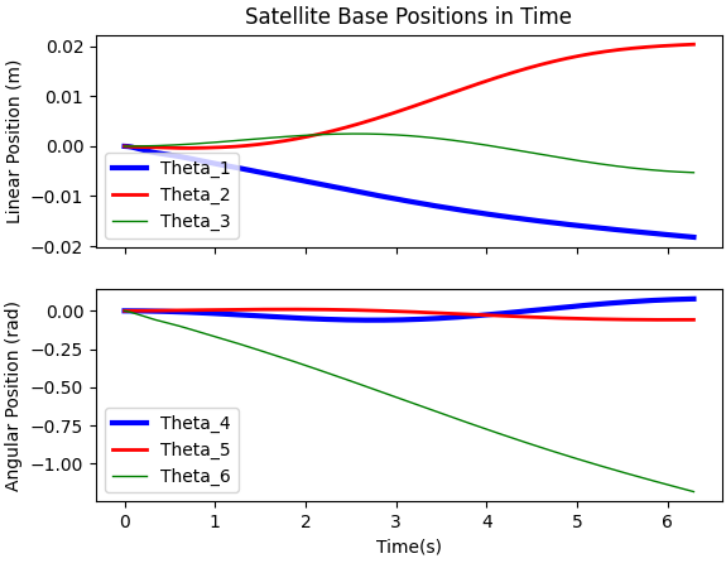


Figure G.6: Base positions for the linear circular trajectory applied to a non-zero momentum system.

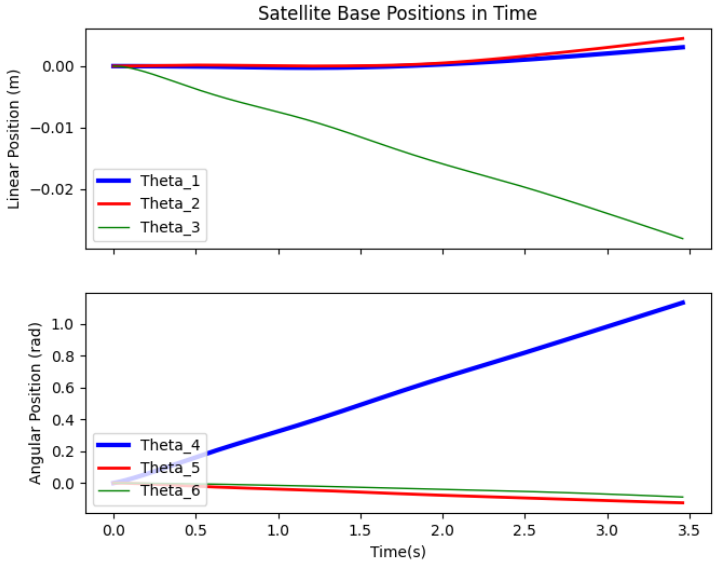


Figure G.7: Base positions for the linear trapezoidal trajectory applied to a non-zero momentum system with uncertainties.

Appendix H

Base Positions for Full Pose Trajectory Following Scenarios

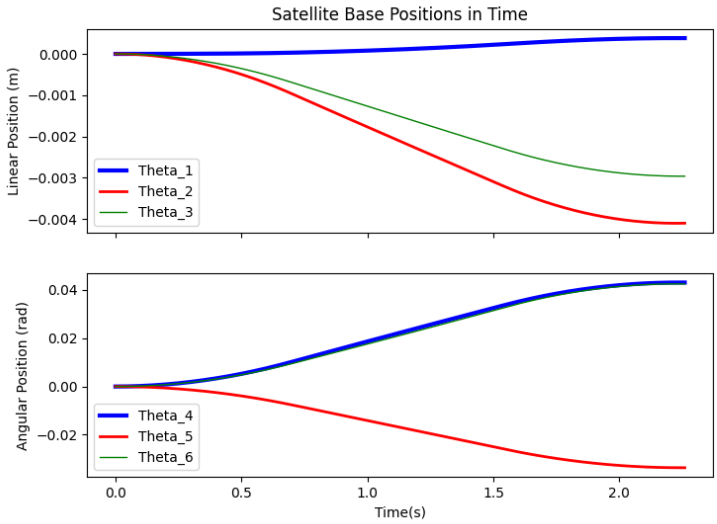


Figure H.1: Base positions for the full pose trapezoidal trajectory applied to a zero momentum system.

