University of Glasgow

Gordon, Robert Alexander Douglas (2025) *Attitude control and design of satellites with flexible structures using Inverse Simulation.* PhD thesis.

https://theses.gla.ac.uk/85386/

Copyright and moral rights for this work are retained by the author

A copy can be downloaded for personal non-commercial research or study, without prior permission or charge

This work cannot be reproduced or quoted extensively from without first obtaining permission from the author

The content must not be changed in any way or sold commercially in any format or medium without the formal permission of the author

When referring to this work, full bibliographic details including the author, title, awarding institution and date of the thesis must be given

Enlighten: Theses
https://theses.gla.ac.uk/
research-enlighten@glasgow.ac.uk

Attitude Control and Design of Satellites with Flexible Structures using Inverse Simulation

Robert Alexander Douglas Gordon

Submitted in fulfilment of the requirements for the Degree of Doctor of Philosophy

---

August, 2025

University of Glasgow

James Watt School of Engineering
College of Science and Engineering
University of Glasgow

"For all its material advantages, the sedentary life has left us edgy, unfulfilled. Even after 400 generations in villages and cities, we haven't forgotten. The open road still softly calls, like a nearly forgotten song of childhood. We invest far-off places with a certain romance. This appeal, I suspect, has been meticulously crafted by **natural selection** as an essential element in our survival. Long summers, mild winters, rich harvests, plentiful game—none of them lasts forever. It is beyond our powers to predict the future. Catastrophic events have a way of sneaking up on us, of catching us unaware. Your own life, or your band's, or even your species' might be owed to a restless few—drawn, by a craving they can hardly articulate or understand, to undiscovered lands and new worlds. Herman Melville, in Moby Dick, spoke for **wanderers** in all epochs and meridians: **'I am tormented with an everlasting itch for things remote. I love to sail forbidden seas...'** "

— Carl Sagan, Pale Blue Dot: A Vision of the Human Future in Space

Abstract

Ensuring the stable and accurate pointing of a satellite's instrumentation is the task of the attitude control system, a problem made substantially more complex by the influence of flexible structural dynamics often exhibited by satellites. The inclusion of flexible structures is necessitated by the desire to maximise the satellite's functionality while ensuring its mass and form factor are conducive to being launched into orbit. Commonly, satellites will make use of lightweight and flexible, deployable appendages to facilitate the mounting of additional instrumentation, larger antennas, and extra solar panels. In addition, the construction of larger flexible space structures, using advancing orbital assembly and manufacturing technologies, opens the door to complex missions such as space stations, space telescopes, large solar sail spacecraft, orbital solar reflectors, and space-based solar power satellites. While the variety of applications for flexible space structures may be wide, each will require stable and accurate attitude control of the satellite platform.

A range of attitude control methodologies for satellites with flexible structures has been studied previously, each attempting to achieve good tracking performance by imitating the satellite's inverse dynamic behaviour. Many of these approaches are based on analytical techniques, requiring the control solution to be derived from a mathematical model that approximates the satellite's dynamics. This thesis proposes that, if a sufficiently accurate mathematical model is available, then an alternative, highly-computational, numerical approach known as Inverse Simulation could instead be employed. Having already been demonstrated as a viable control methodology in several fields of research, such as planetary rovers and helicopters, this thesis will investigate the application of the generalised integral Inverse Simulation algorithm for attitude control of satellites with flexible structures.

As part of the thesis, several mathematical models for the Inverse Simulation numerical process to solve are derived. This includes models that assume only rigid body behaviour, those that consider the flexible dynamics of deployable appendages, and finally a modular model for large flexible space structures. These models also account for the dynamics of various actuator types, some of which may have an impact on the stability and performance of the integral Inverse Simulation algorithm.

The attitude reorientation manoeuvres used to drive the Inverse Simulation solution are provided using a mixture of polynomials, spherical interpolation, and splines. Before implementation as a control technique, the effect of each of the Inverse Simulation's hyperparameters is studied to optimise the accuracy and computational efficiency of the algorithm. Additionally, issues of numerical stability specific to the application of satellites with flexible structures are identified and addressed.

It is found that the integral Inverse Simulation algorithm does provide a more flexible, practical, and accurate method for solving the inverse attitude dynamics of a satellite with flexible structures than alternative analytical techniques. The generality of the numerical process allows for a ready-to-go control solution, provided a suitable up-to-date model is available. The control methodology is however highly sensitive to model uncertainty and requires significant computational power to compute, somewhat limiting its practical use. Additional online parameter estimation and improved computational efficiency of the algorithm will therefore be required if the control approach is to be realistically used.

Inverse Simulation is also employed to improve attitude path planning, evaluating the performance of prospective paths to minimise excitation of the flexible structural dynamics. One of the most common applications of Inverse Simulation within the literature, as a conceptual design tool, is also demonstrated for satellites with flexible structures. It is shown that Inverse Simulation allows for the performance of different actuator configurations for large space structures to be assessed, informing designs that minimise structural deformation during slew manoeuvres. A final contribution of the thesis is the development of a faster integral Inverse Simulation algorithm based on Broyden's method, improving performance particularly when a large number of inputs/actuators are in use.

Acknowledgements

I would like to first start by thanking my supervisors Dr. Matteo Ceriotti and Dr. Kevin Worrall for their invaluable support and advice over these last 4 years. They were there to reassure me when I felt unsure of myself, and push me when I needed pushing.

I would also like to thank all those in the Space and Exploration Technology Group (SET), and the wider gang who call James Watt North Level 4 home. Thank you for your warm friendship, mutual support, the many sessions of collective catharsis over coffee; and the pints in the pub when the coffee wasn't enough. I must also thank my friends outside of the University for providing many welcome distractions and for always being there for me.

Finally, I would also like to thank my whole family for their love and support, particularly my parents David and Jane for raising me, and giving me the opportunities that allowed me to reach this point. Credit must also go to my Grampa George, who I have to thank for feeding my early interest in science, space, and engineering. Thank you also to Mairi for keeping me sane throughout the past couple of years, and for providing some last-minute proofreading and encouragement to get this thesis over the finish line.

Declaration

I hereby declare that except where specific reference is made to others, this submitted thesis is entirely my own original work, and has not been submitted in whole or in part for any other degree or qualification.

Robert Alexander Douglas Gordon

Glasgow, Scotland
7th August, 2025

Contents

Acknowledgements **iv**

Declaration **v**

List of Tables **x**

List of Figures **xi**

List of Algorithms **xviii**

Nomenclature **xix**

List of Acronyms xix
Notation xx
Vectors xx
Matrices xxi
Scalars xxi
Subscripts and superscripts xxii
Sets xxiii

1 Introduction **1**

1.1 Motivation 3
1.1.1 Deployable Appendages 4
1.1.2 Large Space Structures 5
1.1.3 Flexible Dynamics 10
1.2 Literature Review 11
1.2.1 Mathematical Modelling 12
1.2.2 Attitude Control Actuators 14
1.2.3 Attitude Control Methods 15
1.2.4 Inverse Simulation 20
1.3 Aims and Objectives 28
1.4 Thesis Outline 29
1.4.1 Contributions to Literature 30

2 Inverse Simulation of Satellite Attitude Dynamics 32
2.1 The Inverse Simulation Method . . . 32
2.1.1 Forward Simulation . . . 33
2.1.2 Integral Inverse Simulation Algorithm . . . 33
2.2 Application of Inverse Simulation . . . 38
2.2.1 Inverse Simulation Solutions . . . 39
2.2.2 Hyperparameter Study . . . 42
2.2.3 Numerical Stability . . . 51
2.3 Flexibility of Inverse Simulation . . . 58
2.3.1 Euler Angle Paths . . . 59
2.3.2 Different Output Constraints . . . 61
2.4 Practicality of Inverse Simulation . . . 63
2.4.1 Nonlinear Dynamic Inversion (NDI) . . . 64
2.4.2 Comparison of Inverse Simulation and Nonlinear Dynamic Inversion (NDI) . . . 68
2.5 Summary . . . 76

3 Attitude Control of Satellites with Flexible Appendages using Inverse Simulation 78
3.1 Development of a Multi-Fidelity Model . . . 79
3.1.1 Hybrid Coordinate Approach . . . 79
3.1.2 Rigid-Body Dynamics . . . 83
3.1.3 Assumed Modes Model of a Flexible Beam Appendage . . . 84
3.1.4 Finite Element Method (FEM) Model of a Flexible Beam Appendage 85
3.2 Inverse Simulation of Satellites with Flexible Appendages . . . 90
3.2.1 Numerical Integration Stability . . . 91
3.2.2 Speed of Inverse Simulation . . . 96
3.3 Feed-Forward Control Using Inverse Simulation . . . 100
3.3.1 Attitude-Tracking Control Designs . . . 101
3.3.2 Attitude-Tracking Control Stability . . . 105
3.3.3 Effects of Model Fidelity and Uncertainty . . . 108
3.4 Attitude Path Planning using Inverse Simulation . . . 112
3.4.1 Attitude Path Shaping . . . 113
3.4.2 Attitude Path Optimisation . . . 116
3.4.3 Minimisation of Appendage Flexing . . . 118
3.5 Utilisation of Existing Simulation Frameworks . . . 124
3.5.1 Basilisk Astrodynamic Simulation Framework . . . 125
3.5.2 Inverse Simulation Implimentation . . . 126
3.5.3 Lumped Element Appendage Modelling . . . 128

3.5.4 Results . . . 131
3.6 Summary . . . 133

4 Design of Large Flexible Space Structures using Inverse Simulation 135
4.1 Development of a Modular Mathematical Model . . . 136
4.1.1 Structural Forces . . . 136
4.1.2 Actuator Forces . . . 139
4.1.3 Dynamics and Kinematics . . . 149
4.1.4 Linearisation of Structural Dynamics . . . 153
4.1.5 Actuator Mass Scaling Laws . . . 156
4.2 Inverse Simulation for Satellite Design . . . 163
4.2.1 Inverse Simulation of Overactuated Systems . . . 164
4.2.2 Reaction Wheel-Set (RWS) Example . . . 166
4.2.3 Numerical Issues Associated with Magnetic Torques (MAG and CL) . . . 168
4.3 An Investigation into Attitude Control Actuators for Large Flexible Space Structures . . . 171
4.3.1 Design of a Large Flexible Space Structure . . . 172
4.3.2 Single Actuator Type Comparison . . . 174
4.3.3 Actuator Combinations for Larger Space Structures . . . 177
4.4 Summary . . . 183

5 A Faster Inverse Simulation Method 185
5.1 Modified Inverse Simulation Algorithm . . . 186
5.1.1 Broyden's Method for Faster Inverse Simulation . . . 188
5.2 Comparison to Finite Differencing Methods . . . 191
5.2.1 Control Affine System . . . 193
5.2.2 Control Non-affine System . . . 195
5.2.3 Flexible Satellite System . . . 197
5.3 Issues with Jacobian Null Space . . . 199
5.3.1 Orthogonality of the Broyden Update . . . 199
5.3.2 Robust Broyden's Method for Faster Inverse Simulation . . . 202
5.3.3 Performance Comparison . . . 204
5.4 Summary . . . 206

6 Conclusions and Future Work 207
6.1 Conclusions . . . 207
6.1.1 Aim 1 . . . 207
6.1.2 Aim 2 . . . 209

6.1.3 Aim 3 . . . 210
6.2 Future Work . . . 211
6.3 Final Remarks . . . 213

Appendices **216**

A Frames, Vectors and Matrices **216**
A.1 Notation . . . 216
A.2 Derivatives . . . 217
A.3 Rotations . . . 218
A.4 Inverse Matrices . . . 219

B Attitude Representations **221**
B.1 Euler Angles . . . 221
B.2 Quaternions . . . 223
B.3 Kinematics . . . 225
B.4 Spherical-Linear-Interpolation (SLERP) . . . 225

C Dynamics **228**
C.1 Newtonian . . . 228
C.2 Lagrangian . . . 229

D Numerical Methods **230**
D.1 Newton's Method . . . 230
D.1.1 Systems of Equations . . . 231
D.1.2 Optimistation . . . 232
D.2 Sequential-Quadratic-Programming (SQP) . . . 235
D.2.1 Equality Constraints . . . 235
D.2.2 Inequality Constraints . . . 238
D.3 Runge-Kutta Numerical Integration . . . 240
D.3.1 Explicit Runge-Kutta Methods (ERK) . . . 241
D.3.2 Implicit Runge-Kutta Methods (IRK) . . . 243

E Derivation of Rigid-Body Mathematical Models **245**
E.1 External Torques . . . 245
E.2 Magnetic Torques . . . 246
E.3 Reaction Wheels . . . 247
E.3.1 First-Order Actuator Response . . . 248

List of Tables

2.1 Inverse Simulation hyperparameter summary . . . 38
2.2 Rigid-body satellite parameters . . . 39
2.3 SLERP rest-to-rest slew parameters . . . 39
2.4 Inverse Simulation hyperparameters . . . 40
2.5 InvSim hyperparameter monte-carlo ranges . . . 43
2.6 Rigid-body CubeSat parameters . . . 59

3.1 Model parameters of a satellite with flexible appendages . . . 88
3.2 InvSim numerical integration speed comparison: InvSim hyperparameters 96
3.3 PD control gains, $\bar{\omega}_\zeta \approx 1$ rad/s, $\bar{\zeta} \approx 0.7$ . . . 106
3.4 Hybrid controller: InvSim hyperparameters . . . 107
3.5 Monte-Carlo parameter uncertainties . . . 109
3.6 Attitude path planning constraints . . . 119
3.7 Selected path planning results: Unconstrained body rate . . . 119
3.8 Selected path planning results: Constrained body rate . . . 122
3.9 *Basilisk*: InvSim hyperparameters . . . 132

4.1 Static deflection verification . . . 156
4.2 Model parameters and manoeuvre requirements for LSS . . . 160
4.3 Large flexible space structures: InvSim hyperparameters . . . 167
4.4 Modal frequencies (shapes in Fig. 4.20) . . . 175
4.5 Summary of attitude control actuator investigation . . . 184

5.1 Comparison of finite differencing and Broyden's method: InvSim hyperparameters . . . 193
5.2 Comparison of Jacobian calculation methods: Control affine system . . . 194
5.3 Comparison of Jacobian calculation methods: Control-nonaffine system 196
5.4 Comparison of Jacobian calculation methods: Flexible satellite system . 197
5.5 Parameters of a rigid-body satellite with magnetic-only control test case 200
5.6 Issues of Jacobian null space (Rigid-body with magnetic torques): InvSim hyperparameters . . . 201
5.7 Comparison of Jacobian calculation methods: Rigid-body satellite with magnetic torque . . . 205

List of Figures

1.1 Humanity's urge to reach for the stars (Image courtesy of NASA/ESA) . 1
1.2 Tracking of the California drought in 2014 captured by the Terra Earth observation satellite (Image courtesy of NASA/GSFC/METI/ERS-DAC/JAROS, and U.S./Japan ASTER Science Team) 2
1.3 Mars Cube One (MarCO) with deployed solar panels and antenna [9] (Images courtesy of NASA/JPL-Caltech) 4
1.4 Voyager spacecraft with deployed boom [19] (Images courtesy of NASA/JPL-Caltech) 5
1.5 Re-usable SpaceX Falcon Heavy launch vehicle (Images courtesy of SpaceX licensed under CC BY-NC 2.0) 6
1.6 Example of LSS applications: Solar power from space concepts 7
1.7 Example of LSS applications: Space stations (Images courtesy of NASA/JSC) 7
1.8 In-Step Inflatable Antenna Experiment (IAE) (Images courtesy of NASA/JSC) 8
1.9 Tether Unlimited Inc's orbital manufacturing technologies (Images courtesy of NASA/NTRS [43]) 9
1.10 Spacecraft that experienced flexible dynamic phenomena 11

2.1 Comparison of Forward Simulation and Inverse Simulation 34
2.2 Discretisation of the desired output $\boldsymbol{y}_d$ 34
2.3 InvSim solution: rigid-body satellite with external torque input 41
2.4 NASA standard 4-wheel configuration [81] 42
2.5 InvSim solution: rigid-body satellite with three reaction wheels 43
2.6 InvSim solution: rigid-body satellite with four reaction wheels 44
2.7 Effect of discretisation timestep T and relative integration step size $\Delta t/T$ on the attitude pointing error integrated across the entire maneouvre $\int_0^{t_f} \delta\theta \; dt$. Results shown when InvSim targets output position ($\nu = 0$), velocity ($\nu = 1$) and acceleration ($\nu = 2$) 45
2.8 Acceleration ($\nu = 2$) 45

2.9 Effect of discretisation timestep T and NR minimum perturbation Δu_{min} on the attitude pointing error integrated across the entire maneouvre $\int_0^{t_f} \delta\theta \; dt$. Results shown when InvSim targets output position ($\nu = 0$), velocity ($\nu = 1$) and acceleration ($\nu = 2$) 46
2.10 Effect of discretisation timestep T and NR tolerance $\delta\overset{(\nu)}{\dot{y}}_{tol}$ on the attitude pointing error integrated across the entire maneouvre $\int_0^{t_f} \delta\theta \; dt$. Results shown when InvSim targets output position ($\nu = 0$), velocity ($\nu = 1$) and acceleration ($\nu = 2$) 47
2.11 Effect of discretisation timestep T and NR tolerance $\delta\overset{(\nu)}{\dot{y}}_{tol}$ on the mean number of NR iterations per discretisation step executed during the InvSim. Results shown when InvSim targets output position ($\nu = 0$), velocity ($\nu = 1$) and acceleration ($\nu = 2$) 49
2.12 Effect of discretisation timestep T and NR tolerance $\delta\overset{(\nu)}{\dot{y}}_{tol}$ on the total number of model calls executed during the InvSim. Results shown when InvSim targets output position ($\nu = 0$), velocity ($\nu = 1$) and acceleration ($\nu = 2$) 50
2.13 Effect of discretisation timestep T and relative integration step size $\Delta t/T$ on the total number of model calls executed during the InvSim. Results shown when InvSim targets output position ($\nu = 0$), velocity ($\nu = 1$) and acceleration ($\nu = 2$) 51
2.14 Effect of Jacobian inaccuracies on InvSim input solution 52
2.15 Effect of actuator dynamics on InvSim input solution, $\nu = 0$ 54
2.16 Effect of actuator dynamics on InvSim input solution, $\nu = 1$, $T = 0.1$ s . 55
2.17 Magnetic-only control with diagonal inertia matrix and full attitude constrained $\overset{(\nu)}{\dot{\boldsymbol{y}}}_d = \overset{(\nu)}{\dot{\boldsymbol{q}}}^{B/I}$ 57
2.18 Magnetic-only control with non-diagonal inertia matrix and looser constraints $\overset{(\nu)}{\dot{y}}_d = \overset{(\nu)}{\dot{q}}_1^{B/I}$ 57
2.19 Piecewise polynomial paths defined in $\theta_y - \theta_z$ plane 61
2.20 Euler angles attitude-tracking for a non-spinning CubeSat 62
2.21 Euler angles attitude-tracking for a spinning CubeSat 63
2.22 Semi-analytical NDI solution 67
2.23 InvSim vs NDI: External torque system, comparison of input solution, InvSim $\nu = 1$, NDI $\nu = 2$ 69
2.24 InvSim vs NDI: External torque system, comparison of output tracking, InvSim $\nu = 1$, NDI $\nu = 2$ 69
2.25 InvSim vs NDI: External torque system, comparison of output tracking, InvSim $\nu = 2$, NDI $\nu = 2$ 70

2.26 InvSim vs NDI: Reaction wheel system, comparison of output tracking, InvSim $\nu = 1$, NDI $\nu = 2$, $T = 0.1$ s . . . 71
2.27 InvSim vs NDI: Reaction wheel system, comparison of output tracking, InvSim $\nu = 1$ $T = 0.1$ s, NDI $\nu = 2$ $T = 0.01$ s . . . 71
2.28 InvSim vs NDI: Reaction wheel system with first-order response, comparison of output tracking, InvSim $\nu = 1$, NDI $\nu = 3$ correct input solution Eq. (2.46) . . . 72
2.29 InvSim vs NDI: Reaction wheel system with first-order response, comparison of output tracking, InvSim $\nu = 1$, NDI $\nu = 2$ incorrect input solution Eq. (2.40) assuming instant torque . . . 72
2.30 InvSim vs NDI: Reaction wheel system with first-order response, comparison of input solution . . . 73
2.31 NDI: Magnetic control system, output tracking, non-diagonal inertia . . 74
2.32 InvSim vs NDI: Magnetic control system, comparison of output tracking, diagonal inertia, InvSim using $\boldsymbol{u}_0 = \boldsymbol{u}[(k-1)T]$ . . . 74
2.33 InvSim vs NDI: Magnetic control system, comparison of input solution, non-diagonal inertia, InvSim using $\boldsymbol{u}_0 = \boldsymbol{u}[(k-1)T]$ . . . 75
2.34 InvSim vs NDI: Magnetic control system, comparison of input solution, non-diagonal inertia, InvSim using $\boldsymbol{u}_0 = 0$ . . . 76
2.35 InvSim vs NDI: Speed comparison . . . 77

3.1 Satellite with two flexible appendages . . . 79
3.2 Position of mass $\rho_i dl$ along each appendage . . . 81
3.3 FEM appendage: two-element example [161] . . . 85
3.4 Comparison of first four normalised modal shapes of continuous and FEM beams [161] . . . 88
3.5 Natural frequency convergence of first two flexible modes [161] . . . 89
3.6 Modal shape convergence of first two flexible modes [161] . . . 89
3.7 Stability region for explicit RK4 integration. Area of stability (white), and instability (shaded blue) . . . 93
3.8 Stability region for implicit 2^{nd} order runge-kutta (trapezoidal) integration. Area of stability (white), and instability (shaded) . . . 94
3.9 Flexible satellite poles plotted on the $\lambda\Delta t$ plane shown to lie within the stability region for explicit RK4 integration. Area of stability (white), and instability (shaded blue) . . . 95
3.10 Effect of number of finite elements N_{el} on the speed of InvSim for explicit RK4 and implicit trapezoidal integration methods. $\Delta t = 0.1$ s for trapezoidal. . . . 97

3.11 Effect of NR tolerance $\delta\dot{y}_{tol}^{(\nu)}$ on the speed of InvSim for explicit RK4 and implicit trapezoidal integration methods. $\Delta t = 0.1$ s for trapezoidal . . . 98
3.12 Effect of NR tolerance $\delta\dot{y}_{tol}^{(\nu)}$ on the speed of InvSim for explicit RK4 and implicit trapezoidal integration methods. Comparing $\Delta t = 0.1$ s and $\Delta t = 0.05$ s for trapezoidal integration for the $N_{el} = 10$ case 99
3.13 Traditional PD tracking control structure 102
3.14 Hybrid control: Linear PD feedback with offline InvSim feedforward compensation 105
3.15 Comparison of the magnitude of a 7^{th} order polynomial and bang-bang acceleration profile 106
3.16 Hybrid control unstable response: $\bar{\omega}_\zeta \approx 2$ rad/s, $\bar{\zeta} \approx 0.7$ 107
3.17 Hybrid control stable response: $\bar{\omega}_\zeta \approx 1$ rad/s, $\bar{\zeta} \approx 0.7$ 108
3.18 Mean pointing error Monte-Carlo results: with parameter uncertainty (top), and without parameter uncertainty (bottom) 110
3.19 Sensitivity to scalar inertia uncertainty (including zoomed area) 110
3.20 Sensitivity to errors in flexible dynamics 111
3.21 Linear interpolation on the x-y 2D plane 114
3.22 Discrete points of cubic linear 2D interpolation 115
3.23 Distance between successive interpolated points (original and re-parameterised) 115
3.24 Projections of the body axes on a unit sphere for two example rest-to-rest SQUAD slew manoeuvres 117
3.25 InvSim based quaternion path planning algorithm 118
3.26 Convergence of the cost function $s(\boldsymbol{x}(t))$ (unconstrained body rate) . . . 120
3.27 Paths of each body axis projected onto the unit sphere during slew manoeuvres (unconstrained body rate) 121
3.28 First model coordinate $\eta_{i,1}$ of each appendage (unconstrained body rate) 121
3.29 Satellite body rate $\boldsymbol{\omega}_B$ during each slew manoeuvre (unconstrained body rate) 122
3.30 Convergence of the cost function $s(\boldsymbol{x}(t))$ (constrained body rate) 123
3.31 First model coordinate $\eta_{i,1}$ of each appendage (constrained body rate) . 123
3.32 Satellite body rate $\boldsymbol{\omega}_B$ during each slew manoeuvre (constrained body rate) 124
3.33 Paths of each body axis projected onto the unit sphere during slew manoeuvres (constrained body rate) 124
3.34 *Basilisk* model of a satellite with two flexible appendages 126
3.35 Jacobian calculation using *Basilisk* 128
3.36 Example of an $N_{el} = 2$ lumped element flexible appendage 129

3.37 Relative potential energy of a lumped element beam relative to a continuous Euler-Bernoulli beam . . . 131
3.38 InvSim solution from *Basilisk* . . . 132

4.1 Example of LSS planar structure using point-mass model, each connection between point masses produces structural forces as seen in Fig. 4.2 and Eq. (4.2) . . . 137
4.2 Structural forces . . . 138
4.3 Example of a single reaction wheel (image courtesy of NASA) . . . 140
4.4 Reaction wheel torque converted to a force applied to the mass element i 141
4.5 Example of a magnetic coil that can be embedded axially within an LSS structure [181] . . . 142
4.6 Embedded axial magnetorquer (MAG) . . . 142
4.7 Large current loops (CL) . . . 143
4.8 Example of an RCD patch used on the IKAROS solar sail [88] . . . 145
4.9 Reflective control device actuator (RCD) configuration for rotations about longitudinal axes. Showing fully reflecting (yellow), and fully absorbing (grey) patches . . . 145
4.10 Example of a moving mass actuator configuration for a solar sail [183]. Showing the direction of the sun ($\boldsymbol{n}_s$), centre of mass (cm), centre of pressure (cp), and resultant torque $\mathbf{T}$ . . . 146
4.11 Moving mass actuator (MM): Actuator mass (red) and the affected 8 point-masses in the lattice (yellow) . . . 148
4.12 Static deflection of the model with distributed load (side view) . . . 156
4.13 System design process with Inverse Simulation . . . 164
4.14 Inverse simulation using NR and SQP solvers . . . 165
4.15 InvSim example solution: single centralised RWS, $L = 10$ m, $(EI)_{tot} = 1$ Nm^2. a outputs, b inputs. . . . 166
4.16 Current-loop (CL) actuator: Comparison of different axis constraints when using NR method (InvSim-NR) . . . 168
4.17 Magnetic field direction in body axes during 90-deg slew manoeuvre about the positive $\hat{\boldsymbol{x}}^B$ axis . . . 169
4.18 Comparison of effects of initial input estimate when using a InvSim-NR and b InvSim-SQP . . . 170
4.19 Actuator scaling results. Assumes reflector and solar cells are oriented at 45 deg to the sun. Assumes magnetorquers orientated 45 deg to magnetic field $\boldsymbol{b}_{mag}$ . . . 173
4.20 First nine flexible modal shapes of a planar homogenous LSS: $L = 105$ m (frequencies in Table 4.4) . . . 174

4.21 RWS positions within LSS structure (top-down view) 175
4.22 Comparison of LSS deformation for one central RWS vs 4 distributed RWS. $L = 10$ m, $(EI)_{tot} = 1\text{Nm}^2$. a total torque, b deformation across surface at time of max RMS deformation for the centralised configuration, c RMS deformation during the manoeuvre for both configurations, d deformation across surface at time of max RMS deformation for the distributed configuration 176
4.23 Compare deformation of LSS for different actuator types: $L = 10$ m, $(EI)_{tot} = 1$ Nm^2. a zoomed out, b zoomed in 177
4.24 Compare deformation of LSS for different actuator types: $L = 10$ m, $(EI)_{tot} = 100$ Nm^2. a zoomed out, b zoomed in 178
4.25 RCD actuation only: $L = 105$ m, $(EI)_{tot} = 200$ Nm^2. a outputs, b inputs 178
4.26 One RWS only: $L = 105$ m, $(EI)_{tot} = 200$ Nm^2. a outputs, b inputs, c RMS deformation during manoeuvre, d deformation across surface at time of max RMS deformation 179
4.27 Combined single RWS and RCD actuator: $L = 105$ m, $(EI)_{tot} = 200$ Nm^2. a output, b input, c RMS deformation during manoeuvre, d deformation across surface at time of max RMS deformation 180
4.28 Mean RMS deformation for combined single RWS and MAG with varying total magnetic dipole: $L = 105$ m, $(EI)_{tot} = 200$ Nm^2. a mean RMS deformation, b total LSS mass 181
4.29 Combined single RWS and MAG configuration, total magnetic dipole 100 kAm^2, assuming no MAG mass: $L = 105$ m, $(EI)_{tot} = 200$ Nm^2. a output, b input, c RMS deformation during manoeuvre, d deformation across surface at time of max RMS deformation 182
4.30 CL actuator only: $L = 105$ m, $(EI)_{tot} = 200$ Nm^2. a outputs, b inputs, c RMS deformation during manoeuvre, d deformation across surface at time of max RMS deformation 183

