



NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

**DYNAMIC MODELING AND SIMULATION
OF A FLOATING SPACECRAFT SIMULATOR UTILIZING
A KINOVA 7DOF ROBOTIC ARM**

by

Matthew M. Tungett

June 2023

Thesis Advisor:
Second Reader:

Jennifer Hudson
Marcello Romano

Approved for public release. Distribution is unlimited.

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			<i>Form Approved OMB No. 0704-0188</i>	
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington, DC, 20503.				
1. AGENCY USE ONLY (Leave blank)		2. REPORT DATE June 2023	3. REPORT TYPE AND DATES COVERED Master's thesis	
4. TITLE AND SUBTITLE DYNAMIC MODELING AND SIMULATION OF A FLOATING SPACECRAFT SIMULATOR UTILIZING A KINOVA 7DOF ROBOTIC ARM			5. FUNDING NUMBERS	
6. AUTHOR(S) Matthew M. Tungett				
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000			8. PERFORMING ORGANIZATION REPORT NUMBER	
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A			10. SPONSORING / MONITORING AGENCY REPORT NUMBER	
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.				
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release. Distribution is unlimited.			12b. DISTRIBUTION CODE A	
13. ABSTRACT (maximum 200 words) This thesis presents the mathematical modeling of a novel floating spacecraft simulator (FSS) utilizing a Kinova 7-degrees-of-freedom (DoF) robotic manipulator. The FSS system modeled in this thesis enables research into on-orbit servicing platforms, a space technology capable of providing satellite refueling, repair, and space debris mitigation. To accurately represent the FSS system, a multi-body dynamics (MBD) model was employed. The initial phases of this research focused on modeling the robotic manipulator, which was later validated through simulations and physical hardware-in-the-loop testing. These tests verified that the developed robotic manipulator MBD model effectively captured the real-world dynamics of the Kinova arm. The MBD model for the robotic manipulator was then upgraded to incorporate the floating base of the FSS system. This expanded model underwent additional simulations, comparing its performance against hardware-in-the-loop testing of the real FSS system. The results of this testing confirmed that the established MBD model accurately modeled the observed performance of the existing FSS system. The refined MBD model of the FSS system presented in this thesis holds significant potential for enabling the development of future control algorithms and facilitating advanced simulations of the FSS system.				
14. SUBJECT TERMS space-based robotics, on-orbit servicing, Kinova 7 DoF Robotic Manipulator, floating spacecraft simulator, Simulink, MATLAB, virtual simulation, hardware-in-the-loop testing			15. NUMBER OF PAGES 155	
			16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UU	

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. Z39-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release. Distribution is unlimited.

**DYNAMIC MODELING AND SIMULATION OF A FLOATING SPACECRAFT
SIMULATOR UTILIZING A KINOVA 7DOF ROBOTIC ARM**

Matthew M. Tungett
Ensign, United States Navy
BS, United States Naval Academy, 2022

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN ASTRONAUTICAL ENGINEERING

from the

**NAVAL POSTGRADUATE SCHOOL
June 2023**

Approved by: Jennifer Hudson
Advisor

Marcello Romano
Second Reader

Brian S. Bingham
Chair, Department of Mechanical and Aerospace Engineering

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

This thesis presents the mathematical modeling of a novel floating spacecraft simulator (FSS) utilizing a Kinova 7-degree-of-freedom (DoF) robotic manipulator. The FSS system modeled in this thesis enables research into on-orbit servicing platforms, a space technology capable of providing satellite refueling, repair, and space debris mitigation. To accurately represent the FSS system, a multi-body dynamics (MBD) model was employed. The initial phases of this research focused on modeling the robotic manipulator, which was later validated through simulations and physical hardware-in-the-loop testing. These tests verified that the developed robotic manipulator MBD model effectively captured the real-world dynamics of the Kinova arm. The MBD model for the robotic manipulator was then upgraded to incorporate the floating base of the FSS system. This expanded model underwent additional simulations, comparing its performance against hardware-in-the-loop testing of the real FSS system. The results of this testing confirmed that the established MBD model accurately modeled the observed performance of the existing FSS system. The refined MBD model of the FSS system presented in this thesis holds significant potential for enabling the development of future control algorithms and facilitating advanced simulations of the FSS system.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

I.	INTRODUCTION.....	1
A.	RESEARCH MOTIVATION	1
B.	HISTORY OF ON-ORBIT SERVICING.....	2
C.	FUTURE ON-ORBIT SERVICERS	5
D.	RESEARCHING ON-ORBIT SERVICING.....	7
E.	FLOATING SPACECRAFT SIMULATORS	10
F.	RESEARCH FOCUS AND OUTLINE.....	13
II.	FSS SYSTEM OVERVIEW	15
A.	SUMMARY	15
B.	SYSTEM MODEL	16
C.	FSS BASE	18
D.	KINOVA ARM.....	20
E.	ROBOTIQ GRIPPER	24
III.	MODELING AND SIMULATION OF THE KINOVA ARM.....	27
A.	ROBOTIC MANIPULATOR MATHEMATICAL REPRESENTATIONS	27
	1. Denavit-Hartenberg Parameters	27
	2. Unified Robot Description Format.....	27
	3. CAD-Based Models.....	28
	4. Multi-Body Dynamics Model.....	28
B.	KINOVA ARM MODEL DEVELOPMENT	28
C.	SIMULATION SOFTWARE	32
D.	SIMULATION SETUP	33
E.	SIMULATED KINOVA ARM TESTS.....	35
F.	SIMULATED KINOVA ARM RESULTS	37
IV.	KINOVA ARM HARDWARE-IN-THE-LOOP TESTING	41
A.	REAL-LIFE KINOVA ARM.....	41
B.	COMMAND GENERATION AND SENSOR RECORDING	42
C.	HARDWARE-IN-THE-LOOP TESTING	44
D.	HARDWARE-IN-THE-LOOP RESULTS.....	48

V.	MODELING AND SIMULATION OF THE FULL FSS SYSTEM	53
A.	FSS BASE MODEL DEVELOPMENT	53
B.	FSS BASE SIMULATION TESTS	55
C.	FSS BASE SIMULATION RESULTS.....	56
D.	FULL FSS SYSTEM MODEL DEVELOPMENT	58
E.	FULL FSS SYSTEM SIMULATION TESTS.....	61
F.	FULL FSS SYSTEM SIMULATION RESULTS	63
VI.	FULL FSS SYSTEM HARDWARE-IN-THE-LOOP TESTING	67
A.	REAL-LIFE FULL FSS SYSTEM.....	67
B.	COMMAND GENERATION, SENSOR RECORDING, AND TRACKING.....	68
C.	FSS HARDWARE-IN-THE-LOOP TESTS.....	72
D.	FSS HARDWARE-IN-THE-LOOP RESULTS	75
VII.	CONCLUSION	77
A.	SUMMARY OF RESULTS	77
B.	FUTURE WORK	78
1.	MBD Model Refinement and Upgrading.....	78
2.	FSS Control Algorithm Development	78
3.	Demonstrate Approach and Docking Maneuvering.....	78
	APPENDIX A. KINOVA ARM URDF FILE.....	81
	APPENDIX B. KINOVA ARM WITH TEST OBJECT URDF FILE	87
	APPENDIX C. KINOVA ARM URDF AND MBD MODEL SIMULATION SCRIPT	95
	APPENDIX D. KINOVA ARM COMMAND GENERATION AND SENSOR RECORDING SCRIPT	99
	APPENDIX E. REAL KINOVA ARM TESTING VS. URDF MODEL SIMULATION SCRIPT.....	101
	APPENDIX F. FSS BASE DYNAMICAL MODEL SCRIPT	105

APPENDIX G. FSS BASE MBD MODEL VS. DYNAMICAL MODEL SIMULATION SCRIPT	107
APPENDIX H. FULL FSS SYSTEM SIMULATION SCRIPT	111
APPENDIX I. VIDEO FILE TO IMAGE DATASTORE SCRIPT	113
APPENDIX J. IMAGE DISTORTION CORRECTION SCRIPT	115
APPENDIX K. FSS POSITION AND ATTITUDE ESTIMATION SCRIPT	117
APPENDIX L. MEASURED FSS DATA INTERPOLATION SCRIPT	119
APPENDIX M. MOVIE GENERATION SCRIPT	121
APPENDIX N. REAL FSS TESTING VS. SIMULATED MBD MODEL SIMULATION SCRIPT	123
LIST OF REFERENCES	127
INITIAL DISTRIBUTION LIST	133

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF FIGURES

Figure 1.	Active satellites from 1957–2022. Adapted from [2], [5].	1
Figure 2.	Challenger conducting repairs of the SMM satellite during STS-41-C. Source: [11].	3
Figure 3.	Space Shuttle Endeavour grappling the HST during STS-61. Source: [16].	4
Figure 4.	Artist’s depiction of the ASTRO (left) servicing NextSat (right). Source: [17].	5
Figure 5.	Artist’s concept of OSAM-1. Source: [22].	6
Figure 6.	Mock ISS simulator at NASA’s Neutral Buoyancy Laboratory. Source: [24].	8
Figure 7.	Phases of parabolic aircraft flight. Source: [25].	8
Figure 8.	FSS testing on the NPS-SRL POSEIDYN air bearing table. Source: [28].	10
Figure 9.	A SPHERES satellite mounted on an air-bearing at the MIT Space Systems Laboratory. Source: [31].	11
Figure 10.	ASTROBEE testing at the NASA Ames Research Center. Source: [32].	12
Figure 11.	Spacecraft Proximity Operations simulation at the NPS-SRL. Source: [33].	13
Figure 12.	CAD rendering of the complete FSS system.	15
Figure 13.	Use case diagram for the FSS system.	16
Figure 14.	Logical structure for the FSS system.	17
Figure 15.	Labeled components of the FSS base.	19
Figure 16.	FSS base coordinate system and COM location.	20
Figure 17.	Kinova 7 DoF robot manipulator. Source: [36].	21
Figure 18.	Absolute workspace of Kinova 7 DoF robotic manipulator. Source: [35].	22

Figure 19.	Schematic of the Kinova 7 DoF robot manipulator. Adapted from [35].	23
Figure 20.	Robotiq 2F-140 adaptive gripper. Source: [41].	25
Figure 21.	Robotiq adaptive gripper schematic. Adapted from [39].	26
Figure 22.	Visualization of the Kinova arm defined by the URDF model.	29
Figure 23.	MBD block definition for the Kinova shoulder link.	30
Figure 24.	The developed MBD model for the Kinova arm.	31
Figure 25.	Visualization of the Kinova arm defined by the MBD model.	32
Figure 26.	Visualization of the Kinova arm simulated test maneuvers.	36
Figure 27.	Calculated torques from the simulated Kinova arm test maneuvers.	39
Figure 28.	Kinova arm pictured in the NPS-SRL.	41
Figure 29.	Sensor recording gap during Kinova arm maneuver.	43
Figure 30.	Kinova arm grasping the aluminum bar test object.	44
Figure 31.	Visualization of the real-life Kinova arm test maneuvers.	45
Figure 32.	Visualization of the real-life Kinova arm test maneuvers utilizing the test object.	47
Figure 33.	Calculated torques vs. measured real-life Kinova arm torques.	48
Figure 34.	Calculated vs. measured torques for tests utilizing the test object.	50
Figure 35.	The developed MBD model for the FSS base.	54
Figure 36.	Visualization of the FSS base defined by the MBD model.	54
Figure 37.	Simulated FSS base MBD vs. propagated dynamical model results.	57
Figure 38.	The developed MBD model for the full FSS system.	60
Figure 39.	Visualization of the FSS system defined by the MBD model.	61
Figure 40.	Visualization of the full FSS system simulation testing.	62
Figure 41.	Full FSS system simulation results.	64

Figure 42.	Full FSS system pictured in the NPS-SRL.	67
Figure 43.	Camera mount and GoPro used in testing.....	69
Figure 44.	Barrel distortion of a raw video frame.	70
Figure 45.	Corrected video frame.....	71
Figure 46.	Screen capture from a generated movie file.	72
Figure 47.	Visualization of the testing scenarios used with the real and simulated FSS system.	73
Figure 48.	Measured vs. calculated FSS system testing results.	75

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

Table 1.	Kinova Gen3 7 DoF robotic manipulator general specifications. Adapted from [37].....	22
Table 2.	Limits and capabilities of the robotic manipulator. Adapted from [35], [38].	23
Table 3.	Robotiq adaptive gripper specifications. Adapted from [40].....	25
Table 4.	Simulated Kinova arm test maneuvers.	37
Table 5.	Mean average error between the URDF and MBD simulation tests.	39
Table 6.	Isolated real-life Kinova arm test maneuvers.	46
Table 7.	Test object real-life Kinova arm test maneuvers.	47
Table 8.	Mean average error between the calculated and measured real-life Kinova arm torques.....	49
Table 9.	Mean average error between the calculated and measured real-life torques for tests utilizing the test object.	51
Table 10.	Isolated FSS base simulation testing parameters.	56
Table 11.	Absolute error at t_f for isolated FSS base simulation tests.....	58
Table 12.	Full FSS system simulation testing parameters.	63
Table 13.	Testing parameters for the real and simulated FSS system.	74
Table 14.	Absolute error at t_f between the real FSS system and the MBD model simulation tests.....	76

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS AND ABBREVIATIONS

ASTRO	Autonomous Space Transport Robotic Operations
CAD	computer-aided design
CCS	Cartesian coordinate system
COM	center of mass
COTS	commercial-off-the-shelf
DAE	digital asset exchange
DARPA	Defense Advanced Research Projects Agency
DH	Denavit-Hartenberg
DoF	degrees-of-freedom
FCC	Federal Communications Commission
FOV	field of view
FSS	floating spacecraft simulator
GAO	Government Accountability Office
GEO	geosynchronous Earth orbit
HST	Hubble Space Telescope
ISS	International Space Station
LDEF	Long Duration Exposure Facility
LEO	low Earth orbit
MAE	mean absolute error
MATLAB	Matrix Laboratory
MBD	multi-body dynamics
MIT	Massachusetts Institute of Technology
NASA	National Aeronautics and Space Administration
NPS	Naval Postgraduate School
OOS	on-orbit servicing
OSAM	On-orbit Servicing, Assembly, and Manufacturing

POSEIDYN	Proximity Operation of Spacecraft Experimental Hardware-in-the-loop Dynamic Simulator
RMS	Remote Manipulator System
RNEA	recursive Newton-Euler algorithm
ROS	Robot Operating System
RPC	remote procedure call
RS GS	Robotic Servicing of Geosynchronous Satellite
SMM	Solar Maximum Mission
SPART	SPAcecraft Robotics Toolbox
SPHERES	Synchronized Position Hold Engage Re-Orient Experiment Satellites
SPIDER	Space Infrastructure Dexterous Robot
SRL	Spacecraft Robotics Laboratory
STL	stereolithography
SysML	systems modeling language
URDF	unified robot description file

ACKNOWLEDGMENTS

First, I would like to thank Dr. Jennifer Hudson for her guidance and support throughout my research. I appreciate your time mentoring me, answering my questions, and reviewing my work. Thank you for letting me be a part of the exciting research happening in the Spacecraft Robotics Laboratory. I also want to thank Dr. Marcello Romano for reviewing my thesis.

Next, I want to acknowledge my close friends Damien Stonhill, Sebastian Keefer, Jake Durfee, Haofeng Liu, Clay Johnson, and Adam Western. You have all positively touched my life in one way or another. I would not be where I am today without your friendship.

I also want to recognize my family. You are my support structure and the most important thing in my life. Grandpa Roy, you influenced me from a young age and led me to pursue an engineering education. Dad, Poppy, Uncle Dave, and Uncle Allan, you all bravely served our country, and I hope to carry on your legacy for the remainder of my military career. Amelia, you have always supported me, and I am so grateful to be your brother. Mom and Dad, I really cannot put my gratitude into words. Every day of my life, you have encouraged me to pursue my passions, take on responsibility, and work hard. I love you more than you will ever know.

McKennah, you are my rock. Through thick and thin, you have always been there with me. You have given me love, motivation, and support in more ways than I can count. I am where I am today because of you. I love you all the numbers.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. RESEARCH MOTIVATION

Currently, there is unprecedented growth occurring in the space industry. As of early 2023, more than 5,000 active satellites are operating around the Earth, nearly quadrupling the number of active satellites recorded for 2014 [1], [2]. Figure 1 visually depicts the growth rate of active satellites from 1957 to 2022. This rapid expansion of satellites is due in part to interests stemming from commercial industry and government entities [3], [4]. The commercial industry has focused on innovative technologies, including lowering launch costs and investing in large-scale satellite constellations [4]. Government entities have expanded policies and regulations, allowing increased access to space [3]. Countries, including the United States and China, continue to invest in space-based assets for national defense and security.

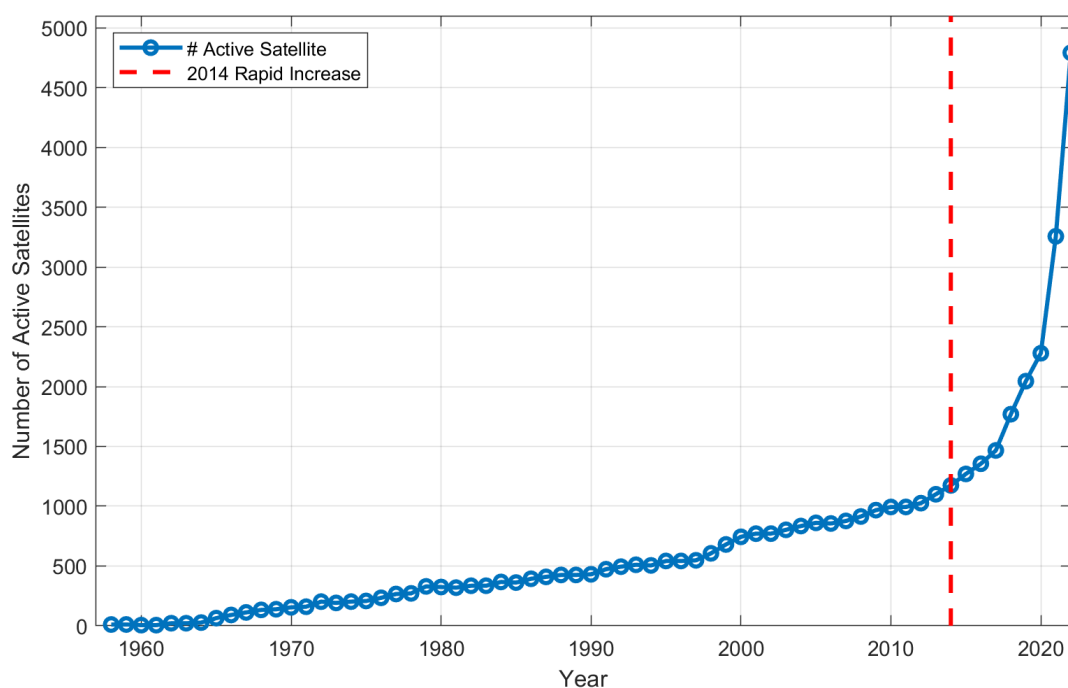


Figure 1. Active satellites from 1957–2022. Adapted from [2], [5].

Future outlooks for the space industry predict that space will only become more congested. A recent report filed by the United States Government Accountability Office (GAO) cites that up to 58,000 satellites may be launched by 2030 based on current Federal Communications Commission (FCC) filings [6]. The FCC filings are mainly for large-scale satellite constellations, driven by private companies interested in providing low-latency global internet coverage. These satellite constellations will only add to the prevalent issue of space debris. Recent estimates suggest there are around 500,000 space debris objects with 1–10 cm diameters, orbiting between 250–40,000 km [7]. Objects of these diameters are challenging to track and pose a danger to operational and defunct satellites. Collisions from these objects could further complicate the current congestion in space.

The expansion of the space industry and the challenges of space congestion highlight the increasing need for on-orbit servicing (OOS) capabilities. OOS affords the ability to conduct repairs, refuel spacecraft, alter orbital characteristics, and mitigate space debris [8], [9]. A limited number of dedicated space platforms currently provide this capability for satellite operators, and more research is needed in advancing OOS technology. This thesis will research the dynamic interactions of OSS platforms by mathematically modeling a floating spacecraft simulator (FSS) utilizing a robotic manipulator.

B. HISTORY OF ON-ORBIT SERVICING

OOS has a history tracing its roots back to NASA space station Skylab [10]. While launching into space, Skylab suffered damage during launch, which severed a micrometeoroid shield and prevented the deployment of Skylab's solar arrays [10]. NASA delayed the launch of the first crew mission to Skylab by ten days to rapidly develop a plan to repair the space station [10]. NASA's plan consisted of deploying a custom parasol and freeing the jammed solar array [10]. On May 25, 1973, the first crew launched to Skylab and immediately began repair work on the space station [10]. After struggling with the repairs, the astronauts successfully freed the solar array and stabilized the internal

temperature of the Skylab with the parasol [10]. Though unplanned, the space repair of Skylab represented the first repair of spacecraft orbiting the Earth.

In 1984, the NASA Space Shuttle Challenger conducted the first planned servicing mission with STS-41-C [11], [12]. This mission involved deploying the Long Duration Exposure Facility (LDEF) and repairing the Solar Maximum Mission (SMM) satellite [13]. The SMM repair component of the mission tasked astronauts with replacing the attitude control system and primary electronics that had failed prematurely on the SMM [11]–[13]. To service SMM, Challenger grappled onto the satellite via the Canadian-built Remote Manipulator System (RMS) [11], [12]. Despite initial issues with the repair mission, the astronauts completed the SMM repairs, proving the feasibility of orbital servicing. Figure 2 shows an image taken during the SMM repairs on STS-41-C. The success of STS-41-C paved the way for future servicing missions such as STS-51-A, responsible for retrieving two malfunctioning satellites, and STS-49, which conducted the Intelsat VI F-3 communication satellite repairs [14], [15].

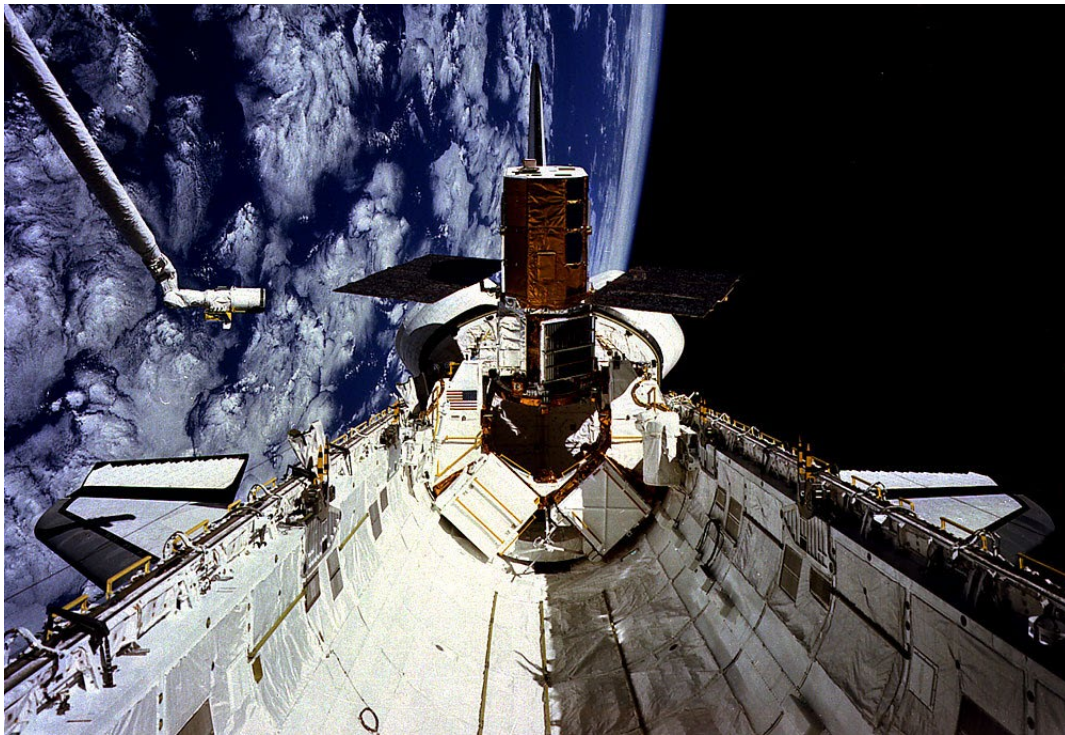


Figure 2. Challenger conducting repairs of the SMM satellite during STS-41-C. Source: [11].

Arguably, the most famous Space Shuttle servicing mission came in 1993 when STS-61 corrected the flawed optics of the Hubble Space Telescope (HST) [16]. The HST quickly became a publicized topic after initial images sent to Earth showed an issue with the telescope's primary mirror [16]. Rather than abandon the HST, NASA devised a repair mission to correct the optical distortion of the HST's primary mirror [16]. The repair mission required an 11-day mission of six spacewalks totaling 35 hours and 25 minutes [16]. Figure 3 is an image taken during the STS-61 HST repair mission. Four more missions were completed in addition to STS-61 to conduct repairs to the HST [16]. The success of the HST repairs and other Space Shuttle servicing missions demonstrated the versatility of OOS and its necessity for future space missions.



Figure 3. Space Shuttle Endeavour grappling the HST during STS-61.
Source: [16].

Limitations of the space shuttle and its retirement in 2011 highlighted the need for upgraded servicing capabilities, which led to the development of autonomous on-orbit servicers. In 2007, the U.S. Defense Advanced Research Projects Agency (DARPA) demonstrated the first autonomous satellite servicing mission utilizing the Orbital Express [17], [18]. This mission consisted of two spacecraft: the Autonomous Space

Transport Robotic Operations (ASTRO) servicing vehicle and client satellite named NextSat [17]. Within this mission, ASTRO demonstrated key capabilities of rendezvous, docking, refueling, and component replacements on the NextSat [17], [18]. Figure 4 shows an artist's interpretation of the Orbital Express servicing mission. The success of the Orbital Express servicing mission has spawned the creation of other autonomous servicing satellites, including the NASA Robotic Refueling Mission, the Northrop Grumman Mission Extension Vehicle, and the Astroscale ELSA-d satellites [19]–[21]. These platforms have performed autonomous rendezvous, satellite capture, repositing of client satellites, and even some orbital debris removal.



Figure 4. Artist's depiction of the ASTRO (left) servicing NextSat (right).
Source: [17].

C. FUTURE ON-ORBIT SERVICERS

Future work for OOS includes the NASA On-orbit Servicing, Assembly, and Manufacturing (OSAM) 1 mission [22]. The OSAM-1 servicer, formerly Restore-L, will demonstrate key technologies needed to extend a satellite's lifespan, including spacecraft

not designed to be serviced [22]. OSAM-1 will utilize the Space Infrastructure Dexterous Robot (SPIDER), a lightweight robotic arm capable of extending up to 16 feet [22]. SPIDER will enable OSAM-1 to grapple with client satellites, including repair work and refueling [22]. Landsat-7 is the first satellite to be serviced by OSAM-1, scheduled for no earlier than 2025 [22]. This mission will involve a complex refueling of Landsat-7, a first for autonomous OOS [22]. Completing this mission will enable future NASA missions to incorporate OOS as a mission planning component. Figure 5 shows an artist's interpretation of the OSAM-1 mission involving the Landsat-7 satellite.



Figure 5. Artist's concept of OSAM-1. Source: [22].

Another on-orbit servicer currently in development is the DARPA Robotic Servicing of Geosynchronous Satellite (RSGS) [23]. The RSGS program is a joint effort between DARPA and private companies, created to develop robotic servicing capabilities for geosynchronous Earth orbit (GEO) satellites [23]. DARPA seeks to demonstrate servicing operations of GEO satellites for varying orbital conditions for commercial and

U.S. government operators [23]. This platform will expand current satellite servicing platforms, which currently cannot operate in high-altitude orbits like GEO. The RSGS program plans to develop a servicer spacecraft capable of operating for several years, servicing multiple spacecraft during its lifespan.

D. RESEARCHING ON-ORBIT SERVICING

OOS research spans numerous domains, including rendezvous and docking, on-orbit refueling, robotic arm technology, and autonomous decision-making. A central focus across these research areas revolves around understanding the dynamic behavior and control characteristics of satellite servicing systems. Satellite servicers operate within a micro-gravity environment, posing a complex challenge for researchers due to the difficulty of replicating such conditions on Earth. Presently, two commonly employed methods for simulating micro-gravity are neutral buoyancy tanks and parabolic flights.

Neutral buoyancy tanks simulate microgravity by submerging hardware with enough ballast to make the hardware neutrally buoyant. This neutral buoyancy effectively eliminates the influences of gravity and thus simulates the conditions of a microgravity environment. Figure 6 shows an image of NASA's Neutral Buoyancy Laboratory. Parabolic flights achieve weightlessness by utilizing an aircraft that follows a parabolic trajectory. By following this trajectory, an aircraft eliminates the gravitational acceleration imposed on an object and allows for short periods of weightlessness. Figure 7 details the parabolic trajectory of an aircraft designed to simulate the microgravity environment. While useful in some cases, these types of testing environments are either cost prohibitive or difficult to control the testing environment.

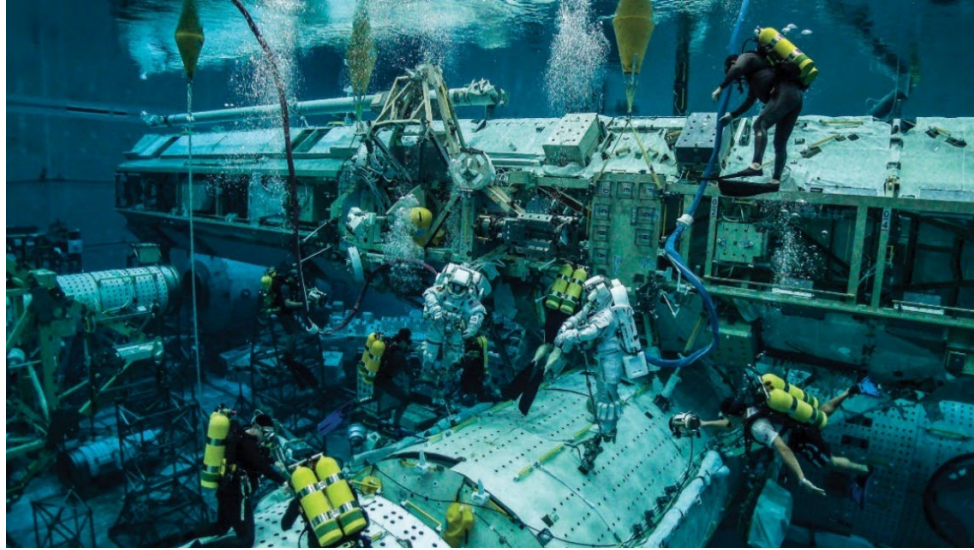


Figure 6. Mock ISS simulator at NASA's Neutral Buoyancy Laboratory.
Source: [24].

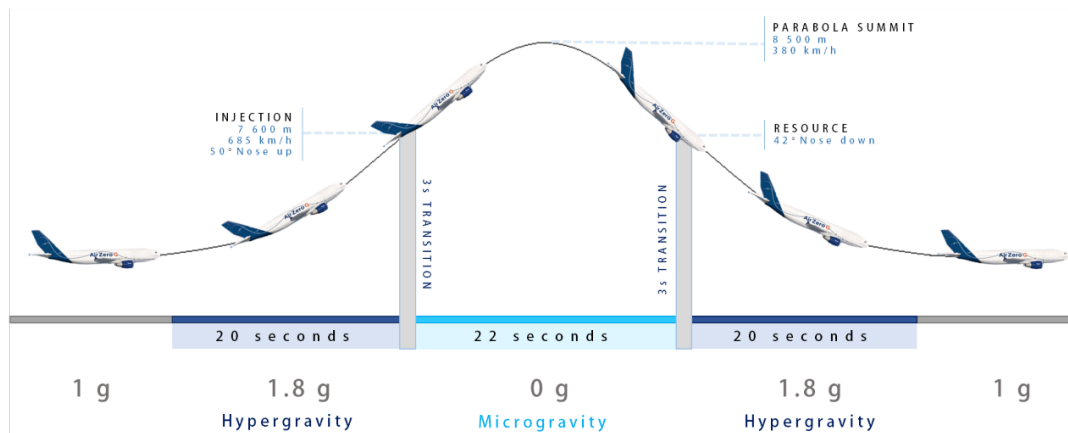


Figure 7. Phases of parabolic aircraft flight. Source: [25].

A more cost-effective method is to simulate the microgravity environment on a low-friction table utilizing a planar floating spacecraft simulator or FSS. In this testing system, a spacecraft simulator utilizes air-bearings to emulate the micro-gravity environment within a 2-D plane. The air-bearings produce a thin film of pressurized air between the bottom of the spacecraft simulator and a large flat surface called an air table. Typically, these air tables are constructed from a durable material capable of maintaining a smooth surface and are adjusted to be parallel with the ground to limit the components

of gravity on a simulator. This testing environment allows engineers to study a spacecraft's dynamic and control characteristics with minimal environmental disturbances. Additionally, this method is more cost-effective than a buoyancy tank or parabolic flight-testing environment.

One main disadvantage of the air-bearing system is that the FSS is limited to 2-D motion, whereas the microgravity environment of space is 3-D. The 2-D testing environment allows a spacecraft simulator to translate in two lateral axes and rotate about one vertical axis. Another disadvantage is that FSS's are subject to Earth's gravity and air resistance within an air-bearing testing environment. Earth's gravity can present challenges when utilizing an air-bearing table for space-optimized hardware. Unique testing configurations are often required to limit the gravitational acceleration of space-optimized hardware, including pulley systems, additional air-bearings, and structural reinforcement of space-optimized hardware.