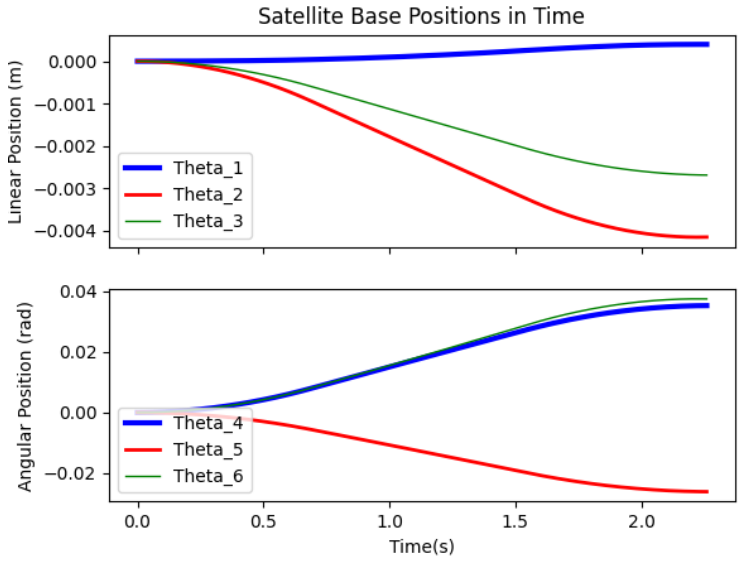


Figure H.2: Base positions for the full pose trapezoidal trajectory applied to a zero momentum system with uncertainties.

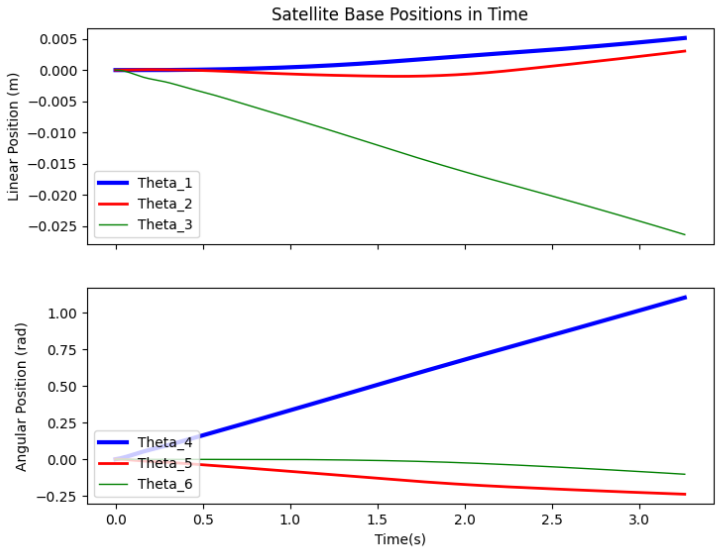


Figure H.3: Base positions for the full pose trapezoidal trajectory applied to a non-zero momentum system.