5.1 Automatic Differentiation example (forward mode). $f(x_1, x_2) = \sin(x_1) + x_1 x_2$ 187
5.2 Control affine system InvSim solution 195
5.3 Control affine system InvSim convergence 195
5.4 Control non-affine system InvSim solution 196
5.5 Control non-affine system InvSim convergence 196
5.6 Flexible satellite system InvSim solutions 198
5.7 Flexible satellite system InvSim convergence 198
5.8 Magnetic field direction in body axes during 90-deg slew manoeuvre about the positive $\hat{\boldsymbol{y}}^B$ axis 200

5.9 Rigid-body satellite with magnetic torque InvSim control solution 201
5.10 Rigid-body satellite with magnetic torque InvSim control solution with null space compensation 204
5.11 Rigid-body satellite with magnetic torque InvSim convergence with null space compensation 205

6.1 Future experimental setup for testing real-time hardware: Simulated satellite platform on an air bearing, equipped with 3 reaction wheels, and an onboard flight computer. Helmholtz cage allows for a simulated magnetic field to test magnetorquer configurations. 212

A.1 Comparison between a coordinate transformation and active rotation . . 219

B.1 *Z-Y-X* Euler rotation sequence 222
B.2 Projections of the body axes on a unit sphere during rest-to-rest SLERP slew manoeuvres 227

D.1 Solving for a root of $f(x)$ using Newton-Raphson (NR) method. $f(x) = x^4 - 2$ 231
D.2 Finding minimum of $f(x_1, x_2)$ using gradient descent and Newton's method. Rosenbrock function: $f(x_1, x_2) = (1 - x_1)^2 + 100(x_2 - x_1^2)^2$ . . . 234
D.3 Example of an equality constraint $c(x_1, x_2) = 2.78(x_1 - 0.4)^2 + 6.25x_2^2 - 1$. Rosenbrock function: $f(x_1, x_2) = (1 - x_1)^2 + 100(x_2 - x_1^2)^2$ 236

E.1 Rigid-body satellite with external torques 246
E.2 Magnetorquer: plane of possible torques 247
E.3 Rigid-body satellite with reaction wheels 247

List of Algorithms

1 Integral Inverse Simulation 37

2 Inverse Simulation with *Basilisk* 127

3 Integral Inverse Simulation with SQP solver (InvSim-SQP) 166

4 Modified Broyden-based Inverse Simulation 191

5 Robust Modified Broyden-based Inverse Simulation 203

Nomenclature

List of Acronyms

BFGS Broyden–Fletcher–Goldfarb–Shanno

CL Current loops

DCM Direction cosine matrix

ERK Explicit Runge-Kutta

FEM Finite element method

GEO Geostationary orbit

InvSim Inverse Simulation

IRK Implicit Runge-Kutta

ISS International Space Station

KKT Karush–Kuhn–Tucker

LEO Low-Earth orbit

LSS Large space structure

LTI Linear time-invariant

MAG Embedded magnetorquer

MM Moving mass

MP Moore-Penrose

MPC Model-predictive control

NDI Nonlinear dynamic inversion

NR Newton-Raphson

ODE Ordinary differential equation

PD Proportional-derivative

PID Proportional-integral-derivative

QN Quasi-Newton

QP Quadratic problem

RCD Reflectivity control device

RK4 4^{th} order Runge-Kutta

RWS Reaction-wheel set

SBSP Space-based solar power

SLERP Spherical-linear-interpolation

SMC Sliding-mode control

SQP Sequential-quadratic-programming

SQUAD Spherical-quadrangle-interpolation

SRP Solar radiation pressure

Notation

$a,\ A$	Scalar quantity
$\boldsymbol{a}^B_{i/j}$	Vector quantity of i relative to j, measured in frame B
$\hat{\boldsymbol{a}}$	Unit vector
$\mathbf{A}$	Matrix
$\boldsymbol{a}^\top$	Transpose of vector $\boldsymbol{a}$
$\boldsymbol{a} \times \boldsymbol{b}$	Cross product of vectors $\boldsymbol{a}$ and $\boldsymbol{b}$
$[\boldsymbol{a}]_\times$	Skew-symmetric cross-product matrix
$\frac{d\boldsymbol{a}}{dt},\ \dot{\boldsymbol{a}}$	Time derivative of vector $\boldsymbol{a}$
$\frac{d^\nu \boldsymbol{a}}{dt},\ \overset{(\nu)}{\boldsymbol{a}},\ \dot{a}^{(\nu)}$	ν^{th}-order time derivative
$\nabla_{\boldsymbol{a}} f$	Gradient vector of f in direction of vector $\boldsymbol{a}$
$\mathbf{J}_{\boldsymbol{b}}(\boldsymbol{a})$	Jacobian matrix of vector $\boldsymbol{a}$ w.r.t vector $\boldsymbol{b}$
$\mathbf{H}_{\boldsymbol{b}}(a)$	Hessian matrix of scalar a w.r.t vector $\boldsymbol{b}$
$\mathbf{C}^{B/I}$	Direction cosine matrix (DCM) rotating a vector from frame I to frame B
$\mathbf{A}^{-1}$	Inverse of matrix $\mathbf{A}$
$\mathbf{A}^{+}$	Moore Penrose (MP) pseudoinverse of matrix $\mathbf{A}$
$\boldsymbol{q}^{B/I}$	Quaternion rotating a vector from frame I to frame B
$\Delta\square$	Perturbation
$\delta\square$	Error

Vectors

$\boldsymbol{a}$	Acceleration
$\boldsymbol{b}_{mag}$	Magnetic field
$\boldsymbol{f}$	Force
$\boldsymbol{h}$	Angular momentum
$\hat{\boldsymbol{n}}$	Normal vector
$\boldsymbol{q}$	Quaternion
$\boldsymbol{r}$	Position
$\boldsymbol{u}$	Inputs/controls
$\boldsymbol{v}$	Velocity
$\boldsymbol{x}$	States
$\boldsymbol{y}$	Outputs
$\boldsymbol{\eta}$	Modal coordinates
$\boldsymbol{\lambda}, \boldsymbol{\sigma}$	Lagrangian multipliers
$\hat{\boldsymbol{v}}$	Null space
$\boldsymbol{\omega}$	Angular velocity, body rates
$\boldsymbol{\phi}$	Modal shapes

$\boldsymbol{\tau}$	Torque
$\boldsymbol{\theta}$	Euler angles

Matrices

$\mathbf{A}$	System matrix
$\mathbf{A}_w$	Reaction wheel distribution matrix
$\mathbf{C}$	Direction cosine matrix
$\mathbf{D}$	Damping matrix
$\mathbf{H}$	Hessian
$\mathbf{I}$	Inertia
$\mathbf{J}$	Jacobian
$\mathbf{K}$	Stiffness matrix
$\mathbf{K}_D$	Derivative gain matrix
$\mathbf{M}$	Mass matrix
$\mathbf{R}$	Active rotation matrix
$\boldsymbol{\Gamma}$	Rigid-flexible coupling matrix
$\boldsymbol{\Lambda}$	Finite-element transformation matrix
$\boldsymbol{\Sigma}$	Covariance matrix
$\boldsymbol{\Omega}$, $\boldsymbol{\Xi}$	Quaternion kinematic matrices

Scalars

A	Area
c	Damping constant / Optimisation constraint
d	Distance
EA	Axial stiffness
EI	Bending stiffness
i_a	Current
k	Spring constant
k_p	Proportional gain
l	Position along length of appendage
L	Length
$\mathcal{L}$	Lagrangian
m	Mass
n	Newton-Raphson iteration
$N_{\square}$	Number of states (N_x), inputs (N_u), outputs (N_y) etc.
p_{sol}	Solar radiation pressure
P	Power

R	Radius
R_{cl}	Resistance of current loop
s	Cost
T	Discretisation step size
T_w	Reaction-wheel time constant
t	Time
Δt	Integration step size
$\mathcal{T}$	Kinetic energy
$\mathcal{V}$	Potential energy
w	Appendage deflection
η	Single modal coordinate
η_s	Reflectivity
η_e	Solar panel efficiency
η_A	Fraction of surface area covered with solar panels
λ	Eigenvalue
ν	Differential order
ν_w	Poisson's ratio of reaction wheel
ϕ	Single modal shape
ϕ_{mag}, ϕ_{sun}	Angle to magnetic field, angle to sun
ψ	Surface deformation angle
ρ	Density
σ	Stress
θ_x, θ_y, θ_z	Euler angles around body x, y, z axes
ζ	Damping ratio
ζ_{cl}	Resistivity of current loop

Subscripts and superscripts

$\square_0$	Initial
$\square_a$	Actuator
$\square_{com}$	Centre of mass
$\square_d$	Desired
$\square_f$	Final
$\square_{hub}$	Main satellite body hub
$\square_{lss}$	Large space structure
$\square_{mag}$	Magnetic
$\square_{max}$	Maximum
$\square_{min}$	Minimum

$\square_s$	Solar
$\square_{struct}$	Structure
$\square_{tol}$	Tolerance
$\square_{tot}$	Total
$\square_w$	Reaction wheel
$\square_\zeta$	Damped
$\square_x$, $\square_y$, $\square_z$	Cartesian axes
$\square^*$	Optimal
$\square^B$, $\square_B$	Body frame
$\square^I$, $\square_I$	Inertial frame

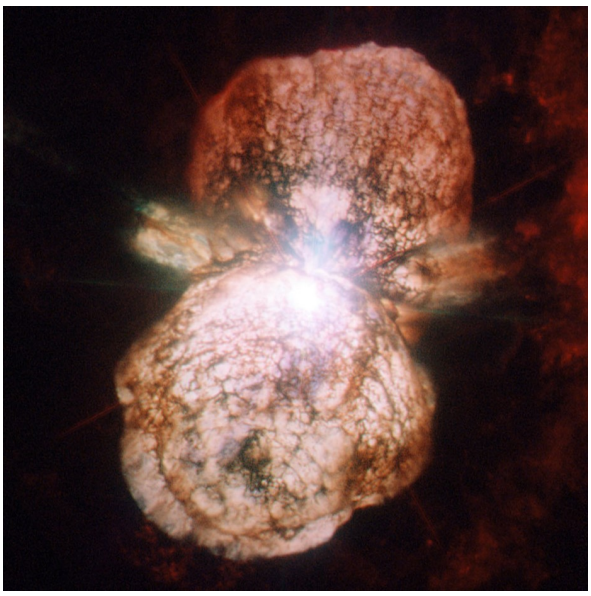
Sets

$\mathbb{R}$	Real numbers
$\mathbb{C}$	Complex numbers
$\mathbb{Z}$	Integer numbers
$\mathcal{A}_a$	Point masses consituting actuator "a"
$\mathcal{B}$	Point masses at each end of current loop segment
$\mathcal{S}$	Point masses constituting the reflective surface
$\mathcal{E}$	Equality constraints
$\mathcal{I}$	Inequality constraints

More details on vector notation, matrix definitions, and reference frames can be found in Appendix A.

Chapter 1 | Introduction

Carl Sagan speaks of humanity's desire to reach for the stars as a primordial *craving* that was *crafted by natural selection as an essential element in our survival*. Humanity's urge to explore and discover new worlds acts as an insurance policy against unpredictable *catastrophic events*. If all of humanity had been happy to sit still, then our species would only ever have been one small volcanic eruption away from extinction. This urge to be *wanderers* is also perhaps a desire for understanding our place in the universe. Given its unimaginable size, it is easy for each of us to feel somewhat insignificant. Reassurance in the modern era comes from Neil deGrasse Tyson, who posted on the social media platform *X*: *Ever look up at night and feel small? Don't. Instead feel large. Atoms in our bodies trace to the remnants of exploded stars. We are Stardust. We are alive in the universe. And the universe is alive within us* (Fig. 1.1a).



(a) We are stardust: A dual star system including the unstable Eta Carinae set to explode in a supernova in the astronomical near future

(b) The overview effect: View of the Earth rising above the lunar horizon during Apollo 10, 1969

Figure 1.1: Humanity's urge to reach for the stars (Image courtesy of NASA/ESA)

As our ancestors built sailing ships and rowing boats to explore the surface of Earth, we now build spaceships to explore the stars above it. Yuri Gagarin, strapping himself aboard one of these spaceships, became the first person to orbit the Earth in 1961; just 8 years later, in 1969, Neil Armstrong became the first person to set foot on another celestial body when he landed on the Moon. These feats of human spaceflight touch the hearts and minds of people across the globe, filling us all with an empowering sense of collective achievement. Many of those who have the opportunity to look back down at Earth from space describe a cognitive shift known as the overview effect, feeling a sense of transcendence, and a closer connection to Earth and all peoples on it [1]. Edgar Mitchell, Apollo 14 astronaut, describes standing on the Moon, looking back at Earth and developing an *instant global consciousness* and *an intense dissatisfaction with the state of the world* (Fig. 1.1b). He describes wanting to *grab a politician by the scruff of the neck and drag him a quarter of a million miles out* to look down upon Earth and see how *petty* all the world's problems really are.

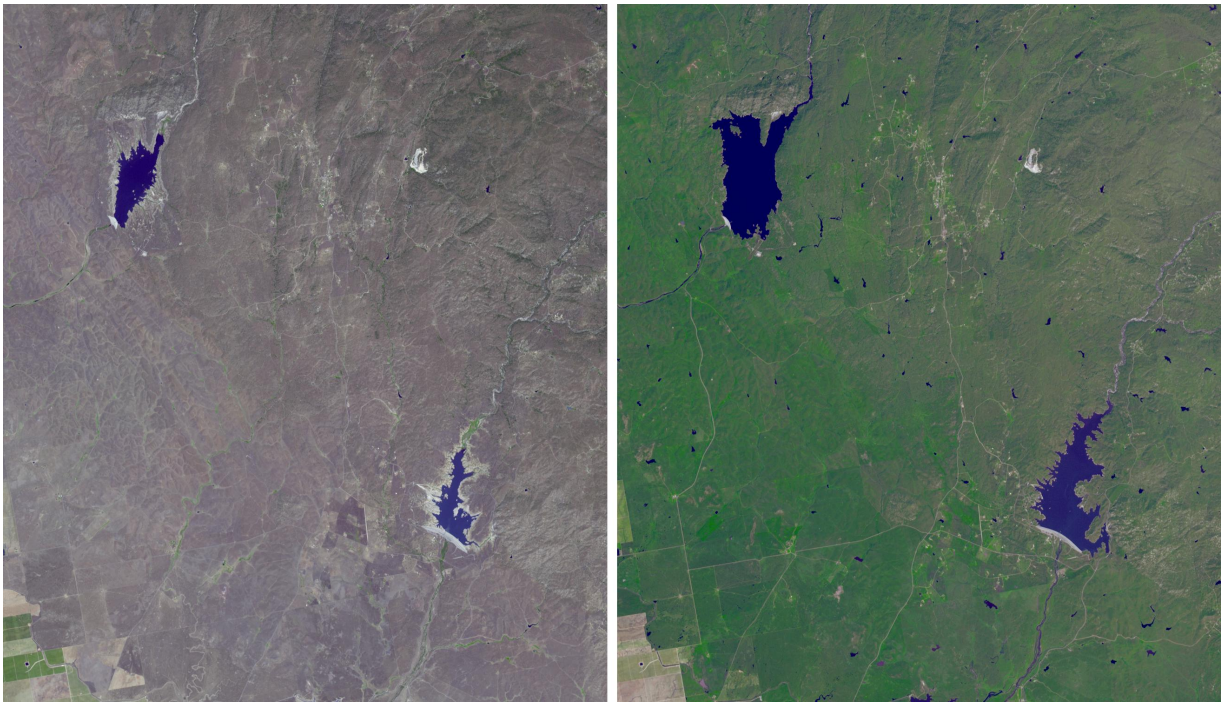


Figure 1.2: Tracking of the California drought in 2014 captured by the Terra Earth observation satellite (Image courtesy of NASA/GSFC/METI/ERSDAC/JAROS, and U.S./Japan ASTER Science Team)

Humans are not the only eyes in space looking down on Earth; thousands of satellites now orbit the planet continuously. At first, only countries such as the United States and the Soviet Union could afford to launch satellites into orbit, a side-project to the more important development of intercontinental ballistic missiles. Now the space sector is a free marketplace, with launch providers and satellite ownership no longer limited to nation-states. Morgan Stanley estimates that the global space economy will continue

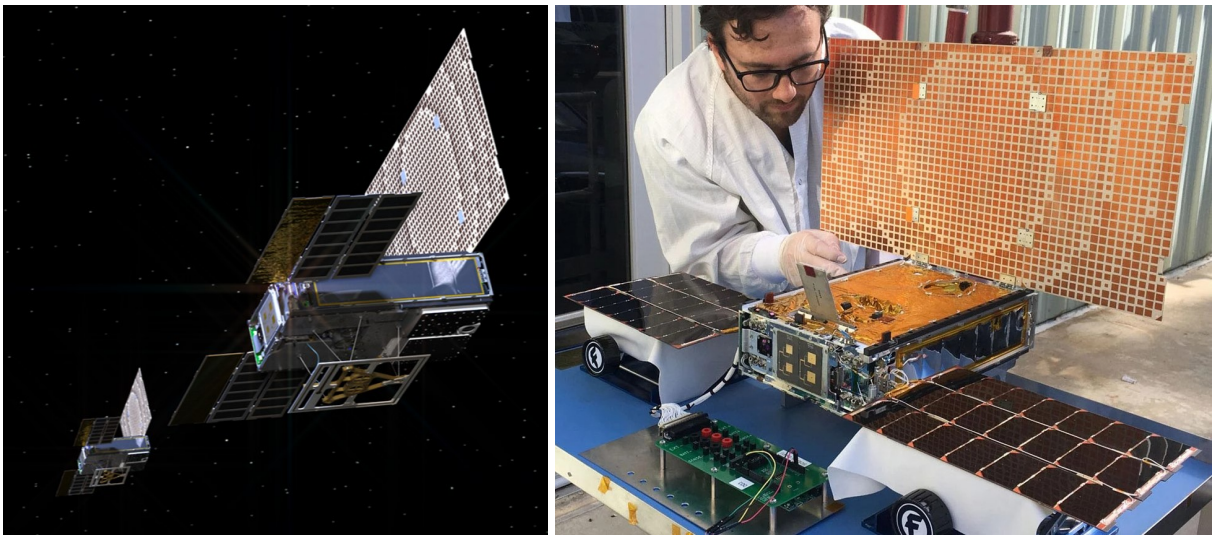
to grow to $1 trillion by 2040 [2], providing satellite telecommunications, Earth observation, and launch services [3]. Earth observation satellites, in particular, will have an important part to play in securing the future sustainability of our planet. Satellites will provide data essential to track and estimate the effects of climate change [4], detect the ever-increasing number of wildfires caused by rising global temperatures [5], warn of droughts [6], and help to improve crop efficiency to ensure food global security [7]. Fig. 1.2 shows images captured by the Terra Earth observation satellite, tracking the effects of the California drought of 2014.

To perform their roles adequately, such satellites must ensure sufficiently accurate pointing of their onboard instrumentation. Engineers achieve this through the design, testing and implementation of key attitude determination and control system (ADCS). This involves the use of sensors such as gyroscopes, accelerometers, and star trackers to measure and determine the current orientation, or attitude, of the satellite. Once this is known, commands can be sent to actuators such as reaction wheels, magnetic torquers, or thrusters to reorientate the satellite and point its instrumentation in the right direction. The attitude control subsystem determines what commands should be sent to each actuator to ensure stability and accurate pointing of the satellite and its instrumentation. The design of attitude control systems, and their underlying algorithms, can consequently be considered mission-critical. This thesis is therefore concerned with the design of attitude control algorithms for current and near-future satellites.

1.1 | Motivation

The two main constraints for access to space have been, and in many ways continue to be, mass and form factor. A satellite being launched into orbit must be sufficiently compact and lightweight to be launched within the payload fairing of a launch vehicle. Legacy launch vehicles, such as NASA's now-discontinued Space Shuttle and the European Ariane 5, cost upwards of $61,700 and $13,100 respectively to place 1 kilogram of payload into low-Earth orbit (LEO) (USD as of 2018)[8].

One solution to minimise the cost of access to space is to make use of commercially available off-the-shelf microelectronics to produce small, cheap, and lightweight *nanosatellites* ranging between 1-10 kg in mass [8]. A specific type of nanosatellite, known as a *CubeSat*, is comprised of $10\times10\times10$ cm^3 units (U); having a mass of around 1.5 kg and conforming to a standardised form factor has allowed for CubeSats to *share a ride* to space with larger spacecraft or other CubeSats, further reducing launch costs.



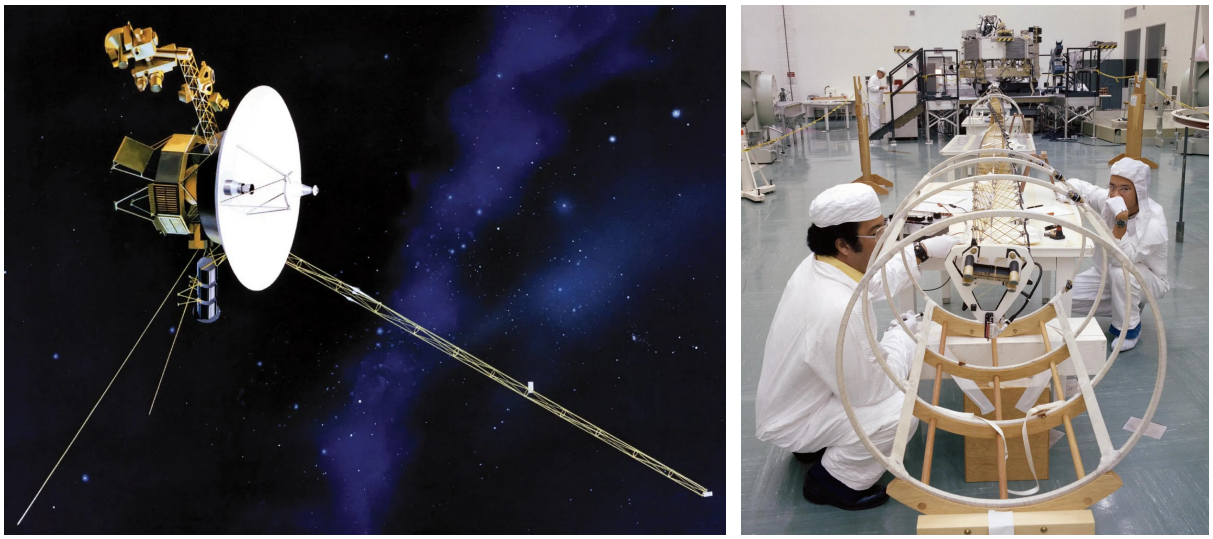
(a) Artistic impression **(b)** JPL engineer inspecting one of the CubeSats

Figure 1.3: Mars Cube One (MarCO) with deployed solar panels and antenna [9] (Images courtesy of NASA/JPL-Caltech)

1.1.1 | Deployable Appendages

While CubeSats were originally developed for educational purposes [10], they are now capable of performing complex scientific [11] and Earth observation [12] missions, which had historically been reserved for larger satellite platforms. To facilitate these more complex roles, and maintain their strict standardised form factor, it is common practice to make use of deployable appendages [13] to support sensitive scientific instrumentation [14, 15], larger antennas for improved connectivity [9, 16], and additional solar arrays to meet higher power requirements [17, 18]. Mars Cube One (MarCO), a pair of two 6U CubeSats as seen in Fig. 1.3, became the first CubeSats to operate beyond Earth orbit and helped to relay communications between the Insight Mars lander and Earth. The complex role of MarCO required the use of deployable appendages, including a high-gain reflectarray antenna [9] for communication with Earth, an ultra-high-frequency (UHF) antenna to communicate with Insight, and deployable solar arrays to provide sufficient power.

Deployable appendages are also useful for increasing the capabilities of larger satellites and spacecraft. For example, the Voyager 1 & 2 spacecraft (Fig. 1.4), both launched in 1977, placed their sensitive magnetometers along deployable appendages to reduce interference with the rest of the spacecraft [19]. Robotic manipulators such as the Canadarm, utilised by the Space Shuttle and International Space Station (ISS) [20], are another form of deployable appendage which will be integral to future in-orbit assembly and manufacturing [21], and orbital debris removal [22].



(a) Artistic impression **(b)** Deployable boom

Figure 1.4: Voyager spacecraft with deployed boom [19] (Images courtesy of NASA/JPL-Caltech)

Robotic appendages for in-orbit assembly and manufacturing are one technology enabling the future development of large space structure (LSS) platforms. Additionally, the recent and continued reduction in launch costs means that mission concepts previously considered by many to be limited to science fiction could now become a scientific reality. As mentioned previously, access to LEO historically cost in the region of $10,000s per kilogram; SpaceX's partially reusable Falcon 9 and Falcon Heavy launchers (Fig. 1.5), reduce this to $2,700 and $1,400 per kilogram to LEO respectively [8] (USD as of 2018). SpaceX has also suggested that this price could get to as low as $10 per pound (≈$20 per kilogram), as reusability of all rocket stages becomes possible [8].

1.1.2 | Large Space Structures

Solar sailing, a propellant-free propulsion technology that utilises solar radiation pressure (SRP), requires LSSs to deploy its highly reflective thin sail membranes. The size of these solar sails has seen an incremental increase as the bounds of the technology are tested and demonstrated. Launched in 2010 by the Japan Aerospace Exploration Agency (JAXA), IKAROS was the world's first successful interplanetary solar sail [23]. Its 196 m^2 sail was deployed across a 20 m span using a combination of flexible structures, tip masses, and centrifugal forces. Lightsail-2 [24], launched in 2019, used a 32 m^2 sail, and more recently, the unsuccessful Near Earth Asteroid (NEA) Scout mission [25] employed an 86 m^2 solar sail. Solar Cruiser [26] was a proposed mission which would have advanced the technology further by attempting to deploy a vast 1,653 m^2 sail using novel 30 m rollable, collapsible booms. To achieve maximum acceleration, a

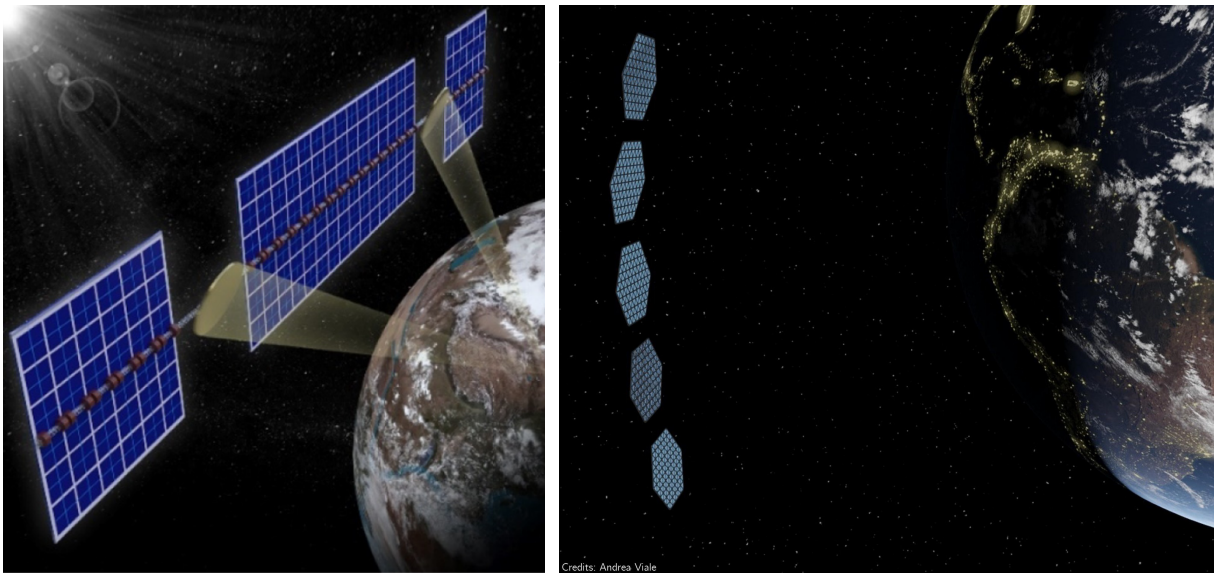
(a) Launch **(b)** Booster landing

Figure 1.5: Re-usable SpaceX Falcon Heavy launch vehicle (Images courtesy of SpaceX licensed under CC BY-NC 2.0)

solar sail must be manufactured to be as lightweight and large as possible, inevitably leading to the use of highly flexible materials in its construction.