Despite these disadvantages, an air table coupled with a spacecraft simulator can provide invaluable insights into the complex dynamics of a spacecraft. Several research facilities currently use air tables in their spacecraft research. Examples include the NASA Flat Floor Facility at the Johnson Space Center and the docking facility located at the University of Surrey in the United Kingdom [26], [27]. These testing facilities have researched areas involving robotics, control, navigation, hardware-in-the-loop testing, and rendezvous docking procedures on an FSS test platform.

The Naval Postgraduate School (NPS) Spacecraft Robotics Laboratory (SRL) utilizes a unique facility called the Proximity Operation of Spacecraft Experimental Hardware-in-the-loop Dynamic Simulator (POSEIDYN) [28]. Figure 8 shows an image of the POSEIDYN testbed. POSEIDYN is a 4x4 m smooth granite monolith with three adjustable pedestals [29]. These pedestals allow POSEIDYN to have a horizontal level accuracy of ± 0.01 deg [29]. In addition to the granite monolith, an advanced motion-capture system called Vicon is used to determine the position of a spacecraft simulator within the POSEIDYN operational area [29]. Vicon is composed of 10 overhead cameras utilizing an external computer [29]. The Vicon motion-capture system measures passive

markers on a spacecraft simulator, providing submillimeter-level accuracy at rates up to 100 Hz position estimates [29].

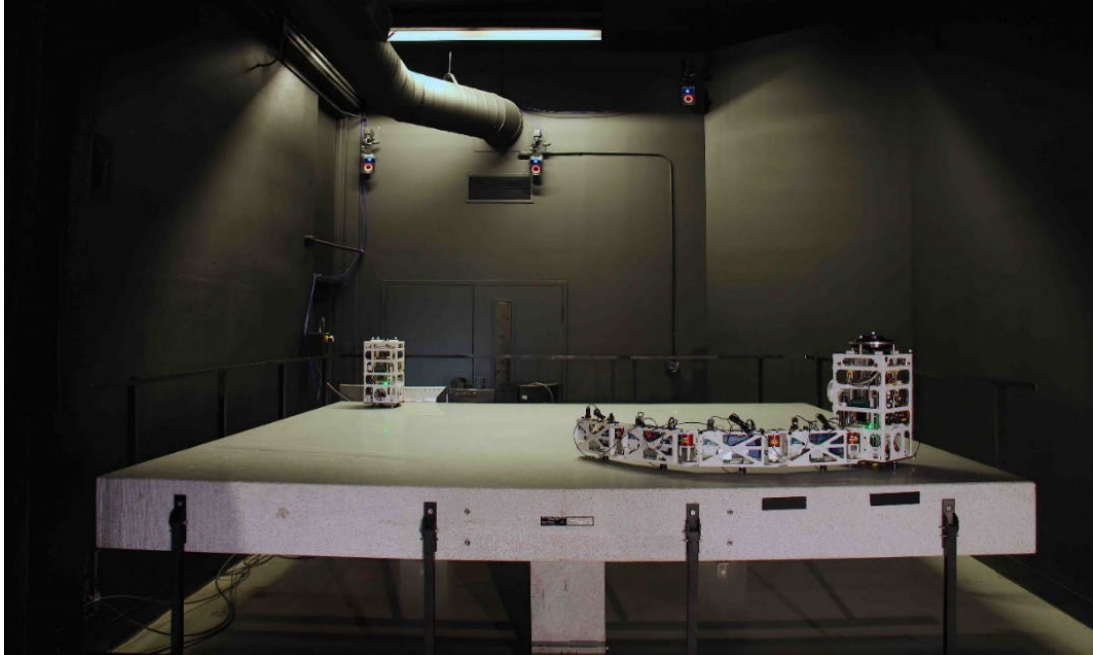


Figure 8. FSS testing on the NPS-SRL POSEIDYN air bearing table.
Source: [28].

E. FLOATING SPACECRAFT SIMULATORS

Since the 1960s, air-bearing tables have enabled researchers to simulate a wide range of FSS's. Past examples of FSS's have included the MIT-produced Synchronized Position Hold Engage Re-Orient Experiment Satellites (SPHERES) [30]. Constructed on an 18-sided polyhedron, SPHERES was designed as an experimental testbed to demonstrate the dynamics of a formation flight configuration for satellite systems [30]. SPHERES has operated on the International Space Station (ISS) since 2006 and utilizes small CO₂ gas thrusters for maneuvering [30]. An air-bearing testing apparatus helped test SPHERES at the MIT Space Systems Laboratory [31]. Figure 9 shows the testing apparatus used to test SPHERES on Earth.

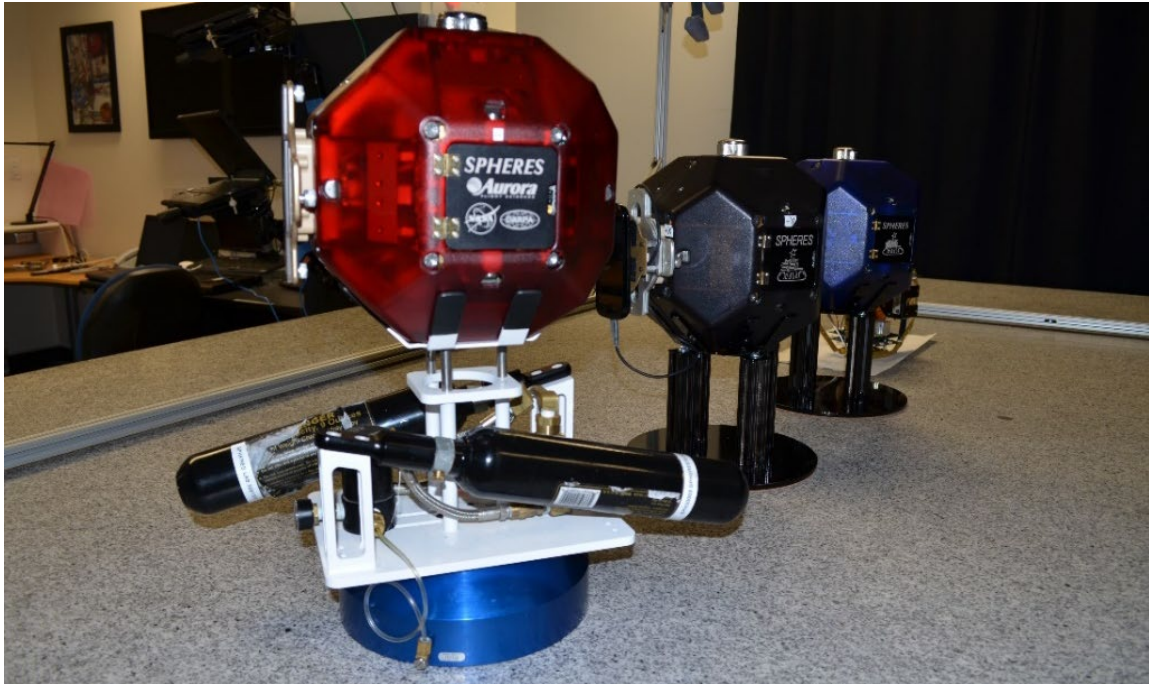


Figure 9. A SPHERES satellite mounted on an air-bearing at the MIT Space Systems Laboratory. Source: [31].

Other examples of FSS's have included unique testing apparatuses used to study the dynamics of the NASA-developed Astrobees [32]. Astrobees are newly developed free-flying robotic systems utilizing an electrically-powered propulsion system designed to operate on the ISS [32]. Centralized in a cube-shaped form factor, Astrobees are the successors to the SPHERES satellites [32]. Astrobees focus on advanced autonomous navigation and robotic manipulation enabled through an advanced suite of sensors and computing hardware [32]. Like the SPHERES spacecraft, Astrobees test control maneuvers on Earth before executing commands on the ISS. Research conducted at the NASA Ames Research Center's air-table has helped refine planned maneuvers for the Astrobees [32]. Figure 10 shows the mock Astrobees tested inside the NASA Ames Research Center.

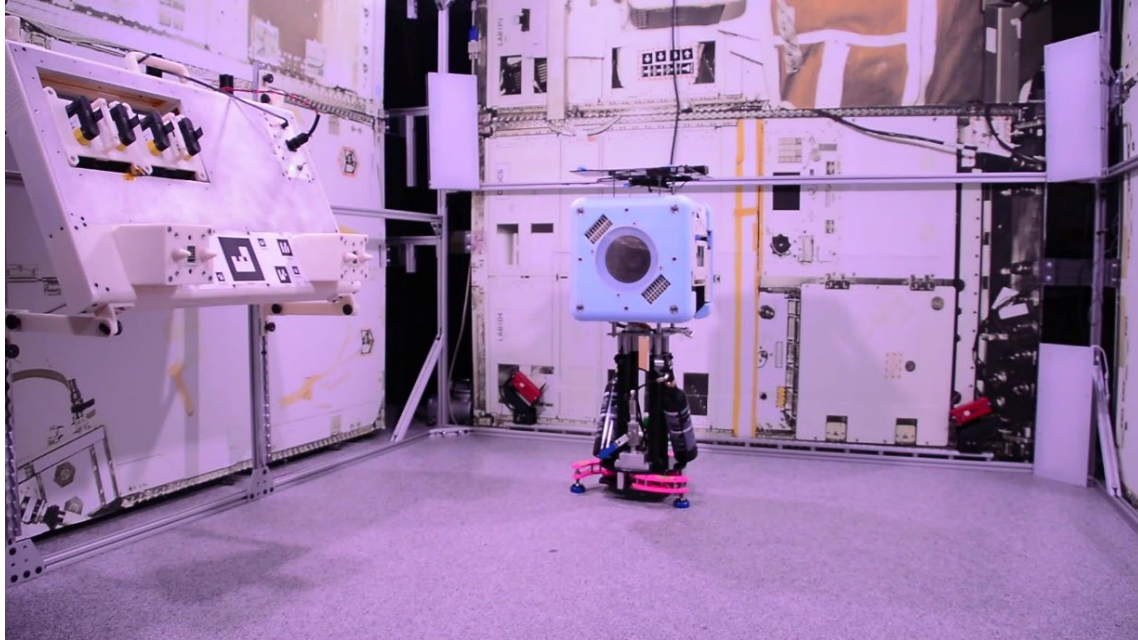


Figure 10. ASTROBEE testing at the NASA Ames Research Center. Source: [32].

The NPS-SRL has developed several generations of FSS's to study control algorithms, docking and rendezvous maneuvers, and robotic arm manipulators for on-orbit servicers [29]. Many of the NPS-developed FSS's have utilized gas thrusters and reaction wheels to control the FSS's, similar to real spacecraft control actuators. One of the more advanced FSS systems developed by the NPS-SRL is the Spacecraft Proximity Operations Simulation [33]. Comprising two FSS's, one FSS consisted of a chaser spacecraft, while the other FSS was a target spacecraft [33]. The chaser FSS utilized a planar robotic manipulator and adaptive gripper to capture the target FSS [33]. This simulation setup aimed to study the complex maneuvers and control needed to attach the chaser FSS to the target FSS [33]. Figure 11 shows an image of the Spacecraft Proximity Operations Simulation.

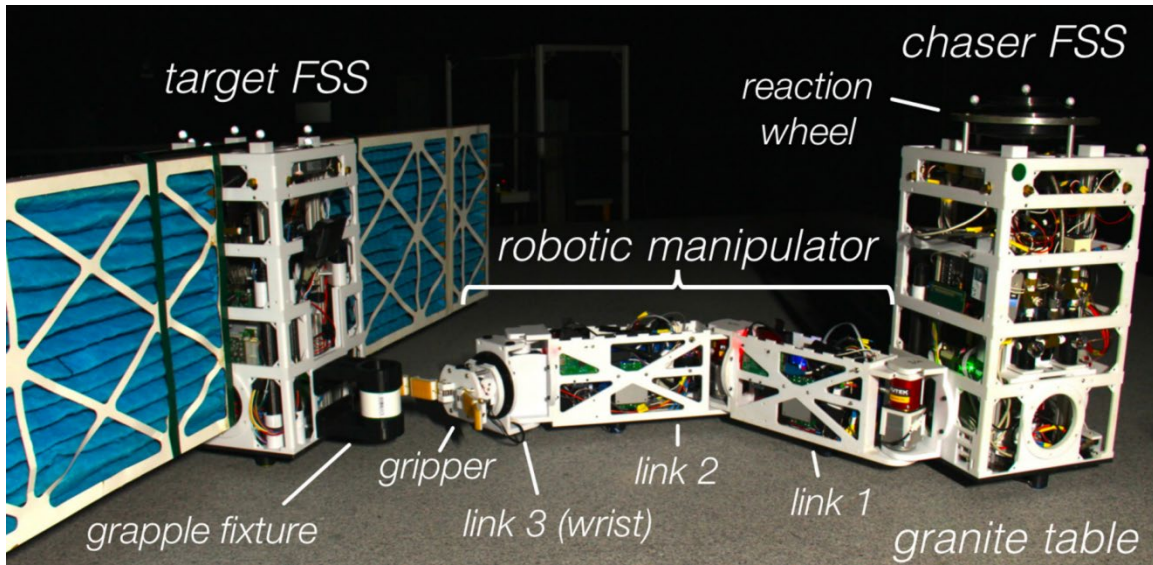


Figure 11. Spacecraft Proximity Operations simulation at the NPS-SRL.

Source: [33].

F. RESEARCH FOCUS AND OUTLINE

Research utilizing FSSs often involves hardware-in-the-loop testing configurations, providing researchers with invaluable insights to dynamic interactions of spacecraft. However, setting up these testing environments requires careful planning and design refinement before testing physical hardware. Researchers commonly use computer-aided design (CAD) to develop FSS designs and mathematical models to refine control algorithms. By using computer-based simulations, researchers can refine FSS designs and control algorithms before testing them in real-world settings.

Research for this thesis involves the mathematical modeling of a novel FSS developed by the NPS-SRL. This FSS system will utilize a Kinova Gen3 7 DoF Robotic Manipulator and other commercial off-the-shelf-components (COTS). Research within this thesis will develop a mathematical model capturing the dynamic interactions of the Kinova robotic manipulator while attached to the FSS base. Additionally, the model will simulate the response of the FSS system when subjected to external forces and torques. Test cases will also demonstrate the functionality of the mathematical model in both simulated and hardware-in-the-loop applications.

Chapter II of this thesis will provide a system overview of the FSS, including the FSS base, Kinova robotic manipulator, and adaptive gripper. Chapter III will provide an overview of the mathematical modeling of the Kinova Arm, followed by hardware-in-the-loop verification and validation in Chapter IV. Chapter V will detail the mathematical modeling of the entire FSS system with the robotic manipulator attached to the FSS base. Simulation-based and hard-ware-in-the-loop validations will verify and validate the FSS system model in Chapter VI. Finally, Chapter VII will be the concluding remarks of the research, addressing areas for future work.

II. FSS SYSTEM OVERVIEW

A. SUMMARY

The FSS system modeled in this research is an original design from the NPS-SRL created by Dr. Bautista Fernandez. The FSS system aims to be the next generation of FSS simulators for the NPS-SRL, representing a testing platform for research relating to OOS. Concurrently with this thesis, Ensign Katie Lupo completed design modifications and the initial construction of the FSS base. The FSS system will comprise an air-bearing base structure, 7-degrees-of-freedom (DoF) robotic manipulator, an adaptive gripper, and other COTS components. Figure 12 shows a CAD model of the FSS system.

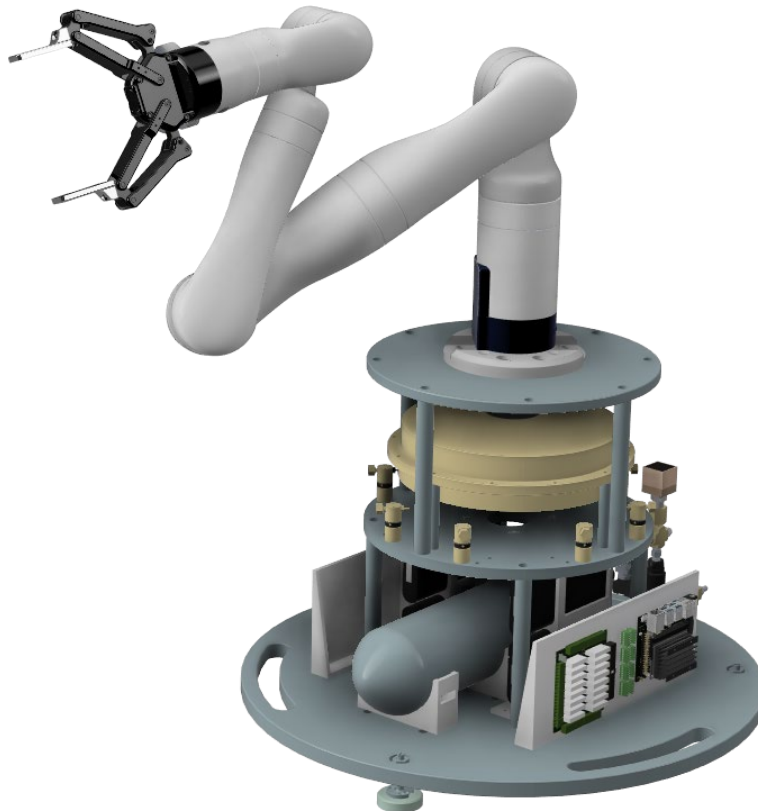


Figure 12. CAD rendering of the complete FSS system.

This FSS system will operate on the POSEIDYN testbed, testing the dynamic coupling between the robotic manipulator and the FSS base. The FSS system will be operated remotely from a pre-programmed control maneuver and a human-in-the-loop control mode. Research with this FSS system will involve control algorithm development, along with utilizing the robotic manipulator to perform tasks simulating OOS scenarios. Future plans for this FSS system may see the spacecraft interacting with other FSS's, executing rendezvous and docking maneuvers. Testing could also employ machine learning navigational systems to demonstrate autonomous control of the FSS system.

B. SYSTEM MODEL

A systems modeling language (SysML) model was developed to depict specific aspects of the FSS system relevant to this research. As a general-purpose modeling language, SysML enables detailed descriptions of complex engineering systems. Figure 13 presents a use case diagram illustrating user interactions and the operation of the FSS system. Figure 14 outlines the FSS system's logical structure, encompassing the FSS system's major components. Figure 14 also details specific parameters describing the link masses, joint limits, joint torques, and other pertinent information specific to the FSS system.

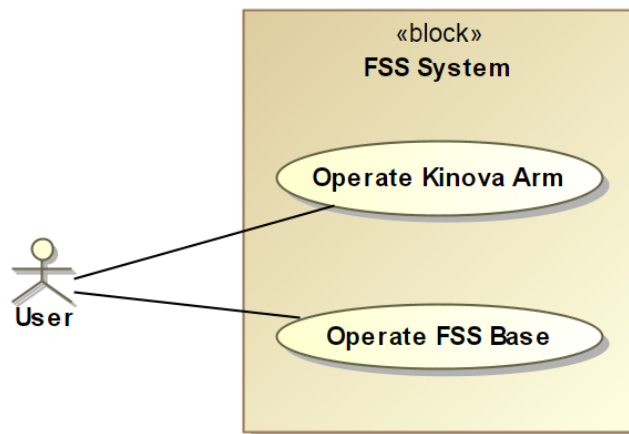


Figure 13. Use case diagram for the FSS system.

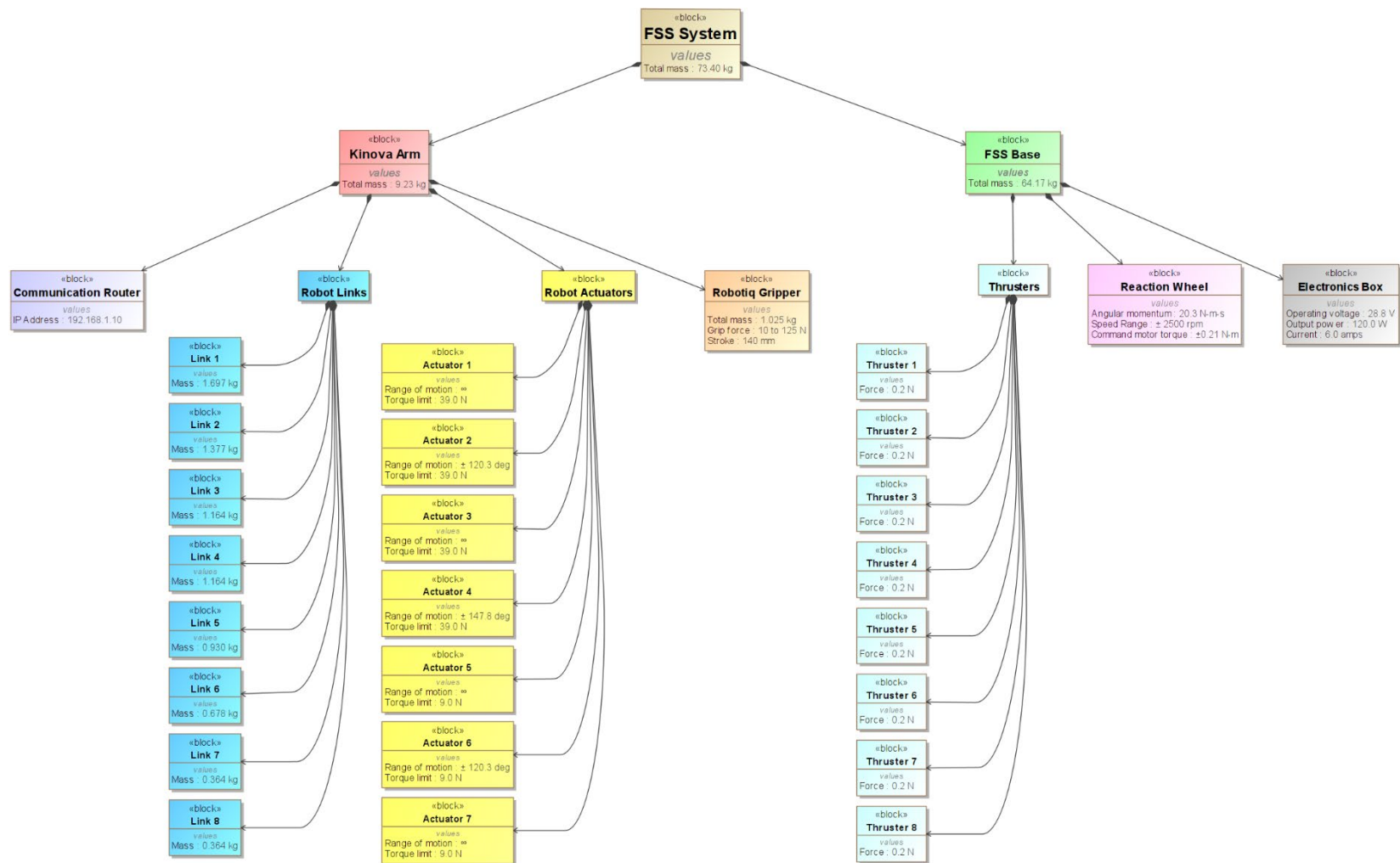


Figure 14. Logical structure for the FSS system.

C. FSS BASE

The FSS base will be the main structural component of the FSS system and will consist of a tiered platform design. This component will house electronics, pressurized air, and control actuators necessary to carry out the functions of the FSS system. The main structure of the FSS base will comprise mainly aluminum hardware. Design trades for the FSS base have focused on maintaining high structural rigidity while conserving the overall mass of the system. CAD model estimates calculate the overall mass to be 64.17 kg.

The first level of the FSS base, between the bottom and middle plates, will house the primary electronics and power storage of the FSS system. It will also store a high-pressure storage tank necessary to run the air-bearings and thrusters of the FSS base. The bottom plate will have three air-bearings to provide low-friction contact between the FSS system and the POSEIDYN granite surface. Power will be provided by four batteries centrally located on the first level. The POSEIDYN Vicon system will track the positional information for the FSS via a VIVE tracker located on the side of the FSS base. The primary electronics will sit opposite the VIVE tracker to maintain equal mass distribution of the FSS base.

The second level, between the middle and top plates, will house the control actuators. These control actuators will dictate the lateral position and rotational orientation of the FSS system. The control actuators will have eight high-pressure gas thrusters and one reaction wheel. The custom-designed gas thrusters will produce an average thrust of 0.2 N per thruster. These thrusters will be paired and attached to the FSS base by a 90 deg spacing, providing control over lateral movement. Each thruster pair will lie equidistance from the z-axis, meaning that the resulting thrust of each thruster pairing will not produce a torque on the FSS base. A Ball Aerospace 20.3 N-m-s reaction wheel, located at the center of the FSS base, will control the angular rotation and orientation of the FSS base [34]. The position of the reaction wheel will provide torque along the vertical axis of the FSS base. Figure 15 shows a CAD model detailing the different components of the FSS base.

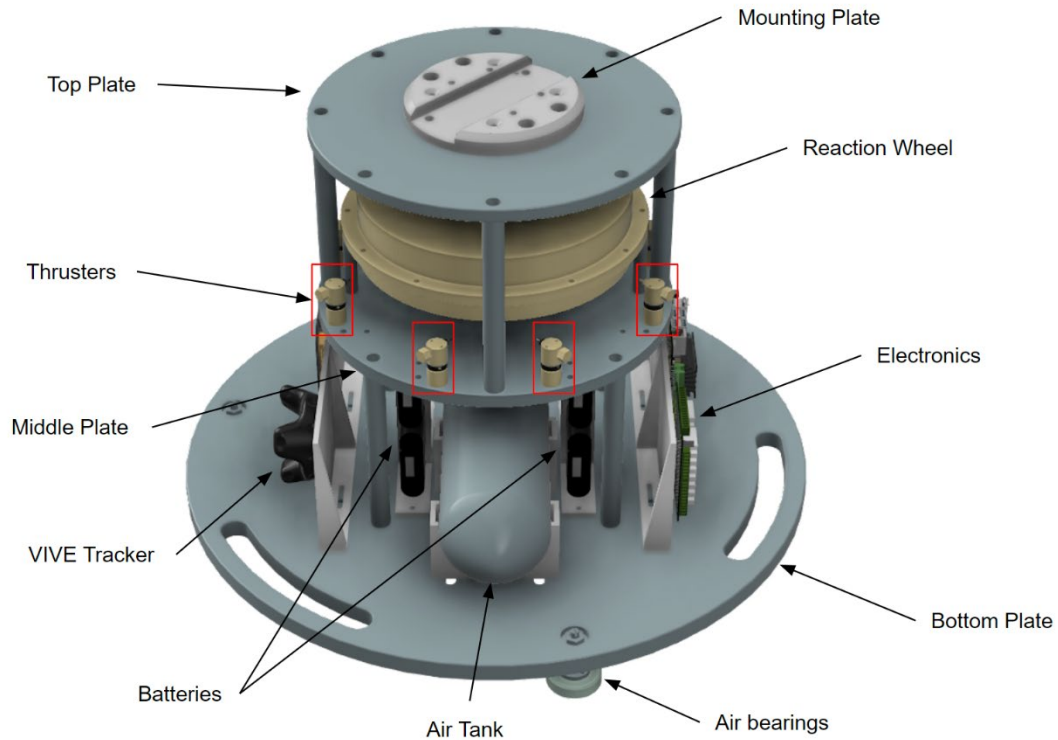


Figure 15. Labeled components of the FSS base.

The FSS base will have an overall height of 454.37 mm and be 608.6 mm at its widest point. It is assumed to be a rigid body structure with three DoF within the POSEIDYN testbed. These three DoF consist of two translational axes and one rotational axis. The two translational axes consist of lateral movement along the x-axis and y-axis, with rotational motion along the z-axis. The mounting plate shown in Figure 15. will be the anchoring point, or base link location, for the robotic manipulator. This mounting plate will place the base link of the robotic manipulator at the x-axis and y-axis origin of the FSS base with a z-axis location of 454.37 mm. All torques and forces produced by the arm will be transmitted through this mounting plate and to the FSS base. The center of mass (COM) of the FSS base will lie at the origin of the x-axis and y-axis with a z-axis position of approximately 228.41 mm, measured from the bottom of the air-bearings. Figure 16 defines the coordinate system and COM of the FSS base.

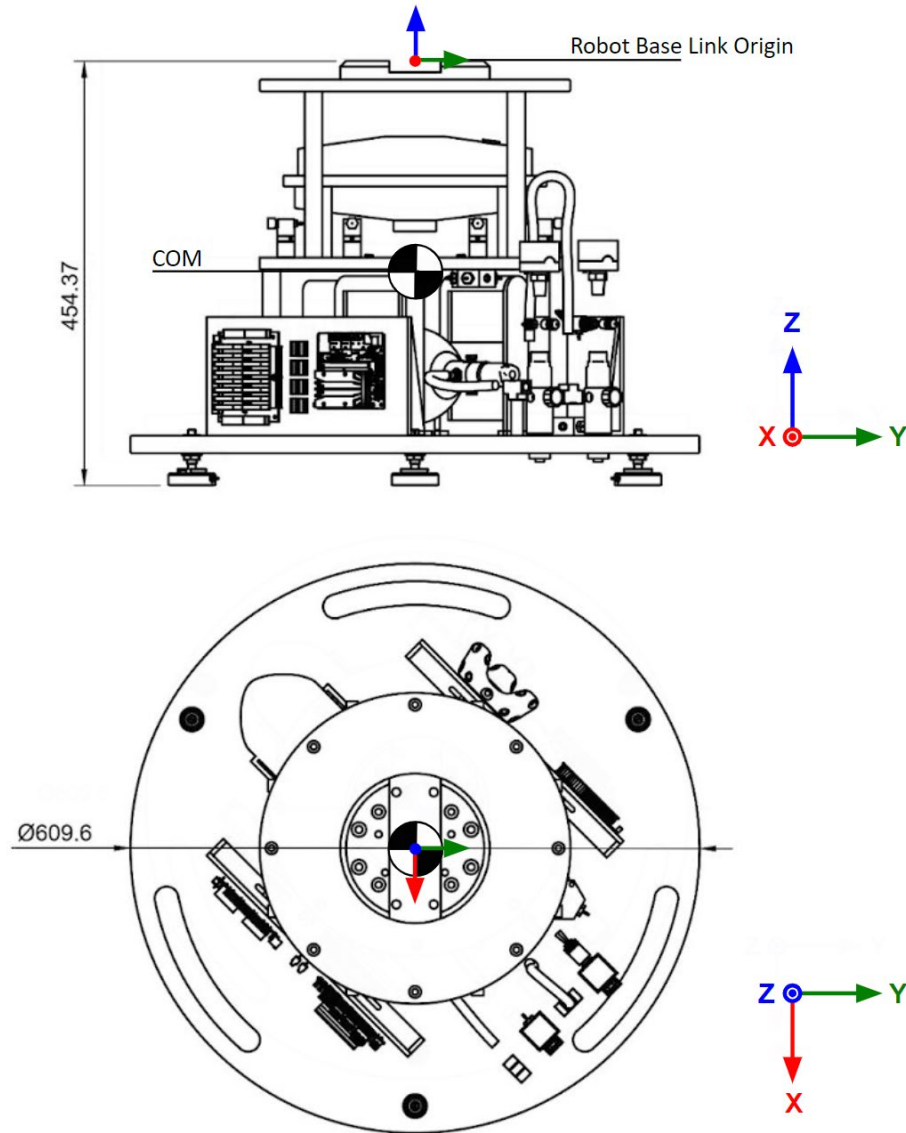


Figure 16. FSS base coordinate system and COM location.

D. KINOVA ARM

The FSS system will utilize the Kinova Gen3 Ultra lightweight robot manipulator designed by the Canadian company Kinova Robotics [35]. Kinova Robotics manufactures two versions of this robotic manipulator: 6 and 7 DoF models [35]. The NPS-SRL has purchased the 7 DoF model and will utilize it for the FSS system. Figure 17 is an image of the Kinova 7 DoF robotic manipulator.



Figure 17. Kinova 7 DoF robot manipulator. Source: [36].

Kinova states that the robotic manipulator can perform various functions ranging from dynamic grasping, haptics, and dexterous assembly. Within the FSS system, the arm will test control algorithms, robot-object interactions, and other tasks relevant to OOS. The Kinova arm has a total mass of 8.2 kg and can support payloads up to 2–4 kg [37]. It can facilitate 4 kg payloads for mid-range continuous motion and up to 2 kg for full-range continuous motion [37]. It has a maximum speed of 50 cm/s second with a maximum reach of 902 mm [37]. Table 1 summarizes the general specifications of the Kinova arm. Figure 18 displays the absolute workspace for the Kinova arm.

Table 1. Kinova Gen3 7 DoF robotic manipulator general specifications.
Adapted from [37].

Specifications	Gen3 7 DoF
Payload (full-range continuous)	2 kg
Payload (mid-range continuous)	4 kg
Total mass	8.2 kg
Maximum reach	902 mm
Maximum speed	50 cm/s
Actuator joint range	∞

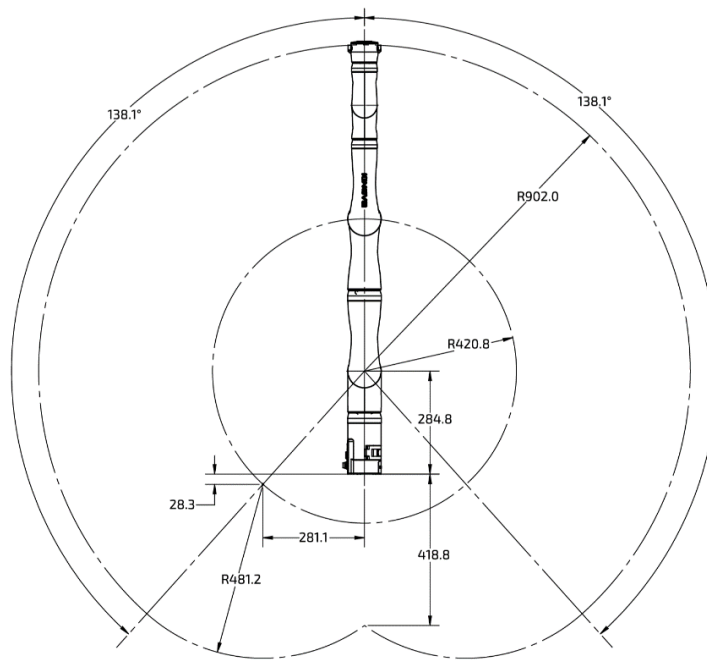


Figure 18. Absolute workspace of Kinova 7 DoF robotic manipulator. Source: [35].