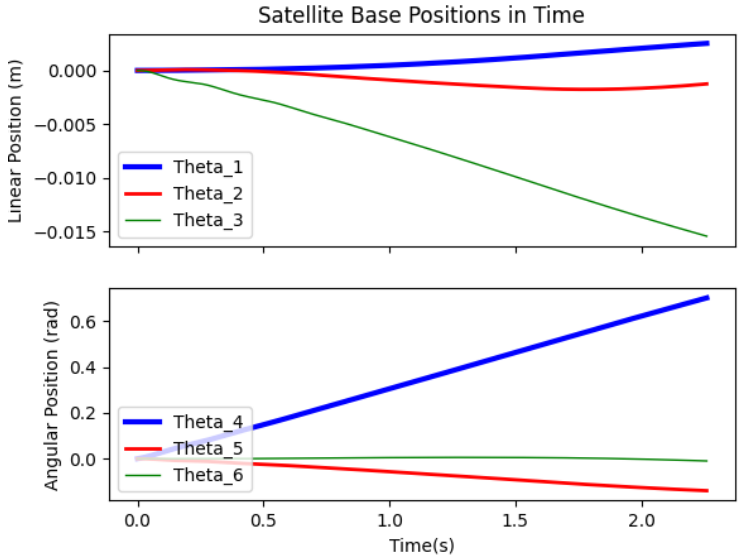


Figure H.4: Base positions for the full pose trapezoidal trajectory applied to a non-zero momentum system with uncertainties.

Bibliography

[1] D. Grover, S. Jacobs, V. Abbasi, D. Cree, M. Daae, J. Hay, W. He, X. Huang, Z. Jun, S. Kearney, *et al.*, "Development of on-orbit servicing concepts, technology option and roadmap (part i)-commercial aspects," *Journal of the British Interplanetary Society*, vol. 61, pp. 203–212, 2008.

[2] A. R. Graham and J. Kingston, "Assessment of the commercial viability of selected options for on-orbit servicing (oos)," *Acta Astronautica*, vol. 117, pp. 38–48, 2015.

[3] K. Daneshjou, A. Mohammadi-Dehabadi, and M. Bakhtiari, "Mission planning for on-orbit servicing through multiple servicing satellites: A new approach," *Advances in Space Research*, vol. 60, no. 6, pp. 1148–1162, 2017.

[4] D. Sternberg, M. Chodas, C. Jewison, M. Jones, and O. De Weck, "Multidisciplinary system design optimization of on orbit satellite assembly architectures," in *2015 IEEE Aerospace Conference*, pp. 1–14, IEEE, 2015.

[5] Y.-H. Wu, Z.-C. Yu, C.-Y. Li, M.-J. He, B. Hua, and Z.-M. Chen, "Reinforcement learning in dual-arm trajectory planning for a free-floating space robot," *Aerospace Science and Technology*, vol. 98, p. 105657, 2020.

[6] C. Kosmas, "Service vehicle for performing in-space operations on a target spacecraft, servicing system and method for using a service vehicle," July 6 2006. US Patent App. 10/539,489.

[7] J.-M. Bourjolly, O. Gurtuna, and A. Lyngvi, "On-orbit servicing: a time-dependent, moving-target traveling salesman problem," *International Transactions in Operational Research*, vol. 13, no. 5, pp. 461–481, 2006.

[8] M. Shan, J. Guo, and E. Gill, "Review and comparison of active space debris capturing and removal methods," *Progress in Aerospace Sciences*, vol. 80, pp. 18–32, 2016.

[9] D. J. Kessler and B. G. Cour-Palais, "Collision frequency of artificial satellites: The creation of a debris belt," *Journal of Geophysical Research: Space Physics*, vol. 83, no. A6, pp. 2637–2646, 1978.

[10] J.-C. Liou and N. L. Johnson, "Risks in space from orbiting debris," *Science*, vol. 311, no. 5759, pp. 340–341, 2006.

[11] J.-C. Liou and N. L. Johnson, "Instability of the present leo satellite populations," *Advances in Space Research*, vol. 41, no. 7, pp. 1046–1053, 2008.

[12] J.-C. Liou, "An active debris removal parametric study for leo environment remediation," *Advances in Space Research*, vol. 47, no. 11, pp. 1865–1876, 2011.

[13] J. S. Hudson and D. Kolosa, "Versatile on-orbit servicing mission design in geosynchronous earth orbit," *Journal of Spacecraft and Rockets*, pp. 1–7, 2020.

[14] W. Xu, B. Liang, B. Li, and Y. Xu, "A universal on-orbit servicing system used in the geostationary orbit," *Advances in Space Research*, vol. 48, no. 1, pp. 95–119, 2011.