Space-based solar power (SBSP) is a future concept for clean and sustainable energy requiring monolithically large LSSs; collected from space using large arrays of solar cells, the solar energy is beamed to ground stations through microwave wireless transmission. The European Space Agency's (ESA) SOLARIS initiative and the UK Space Energy Initiative (SEI) aim to have in-orbit demonstrations by 2030 with commercial SBSP delivering power to the grid by 2040. The idea of SBSP was first suggested by Dr Peter Glaser in 1968 [27] with the first substantive concept being proposed in 1979 by the US Department of Energy and NASA [28]. The 1979 reference architecture consisted of a 10 km x 5 km LSS supporting the solar cells, with an additional 1 km diameter microwave transmitter. This would beam 5-10 GW of continuous energy from geostationary orbit (GEO) to a 10 km x 13 km rectifying antenna on Earth. More recent concepts include SunTower [29], the European Sail Tower [30], SSPS-OMEGA [31], and the Free-Drift Solar Power Satellite concept [32].

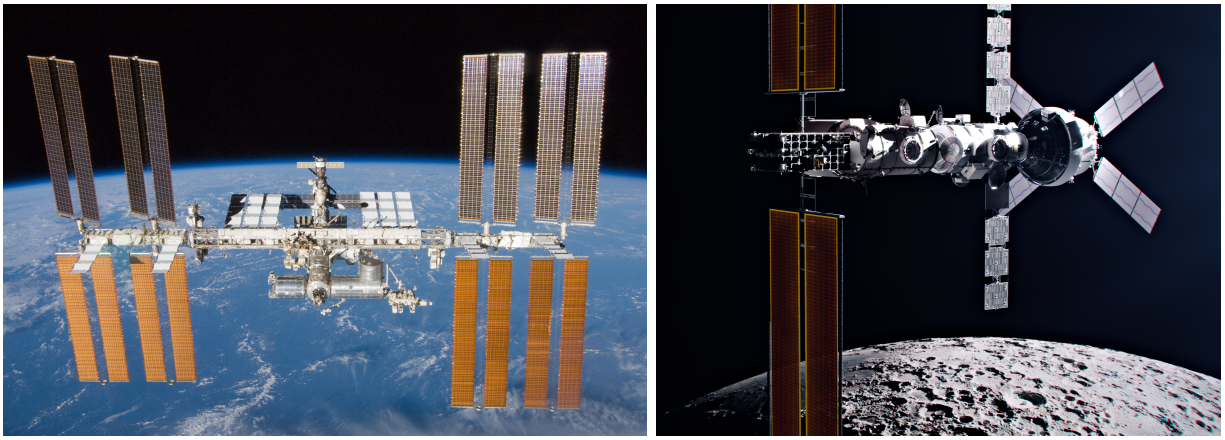
An alternative proposal to SBSP is the use of orbital solar reflectors to reflect sunlight directly to solar power farms on the Earth's surface. This reduces the complexity of the space segment of the concept, leaving the solar cells and power conversion infrastructure to the ground-based segment. A study in the 1970s by NASA [34] considered using up to 54 reflectors in a constellation, allowing for solar rays to be relayed around the Earth, reaching any required ground station. These reflectors would be very large,



(a) Free-Drift SBSP satellite [32] **(b)** SOLSPACE orbital solar reflectors [33]

Figure 1.6: Example of LSS applications: Solar power from space concepts

ranging from 10s to 100s km in diameter. Another approach looks at using clusters of smaller 1 km diameter reflectors in a 10 km diameter array [35]. It is envisaged that 18 such clusters could be deployed evenly around a 1,000 km sun-synchronous orbit to provide solar energy to 40 ground solar power stations at dawn and dusk. A similar idea using sun-synchronous orbits has recently been studied by researchers at the University of Glasgow as part of their ongoing SOLSPACE project (Fig. 1.6b)[33]. Their approach makes use of five small 250 m hexagonal reflectors flying in formation to provide a total reflective area equivalent to that of a 1 km circular reflector.



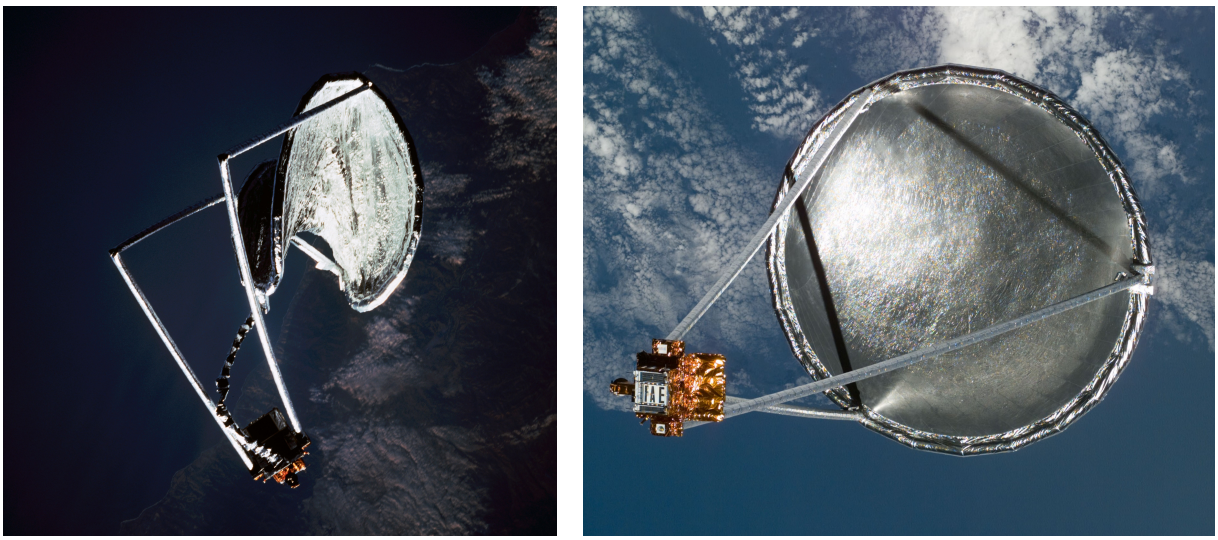
(a) International Space Station (ISS) **(b)** Lunar Gateway concept

Figure 1.7: Example of LSS applications: Space stations (Images courtesy of NASA/JSC)

Large space radio telescopes and antennas for science and communication are another possible application of LSSs. In many ways, these share a close resemblance to orbital solar reflectors, with the main distinction being the range of wavelengths of light being

detected or transmitted. In 1967, a feasibility study for a 1.5 km diameter, 1,300 kg radio telescope was carried out by Astro Research Corporation on behalf of NASA [36]. The telescope would use aluminium ribbons to form a parabolic reflector with a central three-longetron column that could be folded before deployment.

Space stations, such as the International Space Station (Fig. 1.7a)[37], are another example of LSSs and have been an important aspect of human spaceflight. The next generation of space stations, such as the Lunar Gateway project (Fig. 1.7b)[38], will attempt to establish human habitation beyond low Earth orbit.

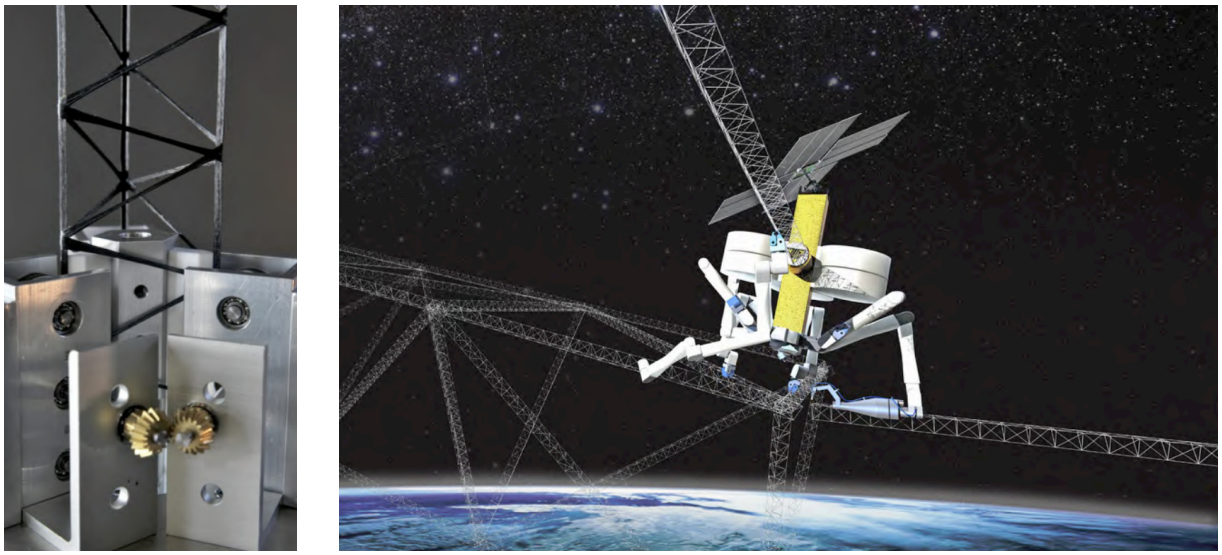


(a) During deployment **(b)** Deployed

Figure 1.8: In-Step Inflatable Antenna Experiment (IAE) (Images courtesy of NASA/JSC)

As highlighted, there are many applications for LSSs, varying in length-scale from tens-of-metres to kilometres. Several existing and emerging technologies can facilitate the construction of such LSSs. Deployable structures, similar to the appendages discussed previously, could allow smaller-scale LSSs to be collapsed into form factors conducive to a single launch. This can be achieved through the use of long telescopic and coilable booms, shape memory composites, and articulation [39]. Additionally, entire truss structures can be deconstructed and reconstructed using combinations of joints and cables [39, 40]. LSSs could also be formed using inflatable structures and booms, as was successfully demonstrated by the In-Step Inflatable Antenna Experiment (IAE), launched in 1996 (Fig. 1.8)[41]. These inflatable structures have a more straightforward deployment mechanism compared to the use of physical joints [42]. This reduces the risk of a failed deployment and avoids the additional mass needed to facilitate the articulated joints.

For larger LSSs, these approaches will still result in form factors and masses inconducive to a single launch. Instead, the structure must be divided into sub-structures,



(a) Trusselator^{TM} prototype extruding a first order truss

(b) SpiderFab^{TM} constructing second order truss structure from Trusselator^{TM} first order truss

Figure 1.9: Tether Unlimited Inc's orbital manufacturing technologies (Images courtesy of NASA/NTRS [43])

which could be launched and then assembled in space. A common example of such a strategy is the ISS, which started construction in the 1990s [37] and is still in operation as of 2024. A more recent example of an in-orbit assembly concept is *TriTruss* [44], a modular truss system that can facilitate the assembly of LSSs for various applications. Additionally, a review paper by Xue et al. [45] outlines several past, present, and future applications of in-space assembly technologies for the construction of LSSs. Many of these make use of robotics to construct LSSs from combinations of smaller sub-structures. One such robotic assembly system is the *SpiderFab^{TM}* system, developed by Tether Unlimited Inc. (Fig. 1.9)[43, 46]. This system can be coupled with another of Tether Unlimited Inc.'s technologies known as *Trusselator^{TM}* [47, 48], which is capable of producing long truss structures through extrusion additive manufacturing techniques. SpiderFab^{TM} can be used to connect several of these extruded truss structures into a larger, second-order truss structure capable of facilitating LSSs [43]. While Trusselator^{TM} utilises an extrusion additive manufacturing approach, other methods exist, such as powder and liquid-based systems [49], or solid foams [50]. Another review on in-space assembly using robotic systems by Li et al. [21] highlights that the future direction of in-orbit assembly research, will see a move towards swarms of assembly robots collaborating to perform complex construction tasks.

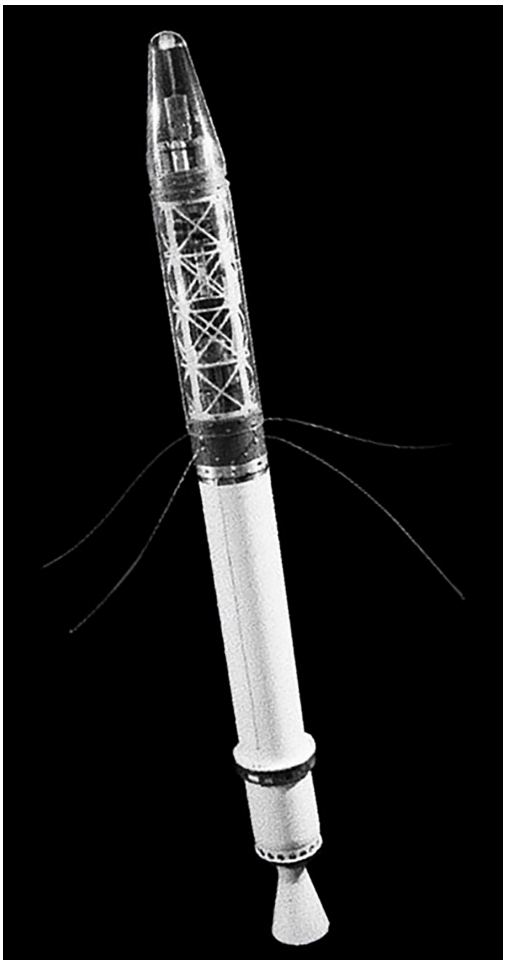
1.1.3 | Flexible Dynamics

As mentioned previously, the mass and form factor of a satellite's appendage or LSS are still limiting factors; these structures will consist of lightweight, and therefore inevitably, flexible materials. Considering the flexible dynamics of these space structures is important, as was first noticed during the Explorer I (Fig. 1.10a) and Alouette I (Fig. 1.10b) missions of the 50s and 60s [51].

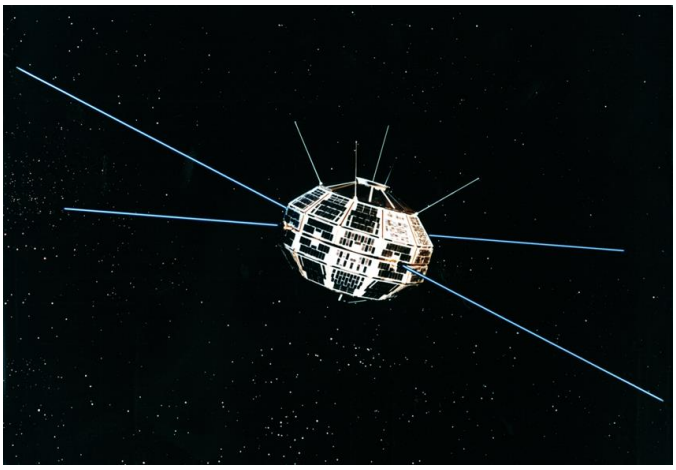
After its launch in 1958, Explorer I's attitude was to be stabilised by spinning about its minimum principal axis of inertia. However, unexpectedly, the satellite's spin-axis began precessing, eventually causing the satellite to spin end over end. The culprit for this instability was mechanical damping in the 4 flexible antennas attached to Explorer I's rigid main body, causing energy to be lost from the overall dynamic system. This caused the satellite to move from its initial high-energy state, spinning about the minimum inertia principle axis, to its minimum-energy state, spinning about the maximum inertia principle axis [51–53].

Another phenomenon was discovered in 1962 when Canada's first satellite, Alouette I, was launched. It was equipped with flexible antennas, similar to Explorer I, which were found to deform when exposed to solar thermal radiation. This deformation of the antennas did not occur uniformly, and therefore the eventual asymmetry in the satellite's overall shape, when exposed to SRP forces, caused a net external torque on the spacecraft. Several properties of the spacecraft, such as the general shape and frequency of the antenna's deformation, would determine what effect this torque had. In the case of Alouette I, the resultant torque opposed the spin induced on the satellite that was meant to provide stability. Therefore, Alouette I began to *de-spin* and after 3 years in orbit, it came to a complete rest [51, 52, 54].

The Hubble Space Telescope (Fig. 1.10c), launched in 1990, suffered its fair share of issues. Most famously, the mirror was found to have an aberration [55], resulting in images not appearing as sharp as expected. Additionally, it was found that excessive jittering of the telescope's line-of-sight could degrade image quality [56]. Contributions to this jittering included vibrations caused by the running of the onboard tape recorder, and the primary mirror vibrating due to thermal shocks experienced as the telescope passed in and out of eclipses. However, a major contributor to the jitter was the disturbances induced by the flexible solar panel appendages [56]. The problem was mitigated in part through the inclusion of a 6^{th} order double notch filter in series with the existing attitude controller [57]. The notch filter was designed to avoid excitation of the flexible modes of the solar panel appendages, attenuating the disturbances contributing to the line-of-sight jitter. Eventually, in 2002, Servicing Mission 3B (SM-3B) [58] would



(a) Explorer I (Image courtesy of NASA licensed under CC BY-SA 2.0)



(b) Alouette I (Image courtesy of CSA)

(c) Hubble Space Telescope with flexible solar panels before Service Mission 3B (Image courtesy of NASA/ESA)

Figure 1.10: Spacecraft that experienced flexible dynamic phenomena

replace the flexible solar panel appendages with more rigid ones. Additional dampers were added to ensure that pointing stability margin requirements were met [57].

It has therefore been demonstrated that current and future satellites, both big (LSSs) and small (CubeSats), will involve the use of flexible structures in their construction. Due to the problematic dynamic behaviours associated with these flexible structures, current and future attitude control algorithms must consider them at the heart of their design to ensure stability and accurate pointing of satellite platforms. Additionally, the satellite itself must be designed with a focus on the interaction between the flexible dynamics and the attitude control system to mitigate any problematic effects.

1.2 | Literature Review

Before developing new approaches to attitude control and design of satellites with flexible structures, the historical and current state-of-the-art literature in the field must be

reviewed. Additional literature is then included on a prospective new approach to the problem, informing the direction that the rest of this thesis will take.

1.2.1 Mathematical Modelling

To develop suitable attitude control algorithms, it is first important to have a detailed understanding of the full dynamics of the satellite, which can be achieved through appropriate mathematical modelling. Mathematical models can also be helpful during the design process of a satellite, ensuring that its structure and actuators meet overall system requirements.

Mathematical modelling of a dynamic system can be defined as the process of formulating a set of ordinary differential equations (ODEs) that describe the rate of change of the system's states, also known as a *state space* model [59]. These ODEs can be numerically integrated to determine the behaviour of the system dynamics over time; a process that can be referred to as *forward* simulation.

Mathematical modelling of satellites with flexible structures can be achieved through several approaches, including considering the satellite as a rigid body, assemblage of rigid bodies, or as a combination of rigid and elastic bodies [51]. Considering the satellite as just one single rigid body may be useful for preliminary studies; however, eventually, consideration of the flexible dynamics, at least to some extent, will be required. A good example of this can be seen in the SOLSPACE project, where the orbital solar reflectors were initially assumed to act as rigid bodies [33], with a later study finding that the flexible dynamics could introduce sufficient noise into the system that additional torques from the attitude control actuators were required [60].

Discrete Coordinate Approach

Considering the satellite as an assembly of rigid bodies, often referred to as the *discrete coordinate* approach [61], can prove versatile, as any satellite configuration can be represented if enough bodies are used. More flexible components of the satellite will, however, require a large number of these interconnected bodies to correctly capture the flexible dynamics, which can lead to the approach becoming computationally limited. A simple discrete-coordinate, first-order representation of a flexible appendage attached to a main satellite body can be achieved by considering both as individual rigid bodies attached by a *torsional* spring-damper hinge [62, 63]. This provides a low-fidelity representation of the dynamics, with the benefit of faster computational time.

In many cases, a higher fidelity representation of the dynamics and capturing of higher frequency modes is required, which can be achieved through a discrete-coordinate

lumped-parameter approach [64, 65]. This simply takes the first-order hinged representation further by adding additional torsional spring-damper hinges connecting a chain of point masses representing each flexible appendage. The lumped-parameter approach can also be extended to LSSs [66, 67], by representing the whole spacecraft as a lattice of point masses connected by *linear* spring dampers. Since the model consists of only point masses, the effects of the environment and actuators on the LSS need to be decomposed into a series of discrete forces which can be applied to each point mass. The lumped-parameter approach, therefore, facilitates modular functionality, in that dynamic effects and actuators can be added and removed from the model simply by adding or removing their associated forces from the associated point masses; particularly useful during the design process of an LSS. Both lumped-parameter approaches, and the first-order hinged appendage approach, model the stiffness of their flexible structures through the appropriate choice of the spring constants of the torsional and linear spring-dampers.

Hybrid-coordinate Approach

An alternative to the discrete-coordinate approach is the use of *hybrid-coordinates*, whereby the rigid and flexible dynamics of the satellite or LSSs are separated [51, 68]. Discrete coordinates are still used to describe the larger rigid-body attitude dynamics, while modal coordinates are used for the smaller deformations of the satellite appendage or LSS. Coupling between these two sets of coordinates ensures that the effect of the flexible structures on the attitude dynamics is captured. The discrete rigid-body dynamics of a satellite are well-known [69, p.200], however, the flexible modes need to be determined for the specific satellite or LSS configuration.

One approach to represent the flexible modal coordinates, known as the *assumed-modes* approach, is to estimate the characteristics of the flexible dynamics ahead of time; simplifying the equations of motion and minimising computational complexity when solving the coupled dynamics [70]. If the flexible appendages or LSS can be represented as a single element, such as a simple beam [71] or thin plate [72], then the assumed modes can be derived from analytical solutions [73]. However, if the structure is more complex and requires the inclusion of several different types of elements, then some form of discretisation method is required to reform the associated partial differential equations into ODEs that can be numerically integrated through time. The lumped-parameter method described previously is one possible approach, however, more commonly the finite element method (FEM), popular throughout engineering, is utilised. FEM can be used to determine the modal characteristics, i.e. *shapes* and *frequencies*, for each assumed mode which can be then used solve the coupled rigid-flexible dynamics [72, 74]. Alternatively, in some cases, it may be chosen to use the full finite element model [71,

75] to represent the flexible dynamics within the hybrid-coordinate system. The latter approach can be more computationally intensive, but may be a more attractive option to avoid truncating some of the higher-order flexible modes.

Software

Numerical simulation of a satellite with flexible structures using one of the aforementioned approaches could be performed using a variety of software packages. Sometimes a lower-level approach may be adopted, whereby the derived ODEs are directly implemented into code and numerically integrated using packages such as *MATLAB*'s ODE-suite [76] or the `odeint()` function available in the *Python SciPy* package [77]. Alternatively, higher-level approaches can be adopted using more sophisticated software packages such as the *Simspace Multibody* package for *MATLAB's Simulink* block diagram simulation environment. *Simscape Multibody* includes blocks which implement both the lumped-parameter and FEM methods to describe the flexible dynamics of beams, plates and solids [78]. Suitable combinations of these flexible bodies would allow for a representation of a satellite with deployable flexible structures to be achieved. Implementation within the *Simulink* environment also allows for easy integration with many of *MATLAB/Simulink*'s other software products, already widely used within aerospace and engineering industries. An open-source modular astrodynamics simulation software package known as *Basilisk* has also been developed by academics at the University of Colorado [79]. The user interacts with the software package through *Python*, configuring modules and executing simulations. One developed module allows for hinged rigid bodies to be attached to the main satellite hub, which can be used as a low-order, lumped-parameter approximation of the flexible structural dynamics [80].

1.2.2 Attitude Control Actuators

The actuators used to impart torques on the satellite body are an important component of the overall attitude control system to consider. Traditionally, these have included those dependent on the exchange of angular momentum (reaction wheels and control moment gyros), those that interact with the Earth's magnetic field (magnetic torquers), and thrusters (cold gas, monopropellant etc)[81]. For satellites with attached flexible appendages, every type of actuator will be best placed centrally within the rigid main hub, as the lightweight flexible appendages will be unable to accommodate the additional mass without significant deformations. In the case of larger, uniformly flexible LSSs, the choice of actuator placement and type is far less trivial.

Large Space Structures (LSSs)

An investigation by Viale and McInnes [82] looked at several actuator types for an orbital solar reflector LSS including reaction wheels [83], control moment gyros [60, 74], distributed magnetic control [66, 84], and SRP torques [33]. The maximum size of LSS that can be actuated by each actuator type was determined, and actuator mass scaling laws were found. Two of the considered actuators make use of the significant SRP forces imparted on the orbital solar reflector's surface. Both approaches attempt to produce an offset between the centre of mass and centre of pressure, which will in turn produce a useful torque that can be used to control the LSS. The former can be achieved through the use of a moving mass that can shift the overall centre of mass of [85, 86]. Alternatively, reflectivity control devices (RCDs) can be used to vary the reflectivity of different areas of the reflector, moving the overall centre of pressure and producing a torque [87–90]. This was successfully utilised by JAXA's IKAROS solar sail [23, 87, 88] to minimise fuel consumption of the attitude control thrusters. RCDs have also been proposed for use by solar sail spacecraft for orbital [91, 92] and coupled orbital/attitude control [93].

Viale and McInnes, while considering the scaling and ability of each actuator to provide sufficient torque, do not consider the effects of the structural flexibility of such LSSs. Additionally, they only consider each actuator type separately and not whether there would be any benefit to combining more than one actuator type. Work by Robb et al. [66, 84] consider the use of magnetic torques embedded within the LSS to provide a distributed torque across the whole LSS. It is shown that this has advantages over a larger central torque, in terms of the deformation of the structure during attitude slew manoeuvres. While the structural deformation of the LSS has been considered in the work by Robb et al., this is limited to magnetic torques only.

1.2.3 Attitude Control Methods

There is a wide variety of control methodologies applied to the attitude control of satellites with flexible appendages; some are more complex than others. One simple method to minimise the effect of appendage and LSS flexible modes is to try and avoid exciting them as much as possible; achieved by the Hubble space telescope through the use of notch filters (Section 1.1.3). For large-angle reorientation slew manoeuvres, this can be achieved through careful selection of the attitude and body rate references to minimise *jerk* (the derivative of acceleration). The time-optimal solution to a large-angle slew is well understood to be the bang-bang manoeuvre [94], a step-wise acceleration profile with theoretically infinite jerk. Modification of the bang-bang manoeuvre through the use of *versine* functions [64, 95] allows for the jerk to be minimised and has been shown

to reduce excitation of flexible modes and minimise disturbances on pointing accuracy. These *versine* functions can be tuned to give the desired trade-off between minimising jerk, the time taken to complete the manoeuvre, and the maximum acceleration required by the actuators. Alternatively, a polynomial-based reference can also produce a low jerk profile [96], at the expense of increasing the manoeuvre time or maximum acceleration of the manoeuvre.

The attitude control algorithms used to actuate a satellite with flexible structures are, for the most part, not too different from those employed when the satellite is assumed to be rigid. Nonlinear control methods tend to be preferable, as the Euler rotational equations describing the rigid body motion of a satellite are nonlinear and coupled [69, p.200].

Inverse Control

Given a mathematical model of the governing ODEs describing the dynamic behaviour of the system for some given input, a suitable control strategy would simply be to find a solution to the system's *inverse dynamics*. This can be achieved through an analytical approach called nonlinear dynamic inversion (NDI), also commonly known as input-output feedback-linearisation [97, p.216]. The approach gains the latter description as the process of solving the inverse dynamics of the nonlinear system allows for the nonlinear system to be transformed into a linear (or partially linear) one. This would then allow for linear control methodologies and theory to be applied to stabilise the transformed system.

Such an approach is adopted by McInnes [98] who solves the inverse rigid body dynamics of a satellite using NDI, producing smooth, continuous, polynomial slew manoeuvres to avoid exciting the flexible modes of the satellite's structure. The inverse control solution is then applied to provide feedback linearisation, which is coupled with linear control terms to help deal with inertia uncertainty and actuator degradation. Singh [99] applies NDI to develop an inverse controller which decouples the rigid body and flexible modes of the satellite. This allows for an additional flexible mode stabilisation controller to dampen the appendage oscillations using velocity feedback provided by sensors attached to the flexible boom appendage. Another approach by Weerdt et al. [100] performs feedback-linearisation for a satellite using NDI, with unknown internal disturbances due to fuel sloshing compensated using an adaptive neural network.

Sliding Mode Control

Sliding-mode control (SMC), or *variable structure* control as it is also known, is a common approach for robust control of nonlinear systems [97]. SMC establishes a *sliding surface*, or hyperplane, which in the case of attitude tracking lies within the space defined by the attitude and body rate errors [81, p.295]. This reframes the tracking problem to finding the *equivalent* control action that drives the system dynamics onto this surface. The equivalent control can be found analytically through techniques such as NDI, as described previously; SMC can therefore be considered a form of feedback-linearisation. SMC, however, includes the addition of a discontinuous *switching* term, also sometimes referred to as the hitting term, ensuring robustness to uncertainties, modelling errors, and disturbances; something to which feedback-linearisation alone is more susceptible. If the system's dynamics start to deviate from the defined sliding surface, then the switching term will force it back to the surface.