The Kinova arm consists of eight links connected by seven joints [35]. The Kinova arm utilizes large and small actuators for joint movements [35]. The large actuators are integrated into joints 1–4, with the small actuators utilized by joints 5–7 [35]. These actuators can provide up to 9.0 N-m and 30.0 N-m of torque for small and large actuators [35]. Large actuators are limited to speeds of 79.64 deg/s and accelerations of 57.30 deg/s², while small actuators are limited to speeds of 69.91 deg/s and accelerations

of 572.95 deg/s^2 [35]. Odd-numbered joints can execute full revolute motion; however, even-numbered joints are range motion limited [35]. Table 2 summarizes the limits and capabilities of the robotic manipulator. Figure 19 defines the coordinate system of the Kinova arm and the actuator frames used in each joint frame.

Table 2. Limits and capabilities of the robotic manipulator. Adapted from [35], [38].

Joint	Range of Motion (deg)	Speed Limits (deg/s)	Acceleration Limits (deg/s ²)	Torque Limits (N-m)
1	∞	79.64	57.30	39.0
2	± 128.9	79.64	57.30	39.0
3	∞	79.64	57.30	39.0
4	± 147.8	79.64	57.30	39.0
5	∞	69.91	572.95	9.0
6	± 120.3	69.91	572.95	9.0
7	∞	69.91	572.95	9.0

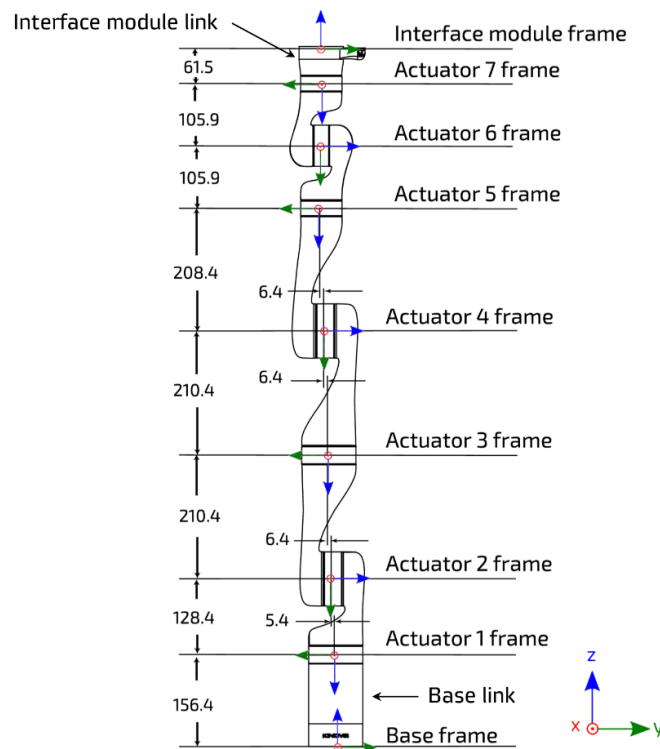


Figure 19. Schematic of the Kinova 7 DoF robot manipulator. Adapted from [35].

The Kinova arm handles communication commands via a remote procedure call (RPC) [35]. Several methods are available to generate RPC commands, including the Kinova Web app, a Robot Operating System (ROS) node, or other robotics-based software [35]. A wired connection from a computer terminal transmits the RPC command to the communication router located in the base link of the Kinova arm [35]. The router forwards the RPC to the embedded controller, where an internal control library computes the necessary joint commands for each robot actuator [35]. The controller operates at a 1 kHz internal loop, comparing reference commands to the measured output of each actuator [35]. The controller facilitates either pre-programmed trajectories or real-time commands of the robotic manipulator [35]. Previous thesis work in the NPS-SRL has focused on task space robotic manipulator control specifically for the Kinova 7 DoF arm [38]. Additionally, this thesis work utilized the onboard camera in the interface module link for color detection functionality [38].

E. ROBOTIQ GRIPPER

The FSS system will utilize the Robotiq 2F-140 gripper as the Kinova arm end-effector tool [39]. This adaptive gripper can interface with several robotic manipulators, including the 7 DoF Kinova arm [39]. The Robotiq gripper features internal and external parallel gripping, providing flexibility in object grabbing [39]. Figure 20 is an image of the Robotiq adaptive gripper. The adaptive gripper features an adjustable stroke of 140 mm with a positional resolution of 0.6 mm [40]. The closing speed of the gripper has an adjustable speed of 30 to 250 mm/s [40]. The mechanical design of the gripper enables it to pick objects in both a form-fitting grip and friction grip mode with a grip force between 10 to 125 N [40]. The form-fitting mode allows the fingers of the gripper to conform to an object's shape [39]. This form-fitting mode gives the gripper more contact surface area with the object. Alternatively, the friction grip mode can use rubber contact surfaces to grasp an object [39].



Figure 20. Robotiq 2F-140 adaptive gripper. Source: [41].

Both gripper modes can handle objects up to 2.5 kg [40]. The mass of the gripper is 1.025 kg meaning that its heaviest configuration, the gripper plus the payload it grabs, will have a maximum mass of approximately 3.5 kg [39], [40]. This total mass of the gripper payload complies with the limits defined by the Kinova user guide for mid-range continuous motion [35]. However, the gripper is limited to a payload of less than 1 kg for full-range continuous motion. Table 3 provides a detailed summary of the Robotic gripper specifications. Figure 21. provides the overall dimensions of the adaptive gripper, including the interface module frame.

Table 3. Robotiq adaptive gripper specifications. Adapted from [40].

Specifications	2F-140 Adaptive Gripper
Stroke (adjustable)	140 mm
Grip force (adjustable)	10 to 125 N
Max payload mass (mid-range motion)	2.5 kg
Max payload mass (full-range motion)	1.0 kg
Gripper mass	1.025 kg
Position resolution (fingertip)	0.6 mm
Closing speed (adjustable)	30 to 250 mm/s

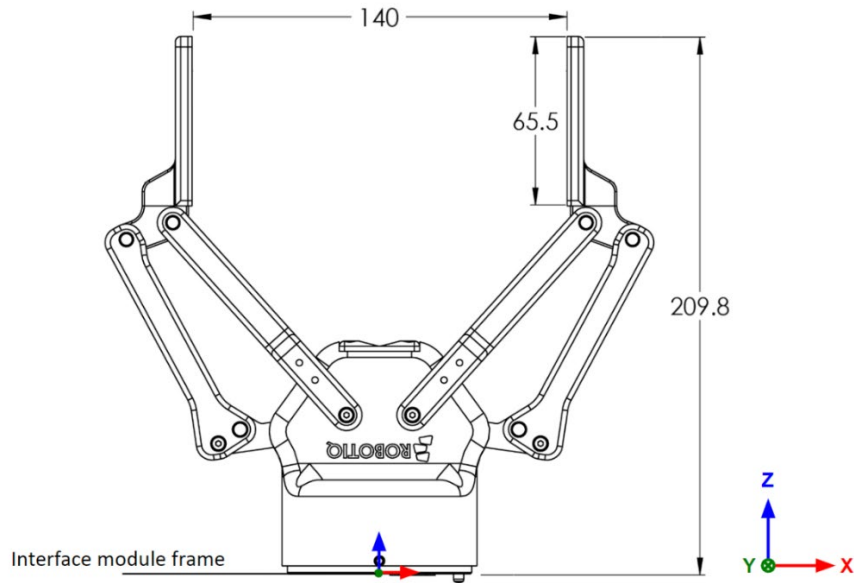


Figure 21. Robotiq adaptive gripper schematic. Adapted from [39].

The Robotiq gripper features an embedded controller similar to the Kinova arm [39]. The interface module link provides power and command signals to the adaptive gripper facilitated by the Kinova arm. The adaptive gripper can detect objects and provide error messages, enabling the user to utilize the end-effector for various applications [39]. The gripper detects objects by measuring the electrical current of the gripper's motor [39]. A higher current drawn from the motor correlates to a higher gripper force. Gripper force limitations are also programmable by the user, allowing for an adjustable grip force for different objects [39].

III. MODELING AND SIMULATION OF THE KINOVA ARM

A. ROBOTIC MANIPULATOR MATHEMATICAL REPRESENTATIONS

To conduct numeric simulations of the Kinova arm, the robotic manipulator required a mathematical representation of its links, joints, and inertias. This thesis assumed rigid body interactions in its modeling, ignoring the deformation of the robot links, frictions associated with joint movements, and other perturbations. Several mathematical methods exist to represent rigid body robotic systems. While some methods only capture the kinematic aspects of a robotic manipulator, others also incorporate geometries and inertial properties. Standard robot conventions are summarized below.

1. Denavit-Hartenberg Parameters

Denavit-Hartenburg (DH) parameters are one of the most common conventions to describe the kinematic structure of a robotic manipulator. Based on a standardized set of parameters, it allows a user to describe coordinate frames to each link and joint of a robotic manipulator system. For each DoF in a robotic manipulator, the DH parameters define a link length, link twist, joint distance, and joint angle. The DH parameters can facilitate any robotic system with revolute and prismatic joint types. DH parameters do not capture joint limits, inertial properties, or geometries of a robotic manipulator. The user must define these parameters separately if desired for virtual simulation. [42] provides a more detailed description of the D-H parameters.

2. Unified Robot Description Format

The unified robot description format (URDF) is an XML-based file format popular with most robot software packages. URDFs provide a robot system's kinematic structure, inertial properties, and robot geometries. Users can produce URDF files manually or can generate files through file converters. URDFs also facilitate more advanced user inputs, such as sensors and control plugins, useful for high-fidelity robot simulations. The compact nature of URDF files enables easy implementation into most robot simulation programs. Tutorials and supplementary information on how to write URDF files is included in [43].

3. CAD-Based Models

CAD-based models are often the most direct way of virtually representing a robotic manipulator and are familiar to most engineers. In a CAD model, users define the geometries of a robot through sketches and transform the sketches into 3D geometries. These geometries are then assigned a material, and the CAD software automatically calculates the inertial properties. Users connect different geometries through assemblies and define the joint connections of a robotic manipulator. CAD models can be directly imported into simulation software or converted to a suitable file format, depending on the CAD software used.

4. Multi-Body Dynamics Model

A multi-body dynamics (MBD) model is a convention that creates an interconnected system of rigid bodies to represent a robotic manipulator [44]. Typically, MBD models are constructed from interconnected blocks, defining joint types, inertias, geometries, and constraints. Blocks can also include contact surfaces and external forces. This block convention allows for high customizability, enabling users to define their robot model's parameters precisely. Users can generate MBD models from CAD files or manually define them. Convertors also exist to take URDF files and translate them to MBD models.

B. KINOVA ARM MODEL DEVELOPMENT

This research developed two mathematical models of the Kinova arm; one model utilized the URDF convention, while the other employed a MBD model. The URDF file selection was a natural choice because Kinova Robotics provided the files necessary to generate a URDF file [45]. The selection of an MBD model was driven by its functionality, allowing for the development of a high-fidelity model of the Kinova arm. Other conventions were avoided because of their limited capability or difficulty implementing them into this research.

The URDF file generated for the Kinova arm utilized files provided by the Kinova Robotics GitHub repository [45]. This repository provided geometry files generated from

Kinova's CAD model of the 7 DoF robotic manipulator. Kinova Robotics also included a generalized URDF file of the Kinova arm, but this file did not include the adaptive gripper. The Robotiq gripper's geometry files and inertias were manually inputted into the URDF and obtained from [39], [46]. Other modifications to the URDF file included converting the digital asset exchange (DAE) files provided by Kinova Robotics into stereolithography (STL) files [45]. The conversion of these geometry files was necessary for later simulation implementation. The completed URDF file developed for the Kinova arm is in Appendix A. Figure 22 shows a visualization of the Kinova arm defined by the URDF model.

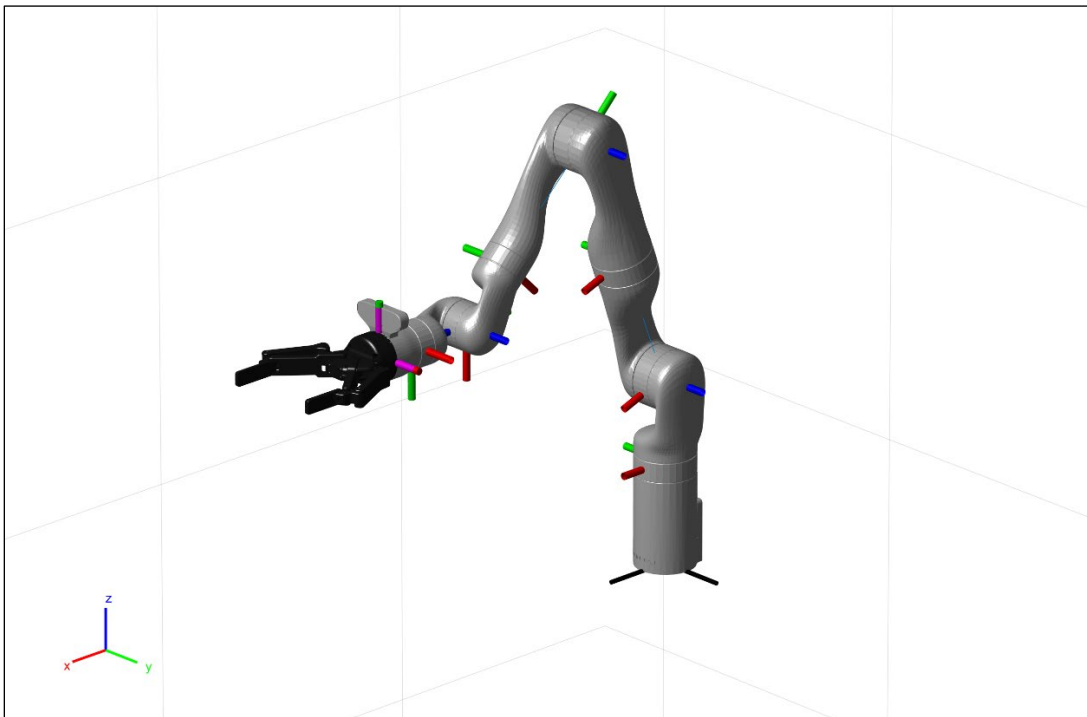


Figure 22. Visualization of the Kinova arm defined by the URDF model.

Development of the MBD model began by converting the contents of the URDF file into a MATLAB-Simulink Simscape model. A built-in conversion function, provided by the MATLAB Multi-Body Simscape toolbox, helped facilitate the MDB model development [47]. The converter created the generalized kinematic structure of the Kinova arm, defining joint connections, coordinate systems, geometries, and inertial properties.

Figure 23 shows an example of block definition comprising the MBD model's shoulder link. The file converter provided a starting point for the MBD model development; however, environmental MBD blocks needed manual input. Manually defined blocks included specifying the inertial Cartesian coordinate system (CCS) and the Kinova arm's attachment to inertial space. Figure 24 shows the completed MBD model. Figure 25 shows a visualization of the Kinova arm defined by the MBD model.

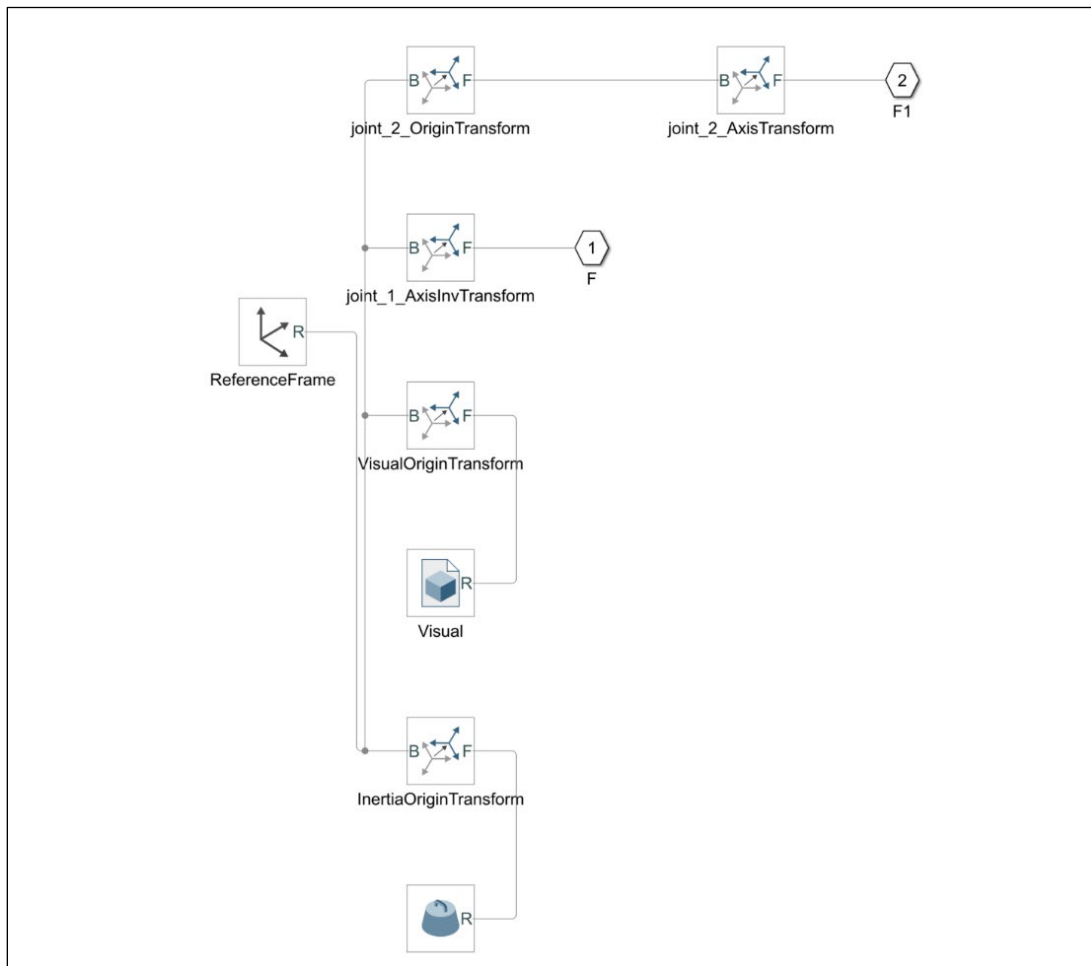


Figure 23. MBD block definition for the Kinova shoulder link.

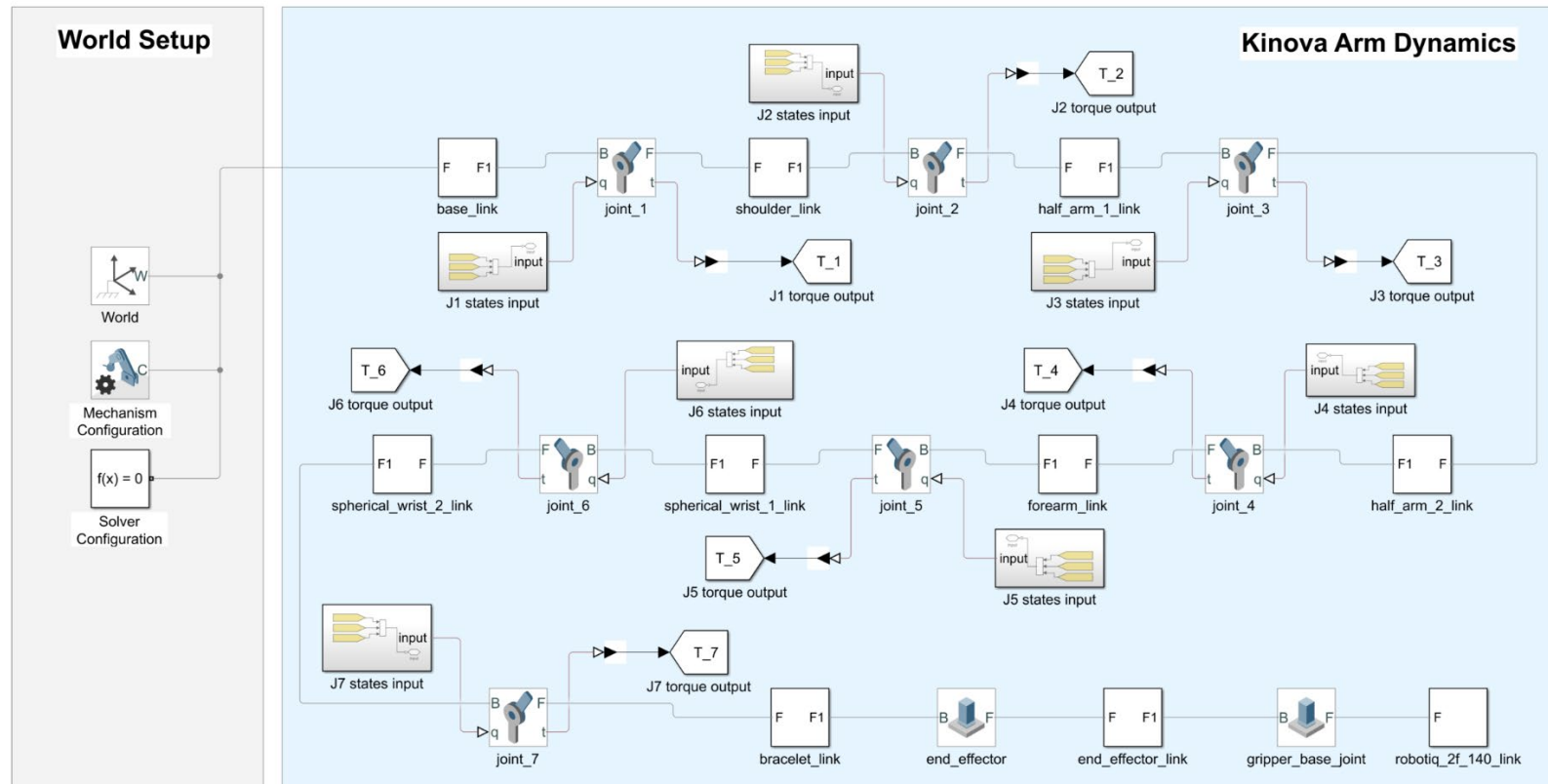


Figure 24. The developed MBD model for the Kinova arm.

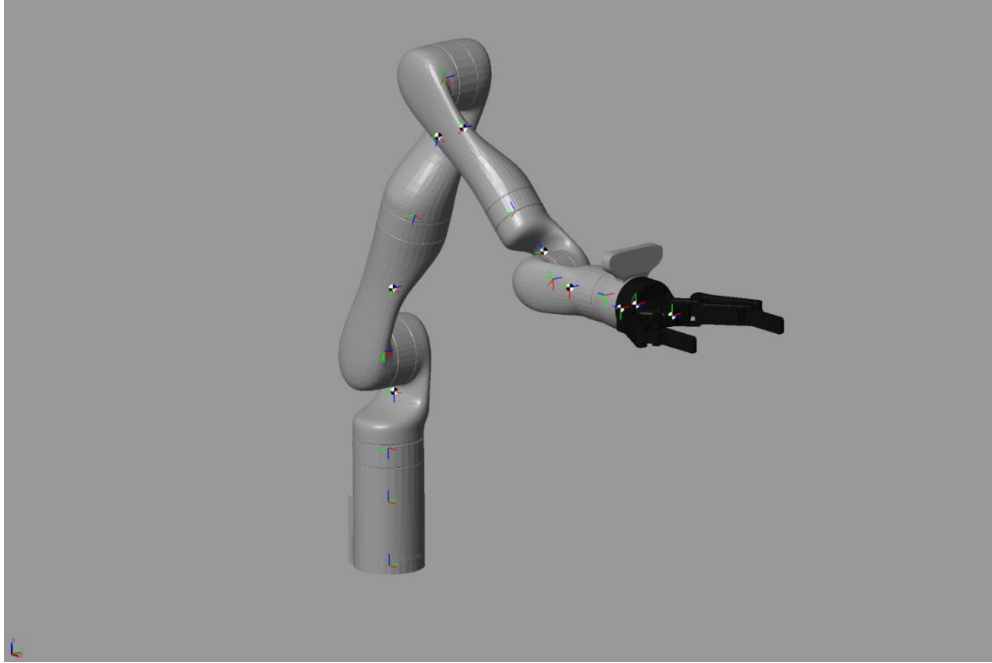


Figure 25. Visualization of the Kinova arm defined by the MBD model.

C. SIMULATION SOFTWARE

Previous graduate thesis work completed by Lieutenant Angelia O'Toole, a former student of the NPS-SRL, focused on researching simulation software capable of handling the complex dynamics involved with space logistics [48]. O'Toole's research also needed simulation software capable of simulating robotic manipulators for space-based applications [48]. Software accessibility, user-friendliness, and available functionality were among the criteria utilized by O'Toole's research [48]. Based on O'Toole's findings and individual research conducted for this thesis, two candidate simulation software were selected: Gazebo and MATLAB-Simulink [48].

Gazebo is an open-source 3D dynamics simulator that provides real-world physics simulations for robot applications [49]. It integrates with the ROS ecosystem, which is an industry-standard robotic application [49]. Gazebo runs on Linux operating systems, allowing for high customizability. Customizations can include sensor models, plugins, and hardware-in-the-loop capabilities [49]. This functionality enables high-quality visualizations of a robotic system, which is useful for understanding the complex dynamics of a robotic system.

MATLAB-Simulink are products developed by MathWorks that enable powerful engineering and science-based computation [50], [51]. MATLAB (short for “Matrix Laboratory”) provides a programming platform to enable matrix-based computation [51]. Simulink provides a virtual block diagram environment and can be used to simulate models and complex engineering systems [50]. These software packages are widely used by academia, research, and industry. MATLAB-Simulink software is not exclusive to robotic simulations as it has applications across many fields, including aerospace, automotive, communications, and electronics.

Between these software options, MATLAB-Simulink satisfied the needed functionality of this research. Gazebo provides an excellent simulation environment but is a complex program to navigate for new users, whereas MATLAB-Simulink is typically a more commonly used software. Gazebo also presented compatibility issues with running hardware-in-the-loop simulations with the real-life Kinova arm. Considerable time and effort could not resolve communications issues.

D. SIMULATION SETUP

Simulation of the Kinova arm URDF model utilized a recursive Newton-Euler algorithm (RNEA). RNEA is a computationally efficient and stable algorithm used in robotics based on equations describing the motion and forces of a rigid-body system [52]. It is less complex and requires fewer computations than a Lagrangian approach [52]. RNEA also enables real-time computations, control algorithm adjustments, and simulations useful for robotic development. The equation that governs the RNEA algorithm’s mechanics is

$$\frac{d}{dt} \begin{bmatrix} q \\ \dot{q} \end{bmatrix} = \begin{bmatrix} \dot{q} \\ Mq^{-1}(-C(q, \dot{q})\dot{q} - G(q) - J(q)^T F_{Ext} + \tau) \end{bmatrix} \quad (1)$$

where M is the joint-space mass matrix defined by the inertial properties of a given robotic manipulator [53]. The Coriolis terms, encompassing the forces associated with objects rotating with a moving frame of reference, are defined as C . The forces and torques of each joint resulting from gravity are defined as G . The geometric Jacobians, specified for each joint robot configuration, are defined as J . The matrix comprising the external torques and

forces, or wrenches, acting on the robotic manipulator is F_{Ext} . The joint torques and forces for each actuator in a rigid body system are defined as τ . Finally, the joint configurations, velocities, and accelerations are q , $\dot{q}$, and $\ddot{q}$, respectively.

Numerous papers offer more complete derivations of the RNEA algorithm, detailing the specific relationships governing Equation 1 [52], [54], [55]. In short, the RNEA algorithm computes the inverse dynamics of a robotic manipulator by using each joint's configuration, velocity, and acceleration information. It consists of two main steps: forward recursion and backward recursion. During forward recursion, it calculates the kinematic properties of each link, starting at the base link and moving to the end-effector. Forward recursion considers the combined effect of internal torques, external forces, and gravity. During backward recursion, the algorithm computes the required joint torques using the information from the forward recursion process. As the RNEA algorithm moves backward, it calculates the resulting forces and torques acting on each link. These forces and torques are then transformed back to the base link frame of the robotic manipulator and summed to obtain the required joint torques.

Implementation of the RNEA algorithm utilized the MATLAB Robotics Toolbox [53]. This toolbox provides functions and algorithms for simulating a robotic manipulator and includes an RNEA-based inverse dynamics solver [53]. It features robot controllers capable of generating joint-space and task-space trajectories for a given robotic manipulator [53]. The Robotics toolbox also features a robot visualization function, enabling screenshots and videos to be generated from simulations [53]. Appendix C details the MATLAB script simulating the Kinova arm URDF model.

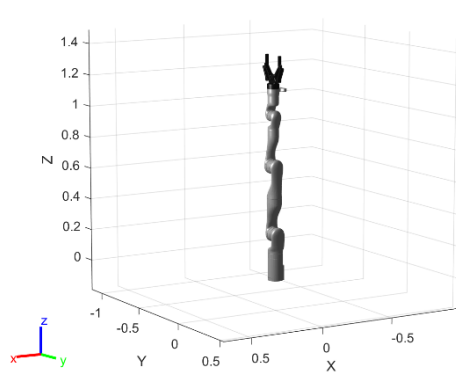
Simulation of the Kinova arm MBD model utilized the same software package, Simscape Multi-Body, to generate the MBD model [47]. This Simulink environment, using the Simscape software package, automatically calculates the kinematics and dynamics of a rigid-body system based on the block parameters created by a user [56]. The software package derives the equations of motion for each time iteration using Lagrangian or Newton-Euler formulations [56]. The solver determines which algorithm is needed based on a system's dynamic response to external and internal forces [56].

It is important to note that simulating the MBD model of the Kinova arm is different from the URDF model using the RNEA algorithm. For one, the simulation of the MBD model relies on a built-in physics engine native to the Simulink environment. Alternatively, the simulation of the URDF model in this thesis focuses on employing a set of kinematics and dynamic equations to analyze the Kinova arm without a dedicated physics engine. Simulation of the URDF model is well-suited for understanding the fundamental principles governing the Kinova arm's motion. However, it does not account for complex interactions such as collision detection or contact dynamics that a dedicated physics engine can handle.

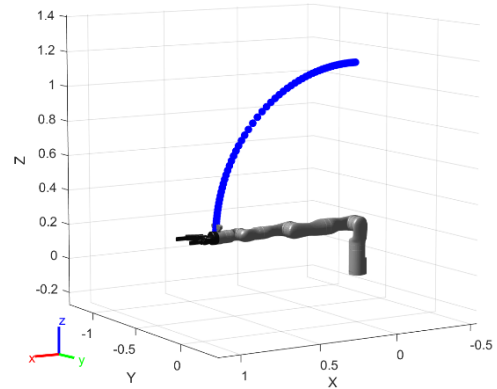
The Simscape Multi-body model provides an integrated simulation environment that accounts for complex physical interactions, offering a more comprehensive analysis of the robot's behavior in real-world scenarios [47], [56]. This integration allows capturing intricate details not encapsulated in simulating the URDF model. It is worth noting that simulation software like Gazebo can work with URDF files and incorporate a dedicated physics engine for more realistic and accurate robot simulations [49].

E. SIMULATED KINOVA ARM TESTS

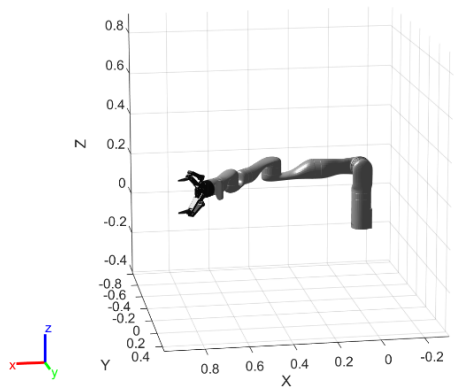
Comprehensive testing was essential to verify and validate the calculated results from both the URDF and MBD Kinova arm models. Simulations were developed to identify performance disparities between the two Kinova arm models. These simulations were divided into three distinct test cases: single actuator, multi-actuator, and full-actuator maneuvers. The first test, focusing on single actuator motion, confined the Kinova arm's movement to a planar range. The second test delved into more intricate dynamics of the Kinova arm by simulating multi-actuator motion. Finally, the third test examined the full-actuator motion maneuver, rigorously evaluating the calculated dynamics of each Kinova arm model. Figure 26 and Table 4 provide specific information for these tests.



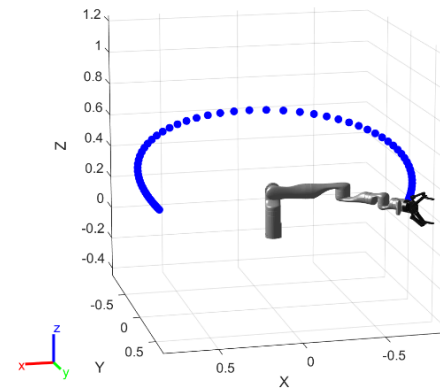
a) Test 1 start position.



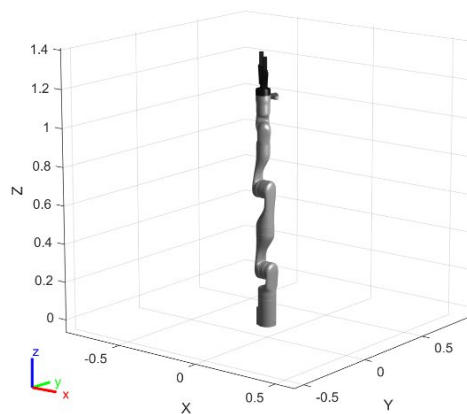
b) Test 1 end position and motion path.