[15] W. Xu, B. Liang, D. Gao, and Y. Xu, "A space robotic system used for on-orbit servicing in the geostationary orbit," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 4089–4094, IEEE, 2010.

[16] Y. Qiu, B. Guo, L. Xue, B. Liang, and C. Li, "On-orbit servicing to geo satellite using dual arm free-flying space robot," in *2009 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pp. 2451–2456, IEEE, 2009.

[17] Z. Zhao, J. Zhang, H.-y. Li, and J.-y. Zhou, "Leo cooperative multi-spacecraft refueling mission optimization considering j2 perturbation and target's surplus propellant constraint," *Advances in Space Research*, vol. 59, no. 1, pp. 252–262, 2017.

[18] J. Yu, X.-q. Chen, and L.-h. Chen, "Optimal planning of leo active debris removal based on hybrid optimal control theory," *Advances in Space Research*, vol. 55, no. 11, pp. 2628–2640, 2015.

[19] V. Braun, A. Lüpken, S. Flegel, J. Gelhaus, M. Möckel, C. Kebschull, C. Wiedemann, and P. Vörsmann, "Active debris removal of multiple priority targets," *Advances in Space Research*, vol. 51, no. 9, pp. 1638–1648, 2013.

[20] J. Yang, Y. H. Hu, Y. Liu, and Q. Pan, "A maximal-reward preliminary planning for multi-debris active removal mission in leo with a greedy heuristic method," *Acta Astronautica*, vol. 149, pp. 123–142, 2018.

[21] W. Fehse, *Automated rendezvous and docking of spacecraft*, vol. 16. Cambridge university press, 2003.

[22] A. Shirazi, J. Ceberio, and J. A. Lozano, "Spacecraft trajectory optimization: A review of models, objectives, approaches and solutions," *Prog. Aerosp. Sci.*, vol. 102, pp. 76–98, 2018.

[23] J. T. Betts, "Survey of numerical methods for trajectory optimization," *J. Guid. Control Dyn.*, vol. 21, no. 2, pp. 193–207, 1998.

[24] B. A. Conway, "A survey of methods available for the numerical optimization of continuous dynamic systems," *Journal of Optimization Theory and Applications*, vol. 152, no. 2, pp. 271–306, 2012.

[25] J. A. Kechichian, "The algorithm of the two-impulse time-fixed noncoplanar rendezvous with drag and oblateness effects," *J. Astronaut. Sci.*, pp. 47–64, 1998.

[26] G. Zhang, D. Mortari, and D. Zhou, "Constrained multiple-revolution lambert's problem," *J. Guid. Control Dyn.*, vol. 33, no. 6, pp. 1779–1786, 2010.

[27] W. Hohmann, *The Attainability of Heavenly Bodies*. NASA technical translation, National Aeronautics and Space Administration, 1960.

[28] S. Samsam and R. Chhabra, "Multi-impulse smooth trajectory design for long-range rendezvous with an orbiting target using multi-objective non-dominated sorting genetic algorithm," *Aerospace Science and Technology*, vol. 120, p. 107285, 2022.

[29] A. Shakouri, M. Kiani, and S. H. Pourtakdoust, "A new shape-based multiple-impulse strategy for coplanar orbital maneuvers," *Acta Astronaut.*, vol. 161, pp. 200 – 208, 2019.

[30] E. Taheri and O. Abdelkhalik, "Initial three-dimensional low-thrust trajectory design," *Adv. Space Res.*, vol. 57, no. 3, pp. 889–903, 2016.

[31] B. A. Conway, *Spacecraft Trajectory Optimization*, ch. 1.2. Cambridge Aerospace Series, Cambridge University Press, 2010.

[32] J. Robinson, "On the hamiltonian game (a traveling salesman problem)," tech. rep., RAND PROJECT AIR FORCE ARLINGTON VA, 1949.