The switching term can, however, lead to *chattering* in the control signal as the system passes from one side of the sliding surface to the other in quick succession. It can be easily understood that the resultant high-frequency, discontinuous chattering in the control signal could excite the flexible modes of the satellite structure; implementations of SMC to satellites with flexible structure must therefore take chattering into account. One common approach is to define a *boundary layer* around the sliding surface within which the control can be interpolated between the equivalent and switching terms [97]. This leads to tracking within a *guaranteed precision* rather than the *perfect* tracking achieved without a boundary layer.

Saturation functions offer one way to include a boundary layer in SMC, and have been applied in several cases to attitude control of satellites with flexible structures [71, 101]. Another approach [102] deals with chattering through the use of *fuzzy logic* rules to smooth between the equivalent and switching control terms, considering the effect of appendage flexing as uncertainties in the system dynamics. An additional adaptive fuzzy logic system is used to estimate the equivalent control term in the presence of these uncertain flexible dynamic effects. Some applications of SMC also try to deal with the flexible modes more directly. Bang et al. [71], for instance, include measurements of the reaction torque between the main rigid body and the flexible appendage in the sliding surface to help suppress vibrations. However, this assumes that the internal torque can be measured through some form of strain gauge, or can be estimated/observed indirectly through other measurements.

Another approach adopted by Hu [103] is to augment the SMC with active vibration suppression using piezoelectric smart materials, dealing with any flexible modes that

may be excited due to high-frequency chattering of the switching term. These smart materials earn their name from the dual sensing/actuating purpose that they serve. When subject to mechanical forces, they will produce electrical energy (sensing), and conversely, when electrical energy is applied, the smart materials produce mechanical work/damping (actuating)[104]. Optimal placement of these smart materials across a satellite's flexible structure can allow for sufficient observability and controllability of the flexible modes [105], ensuring that effective vibration suppression can be achieved. Excitation of the flexible structural modes during a slew manoeuvre can never be completely avoided, therefore, smart materials have become a popular addition to many proposed attitude control approaches.

Model-predictive Control

Model-predictive control (MPC) is a model-based method which produces a feedback control solution through an iterative online optimisation process [106]. An *optimal* control sequence is found over a finite horizon through numerical optimisation of a defined cost function, with predictions made by the model providing equality constraints. The cost function is usually chosen to provide a good compromise between close reference tracking, and minimisation of the control effort. The first element of the optimised control sequence is executed, and measurements of the real state of the system are made to update the MPC's predictions and inform the next optimal control solution.

Iannelli et al. [107] developed an MPC controller for a satellite with a flexible rotating payload. The authors highlight the benefit of the inclusion of input constraints in the MPC process, allowing for limitations in the reaction wheel actuators of the satellite to be accounted for. The controller used a simplified model for faster online computation of the optimal control solution. Testing of the MPC's performance using a more complex *complete* model revealed poor robustness to model uncertainty, particularly off-diagonal inertia terms, and centre of mass position. The authors suggest that the addition of a parameter identification/estimation technique could be employed to improve the overall robustness of the controller design. TayyebTaher and Esmaeilzadeh [108] implement an MPC solution using a linearised model, and a genetic algorithm to tune hyperparameters such as the control horizon, prediction horizon and sampling time. The genetic algorithm tuning optimises for rise time, settling time, overshoot, and minimisation of structural flexing. A limitation of MPC is the assumption that the full state vector can be measured or estimated, including the flexible modes.

Proportional-Integral-Derivative Control

Easy to implement and widely understood, proportional-derivative (PD) and proportional-integral-derivative (PID) based control techniques still earn their place among more complex methods for attitude control of satellites with flexible structures. These controllers simply apply control actions proportional to the error, the derivative of the error (damping), and the integral of the error (steady-state convergence). Song and Agrawal [109] utilise the same piezoelectric smart materials for vibration suppression as was used by Hu [103]; however, instead of SMC, the authors use a simpler PD control scheme. The paper is, however, limited to a demonstration of rotation about one axis; the dynamics of full three-axis rotations will include nonlinear terms which the linear PD control structure may struggle to compensate for. Liu et al. [110] develop a PD based controller with a disturbance-observer-based feedforward term to compensate for vibrations in the appendages. The proposed composite control structure is shown to reduce the effects of appendage flexing and external disturbances on the main hub, when compared to PD control only. The work carried out in this paper is, as seen previously, limited to rotation about one axis only. To produce a PD based solution for control about three axes, Baghi et al. [111] develop a nonlinear variable gain PD controller which is capable of dealing with the associated nonlinear three-axis dynamics. The PD controller's gains are scheduled using Gaussian functions of the satellite's attitude and angular velocity. It is found that the inclusion of this gain scheduling technique reduces the maximum control torque, produces a more uniform control action, decreases peak deflection of the flexible appendages, and increases robustness to external torques when compared to a simpler PD controller with static gains. Furthermore, the authors state that the advantage of such a control law, in terms of its implementation, is that it does not require any observations of disturbances, uncertainties or appendage flexing, as has been required in other cases.

Feedforward Control

An alternative, but common, approach for improved tracking performance adds a feedforward control element to an existing feedback controller [112, 113]. The feedforward term can be derived through a suitable analytical model-inversion technique such as NDI, as described previously. With a model that sufficiently describes a satellite's flexible dynamics, it is suggested that their effects on attitude tracking can be compensated for using feedforward control [114].

Other feedforward approaches instead target the flexible modes directly through an approach known as *input-shaping*. Input shaping eliminates resonant responses of the system by convoluting the reference signal with a series of impulses such that the closed-

loop system (i.e. including the feedback controller) follows the original unshaped reference, without vibrations [115]. Careful selection of the impulse magnitudes and spacing within the time domain allows for zeros to be added at the locations of the poles of the flexible modes that are to be cancelled. Tzes and Yurkovich [116] apply input shaping to produce vibration-free attitude control of a satellite with a flexible-link robotic manipulator appendage, ensuring accurate end-point positioning. The authors highlight that robustness to parameter uncertainties and modelling errors can be improved by increasing the number of impulses in the convolution sequence. The additional impulses increase the time delay in the reference signal, unsuitable for rapid manoeuvring. Instead, adaptive frequency domain identification techniques can be used to better target and cancel out the flexible dynamic modes.

Several practical experiments have been performed, proving the efficacy of input-shaping for attitude control of satellites with flexible structures [95, 117]. Romano et al. [117] perform experiments on a two-link space robot simulator, finding that input-shaping proves effective in reducing vibrations in the links during near-minimum-time manoeuvres. The authors also find that a smoothed bang-bang manoeuvre, similar to the versine-smoothed profiles mentioned previously [64, 95], performs similarly well to input shaping when only one of the two links is flexible. However, in the case where both links are flexible, only the input-shaping approach proves effective. A paper published by Singhose and Vaughan [118] compared the performance of input-shaping to digital filtering techniques, similar to those used by the Hubble Space Telescope [57]. The authors prove rigorously that the time delay of an input shaper will always be shorter than that of a low-pass or notch filter with identical vibration suppression constraints. The authors also find that input shapers are easier to design, compute, and implement than their digital filtering counterparts.

1.2.4 | Inverse Simulation

Whilst the selection of attitude control methodologies is wide and varied, each attempts to provide good tracking performance by replicating the inverse dynamics of the system, with additional terms to provide robustness and stability. This is most clearly illustrated by the inverse control approach, usually derived using an analytical model-inversion methodology such as NDI. A numerical model-inversion approach known as Inverse Simulation (InvSim) may provide a viable alternative.

InvSim can generally be considered as a collection of iterative numerical methods used to solve the inverse dynamics of a system. Traditional forward simulation maps a time-series of inputs to a timeseries of outputs by numerically integrating a set of ODEs.

InvSim techniques aim to reverse that mapping, providing a timeseries of inputs given a timeseries of required outputs, through some numerical process.

Research in the field was seen as early as 1936 when Jones inverted a simplified linear aircraft model to investigate the effects of gust disturbances [119]. There was then a significant period of inactivity, as research into InvSim was considerably limited due to a lack of computational power at the time. The invention of the first transistor in the late 1940s [120] started a revolution in computing, which by the 1980s had progressed to inexpensive personal computers, such as those produced by Apple and IBM, becoming widely available [121]. It is therefore no surprise that many of the advances in InvSim occurred during this period (1980s and 1990s).

Methods

The first InvSim approaches explored during this resurgence worked by reforming the governing ODEs into a set of simultaneous algebraic equations through numerical *time differentiation* of the system state variables [122, 123]. Discretising the desired output trajectory by taking samples at a fixed discretisation timestep allows for the simultaneous algebraic equations to be solved using the iterative Newton-Raphson (NR) method, converging upon a solution to the input and state that satisfies the ODEs and sampled desired output. The process is then repeated at each discretisation timestep until the full InvSim is solved. Two main disadvantages arise with the differential method: numerical instability due to the numerical time differentiation [124, 125], and lack of generality of the solution process; major changes to the model requiring major changes to the InvSim procedure.

Hess and Gao [124], and Rutherford and Thomson [125] therefore developed an alternative InvSim approach based on the more stable and commonly used numerical time *integration* methods. While these authors initially applied their new proposed techniques to investigate the dynamics of rotorcraft, the developed integral-based InvSim method is general and independent of the model being solved, while providing similar levels of accuracy in the final result [125]. Rutherford and Thomson thus refer to the method as the *generic* InvSim algorithm, or *GENISA* for short. Numerical integration of the governing ODEs over the discretisation timestep allows for the algorithm to form a set of simultaneous algebraic equations which, similarly to the derivative approach, can be solved using the NR method. This, however, requires that the partial derivative of each system output with respect to (w.r.t.) each input be calculated to form a Jacobian, which must then be inverted during each NR iteration to converge towards a solution. Commonly, this is achieved by perturbing the inputs to the model in turn and determining what effect they have on the output using numerical differencing. Although

this is clearly an example of numerical differentiation, which, as mentioned previously, leads to instabilities in the derivative InvSim method, the derivatives here are being taken w.r.t. the inputs and not time [126]. Numerically integrating the ODEs over a discretisation timestep to determine the behaviour of the system proves more accurate than attempting to approximate the same behaviour using a single finite difference over the same period. The generality of the solution process comes at the cost of increased computational complexity, taking an order of magnitude longer than derivative-based methods to compute the InvSim solution. This is due to the integral-based method needing to numerically integrate the ODEs across the discretisation timestep many times to populate the Jacobian used in the NR solver. A review of InvSim methods by Murray-Smith in 2014 provides a clear illustration of the differences between the two main approaches, i.e. derivative and integral-based InvSim methods [127].

While the integral InvSim approach is shown to increase computational stability, there are still several numerical phenomena of concern [128]. One issue encountered is the appearance of unwanted high-frequency low-amplitude oscillations superimposed over the input solution. This problem was identified by Hess and Gao [126] when they first conceived the integral InvSim approach in 1991, who determined that it was caused by the existence of multiple local minima in the formed simultaneous algebraic equations. Their somewhat ad-hoc solution was to perform post-processing input smoothing using a fifth-order low-pass digital filter. Lin [129] discovered that the oscillations could be caused by excitation of unobservable internal dynamics when the discretisation timestep is too small. Shilliday [130] found similar high-frequency chattering oscillations in the input solution, which could be attenuated through the application of stricter tolerances in the NR subroutine. One of the best solutions to this problem was, however, discovered by Rutherford and Thomson, who found that the oscillations could be removed by using derivatives of the desired output trajectory, rather than the output directly, to form the simultaneous algebraic equations [125]. They reason that perturbations in the inputs will have a more instantaneous effect on the acceleration and velocity of the system dynamics, taking more time to propagate through into a change in the position. This therefore avoids issues of similar differencing when calculating the partial derivatives of the Jacobian, improving stability.

Another issue with integral-based InvSim that appears frequently is the issue of redundant systems, i.e. systems that have more inputs than outputs which the InvSim needs to constrain. This results in the set of simultaneous algebraic equations defining the InvSim problem becoming under-determined, meaning that there could be more than one unique input solution that satisfies the desired output trajectory. The problem is manifested in the Jacobian, which for a redundant system will be non-square and not

directly invertible. This poses a problem for the NR solver, which must obtain an inverse of the Jacobian to update the input estimate. Hess and Gao [124] borrow from robotics literature and suggest the use of a Moore-Penrose (MP) pseudoinverse to invert the non-square Jacobian. The authors also point out that a MP pseudoinverse can be used to solve an under-actuated InvSim problem, i.e. fewer inputs than outputs. It is shown that in the case of a redundant system, this pseudoinverse approach provides an input solution with good tracking of the desired output trajectory. However, in the under-actuated case, only convergence to a solution that provides a least-squares fit to the desired output trajectory can be guaranteed [124].

De Matteis et al. [131], however, point out that the MP pseudoinverse has limitations when used to perform InvSim of redundant systems, highlighting that it provides a minimum norm update to the input solution during each iteration of the NR subroutine. Therefore, the input solution converged upon at each discretisation timestep will be the closest to the initial input estimate. In the case of a redundant system where multiple possible solutions may exist, careful choice of this initial input estimate must be made to ensure that a preferable InvSim solution is converged to. To alleviate this requirement, De Matteis et al. [131] propose replacing the usual NR solver with a constrained local optimisation method. In this approach, the simultaneous algebraic equations obtained through numerical integration are treated as equality constraints, with a particular solution for a redundant system being converged upon through a suitable choice of cost function, avoiding the need to select the initial input estimate carefully. Inequality constraints can also be included to consider limitations in the system, such as actuator control saturation.

Lee and Kim [132] also develop a local optimisation-based InvSim method which requires no numerical integration of the ODEs. The authors use time-domain finite elements to discretise the InvSim problem. Each timestep of the InvSim is then solved in parallel using a least-squares optimisation algorithm; as mentioned, integral-based methods are solved iteratively one timestep at a time. Their method also ensures a specific input solution is converged upon, provided an appropriate cost function is chosen. The integration-free method does still however exhibit the high-frequency low-amplitude oscillations discussed previously. Borri et al. [133] also developed a parallel, integration-free InvSim method using time-domain finite elements. The authors used the finite elements to discretise the ODEs of the dynamics and form algebraic equations. Additional equations are included to constrain the outputs, and the full system of simultaneous equations is solved using the NR method.

One requirement for all the aforementioned InvSim methods is that a full-time history of the desired output must be available. In many cases, this desired output trajectory is

strictly known; however, in some cases, the requirements for the manoeuvre being performed are more general. Celi [134] proposes an optimisation-based method for InvSim of a helicopter performing a slalom manoeuvre. A set of constraints is used to define a *family* of possible output trajectories, with the optimisation-based InvSim method converging upon the one that minimises a defined cost function. A modification to the integral InvSim algorithm is made by Bagiev et al. [135] to account for constraints on the inputs, i.e. actuator saturation. In cases where the desired output trajectory is particularly *aggressive*, these constraints on the inputs may be violated. The authors therefore present a *predictive* integral-based InvSim algorithm (PRISM), which utilises a *receding horizon* to determine whether the input constraints will be violated. If the InvSim algorithm predicts that the inputs will become saturated following the current desired trajectory, then the parameters of the polynomial-shaped trajectory are modified using a decision tree. Once a trajectory is found that satisfies the input constraints within the receding horizon, the InvSim can accept the input solution for the current discretisation timestep and move the receding horizon forward.

Avanzini et al. [136] also make use of a receding horizon within an MPC architecture for their integral-based InvSim algorithm. The NR solver uses a faster-to-compute lower-fidelity model to find the input that gives the required output at the end of the receding horizon. The dynamics are then propagated forward over the smaller discretisation timestep using a higher-fidelity model to give the initial conditions for the next iteration of the InvSim. To account for the differences between the two models, a simple proportional guidance scheme is applied between each InvSim iteration to avoid divergence from the desired trajectory. It is demonstrated that the MPC architecture significantly improves computational speed, at the cost of some small divergence from the original desired trajectory. Zhou et al. [137] iterate on the MPC InvSim architecture, replacing the iterative numerical optimisation method with a neural network trained to provide an approximation of the inverse dynamic behaviour. The neural network evaluates the output to input mapping instantaneously, with the receding horizon compensating for discrepancies in its approximation.

The two-timescale approach is another proposed by Avanzini et al. [138] to speed up the computation of the integral-based InvSim method. One determining factor in the speed of the InvSim will be the size of the discretisation timestep chosen. It needs to be small enough to capture the system's dynamics, but not so small that the InvSim has to solve for too many iterations. In the two-timescale approach, the fast and slow dynamics are decoupled into two separate systems. This allows for a coarser, larger discretisation timestep to be used for a portion of the dynamics, thus increasing the overall speed of the InvSim. This separation of the dynamics is, however, based on

some physical reasoning and knowledge of both the dynamics and the particular model being used. Therefore, some generality in the solution process has been lost; a major selling point of integral-based InvSim.

The major limiting factor in the computational speed of the integral-based InvSim method is the number of times the ODEs must be numerically integrated to form the Jacobian. Additionally, the number of integrations increases with the number of inputs in the system, further slowing the InvSim down, and eventually becoming prohibitive. Yuan et al. [139] propose the use of automatic differentiation, a method widely used within machine learning, to speed up calculation of the Jacobian. The method works by leveraging the chain rule to calculate the partial derivatives of the Jacobian, reducing the number of times the ODEs must be numerically integrated to just two, no matter the number of inputs. The speed improvement comes at the cost of requiring intermediate values obtained during the elementary operations (addition, subtraction, multiplication etc.) of the ODEs to be stored in memory. This could potentially become a limiting factor if the model being solved is complex, or if trying to implement the InvSim algorithm on embedded hardware for real-time execution. Many implementations of automatic differentiation require external software to modify the source code of the model into a form that accommodates the chain rule method. This introduces an additional step into the InvSim implementation, requiring that code be regenerated every time a modification is made to the model being solved. Again, this means an undesirable loss of generality in the integral-based InvSim process.

Most of the aforementioned InvSim methods have used the calculation of derivatives at some point during their solution process (even the integral-based methods), either during the discretisation process or by the numerical solver. However, where discontinuities exist in the dynamics of the system, or the prescribed desired output, these InvSim methods will fail. Lu et. al [128] present a derivative-free search-based optimisation approach to deal with such systems. The authors use a Nelder-Mead downhill simplex search algorithm to solve an unconstrained optimisation problem. The constraints on the output are implied through inclusion in the cost function, with input saturations handled by a transformation of the constrained input space to an unconstrained one that the search algorithm can handle. The authors demonstrate improved convergence compared to the usual NR solver in cases where input saturation is reached, or when the desired manoeuvre is discontinuous.

Of all the InvSim methods outlined herein, those based upon numerical integration prove to be the most popular [140]. Despite the additional computational cost, when compared to those based on numerical differentiation, the generality of the solution process allows for easy application to a multitude of systems. Many variations on the

integral-based InvSim approach have been seen, those based upon NR [124, 125], local gradient-based optimisation [131], and search-based optimisation [128]. Other modifications, such as the use of automatic-differentiation [139], MPC architecture [136], or neural networks [137], improve the computational efficiency of the InvSim. The use of a predictive receding horizon also ensures that the InvSim solution follows a trajectory realisable by limitations in the system's actuators and dynamics [135].

Applications

Many of the major developments in InvSim, particularly during the 1980s and 1990s, were focused on applying InvSim to study the dynamics of rotorcraft [123–125, 141, 142], and fixed-wing aircraft [119, 122, 126, 131, 132, 138, 143]. There continues to be much interest in the application of InvSim within the fields of rotorcraft [135, 136, 144–146], including quadrotor rotorcraft [147], and fixed-wing aircraft [145, 146]. More exotic aerospace vehicles, including hybrid tilt-rotor aircraft [139, 148], and autogyros [149] have also seen InvSim applications. Other applications include amphibious underwater vehicles [150, 151], and container ships [146]. Applications within space engineering have thus far been limited to planetary rovers [130, 152–156], and controlled rendezvous and docking of two spacecraft in orbit [137]. Therefore, this thesis aims, in part, to be the first to investigate the application of InvSim to satellite attitude control, and more specifically to satellites with flexible structures.

It can be seen that InvSim has been successfully applied to a variety of different vehicle types, with the utility that InvSim serves in each of these applications being equally as varied. Thomson and Bradley [140] outline the use of InvSim as a tool for flight dynamic research. They discuss using InvSim to model the behaviour of pilots, perform model validation, and complete investigations into the performance and handling qualities of different aircraft designs. The authors demonstrate that InvSim can be used to validate mathematical models when provided with measured flight data [157]. Feeding the measured data into the InvSim produces the inputs required for the model to reproduce the original measurements, which can be compared to the inputs that were applied to the real vehicle. This will help identify parameters in the model that need adjusting to improve model validity and also highlight any fundamental shortcomings in the type of mathematical model being used. One limitation is that most InvSim methods, apart from those that are gradient-free [128], require that the desired output time history be smooth and continuous. Real-world measured flight data will exhibit noise, which must therefore be smoothed through appropriate filtering before being given to the InvSim for model validation.

Thomson and Bradley [140] also highlight that InvSim provides useful information when assessing an aircraft's characteristics, and performance when completing a required manoeuvre. Using InvSim allows for airworthiness and flying qualities criteria, specified by regulatory bodies, to be considered during the preliminary conceptual design process [141]. Thomson and Bradley [142] describe that the study of complex manoeuvres using traditional forward simulation techniques can become difficult, as the required inputs to perform such manoeuvres cannot necessarily be known beforehand. Hess and Gao [124] reiterate that the study of these complex manoeuvres using forward simulation would require the inclusion of a closed-loop controller within the model. The authors point out that InvSim provides an alternative approach to perform these studies, without the need for this bespoke controller to be designed. However, inevitably, a bespoke controller will have to be designed eventually. At this point, InvSim can be used to determine bandwidth requirements that the bespoke controller must achieve, in order to attain good tracking performance.

The input solution converged upon during InvSim has also been used directly within some control system designs. Worrall et al. [152] used InvSim with a validated model to produce an offline, open-loop controller for a planetary rover. The authors include practical testing of their InvSim-based controller using an existing real rover. They find that the controller achieves high positional accuracy when performing a relatively simple manoeuvre. Lu et al. [146] investigated whether InvSim could replace traditional model-inversion techniques, such as NDI, to produce the feedforward control element in a model-following architecture (i.e. includes a feedback term too). The authors demonstrated that the InvSim feedforward term improves tracking performance when applied to both a linearised multi-input-multi-output (MIMO) helicopter model, and a nonlinear single-input-single-output (SISO) model of a container ship. An additional feedback term in the model-following architecture is provided using H^{∞} control, ensuring robustness to measurement noise and output disturbances.

Ireland et al. [158] performed a detailed comparison of InvSim and NDI as suitable model-inversion techniques. It was found that InvSim provided a more accurate model inversion, particularly for higher-order systems, at the cost of greater computational effort. Furthermore, the computational time of NDI remains inversely proportional to the size of the discretisation timestep; in contrast, the time taken to execute InvSim can vary depending on the number of iterations required by the solver (eg. NR) to converge. The authors, however, highlight that the main benefit of InvSim, specifically integral-based InvSim, is in its practicality; not requiring any modification to the solution process whenever a change is made to the model. Conversely, NDI requires that the inverse solution be redefined, and redeployed whenever changes are made to the

model. InvSim is also able to handle systems that are control non-affine, i.e. have a non-linear relationship to the inputs, which NDI cannot handle without performing some additional linearisation of the system.

Finally, Shilliday et al. [156] applies InvSim as part of a model-based fault detection procedure. The model-inversion provided by InvSim allows for residuals to be generated at different locations within the system, giving information that can be used for health monitoring. Combined with residuals generated through traditional forward simulation, the approach allowed for the detection and diagnosis of singular, additive faults in both the inputs and outputs.

1.3 | Aims and Objectives

Now that an understanding of the current state-of-the-art in attitude control of satellites with flexible structures, and InvSim numerical techniques has been established, the overall research direction for this thesis can be laid out.

It has been described that accurate attitude pointing is an important consideration for almost all satellite missions, proving a particularly challenging problem when flexible structural behaviour is exhibited. A wide variety of attitude control algorithms and strategies found in the literature have been presented, many of which rely on a form of model-inversion in their derivation.

It has been shown in several fields, including aerospace, that numerical integral-based InvSim methods provide a viable, and flexible alternative to traditional analytical model-inversion techniques. The integral InvSim algorithm has also been demonstrated as a useful tool in conceptual design, allowing for assessment of the system's dynamic performance during the execution of specific manoeuvres. Therefore, this thesis will investigate the use of integral-based InvSim methods for the attitude control and design of satellites with flexible structures, attempting to answer the research question:

Can Inverse Simulation improve the attitude control and design process of satellites with flexible structures, when compared to traditional techniques?

This research question will be answered by setting out the following three aims:

AIM 1 | To develop novel InvSim methodologies to investigate the attitude dynamics and control of satellites with flexible structures

AIM 2 | To develop an InvSim-based attitude control architecture, assessing its performance and practicality

AIM 3 | To develop a framework that utilises InvSim for the design of satellites with flexible structures

These aims will be met by tackling ten objectives as detailed below:

OBJ 1 | Develop suitable mathematical models and forward simulations

OBJ 2 | Apply the integral InvSim algorithm to the attitude dynamics of satellites with flexible structures

OBJ 3 | Investigate the effect of different hyperparameters on the accuracy and computational efficiency of InvSim

OBJ 4 | Investigate numerical issues associated with InvSim unique to the dynamics of satellites with flexible structures

OBJ 5 | Compare InvSim to alternative model inversion techniques

OBJ 6 | Develop an InvSim-based control architecture

OBJ 7 | Determine the performance of an InvSim-based controller and how it compares to traditional methods

OBJ 8 | Determine the effect of model accuracy on the InvSim-based controller

OBJ 9 | Investigate the use of InvSim to reduce deformation in a satellite's flexible structures

OBJ 10 | Improve numerical performance of existing InvSim methodologies

1.4 | Thesis Outline

The thesis is laid out as follows. Chapter 2 demonstrates the application, flexibility, and practicality of using the integral-based InvSim method to solve the inverse attitude dynamics of satellites. The effect of each of the InvSim hyperparameters on accuracy, and computational efficiency is studied. Numerical stability issues specific to satellite attitude dynamics are investigated. Performance and practicality of the integral InvSim algorithm is also compared to the alternative analytical nonlinear dynamic inversion (NDI) method. Satellites are assumed at this early point in the thesis to act as rigid bodies to simplify analysis and tractability of the results.

Chapter 3 applies InvSim to the development of a hybrid, model-based, attitude control architecture for satellites with flexible appendages. This includes the development of a multi-fidelity mathematical model, consideration of dynamic stiffness on InvSim

stability and computational efficiency, and optimisation of attitude slew manoeuvres to minimise appendage flexing. The effect of model fidelity and uncertainty on the performance of the proposed InvSim-based controller is investigated, and it is demonstrated that models developed in existing simulation frameworks can be utilised by InvSim to produce the inverse dynamic solution, avoiding the need to develop new bespoke models.

Next, Chapter 4 makes use of InvSim to design flexible large space structures (LSSs), investigating the performance of various actuator configurations. This includes the development of a modular mathematical model, actuator scaling laws, and consideration of InvSim for overactuated systems. Chapter 5 then presents a simple, easy-to-implement modification to the integral-based InvSim method to improve computational efficiency. Stability issues specific to the new InvSim method are also addressed.

Finally, Chapter 6 summarises the thesis, evaluating the outcomes of the aforementioned research aims, and drawing conclusions. Future work is detailed before some final concluding remarks are made.