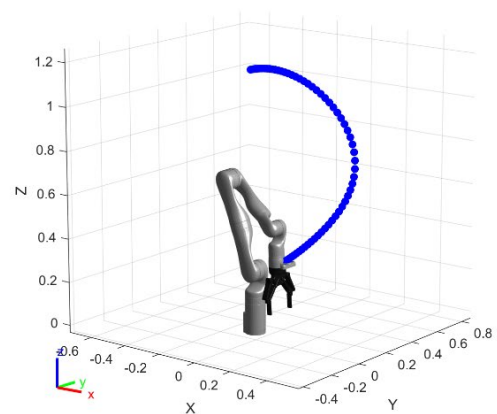
c) Test 2 start position.



d) Test 2 end position and motion path.



e) Test 3 start position.



f) Test 3 end position and motion path.

Figure 26. Visualization of the Kinova arm simulated test maneuvers.

Table 4. Simulated Kinova arm test maneuvers.

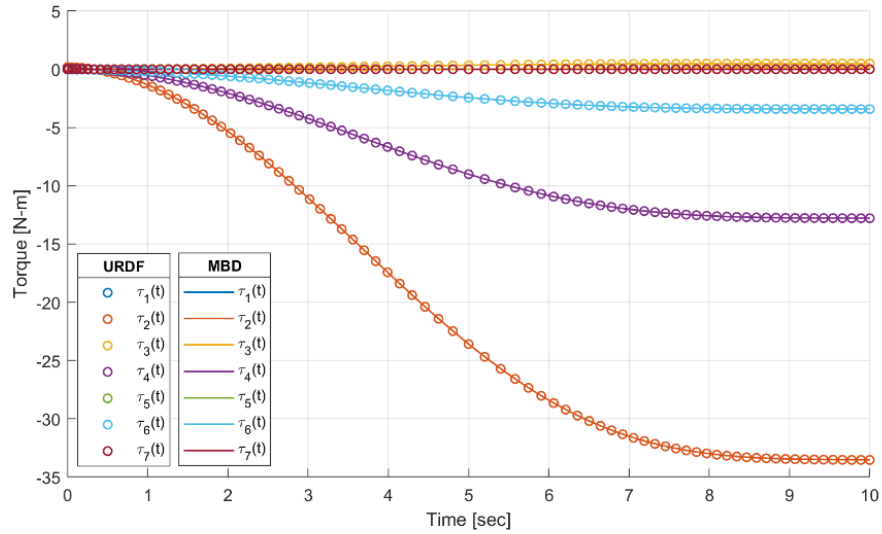
Joint	Test 1: Single Actuator Maneuver		Test 2: Multi-actuator Maneuver		Test 3: Full-actuator Maneuver	
	Start Position [deg]	End Position [deg]	Start Position [deg]	End Position [deg]	Start Position [deg]	End Position [deg]
1	0.0	0.0	0.0	190.0	90.0	0.0
2	0.0	90.0	90.0	90.0	0.0	340.0
3	0.0	0.0	90.0	270.0	0.0	180.0
4	0.0	0.0	315.0	315.0	0.0	214.0
5	0.0	0.0	0.0	0.0	45.0	0.0
6	0.0	0.0	0.0	0.0	0.0	310.0
7	0.0	0.0	270.0	270.0	270.0	90.0

Figure 26 offers a visual representation of the control trajectory for each test case, illustrating the Kinova arm’s initial and final configurations. The motion path for each maneuver is emphasized with blue dots. Table 4 presents detailed information for each test case, displaying the starting and ending positions. Figures 26a and 26b depict the Kinova arm performing an “arm-curl” maneuver during test 1, with Table 4 indicating that this maneuver revolves around joint two. For test 2, Figures 26c and 26d demonstrate an “arm-swing” maneuver, with Table 4 detailing the maneuver revolving around joints one and three. Finally, Figures 26e and 26f display the Kinova arm transitioning from a fully extended vertical configuration to a retracted pose. Table 4 specifies that all seven joints are used for the maneuver.

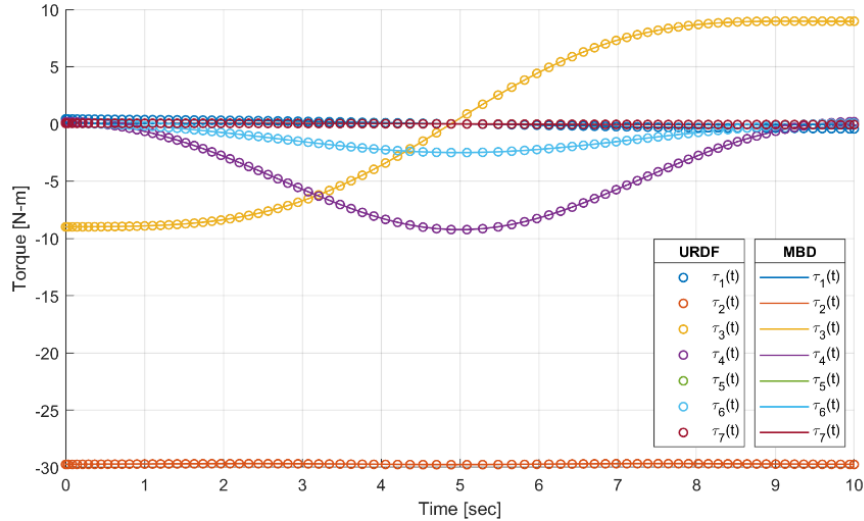
F. SIMULATED KINOVA ARM RESULTS

Simulation testing showed a close correlation between the URDF and MBD Kinova arm models. The calculated torques for each simulation test served as the performance metric to compare the Kinova arm model. These calculated values were selected because accurate joint torque calculations depended on the proper functioning of each model’s solver implementation. Figure 27 presents the torque history from both models across each test case. Figure 27 shows a close correlation between the URDF and MBD simulations, showing a near-identical match in the general shape and magnitude of the calculated torques. To quantitatively compare the URDF and MBD simulations, the mean absolute

error (MAE) was computed for each torque calculation in every test case. The findings are summarized in Table 5.

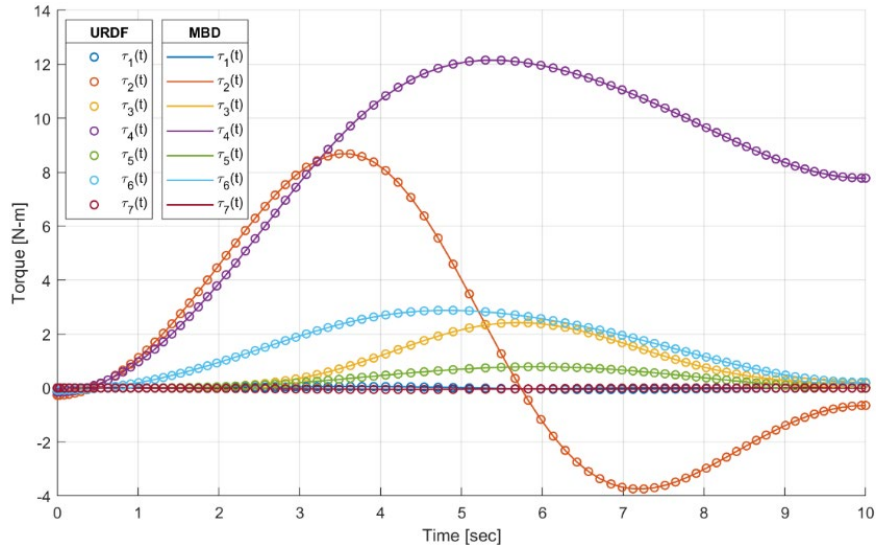


a) Test 1 torque calculation results.



b) Test 2 torque calculation results.

(Continued on Next Page)



c) Test 3 torque calculation results.

Figure 27. Calculated torques from the simulated Kinova arm test maneuvers.

Table 5. Mean average error between the URDF and MBD simulation tests.

Joint	Test 1: Joint Torque MAE [N-m]	Test 2: Joint Torque MAE [N-m]	Test 3: Joint Torque MAE [N-m]
1	$0.0032 * 10^{-12}$	$0.0120 * 10^{-12}$	$0.0397 * 10^{-13}$
2	$0.1187 * 10^{-12}$	$0.1567 * 10^{-12}$	$0.2891 * 10^{-13}$
3	$0.0016 * 10^{-12}$	$0.0422 * 10^{-12}$	$0.0236 * 10^{-13}$
4	$0.0332 * 10^{-12}$	$0.0186 * 10^{-12}$	$0.3696 * 10^{-13}$
5	$0.0001 * 10^{-12}$	$0.0002 * 10^{-12}$	$0.1071 * 10^{-13}$
6	$0.0028 * 10^{-12}$	$0.0224 * 10^{-12}$	$0.0413 * 10^{-13}$
7	$0.0002 * 10^{-12}$	$0.0002 * 10^{-12}$	$0.0024 * 10^{-13}$

Overall, the MAE was minimal for each test case, likely hitting the numerical precision of the computer employed in these simulations. In test 1, the highest MAE calculated was 0.1187×10^{-12} N-m for joint two. Joint 2 also exhibited the poorest performance in test 2, with an MAE of 0.1567×10^{-12} . The highest calculated MAE for test 3 was 0.3696×10^{-13} for joint four. The negligible MAEs calculated across the three test cases indicate no significant discrepancies between the URDF and MBD models. The low MAE values suggest that both models accurately and independently reproduce the same calculated results for the simulated Kinova arm model.

THIS PAGE INTENTIONALLY LEFT BLANK

IV. KINOVA ARM HARDWARE-IN-THE-LOOP TESTING

A. REAL-LIFE KINOVA ARM

Extensive hardware-in-the-loop testing was needed to validate the simulated Kinova arm models, as detailed in Chapter III. The Gen3 7 DoF Kinova arm, located in the NPS-SRL, was used to carry out this testing. Testing of the Kinova arm was performed on a mobile table to safely distance the arm from obstructions. Kinova Robotics provides a mounting clamp designed for the Kinova arm, and this clamp was used to securely fasten the Kinova arm to the table. Figure 28 is an image of the test setup with the real Kinova arm. To ensure safe testing of the real Kinova arm, precautions such as outlining keep-out zones and incorporating foam padding to the mobile table were implemented. These measures aimed to prevent potential accidents and harm to the Kinova arm during testing. All test maneuvers were simulated virtually before executing the commands on the real Kinova arm.



Figure 28. Kinova arm pictured in the NPS-SRL.

B. COMMAND GENERATION AND SENSOR RECORDING

The real Kinova arm was controlled using a MATLAB script, which transmitted commands via an ethernet cable connected to the base link of the Kinova arm. The Kinova arm required a 1000 Hz signal for executing maneuvers, comprising control histories that dictated each joint's configuration, velocities, and accelerations [35]. To plan control trajectories, joint-space waypoints were established, and a polynomial trajectory planner was utilized to solve the joint control trajectories. Once commands were sent to the Kinova arm, the MATLAB script recorded the arm's sensor data. This sensor information encompassed the joint configurations, velocities, and measured torques. The MATLAB script was designed to store this sensor data in Excel files for subsequent analysis. Appendix D provides the MATLAB script used for sending commands and recording sensor information.

The sensor data collected during commanded maneuvers was critical for assessing the performance of the simulated Kinova arm models, and collecting this information proved challenging. Primarily, communication through the ethernet cable proved problematic, often leading to bottlenecks as commands were sent to the Kinova arm and sensor data was transmitted for recording. The MATLAB script's recording rate was approximately 0.025 seconds or 40 Hz. Multiple attempts were made to enhance the sampling rate, but only minor improvements were achieved. This communication bottleneck also caused intermittent lapses in the recorded sensor data. Occasionally, sensor information was not received for over 0.15 seconds at the beginning and end of commanded maneuvers, resulting in gaps in the recorded data. Figure 29 highlights an example of a 0.115 sec gap during a recorded maneuver.

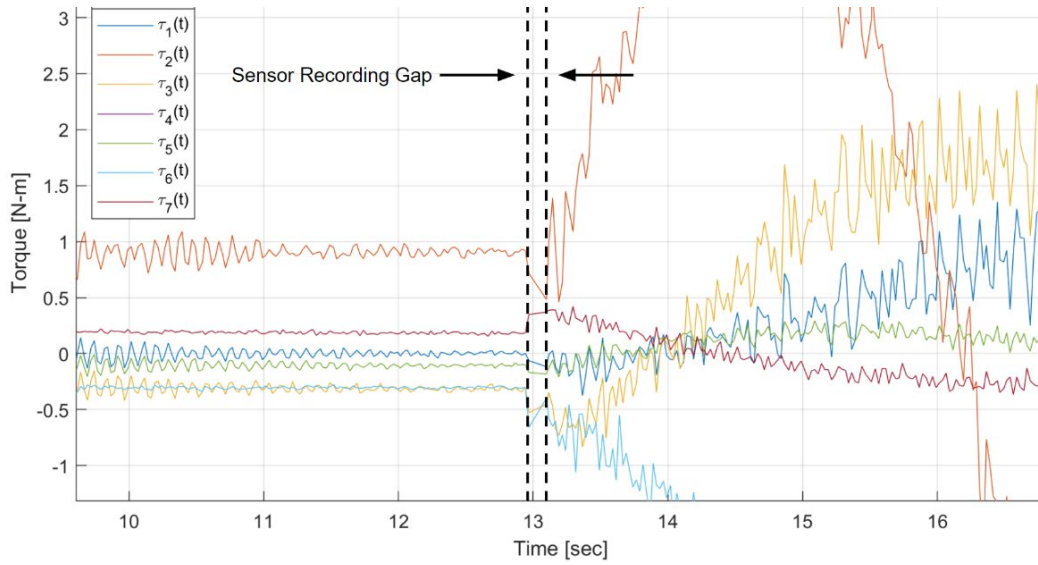


Figure 29. Sensor recording gap during Kinova arm maneuver.

Figure 29 also illustrates the noisy nature of the recorded torque history. Multiple factors likely contribute to this signal noise, but it is primarily caused by the table used to secure the Kinova arm. As the arm executed the commanded maneuvers, the generated torques caused the table to vibrate. This vibration subsequently led to oscillatory motion in the Kinova arm, altering the recorded torque sensor data. Other potential causes for the noisy sensor readings included joint frictions and the non-rigidity of the robot links.

Another problem encountered was that the Kinova arm lacked an adequate calibration functionality for its torque sensors. Specifically, torque values that should have indicated no torque measurements were measuring ± 5 N-m torques. Several efforts were made to adjust the drift of the torque sensors, but these approaches proved unreliable [45]. As an alternative, a unique calibration function

$$\tau_{offset} = \tau_{measured}(t_0) - \tau_{calculated}(t_0) \quad (2)$$

was implemented to provide a correction to the observed torque sensor drifts. Equation 2 modified the initially measured torque values, $\tau_{measured}$, by artificially offsetting their magnitude using the initially calculated torque values, $\tau_{calculated}$. This calculated torque offset, τ_{offset} , was applied to the entire measured torque history and effectively zeroed out the torque sensors.

C. HARDWARE-IN-THE-LOOP TESTING

A comprehensive series of tests were conducted to assess the performance of the simulated and real Kinova arm. Simulation models calculated the required joint torques to compare against the actual Kinova arm. In the real Kinova arm, these required joint torques were measured directly. By comparing the calculated versus measured torques, the accuracy of the simulated Kinova arm models could be evaluated. Given the near-identical performance of both the URDF and MBD models, only one model was chosen to evaluate the simulated Kinova arm performance. The URDF model was selected as the simulation model to compare against the actual Kinova arm. Appendix E details the MATLAB script used for simulating the URDF model against the real Kinova arm.

Two categories of testing were conducted during the hardware-in-the-loop testing: isolated arm tests and tests incorporating a test object. Isolated arm tests focused on confirming that the Kinova arm URDF model correctly modeled the real Kinova arm. For tests including a test object, they aimed to showcase the Kinova arm model's ability to model objects grasped by the adaptive gripper. The chosen test object was a hollow aluminum bar measuring 285.0 mm x 25.4 mm x 25.4 mm with a mass of 340.2 g. Figure 30 depicts the real Kinova arm gripping the aluminum test object.

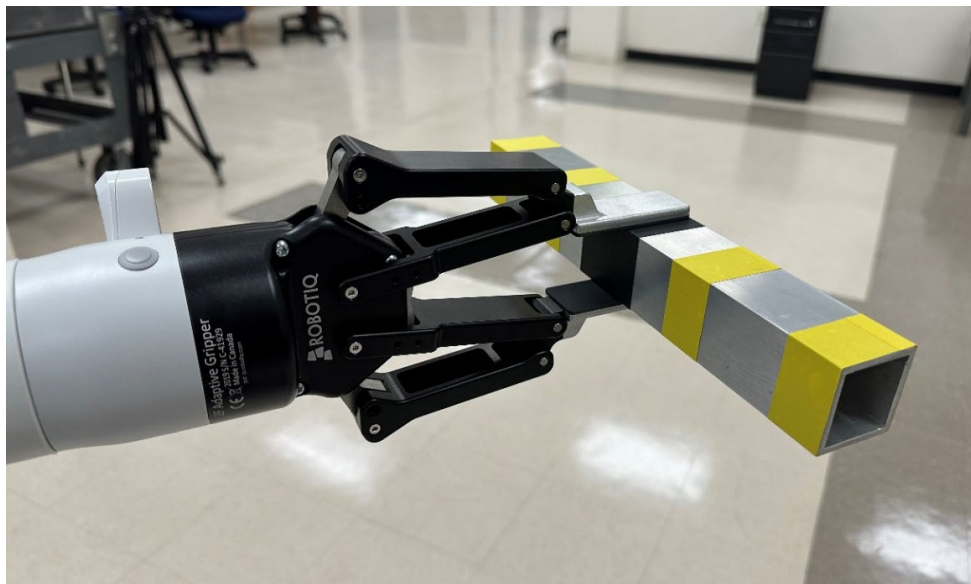
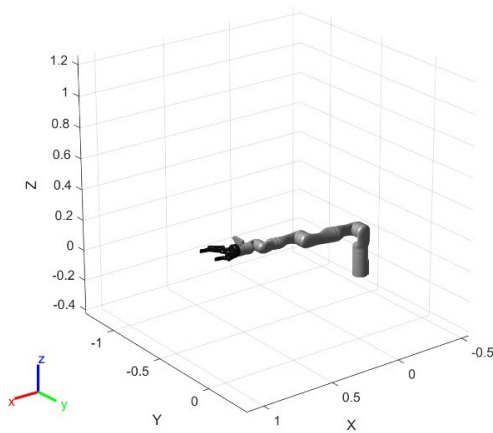


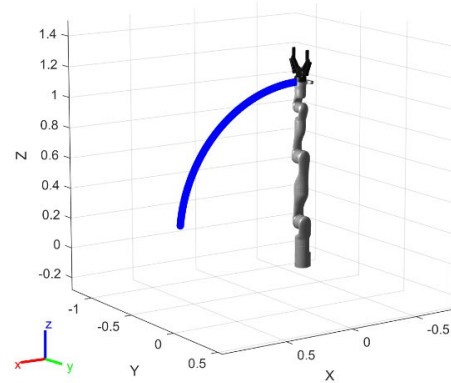
Figure 30. Kinova arm grasping the aluminum bar test object.

An updated URDF model was used for tests that incorporated the test object. This model included the geometric and inertial properties of the test object and adjusted the adaptive gripper's configuration. This URDF file is provided in Appendix B.

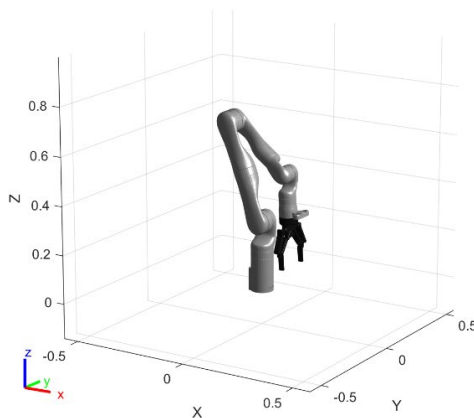
Four tests were conducted using the real Kinova arm. Tests 1 and 2 were isolated Kinova arm tests, while tests 3 and 4 involved the test object. Test 1 demonstrated a single actuator maneuver, restricting the Kinova arm to a planar range of motion, rotating about joint two. Test 2 showcased a multi-actuator maneuver featuring five actuators and demonstrating a more dynamic control trajectory. Figure 31 and Table 6 provide specific information for tests 1 and 2.



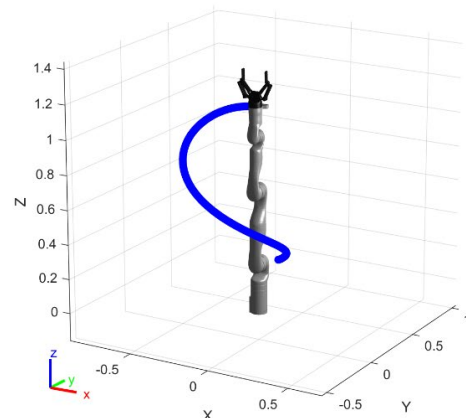
a) Test 1 start position.



b) Test 1 end position and motion path.



c) Test 2 start position.



d) Test 2 end position and motion path.

Figure 31. Visualization of the real-life Kinova arm test maneuvers.

Table 6. Isolated real-life Kinova arm test maneuvers.

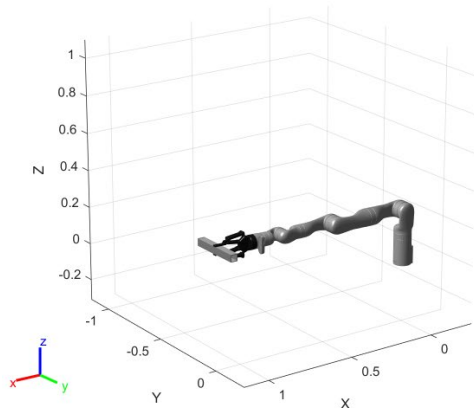
Joint	Test 1: Single Actuator Maneuver		Test 2: Multi-actuator Maneuver	
	Start Position [deg]	End Position [deg]	Start Position [deg]	End Position [deg]
1	0.0	0.0	0.0	0.0
2	90.0	0.0	340.0	0.0
3	0.0	0.0	180.0	0.0
4	0.0	0.0	214.0	0.0
5	0.0	0.0	0.0	0.0
6	0.0	0.0	310.0	0.0
7	270.0	270.0	90.0	0.0

Figure 31 visually presents the control trajectories for each test, illustrating the Kinova arm's initial and final configurations. The motion paths for tests 1 and 2 are highlighted with blue dots. Table 6 provides detailed information for each test case, displaying the starting and ending positions.

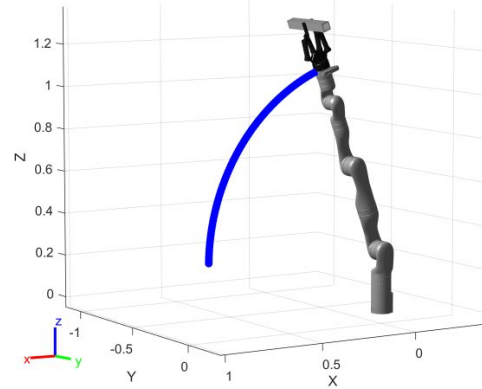
Test 3 was similar to test 1, except it incorporated the test object. Its maneuver consisted of lifting the test object from a horizontally extended position to a near-vertical final configuration. For safety purposes, the Kinova arm was not commanded to a fully vertical position to prevent the test object from accidentally falling on the Kinova arm. Test 4 exercised the most intricate maneuver, employing six actuators. This maneuver was designed to emulate a real-life task. The Kinova arm was commanded to lower the test object from a near-vertical configuration and maneuver the object to a position below the base link for the Kinova arm. Table 7 and Figure 32 supply specific information for tests 3 and 4. Table 7 presents detailed information for the starting and ending positions of the Kinova arm during these tests. Figure 32 visually represents the control trajectory for these tests, illustrating the Kinova arm's initial and final configurations. The motion paths for tests 3 and 4 are highlighted with blue dots.

Table 7. Test object real-life Kinova arm test maneuvers.

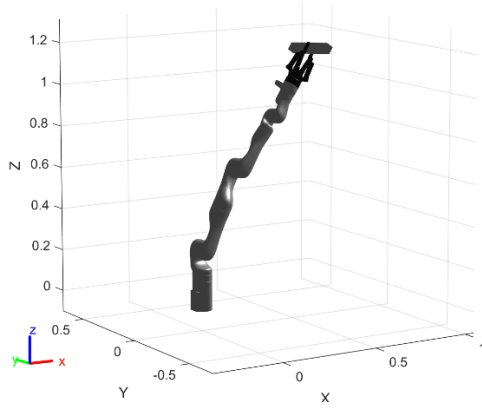
Joint	Test 3: Single Actuator Maneuver with Test Object		Test 4: Multi-actuator Maneuver with Test Object	
	Start Position [deg]	End Position [deg]	Start Position [deg]	End Position [deg]
1	0.0	0.0	340.0	25.0
2	90.0	20.0	20.0	90.0
3	0.0	0.0	0.0	45.0
4	0.0	0.0	0.0	90.0
5	0.0	0.0	0.0	10.0
6	0.0	0.0	0.0	320.0
7	0.0	0.0	0.0	0.0



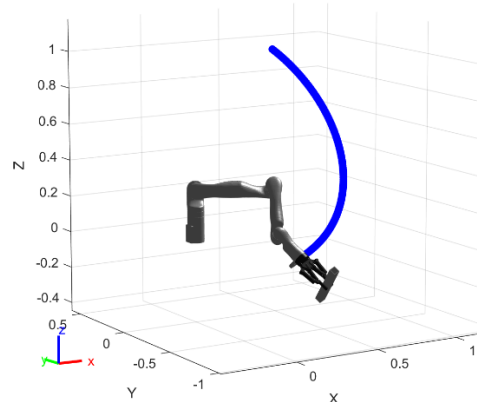
a) Test 3 start position.



b) Test 3 end position and motion path.



c) Test 4 start position.

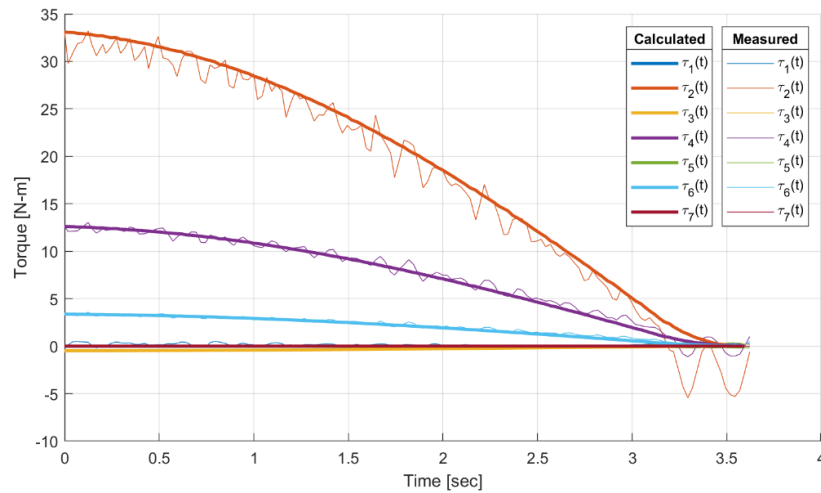


d) Test 4 end position and motion path.

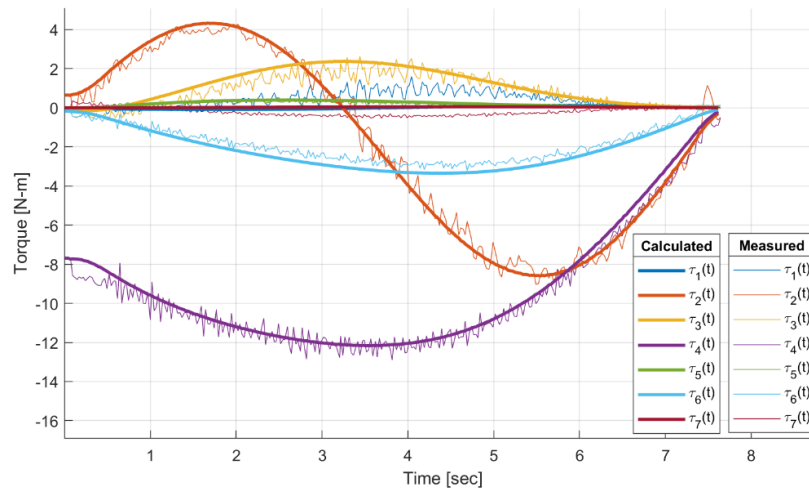
Figure 32. Visualization of the real-life Kinova arm test maneuvers utilizing the test object.

D. HARDWARE-IN-THE-LOOP RESULTS

During simulation tests 1 and 2, a strong correspondence was observed between the predicted and actual torque values. Some irregular readings were identified, specifically in the measured torque histories of joints one and seven during test 2. Despite these discrepancies, the measured torque histories support the feasibility of the Kinova arm simulation model. Figure 33 presents the calculated and measured torque histories.



a) Test 1 calculated vs. measured torques.



b) Test 2 calculated vs. measured torques.

Figure 33. Calculated torques vs. measured real-life Kinova arm torques.

The sensor drift in joint one during test 2 is likely attributed to the joint's inactivity during the control maneuver, which led to the sensor drifting naturally throughout the commanded maneuver. For joint seven, the recorded sensor noise displayed a variation of ± 0.25 N-m, while the calculated torques remained below 0.1 N-m. Thus, no meaningful analysis can be derived from the measured torque history of joint seven. For further analysis tests 1 and 2, the MAE was calculated to compare the calculated and simulated torques. The findings are summarized in Table 8.

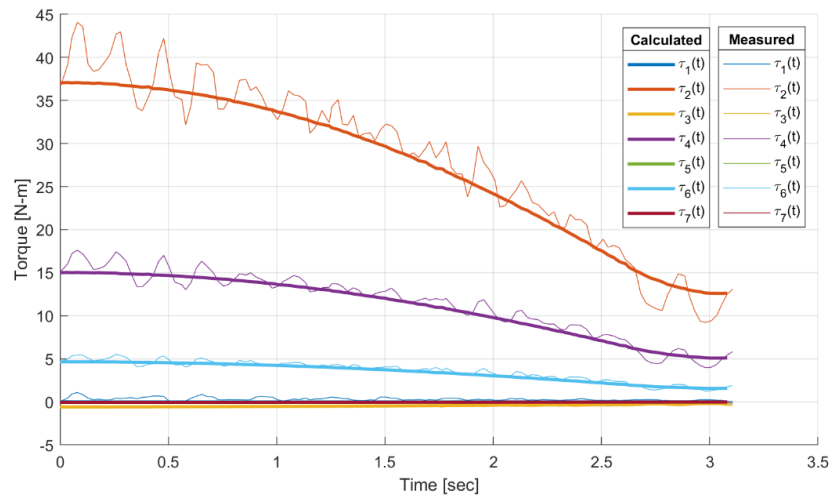
Table 8. Mean average error between the calculated and measured real-life Kinova arm torques.

Joint	Test 1: Joint Torque MAE [N-m]	Test 2: Joint Torque MAE [N-m]
1	0.1725	0.4073
2	1.3516	0.3638
3	0.0718	0.3040
4	0.4172	0.4138
5	0.0780	0.0663
6	0.1541	0.3184
7	0.0148	0.2640

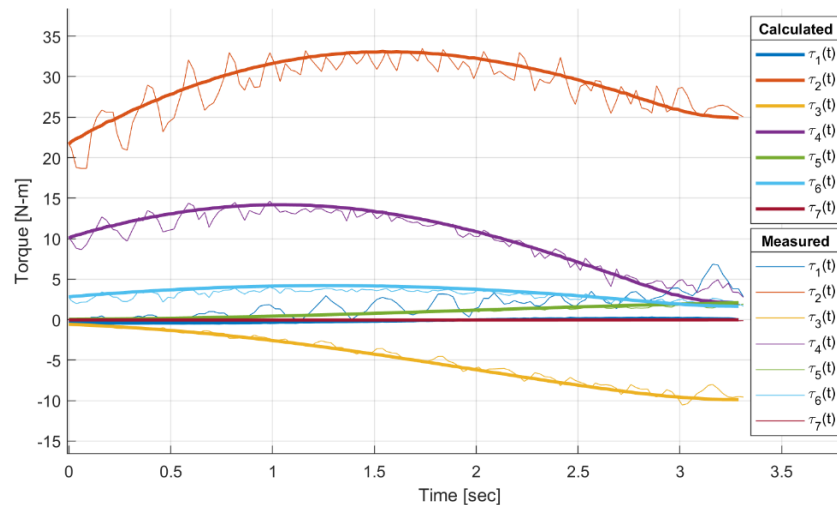
Table 8 reveals that the maximum MAE was 1.3516 N-m for joint two in test 1 and 0.4138 N-m for joint four in test 2. The MAE observed in joint two of test 1 can be attributed to the Kinova arm's attachment to the mobile table and its configuration during the executed maneuver. With the arm fully extended, joint two bore most of the Kinova arm's mass. As the table vibrated, the fully extended mass of the Kinova arm induced oscillations, consequently affecting the sensor readings for joint two. Due to the absorption of most vibrations in joint two, the recorded torque value deviated as much as ± 2 N-m from the calculated value.

The MAE for joint four in test 2 was only slightly higher than other recorded MAEs and does not raise concerns about the validity of the test 2 results. The average MAE in test 2 was likely lower than in test 1 because the Kinova arm's mass distribution was more centralized around the center of mass. This mass distribution led to a reduced impact of the mobile table vibrations on the capture sensor data.

Simulation tests 3 and 4 also demonstrated promising results. Again, some discrepancies were observed in the measured torque histories; however, results supported the feasibility of the simulated Kinova arm model. Additionally, the simulated Kinova arm appears to have correctly modeled the dynamic interaction of a test object in addition to the dynamics of the Kinova arm. Figure 34 displays the calculated and measured torque histories.



a) Test 3 calculated vs. measured torques.



b) Test 4 calculated vs. measured torques.