[33] Ö. Gürtuna and J. Trépanier, "On-orbit satellite servicing: a space-based vehicle routing problem," in *Operations Research in Space and Air*, pp. 123–141, Springer, 2003.

[34] H. Shen and P. Tsiotras, "Optimal scheduling for servicing multiple satellites in a circular constellation," in *AIAA/AAS Astrodynamics Specialist Conference and Exhibit*, p. 4907, 2002.

[35] X.-q. Chen and J. Yu, "Optimal mission planning of geo on-orbit refueling in mixed strategy," *Acta Astronautica*, vol. 133, pp. 63–72, 2017.

[36] A. Dutta and P. Tsiotras, "Asynchronous optimal mixed p2p satellite refueling strategies," *The Journal of the Astronautical Sciences*, vol. 54, no. 3-4, pp. 543–565, 2006.

[37] J. Yu, Y.-g. Yu, J.-t. Huang, X.-q. Chen, and H.-y. Liu, "Optimal scheduling of geo on-orbit refuelling with uncertain object satellites," in *MATEC Web of Conferences*, vol. 114, p. 03001, EDP Sciences, 2017.

[38] R. Lampariello and G. Hirzinger, "Generating feasible trajectories for autonomous on-orbit grasping of spinning debris in a useful time," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5652–5659, IEEE, 2013.

[39] P. Huang, Y. Xu, and B. Liang, "Tracking trajectory planning of space manipulator for capturing operation," *International Journal of Advanced Robotic Systems*, vol. 3, no. 3, p. 31, 2006.

[40] P. Evangelos and D. Steven, "On the nature of control algorithms for free-floating space manipulators," *Robotics and Automation, IEEE Transactions on*, vol. 7, pp. 750–758, 1991.

[41] I. Tortopidis and E. Papadopoulos, "Point-to-point planning: methodologies for underactuated space robots," in *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, pp. 3861–3866, IEEE, 2006.

[42] D. N. Dimitrov, *Dynamics and control of space manipulators during a satellite capturing operation*. PhD thesis, , 2006.

[43] A. Ellery, "An engineering approach to the dynamic control of space robotic on-orbit servicers," *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, vol. 218, no. 2, pp. 79–98, 2004.

[44] B. M. Moghaddam and R. Chhabra, "On the guidance, navigation and control of in-orbit space robotic missions: A survey and prospective vision," *Acta Astronaut.*, vol. 184, pp. 70–100, 2021.

[45] R. M. Murray, Z. Li, S. S. Sastry, and S. S. Sastry, *A mathematical introduction to robotic manipulation*. CRC press, 1994.

[46] F. C. Park, J. E. Bobrow, and S. R. Ploen, "A lie group formulation of robot dynamics," *The International journal of robotics research*, vol. 14, no. 6, pp. 609–618, 1995.

[47] S. Stramigioli, B. Maschke, and C. Bidard, "On the geometry of rigid-body motions: The relation between lie groups and screws," *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, vol. 216, no. 1, pp. 13–23, 2002.

[48] A. Perez-Gracia and J. M. McCarthy, "Kinematic synthesis of spatial serial chains using clifford algebra exponentials," *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, vol. 220, no. 7, pp. 953–968, 2006.

[49] A. Muller and P. Maisser, "A lie-group formulation of kinematics and dynamics of constrained mbs and its application to analytical mechanics," *Multibody system dynamics*, vol. 9, no. 4, pp. 311–352, 2003.

[50] R. Chhabra and M. R. Emami, "A generalized exponential formula for forward and differential kinematics of open-chain multi-body systems," *Mechanism and Machine Theory*, vol. 73, pp. 61–75, 2014.

[51] R. Chhabra and M. R. Emami, "Symplectic reduction of holonomic open-chain multi-body systems with constant momentum," *Journal of Geometry and Physics*, vol. 89, pp. 82–110, 2015.