1.4.1 | Contributions to Literature

During the completion of this PhD, several articles were published. The first two papers were presented at the 72nd and 73rd International Astronautical Congresses (IAC) in Dubai and Paris respectively:

R. Gordon, K. Worrall, and M. Ceriotti, "Attitude Control of a Nanosatellite using Inverse Simulation," in *72nd International Astronautical Congress*, Dubai, UAE, 2021, IAC–21, C1, 8, 12, x64025. [Online]. Available: https://eprints.gla.ac.uk/257797/

R. Gordon, K. Worrall, and M. Ceriotti, "Attitude Control of Satellites with Flexible Appendages Using Inverse Simulation," in *73rd International Astronautical Congress*, Paris, France, 2022, IAC-22–C1.1.6. [Online]. Available: https://eprints.gla.ac.uk/278365/

IAC 2021 unfortunately had to be attended virtually in the midst of the COVID-19 pandemic. IAC 2022 in Paris was attended in person with sponsorship from the European Space Agency (ESA) as part of the International Space Education Board's (ISEB) program. These papers included some of the preliminary work completed in the first years of this PhD and laid much of the groundwork for chapters 2 and 3 of this thesis. This

initial groundwork was further developed with three additional papers being accepted and published in three different peer-reviewed journals:

R. Gordon, M. Ceriotti, and K. Worrall, "Effects of model fidelity and uncertainty on a model-based attitude controller for satellites with flexible appendages," *Acta Astronautica*, 2023. DOI: 10.1016/j.actaastro.2023.10.011

R. Gordon, K. Worrall, and M. Ceriotti, "Broyden's Method for Faster Inverse Simulation," *Journal of Guidance, Control, and Dynamics*, vol. 47, no. 9, pp. 1939–1945, 2024. DOI: 10.2514/1.G007975

R. Gordon, M. Ceriotti, and K. Worrall, "Investigation of attitude control actuators for large flexible space structures using inverse simulation," *Advances in Space Research*, vol. 75, no. 2, pp. 2062–2087, 2025. DOI: 10.1016/j.asr.2024.11.010

The first journal publication built upon the work of the second IAC paper and contributes to much of Chapter 3. The second journal paper looked to improve the existing InvSim algorithm through a minor modification and is presented in Chapter 5. Chapter 5 will include some additional work not published in the original paper, dealing with limitations in the new method when applied to specific types of systems. Much of Chapter 4, covering the use of InvSim as a design tool for large flexible space structures, was published in the final journal paper detailed above.

Chapter 2 | Inverse Simulation of Satellite Attitude Dynamics

As discussed in Chapter 1, a common element in many attitude control architectures involves the use of model-inversion techniques; i.e. feedforward, feedback linearisation, sliding mode control etc. While analytical approaches do exist, it was suggested in Section 1.3 that numerical Inverse Simulation (InvSim) techniques could provide a general, ready-to-go inverse dynamic solution. Before attempting to develop an InvSim-based attitude control architecture, it is first important that the InvSim numerical technique is well understood, particularly in its application to satellite attitude dynamics.

To simplify analysis, this chapter will only consider rigid body attitude dynamics, as described in Appendix E, leaving consideration of flexible modes to subsequent chapters. The dynamics of several attitude control actuators are included, which highlights some cases of InvSim numerical instability unique to satellites. Performance of the numerical InvSim process, both in terms of accuracy and computational complexity, is also explored; the latter of which could have a profound impact when attempting to deploy an InvSim-based attitude control architecture onto real-time hardware. The flexibility and practicality of the InvSim methodology is also explored and compared to traditional analytical model inversion techniques. First, however, the specific InvSim method utilised in this thesis is laid out.

2.1 | The Inverse Simulation Method

As discussed in Section 1.2.4, there are several different InvSim methods primarily based on either numerical differentiation or numerical integration. The popular numerical integration approach is adopted in this thesis for its general solution process, and versatility in solving different dynamic systems. Outlined in this section is the most basic form of the integral InvSim algorithm developed by Hess *et al.* [126], and Rutherford and Thomson [125]; based on the NR numerical method (Appendix D.1.1). Variations of this algorithm, some from literature and some developed as part of this

thesis, will be seen in later chapters. However, in every case, the InvSim is based upon the algorithm outlined here.

2.1.1 | Forward Simulation

The integral InvSim at its most basic applies a numerical method to solve the inverse dynamic problem by evaluating the forward dynamics. The forward dynamic solutions can be found through numerical integration of the ODEs of motion, i.e. forward simulation. First, consider a nonlinear dynamic system in the following form:

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}) \tag{2.1}$$

$$\boldsymbol{y} = \boldsymbol{g}(\boldsymbol{x}) \tag{2.2}$$

where $\dot{\boldsymbol{x}}$ is the state derivative, $\boldsymbol{x}$ is the state, $\boldsymbol{u}$ is the input, $\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u})$ is a set of nonlinear ODEs, $\boldsymbol{y}$ is the output of the system, and $\boldsymbol{g}(\boldsymbol{x})$ is a set of functions mapping the state to the output. The state time history $\boldsymbol{x}(t)$ can therefore be solved through appropriate integration:

$$\boldsymbol{x}(t) = \int_{t_0}^{t} \dot{\boldsymbol{x}}(\tau)\, d\tau + \boldsymbol{x}(t_0) = \int_{t_0}^{t} \boldsymbol{f}(\boldsymbol{x}(\tau), \boldsymbol{u}(\tau))\, d\tau + \boldsymbol{x}(t_0) \tag{2.3}$$

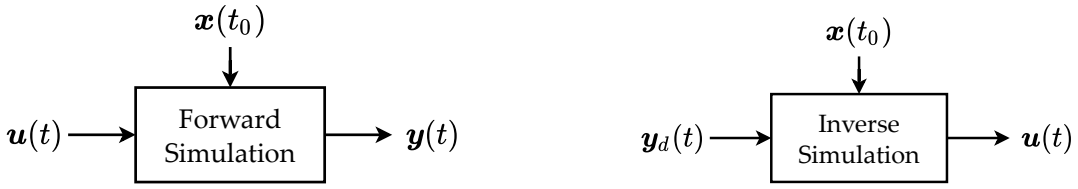
where $\boldsymbol{x}(t_0)$ is the initial state of the system. The integration can be performed numerically using methods detailed in Appendix D.3 to obtain a solution to the system state dynamics. Similarly the time history of the output $\boldsymbol{y}(t)$ can be found through substitution of Eq. (2.3) into Eq. (2.2):

$$\boldsymbol{y}(t) = \boldsymbol{g}(\boldsymbol{x}(t)) = \boldsymbol{g}\left(\int_{t_0}^{t} \boldsymbol{f}(\boldsymbol{x}(\tau), \boldsymbol{u}(\tau))\, d\tau + \boldsymbol{x}(t_0)\right) = \boldsymbol{h}(\boldsymbol{x}(t_0), \boldsymbol{u}(t)) \tag{2.4}$$

where $\boldsymbol{h}(\boldsymbol{x}(t_0), \boldsymbol{u}(t))$ represents the *forward* simulation, encapsulating the numerical integration of the system dynamics. Forward simulation is therefore a numerical process that maps an input (usually the control) time history $\boldsymbol{u}(t)$ to an output time history $\boldsymbol{y}(t)$, subject to the initial state $\boldsymbol{x}(t_0)$ of the system as illustrated in Fig. 2.1a. InvSim, as shown in Fig. 2.1b, is a numerical process that will conversely map a time history of desired output states $\boldsymbol{y}_d(t)$ to a time history of the required inputs $\boldsymbol{u}(t)$.

2.1.2 | Integral Inverse Simulation Algorithm

The first step to the InvSim process is to define this time history of desired output states $\boldsymbol{y}_d(t)$ that the dynamics of the system must follow. This poses a *path planning* problem in of itself, as the desired outputs $\boldsymbol{y}_d(t)$ must be realisable by the system dynamics



(a) Forward Simulation: A numerical process that maps inputs $\boldsymbol{u}(t)$ to outputs $\boldsymbol{y}(t)$

(b) Inverse Simulation: A numerical process that maps desired outputs $\boldsymbol{y}_d(t)$ to required inputs $\boldsymbol{u}(t)$

Figure 2.1: Comparison of Forward Simulation and Inverse Simulation

for the InvSim to converge. The issue of non-convergence and unrealisable output trajectories will be discussed later in this chapter. It will, for now, be assumed that a realisable desired output time history $\boldsymbol{y}_d(t)$ is generated through a suitable method, such as spherical-linear-interpolation (SLERP) (Appendix B.4) in the case of satellite attitude dynamics.

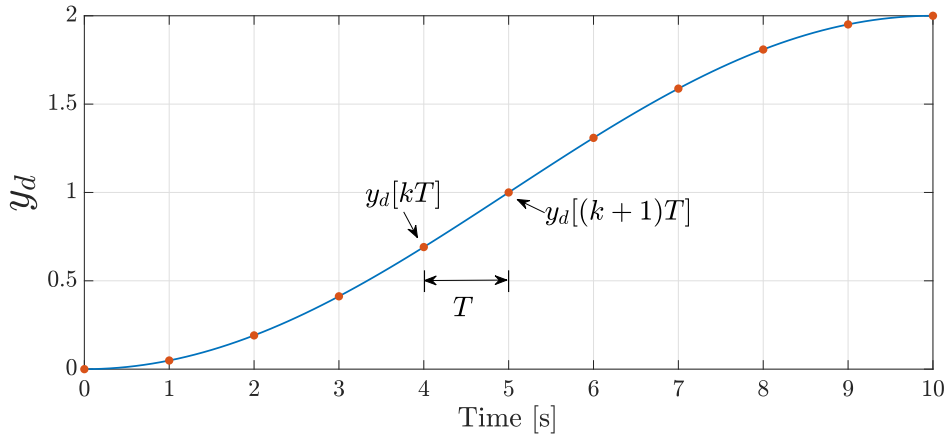


Figure 2.2: Discretisation of the desired output $\boldsymbol{y}_d$

The InvSim must therefore find the input time history $\boldsymbol{u}(t)$ such that the output of the system $\boldsymbol{y}(t)$ matches the desired $\boldsymbol{y}_d(t)$. The desired output time history $\boldsymbol{y}_d(t)$ is first *discretised* by a discretisation timestep T, as can be seen in Fig. 2.2. Integrating the dynamics forward for a constant input $\boldsymbol{u}(t) = \boldsymbol{u}[kT]$ over the current discretisation timestep using Eq. (2.4) allows for the InvSim problem to be formed as a set of simultaneous equations:

$$\delta \boldsymbol{y} = \boldsymbol{y}[(k+1)T] - \boldsymbol{y}_d[(k+1)T] = \boldsymbol{h}(\boldsymbol{x}[kT], \boldsymbol{u}[kT]) - \boldsymbol{y}_d[(k+1)T] = 0 \tag{2.5}$$

where $\boldsymbol{y}[(k+1)T]$ and $\boldsymbol{y}_d[(k+1)T]$ are the outputs and desired outputs at the end of the current discretisation timestep respectively, $\delta \boldsymbol{y}$ is the error between them, and $\boldsymbol{h}(\boldsymbol{x}[kT], \boldsymbol{u}[kT])$ is the forward simulation from $t = kT$ to $t = (k+1)T$. The solution to Eq. (2.5) can then be found numerically using the NR method (Appendix D.1.1).

However, it has been found that using first, or higher-order derivatives, of the output as the target for the InvSim can improve numerical stability and avoid issues of high frequency, low amplitude oscillations [125]. For example, using InvSim to track the velocity along a path should also ensure that the position along the path is tracked. Defining the ν^{th}-order derivative of the output $\overset{(\nu)}{\dot{\boldsymbol{y}}}$, the InvSim problem in Eq. (2.5) can be generalised as:

$$\delta\overset{(\nu)}{\dot{\boldsymbol{y}}} = \overset{(\nu)}{\dot{\boldsymbol{y}}}[(k+1)T] - \overset{(\nu)}{\dot{\boldsymbol{y}}}_d[(k+1)T] = \overset{(\nu)}{\dot{\boldsymbol{h}}}(\boldsymbol{x}[kT], \boldsymbol{u}[kT]) - \overset{(\nu)}{\dot{\boldsymbol{y}}}_d[(k+1)T] = 0 \tag{2.6}$$

where $\dot{\boldsymbol{h}}^{(\nu)}(\boldsymbol{x}[kT], \boldsymbol{u}[kT])$ includes the forward simulation and mapping to the ν^{th}-order derivative of the output $\overset{(\nu)}{\dot{\boldsymbol{y}}}[(k+1)T]$. To target the output directly in the InvSim ν can be set to $\nu = 0$ in Eq. (2.6), which is then equivalent to Eq. (2.5). The velocity and acceleration of the output can be targeted by setting ν to $\nu = 2$ and $\nu = 3$ respectively. In each case the input solution $\boldsymbol{u}[kT]$ is converged upon using the NR update, as defined in Appendix D.1 (Eq. (D.7)), which can be generalised for any ν:

$$\boldsymbol{u}_n[kT] = \boldsymbol{u}_{n-1}[kT] - \mathbf{J}_{\boldsymbol{u}}^{-1}\left(\dot{\boldsymbol{h}}^{(\nu)}\right)\delta\overset{(\nu)}{\dot{\boldsymbol{y}}} \tag{2.7}$$

where n is the current iteration of the NR method, and $\mathbf{J}_{\boldsymbol{u}}\left(\dot{\boldsymbol{h}}^{(\nu)}\right)$ is the Jacobian which consists of the partial derivatives of each output/output derivative w.r.t each input:

$$\begin{aligned}\mathbf{J}_{\boldsymbol{u}}\left(\dot{\boldsymbol{h}}^{(\nu)}\right) &= \begin{bmatrix} \dfrac{\partial\overset{(\nu)}{\dot{\boldsymbol{h}}}(\boldsymbol{x}[kT], \boldsymbol{u}_{n-1}[kT])}{\partial u_1} & \dfrac{\partial\overset{(\nu)}{\dot{\boldsymbol{h}}}(\boldsymbol{x}[kT], \boldsymbol{u}_{n-1}[kT])}{\partial u_2} & \cdots & \dfrac{\partial\overset{(\nu)}{\dot{\boldsymbol{h}}}(\boldsymbol{x}[kT], \boldsymbol{u}_{n-1}[kT])}{\partial u_{N_u}} \end{bmatrix} \\ &= \begin{bmatrix} \dfrac{\partial y_1}{\partial u_1} & \dfrac{\partial y_1}{\partial u_2} & \cdots & \dfrac{\partial y_1}{\partial u_{N_u}} \\ \dfrac{\partial y_2}{\partial u_1} & \dfrac{\partial y_2}{\partial u_2} & \cdots & \dfrac{\partial y_2}{\partial u_{N_u}} \\ \vdots & \vdots & \ddots & \vdots \\ \dfrac{\partial y_{N_y}}{\partial u_1} & \dfrac{\partial y_{N_y}}{\partial u_2} & \cdots & \dfrac{\partial y_{N_y}}{\partial u_{N_u}} \end{bmatrix}\end{aligned} \tag{2.8}$$

where N_u and N_y are the number of inputs and outputs in the system. The Jacobian in Eq. (2.8) is calculated numerically to ensure generality of the InvSim process. Greater computational efficiency could be achieved through analytical derivation of each of the partial derivatives, however, this would be model and system-specific, requiring re-derivation each time the model or system being solved changes. The partial derivatives

can be found through numerical central differencing [125]:

$$\frac{\partial \overset{(\nu)}{\dot{\boldsymbol{h}}}(\boldsymbol{x}[kT], \boldsymbol{u}_{n-1}[kT])}{\partial u_i} = \frac{\overset{(\nu)}{\dot{\boldsymbol{h}}}(\boldsymbol{x}[kT], \boldsymbol{u}_{n-1}[kT] + \Delta u_i \hat{\boldsymbol{u}}_{\boldsymbol{i}}) - \overset{(\nu)}{\dot{\boldsymbol{h}}}(\boldsymbol{x}[kT], \boldsymbol{u}_{n-1}[kT] - \Delta u_i \hat{\boldsymbol{u}}_{\boldsymbol{i}})}{2\Delta u_i} \tag{2.9}$$

where Δu_i is the perturbation applied to the i^{th} input, and $\hat{\boldsymbol{u}}_{\boldsymbol{i}}$ is the unit vector direction of the i^{th} input (e.g. $\hat{\boldsymbol{u}}_2 = [0\ 1\ 0]^\top$). A vector of perturbations that will cause the output to change sufficiently for numerical differencing can be found through scaling of the previous input solution by the discretisation step size:

$$\boldsymbol{\Delta u} = T\boldsymbol{u}[(k-1)T] = \begin{bmatrix} \Delta u_1 & \Delta u_2 & \cdots & \Delta u_i & \cdots & \Delta u_{N_u} \end{bmatrix}^\top \tag{2.10}$$

with a minimum perturbation size Δu_{min} enforced to avoid numerical rounding errors:

$$\Delta u_i > \Delta u_{min} \tag{2.11}$$

The NR process iterates until either the magnitude of the error meets some defined tolerance $\delta \overset{(\nu)}{\dot{\boldsymbol{y}}}_{tol}$, or the number of NR iterations exceeds some defined maximum n_{max}:

$$||\delta \overset{(\nu)}{\dot{\boldsymbol{y}}}|| < \delta \overset{(\nu)}{\dot{\boldsymbol{y}}}_{tol} \quad \text{or} \quad n > n_{max} \tag{2.12}$$

at which point the current estimate of the input solution for the current timestep is accepted as the solution:

$$\boldsymbol{u}[kT] \leftarrow \boldsymbol{u}_n[kT] \tag{2.13}$$

and the process iterates onto solving the input at the next discretisation timestep until the full timeseries is solved, and the InvSim is completed. An in-depth step-by-step process of the full Integral InvSim algorithm can be seen in Algorithm 1. One additional consideration is the need to apply a MP pseudoinverse (Eq. (A.19)) on the Jacobian $\mathbf{J}_{\boldsymbol{u}}$ when it is non-square ($N_u \neq N_y$); commonly used when solving underdetermined and overdetermined systems of equations (Appendix D.1.1). When $N_u < N_y$, the system is said to be *underactuated* with the InvSim not guaranteed to converge upon an exact solution that satisfies Eq. (2.6). Instead, the InvSim will converge to an input solution $\boldsymbol{u}(t)$ that gives a least-squares fit to Eq. (2.6).

In the case where $N_u > N_y$ the system is said to be *overactuated* or *redundant* such that more than one solution of $\boldsymbol{u}[kT]$ could satisfy Eq. (2.6). The initial input estimate at the start of each timestep $\boldsymbol{u}_0[kT]$ will therefore determine which of the multiple solutions is converged upon. As shown in Algorithm 1 this is usually taken to be the solution from the previous timestep $\boldsymbol{u}_0[kT] = \boldsymbol{u}[(k-1)T]$, except in the case of the first timestep

Algorithm 1: Integral Inverse Simulation

Generate timeseries of desired output/output derivative states $\dot{\boldsymbol{y}}_d^{(\nu)}(t)$;
Define an initial input estimate for the first step $\boldsymbol{u}_0$;
for $k = 1 \rightarrow \texttt{length}(\dot{\boldsymbol{y}}_d^{(\nu)}[kT])$ **do**
    Sample desired output/output derivative $\dot{\boldsymbol{y}}_d^{(\nu)}[(k+1)T]$;
    Initialise input estimate $\boldsymbol{u}_0[kT] = \begin{cases} \boldsymbol{u}[(k-1)T] & \text{for } k > 1 \\ \boldsymbol{u}_0 & \text{for } k = 1 \end{cases}$;
    Run forward simulation with initial input estimate $\dot{\boldsymbol{h}}^{(\nu)}(\boldsymbol{x}[kT], \boldsymbol{u}_0[kT])$;
    while $n < n_{max}$ **and** $\delta\dot{\boldsymbol{y}}^{(\nu)} > \delta\dot{y}_{tol}^{(\nu)}$ **do**
        Calculate error $\delta\dot{\boldsymbol{y}}^{(\nu)}$ (Eq. (2.6));
        Calculate Jacobian $\mathbf{J}_{\boldsymbol{u}}$ (Eq. (2.9));
        Update input estimate $\boldsymbol{u}_n[kT] = \boldsymbol{u}_{n-1}[kT] - \mathbf{J}_{\boldsymbol{u}}^{-1}\delta\dot{\boldsymbol{y}}^{(\nu)}$;
        Run forward simulation with updated input solution $\dot{\boldsymbol{h}}^{(\nu)}(\boldsymbol{x}[kT], \boldsymbol{u}_n[kT])$;
        Update error to check tolerance $\delta\dot{\boldsymbol{y}}^{(\nu)}$ (Eq. (2.6));
    end
    Save input solution $\boldsymbol{u}[kT] \leftarrow \boldsymbol{u}_n[kT]$;
end

($k = 1$) where a user-defined value $\boldsymbol{u}_0$ must be used. This user-defined initial input estimate $\boldsymbol{u}_0$ is one of several hyperparameters of the InvSim that must be selected. A list of each hyperparameter and its description can be seen in Table 2.1.

Now that the specific integral-based InvSim algorithm utilised in this chapter, and for the majority of this thesis, has been presented, its application to solving the inverse attitude dynamics of satellites can be demonstrated.

Table 2.1: Inverse Simulation hyperparameter summary

Hyperparameter		Description
Discretisation timestep	T	Time between successive InvSim solutions.
Integration timestep	Δt	Step size used by numerical integration methods to solve the forward simulation. Should ensure numerical stability of the forward simulation; determined by the speed of the dynamics.
NR tolerance	$\delta \dot{\boldsymbol{y}}_{tol}^{(\nu)}$	Input solution iteration $\boldsymbol{u}_n[kT]$ accepted should Eq. (2.6) be satisfied within this tolerance.
Maximum NR iterations	n_{max}	Maximum number of iterations of the NR loop before the current iterate $\boldsymbol{u}_n[kT]$ is accepted.
Minimum finite difference perturbation	Δu_{min}	Minimum size of perturbation applied to the inputs when calculating the partial derivatives of the Jacobian through numerical (central) differencing. Should be large enough to avoid numerical rounding errors.
Desired output	$\dot{\boldsymbol{y}}_d^{(\nu)}$	The output of the system to be targeted by the InvSim through Eq. (2.6), e.g. for attitude dynamics Euler Angles and quaternions, as detailed in Appendix B, could be used.
Desired output order	ν	Differential order of the output used by the InvSim to target through Eq. (2.6), e.g. position (ν=0), velocity (ν=1), or acceleration (ν=2).

2.2 | Application of Inverse Simulation

As mentioned previously, demonstration of the applicability of the integral-based InvSim algorithm (Algorithm 1) is achieved in this chapter using several simple, rigid-body models, with various attitude control actuators. Full description of these models and their associated ODEs can be found in Appendix E. The model parameters shown in Table 2.2 are used, for the most part, consistently throughout this section. The inertia $\mathbf{I}_{tot}$ is arbitrary and chosen to provide three distinct moments of inertia about each of the body axes, and to include off-diagonal terms for added complexity. Some analysis

later in the section will require modification of the parameters, however, this will be detailed where and when the changes are made.

Table 2.2: Rigid-body satellite parameters

Parameter	Value
Inertia, $\mathbf{I}_{tot}$	$\begin{bmatrix} 500 & -10 & -20 \\ -10 & 400 & -30 \\ -20 & -30 & 350 \end{bmatrix}$ kgm^2
Reaction wheel time constant, T_w	0.01 s
Magnetic field strength, $\|\|\boldsymbol{b}_{mag}\|\|$	25.9 µT
Reaction wheel distribution matrix, $\mathbf{A}_w$	$\underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{\text{3-wheel}}$ $\underbrace{\begin{bmatrix} 1 & 0 & 0 & 1/\sqrt{3} \\ 0 & 1 & 0 & 1/\sqrt{3} \\ 0 & 0 & 1 & 1/\sqrt{3} \end{bmatrix}}_{\text{4-wheel}}$

The same rest-to-rest attitude slew manoeuvre will be solved by the InvSim throughout this section. This slew manoeuvre is generated using SLERP and a 7$^{\text{th}}$ order polynomial, as described in Appendix B.4. The initial quaternion $\boldsymbol{q}_i^{B/I}$, final quaternion $\boldsymbol{q}_f^{B/I}$, and slew duration t_f are all given in Table 2.3, equivalent to a 90 deg rotation about the body $\hat{\boldsymbol{x}}^B$ axis. More complex attitude slew manoeuvres will be seen later in this chapter.

Table 2.3: SLERP rest-to-rest slew parameters

Parameter	Value
Initial quaternion, $\boldsymbol{q}_i^{B/I}$	$[1\ 0\ 0\ 0]^\top$
Final quaternion, $\boldsymbol{q}_f^{B/I}$	$[\frac{1}{\sqrt{2}}\ \frac{1}{\sqrt{2}}\ 0\ 0]^\top$
Slew duration, t_f	60 s

This section will include an exhaustive hyperparameter study to ascertain which InvSim hyperparameters have the biggest impact on the accuracy and computational efficiency of the solution. Additionally, some numerical stability issues will be investigated, including some particularly prevalent to satellite attitude dynamics. But first, some example solutions will be presented to demonstrate, in an ideal case, the use of InvSim for solving the inverse attitude dynamics of rigid-body satellites.

2.2.1 | Inverse Simulation Solutions

Here considered is the simplest application of InvSim to the attitude dynamic problem. The integral InvSim algorithm, detailed in Section 2.1, is applied to rigid-body satellites with both external and reaction wheel torque inputs (Appendix E). The values for each InvSim hyperparameter are chosen based on engineering judgement and previous

experience working with InvSim. The hyperparameter study carried out later in this section will better illustrate why some of these hyperparameter values were chosen, but for the time being, they are simply stated in Table 2.4 to obtain some illustrative example solutions.

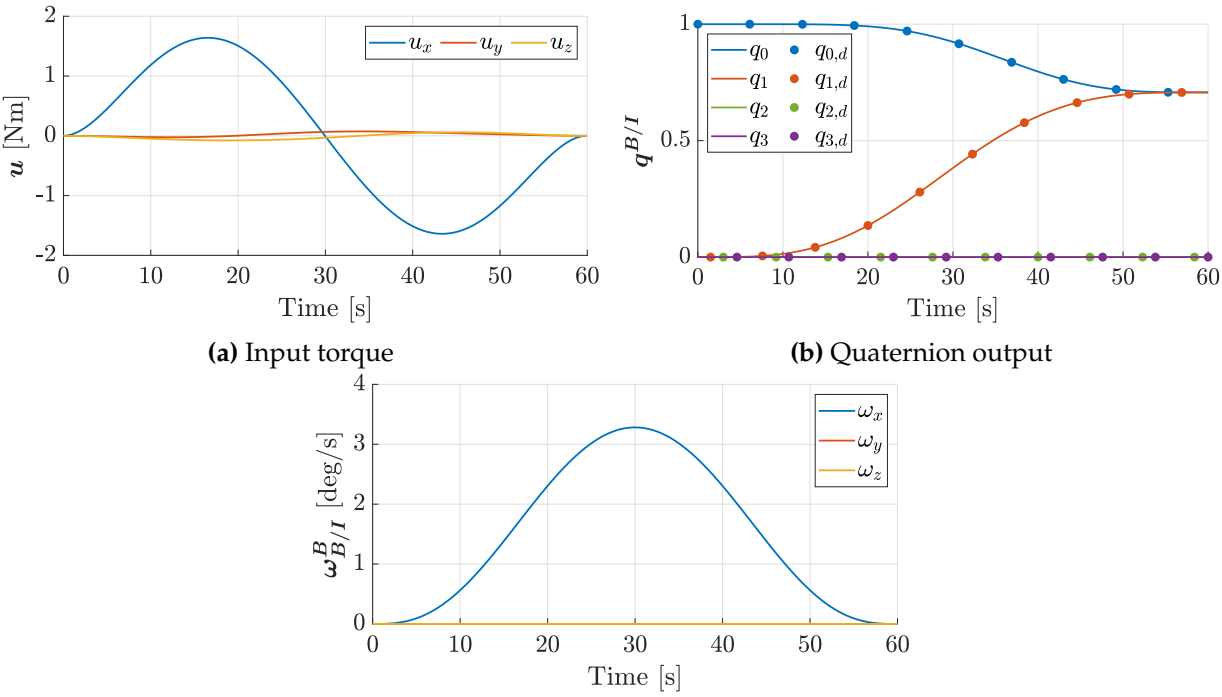
Table 2.4: Inverse Simulation hyperparameters

Hyperparameter	**Value**
Discretisation timestep, T	0.1 s
Integration timestep, Δt	0.01 s
NR tolerance, $\delta\dot{\boldsymbol{y}}_{tol}^{(\nu)}$	1×10^{-10}
Maximum NR iterations, n_{max}	30
Minimum finite difference perturbation, Δu_{min}	1 Nm
Desired output, $\dot{\boldsymbol{y}}_d^{(\nu)}$	$\dot{\boldsymbol{q}}_{d,1:3}^{(\nu)B/I}$
Desired output order, ν	1 (velocity)

External Torques

The first model that the InvSim is applied to is that of a rigid-body with external torque inputs. The InvSim successfully solves the inverse dynamics, producing a timeseries of the input $\boldsymbol{u}(t)$ in Fig. 2.3a. The quaternion response of the InvSim solution and the desired quaternion path of the rest-to-rest SLERP slew manoeuvre, as defined in Table 2.3, are shown in Fig. 2.3b. The desired quaternion path $\boldsymbol{q}_d^{B/I} = [q_{0,d}\ q_{1,d}\ q_{2,d}\ q_{3,d}]^\top$ is marked on Fig. 2.3b with markers (•), with the quaternion response of the InvSim solution $\boldsymbol{q}^{B/I} = [q_0\ q_1\ q_2\ q_3]^\top$ marked with solid lines. It can therefore be seen in Fig. 2.3b that the output has been successfully tracked. The body rate $\boldsymbol{\omega}_{B/I}^B$ of the rigid-body satellite is also plotted and can be found in Fig. 2.3c.