Figure 34. Calculated vs. measured torques for tests utilizing the test object.

The only irregular sensor reading detected between the two test cases was the measured torque history of joint one in test 4. During the commanded maneuver, the calculated joint torque was below 0.5 N-m. However, the sensor noise observed in the measured torque history of joint one displayed a variation of ± 1.5 N-m, rendering any meaningful analysis of its measured torque history unattainable. To reinforce the findings depicted in Figure 34, the MAE was calculated for both test cases to compare the calculated and measured simulation torques. These MAE calculations are summarized in Table 9.

Table 9. Mean average error between the calculated and measured real-life torques for tests utilizing the test object.

Joint	Test 3: Joint Torque MAE [N-m]	Test 4: Joint Torque MAE [N-m]
1	0.3327	1.8568
2	1.8624	1.3981
3	0.1040	0.3430
4	0.7698	0.7849
5	0.0925	0.1123
6	0.2927	0.4450
7	0.0313	0.0492

Table 9 reveals that the highest MAE was 1.8624 N-m for joint two in test 3 and 1.8568 N-m for joint one in test 4. The MAE observed in joint two of test 3 closely resembles the MAE noted in joint two of test 1. Given that tests 1 and 3 involved similar maneuvers, the correlation of the high MAE was not unexpected. The cause for this high MAE is likely attributed to the oscillatory nature of the Kinova arm. When comparing the MAEs of joint two between tests 1 and 3, the MAE recorded for test 3 was approximately 38% higher than the observed MAE in test 1. This increase is due to the added mass from the test object implemented in test 3. Furthermore, the sensor variation for joint two in test 3 rose to approximately ± 3 N-m, which was higher than the observed variation in test 1.

The highest MAE calculated during test 4 in joint one is likely attributed to the previously mentioned sensor drift. The sensor variation in this joint was significantly larger than the calculated joint torque, resulting in a larger MAE. The second-highest MAE of

test 4 was recorded for joint 2, with a value of 1.3981 N-m. This elevated MAE results from the same oscillatory behavior of the Kinova arm observed in joint two for tests 1 and 3. Neither of the high MAE values observed in test 4 point to infeasibility within the simulated Kinova arm model.

V. MODELING AND SIMULATION OF THE FULL FSS SYSTEM

A. FSS BASE MODEL DEVELOPMENT

Before developing a comprehensive model for the entire FSS system, it was crucial first to create and test the FSS base model. Based on insights from Chapter III, an MBD model was chosen to represent the FSS base. This choice established that the final FSS system model would adopt an MBD architecture, enabling seamless integration with the validated Kinova arm MBD model. The MBD architecture was favored due to its extensive customization capabilities within the Simscape environment and its utilization of Simulink's built-in physics engine. Had this research employed simulation software such as ROS-Gazebo, a URDF representation would have been a suitable alternative for modeling the FSS base.

The process of modeling the FSS base started by leveraging an existing CAD model, encompassing the geometry, mass distribution, and inertial characteristics of the FSS base. The CAD model was first converted into an STL file to create a reference geometry for the base's modeled properties. This STL file defined the visual and collision geometries of the FSS base. Following this, the mass and inertial values of the FSS base were added. The mass values and inertial characteristics were based on calculations determined within the CAD model. Finally, the reference frame for the FSS base was established, defining its body-fixed coordinate system at its COM.

Other block definitions were also needed to complete the FSS base MBD model. First, a motion constraint block was added, limiting the FSS base's motion to 3 DoF. This block controlled the FSS base's translational movement along the x and y-axes and rotation about the z-axis. An external force block was also added to model wrench inputs, or forces and torques, applied to the FSS base. Lastly, the world setup block definitions required by Simscape were connected to the 3 DoF block. These blocks included defining the granite table visualization block, the inertial reference frame, the solver configuration, and the environmental parameters. Figure 35 shows the completed FSS base MBD model. Figure 36 provides a visualization of the modeled FSS base.

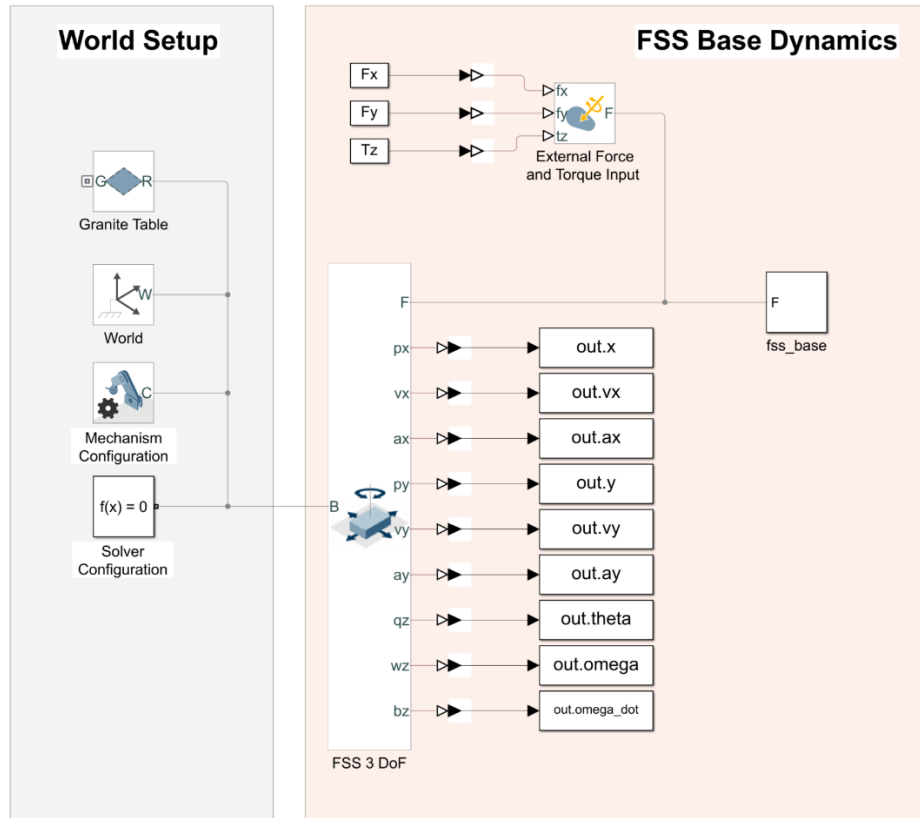


Figure 35. The developed MBD model for the FSS base.

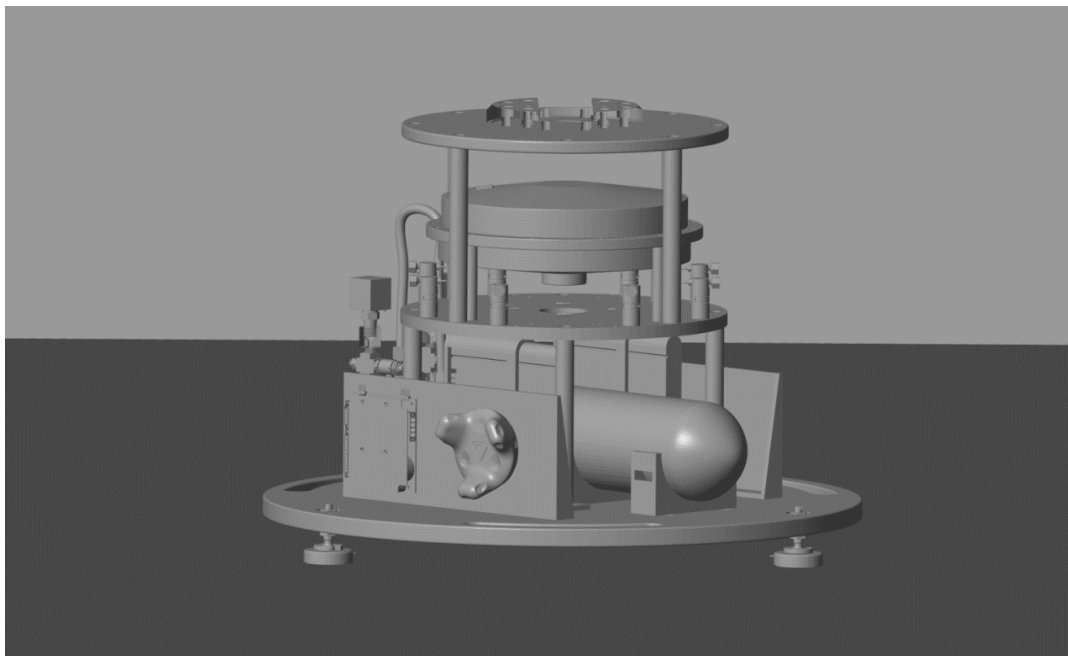


Figure 36. Visualization of the FSS base defined by the MBD model.

B. FSS BASE SIMULATION TESTS

To evaluate the performance of the MBD model for the FSS base, a series of tests were carried out against a dynamical model encompassing the FSS base's motion. The dynamical model, consisting of six equations, captured the FSS base's translational and rotational motion in response to wrench inputs. This model, refined based on the work completed in [57], is detailed in the FSS base dynamical model

$$\begin{aligned} \mathbf{x}^T &= [x, y, v_x, v_y, \theta, \omega] & \mathbf{u}^T &= [F_x, F_y, \tau_z] \\ \text{FSS Base Dynamical Model} &\left\{ \begin{aligned} \dot{x} &= v_x \cos \theta - v_y \sin \theta \\ \dot{y} &= v_x \sin \theta + v_y \cos \theta \\ \dot{v}_x &= \omega v_y + \frac{F_x}{m} \\ \dot{v}_y &= -\omega v_x + \frac{F_y}{m} \\ \dot{\theta} &= \omega \\ \dot{\omega} &= \frac{\tau_z}{I_{zz}} \end{aligned} \right. \end{aligned} \quad (3)$$

comprising six state variables, $\mathbf{x}$, and three wrench inputs or control variables, $\mathbf{u}$. The state variables included x and y representing the position of the FSS base along the x and y axes, v_x and v_y describing the FSS base's velocity, θ indicating the angular attitude along the z -axis, and ω denoting the angular rate. For the control variables, F_x and F_y represented external forces, while τ_z described external torques acting on the FSS base. Control inputs were defined in terms of the body-fixed frame of the FSS base.

Two additional variables, I_{zz} and m , were necessary for the dynamical model. I_{zz} represented the moment of inertia along the z -axis, while m denoted the mass of the FSS base. I_{zz} was assigned a value of 1.203 kg-m², and m was set to 64.17 kg to align with the inertial parameters of the MBD model. This complete dynamical model was implemented within an ode45 propagation function with a time step of 0.20 seconds. The MATLAB script used to define this function can be found in Appendix F.

Three tests were conducted to compare the MBD model with the dynamical model of the FSS base. These tests aimed to verify the translational and rotational motion of the FSS base resulting from wrench inputs, as predicted by the MBD model. Wrench inputs were designed to simulate the thrust generated by gas thrusters and the torque generated by

the RW located on the FSS base. The configurations for these wrench inputs are detailed in Table 10.

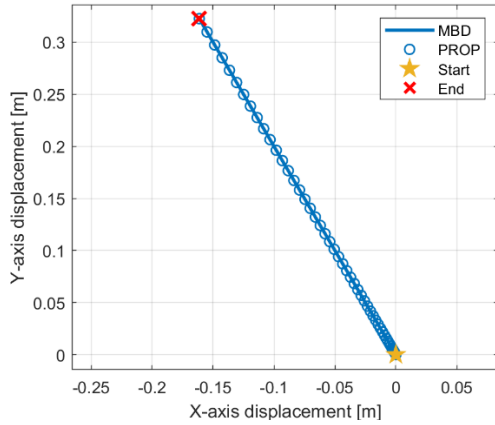
Table 10. Isolated FSS base simulation testing parameters.

Wrench	Test 1: Translation	Test 2: Rotation	Test 3: Combined
F_x [N]	-0.2	0.0	0.4
F_y [N]	0.4	0.0	0.4
τ_z [N-m]	0.0	0.2	-0.2

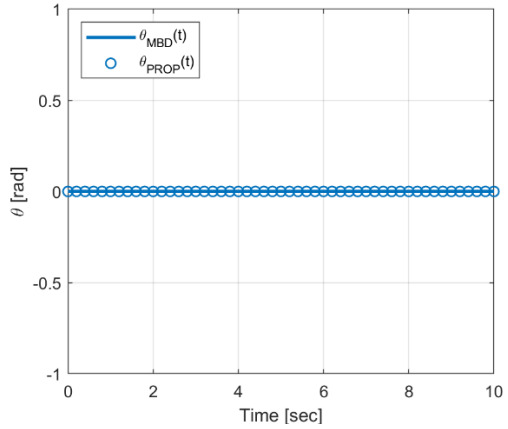
Test 1 aimed to validate the motion resulting from gas thrusters on the FSS base. This test applied force along the x and y axes while maintaining zero torque on the FSS base. Test 2 sought to validate the rotational motion resulting from an applied torque. The applied forces were set to zero, and a torque of 0.2 N-m was applied to the FSS base. Test 3 applied all three wrench inputs, facilitating a complex response of FSS base. Appendix G details the MATLAB script simulating the MBD and dynamical model of the FSS base.

C. FSS BASE SIMULATION RESULTS

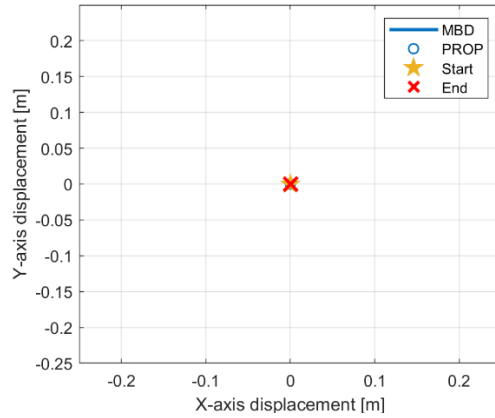
Testing revealed a strong correlation between the MBD model and the propagated dynamical model of the FSS base. The calculated trajectories and attitude histories of both models were plotted together, providing a benchmark for evaluating various test scenarios. Figure 37 displays the results of the plotted trajectories and attitude histories for both models. Figure 37 shows a strong agreement between the two models, revealing nearly identical matches in overall shape and magnitude. To quantitatively compare the models, the absolute error at the final time, t_f , was calculated for each trajectory and attitude history across all three test cases. The findings are summarized in Table 11.



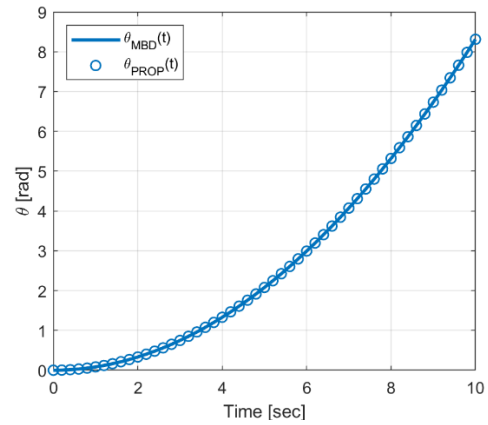
a) Test 1 positional path.



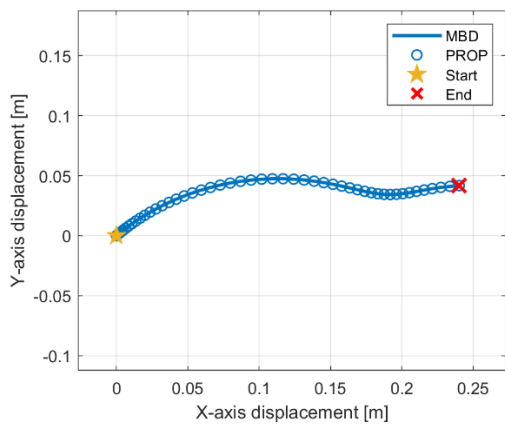
b) Test 1 attitude vs. time.



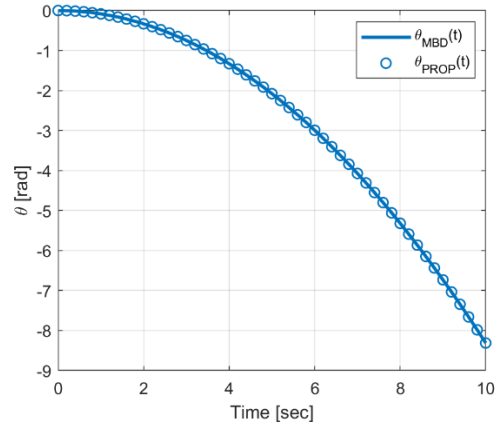
c) Test 2 positional path.



d) Test 2 attitude vs. time.



e) Test 3 positional path.



f) Test 3 attitude vs. time.

Figure 37. Simulated FSS base MBD vs. propagated dynamical model results.

Table 11. Absolute error at t_f for isolated FSS base simulation tests.

Absolute Error	Test 1: Translation	Test 2: Rotation	Test 3: Combined
$x(t_f)$ [m]	$8.3267 * 10^{-17}$	$1.1752 * 10^{-15}$	$6.2320 * 10^{-5}$
$y(t_f)$ [m]	$4.4187 * 10^{-14}$	$2.5923 * 10^{-15}$	$8.2949 * 10^{-5}$
$\theta(t_f)$ [m]	$1.3756 * 10^{-17}$	$1.4158 * 10^{-10}$	$1.4158 * 10^{-10}$

The calculated absolute errors at t_f for each test case demonstrated that the MBD models strongly correlated with the results calculated in the dynamical model. In test case 1, the maximum absolute error was 4.4187×10^{-14} for $y(t_f)$. Test 2 showed the least favorable performance in $\theta(t_f)$, with an absolute error of 1.4158×10^{-10} . For test 3, the highest calculated absolute error was 8.2949×10^{-5} for $y(t_f)$. These absolute errors indicate there were no significant differences between the MBD and dynamical models of the FSS base. The minimal errors and close alignment of the plotted results suggest that both models accurately and independently replicate the same results for the simulated FSS base.

D. FULL FSS SYSTEM MODEL DEVELOPMENT

Having independently validated and tested the MBD models for the Kinova arm and FSS base, the two models were integrated to form the complete FSS system. Again, an MBD model architecture was chosen to represent the entire system. To connect the validated Kinova arm model with the FSS base, the Kinova arm's base link block definitions needed to be modified. In the original Kinova arm model, the base link was inertially fixed to the simulation environment. Within the full FSS system model, this base link connection was altered to rigidly connect the Kinova arm base link to the FSS base. This rigid connection between the components dynamically coupled the Kinova arm and the FSS base.

A Kinova arm trajectory planning block was also incorporated into the simulation environment to control the Kinova arm model. This block automatically calculated the joint space histories of the Kinova arm based on user-defined waypoints. Figure 38 displays the

completed full FSS system MBD model. Figure 39 shows a visualization of the full FSS system defined by the MBD model.

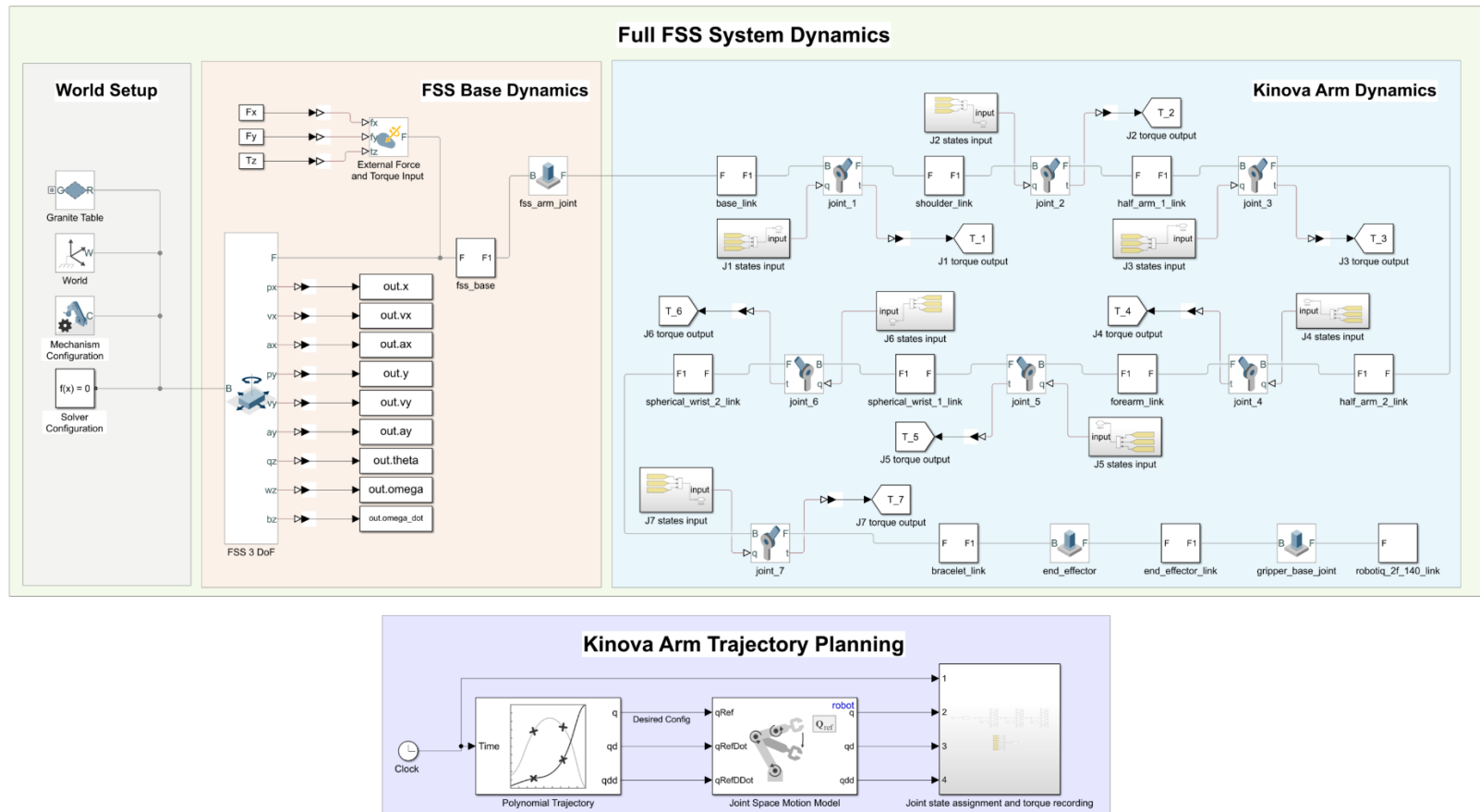


Figure 38. The developed MBD model for the full FSS system.

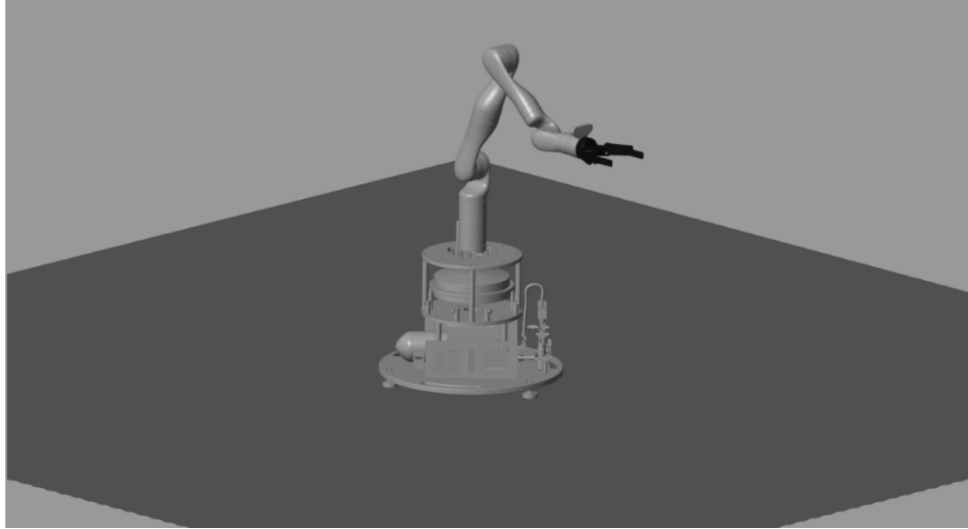
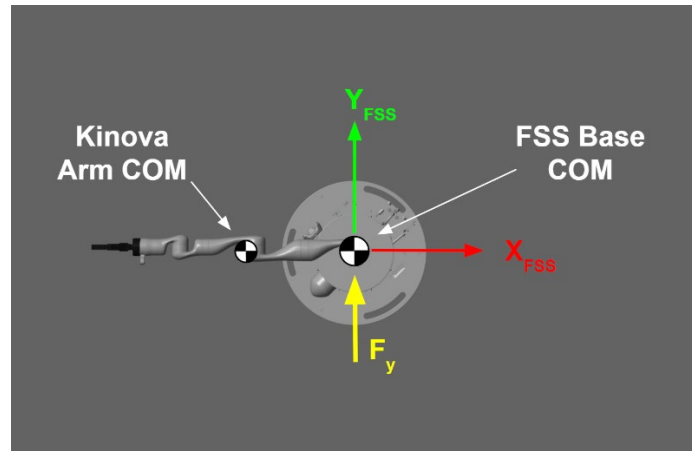


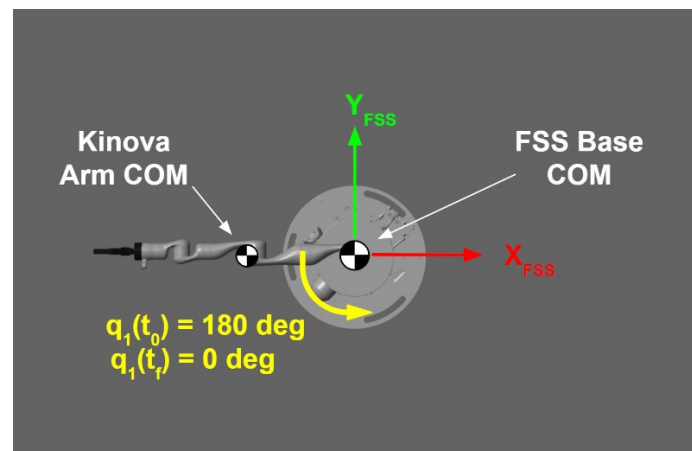
Figure 39. Visualization of the FSS system defined by the MBD model.

E. FULL FSS SYSTEM SIMULATION TESTS

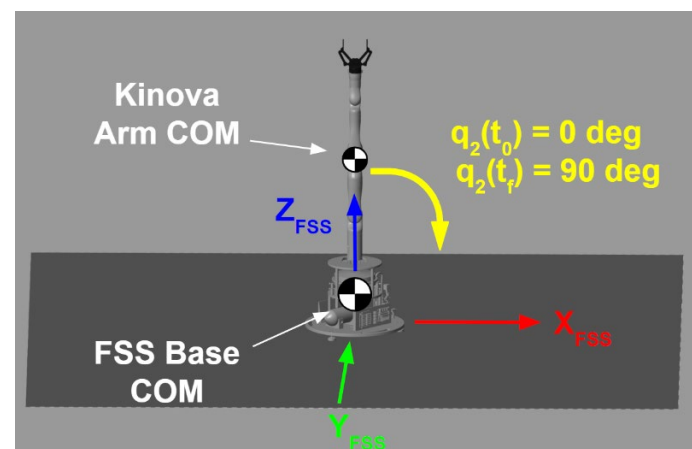
Due to the dynamic coupling between the Kinova arm and the FSS base, creating a secondary model for comparison with the MBD model was deemed infeasible. The construction of an additional model would have been time prohibitive and redundant, as the MBD models for the Kinova arm and FSS base had been thoroughly tested. Without a secondary model, the FSS system MBD model validation tests concentrated on demonstrating test cases with predictable outcomes. Three tests were conducted to simulate the FSS system's response to Kinova arm maneuvers and wrench inputs applied to the FSS base. Note that the wrench inputs did not stabilize or correct the FSS base's position, leaving the base in an uncontrolled, floating mode. Therefore, the FSS base could react to forces and torques generated by the Kinova arm and wrench inputs. Figure 40 visually depicts the three test cases, while Table 12 outlines the Kinova arm maneuvers and wrench configurations for each test.



a) Test 1 visualization.



b) Test 2 visualization.



c) Test 3 visualization.

Figure 40. Visualization of the full FSS system simulation testing.

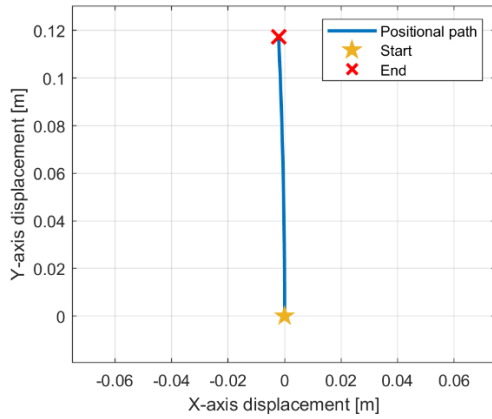
Table 12. Full FSS system simulation testing parameters.

	Test 1: Fixed Arm Maneuver		Test 2: Arm Swing Maneuver		Test 3: Arm Bend Maneuver	
	Kinova Arm Control		Kinova Arm Control		Kinova Arm Control	
Joint	$\mathbf{q}(t_0)$ [deg]	$\mathbf{q}(t_f)$ [deg]	$\mathbf{q}(t_0)$ [deg]	$\mathbf{q}(t_f)$ [deg]	$\mathbf{q}(t_0)$ [deg]	$\mathbf{q}(t_f)$ [deg]
1	180.0	180.0	180.0	0.0	0.0	0.0
2	90.0	90.0	90.0	90.0	0.0	90.0
3	0.0	0.0	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	0.0	0.0	0.0
5	0.0	0.0	0.0	0.0	0.0	0.0
6	0.0	0.0	0.0	0.0	0.0	0.0
7	0.0	0.0	0.0	0.0	0.0	0.0
Wrench	FSS Base Actuator Control		FSS Base Actuator Control		FSS Base Actuator Control	
F_x Thrust [N]	0.0		0.0		0.0	
F_y Thrust [N]	1.0		0.0		0.0	
τ_z [N-m]	0.0		0.0		0.0	

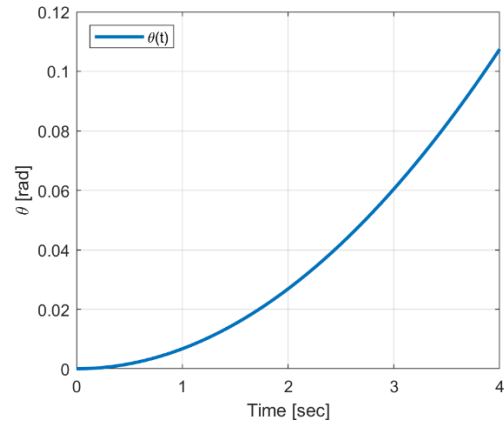
Test 1 assessed the FSS system during a static Kinova arm maneuver, with a wrench input of 1 N applied to the positive y-axis of the FSS base. The expected outcome from this test was a predominant translation of the FSS base along the y-axis, with the Kinova arm's COM location causing a counterclockwise rotation of the FSS base. Test 2 involved a Kinova arm swing maneuver, rotating the Kinova arm's first joint in a counterclockwise direction. Without wrench inputs, the FSS base was predicted to rotate in the opposite, clockwise direction with minimal lateral translation. In Test 3, no wrench inputs were applied to the FSS base. The Kinova arm executed an arm bending maneuver, rotating around the arm's second joint and transitioning from a vertical to a horizontal pose. It was anticipated that the FSS base would move along the negative x-axis with minimal z-axis rotation and y-axis translation.

F. FULL FSS SYSTEM SIMULATION RESULTS

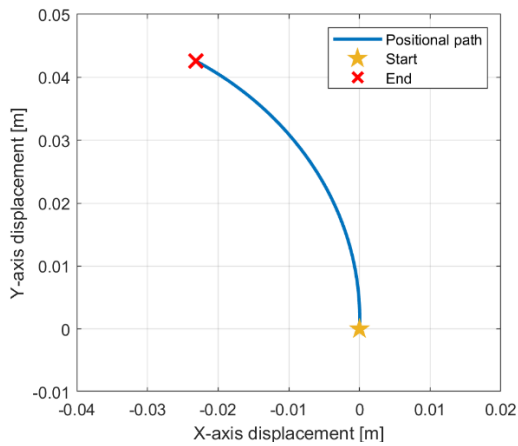
The results from the three test cases were consistent with the expected behavior of the MBD model. Figure 41 displays the positional path and attitude of the FSS base during the commanded maneuvers.



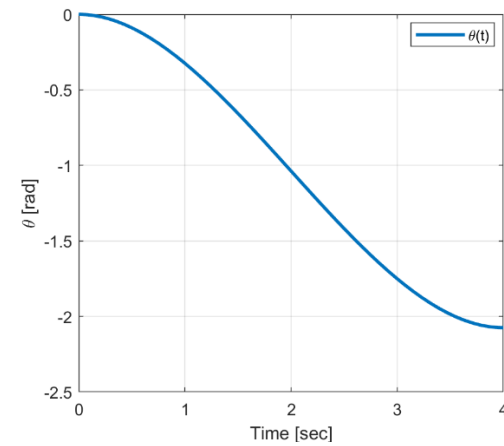
a) Test 1 FSS base positional path.



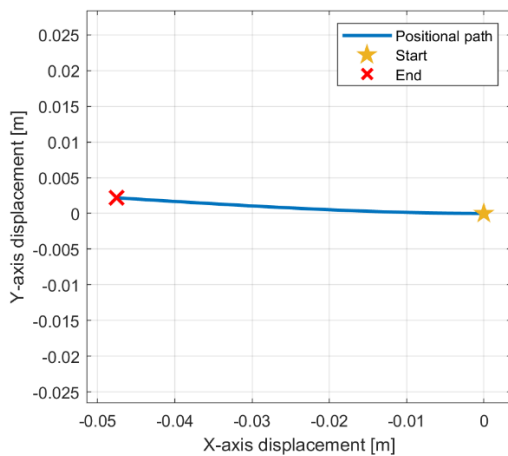
b) Test 1 FSS base attitude vs. time.