[52] S. Stramigioli, B. Maschke, and C. Bidard, "A hamiltonian formulation of the dynamics of spatial mechanisms using lie groups and screw theory," in *Proc. Symposium*

Commemorating the Legacy, Work and Life of Sir RS Ball, J. Duffy and H. Lipkin organizers, 2000.

[53] K. Yoshida, "Space robot dynamics and control: to orbit, from orbit, and future," in *Robotics Research*, pp. 449–456, Springer, 2000.

[54] T. Rybus, K. Seweryn, and J. Z. Sasiadek, "Trajectory optimization of space manipulator with non-zero angular momentum during orbital capture maneuver," in *AIAA Guidance, Navigation, and Control Conference*, p. 0885, 2016.

[55] P. Huang, M. Wang, Z. Meng, F. Zhang, Z. Liu, and H. Chang, "Reconfigurable spacecraft attitude takeover control in post-capture of target by space manipulators," *Journal of the Franklin Institute*, vol. 353, no. 9, pp. 1985–2008, 2016.

[56] M. Al-Isawi and J. Z. Sasiadek, "Guidance and control of a robot capturing an uncooperative space target," *Journal of Intelligent & Robotic Systems*, vol. 93, no. 3, pp. 713–721, 2019.

[57] H. Wang, "On adaptive inverse dynamics for free-floating space manipulators," *Robotics and Autonomous Systems*, vol. 59, no. 10, pp. 782–788, 2011.

[58] S. Ulrich and J. Z. Sasiadek, "On the simple adaptive control of flexible-joint space manipulators with uncertainties," in *Aerospace Robotics II*, pp. 13–23, Springer, 2015.

[59] Z. Vafa and S. Dubowsky, "On the dynamics of manipulators in space using the virtual manipulator approach," in *Proceedings. 1987 IEEE International Conference on Robotics and Automation*, vol. 4, pp. 579–585, IEEE, 1987.

[60] S. Dubowsky and Z. Vafa, "A virtual manipulator model for space robotic systems," in *Jet Propulsion Lab., California Inst. of Tech., Proceedings of the Workshop on Space Telerobotics, Volume 3*, 1987.

[61] Z. Vafa and S. Dubowsky, "The kinematics and dynamics of space manipulators: The virtual manipulator approach," *The International Journal of Robotics Research*, vol. 9, no. 4, pp. 3–21, 1990.

[62] O. Parlaktuna and M. Ozkan, "Adaptive control of free-floating space manipulators using dynamically equivalent manipulator model," *Robotics and Autonomous Systems*, vol. 46, no. 3, pp. 185–193, 2004.

[63] E. G. Papadopoulos, *On the dynamics and control of space manipulators*. PhD thesis, Massachusetts Institute of Technology, 1990.

[64] M. A. Torres and S. Dubowsky, "Minimizing spacecraft attitude disturbances in space manipulator systems," *Journal of guidance, control, and dynamics*, vol. 15, no. 4, pp. 1010–1017, 1992.

[65] S. Dubowsky and M. A. Torres, "Path planning for space manipulators to minimize spacecraft attitude disturbances," in *Proceedings of IEEE International Conference on Robotics and Automation*, vol. 3, pp. 2522–2528, Citeseer, 1991.

[66] M. A. Torres and S. Dubowsky, "Path-planning for elastically constrained space manipulator systems," in *[1993] Proceedings IEEE International Conference on Robotics and Automation*, pp. 812–817, IEEE, 1993.

[67] Y. Umetani and K. Yoshida, "Continuous path control of space manipulators mounted on omv," *Acta Astronautica*, vol. 15, no. 12, pp. 981–986, 1987.

[68] Y. Umetani, K. Yoshida, *et al.*, "Resolved motion rate control of space manipulators with generalized jacobian matrix," *IEEE Transactions on robotics and automation*, vol. 5, no. 3, pp. 303–314, 1989.

[69] K. Yoshida, D. Dimitrov, and H. Nakanishi, "On the capture of tumbling satellite by a space robot," in *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 4127–4132, IEEE, 2006.