The input solution produced by the InvSim, Fig. 2.3a, indicates that the satellite requires torques predominantly around the body $\hat{\boldsymbol{x}}^B$ axis when performing the attitude slew manoeuvre. This should be expected as the commanded slew is a 90 deg rotation about the body $\hat{\boldsymbol{x}}^B$ axis only, as defined in Table 2.3. However, it can be noted that some small torques are required in both the $\hat{\boldsymbol{y}}^B$ and $\hat{\boldsymbol{z}}^B$ directions. This is due to a combination of the off-diagonal products of inertia in $\mathbf{I}_{tot}$ (Table 2.2) and the cross product term that appears in *Euler's rotational equation* $\boldsymbol{\omega}_B \times \mathbf{I}_{tot}\boldsymbol{\omega}_B$. Later, it will be seen that this dynamic coupling can have an impact on numerical stability when a specific type of actuator is used.



(a) Input torque

(b) Quaternion output

(c) Body rate

Figure 2.3: InvSim solution: rigid-body satellite with external torque input

Reaction Wheels

Similarly, the InvSim solution can be found for a rigid-body satellite with reaction wheel actuators. This only requires that the models be swapped, with no modifications to the InvSim parameters or process required. The InvSim solutions for tracking the same attitude slew manoeuvre for both a 3-wheel and 4-wheel reaction wheel configuration can be found in Fig. 2.5 and Fig. 2.6 respectively. The 3-wheel configuration has a reaction wheel oriented along each of the satellite's body axes, with the 4-wheel set chosen as the NASA standard configuration [81] as illustrated in Fig. 2.4. The actuator distribution matrices $\mathbf{A}_w$ for each configuration are found in Table 2.2.

Again, it can be seen in Figs. 2.5b and 2.6b that the quaternion response of the InvSim solution tracks the desired slew manoeuvre. The reaction wheel torque inputs, Figs. 2.5a and 2.6a, are the inverse of the external torque input seen previously in Fig. 2.3a. This is as expected, as the torque imparted on the satellite's main body will be the opposite *reaction* to the torque applied to the wheels. Additionally, the maximum torque for any one reaction wheel is lower for the 4-wheel configuration, as the 4^{th} wheel contributes to some of the total torque about the $\hat{\boldsymbol{x}}^B$ axis. The angular momentum stored in each wheel can also be plotted, as displayed in Figs. 2.5d and 2.6d. This can be important to consider as reaction wheels will have a maximum angular momentum that can be stored before they become saturated and cannot provide further torques in that direction. It can be seen, therefore, that not only could InvSim be useful

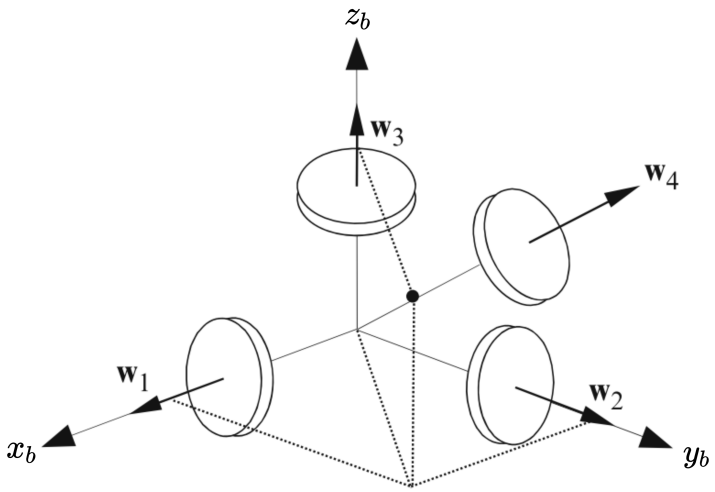
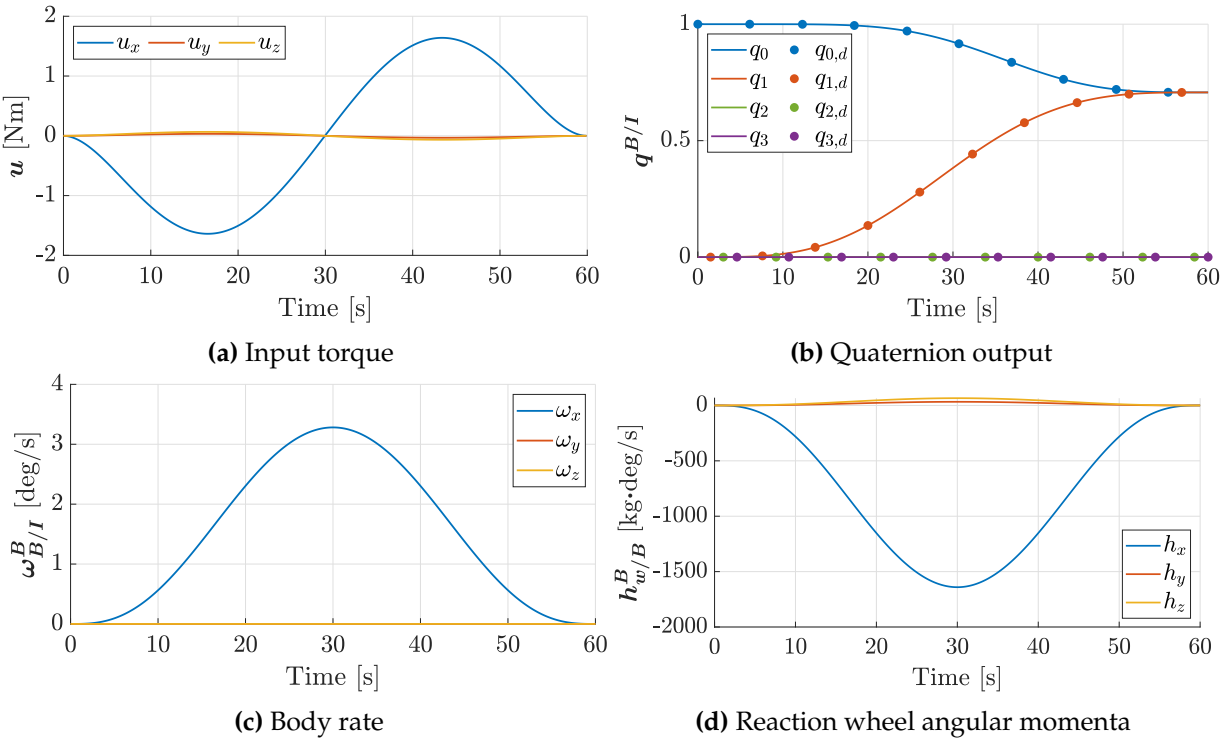


Figure 2.4: NASA standard 4-wheel configuration [81]

in providing the inverse dynamic input solution, but could also be used to assess the performance of the satellite dynamics when completing a prescribed set of manoeuvres. Before the applications of InvSim within the domain of attitude dynamics and control are investigated, further analysis of the InvSim method is required. To this end, the effect of each of InvSim's hyperparameters will be investigated in terms of both accuracy, and computational efficiency.

2.2.2 | Hyperparameter Study

Now that it has been shown that the integral InvSim algorithm can indeed be successfully applied to the attitude dynamics of satellites, a comprehensive analysis of the effect of each InvSim hyperparameter must be performed before the InvSim is applied extensively. Particularly important is to consider the accuracy of InvSim, how well the desired outputs $\boldsymbol{y}_d$ are tracked, and how computationally efficient the InvSim process is. The importance of the former is rather obvious, both for attitude control and the design of satellites with flexible structures. The latter, however, is also particularly pertinent when considering InvSim for use within control applications, as it may be required to run on embedded hardware and run in real-time. In a later chapter, the InvSim will be run within an optimisation problem, requiring a large number of InvSim solutions to be evaluated; a further reason why consideration and understanding of the computational efficiency is pertinent. Some work later in this thesis will consider the speed of InvSim with different numerical integration methods, and modification of the algorithm itself will be performed to improve overall efficiency. However, this section will investigate the most basic form of the integration-based InvSim algorithm, to build a basis of understanding that can be taken through the remainder of this thesis.



(a) Input torque

(b) Quaternion output

(c) Body rate

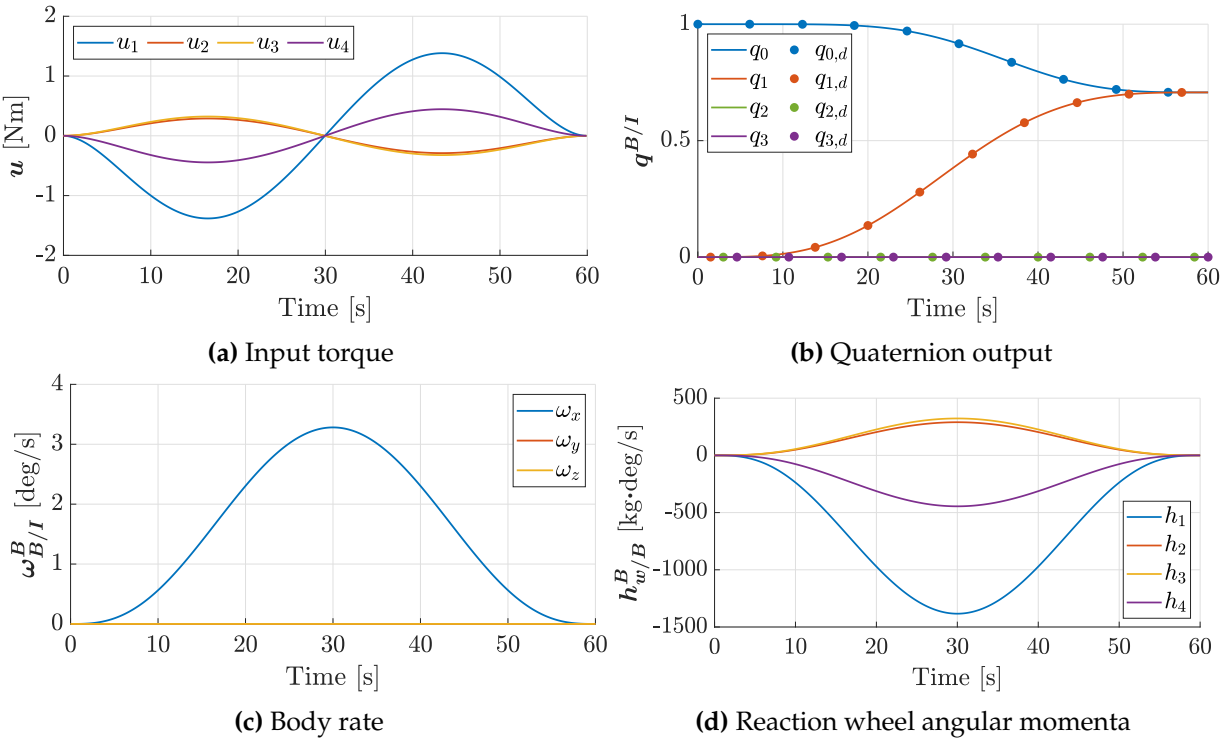
(d) Reaction wheel angular momenta

Figure 2.5: InvSim solution: rigid-body satellite with three reaction wheels

The hyperparameter study will find the InvSim solution for a rigid-body satellite with external torque input, with parameters in Table 2.2, following the same 90 deg rest-to-rest slew manoeuvre seen previously (Table 2.3), while varying each of the hyperparameters in turn. This is a Monte-Carlo approach, with the range of hyperparameters investigated seen in Table 2.5. Note that the integration timestep Δt is defined as a fraction of the discretisation timestep T, i.e. as a *relative* integration step size $\Delta t/T$. In all cases, a quaternion parameterisation of varying differential orders ν, will be used to define the attitude of the satellite, and a constant value will be used for the maximum number of NR iterations n_{max}.

Table 2.5: InvSim hyperparameter monte-carlo ranges

Parameter	**Min**	**Max**	**Units**
Discretisation timestep, T	0.01	2.00	s
Integration timestep, Δt	$0.1T$	T	s
NR tolerance, $\delta\dot{y}^{(\nu)}_{tol}$	1×10^{-20}	0.1	—
Minimum finite difference perturbation, Δu_{min}	1×10^{-3}	10	Nm
Desired output order, ν	0	2	—
Desired output, $\dot{\boldsymbol{y}}^{(\nu)}$	$\dot{\boldsymbol{q}}^{(\nu)B/I}$		—
Maximum NR iterations, n_{max}	30		—



(a) Input torque

(b) Quaternion output

(c) Body rate

(d) Reaction wheel angular momenta

Figure 2.6: InvSim solution: rigid-body satellite with four reaction wheels

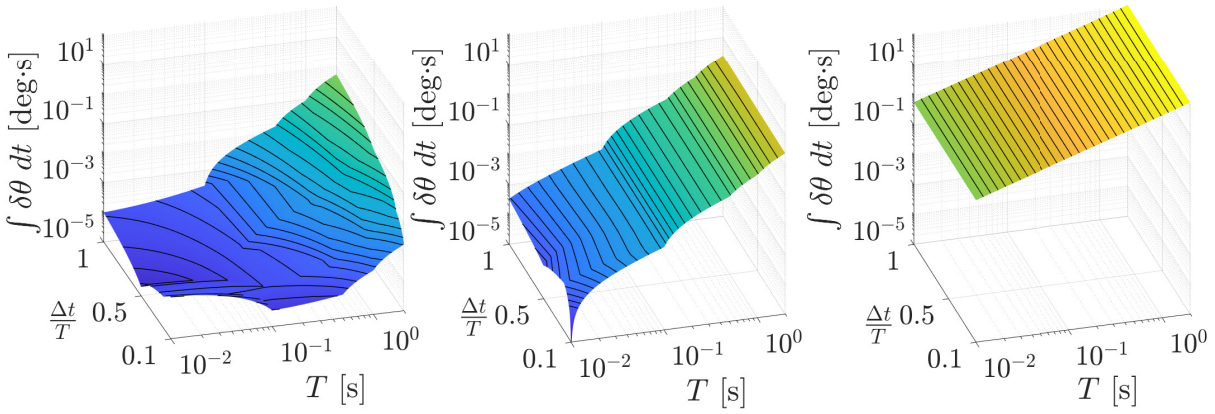
InvSim Accuracy

The accuracy of the InvSim when applied to the attitude dynamics of satellites will be quantified by the integral of the pointing error $\delta\theta$. This can be calculated from the desired quaternion $\boldsymbol{q}_d^{B/I}$ of the slew manoeuvre, and the quaternion response $\boldsymbol{q}^{B/I}$ from the InvSim solution, as defined in Appendix B.2 (Eq. (B.14)):

$$\int_0^{t_f} \delta\theta \, dt = \int_0^{t_f} 2\cos^{-1}\left(\boldsymbol{q}_d^{B/I} \cdot \boldsymbol{q}^{B/I}\right) \, dt \tag{2.14}$$

First considered is the effect that the discretisation timestep T and relative integration step size $\Delta t/T$ have on the accuracy of the InvSim when different orders of the desired output ν are used. The results for the position ($\nu = 0$), velocity ($\nu = 1$), and acceleration ($\nu = 2$) cases can be found in Fig. 2.7a, Fig. 2.7b, and Fig. 2.7c respectively. At first glance, it can be seen that as the desired output order ν increases, the accuracy of the InvSim deteriorates.

The correlation between the desired output order ν and integral of the pointing error (Eq. (2.14)) can be illustrated by considering Fig. 2.8 for the acceleration desired output case ($\nu = 2$), and a large discretisation timestep ($T = 2.0$ s). Only one component $\ddot{q}_1$ of the quaternion acceleration is shown, however, similar trends would be seen across all components $\ddot{\boldsymbol{q}}^{B/I} = [\ddot{q}_0 \; \ddot{q}_1 \; \ddot{q}_2 \; \ddot{q}_3]^\top$. It can be seen that the InvSim solution performs



(a) $\nu = 0$ **(b)** $\nu = 1$ **(c)** $\nu = 2$

Figure 2.7: Effect of discretisation timestep T and relative integration step size $\Delta t/T$ on the attitude pointing error integrated across the entire maneouvre $\int_0^{t_f} \delta\theta\ dt$. Results shown when InvSim targets output position ($\nu = 0$), velocity ($\nu = 1$) and acceleration ($\nu = 2$)

correctly, matching the desired value of $\ddot{q}_1$ at each discretisation step kT. However, over the duration of the discretisation timestep T, there is significant deviation resulting in an error $\delta\ddot{q}_1$.

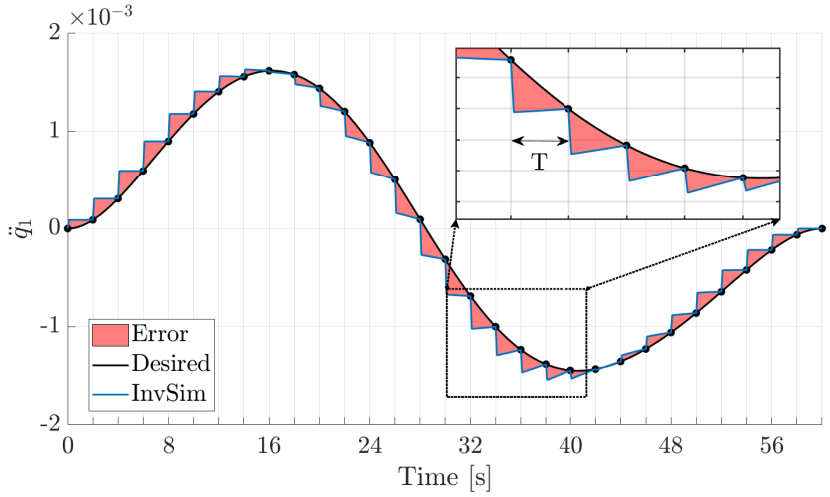


Figure 2.8: Acceleration ($\nu = 2$)

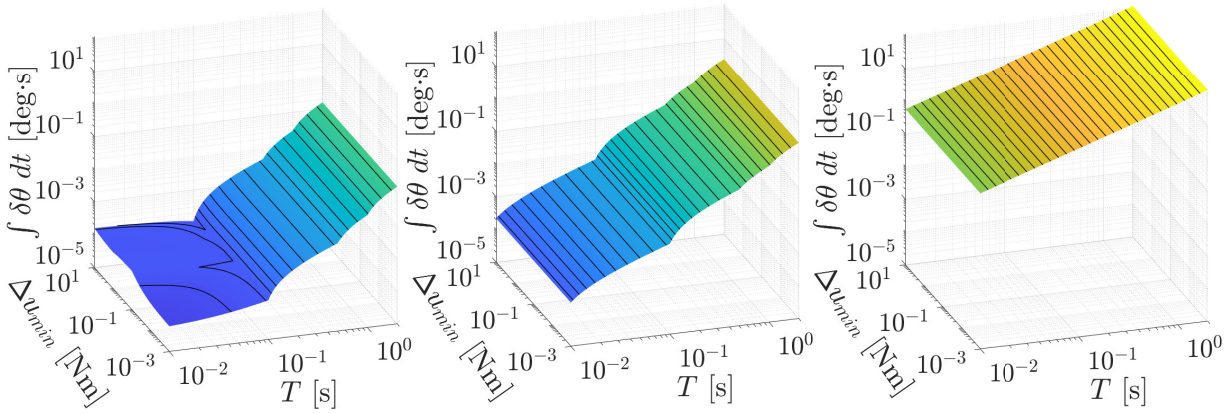
This error in the acceleration $\delta\ddot{q}_1$ will accumulate and integrate down to an error in the velocity $\delta\dot{q}_1$, as represented by the area between the curves marked red in Fig. 2.8. This error in the velocity $\delta\dot{q}_1$ will then further integrate down into an error in the quaternion position δq_1:

$$\delta q_1 = \int_0^{t_f} \delta\dot{q}_1\ dt = \int_0^{t_f} \int_0^{t_f} \delta\ddot{q}_1\ dt\ dt \tag{2.15}$$

It is therefore clear that as the order of output tracked by the InvSim ν increases, the more times any errors in $\dot{\boldsymbol{y}}^{(\nu)}$ will be integrated, leading to greater deviation from the

desired output position path $\boldsymbol{y}_d$. This effect can be mitigated somewhat through the reduction of the discretisation timestep T, which reduces the error accumulated in the InvSim down to values similar to those seen for lower ν (Fig. 2.7). Reducing T effectively gives less time between successive discretisation steps kT for the error to accumulate, shrinking the areas between the desired and InvSim curves in Fig. 2.8.

Through further consideration of Fig. 2.7 it can be seen that the relative integration step size $\Delta t/T$ has little to no impact on the accuracy of the InvSim when the desired output acceleration ($\nu = 2$) is used. However, as the desired output order ν is reduced, the effect of $\Delta t/T$ becomes more prevalent. In Fig. 2.7b it is clear that the effect of $\Delta t/T$ is limited for larger discretisation step sizes T, however as $T \to 10^{-2}$ s reduction in the relative integration step size $\Delta t/T$ begins to have a positive effect on accuracy. In the case where the desired output position ($\nu = 0$) is targeted directly, both discretisation step size T and relative integration step size $\Delta t/T$ have a significant effect, with the InvSim accuracy generally improving as both T and $\Delta t/T$ are reduced. However, for very small values of both T and $\Delta t/T$ the InvSim accuracy begins to deteriorate in the $\nu = 0$ case, most likely due to the accumulation of truncation errors in the integration process.

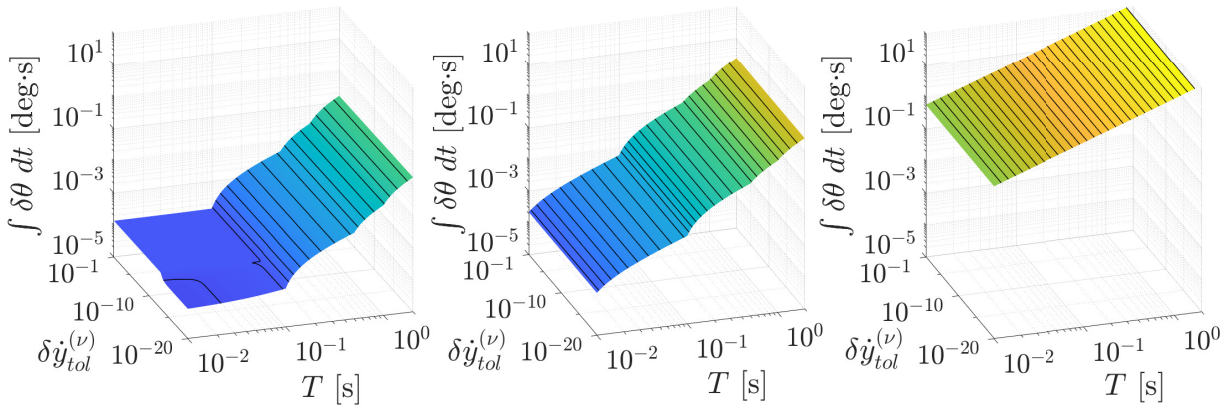


(a) $\nu = 0$ **(b)** $\nu = 1$ **(c)** $\nu = 2$

Figure 2.9: Effect of discretisation timestep T and NR minimum perturbation Δu_{min} on the attitude pointing error integrated across the entire maneouvre $\int_0^{t_f} \delta\theta\ dt$. Results shown when InvSim targets output position ($\nu = 0$), velocity ($\nu = 1$) and acceleration ($\nu = 2$)

The effect of the minimum finite difference perturbation Δu_{min} and discretiation timestep T on the InvSim accuracy is plotted in Fig. 2.9. Similar to the previous cases, the effect of ν and T are dominant, with the minimum finite difference perturbation Δu_{min} only having an effect when the desired output position ($\nu = 0$) is targeted and discretisation timestep $T < 10^{-1}$ s. Recalling from Section 2.1, the size of the finite difference perturbation Δu is, for the most part, taken by scaling the solution to the previous timestep $\Delta \boldsymbol{u} = T|\boldsymbol{u}[(k-1)T]|$. It can therefore be noted that the minimum finite

perturbation size Δu_{min} will only come into effect for the first timestep, where no previous input solution is available, or if $\Delta \boldsymbol{u} = T|\boldsymbol{u}[(k-1)T]| < \Delta u_{min}$. Therefore, it can be understood why Δu_{min} has minimal impact on the InvSim accuracy in most cases, as seen across Fig. 2.9. However, Δu_{min} was seen to have an impact on the desired output position ($\nu = 0$) case when $T < 10^{-1}$ s. In this region, it can be seen that the InvSim accuracy improves as Δu_{min} is reduced. Larger values of Δu_{min} will effectively override the scaling $\Delta \boldsymbol{u} = T|\boldsymbol{u}[(k-1)T]|$ when T is sufficiently small, reducing the accuracy of the linear approximation made during the finite differencing calculation of the Jacobian. The minimum finite difference perturbation Δu_{min} should therefore not be set too high as to cause these inaccuracies in the Jacobian. If set too low, however, the perturbed outputs will be similar, causing rounding errors in the numerical differencing process, which has been known to cause instabilities in the InvSim [125].



(a) $\nu = 0$ **(b)** $\nu = 1$ **(c)** $\nu = 2$

Figure 2.10: Effect of discretisation timestep T and NR tolerance $\dot{\delta y}_{tol}^{(\nu)}$ on the attitude pointing error integrated across the entire maneouvre $\int_0^{t_f} \delta\theta\ dt$. Results shown when InvSim targets output position ($\nu = 0$), velocity ($\nu = 1$) and acceleration ($\nu = 2$)

Finally, the effect of the NR tolerance $\dot{\delta y}_{tol}^{(\nu)}$ on the InvSim accuracy is investigated. Fig. 2.10 shows both the impact of the NR tolerance $\dot{\delta y}_{tol}^{(\nu)}$ and discretisation step size T. Once again, the effect of the desired output order ν and discretisation timestep T are dominant, with the NR tolerance $\dot{\delta y}_{tol}^{(\nu)}$ having less of an effect. Comparing Fig. 2.9a and Fig. 2.10a it can be seen that when $\nu = 0$ the minimum finite difference perturbation Δu_{min} has a greater impact on accuracy than NR tolerance $\dot{\delta y}_{tol}^{(\nu)}$.

The limited effect the NR tolerance $\dot{\delta y}_{tol}^{(\nu)}$ has on the InvSim accuracy can be explained by considering the behaviour of the dynamics being solved. The Jacobian used to update the input solution estimate $\boldsymbol{u}_n$ effectively linearises the input-output dynamics about the current state of the system $\boldsymbol{x}[kT]$. Therefore, the more linear the behaviour of the input-output dynamics, the more accurate an estimation of the system the cal-

culated Jacobian will be. While the rigid-body dynamics do include nonlinearities, namely the $\boldsymbol{\omega}_B \times \mathbf{I}_{tot}\boldsymbol{\omega}_B$ term, over a short discretisation timestep T, the dynamics will act almost linearly. Therefore, the calculated Jacobian is capable of achieving an input solution that meets a very stringent tolerance $\delta\dot{y}_{tol}^{(\nu)}$ in a single NR iteration, meaning that in many cases there will be no difference to the input solution converged upon should a larger or smaller $\delta\dot{y}_{tol}^{(\nu)}$ value be used.

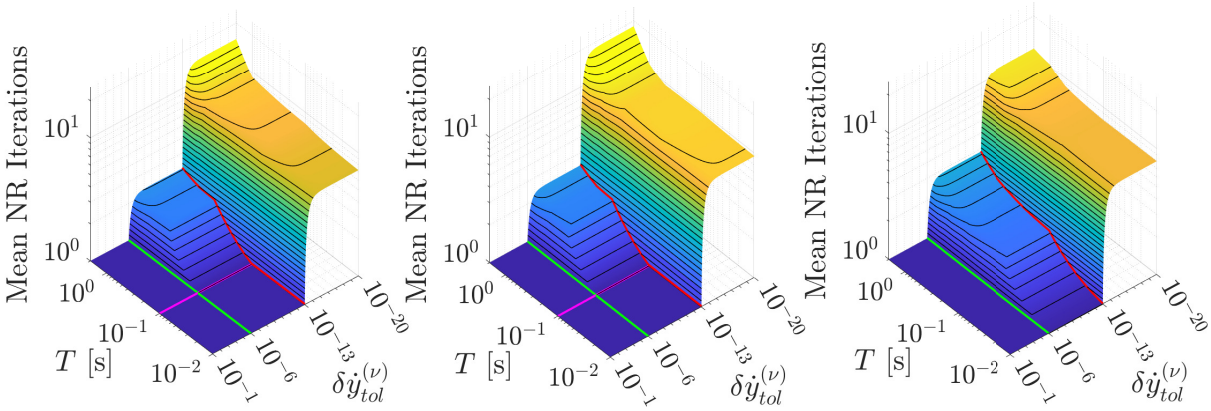
However, looking at Fig. 2.10a it can be seen that for small discretisation step sizes T, a decrease in the tolerance $\delta\dot{y}_{tol}^{(\nu)} < 1\times10^{-13}$ garners some improvement in InvSim accuracy. It can be understood that within this region, reducing the tolerance to values $\delta\dot{y}_{tol}^{(\nu)} < 1\times10^{-13}$ requires further NR iterations. This small improvement in accuracy comes with a significant increase in the number of NR iterations required, which will slow down the InvSim process significantly. Therefore, further investigation will be performed to ensure that the choice of NR tolerance and other hyperparameters gives a good compromise between speed and accuracy.