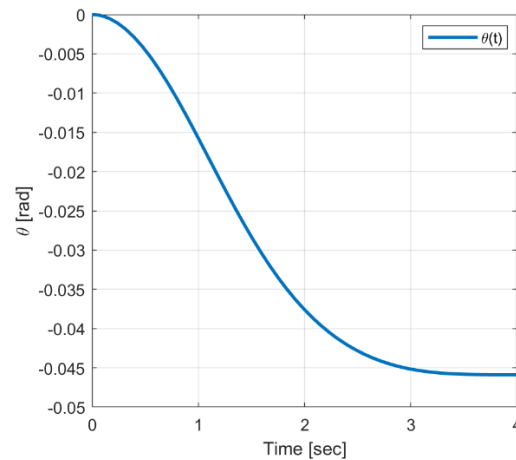
c) Test 2 FSS base positional path.



d) Test 2 FSS base attitude vs. time.



e) Test 3 FSS base positional path.



f) Test 3 FSS base attitude vs. time.

Figure 41. Full FSS system simulation results.

In test 1, the FSS base was expected to translate along the positive y-axis and rotate counterclockwise due to the force F_y . As seen in Figure 41a, the FSS base reached a final position of $y_f = 0.7217$ meters. The observed rotation of the FSS base resulted from the Kinova arm's COM location. The Kinova arm's COM shifted the entire FSS system's COM to be positioned along the negative x-axis. Consequently, the line of action of the F_y force did not go through the FSS system's COM, causing a positive torque to be generated. The MBD model established the final attitude of the FSS base at $\theta_f = 0.6716$ radians.

For test 2, the results showed minimal translation of the FSS base, as no external force was exerted on the system. However, Figure 41d indicates that the Kinova arm's commanded maneuver led to a substantial clockwise rotation of the FSS base. As the Kinova arm rotated counterclockwise, the torque generated by joint actuator 1 caused a clockwise rotation of the FSS base. The final attitude of the FSS base was calculated to be $\theta_f = -2.0750$ rad. Under the conservation of momentum, the observed rotation of the FSS base matched the predicted behavior for test 2.

In Test 3, Figure 41e illustrates that the FSS base predominantly translated along the negative x-axis, experiencing a minimal change in attitude. The base started at a position of $(x_0, y_0) = [0, 0]$ m and concluded at $(x_f, y_f) = [-0.0475, 0.0022]$ m. As indicated in Figure 40c, the Kinova arm's COM location shifted along the positive x-axis, causing the FSS base to move in the opposite direction. This conservation of linear momentum reinforced the predicted outcomes for this test case.

Based on these three test cases, the fully integrated model of the entire FSS system exhibited the anticipated dynamic behavior. The model demonstrates that it captures the resulting base motion induced by the Kinova arm maneuvers and accurately calculates the dynamic effects of wrench inputs on the FSS base.

THIS PAGE INTENTIONALLY LEFT BLANK

VI. FULL FSS SYSTEM HARDWARE-IN-THE-LOOP TESTING

A. REAL-LIFE FULL FSS SYSTEM

After completing simulation testing of the FSS MBD model, the model's performance was compared against the actual FSS system. During the development of the FSS system model, the construction of the FSS base was mostly finished, allowing for the opportunity to conduct hardware-in-the-loop testing. Hardware-in-the-loop testing enabled a performance comparison between the simulated MBD model and the real-world FSS system. The Kinova robotic arm, previously used in other tests, was mounted onto the FSS base for the testing process. The assembled FSS system is depicted in Figure 42.



Figure 42. Full FSS system pictured in the NPS-SRL.

Figure 42 displays the FSS system with the FSS base's planned structural hardware and most of the non-structural components. All available components were attached to the

FSS base to ensure the overall mass distribution closely mirrored the estimated CAD model. During testing, the air supply necessary for operating the air-bearings was functional, allowing the FSS base to float freely in the testing environment. It is important to note that Figure 42 shows the FSS system on the FSS development granite table, a smaller testbed located in the NPS-SRL. Although the POSEIDYN testbed will eventually serve as the testing area for the FSS, the decision was made to conduct initial tests on the smaller granite table.

Given the state of the FSS system, testing faced several limitations. The FSS system did not have all the electronic hardware necessary for controlling the RW and gas thrusters, resulting in an uncontrolled floating mode for the FSS base. Consequently, testing focused on Kinova arm maneuvers in this floating base environment. Moreover, the electrical and communication systems for the Kinova arm were not yet integrated, requiring the use of external ethernet communications and power during testing. Testing setups were chosen to both validate the MBD model's performance and accommodate the limitations of the FSS system.

B. COMMAND GENERATION, SENSOR RECORDING, AND TRACKING

Chapter IV described the control of the Kinova robotic arm using a MATLAB script detailed in Appendix D. This script was utilized again for the FSS system testing. The script established a control sequence using pre-selected waypoints for each arm maneuver. Within the MATLAB script, a polynomial trajectory planner computed the joint space control trajectories necessary to execute the commanded Kinova arm maneuvers. Simultaneously, sensor data from the arm, including joint positions, velocities, and torques, were collected and saved in Excel files for further analysis.

The initial plan for tracking the position and attitude of the FSS base involved utilizing a VIVE tracking system, specifically calibrated for the FSS development granite table. This tracking system determines the FSS base's position and attitude using the FSS Base's VIVE tracker. Infrared light is emitted from two base stations surrounding the granite table, directed at the VIVE tracker's reflective markers. The base stations measure the direction and time taken for the light to reflect back, enabling the determination of

position and attitude information. However, due to technical issues with the computer running the software, the VIVE tracking system was not utilized.

As an alternative, a camera-based approach was employed to track the position and attitude of the FSS base. The camera-based approach involved mounting a GoPro above the granite table to record video footage of tests conducted on the table. The changes in position and attitude of the FSS were captured in the video frames, providing a means to measure the FSS base's motion during testing. Figure 43 shows the camera mount and GoPro used for tracking the FSS during testing.

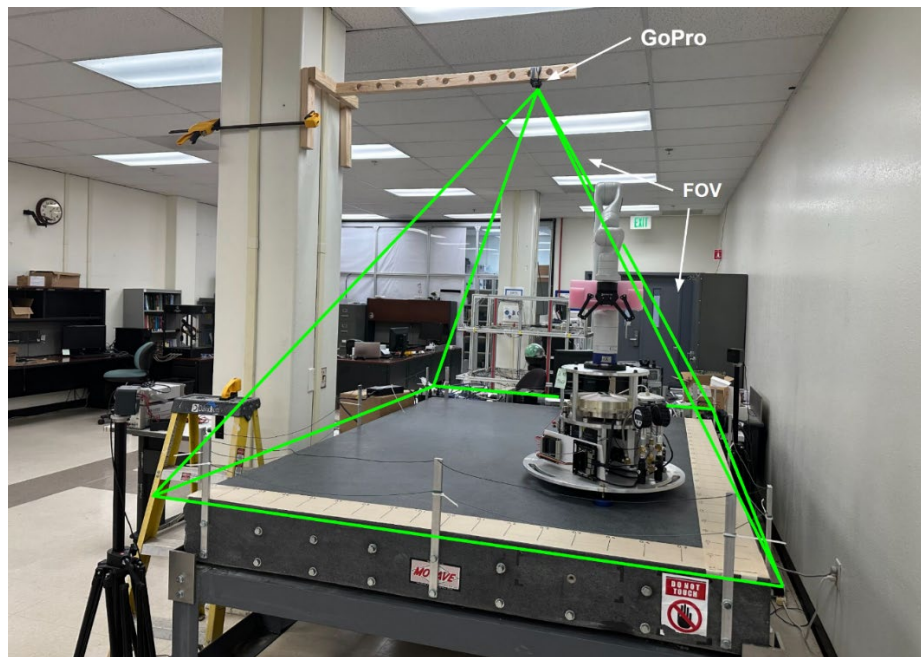


Figure 43. Camera mount and GoPro used in testing.

In Figure 43, the green lines denote the GoPro camera's field of view (FOV). The camera was positioned to capture the entire testing area of the granite table while maintaining an appropriate distance from the table surface. The camera was set to record at 1080p resolution at 30 frames per second. Given the GoPro's distance from the table and its recording resolution, it had a spatial resolution of approximately 0.001 meters per pixel of the granite surface. This spatial resolution meant that the GoPro could capture displacements of the FSS base greater than 1 mm.

To obtain position and attitude information, recorded videos were trimmed to focus on the time period when the FSS system was being tested. These trimmed videos were then converted into an image datastore, a file containing all the individual frames of a video. The MATLAB script for creating image datastores is detailed in Appendix I. However, the raw image datastores could not be directly used for position and attitude tracking due to lens distortion caused by the GoPro's wide field of view (FOV). This distortion occurred due to the wide FOV lens used by the GoPro, causing light to bend more at the edges than at the center of the lens. This bending of light created a barrel distortion in the captured video frames, making it difficult to measure the position and attitude of the FSS base accurately. Figure 44 illustrates an example of a raw video frame exhibiting this barrel lens distortion.

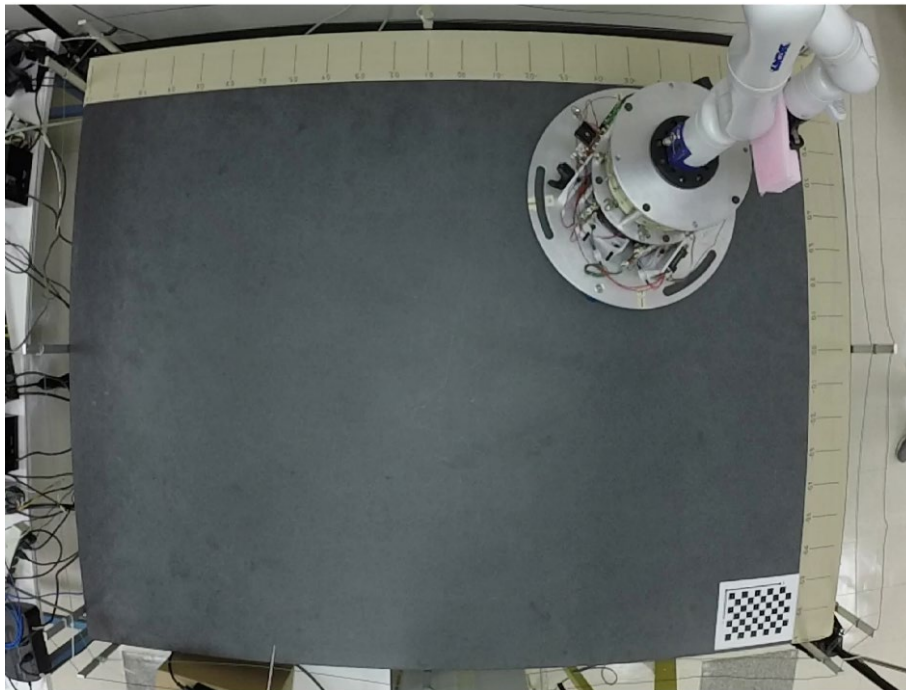


Figure 44. Barrel distortion of a raw video frame.

To correct the video frames, a MATLAB script leveraging the MathWorks Computer Vision Toolbox was developed to perform geometric camera calibration and correct the image datastores [58]. This calibration process estimated the GoPro's

parameters, such as lens type and sensor type, allowing for the reversal of image distortion. Camera correction parameters were derived for the GoPro using a series of calibration images. Calibration images utilized a checkboard pattern placed at the edges of the granite table, providing an easily identifiable pattern for image processing software to recognize. These images were processed by the MATLAB script, where an image detection algorithm identified distortions in the checkerboard pattern. This MATLAB script determined the camera calibration parameters and corrected the video frames captured during FSS testing. Figure 45 displays the corrected version of the image previously shown in Figure 44. Appendix J provides the MATLAB script used for correcting the image datastores.



Figure 45. Corrected video frame.

After correcting the image datastores, images were manually analyzed to measure the position and attitude of the FSS base. Fiducial markers placed on the edges of the granite table served as reference points, establishing the inertial reference frame for the table. Images were then individually evaluated using a trial-and-error process to determine the position and attitude of the FSS base relative to the reference frame. Only a subset of

images was analyzed, as it was found that using every 10th frame of the collected video provided an adequate measure of the changes in position and attitude. Linear interpolation was applied to the measured data points to estimate the dynamics between non-measured frames. Although time-consuming, this method accurately measured the position and attitude data for the FSS base. Appendix K and L detail the MATLAB scripts used to determine position and attitude information from the image datastores. An additional script in Appendix M generates movies of FSS system tests. This file takes the interpolated data produced by MATLAB script in Appendix L and converts it into a visual overlay of the collected image datastores. Figure 46 shows an example of a movie frame generated by the MATLAB script in Appendix M.

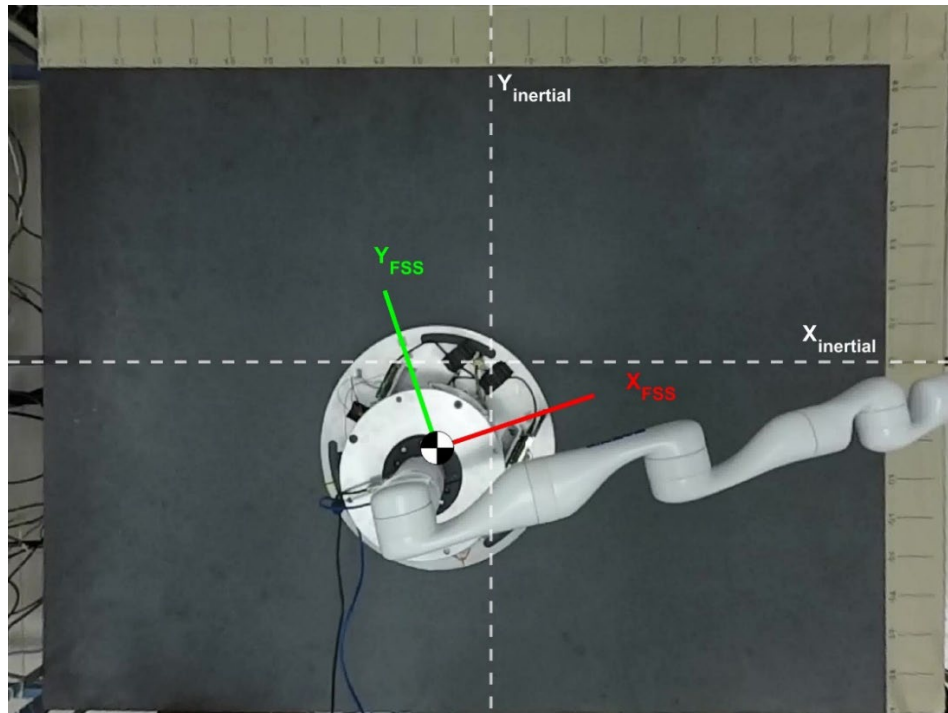
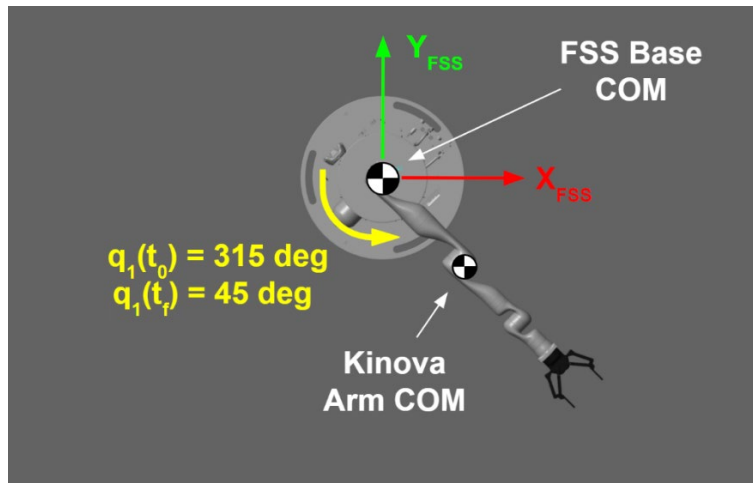


Figure 46. Screen capture from a generated movie file.

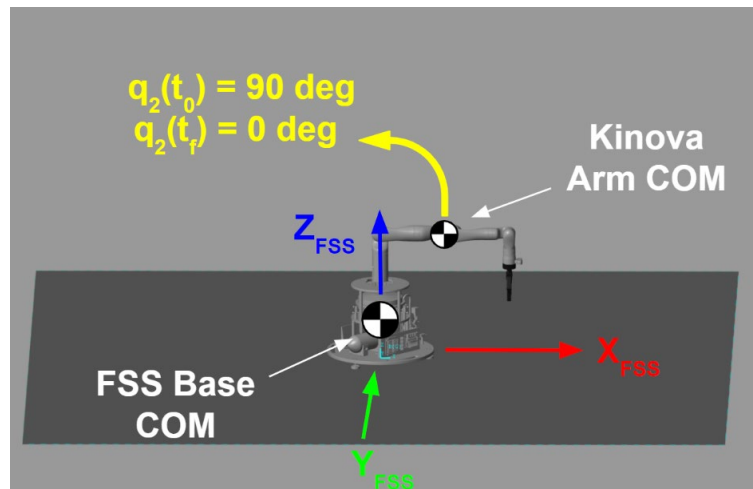
C. FSS HARDWARE-IN-THE-LOOP TESTS

As previously mentioned, the types of tests that could be conducted with the real FSS system were limited to Kinova arm maneuvering without control actuation of the base. Consequently, two tests were carried out: test 1 demonstrated a Kinova arm swing, while

test 2 involved a Kinova arm bending maneuver. These tests closely paralleled the testing conducted during the MBD model simulations, as the resulting dynamic motion of the floating FSS base was predictable. Figure 47 provides a visualization of the test cases, and Table 13 presents the testing parameters defining the Kinova arm control trajectory and the FSS base wrench inputs. These parameters were applied to both the real FSS system and the simulated MBD model for analytical comparison.



a) Test 1 visualization.



b) Test 2 visualization.

Figure 47. Visualization of the testing scenarios used with the real and simulated FSS system.

Table 13. Testing parameters for the real and simulated FSS system.

	Test 1: Arm Swing Maneuver		Test 2: Arm Bend Maneuver	
	Kinova Arm Control		Kinova Arm Control	
Joint	$\mathbf{q}(t_0)$ [deg]	$\mathbf{q}(t_f)$ [deg]	$\mathbf{q}(t_0)$ [deg]	$\mathbf{q}(t_f)$ [deg]
1	315.0	45.0	0.0	0.0
2	90.0	90.0	0.0	90.0
3	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	0.0
5	0.0	0.0	0.0	0.0
6	0.0	0.0	90.0	90.0
7	270.0	270.0	270.0	270.0
Wrench	FSS Base Actuator Control		FSS Base Actuator Control	
F_x Thrust [N]	0.0		0.0	
F_y Thrust [N]	0.0		0.0	
τ_z [N-m]	0.0		0.0	

In Test 1, Figure 47a shows the Kinova arm in a horizontally extended pose. The arm was commanded to rotate about its first joint, moving counterclockwise from a position of $q_1(t_0) = 315$ degrees to $q_1(t_f) = 45$ degrees. Without wrench inputs applied to the FSS base, the 90-degree rotation of the Kinova arm was expected to cause a clockwise rotation of the FSS base due to the conservation of angular momentum. A slight translation of the FSS base along the negative x and y axes was also anticipated, but the base's rotation was the test's primary focus.

For Test 2, Figure 47b displays the Kinova arm maneuvering from a horizontally extended position to a vertical pose, rotating about its second joint actuator. Note that in Table 13, joint 6 was commanded to maintain a 90-degree pose during testing. This configuration ensured that the real-life Kinova arm would maintain enough clearance from the camera rig used to capture position and attitude information. The expectation for this maneuver was that the FSS base would predominantly translate along the positive x-axis as the Kinova arm rotated from $q_2(t_0) = 90$ degrees to $q_2(t_f) = 0$ degrees. This translation of the FSS base was anticipated due to the conservation of linear momentum. The MATLAB script to conduct the FSS system tests is provided in Appendix N.

D. FSS HARDWARE-IN-THE-LOOP RESULTS

Comparing the performance of the real FSS system against the MBD model during tests 1 and 2, the measured behavior of the real FSS system showed a strong correlation with the calculated performance determined by the MBD model. Generally, the MBD model accurately predicted the direction of positional and attitude changes of the FSS base observed in the real FSS system. Although there were some differences in magnitude between the measured and calculated performance of the FSS base, these discrepancies do not indicate the infeasibility of the MBD model. The results for these two test cases are shown in Figure 48, while Table 14 presents the absolute error for the position and attitude of the FSS base at t_f .

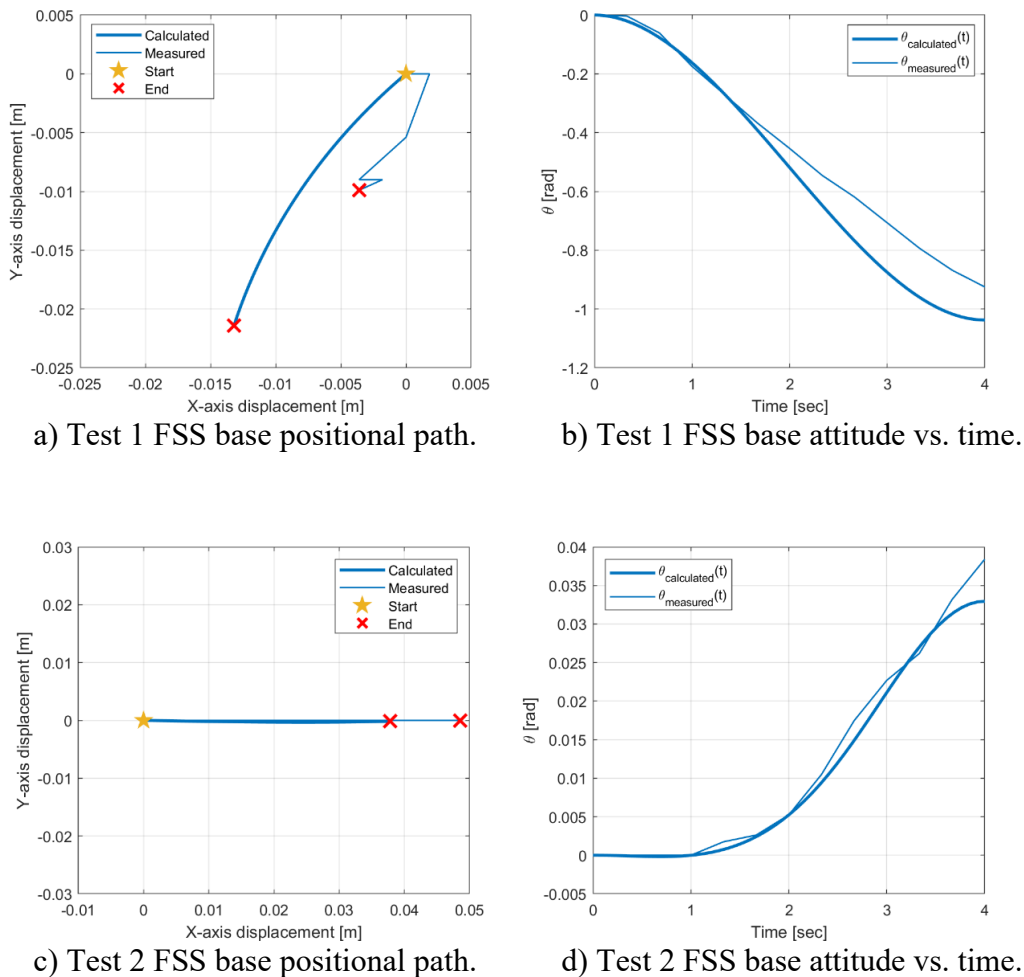


Figure 48. Measured vs. calculated FSS system testing results.

Table 14. Absolute error at t_f between the real FSS system and the MBD model simulation tests.

Absolute Error	Test 1: Arm Swing Maneuver	Test 2: Arm Bend Maneuver
$x(t_f)$ [m]	0.0096	0.0108
$y(t_f)$ [m]	0.0115	0.0002
$\theta(t_f)$ [rad]	0.1127	0.0054

Figure 48a displays the positional path of the FSS base, while Figure 48b presents the change in attitude of the FSS base during test 1. During this test, the real FSS system's attitude change closely resembled the MBD model's calculated general shape and magnitude results. The MBD model accurately calculated the FSS base's attitude change direction, with an absolute attitude error determined at $\theta(t_f) = 0.108$ rad compared to the measured performance. The MBD model also correctly predicted the direction of the real FSS system's translation, although the magnitude of the position change did not align with the calculated MBD model simulation. The absolute errors for x and y were 0.0096 m and 0.0115 m, respectively.

In Figure 48c, a predominant translation of the FSS base along the positive x-axis was evident in both the simulated and real FSS system. Despite some differences in the final state of $x(t_f)$, the MBD model captured the general dynamic behavior. The absolute error for $x(t_f) = 0.0108$ m. Figure 48d also reveals that the MBD model calculated a nearly identical attitude change of the FSS base compared to the measured FSS system. The magnitudes of the measured and calculated attitudes correlated closely. Although the final attitude for the measured and calculated FSS system did not match, test 2 still demonstrated the validity of the MBD model. The absolute attitude error was $\theta(t_f) = 0.0054$ rad.

VII. CONCLUSION

A. SUMMARY OF RESULTS

Overall, the developed MBD model effectively represented the measured and observed dynamic behavior of the actual FSS system, with only minor discrepancies in performance. The differences between the measured real FSS system and the calculated MBD model can be attributed to several factors. First, the precise inertial properties of the FSS base were only estimated through the CAD model and were never directly measured for the physical base. In the future, it would be beneficial to measure these parameters with the physical system and adjust the MBD model parameters accordingly.

The MBD model also did not account for the contact friction between the air-bearings and the granite surface. During the model development, it was assumed that the friction between the air-bearing and the granite surface would be negligible. However, friction was observed during testing as the FSS base moved along the surface. The granite table used for testing also had not been precisely leveled or cleaned, leading to imperfect test conditions. Future testing should ensure a clean test environment and proper calibration of the test bed for high-precision testing.

Challenges associated with the VIVE tracking system meant that developing a camera-based position and attitude tracking system introduced errors in measuring the FSS base's position and attitude during testing. Although efforts were made to accurately mark the FSS's position in the recorded video frames, the manual nature of the measurements introduced the possibility of user error. To improve the camera system, fiducial markers could be placed on the FSS base, allowing for image processing algorithms to detect and measure the FSS's position and attitude automatically.

Despite these factors, hardware-in-the-loop testing of the FSS base against the MBD model demonstrated the developed model's accuracy in predicting the FSS system's observed behavior. The developed MBD model holds promise for use in future research involving the FSS system.

B. FUTURE WORK

1. MBD Model Refinement and Upgrading

As highlighted in the summary of results, there are several areas in which the MBD model of the FSS system can be enhanced and refined. Beyond adjusting the contact friction of the air-bearings and updating the inertial properties of the FSS base, the MBD model could leverage more advanced features available within the Simscape environment. These improvements include accounting for the non-rigidity of robot links, specifying joint frictions, and utilizing sensor plugins for hardware-in-the-loop testing. Additionally, incorporating the physical interactions of the robotic gripper is essential for future use of the MBD model. An upgraded Robotiq gripper model could include adding its physical actuation, contact frictions, and variations of its inertial properties when simulated in different gripper configurations.

2. FSS Control Algorithm Development

Further work is necessary to develop controllers capable of operating the FSS system. Controllers for both the FSS base and the Kinova arm are required. The FSS base will likely operate in either a fixed base or flying mode, while controllers for the Kinova arm must be designed for specific object grasping or maneuvering tasks. The MBD model developed in this thesis can facilitate control algorithm development. By creating control algorithms within the MBD model, researchers can save time, minimize the hassle of setting up physical hardware testing, and iterate the design of controllers. Once the controllers satisfactorily control the MBD model of the FSS system, the MBD model can be replaced with the actual FSS system for real-world implementation.

3. Demonstrate Approach and Docking Maneuvering

Future research with the FSS system could also concentrate on demonstrating approach and docking maneuvers. Showcasing this advanced maneuvering would require a testing scenario involving the FSS system and a client spacecraft simulator. Initially, approach and docking maneuvers could be simulated within the MBD model before progressing to a full hardware-in-the-loop simulation. The FSS system's MBD model will

require implementing the proposed upgrades and control algorithm advancements to accomplish this. Moreover, if a virtual simulation is desired, efforts should be directed toward creating a virtual representation of a client spacecraft simulator. Demonstrating approach and docking maneuvering within the FSS system will significantly contribute to research in OOS capabilities.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX A. KINOVA ARM URDF FILE

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

<!-- This file was modified from the Kinova Robotics GitHub Repository -->
<!-- Created by Matthew Tungett, Last Edit: 4/14/23 -->

<robot name="Kinova_Arm">

  <!-- Run the macros -->

  <link name="base_link">
    <inertial>
      <origin rpy="0 0 0" xyz="-0.000648 -0.000166 0.084487"/>
      <mass value="1.697"/>
      <inertia ixx="0.004622" ixy="9E-06" ixz="6E-05" iyy="0.004495" iyz="9E-
06" izz="0.002079"/>
    </inertial>
    <visual>
      <origin rpy="0 0 0" xyz="0 0 0"/>
      <geometry>
        <mesh filename="base_link.STL"/>
      </geometry>
      <material name="">
        <color rgba="0.75294 0.75294 0.75294 1"/>
      </material>
    </visual>
    <collision>
      <origin rpy="0 0 0" xyz="0 0 0"/>
      <geometry>
        <mesh filename="base_link.STL"/>
      </geometry>
    </collision>
  </link>

  <link name="shoulder_link">
    <inertial>
      <origin rpy="0 0 0" xyz="-2.3E-05 -0.010364 -0.07336"/>
      <mass value="1.3773"/>
      <inertia ixx="0.00457" ixy="1E-06" ixz="2E-06" iyy="0.004831"
iyz="0.000448" izz="0.001409"/>
    </inertial>
    <visual>
      <origin rpy="0 0 0" xyz="0 0 0"/>
      <geometry>
        <mesh filename="shoulder_link.STL"/>
      </geometry>
      <material name="">
        <color rgba="0.75294 0.75294 0.75294 1"/>
      </material>
    </visual>
    <collision>
```

```

    <origin rpy="0 0 0" xyz="0 0 0"/>
    <geometry>
      <mesh filename="shoulder_link.STL"/>
    </geometry>
  </collision>
</link>

<joint name="joint_1" type="continuous">
  <origin rpy="3.1416 2.7629E-18 -4.9305E-36" xyz="0 0 0.15643"/>
  <parent link="base_link"/>
  <child link="shoulder_link"/>
  <axis xyz="0 0 1"/>
  <limit effort="39" velocity="1.3963"/>
</joint>

<link name="half_arm_1_link">
  <inertial>
    <origin rpy="0 0 0" xyz="-4.4E-05 -0.09958 -0.013278"/>
    <mass value="1.1636"/>
    <inertia ixx="0.011088" ixy="5E-06" ixz="0" iyy="0.001072" iyz="-
0.000691" izz="0.011255"/>
  </inertial>
  <visual>
    <origin rpy="0 0 0" xyz="0 0 0"/>
    <geometry>
      <mesh filename="half_arm_1_link.STL"/>
    </geometry>
    <material name="">
      <color rgba="0.75294 0.75294 0.75294 1"/>
    </material>
  </visual>
  <collision>
    <origin rpy="0 0 0" xyz="0 0 0"/>
    <geometry>
      <mesh filename="half_arm_1_link.STL"/>
    </geometry>
  </collision>
</link>

<joint name="joint_2" type="revolute">
  <origin rpy="1.5708 2.1343E-17 -1.1102E-16" xyz="0 0.005375 -0.12838"/>
  <parent link="shoulder_link"/>
  <child link="half_arm_1_link"/>
  <axis xyz="0 0 1"/>
  <limit effort="39" lower="-2.41" upper="2.41" velocity="1.3963"/>
</joint>

<link name="half_arm_2_link">
  <inertial>
    <origin rpy="0 0 0" xyz="-4.4E-05 -0.006641 -0.117892"/>
    <mass value="1.1636"/>
    <inertia ixx="0.010932" ixy="0" ixz="-7E-06" iyy="0.011127"
iyz="0.000606" izz="0.001043"/>
  </inertial>

```

```

<visual>
  <origin rpy="0 0 0" xyz="0 0 0"/>
  <geometry>
    <mesh filename="half_arm_2_link.STL"/>
  </geometry>
  <material name="">
    <color rgba="0.75294 0.75294 0.75294 1"/>
  </material>
</visual>
<collision>
  <origin rpy="0 0 0" xyz="0 0 0"/>
  <geometry>
    <mesh filename="half_arm_2_link.STL"/>
  </geometry>
</collision>
</link>

<joint name="joint_3" type="continuous">
  <origin rpy="-1.5708 1.2326E-32 -2.9122E-16" xyz="0 -0.21038 -0.006375"/>
  <parent link="half_arm_1_link"/>
  <child link="half_arm_2_link"/>
  <axis xyz="0 0 1"/>
  <limit effort="39" velocity="1.3963"/>
</joint>

<link name="forearm_link">
  <inertial>
    <origin rpy="0 0 0" xyz="-1.8E-05 -0.075478 -0.015006"/>
    <mass value="0.9302"/>
    <inertia ixx="0.008147" ixy="-1E-06" ixz="0" iyy="0.000631" iyz="-
0.0005" izz="0.008316"/>
  </inertial>
  <visual>
    <origin rpy="0 0 0" xyz="0 0 0"/>
    <geometry>
      <mesh filename="forearm_link.STL"/>
    </geometry>
    <material name="">
      <color rgba="0.75294 0.75294 0.75294 1"/>
    </material>
  </visual>
  <collision>
    <origin rpy="0 0 0" xyz="0 0 0"/>
    <geometry>
      <mesh filename="forearm_link.STL"/>
    </geometry>
  </collision>
</link>