[70] K. Yoshida, "Engineering test satellite vii flight experiments for space robot dynamics and control: theories on laboratory test beds ten years ago, now in orbit," *The International Journal of Robotics Research*, vol. 22, no. 5, pp. 321–335, 2003.

[71] R. Chhabra and M. R. Emami, "A unified approach to input-output linearization and concurrent control of underactuated open-chain multi-body systems with holonomic and nonholonomic constraints," *Journal of dynamical and control systems*, vol. 22, no. 1, pp. 129–168, 2016.

[72] T. Barcinski, J. Lisowski, T. Rybus, and K. Seweryn, "Controlled zero dynamics feedback linearization with application to free-floating redundant orbital manipulator," in *2013 American Control Conference*, pp. 1834–1839, IEEE, 2013.

[73] F. Aghili, "Coordination control of a free-flying manipulator and its base attitude to capture and detumble a noncooperative satellite," in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2365–2372, IEEE, 2009.

[74] K. Nanos and E. G. Papadopoulos, "On the dynamics and control of free-floating space manipulator systems in the presence of angular momentum," *Frontiers in Robotics and AI*, vol. 4, p. 26, 2017.

[75] F. Bullo and R. M. Murray, "Tracking for fully actuated mechanical systems: a geometric framework," *Automatica*, vol. 35, no. 1, pp. 17–34, 1999.

[76] P. Malaviarachchi, T. Reedman, A. Allen, and D. Sinclair, "A small satellite concept for on-orbit servicing of spacecraft," 2003.

[77] B. Chamot and M. Richard, "Mission and system architecture design for active removal of rocket bodies in low earth orbit," *Master's thesis, Massachusetts Institute of Technology, US*, 2012.

[78] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[79] H. Sze and R. Chhabra, "Online Trajectory Generation for Uncertain Manipulators Capturing Moving Targets using Transfer Learning in DDPG," *IEEE Robotics and Automation Letters*, Submitted, 2021.

[80] D. E. Koditschek, "The application of total energy as a lyapunov function for mechanical control systems," *Contemporary mathematics*, vol. 97, p. 131, 1989.

[81] J. Jost and J. Jost, *Riemannian geometry and geometric analysis*, vol. 42005. Springer, 2008.

[82] Y. Nakamura and H. Hanafusa, "Inverse kinematic solutions with singularity robustness for robot manipulator control," 1986.

[83] C. W. Wampler, "Manipulator inverse kinematic solutions based on vector formulations and damped least-squares methods," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 16, no. 1, pp. 93–101, 1986.

[84] A. S. Deo and I. D. Walker, "Overview of damped least-squares methods for inverse kinematics of robot manipulators," *Journal of Intelligent and Robotic Systems*, vol. 14, no. 1, pp. 43–68, 1995.

[85] T. Yoshikawa, "Manipulability of robotic mechanisms," *The international journal of Robotics Research*, vol. 4, no. 2, pp. 3–9, 1985.

[86] L. Kelmar and P. K. Khosla, "Automatic generation of kinematics for a reconfigurable modular manipulator system," in *Proceedings. 1988 IEEE international conference on robotics and automation*, pp. 663–668, IEEE, 1988.

[87] S. K. Chan and P. D. Lawrence, "General inverse kinematics with the error damped pseudoinverse," in *Proceedings. 1988 IEEE international conference on robotics and automation*, pp. 834–839, IEEE, 1988.

[88] A. A. Maciejewski and C. A. Klein, "Numerical filtering for the operation of robotic manipulators through kinematically singular configurations," *Journal of Robotic systems*, vol. 5, no. 6, pp. 527–552, 1988.

[89] R. V. Mayorga, A. K. Wong, and N. Milano, "A fast damped least-squares solution to manipulator inverse kinematics and singularities prevention," in *Proceedings of the IEEE/RSJ international conference on intelligent robots and systems*, vol. 2, pp. 1177–1184, IEEE, 1992.

[90] R. M. Murray, Z. Li, and S. S. Sastry, *A mathematical introduction to robotic manipulation*. CRC press, 2017.