InvSim Speed

As mentioned, reduction of the NR tolerance $\delta\dot{y}_{tol}^{(\nu)}$ can lead to an increase in the number of NR iterations. This will be one of the primary contributors to the overall time complexity of the InvSim. Therefore, the effect of the NR tolerance $\delta\dot{y}_{tol}^{(\nu)}$ and discretisation timestep T on the mean number of NR iterations required to solve each timestep of the InvSim is shown in Fig. 2.11 for each desired output order case.

It was shown previously that in some cases reduction of the NR tolerance to $\delta\dot{y}_{tol}^{(\nu)} < 10^{-13}$ would lead to improved accuracy of the InvSim, as more NR iterations are used to converge upon the solution. This can be seen clearly illustrated in Fig. 2.11 by the red line denoting $\delta\dot{y}_{tol}^{(\nu)} = 1\times10^{-13}$. As the NR tolerance $\delta\dot{y}_{tol}^{(\nu)} \rightarrow 10^{-20}$ the mean number of NR iterations increases drastically. This drastic increase in mean NR iterations will lead to increased time complexity, which only comes with minimal improvements to the InvSim accuracy, as seen previously in Fig. 2.10. Careful selection of $\delta\dot{y}_{tol}^{(\nu)}$ should therefore take place to ensure that the NR loop does not perform needless additional iterations, while garnering very little improvement in accuracy. It is also important to note in Fig. 2.11 that the peak mean NR iterations shown are limited by the maximum NR iteration n_{max} hyperparameter. Therefore, reducing n_{max} could allow for mitigation of these needless NR iterations, reducing the sharp increase seen for $\delta\dot{y}_{tol}^{(\nu)} < 10^{-13}$.

Large regions of T and $\delta\dot{y}_{tol}^{(\nu)}$ show a flat plane corresponding to mean NR iterations of 1. This relates to the discussion previously, whereby the calculated Jacobian gives a sufficiently accurate representation of the input-output dynamics such that the in-



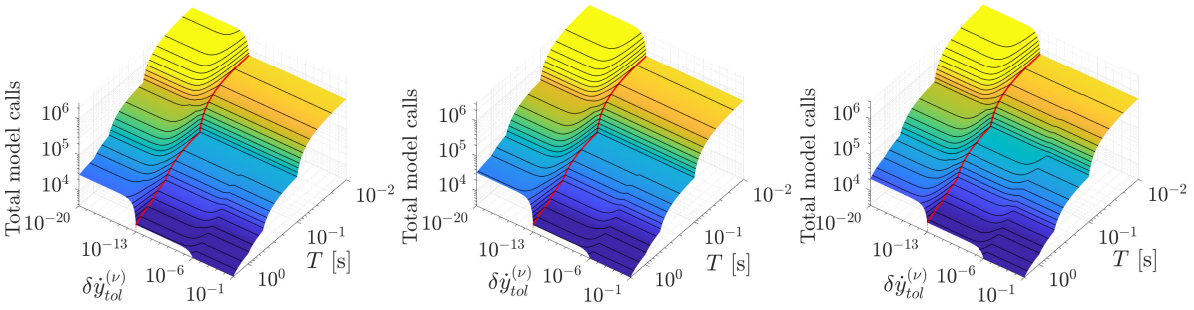
(a) $\nu = 0$ **(b)** $\nu = 1$ **(c)** $\nu = 2$

Figure 2.11: Effect of discretisation timestep T and NR tolerance $\delta\dot{y}_{tol}^{(\nu)}$ on the mean number of NR iterations per discretisation step executed during the InvSim. Results shown when InvSim targets output position ($\nu = 0$), velocity ($\nu = 1$) and acceleration ($\nu = 2$)

put solution can be converged upon within 1 NR iteration. It can also be seen across Fig. 2.11 that, in general, the mean number of NR iterations increases as the discretisation timestep T is increased. Considering the dynamics of the system again, it can be understood that as the discretisation timestep T increases, the dynamics have more time to diverge, making the linear approximation assumed by the calculated Jacobian less valid. Therefore, the more nonlinear input-output dynamics require additional NR iterations to solve. As mentioned previously, in the desired output acceleration case ($\nu = 2$), the input-output dynamics will appear more linear as the input torque has an instantaneous effect on the acceleration of the system. This therefore leads to the peak mean NR iterations for this case ($\nu = 2$) being lower than the position ($\nu = 0$) and velocity ($\nu = 1$) cases, as seen in Fig. 2.11c.

Looking at the region $10^{-6} < \delta\dot{y}_{tol}^{(\nu)} < 10^{-13}$ in Fig. 2.11a and Fig. 2.11b for the desired output position ($\nu = 0$) and velocity ($\nu = 1$) cases, the mean NR iterations start to increase significantly for $T > 0.1$ s. Within the same region of the desired output acceleration case ($\nu = 2$), the mean NR iterations increase for all values of T. Since for the most part, the finite difference perturbation is scaled $\Delta\boldsymbol{u} = T|\boldsymbol{u}[(k-1)T]|$, larger values of T will lead to larger perturbations which can degrade the accuracy of the Jacobian around the current estimate. The desired output acceleration ($\nu = 2$) will be more sensitive to the perturbation $\Delta\boldsymbol{u}$, as there is no time delay between a torque being applied and a change in the acceleration. In the velocity ($\nu = 1$) and position ($\nu = 0$) cases, it will take some finite time for the output to be significantly perturbed. While using output accelerations ($\nu = 2$) has been shown to improve the numerical stability of the InvSim [125], the linear approximation made by the associated Jacobian will become less accurate for short discretisation timesteps T, due to the higher sensitivity to

input perturbations. Degradation in the Jacobian accuracy inevitably leads to a greater number of NR iterations before convergence to a solution.



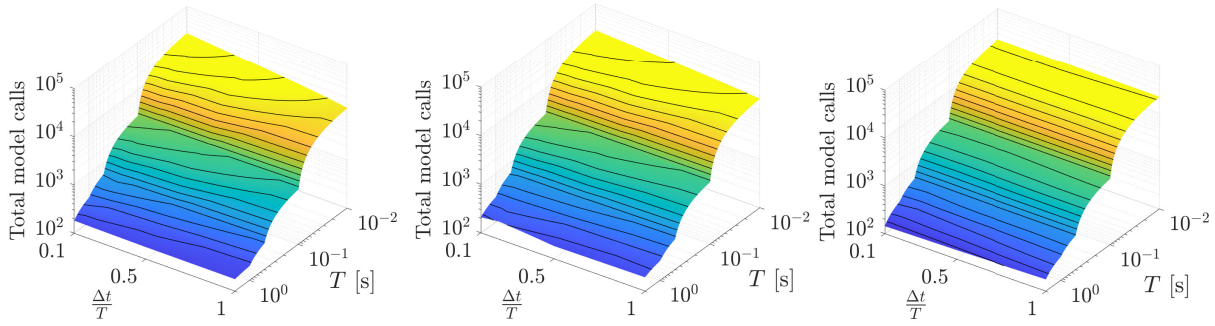
(a) $\nu = 0$ **(b)** $\nu = 1$ **(c)** $\nu = 2$

Figure 2.12: Effect of discretisation timestep T and NR tolerance $\delta\dot{\overset{(\nu)}{y}}_{tol}$ on the total number of model calls executed during the InvSim. Results shown when InvSim targets output position ($\nu = 0$), velocity ($\nu = 1$) and acceleration ($\nu = 2$)

While an increase in the number of NR iterations is a strong driving factor in the overall time complexity and speed of the InvSim, there are other factors at play. Here, the computational speed of the InvSim will be quantified as the total number of *model calls*, i.e. how many times the model is integrated forward over the integration timestep Δt. It can be assumed that one model call will take a fixed amount of time to complete, as the 4^{th} order Runge-Kutta (RK4) numerical integration method used will always evaluate the ODEs describing the dynamics four times per integration step. The number of times the model will be called/integrated will therefore depend on the discretisation timestep T, integration timestep Δt (or the relative integration step size $\Delta t/T$ in this case), and number of NR iterations.

Fig. 2.11 previously showed the effect of the discretisation timestep T and NR tolerance $\delta\dot{y}_{tol}^{(\nu)}$ on the mean number of NR iterations; their effect on the overall speed of the InvSim can also be shown in Fig. 2.12 in terms of total model calls. The red lines in Fig. 2.12 are lines of constant $\dot{y}_{tol}^{(\nu)} = 1\times10^{-13}$, seen previously in Fig. 2.11. The impact of the drastic increase in NR iterations caused by the NR tolerance being reduced to $\dot{y}_{tol}^{(\nu)} < 10^{-13}$ on the overall speed of the InvSim can therefore be seen. However, it was seen previously in Fig. 2.11 that the mean number of NR iterations increased as the discretisation timestep T was increased. Based on previous correlations between mean NR iterations and the overall speed of the InvSim, it would be fair to assume that an increase in the discretisation timestep T would slow the InvSim down; however, quite the opposite is the case. Increasing T reduces the total number of steps kT in the InvSim, therefore reducing the number of times the NR loop is required to converge to a solution. Despite an increase in the mean number of NR iterations, the increase in T increases the speed of the InvSim dramatically. This improvement in speed does

not come for free, with larger discretisation timesteps T reducing the overall accuracy of the InvSim, as seen previously in Figs. 2.7, 2.9 and 2.10. There is, therefore, a fine balance to be had in the choice of T to ensure good accuracy, but also suitable speed of InvSim. The speed required for the InvSim will be dependent on the application, with use as a real-time control solution requiring much faster computation than use as a design tool.



(a) $\nu = 0$ **(b)** $\nu = 1$ **(c)** $\nu = 2$

Figure 2.13: Effect of discretisation timestep T and relative integration step size $\Delta t/T$ on the total number of model calls executed during the InvSim. Results shown when InvSim targets output position ($\nu = 0$), velocity ($\nu = 1$) and acceleration ($\nu = 2$)

A similar trend can be seen with the effect the relative integration step size $\Delta t/T$ has on the speed of the InvSim in Fig. 2.13. Larger relative integration timesteps $\Delta t/T$ require fewer RK4 integrations, therefore improving overall speed. It is clear, however, that the discretisation timestep T is still by far the most influential hyperparameter affecting the InvSim speed. Additionally, while the choice of desired output order ν had a profound impact on accuracy, its influence on the overall speed of the InvSim was negligible (Figs. 2.12 and 2.13).

The hyperparameter study has revealed that the choice of discretisation timestep T is by far the most influential in terms of both accuracy, and speed of the InvSim. To achieve high accuracy in the InvSim the discretisation timestep T must be small; to achieve a fast InvSim, the discretisation timestep T needs to be large. Therefore, there is an engineering trade-off to be had, whereby the selection of T must balance both accuracy, and computational speed. Additionally, lower desired output orders ν achieved better accuracy with the same discretisation timestep T. It may therefore be tempting to set $\nu = 0$ to allow for T to be larger while still achieving good accuracy; however, as will be seen next, the stability of the InvSim also has to be taken into account.

2.2.3 | Numerical Stability

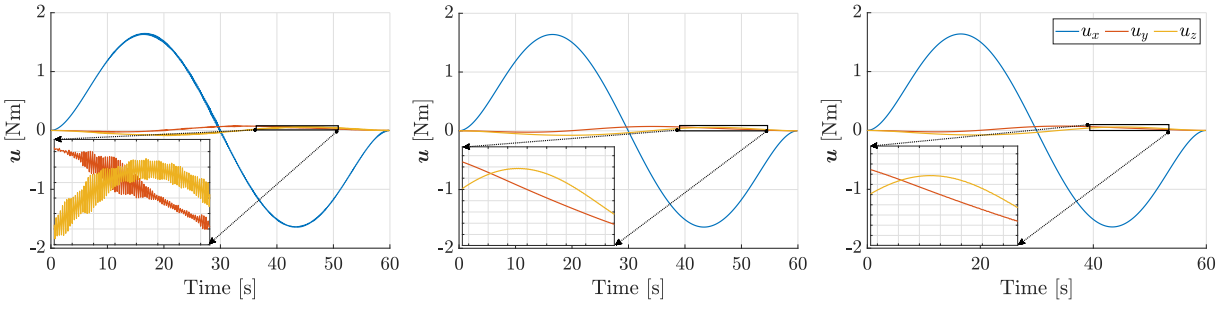
As discussed in Section 1.2.4, there are several issues of numerical stability associated with InvSim that have been studied previously in literature [128]. In this section, spe-

cific issues of numerical stability that have been encountered during the application of InvSim to satellite attitude dynamics will be presented. These include high-frequency, low-amplitude oscillations superimposed over the input solution due to inaccuracies in the calculated Jacobian, instability due to higher-order dynamics, and instability due to magnetic-only control constraints. These investigations will help choose hyperparameters not only to maximise the speed and accuracy of InvSim, but also to ensure high robustness to instabilities.

Jacobian Inaccuracies

As mentioned in this chapter, the rigid-body dynamics currently being adopted (Appendix E) can be accurately represented through the calculated Jacobian. Therefore, uniform random noise will be added to the Jacobian to replicate inaccuracies that could be found when InvSim is applied to more complex models, such as those seen later in this thesis. It has been established that increasing the NR tolerance $\delta\dot{y}_{tol}^{(\nu)}$ can slow down the overall speed of the InvSim, without necessarily improving accuracy significantly. Additionally, it has been seen that the best accuracy comes when utilising the position as the desired output ($\nu = 0$).

Applying these previous conclusions, the desired output order may be chosen as $\nu = 0$, NR tolerance chosen to be $\delta\dot{y}_{tol}^{(\nu)} = 10^{-3}$, with all other hyper-paramters chosen as seen in Table 2.4. The InvSim input solution for this choice of hyperparameters solving the same rest-to-rest slew manoeuvre used previously, for a rigid-body satellite with external torque inputs, can be found in Fig. 2.14a. While the solution at first glance looks similar to those seen previously in this chapter, looking closer, it can be seen that high-frequency, low-amplitude oscillations are superimposed on top of the input solution.



(a) $\nu = 0$, $\delta\dot{y}_{tol}^{(\nu)} = 10^{-3}$ **(b)** $\nu = 0$, $\delta\dot{y}_{tol}^{(\nu)} = 10^{-10}$ **(c)** $\nu = 1$, $\delta\dot{y}_{tol}^{(\nu)} = 10^{-3}$

Figure 2.14: Effect of Jacobian inaccuracies on InvSim input solution

Previously, in the literature, it has been suggested that these oscillations simply be removed through the use of a low-pass filter on the input solution timeseries [126], by

changing the discretisation timestep T [128], or reducing the NR tolerance $\delta\dot{y}_{tol}^{(\nu)}$ [130]. It is found that by reducing the NR tolerance to $\delta\dot{y}_{tol}^{(\nu)} = 10^{-10}$ the oscillations can be removed, as shown in Fig. 2.14b. It can be concluded that the inaccuracies in the Jacobian can cause the InvSim's input estimate $\boldsymbol{u}_n$ to oscillate around the *true* solution. When the NR tolerance $\delta\dot{y}_{tol}^{(\nu)}$ is loose, these deviated input solutions can produce errors $\delta\dot{y}^{(\nu)} < \delta\dot{y}_{tol}^{(\nu)}$ which will therefore be accepted by the InvSim. Decreasing the NR tolerance $\delta\dot{y}_{tol}^{(\nu)}$ forces the NR loop to perform additional iterations until the tolerance is met ($\delta\dot{y}^{(\nu)} < \delta\dot{y}_{tol}^{(\nu)}$), ensuring that the accepted input solution does not deviate significantly, thus reducing or removing the oscillations completely.

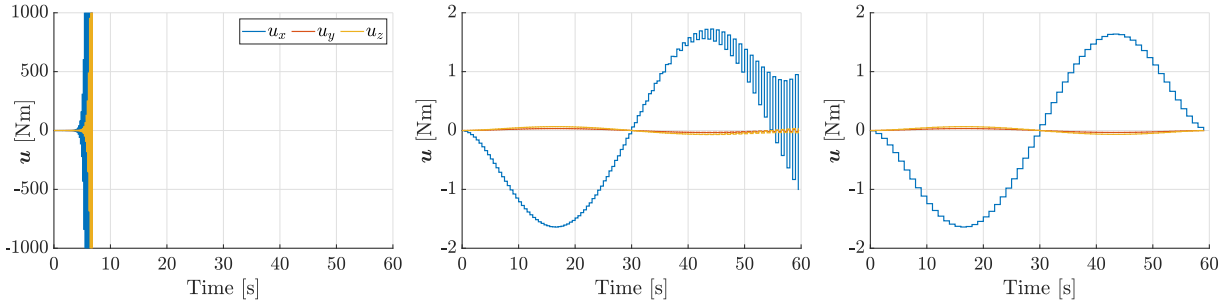
Additionally, as suggested by [125], these oscillations can be removed by using higher-order derivatives of the output ν. This is illustrated by adopting the looser NR tolerance $\delta\dot{y}_{tol}^{(\nu)} = 10^{-3}$ that produced the result in Fig. 2.14a, and targeting the desired output velocity ($\nu = 1$) instead of the position ($\nu = 0$). It is evident from Fig. 2.14c that the high-frequency, low-amplitude oscillations have successfully been removed. It can therefore be concluded that the use of higher-order outputs ν will improve the robustness of InvSim to high-frequency, low-amplitude oscillations for a wider range of NR tolerances $\delta\dot{y}_{tol}^{(\nu)}$.

Actuator Dynamics

It has also been found that the introduction of higher-order dynamics, such as the inclusion of a first-order reaction wheel torque response (Appendix E), can introduce instabilities in the InvSim. This behaviour is illustrated in Fig. 2.15 for a satellite with a first-order reaction wheel torque response, with time-constant $T_w = 0.01$ s (Table 2.2); shown is the InvSim input solution when the desired output position ($\nu = 0$) is targeted for varying sizes of discretisation timestep T. All other hyperparameters are again selected as shown in Table 2.4.

It can be seen in Fig. 2.15a that for the lowest discretisation timestep, $T = 0.1$ s, the InvSim input solution exhibits unstable oscillations that tend to infinity. The InvSim does not reach 10 s into the simulation before the inverse and pseudoinverse of the Jacobian become undefined. Increasing the discretisation timestep to $T = 0.5$ s allows for the InvSim to finish, however, divergent unstable oscillations still begin to appear at around 40 s (Fig. 2.15b). Further increase of the discretisation timestep to $T = 1.0$ s, as shown in Fig. 2.15c, leads to a completely stable InvSim solution.

The input solution in Fig. 2.15c is, however, very step-wise due to the large discretisation timestep T used, which may not be desirable for practical applications such as attitude control of satellites with flexible structures. As mentioned in Section 1.2.3,



(a) $T = 0.1$ s **(b)** $T = 0.5$ s **(c)** $T = 1.0$ s

Figure 2.15: Effect of actuator dynamics on InvSim input solution, $\nu = 0$

step-wise torques will induce theoretically infinite jerk on the satellite, which could cause unwanted excitation of the flexible modes. Therefore, an InvSim input solution that is more smooth and continuous is desirable.

The oscillations seen in Fig. 2.15a and Fig. 2.15b are a consequence of the relationship between the input $\boldsymbol{u}$, and desired output order of the InvSim ν. It was highlighted earlier that a torque input to a satellite with external torque inputs will have an instantaneous effect on the acceleration of the system ($\nu = 2$), while taking some finite amount of time to affect the velocity ($\nu = 1$) and position ($\nu = 0$). This effective delay due to the input and the output order used can cause rounding errors due to similar differencing in the calculation of the Jacobian if the discretisation timestep T is too low. The input is not being given enough time to propagate through the internal dynamics of the system, causing the desired output to not be significantly perturbed.

In the case of a satellite with a first-order reaction wheel torque response, this delay is further exacerbated as the order of the overall system dynamics is increased by one (Appendix E). The length of the delay between the input and the desired output will also, in this case, be further influenced by the time constant T_w of the reaction wheel torque response (Table 2.2). To compensate for the delay between the input $\boldsymbol{u}$ and the output position ($\nu = 0$), as used in Fig. 2.15, the discretisation timestep had to be increased to $T = 1.0$s to ensure stability. Instead, by increasing the desired output differential order ν, the delay between the input and desired output $\dot{y}^{(\nu)}$ can be reduced, allowing for a lower discretisation timestep T to be used. This is illustrated for a satellite with first-order reaction wheel torque response in Fig. 2.16, which uses the desired output velocity ($\nu = 1$) as the target for the InvSim, and a discretisation timestep of $T = 0.1$ s. It can therefore be seen again that using a higher output derivative order ν improves the robustness of the InvSim, in this case to higher-order dynamics.

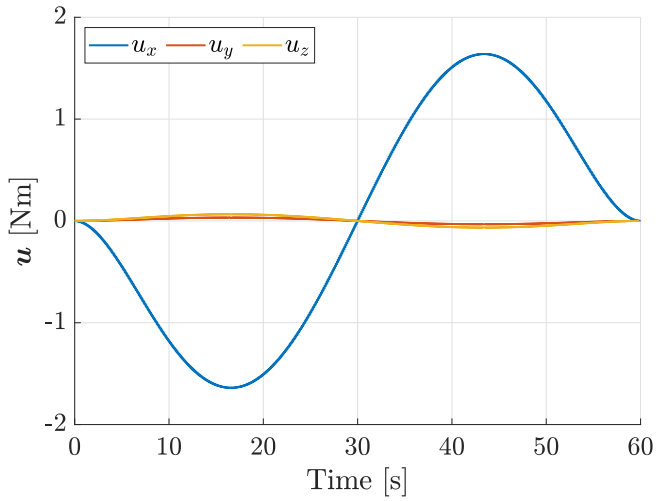


Figure 2.16: Effect of actuator dynamics on InvSim input solution, $\nu = 1$, $T = 0.1$ s

Magnetic-Only Control

The final stability issue considered in this section is that associated with using actuators, such as magnetorquers, which will implicitly be constrained by the direction of the magnetic field (Appendix E). The impact of this constraint on the stability of the InvSim can be illustrated by first considering the Jacobian utilising the quaternion parameterisation of the attitude:

$$\mathbf{J}_u = \begin{bmatrix} \frac{\partial \overset{(\nu)}{\dot{q}_1}}{\partial u_x} & \frac{\partial \overset{(\nu)}{\dot{q}_1}}{\partial u_y} & \frac{\partial \overset{(\nu)}{\dot{q}_3}}{\partial u_z} \\ \frac{\partial \overset{(\nu)}{\dot{q}_2}}{\partial u_x} & \frac{\partial \overset{(\nu)}{\dot{q}_2}}{\partial u_y} & \frac{\partial \overset{(\nu)}{\dot{q}_2}}{\partial u_z} \\ \frac{\partial \overset{(\nu)}{\dot{q}_3}}{\partial u_x} & \frac{\partial \overset{(\nu)}{\dot{q}_3}}{\partial u_y} & \frac{\partial \overset{(\nu)}{\dot{q}_3}}{\partial u_z} \end{bmatrix} \tag{2.16}$$

where $\boldsymbol{u} = [u_x \; u_y \; u_z]$ are the magnetic dipoles in each direction being controlled in the direction of the local magnetic field. If only magnetic control is used, this Jacobian will always have a null space. This null space, also known as the kernel, is the vector space (directions in $\boldsymbol{u}$) which, when multiplied by the Jacobian, will map to zero:

$$\mathbf{J}_u \hat{\boldsymbol{u}}_{null} = \mathbf{0}^{N_u \times 1} \tag{2.17}$$

i.e. given the system's current state, any vector in the direction $\hat{\boldsymbol{u}}_{null}$ will have no impact on the output. This can be clearly illustrated by considering the case where the magnetic field direction $\hat{\boldsymbol{b}}^B_{mag}$ aligns with the $\hat{z}^B$ axis of the body frame resulting in the associated column of u_z going to zero:

$$\mathbf{J}_u = \begin{bmatrix} \frac{\partial \overset{(\nu)}{\dot{q}_1}}{\partial u_x} & \frac{\partial \overset{(\nu)}{\dot{q}_1}}{\partial u_y} & 0 \\ \frac{\partial \overset{(\nu)}{\dot{q}_2}}{\partial u_x} & \frac{\partial \overset{(\nu)}{\dot{q}_2}}{\partial u_y} & 0 \\ \frac{\partial \overset{(\nu)}{\dot{q}_3}}{\partial u_x} & \frac{\partial \overset{(\nu)}{\dot{q}_3}}{\partial u_y} & 0 \end{bmatrix} \tag{2.18}$$

The inverse of a square Jacobian such as this would usually be calculated as:

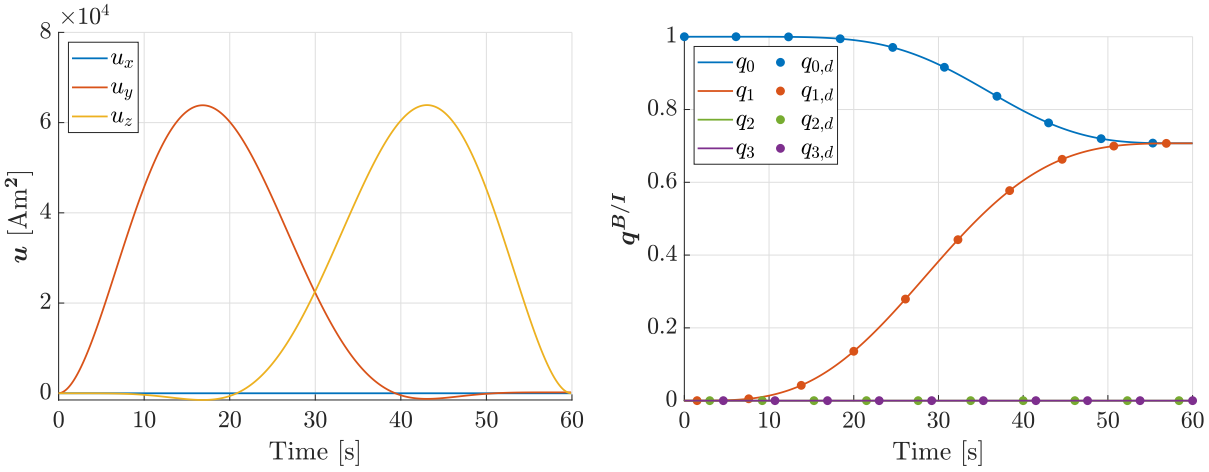
$$\mathbf{J}_u^{-1} = \frac{1}{\det(\mathbf{J}_u)} \text{adj}(\mathbf{J}_u) \tag{2.19}$$

where $\text{adj}(\cdot)$ is the adjugate matrix and $\det(\cdot)$ is the determinant. The determinant is clearly zero due to the Jacobian containing a full column of zeros, therefore, the Jacobian is said to be *singular*, and will not have a definable inverse as seen in Eq. (2.19). Instead, the MP pseudoinverse, as discussed in Section 2.1, can be used to solve the InvSim, already commonly used for the non-square Jacobians of redundant and under-actuated systems. This will work and give a suitable solution in the case where the attitude path does not require any torques outwith the plane defined by $\hat{\boldsymbol{b}}_{mag}^{B}$. However, if there are torques required outwith the plane, then there will be no physical solution, causing the InvSim's input estimate $\boldsymbol{u}_n$ to tend to infinity.

A good illustration of this is to consider a satellite tasked with a slew manoeuvre about the body $\hat{\boldsymbol{x}}^B$ axis with the magnetic field direction $\hat{\boldsymbol{b}}_{mag}^{B}$ orientated along the body $\hat{\boldsymbol{z}}^B$ axis for a non-diagonal and diagonal inertia matrix $\mathbf{I}_{tot}$. The non-diagonal inertia used is given previously in Table 2.2, with the diagonal inertia used here being $\mathbf{I}_{tot} = \text{diag}(500,\ 400,\ 350)\ \text{kgm}^2$.