<joint name="joint_4" type="revolute">
  <origin rpy="1.5708 -6.6954E-17 -1.6653E-16" xyz="0 0.006375 -0.21038"/>
  <parent link="half_arm_2_link"/>
  <child link="forearm_link"/>
  <axis xyz="0 0 1"/>

```

```

    <limit effort="39" lower="-2.66" upper="2.66" velocity="1.3963"/>
  </joint>

  <link name="spherical_wrist_1_link">
    <inertial>
      <origin rpy="0 0 0" xyz="1E-06 -0.009432 -0.063883"/>
      <mass value="0.6781"/>
      <inertia ixx="0.001596" ixy="0" ixz="0" iyy="0.001607" iyz="0.000256"
izz="0.000399"/>
    </inertial>
    <visual>
      <origin rpy="0 0 0" xyz="0 0 0"/>
      <geometry>
        <mesh filename="spherical_wrist_1_link.STL"/>
      </geometry>
      <material name="">
        <color rgba="0.75294 0.75294 0.75294 1"/>
      </material>
    </visual>
    <collision>
      <origin rpy="0 0 0" xyz="0 0 0"/>
      <geometry>
        <mesh filename="spherical_wrist_1_link.STL"/>
      </geometry>
    </collision>
  </link>

  <joint name="joint_5" type="continuous">
    <origin rpy="-1.5708 2.2204E-16 -6.373E-17" xyz="0 -0.20843 -0.006375"/>
    <parent link="forearm_link"/>
    <child link="spherical_wrist_1_link"/>
    <axis xyz="0 0 1"/>
    <limit effort="9" velocity="1.2218"/>
  </joint>

  <link name="spherical_wrist_2_link">
    <inertial>
      <origin rpy="0 0 0" xyz="1E-06 -0.045483 -0.00965"/>
      <mass value="0.6781"/>
      <inertia ixx="0.001641" ixy="0" ixz="0" iyy="0.00041" iyz="-0.000278"
izz="0.001641"/>
    </inertial>
    <visual>
      <origin rpy="0 0 0" xyz="0 0 0"/>
      <geometry>
        <mesh filename="spherical_wrist_2_link.STL"/>
      </geometry>
      <material name="">
        <color rgba="0.75294 0.75294 0.75294 1"/>
      </material>
    </visual>
    <collision>
      <origin rpy="0 0 0" xyz="0 0 0"/>
      <geometry>

```

```

        <mesh filename="spherical_wrist_2_link.STL"/>
    </geometry>
</collision>
</link>

<joint name="joint_6" type="revolute">
    <origin rpy="1.5708 9.2076E-28 -8.2157E-15" xyz="0 0.00017505 -0.10593"/>
    <parent link="spherical_wrist_1_link"/>
    <child link="spherical_wrist_2_link"/>
    <axis xyz="0 0 1"/>
    <limit effort="9" lower="-2.23" upper="2.23" velocity="1.2218"/>
</joint>

<link name="bracelet_link">
    <inertial>
        <origin rpy="0 0 0" xyz="0.000281 0.011402 -0.029798"/>
        <mass value="0.5"/>
        <inertia ixx="0.000587" ixy="3E-06" ixz="3E-06" iyy="0.000369" iyz="-
0.000118" izz="0.000609"/>
    </inertial>
    <visual>
        <origin rpy="0 0 0" xyz="0 0 0"/>
        <geometry>
            <mesh filename="bracelet_with_vision_link.STL"/>
        </geometry>
        <material name="">
            <color rgba="0.75294 0.75294 0.75294 1"/>
        </material>
    </visual>
    <collision>
        <origin rpy="0 0 0" xyz="0 0 0"/>
        <geometry>
            <mesh filename="bracelet_with_vision_link.STL"/>
        </geometry>
    </collision>
</link>

<joint name="joint_7" type="continuous">
    <origin rpy="-1.5708 -5.5511E-17 9.6396E-17" xyz="0 -0.10593 -
0.00017505"/>
    <parent link="spherical_wrist_2_link"/>
    <child link="bracelet_link"/>
    <axis xyz="0 0 1"/>
    <limit effort="9" velocity="1.2218"/>
</joint>

<link name="end_effector_link"/>
<joint name="end_effector" type="fixed">
    <origin rpy="3.14159265358979 1.09937075168372E-32 0" xyz="0 0 -
0.0615250000000001"/>
    <parent link="bracelet_link"/>
    <child link="end_effector_link"/>
    <axis xyz="0 0 0"/>
</joint>

```

```

<link name="robotiq_2f_140_link">
  <inertial>
    <origin rpy="0 0 0" xyz="0.0 0.0 0.073"/>
    <mass value="1.025"/>
    <inertia ixx="0.0074" ixy="0.0" ixz="0.0" iyy="0.00932" iyz="0.0"
izz="0.00226"/>
  </inertial>
  <visual>
    <origin rpy="0 0 0" xyz="0 0 0"/>
    <geometry>
      <mesh filename="robotiq_2f_140.STL" scale="0.001 0.001 0.001"/>
    </geometry>
    <material name="">
      <color rgba="0.1 0.1 0.1 1"/>
    </material>
  </visual>
  <collision>
    <origin rpy="0 0 0" xyz="0 0 0"/>
    <geometry>
      <mesh filename="robotiq_2f_140.STL" scale="0.001 0.001 0.001"/>
    </geometry>
  </collision>
</link>

<joint name="gripper_base_joint" type="fixed">
  <parent link="end_effector_link"/>
  <child link="robotiq_2f_140_link"/>
  <origin rpy="0.0 0.0 0.0" xyz="0.0 0.0 0.0"/>
</joint>

</robot>

```

APPENDIX B. KINOVA ARM WITH TEST OBJECT URDF FILE

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

<!-- This file was modified from the Kinova Robotics GitHub Repository -->
<!-- Created by Matthew Tungett, Last Edit: 4/14/23 -->

<robot name="Kinova_Arm_With_Test_Object">

  <link name="base_link">
    <inertial>
      <origin rpy="0 0 0" xyz="-0.000648 -0.000166 0.084487"/>
      <mass value="1.697"/>
      <inertia ixx="0.004622" ixy="9E-06" ixz="6E-05" iyy="0.004495" iyz="9E-
16" izz="0.002079"/>
    </inertial>
    <visual>
      <origin rpy="0 0 0" xyz="0 0 0"/>
      <geometry>
        <mesh filename="base_link.STL"/>
      </geometry>
      <material name="">
        <color rgba="0.75294 0.75294 0.75294 1"/>
      </material>
    </visual>
    <collision>
      <origin rpy="0 0 0" xyz="0 0 0"/>
      <geometry>
        <mesh filename="base_link.STL"/>
      </geometry>
    </collision>
  </link>

  <link name="shoulder_link">
    <inertial>
      <origin rpy="0 0 0" xyz="-2.3E-05 -0.010364 -0.07336"/>
      <mass value="1.3773"/>
      <inertia ixx="0.00457" ixy="1E-06" ixz="2E-06" iyy="0.004831"
iyz="0.000448" izz="0.001409"/>
    </inertial>
    <visual>
      <origin rpy="0 0 0" xyz="0 0 0"/>
      <geometry>
        <mesh filename="shoulder_link.STL"/>
      </geometry>
      <material name="">
        <color rgba="0.75294 0.75294 0.75294 1"/>
      </material>
    </visual>
    <collision>
      <origin rpy="0 0 0" xyz="0 0 0"/>
      <geometry>
```

```

        <mesh filename="shoulder_link.STL"/>
    </geometry>
</collision>
</link>

<joint name="joint_1" type="continuous">
    <origin rpy="3.1416 2.7629E-18 -4.9305E-36" xyz="0 0 0.15643"/>
    <parent link="base_link"/>
    <child link="shoulder_link"/>
    <axis xyz="0 0 1"/>
    <limit effort="39" velocity="1.3963"/>
</joint>

<link name="half_arm_1_link">
    <inertial>
        <origin rpy="0 0 0" xyz="-4.4E-05 -0.09958 -0.013278"/>
        <mass value="1.1636"/>
        <inertia ixx="0.011088" ixy="5E-06" ixz="0" iyy="0.001072" iyz="-
0.000691" izz="0.011255"/>
    </inertial>
    <visual>
        <origin rpy="0 0 0" xyz="0 0 0"/>
        <geometry>
            <mesh filename="half_arm_1_link.STL"/>
        </geometry>
        <material name="">
            <color rgba="0.75294 0.75294 0.75294 1"/>
        </material>
    </visual>
    <collision>
        <origin rpy="0 0 0" xyz="0 0 0"/>
        <geometry>
            <mesh filename="half_arm_1_link.STL"/>
        </geometry>
    </collision>
</link>

<joint name="joint_2" type="revolute">
    <origin rpy="1.5708 2.1343E-17 -1.1102E-16" xyz="0 0.005375 -0.12838"/>
    <parent link="shoulder_link"/>
    <child link="half_arm_1_link"/>
    <axis xyz="0 0 1"/>
    <limit effort="39" lower="-2.41" upper="2.41" velocity="1.3963"/>
</joint>

<link name="half_arm_2_link">
    <inertial>
        <origin rpy="0 0 0" xyz="-4.4E-05 -0.006641 -0.117892"/>
        <mass value="1.1636"/>
        <inertia ixx="0.010932" ixy="0" ixz="-7E-06" iyy="0.011127"
iyz="0.000606" izz="0.001043"/>
    </inertial>
    <visual>
        <origin rpy="0 0 0" xyz="0 0 0"/>

```

```

    <geometry>
      <mesh filename="half_arm_2_link.STL"/>
    </geometry>
    <material name="">
      <color rgba="0.75294 0.75294 0.75294 1"/>
    </material>
  </visual>
  <collision>
    <origin rpy="0 0 0" xyz="0 0 0"/>
    <geometry>
      <mesh filename="half_arm_2_link.STL"/>
    </geometry>
  </collision>
</link>

<joint name="joint_3" type="continuous">
  <origin rpy="-1.5708 1.2326E-32 -2.9122E-16" xyz="0 -0.21038 -0.006375"/>
  <parent link="half_arm_1_link"/>
  <child link="half_arm_2_link"/>
  <axis xyz="0 0 1"/>
  <limit effort="39" velocity="1.3963"/>
</joint>

<link name="forearm_link">
  <inertial>
    <origin rpy="0 0 0" xyz="-1.8E-05 -0.075478 -0.015006"/>
    <mass value="0.9302"/>
    <inertia ixx="0.008147" ixy="-1E-06" ixz="0" iyy="0.000631" iyz="-
0.0005" izz="0.008316"/>
  </inertial>
  <visual>
    <origin rpy="0 0 0" xyz="0 0 0"/>
    <geometry>
      <mesh filename="forearm_link.STL"/>
    </geometry>
    <material name="">
      <color rgba="0.75294 0.75294 0.75294 1"/>
    </material>
  </visual>
  <collision>
    <origin rpy="0 0 0" xyz="0 0 0"/>
    <geometry>
      <mesh filename="forearm_link.STL"/>
    </geometry>
  </collision>
</link>

<joint name="joint_4" type="revolute">
  <origin rpy="1.5708 -6.6954E-17 -1.6653E-16" xyz="0 0.006375 -0.21038"/>
  <parent link="half_arm_2_link"/>
  <child link="forearm_link"/>
  <axis xyz="0 0 1"/>
  <limit effort="39" lower="-2.66" upper="2.66" velocity="1.3963"/>
</joint>

```

```

<link name="spherical_wrist_1_link">
  <inertial>
    <origin rpy="0 0 0" xyz="1E-06 -0.009432 -0.063883"/>
    <mass value="0.6781"/>
    <inertia ixx="0.001596" ixy="0" ixz="0" iyy="0.001607" iyz="0.000256"
izz="0.000399"/>
  </inertial>
  <visual>
    <origin rpy="0 0 0" xyz="0 0 0"/>
    <geometry>
      <mesh filename="spherical_wrist_1_link.STL"/>
    </geometry>
    <material name="">
      <color rgba="0.75294 0.75294 0.75294 1"/>
    </material>
  </visual>
  <collision>
    <origin rpy="0 0 0" xyz="0 0 0"/>
    <geometry>
      <mesh filename="spherical_wrist_1_link.STL"/>
    </geometry>
  </collision>
</link>

<joint name="joint_5" type="continuous">
  <origin rpy="-1.5708 2.2204E-16 -6.373E-17" xyz="0 -0.20843 -0.006375"/>
  <parent link="forearm_link"/>
  <child link="spherical_wrist_1_link"/>
  <axis xyz="0 0 1"/>
  <limit effort="9" velocity="1.2218"/>
</joint>

<link name="spherical_wrist_2_link">
  <inertial>
    <origin rpy="0 0 0" xyz="1E-06 -0.045483 -0.00965"/>
    <mass value="0.6781"/>
    <inertia ixx="0.001641" ixy="0" ixz="0" iyy="0.00041" iyz="-0.000278"
izz="0.001641"/>
  </inertial>
  <visual>
    <origin rpy="0 0 0" xyz="0 0 0"/>
    <geometry>
      <mesh filename="spherical_wrist_2_link.STL"/>
    </geometry>
    <material name="">
      <color rgba="0.75294 0.75294 0.75294 1"/>
    </material>
  </visual>
  <collision>
    <origin rpy="0 0 0" xyz="0 0 0"/>
    <geometry>
      <mesh filename="spherical_wrist_2_link.STL"/>
    </geometry>
  </collision>
</link>

```

```

    </collision>
</link>

<joint name="joint_6" type="revolute">
  <origin rpy="1.5708 9.2076E-28 -8.2157E-15" xyz="0 0.00017505 -0.10593"/>
  <parent link="spherical_wrist_1_link"/>
  <child link="spherical_wrist_2_link"/>
  <axis xyz="0 0 1"/>
  <limit effort="9" lower="-2.23" upper="2.23" velocity="1.2218"/>
</joint>

<link name="bracelet_link">
  <inertial>
    <origin rpy="0 0 0" xyz="0.000281 0.011402 -0.029798"/>
    <mass value="0.5"/>
    <inertia ixx="0.000587" ixy="3E-06" ixz="3E-06" iyy="0.000369" iyz="-
0.000118" izz="0.000609"/>
  </inertial>
  <visual>
    <origin rpy="0 0 0" xyz="0 0 0"/>
    <geometry>
      <mesh filename="bracelet_with_vision_link.STL"/>
    </geometry>
    <material name="">
      <color rgba="0.75294 0.75294 0.75294 1"/>
    </material>
  </visual>
  <collision>
    <origin rpy="0 0 0" xyz="0 0 0"/>
    <geometry>
      <mesh filename="bracelet_with_vision_link.STL"/>
    </geometry>
  </collision>
</link>

<joint name="joint_7" type="continuous">
  <origin rpy="-1.5708 -5.5511E-17 9.6396E-17" xyz="0 -0.10593 -
0.00017505"/>
  <parent link="spherical_wrist_2_link"/>
  <child link="bracelet_link"/>
  <axis xyz="0 0 1"/>
  <limit effort="9" velocity="1.2218"/>
</joint>

<link name="end_effector_link"/>
<joint name="end_effector" type="fixed">
  <origin rpy="3.14159265358979 1.09937075168372E-32 0" xyz="0 0 -
0.0615250000000001"/>
  <parent link="bracelet_link"/>
  <child link="end_effector_link"/>
  <axis xyz="0 0 0"/>
</joint>

<link name="robotiq_2f_140_link">

```

```

    <inertial>
      <origin rpy="0 0 0" xyz="0.0 0.0 0.073"/>
      <mass value="1.025"/>
      <inertia ixx="0.0074" ixy="0.0" ixz="0.0" iyy="0.00932" iyz="0.0"
izz="0.00226"/>
    </inertial>
    <visual>
      <origin rpy="0 0 0" xyz="0 0 0"/>
      <geometry>
        <mesh filename="robotiq_2f_140_closed.STL" scale="0.0254 0.0254
0.0254"/>
      </geometry>
      <material name="">
        <color rgba="0.1 0.1 0.1 1"/>
      </material>
    </visual>
    <collision>
      <origin rpy="0 0 0" xyz="0 0 0"/>
      <geometry>
        <mesh filename="robotiq_2f_140_closed.STL" scale="0.0254 0.0254
0.0254"/>
      </geometry>
    </collision>
  </link>

  <joint name="gripper_base_joint" type="fixed">
    <parent link="end_effector_link"/>
    <child link="robotiq_2f_140_link"/>
    <origin rpy="0.0 0.0 0.0" xyz="0.0 0.0 0.0"/>
  </joint>

  <link name="object_link">
    <inertial>
      <origin rpy="0 0 0" xyz="0.0 0.0 0.0"/>
      <mass value="0.340"/>
      <inertia ixx="0.000375" ixy="0.0" ixz="0.0" iyy="0.000375" iyz="0.0"
izz="0.000022"/>
    </inertial>
    <visual>
      <origin rpy="0 0 0" xyz="0 0 0"/>
      <geometry>
        <mesh filename="object.STL" scale="0.0254 0.0254 0.0254"/>
      </geometry>
      <material name="">
        <color rgba="0.75294 0.75294 0.75294 1"/>
      </material>
    </visual>
    <collision>
      <origin rpy="0 0 0" xyz="0 0 0"/>
      <geometry>
        <mesh filename="object.STL" scale="0.0254 0.0254 0.0254"/>
      </geometry>
    </collision>
  </link>

```

```
<joint name="object_joint" type="fixed">
  <parent link="end_effector_link"/>
  <child link="object_link"/>
  <origin rpy="1.57 0.0 0.0" xyz="0.0 0.0 0.21"/>
</joint>

</robot>
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX C. KINOVA ARM URDF AND MBD MODEL SIMULATION SCRIPT

```
%% Kinova Arm URDF and MBD Model Simulation
% Created by Matthew Tungett, Last edit: 4/15/23

%% Script Cleanup
close all
clear
clc

%% Simulation setup

% Load Robot
robot = importrobot('Kinova Arm/Kinova_7_DoF_arm.urdf', ...
    DataFormat='column');

% Define Gravity
robot.Gravity = [0 0 -9.81]; % [m/s^2]

% Timespan
endtime = 10;

% Simulation Waypoint Selection

% Specify Desired Test
flag = input("Specify Desired Test --> ");

% Test selection
if flag == 1
    % Test 1
    w1 = [0;0;0;0;0;-90]*(pi/180); % start
    w2 = [0;90;0;0;0;-90]*(pi/180); % end
elseif flag == 2
    % Test 2
    w1 = [0;90;90;-45;0;0;-90]*(pi/180); % start
    w2 = [190;90;-90;-45;0;0;-90]*(pi/180); % end
elseif flag == 3
    % Test 3
    w1 = [90;0;0;0;45;0;-90]*(pi/180); % start
    w2 = [0;-20;180;-146;0;-50;90]*(pi/180); % end
else
    % Error message
    error("Incorrect input. Please input a 0, 1, or 2")
end

%% MBD Model Simulation

% Start simulation
MBD_out = sim('Kinova_7_DoF_arm.slx','StartTime','0',...
    'StopTime',num2str(endtime));
```

```

% Un-pack data
t_hist = MBD_out.t_hist;
q_hist = MBD_out.q_hist;
qd_hist = MBD_out.qd_hist;
qdd_hist = MBD_out.qdd_hist;
MBD_torque_hist = MBD_out.torque_hist;

%% URDF Model Simulation

% URDF Torque Calculation
for i = 1:length(t_hist)
    URDF_torque_hist(i,:) = inverseDynamics(robot,...
        q_hist(i,:)',qd_hist(i,:)',qdd_hist(i,:)');
end

%% Control Trajectory Plot

% Plot Control Trajectory
figure(1)

% Initial Kinova Arm Configuration
subplot(1,2,1)
show(robot,q_hist(1,:)', 'PreservePlot',true,'Frames','off');

% Motion path and Final Kinova arm configuration
subplot(1,2,2)
show(robot,q_hist(1,:)', 'PreservePlot',false,'Frames','off');
hold on
for i = 1:length(t_hist)
    % Current Time
    t_now = t_hist(i);
    % Define end-effector joint marker
    pose_now = getTransform(robot,q_hist(i,:)',...
        'robotiq_2f_140_link');
    show(robot,q_hist(i,:)', 'PreservePlot',false,...
        'Frames','off');
    taskSpaceMarker = plot3(pose_now(1,4),pose_now(2,4),...
        pose_now(3,4),'b.','MarkerSize',20);
    drawnow
end

%% Simulation Performance Plot

% Calculate Mean Absolute Error Between the URDF and MBD Simulations
for i = 1:7
    mean_abs_error(i,:) = sum(abs(MBD_torque_hist(:,i)-...
        URDF_torque_hist(:,i)))/length(t_hist);
end

% Plot Simulation Torque Histories
figure(2)
grid on, hold on
xlabel('Time [sec]')

```

```

ylabel('Torque [N-m]')
p = plot(t_hist,MBD_torque_hist,'LineWidth',1);
p1 = plot(t_hist,URDF_torque_hist,'o','MarkerSize',5,...
          'LineWidth',1);

MBD_legend = legend(p,'\tau_1(t)','\tau_2(t)','\tau_3(t)',...
                  '\tau_4(t)','\tau_5(t)','\tau_6(t)',...
                  '\tau_7(t)','Location','NorthEast');

MBD_legend.Title.String = 'MBD';
MBD_legend.FontWeight = 'normal';

z = axes('position',get(gca,'position'),'visible','off');
URDF_legend = legend(z,p1,'\tau_1(t)','\tau_2(t)','\tau_3(t)',...
                    '\tau_4(t)','\tau_5(t)','\tau_6(t)',...
                    '\tau_7(t)','Location','NorthWest');

URDF_legend.Title.String = 'URDF';
URDF_legend.FontWeight = 'normal';

```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX D. KINOVA ARM COMMAND GENERATION AND SENSOR RECORDING SCRIPT

```
%% Command and Record gen3 Kinova Arm
% Created by Matthew Tungett, Last edit: 4/2/23

%% Script Cleanup
close all
clear
clc

%% Connect to Kinova arm

% Create API instance to Connect to Robot
Simulink.importExternalCTypes(which('kortex_wrapper_data.h'));
gen3Kinova = kortex();
gen3Kinova.ip_address = '192.168.1.10';
gen3Kinova.user = 'admin';
gen3Kinova.password = 'admin';

% Connect to Robot
isOk = gen3Kinova.CreateRobotApisWrapper();
if isOk
    disp('You are connected to the robot!');
else
    error('Failed to establish a valid connection!');
end

%% Define Kinova arm Maneuver

% Kinova arm start and end position
start_pose = [0 0 0 0 0 0]; % [deg]
end_pose = [0 0 0 0 0 0]; % [deg]

% Constraints
constraintType = int32(0);
speed = 0;
duration = 0;
t_hist = 0;

%% Send commands and record sensors

% Record sensor readings
for i = 1:1000
    tic
    [isOk,baseFb, actuatorFb, interconnectFb] = ...
        gen3Kinova.SendRefreshFeedback();
    q_hist(i,:) = actuatorFb.position;
    qd_hist(i,:) = actuatorFb.velocity;
    torq_hist(i,:) = actuatorFb.torque;
    wrench_F(i,:) = baseFb.tool_external_wrench_force;
end
```

```

wrench_T(i,:) = baseFb.tool_external_wrench_torque;
t_hist(i+1) = toc+t_hist(i);

% Send command
if i == 50
    isOk = gen3Kinova.SendJointAngles(start_pose, ...
                                      constraintType, speed, duration);
elseif i == 550
    isOk = gen3Kinova.SendJointAngles(end_pose, ...
                                      constraintType, speed, duration);
end

end

%% Store DATA

% User Input
decision = input("Do you want to store this test " + ...
                "(1 = yes / 0 = no)? --> ");

if decision == 1
    % Define Data Label
    data_label = input("Specify Data Set Label --> ");

    % Define File Names
    time_label = ['Kinova Arm Tests\Test ', num2str(data_label), ...
                  ' - Time.xls'];
    q_label = ['Kinova Arm Tests\Test ', num2str(data_label), ...
              ' - q.xls'];
    qd_label = ['Kinova Arm Tests\Test ', num2str(data_label), ...
               ' - qd.xls'];
    torque_label = ['Kinova Arm Tests\Test ', num2str(data_label), ...
                   ' - torque.xls'];
    wrench_F_label = ['Kinova Arm Tests\Test ', num2str(data_label), ...
                     ' - wrench force.xls'];
    wrench_T_label = ['Kinova Arm Tests\Test ', num2str(data_label), ...
                     ' - wrench torque.xls'];

    % Record Data
    xlswrite(time_label,t_hist,1);
    xlswrite(q_label,q_hist,1);
    xlswrite(qd_label,qd_hist,1);
    xlswrite(torque_label,torq_hist,1);
    xlswrite(wrench_F_label,wrench_F,1);
    xlswrite(wrench_T_label,wrench_T,1);
    disp('DATA has been stored successfully!')
elseif decision == 0
    disp('DATA will not be stored')
else
    error('Incorrect Input -- run this section again')
end

```

APPENDIX E. REAL KINOVA ARM TESTING VS. URDF MODEL SIMULATION SCRIPT

```
%% Real Kinova Arm vs. URDF Model Testing
% Created by Matthew Tungett, Last edit: 4/16/23

%% Script Cleanup
close all
clear
clc

%% Load Data

% Specify Data to be displayed
data_label = input("Specify Data Set --> ");

% Define File Names
time_label = ['Kinova Arm Tests\Test ', num2str(data_label), ...
              ' - Time.xls'];
q_label = ['Kinova Arm Tests\Test ', num2str(data_label), ...
           ' - q.xls'];
qd_label = ['Kinova Arm Tests\Test ', num2str(data_label), ...
            ' - qd.xls'];
torque_label = ['Kinova Arm Tests\Test ', num2str(data_label), ...
                ' - torque.xls'];

% Load Data
t_hist = xlsread(time_label);
q_hist = xlsread(q_label);
qd_hist = xlsread(qd_label);
measured_torque_hist = xlsread(torque_label);

% Data Index For Presented Data
% if data_label == 1.3
%     start_time = 509;
%     end_time = 654;
% elseif data_label == 2.2
%     start_time = 509;
%     end_time = 814;
% elseif data_label == 3.3
%     start_time = 507;
%     end_time = 632;
% elseif data_label == 4.2
%     start_time = 508;
%     end_time = 641;
% else
%     start_time = 1;
%     end_time = length(t_hist)-1;
% end

% Default Data Index
```

```

start_time = 1;
end_time = length(t_hist)-1;

% Format Data
time = t_hist(start_time:end_time)-t_hist(start_time);
qd = [qd_hist(start_time:end_time,:)]*pi/180;
qdd = diff(qd);
measured_torque = measured_torque_hist(start_time:end_time,:);

% Adjust Near Zero Joint Positions
for k = 1:7
    q(:,k) = [q_hist(start_time:end_time,k)]*pi/180;
    for i = 1:end_time-start_time+1
        if q(i,k) >= 6.28
            q(i,k) = 0;
        end
    end
end

%% URDF Calculated Torques

% Robot models for presented data
% if round(data_label)<=2 % Isolated Kinova Arm
%     robot = importrobot('Kinova Arm\Kinova_7_DoF_arm.urdf',...
%                         DataFormat='row');
% elseif round(data_label)<=4 % Kinova Arm with Test Object
%     robot = importrobot('Kinova Arm\Kinova_7_DoF_arm_object.urdf',...
%                         DataFormat='row');
% else % Isolated Kinova Arm
%     robot = importrobot('Kinova Arm\Kinova_7_DoF_arm.urdf',...
%                         DataFormat='row');
% end

% Default Robot Model
robot = importrobot('Kinova Arm\Kinova_7_DoF_arm.urdf',...
                    DataFormat='row');

% Define Gravitational Acceleration
robot.Gravity = [0 0 -9.81];

% Calculated Torque
for i = 1:length(qdd)
    calculated_torque(i,:) = inverseDynamics(robot,q(i,:),...
                                              qd(i,:),qdd(i,:));
end

% Estimated Calibration
q_offset = -measured_torque(1,:)-calculated_torque(1,:);

%% Display Kinova Arm Control Maneuver

% Plot Control Trajectory
figure(1)

```

```

% Initial Kinova Arm Configuration
subplot(1,2,1)
show(robot,q(1,:), 'PreservePlot', false, 'Frames', 'off');

% Motion path and Final Kinova arm configuration
subplot(1,2,2)
show(robot,q(1,:), 'PreservePlot', false, 'Frames', 'off');
hold on
for i = 1:length(time)
    % Current Time
    t_now = time(i);
    % Define end-effector joint marker
    pose_now = getTransform(robot,q(i,:),...
                             'robotiq_2f_140_link');
    show(robot,q(i,:), 'PreservePlot', false, 'Frames', 'off');
    taskSpaceMarker = plot3(pose_now(1,4),pose_now(2,4),...
                             pose_now(3,4), 'b.', 'MarkerSize', 20);
    drawnow
end

%% Measured vs. Calculated Torques

% Calculate Sample Rate
for i = 1:length(qdd)
    dt(i,:) = time(i+1)-time(i);
end
avg_sample_rate = sum(dt)/length(qdd);

% Calculate MAE Between the Measured and Simulated Kinova Arm
for i = 1:7
    mean_absolute_error(i,:) = sum(abs...
        (-measured_torque(1:end-1,i)-...
        calculated_torque(:,i)-...
        q_offset(:,i)))/length(qdd);
end

% Plot Measured vs. Calculated Torques
figure(2)
grid on, hold on
xlabel('Time [sec]')
ylabel('Torque [N-m]')
p = plot(time,measured_torque+q_offset);
p1 = plot(time(1:end-1),-calculated_torque, 'LineWidth', 2);
measured_legend = legend(p, '\tau_1(t)', '\tau_2(t)', '\tau_3(t)',...
    '\tau_4(t)', '\tau_5(t)', '\tau_6(t)',...
    '\tau_7(t)', 'Location', 'NorthEast');
measured_legend.Title.String = 'Measured';
measured_legend.FontWeight = 'normal';
z = axes('position',get(gca,'position'),'visible','off');
calculated_legend = legend(z,p1, '\tau_1(t)', '\tau_2(t)',...
    '\tau_3(t)', '\tau_4(t)',...
    '\tau_5(t)', '\tau_6(t)',...
    '\tau_7(t)', 'Location',...

```

```
                                'NorthWest');  
calculated_legend.Title.String = 'Calculated';  
calculated_legend.FontWeight = 'normal';
```

APPENDIX F. FSS BASE DYNAMICAL MODEL SCRIPT

```
function dydt = FSS_Base_Dynamical_Model(t,X,t_opt,U, constants)

% Six state variables
x = X(1);
y = X(2);
vx = X(3);
vy = X(4);
theta = X(5);
omega = X(6);

% Three control variables
Fx = U(1,1:end);
Fy = U(2,1:end);
tau = U(3,1:end);

% Interpolate control variables
Fx_i = interp1(t_opt,Fx,t,'linear',Fx(end));
Fy_i = interp1(t_opt,Fy,t,'linear',Fy(end));
tau_i = interp1(t_opt,tau,t,'linear',tau(end));

% Equations of motion
x_dot = vx.*cos(theta)-vy.*sin(theta);
y_dot = vx.*sin(theta)+vy.*cos(theta);
vx_dot = omega.*vy+Fx_i/constants.m;
vy_dot = -omega.*vx+Fy_i/constants.m;
theta_dot = omega;
omega_dot = tau_i/constants.Izz;

% Output
dydt = [ x_dot;
        y_dot;
        vx_dot;
        vy_dot;
        theta_dot;
        omega_dot];

end
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX G. FSS BASE MBD MODEL VS. DYNAMICAL MODEL SIMULATION SCRIPT

```
% FSS Base MBD and Dynamical Model Testing
% Created by Matthew Tungett, Last edit: 4/23/23

%% Script cleanup
close all
clear
clc

%% Simulation Setup

% Specify test to be displayed
flag = input("Specify test --> ");

% Control Selection

% Fx = x-axis thrust [N]
% Fy = y-axis thrust [N]
% Tz = RW torque [N-m]
% crop = specialized plot adjustment for giving test

if flag == 1          % Test 1
    Fx = -0.2;
    Fy = 0.4;
    Tz = 0.0;
    crop = [-0.30 0.05 -0.025 0.325];
elseif flag == 2      % Test 2
    Fx = 0.0;
    Fy = 0.0;
    Tz = 0.2;
    crop = [-0.25 0.25 -0.25 0.25];
elseif flag == 3      % Test 3
    Fx = 0.4;
    Fy = 0.4;
    Tz = -0.2;
    crop = [-0.025 0.275 -0.10 0.20 ];
else                  % Zero control
    Fx = 0.0;
    Fy = 0.0;
    Tz = 0.0;
    crop = [-1 1 -1 1];
end

% Simulation Time
endtime = 10;

%% MBD Simulation
```

```

% Run MBD Simulation
simout = sim('FSS_Base.slx','StartTime','0','StopTime',...
            num2str(endtime));

% Unpack MBD Data
t_MBD = simout.t_hist;
x_MBD = simout.x;
y_MBD = simout.y;
theta_MBD = simout.theta;
vx_MBD = simout.vx;
vy_MBD = simout.vy;
omega_MBD = simout.omega;
ax_MBD = simout.ax;
ay_MBD = simout.ay;
omega_dot_MBD = simout.omega_dot;

%% Propogated Simulation

% Constants
constants.m = 62.0;      % FSS base mass [kg]
constants.Izz = 1.203;   % FSS base Izz [kg*m^2]

% Inital States
x0 = 0;
y0 = 0;
vx0 = 0;
vy0 = 0;
theta0 = 0;
omega0 = 0;

% Timespan
dt = 0.20;
t = [0:dt:endtime];
t_span = [0:dt:endtime];

% Controls
Fx = zeros(1,length(t_span))+Fx;
Fy = zeros(1,length(t_span))+Fy;
tau = zeros(1,length(t_span))+Tz;

% Set initial conditions
X0 = [x0;y0;vx0;vy0;theta0;omega0];
U = [Fx;Fy;tau];

% Run propagation simulation
[t_prop,X_prop] = ode45(@FSS_Base_Dynamical_Model,t_span, X0,[],t,U,...
                        constants);

% Un-pack propogation data
x_prop = X_prop(:,1);
y_prop = X_prop(:,2);
vx_prop = X_prop(:,3);
vy_prop = X_prop(:,4);
theta_prop = X_prop(:,5);

```

```

omega_prop = X_prop(:,6);

%% Results of MBD vs. Prop

% Calculate Absolute Error @tf
x_error = abs(x_prop(end)-x_MBD(end));
y_error = abs(y_prop(end)-y_MBD(end));
theta_error = abs(theta_prop(end)-theta_MBD(end));

% Plots
figure(1)
subplot(1,2,1)
plot(x_MBD,y_MBD,'Color','#0072BD','LineWidth',2), hold on
plot(x_prop,y_prop,'o','Color','#0072BD','MarkerSize',6,...
'LineWidth',1)
xlabel('X-axis displacement [m]')
ylabel('Y-axis displacement [m]')
plot(x_MBD(1),y_MBD(1),'pentagram','Color','#EDB120',...
'MarkerSize',4,'LineWidth',3)
plot(x_MBD(end),y_MBD(end),'rx','MarkerSize',12,'LineWidth',2)
legend('MBD','PROP','Start','End')
grid on, axis(crop)
subplot(1,2,2)
plot(t_MBD,theta_MBD,'Color','#0072BD','LineWidth',2), hold on
plot(t_prop,theta_prop,'o','Color','#0072BD','MarkerSize',6,...
'LineWidth',1)
xlabel('Time [sec]')
ylabel('\theta [rad]')
legend('\theta_{MBD}(t)','\theta_{PROP}(t)','Location','Northwest')
grid on

```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX H. FULL FSS SYSTEM SIMULATION SCRIPT

```
% Simulated Full FSS Testing
% Created by Matthew Tungett, Last edit: 4/21/23

%% Script cleanup
close all
clear
clc

%% Simulation

% Load Robot for trajectory planning
robot = importrobot('Kinova Arm/Kinova_7_DoF_arm.urdf', ...
    DataFormat='column');
robot.Gravity = [0 0 -9.81];

% Specify test to be displayed
flag = input("Specify test --> ");

% Control Input

    % Fx = x-axis thrust [N]
    % Fy = y-axis thrust [N]
    % Tz = RW torque [N-m]
    % w1 = waypoint 1 for the Kinova arm [deg]
    % w2 = waypoint for the Kinova arm [deg]
    % crop = adjust plot frame

if flag == 1          % Test 1
    Fx = 0.0;
    Fy = 1.0;
    Tz = 0.0;
    w1 = [180;90;0;0;0;0;0]*(pi/180);
    w2 = [180;90;0;0;0;0;0]*(pi/180);
    crop = [-0.075 0.075 -0.03 0.12];
elseif flag == 2      % Test 2
    Fx = 0.0;
    Fy = 0.0;
    Tz = 0.0;
    w1 = [180;90;0;0;0;0;0]*(pi/180);
    w2 = [0;90;0;0;0;0;0]*(pi/180);
    crop = [-0.04 0.02 -0.01 0.05];
elseif flag == 3      % Test 3
    Fx = 0.0;
    Fy = 0.0;
    Tz = 0.0;
    w1 = [0;0;0;0;0;0;0]*(pi/180);
    w2 = [0;90;0;0;0;0;0]*(pi/180);
    crop = [-0.05 0.005 -0.0275 0.0275];
else                  % Zero control
    Fx = 0.0;
```

```

        Fy = 0.0;
        Tz = 0.0;
        w1 = [0;0;0;0;0;0;0]*(pi/180);
        w2 = [0;0;0;0;0;0;0]*(pi/180);
        crop = [-1 1 -1 1];
    end

% Simulation Time
    endtime = 4;

% Simulation
    simout = sim('Full_FSS.slx','StartTime','0','StopTime',...
        num2str(endtime));

% Unpack Data
    t_MBD = simout.time;
    x_MBD = simout.x;
    y_MBD = simout.y;
    theta_MBD = simout.theta;
    vx_MBD = simout.vx;
    vy_MBD = simout.vy;
    omega_MBD = simout.omega;
    ax_MBD = simout.ax;
    ay_MBD = simout.ay;
    omega_dot_MBD = simout.omega_dot;
    q_MBD = simout.q;
    qd_MBD = simout.qd;
    qdd_MBD = simout.qdd;
    torque_MBD = simout.torque;

%% Results of MBD vs. Prop

% Plots
    figure(1)
    subplot(1,2,1)
    plot(x_MBD,y_MBD,'Color','#0072BD','LineWidth',2), hold on
    xlabel('X-axis displacement [m]')
    ylabel('Y-axis displacement [m]')
    plot(x_MBD(1),y_MBD(1),'pentagram','Color','#EDB120',...
        'MarkerSize',4,'Linewidth',3)
    plot(x_MBD(end),y_MBD(end),'rx','MarkerSize',12,'Linewidth',2)
    legend('Positional path','Start','End')
    grid on, axis (crop)
    subplot(1,2,2)
    plot(t_MBD,theta_MBD,'Color','#0072BD','LineWidth',2), hold on
    xlabel('Time [sec]')
    ylabel('\theta [rad]')
    legend('\theta(t)','Location','Northwest')
    grid on

```

APPENDIX I. VIDEO FILE TO IMAGE DATASTORE SCRIPT

```
%% Movie To Image Datastore
% Created by Matthew Tungett, Last edit: 5/7/23

%% Script Cleanup
close all
clear
clc

%% Movie to Image Datastore

% Load movie
mov_label = 'File Name.mp4'; % **Change label to movie name**
movie=VideoReader(mov_label);

% Number of frames
n = movie.NumberOfFrames;

% Create Datastore
for i = 1:2:n
    frames = read(movie,i);
    imwrite(frames,['File Name\image_' int2str(i), '.jpg']);
    % **change file location and naming**
    im(i)=image(frames);
end
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX J. IMAGE DISTORTION CORRECTION SCRIPT

```
%% Correct Image Distortion
% Created by Matthew Tungett, Last edit: 5/6/23

%% Script Cleanup
close all
clear
clc

%% Calibration

% Load Calibration Images
calibration_imgs = imageDatastore('Calibration');

% Detect calibration pattern
[imagePoints,boardSize] = detectCheckerboardPoints(...
    calibration_imgs.Files);

% World coordinates
squareSize = 22.5; % mm
worldPoints = generateCheckerboardPoints(boardSize,squareSize);

% Calibrate camera
I = readimage(calibration_imgs,1);
imageSize = [size(I,1),size(I,2)];
cameraParams = estimateCameraParameters(imagePoints,worldPoints,...
    'ImageSize',imageSize);

% Show Marker Detection
Im=insertMarker(I,imagePoints(:,:,1));
figure(1)
imshow(Im)

% Remove lens distortion
I = calibration_imgs.readimage(1);
J = undistortImage(I,cameraParams);

% Show results
figure(2)
imshowpair(I,J,'montage');

%% Correct Test Images

% Load Images
test_1_images = imageDatastore('Test 1');
test_2_images = imageDatastore('Test 2');

% Number of Images
n1 = length(test_1_images.Files);
n2 = length(test_2_images.Files);
```

```
% Undistort Images

% Test 1
for i = 1:n1
    J = undistortImage(test_1_images.readimage(i),cameraParams);
    name = ['Test 1 - Corrected\image_',num2str(i),'.jpg'];
    imwrite(J,name)
end

% Test 2
for i = 1:n2
    J = undistortImage(test_2_images.readimage(i),cameraParams);
    name = ['Test 2 - Corrected\image_',num2str(i),'.jpg'];
    imwrite(J,name)
end
```

APPENDIX K. FSS POSITION AND ATTITUDE ESTIMATION SCRIPT

```
%% Position and Attitude Estimation
% Created by Matthew Tungett, Last edit: 5/7/23

%% Script cleanup
close all
clear
clc

%% Estimate Position and Attitude

% Load data
flag1 = input('Test # -->');
flag2 = input('Image # -->');
I = imread(['Test ', num2str(flag1), ' - Corrected\image_', ...
            num2str(flag2), '.jpg']);

% Position and Orientation Estimate
x = 0;      % **input guess**
y = 0;      % **input guess**
theta = 0;  % **input guess**

% Plotting Parameters
x_cord = 0; % **insert x_inertial location**
y_cord = 0; % **insert y_inertial location**
x_pos = x_cord+x;
y_pos = y_cord-y;

% FSS Coordinate System Equations
for i = 1:500
    cross_1(:,i) = i*sind(theta);
    cross_2(:,i) = i*cosd(theta);
end

% Add Coordinates to image
figure(1)
imshow(I)
hold on
plot(x_pos,y_pos,'bo','MarkerSize',160,'LineWidth',2), hold on
plot(x_pos-cross_1,y_pos-cross_2,'g','LineWidth',3), hold on
plot(x_pos+cross_2,y_pos-cross_1,'r','LineWidth',3), hold on
xline(x_cord,'--w','LineWidth',2);
yline(y_cord,'--w','LineWidth',2);
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX L. MEASURED FSS DATA INTERPOLATION SCRIPT

```
%% FSS Data Interpolation
% Created by Matthew Tungett, Last edit: 5/7/23

%% Script cleanup
close all
clear
clc

%% Interpolation

% Time
dt = 1/30; % time step
t = [0:(10*dt):(121/30)]; % time span
t_int = [0:dt:(120/30)]; % interpolated time span

% Test 1 - Estimated and Interpolated Key Framing
x_1 = [-158 -156 -156 -158 -160 -160 -162 -160 -160 -162 -160 -162, ...
        -162];
y_1 = [-249 -249 -249 -255 -257 -257 -259 -259 -259 -259 -259 -260, ...
        -260];
theta_1 = [45 44.8 41.5 35 29.5 24 19 13.75 9.5 4.5 -0.5 -4.75 -8];
x_int_1 = interp1(t,x_1,t_int,'linear');
y_int_1 = interp1(t,y_1,t_int,'linear');
theta_int_1 = interp1(t,theta_1,t_int,'linear');

% Test 2 - Estimated and Interpolated Key Framing
x_2 = [-145 -145 -144 -143 -138 -132 -127 -122 -117 -113 -105 -97 -91];
y_2 = [40 40 40 40 40 40 40 40 40 40 40 40 40];
theta_2 = [2 2 2 2 2.1 2.15 2.3 2.6 3.0 3.3 3.5 3.9 4.2];
x_int_2 = interp1(t,x_2,t_int,'linear');
y_int_2 = interp1(t,y_2,t_int,'linear');
theta_int_2 = interp1(t,theta_2,t_int,'linear');

%% Store Raw Interpolated Data

% Test 1
xlswrite('Raw Interpolated FSS Data/Test 1 - time',t_int);
xlswrite('Raw Interpolated FSS Data/Test 1 - x_hist_pixels',x_int_1);
xlswrite('Raw Interpolated FSS Data/Test 1 - y_hist_pixels',y_int_1);
xlswrite('Raw Interpolated FSS Data/Test 1 - theta_hist_deg',...
        theta_int_1);

% Test 2
xlswrite('Raw Interpolated FSS Data/Test 2 - time',t_int);
xlswrite('Raw Interpolated FSS Data/Test 2 - x_hist_pixels',x_int_2);
xlswrite('Raw Interpolated FSS Data/Test 2 - y_hist_pixels',y_int_2);
xlswrite('Raw Interpolated FSS Data/Test 2 - theta_hist_deg',...
        theta_int_2);
```

```

%% Store Converted Interpolated Data

% Conversion
pixel_2_meters = 0.0009; % meters per pixel
deg_2_rad = pi/180;

% Test 1
xlswrite('Interpolated FSS Data/Test 1 - time',t_int);
xlswrite('Interpolated FSS Data/Test 1 - x_hist_meters',...
        x_int_1*pixel_2_meters-x_int_1(1)*pixel_2_meters);
xlswrite('Interpolated FSS Data/Test 1 - y_hist_meters',...
        y_int_1*pixel_2_meters-y_int_1(1)*pixel_2_meters);
xlswrite('Interpolated FSS Data/Test 1 - theta_hist_rad',...
        theta_int_1*deg_2_rad-theta_int_1(1)*deg_2_rad);

% Test 2
xlswrite('Interpolated FSS Data/Test 2 - time',t_int);
xlswrite('Interpolated FSS Data/Test 2 - x_hist_meters',...
        x_int_2*pixel_2_meters-x_int_2(1)*pixel_2_meters);
xlswrite('Interpolated FSS Data/Test 2 - y_hist_meters',...
        y_int_2*pixel_2_meters-y_int_2(1)*pixel_2_meters);
xlswrite('Interpolated FSS Data/Test 2 - theta_hist_rad',...
        theta_int_2*deg_2_rad-theta_int_2(1)*deg_2_rad);

```

APPENDIX M. MOVIE GENERATION SCRIPT

```
%% FSS Test Movie Generation
% Created by Matthew Tungett, Last edit: 5/7/23

%% Script Cleanup
close all
clear
clc

%% Load Data

% Determine which test to be displayed
test = input('Test # -->');
title = ['Test ', num2str(test), ' - Movie'];

% Load Data
t_int = xlsread(['Raw Interpolated FSS Data\Test ', num2str(test), ...
                ' - time.xls']);
x_int = xlsread(['Raw Interpolated FSS Data\Test ', num2str(test), ...
                ' - x_hist_pixels.xls']);
y_int = xlsread(['Raw Interpolated FSS Data\Test ', num2str(test), ...
                ' - y_hist_pixels.xls']);
theta_int = xlsread(['Raw Interpolated FSS Data\Test ', ...
                    num2str(test), ' - theta_hist_deg.xls']);

%% Movie Generation

% Set figure parameters
fig_num = 1;
mov = figure(fig_num);
pause(1);

% Capture Frames
for k = 1:length(t_int)

    % Clear figure for next plot
    clf

    % Data Index
    x_index = x_int(k);
    y_index = y_int(k);
    theta_index = theta_int(k);

    % Plotting Paramters
    x_cord = 1460; % x_inertial location
    y_cord = 1080; % y_inertial location
    x_pos = x_cord+x_index;
    y_pos = y_cord-y_index;

    % FSS Coordinate System Equations
```

```

    for i = 1:600
        cross_1(:,i) = i*sind(theta_index);
        cross_2(:,i) = i*cosd(theta_index);
    end

% Grab image
I = imread(['Test ',num2str(test),' - Corrected\image_',...
            num2str(k),'.jpg']);

% Plot

% FSS coordinates
imshow(I),hold on
plot(x_pos-cross_1(1:500),y_pos-cross_2(1:500),'g',...
     'LineWidth',3)
plot(x_pos+cross_2(1:500),y_pos-cross_1(1:500),'r',...
     'LineWidth',3)
text(x_pos-cross_1(end),y_pos-cross_2(end),'Y_{FSS}',...
     'FontSize',15,'FontWeight','bold','Color','g')
text(x_pos+cross_2(end),y_pos-cross_1(end),'X_{FSS}',...
     'FontSize',15,'FontWeight','bold','Color','r')

% FSS COM
r = 50;
a = [0 0:.01:r 0];
b = [0 sqrt(r^2-a(2:end-1).^2) 0];
fill(x_pos+a,y_pos+b,'k'), hold on
fill(x_pos-a,y_pos-b,'k')
fill(x_pos+a,y_pos-b,'w')
fill(x_pos-a,y_pos+b,'w')

% Inertial Coordinates
xline(x_cord,'--w','LineWidth',2);
yline(y_cord,'--w','LineWidth',2);
text(2400,y_cord-75,'X_{inertial}','FontSize',15,...
     'FontWeight','bold','Color','w')
text(x_cord+20,250,'Y_{inertial}','FontSize',15,...
     'FontWeight','bold','Color','w')

% Take Snapshot
movieVector(k) = getframe(ffigure(fig_num));

end

% Save Movie
myWriter = VideoWriter(title);
myWriter.FrameRate = 30;
open(myWriter);
writeVideo(myWriter,movieVector);
close(myWriter);

```

APPENDIX N. REAL FSS TESTING VS. SIMULATED MBD MODEL SIMULATION SCRIPT

```
% Simulated Vs. Real FSS Testing
% Created by Matthew Tungett, Last edit: 5/7/23

%% Script cleanup
close all
clear
clc

%% Simulation

% Load Robot for trajectory planning
robot = importrobot('Kinova Arm/Kinova_7_DoF_arm.urdf', ...
    DataFormat='column');
robot.Gravity = [0 0 -9.81];

% Specify test to be displayed
test = input("Specify test --> ");

% Control Input

    % Fx = x-axis thrust [N]
    % Fy = y-axis thrust [N]
    % Tz = RW torque [N-m]
    % w1 = waypoint 1 for the Kinova arm [deg]
    % w2 = waypoint for the Kinova arm [deg]
    % crop = adjust plot frame

if test == 1          % Test 1
    Fx = 0.0;
    Fy = 0.0;
    Tz = 0.0;
    w1 = [45;90;0;0;0;0;270]*(pi/180);
    w2 = [-45;90;0;0;0;0;270]*(pi/180);
    crop = [-0.025 0.005 -0.025 0.005];
else                  % Test 2
    Fx = 0.0;
    Fy = 0.0;
    Tz = 0.0;
    w1 = [0;90;0;0;0;90;270]*(pi/180);
    w2 = [0;0;0;0;0;90;270]*(pi/180);
    crop = [-0.01 0.05 -0.03 0.03];
end

% Simulation Time
endtime = 4;

% Simulation
simout = sim('Full_FSS.slx', 'StartTime', '0', 'StopTime', ...
```

```

        num2str(endtime));

% Unpack Data
t_MBD = simout.time;
x_MBD = simout.x;
y_MBD = simout.y;
theta_MBD = simout.theta;
vx_MBD = simout.vx;
vy_MBD = simout.vy;
omega_MBD = simout.omega;
ax_MBD = simout.ax;
ay_MBD = simout.ay;
omega_dot_MBD = simout.omega_dot;
q_MBD = simout.q;
qd_MBD = simout.qd;
qdd_MBD = simout.qdd;
torque_MBD = simout.torque;

%% Load Measured FSS Data
t_measured = xlsread(['Interpolated FSS Data\Test ', num2str(test), ...
    ' - time.xls']);
x_measured = xlsread(['Interpolated FSS Data\Test ', num2str(test), ...
    ' - x_hist_meters.xls']);
y_measured = xlsread(['Interpolated FSS Data\Test ', num2str(test), ...
    ' - y_hist_meters.xls']);
theta_measured = xlsread(['Interpolated FSS Data\Test ', ...
    num2str(test), ' - theta_hist_rad.xls']);

%% Results of MBD vs. Prop

% Calculate Absolute Error @tf
x_error = abs(x_measured(end)-x_MBD(end));
y_error = abs(y_measured(end)-y_MBD(end));
theta_error = abs(theta_measured(end)-theta_MBD(end));

% Plots
figure(1)
subplot(1,2,1)
plot(x_MBD,y_MBD,'Color','#0072BD','LineWidth',2), hold on

plot(x_measured,y_measured,'Color','#0072BD','LineWidth',1)
xlabel('X-axis displacement [m]')
ylabel('Y-axis displacement [m]')
plot(x_MBD(1),y_MBD(1),'pentagram','Color','#E8B120',...
    'MarkerSize',4,'Linewidth',3)
plot(x_MBD(end),y_MBD(end),'rx','MarkerSize',12,'Linewidth',2)
plot(x_measured(end),y_measured(end),'rx','MarkerSize',12,...
    'Linewidth',2)

legend('Calculated','Measured','Start','End')
grid on, axis (crop)
subplot(1,2,2)
plot(t_MBD,theta_MBD,'Color','#0072BD','LineWidth',2), hold on
plot(t_measured,theta_measured,'Color','#0072BD','LineWidth',1)

```

```
xlabel('Time [sec]')
ylabel('\theta [rad]')
legend('\theta_{calculated}(t)', '\theta_{measured}(t)', 'Location', ...
      'Northwest')
grid on
```

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

- [1] “UCS 2022 satellite database.” USC-Satellite-Database-5-1-20-22. Accessed: Mar. 05, 2023 [Online]. Available: <https://www.ucsusa.org/resources/satellite-database>
- [2] S. Young, “The number of active satellites in space skyrockets . . . literally,” The Equation, blog, July 27, 2021 [Online]. Available: <https://blog.ucsusa.org/syoung/number-of-satellites-skyrockets/>
- [4] Bank of America, “The new space era: expansion of the space economy,” January 27, 2023 [Online]. Available: <https://business.bofa.com/en-us/content/bank-of-america-institute/transformation/expansion-of-the-space-economy-january-2023.html>
- [5] M. Weinzierl and M. Sarang, “The commercial space age is here,” *Harvard Business Review*, Feb. 12, 2021 [Online]. Available: <https://hbr.org/2021/02/the-commercial-space-age-is-here>
- [3] Statista, “Number of active satellites from 1957 to 2022,” March 23, 2023 [Online]. Available: <https://www.statista.com/statistics/897719/number-of-active-satellites-by-year/>
- [6] K. Howard and A. Von Ah, “Large constellations of satellites: mitigating environmental and other effects,” Government Accountability Office, Washington, DC, USA, GAO Report No. GAO-22-105166, 2022.
- [7] National Aeronautics and Space Administration, “State-of-the-art small spacecraft technology,” Moffett Field, CA, USA, 2022 [Online]. Available: https://www.nasa.gov/sites/default/files/atoms/files/2022_soa_full_0.pdf
- [8] A. Cavaciuti, J. Heying, and J. Davis, “In-space servicing, assembly, and manufacturing for the new space economy,” Aerospace Center for Space Policy and Strategy, Arlington, VA, USA, Rep. OTR2022-00778, 2022 [Online]. Available: https://csps.aerospace.org/sites/default/files/2022-07/Cavaciuti-Davis-Heying_ISAM_20220715.pdf
- [9] H. Duke, “On-orbit servicing: opportunities for U.S. military servicing resiliency,” Aerospace Security, September 21, 2021 [Online]. Available: <https://aerospace.csis.org/on-orbit-servicing/>
- [10] National Aeronautics and Space Administration, “Skylab 2: First repair spacewalk,” June 06, 2018 [Online]. Available: <https://www.nasa.gov/feature/skylab-2-first-repair-spacewalk>

- [11] B. Evans, “Fixing solar max: 30 years since mission 41C (part 2),” *AmericaSpace* for space explorers, April 06, 2014 [Online]. Available: <https://www.americaspace.com/2014/04/06/fixing-solar-max-30-years-since-mission-41c-part-2/>
- [12] B. Evans, “Fixing solar max: 30 years since mission 41C (part 1),” *AmericaSpace For Space Explorers*, April 05, 2014 [Online]. Available: <https://www.americaspace.com/2014/04/05/fixing-solar-max-30-years-since-mission-41c-part-1/>
- [13] National Aeronautics and Space Administration, “STS-41C,” February 18, 2010 [Online]. Available: https://www.nasa.gov/mission_pages/shuttle/shuttlemissions/archives/sts-41C.html
- [14] National Aeronautics and Space Administration, “STS-49,” March 31, 2010 [Online]. Available: https://www.nasa.gov/mission_pages/shuttle/shuttlemissions/archives/sts-49.html
- [15] National Aeronautics and Space Administration, “STS-51A,” February 18, 2010 [Online]. Available: https://www.nasa.gov/mission_pages/shuttle/shuttlemissions/archives/sts-51A.html
- [16] National Aeronautics and Space Administration, “Servicing missions.” Accessed: Apr. 18, 2023 [Online]. Available: <https://hubblesite.org/home/mission-and-telescope/servicing-missions>
- [17] European Space Agency, “STP-1,” June 14, 2012 [Online]. Available: <https://www.eoportal.org/satellite-missions/stp-1#summary-of-oe-in-orbit-firsts>
- [18] Defense Advanced Research Projects Agency, “Orbital Express.” Accessed: Apr. 11, 2023 [Online]. Available: <https://www.darpa.mil/about-us/timeline/orbital-express>
- [19] Astroscale, “ELSA-d press kit.” Accessed: Apr. 11, 2023 [Online]. Available: <https://astroscale.com/wp-content/uploads/2021/08/ELSA-d-Press-Kit-2021.pdf>
- [20] National Aeronautics and Space Administration, “Robotic refueling mission.” Accessed: Apr. 11, 2023 [Online]. Available: <https://nexis.gsfc.nasa.gov/>
- [21] Northrop Grumman, “SpaceLogistics: Mission extension vehicle.” Accessed: Apr. 12, 2023 [Online]. Available: <https://www.northropgrumman.com/space/space-logistics-services>
- [22] National Aeronautics and Space Administration, “OSAM-1 mission.” Accessed: Apr. 10, 2023 [Online]. Available: <https://nexis.gsfc.nasa.gov/OSAM-1.html>

- [23] Defense Advanced Research Projects Agency, “Robotic servicing of geosynchronous satellites (RSGS),” Accessed: Apr. 10, 2023 [Online]. Available: <https://www.darpa.mil/program/robotic-servicing-of-geosynchronous-satellites>
- [24] Scuba Diving, “Underwater at NASA’s neutral buoyancy laboratory,” April 27, 2015 [Online]. Available: <https://www.scubadiving.com/nasas-underwater-lab>
- [25] European Space Agency, “Parabolic manoeuvres.” Accessed: Apr. 10, 2023 [Online]. Available: https://www.esa.int/Education/Fly_Your_Thesis/Parabolic_manoeuvres
- [26] University of Surrey, “Docking facilities.” Accessed: Apr. 13, 2023 [Online]. Available: <https://www.surrey.ac.uk/surrey-space-centre/facilities/docking-facilities>
- [27] National Aeronautics and Space Administration, “Space transportation systems: guidance, navigation, and control (GN&C),” Huntsville, AL, USA, Rep. NP-2014-07-82-MSFC, 2014 [Online]. Available: <https://www.nasa.gov/sites/default/files/files/GNC.pdf>
- [28] Naval Postgraduate School, “Spacecraft Robotics Laboratory.” Accessed: Apr. 13, 2023 [Online]. Available: <https://nps.edu/web/srl/facilities>
- [29] R. Zappulla *et al.*, “Floating spacecraft simulator test bed for the experimental testing of autonomous guidance, navigation, and control of spacecraft proximity maneuvers and operations,” *Am. Inst. Aeronaut. Astronaut.*, Sep. 2016 [Online]. Available: <https://doi.org/10.2514/6.2016-5268>
- [30] European Space Agency, “ISS utilization: SPHERES (Synchronized Position Hold Engage Re-orient Experiment Satellites),” June 07, 2012 [Online]. Available: <https://www.eoportal.org/other-space-activities/iss-spheres>
- [31] National Aeronautics and Space Administration, “SPHERES satellites and facilities.” Accessed: Apr. 14, 2023 [Online]. Available: <https://www.nasa.gov/spheres/spheres-satellites>
- [32] E. Ackerman, “How NASA’s Astrobee robot is bringing useful autonomy to the ISS,” *IEEE Spectrum*, Feb. 09, 2017 [Online]. Available: <https://spectrum.ieee.org/how-nasa-astrobee-robot-is-bringing-useful-autonomy-to-the-iss>
- [33] J. Virgili-Llop and M. Romano, “Simultaneous capture and detumble of a resident space object by a free-flying spacecraft-manipulator System,” *Front. Robot. AI*, vol. 6, no. 14, pp. 1–24, Mar. 2019 [Online]. Available: <https://doi.org/10.3389/frobt.2019.00014>

- [34] Ball Aerospace & Technologies Corporation, “Final performance specification 20.3 NMS reaction wheel for the US Naval Postgraduate School,” unpublished.
- [35] *KINOVA Gen3 Ultra lightweight robot User Guide*, 1st ed., Kinova Robotics, Boisbriand, QC, CA, 2022 [Online]. Available: <https://www.kinovarobotics.com/uploads/User-Guide-Gen3-R07.pdf>
- [36] ROS Components, “Gen3.” Accessed: Apr. 17, 2023 [Online]. Available: <http://www.roscomponents.com/en/robotic-arms/234-gen3.html>
- [37] Kinova Robotics, *Gen3 Specifications Sheet*, 2022 [Online]. Available: <https://drive.google.com/file/d/1UbCKUgOjKqTvQNwXWN5gIdBiP9Q4hP5d/view>
- [38] C. Vincent, “Tracking and capture of an object using a robotic manipulator,” M.S. thesis, Dept. of MAE, NPS, Monterey, CA, USA, 2022.
- [39] *Robotiq 2F-85 & 2F-140 for e-series Universal Robots*, 1st ed., Robotiq Inc., Chicago, IL, USA, 2018 [Online]. Available: https://assets.robotiq.com/website-assets/support_documents/document/2F-85_2F-140_Instruction_Manual_e-Series_PDF_20190206.pdf
- [41] Robotiq Inc, “Adaptive grippers product sheet.” Accessed: Mar. 23, 2023 [Online]. Available: <https://blog.robotiq.com/hubfs/Product-sheets/Adaptive%20Grippers/Product-sheet-Adaptive-Grippers-EN.pdf>
- [40] Robotiq Inc, “2F-85 and 2F-140 grippers.” Accessed: Apr. 12, 2023 [Online]. Available: <https://robotiq.com/products/2f85-140-adaptive-robot-gripper>
- [42] P. I. Corke, “A simple and systematic approach to assigning Denavit–Hartenberg parameters,” *IEEE Trans. Robot.*, vol. 23, no. 3, pp. 590–594, Jun. 2007 [Online]. Available: <https://doi.org/10.1109/TRO.2007.896765>
- [43] ROS Wiki, “XML robot description format (URDF).” Accessed: Apr. 09, 2023 [Online]. Available: [http://library.isr.ist.utl.pt/docs/roswiki/urdf\(2f\)XML.html#A.3Cgazebo.3E_element](http://library.isr.ist.utl.pt/docs/roswiki/urdf(2f)XML.html#A.3Cgazebo.3E_element)
- [44] L. Žlajpah, “Simulation in robotics,” *Math. Comput. Simul.*, vol. 79, no. 4, pp. 879–897, Dec. 2008 [Online]. Available: <https://doi.org/10.1016/j.matcom.2008.02.017>
- [45] Kinova Robotics, “ros_kortex,” GitHub, June 2021 [Online]. Available: https://github.com/Kinovarobotics/ros_kortex
- [46] GrabCAD, “2F-140 grippers (Robotiq),” May 17, 2020 [Online]. Available: <https://grabcad.com/library/2f-140-grippers-robotiq-1>

- [47] *Simscape Multibody User's Guide*, R2023a ed., MathWorks, Natick, MA, USA, 2023 [Online]. Available: https://www.mathworks.com/help/pdf_doc/sm/sm_ug.pdf
- [48] A. O'Toole, "Modeling and simulation environment for comparative analysis and synthesis of distributed space logistics missions," M.S. thesis, Dept. of MAE, NPS, Monterey, CA, USA, 2022.
- [49] Gazebo, "About – Gazebo." Accessed: Apr. 17, 2023 [Online]. Available: <https://gazebo.org/about>
- [50] *Simulink User's Guide*, R2023a ed., MathWorks, Natick, MA, USA, 2023 [Online]. Available: https://www.mathworks.com/help/pdf_doc/simulink/simulink_ug.pdf
- [51] *MATLAB Primer*, R2023a ed., MathWorks, Natick, MA, USA, 2023 [Online]. Available: https://www.mathworks.com/help/pdf_doc/matlab/learn_matlab.pdf
- [52] J. Virgili-Llop, M. Romano, and J. Drew, "Spacecraft Robotics Toolkit: An open-source simulator for spacecraft robotic arm dynamic modeling and control," *Int. Conf. Astrodyn. Tools Tech.*, 2016 [Online]. Available: https://indico.esa.int/event/111/contributions/406/attachments/329/370/icatt16-mansw_r2.pdf
- [53] *Robotics System Toolbox User's Guide*, R2023a ed., MathWorks, Natick, MA, USA, 2023 [Online]. Available: https://www.mathworks.com/help/pdf_doc/robotics/robotics_ug.pdf
- [54] R. Featherstone, "The calculation of robot dynamics using articulated-body inertias," *Int. J. Robot. Res.*, vol. 2, no. 1, pp. 13–30, Mar. 1983 [Online]. Available: <https://doi.org/10.1177/027836498300200102>.
- [55] M. W. Walker and D. E. Orin, "Efficient dynamic computer simulation of robotic mechanisms," *J. Dyn. Syst. Meas. Control*, vol. 104, no. 3, pp. 205–211, Sep. 1982 [Online]. Available: <https://doi.org/10.1115/1.3139699>.
- [56] *Simscape User's Guide*, R2023a ed., MathWorks, Natick, MA, USA, 2023 [Online]. Available: https://www.mathworks.com/help/pdf_doc/simscape/simscape_ug.pdf
- [57] A. Banerjee, J. Haluska, S. Satpute, D. Kominiak, and G. Nikolakopoulos, "Slider: On the design and modeling of a 2D floating satellite platform," working paper, Luleå University of Technology, Luleå, SE, 2021 [Online]. Available: <https://arxiv.org/pdf/2101.06335.pdf>
- [58] *Computer Vision Toolbox User's Guide*, R2023a ed., MathWorks, Natick, MA, USA, 2023 [Online]. Available: https://www.mathworks.com/help/pdf_doc/vision/vision_ug.pdf

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
Ft. Belvoir, Virginia
2. Dudley Knox Library
Naval Postgraduate School
Monterey, California



DUDLEY KNOX LIBRARY

NAVAL POSTGRADUATE SCHOOL

WWW.NPS.EDU

WHERE SCIENCE MEETS THE ART OF WARFARE