In the case of the diagonal inertia matrix, the InvSim converges through the use of the MP pseudoinverse, as illustrated in Fig. 2.17, as there are no torques required in the direction of the magnetic field $\hat{\boldsymbol{b}}_{mag}^{B}$. Note that as the body of the satellite rotates, the direction of the magnetic field in the body frame $\hat{\boldsymbol{b}}_{mag}^{B}$ will change. This leads to the commanded magnetic dipole in Fig. 2.17 moving from the $\boldsymbol{y}^B$ to the $\boldsymbol{z}^B$ direction.

However, if the non-diagonal inertia matrix is used, the InvSim will fail to converge. This is caused by the cross-product term, $\boldsymbol{\omega}_B \times \mathbf{I}_{tot}\boldsymbol{\omega}_B$, in Euler's rotational equation (Appendix E), inducing additional body rates around the other body axes, as the satellite slews around $\hat{\boldsymbol{x}}^B$. When some components of these additional body rates align with the magnetic field direction $\hat{\boldsymbol{b}}_{mag}^{B}$ the satellite is unable to apply any torques to compensate.



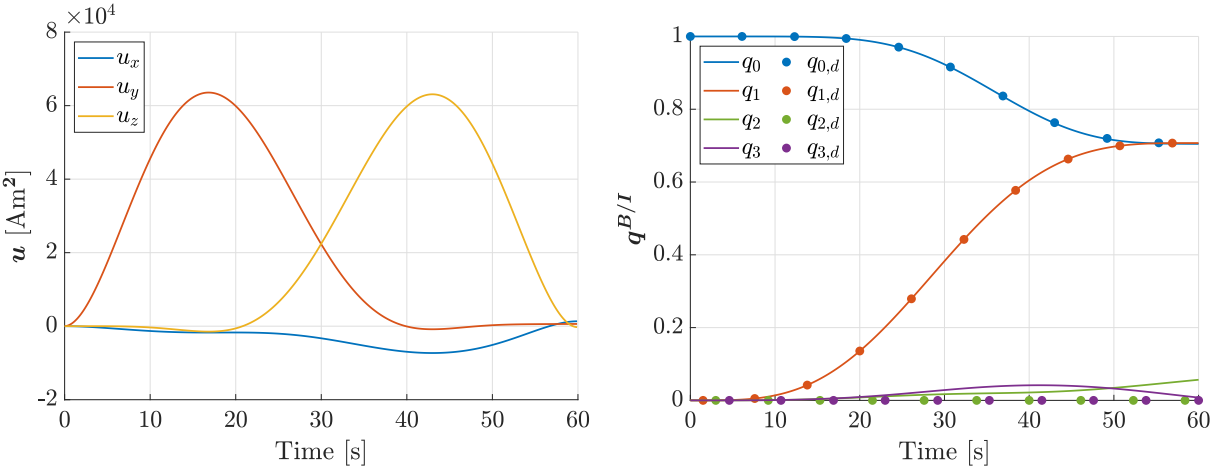
(a) Input magnetic dipole **(b)** Quaternion output

Figure 2.17: Magnetic-only control with diagonal inertia matrix and full attitude constrained $\overset{(v)}{\dot{\boldsymbol{y}}}_d = \overset{(v)}{\dot{\boldsymbol{q}}}^{B/I}$

Therefore, due to the strictness of the InvSim output constraints, the InvSim input estimate $\boldsymbol{u}_n$ tends to infinity. Given $\boldsymbol{q}^{B/I} = [q_0 \; q_1 \; q_2 \; q_3]^\top$, the constraints on the output of the system can be loosened to:

$$\overset{(v)}{\dot{y}}_d = \overset{(v)}{\dot{q}}_{d,1} \tag{2.20}$$

which will only constrain rotation about one axis. The InvSim results with these looser output constraints can be seen in Fig. 2.18. Fig. 2.18b shows that $q_{1,d}$ is tracked closely with $q_{2,d}$ and $q_{3,d}$ diverging as they are not constrained. It can also be understood that the cross-product $\boldsymbol{\omega}_B \times \mathbf{I}_{tot}\boldsymbol{\omega}_B$ induces torques that cannot be compensated for with magnetic-only control.



(a) Input magnetic dipole **(b)** Quaternion output

Figure 2.18: Magnetic-only control with non-diagonal inertia matrix and looser constraints $\overset{(v)}{\dot{y}}_d = \overset{(v)}{\dot{q}}_1^{B/I}$

The instability of the InvSim due to the constraints in the actuators, such as the example detailed above, can also be considered a feature, rather than an issue. Failure of the InvSim to converge identifies that the prescribed desired output trajectory is unrealisable by the system, which could be useful information during the design process of a satellite's attitude control system.

The application of InvSim to the attitude dynamics of satellites with various actuator types has now been established. Analysis has been performed to allow for a good selection of hyperparameters that maximise both the accuracy and speed of the numerical process. Instabilities in the method, some of which will be revisited later in this thesis, have been investigated to ensure the InvSim remains robust. This is particularly important to guarantee that any failure of the InvSim to converge is due to the prescribed output trajectory not being physically possible, and not due to the InvSim numerical process itself. This will become important when the InvSim is applied as a design tool in a later chapter.

Thus far, the attitude kinematics of satellites performing their prescribed manoeuvres has been described using unit quaternions (and their derivatives). However, in some cases, the fully constrained orientations provided by a unit quaternion may not be the most convenient for some mission profiles. In this case, alternative parameterisations of the satellite's kinematics will be required, which, as will be shown in the next section, InvSim is highly flexible in accommodating.

2.3 | Flexibility of Inverse Simulation

It has been seen that InvSim was capable of solving the inverse dynamics when different differential orders of the output ν were used. Further, it was shown that InvSim could find a solution when the attitude is fully constrained or only partially constrained. So far, however, the use of different constraints has only been used to alleviate the numerical stability issues associated with magnetic attitude control. Here, the use of different attitude constraints and parameterisations will be investigated further to illustrate the flexibility of InvSim. Additionally, different model parameters, as seen in Table 2.6, are chosen to resemble a 3U CubeSat with reaction wheels. This further illustrates the flexibility of InvSim, which can easily solve the inverse dynamics for a variety of satellite architectures and configurations.

Whilst quaternions tend to be the attitude parameterisation of choice, due to being singularity-free, Euler angles are still widely used thanks to their more tractable and intuitive description. The use of two Euler angles can fully define the pointing of a single axis whilst leaving rotation about that pointing axis free and unconstrained. It

Table 2.6: Rigid-body CubeSat parameters

Parameter	Value
Inertia, $\mathbf{I}_{tot}$	$\begin{bmatrix} 0.01 & 0 & 0 \\ 0 & 0.05 & 0 \\ 0 & 0 & 0.05 \end{bmatrix}$ kgm^2
Reaction wheel distribution matrix, $\mathbf{A}_w$	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$
Reaction wheel time constant, T_w	0.1 s

can also be easier for an engineer to define *keep-out* zones with Euler-Angles, ensuring that sensitive sensors and equipment are not pointed at the sun [164]. Quaternions can still be employed as the attitude kinematic representation during the numerical integration process, simply applying a transformation into the Euler angle set of choice for the InvSim to target.

2.3.1 | Euler Angle Paths

Therefore, a suitable attitude path-planning algorithm needs to be implemented to find an Euler angle path that is both realisable by the satellite's dynamics and conforms to the constraints defined by any prescribed *keep-out* zones. The problem of attitude path-planning will, for the most part, not be considered in this thesis. Instead, to demonstrate the use of InvSim to solve more complex attitude trajectories, such as those that would be produced by a suitable path-planning algorithm, polynomial rest-to-rest manoeuvres with intermediate waypoints can be defined in Euler angle space.

Considering a series of n_{wps} number of arbitrary Euler angle waypoints:

$$\boldsymbol{\theta}_{wps} = \begin{bmatrix} \theta_0 & \theta_1 & \cdots & \theta_i & \cdots & \theta_{n_{wps}} \end{bmatrix} \tag{2.21}$$

which need to be reached at times:

$$\boldsymbol{t}_{wps} = \begin{bmatrix} t_0 & t_1 & \cdots & t_i & \cdots & t_{n_{wps}} \end{bmatrix} \tag{2.22}$$

These can be traversed using a set of $n_{wps} - 1$ piecewise polynomials with position, velocity and acceleration continuity to ensure a smooth, continuous path for the InvSim to solve. The position, velocity and acceleration profiles of the i$^{\text{th}}$, 4$^{\text{th}}$-order piecewise polynomial is given by:

$$\theta_i(\tau) = a_i\tau^4 + b_i\tau^3 + c_i\tau^2 + d_i\tau + e_i \tag{2.23a}$$

$$\dot{\theta}_i(\tau) = 4a_i\tau^3 + 3b_i\tau^2 + 2c_i\tau + d_i \tag{2.23b}$$

$$\ddot{\theta}_i(\tau) = 12a_i\tau^2 + 6b_i\tau + 2c_i \tag{2.23c}$$

$$\dddot{\theta}_i(\tau) = 24a_i\tau^2 + 6b_i \tag{2.23d}$$

where $\tau \in [0,\ t_i - t_{i-1}]$ for $\tau \in \mathbb{R}$, and $i \in [1, n_{wps} - 1]$ for $i \in \mathbb{Z}$, where $\mathbb{Z}$ is the set of all integers. It may also be useful to consider the final time of each piecewise polynomial as $\tau_{i,f} = t_i - t_{i-1}$. Using τ instead of the *absolute* time t allows for the coefficient e_i for each polynomial to be given simply by the position continuity requirement:

$$e_i = \theta_{i-1} \tag{2.24}$$

Similarly, since the satellite will always start at rest, the velocity and acceleration at the start of the first polynomial will be zero. Therefore using Eqs. (2.23b) and (2.23c) the coefficients c_1 and d_1 from the first polynomial are constrained:

$$c_1 = d_1 = 0 \tag{2.25}$$

The remaining coefficients can be solved by considering velocity and acceleration continuity and forming a system of simultaneous equations. Additionally, equations for the continuity of the *jerk* are included. While no solution will exist with this constraint, its inclusion will allow for convergence to the solution that minimises the discontinuity in *jerk* (Eq. (2.23d)). An example of the system of simultaneous equations for a three-waypoint path is given:

Ensures continuity between $\ddot{\theta}_1$ and $\ddot{\theta}_2$

Ensures continuity between $\dot{\theta}_1$ and $\dot{\theta}_2$

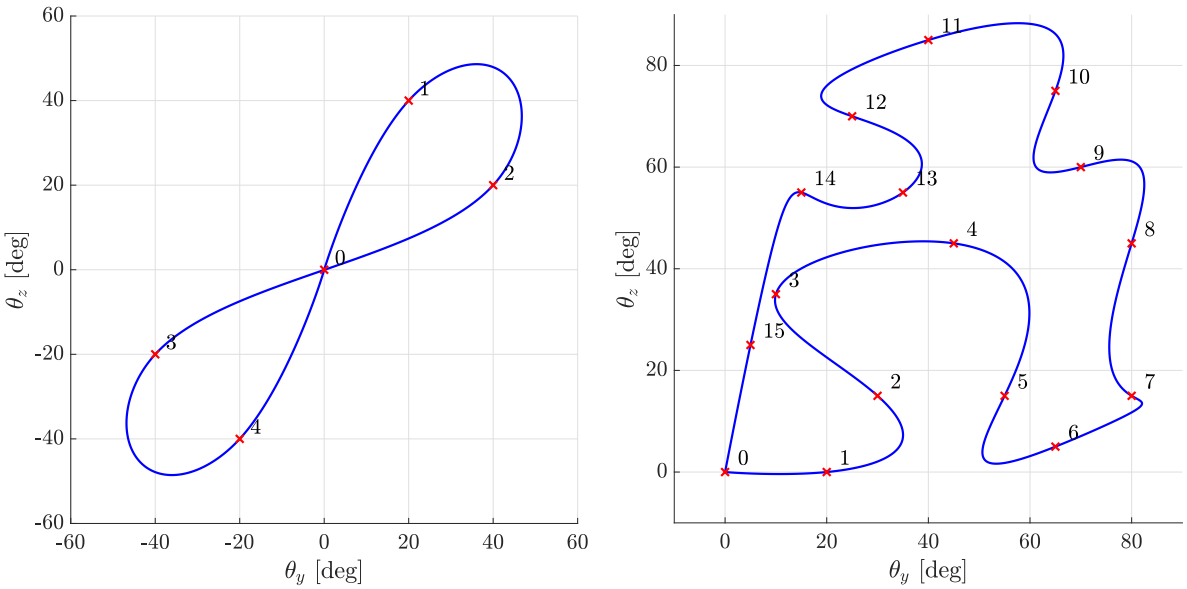
$$\begin{array}{r} \text{Position constraint} \\ \text{Velocity continuity} \\ \text{Acceleration continuity} \\ \text{Minimise jerk discontinuity} \\ \text{Position constraint} \\ \text{Final velocity} \\ \text{Final acceleration} \end{array} \overbrace{\begin{bmatrix} \tau_{1,f}^4 & \tau_{1,f}^3 & 0 & 0 & 0 & 0 \\ 4\tau_{1,f}^3 & 3\tau_{1,f}^2 & 0 & 0 & 0 & -1 \\ 12\tau_{1,f}^2 & 6\tau_{1,f} & 0 & 0 & -2 & 0 \\ 24\tau_{1,f} & 6 & 0 & -6 & 0 & 0 \\ 0 & 0 & \tau_{2,f}^4 & \tau_{2,f}^3 & \tau_{2,f}^2 & \tau_{2,f}^3 \\ 0 & 0 & 4\tau_{2,f}^3 & 3\tau_{2,f}^2 & 2\tau_{2,f}^1 & 1 \\ 0 & 0 & 12\tau_{2,f}^4 & 6\tau_{2,f}^3 & 2 & 0 \end{bmatrix}}^{\mathbf{A}} \begin{bmatrix} a_1 \\ b_1 \\ a_2 \\ b_2 \\ c_2 \\ d_2 \end{bmatrix}^{\boldsymbol{x}} = \overbrace{\begin{bmatrix} \theta_1 - \theta_0 \\ 0 \\ 0 \\ 0 \\ \theta_2 - \theta_1 \\ 0 \\ 0 \end{bmatrix}}^{\boldsymbol{y}}$$

Minimises discontinuity between $\dddot{\theta}_1$ and $\dddot{\theta}_2$

$$\mathbf{A}\boldsymbol{x} = \boldsymbol{y}$$

(2.26)

As can be seen, this linear system of equations is in the form $\mathbf{A}\boldsymbol{x} = \boldsymbol{y}$, which can be solved using a MP pseudoinverse, as described in Appendix A.4. Each piecewise polynomial can be sampled, and then appended together to construct a continuous time-series of the full path. This is repeated for each Euler Angle included in the desired output constrained by the InvSim. Two example attitude paths defined in the $\theta_y - \theta_z$ plane are shown in Fig. 2.19 with each waypoint numbered and marked as a red "x". The difference in the Euler angles from one waypoint to the next will populate the vector $\boldsymbol{y}$ in Eq. (2.26), which can be used to solve the full piecewise path as shown in Fig. 2.19.



(a) Figure of 8 path

(b) Long path

Figure 2.19: Piecewise polynomial paths defined in $\theta_y - \theta_z$ plane

2.3.2 | Different Output Constraints

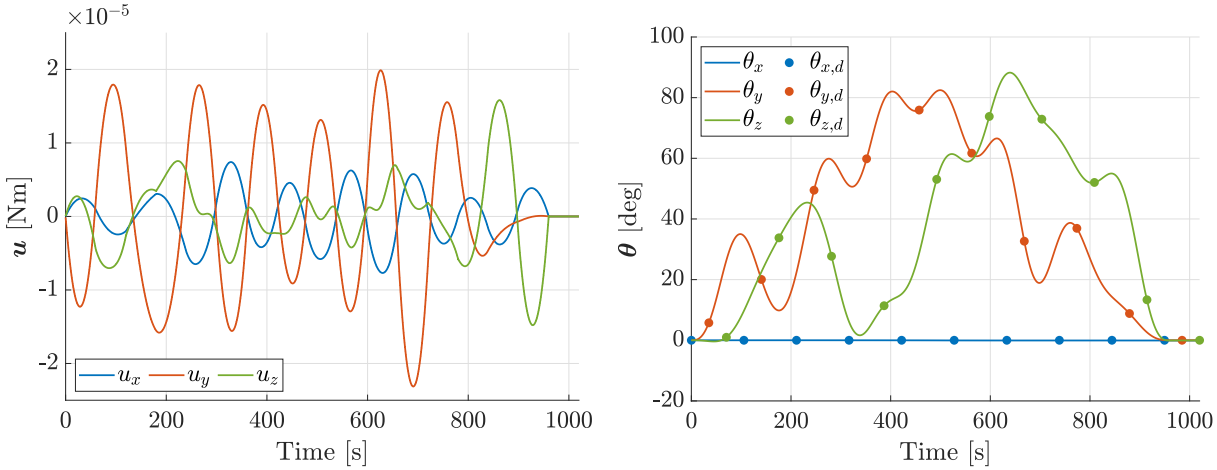
The *long path*, shown in Fig. 2.19b, will now be used as a test case to illustrate the flexibility of the InvSim. Most of the InvSim hyperparameters detailed in Table 2.4 are used here, with the only difference being the desired output constraint used.

Two different sets of desired output constraints will be considered, the first simply constraining all three Euler angles. With the desired output derivative order set to $\nu = 1$, as seen in Table 2.4, the desired output targeted by the InvSim will be:

$$\overset{(\nu)}{\boldsymbol{y}}_d = \begin{bmatrix} \dot{\theta}_x & \dot{\theta}_y & \dot{\theta}_z \end{bmatrix}^\top \tag{2.27}$$

where the Euler angle θ_x and its associated derivative $\dot{\theta}_x$ are set to zero, with the attitude path defined in the $\theta_y - \theta_z$ (Fig. 2.19b). The InvSim result for this set of desired

output constraints can be seen in Fig. 2.20. The attitude slew path is followed precisely, with the CubeSat's attitude passing through each of the defined waypoints. The input solution converged upon by the InvSim is smooth, and free of any high-frequency oscillations.



(a) Input torque

(b) Euler angle output

Figure 2.20: Euler angles attitude-tracking for a non-spinning CubeSat

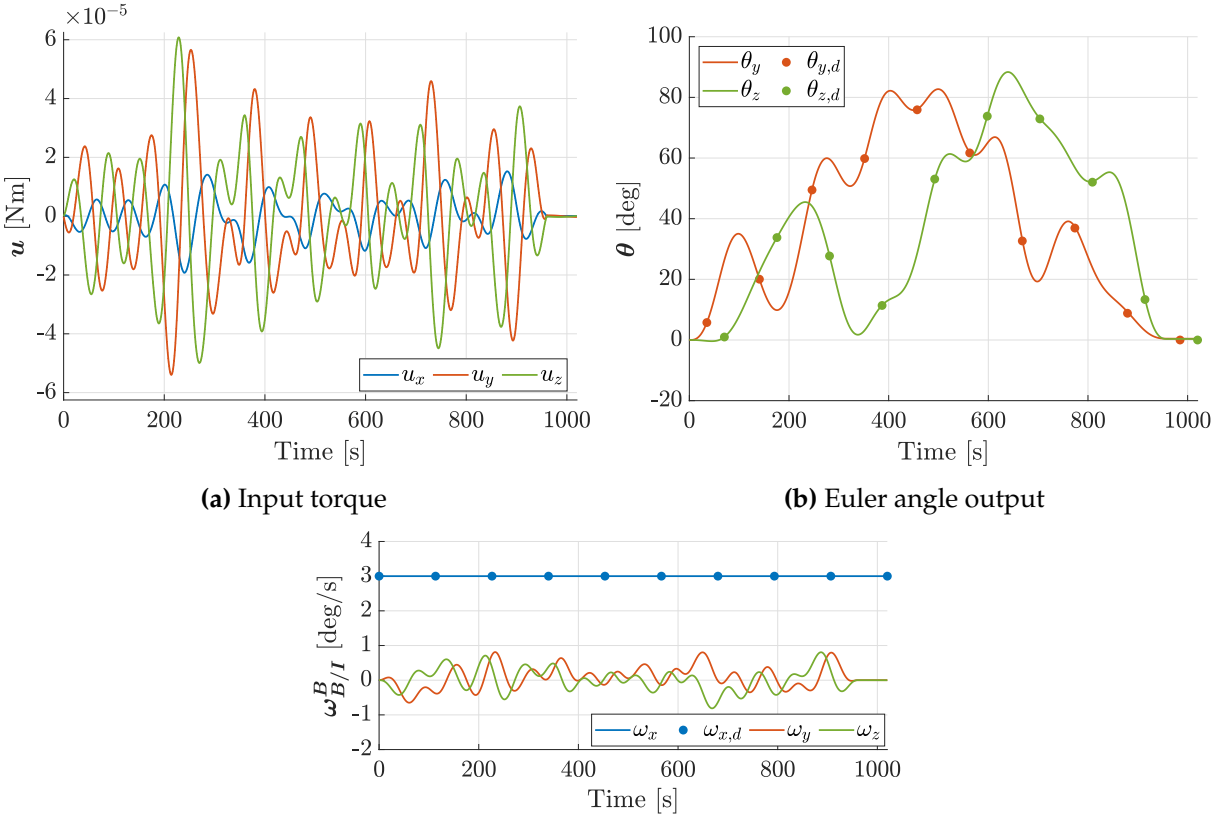
The second set of desired output constraints are chosen as:

$$\overset{(\nu)}{\boldsymbol{y}}_d = \begin{bmatrix} \omega_x & \dot{\theta}_y & \dot{\theta}_z \end{bmatrix}^\top \tag{2.28}$$

where ω_x is the body rate about the body $\hat{\boldsymbol{x}}^B$ axis. Depending on the order of the Euler angle set this body rate may be equal to an Euler angle rate, however, this will not always be the case. Spin-axis stabilisation is a common passive attitude control approach that utilises the conservation of angular momentum to resist disturbances. The set of desired output constraints given above defines the reorientation of a spin-stabilised $\hat{\boldsymbol{x}}^B$ axis, without the need to de-spin. A constant spin rate about the $\hat{\boldsymbol{x}}^B$ axis of 3 deg/s is chosen.

The InvSim solution for the spinning CubeSat is given in Fig. 2.21. Again, the input solution converged upon by the InvSim is smooth and continuous. The attitude path defined on the $\theta_y - \theta_z$ plane is followed, and the constant spin rate about the $\hat{\boldsymbol{x}}^B$ axis is maintained.

It can therefore be seen that the InvSim approach to solving the inverse dynamics of satellites is highly flexible. The desired output constraint parameterisations used to drive the InvSim solution can be mixed and matched easily to facilitate the simulation of any attitude-pointing scenario. The model parameters can easily be changed to rep-



(a) Input torque

(b) Euler angle output

(c) Body rate output

Figure 2.21: Euler angles attitude-tracking for a spinning CubeSat

resent a new satellite design or architecture with no changes needing to be made to the InvSim process or hyperparameters.

Whilst the applicability and flexibility of using InvSim to solve the inverse attitude dynamics of satellites has been demonstrated, it is still unclear what benefits this numerical, computationally intensive approach would have over other traditional methods. The next section of this chapter will therefore investigate the practicality of using InvSim over analytical model-inversion techniques that are commonly utilised in the development of attitude control architectures.

2.4 | Practicality of Inverse Simulation

An analysis will now be performed to compare the practical benefits of InvSim and other dynamic inversion approaches, with particular attention paid to the labour and effort required to achieve the inverse dynamic solution. It will be shown that other inversion techniques, specifically analytical or semi-analytical approaches, can suffer when significant changes are made to the underlying mathematical model. A major alteration to the model may require the inverse solution to be completely or partially

re-derived, which could be both tedious and time-consuming. It will be shown instead that InvSim can provide a *ready-to-go* inverse solution, solving a modified or completely new model without any modifications to the solution process. To illustrate this point, a comparison will be drawn using analytical and semi-analytical inverse dynamic solutions.

2.4.1 | Nonlinear Dynamic Inversion (NDI)

An approach commonly known as nonlinear dynamic inversion (NDI) provides a suitable methodology for finding an analytical or semi-analytical solution to the inverse dynamics of a system [158].

External Torques

First, consider the dynamics (Eq. (E.5)) and quaternion kinematics (Eq. (B.18)) of a rigid-body satellite with an external torque $\boldsymbol{u}$ in nonlinear state space form as given in Appendices B.3 and E.1:

$$\begin{aligned} \dot{\boldsymbol{x}} &= \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}) \\ \begin{bmatrix} \dot{\boldsymbol{\omega}}_B \\ \dot{\boldsymbol{q}}^{B/I} \end{bmatrix} &= \begin{bmatrix} \mathbf{I}_{hub}^{-1} \left(\boldsymbol{u} - [\boldsymbol{\omega}_B]_\times \mathbf{I}_{hub} \boldsymbol{\omega}_B \right) \\ \frac{1}{2} \boldsymbol{\Xi}(\boldsymbol{q}^{B/I})\, \boldsymbol{\omega}_B \end{bmatrix} \end{aligned} \tag{2.29}$$

where the matrix $\boldsymbol{\Xi}(\cdot)$ is defined in Appendix B.3 (Eq. (B.18)). To solve the inverse dynamics using NDI, the output of the system needs to be chosen. For the case of attitude-tracking, this will simply be the attitude quaternion:

$$\boldsymbol{y} = \boldsymbol{g}(\boldsymbol{x}) = \boldsymbol{q}^{B/I} \tag{2.30}$$

The inverse dynamics, i.e. the mapping of outputs $\boldsymbol{y}$ to inputs $\boldsymbol{u}$ can then be found by successively taking the time derivative of the output, and substituting in the system dynamics/kinematics until the input appears. The inverse solution can then be found through algebraic manipulation to make $\boldsymbol{u}$ the subject of the equation. The number of times that the output needs to be differentiated before the input appears will be the *relative degree* of the system. For the rigid-body satellite being considered here, the output can be differentiated:

$$\dot{\boldsymbol{y}} = \dot{\boldsymbol{q}}^{B/I} = \frac{1}{2} \boldsymbol{\Xi}(\boldsymbol{q}^{B/I}) \boldsymbol{\omega}_B \tag{2.31}$$

and then for a second time:

$$\begin{aligned}\ddot{\boldsymbol{y}} &= \frac{1}{2}\dot{\boldsymbol{\Xi}}(\dot{\boldsymbol{q}}^{B/I})\boldsymbol{\omega}_B + \frac{1}{2}\boldsymbol{\Xi}(\boldsymbol{q}^{B/I})\dot{\boldsymbol{\omega}}_B \\ &= \frac{1}{2}\dot{\boldsymbol{\Xi}}(\dot{\boldsymbol{y}})\boldsymbol{\omega}_B + \frac{1}{2}\boldsymbol{\Xi}(\boldsymbol{y})\left[\mathbf{I}_{hub}^{-1}\left(-\left[\boldsymbol{\omega}_B\right]_\times \mathbf{I}_{hub}\,\boldsymbol{\omega}_B + \boldsymbol{u}\right)\right]\end{aligned} \tag{2.32}$$

where, after substituting in the system dynamics and kinematics, the input $\boldsymbol{u}$ appears. Rearranging for the input $\boldsymbol{u}$:

$$\boldsymbol{u} = \mathbf{I}_{hub}\,\boldsymbol{\Xi}^\top(\boldsymbol{y})\left[2\ddot{\boldsymbol{y}} - \dot{\boldsymbol{\Xi}}(\dot{\boldsymbol{y}})\boldsymbol{\omega}_B\right] + \left[\boldsymbol{\omega}_B\right]_\times \mathbf{I}_{hub}\,\boldsymbol{\omega}_B \tag{2.33}$$

where the inverse of $\boldsymbol{\Xi}(\cdot)$ will simply be the transpose $\boldsymbol{\Xi}^\top(\cdot)$ as is evident from the inverse quaternion kinematics definition found in Appendix B.3 (Eq. (B.19)). In many cases, when using NDI, it is found that some states of the system on the right-hand side of Eq. (2.33) are not analytically solvable. Instead, Eq. (2.33) would need to be solved iteratively with the unknown states found through numerical integration of the mathematical model. However, the only unsolved state in Eq. (2.33) is $\boldsymbol{\omega}_B$, which can be found from applying inverse kinematics (Eq. (B.19)) to the derivative of the output (Eq. (2.31)). Thus, a fully closed-form solution of the inverse dynamics of Eq. (2.29) can be found:

$$\boldsymbol{u} = \mathbf{I}_{hub}\,\boldsymbol{\Xi}^\top(\boldsymbol{y})\left[2\ddot{\boldsymbol{y}} - 2\dot{\boldsymbol{\Xi}}(\dot{\boldsymbol{y}})\boldsymbol{\Xi}^\top(\boldsymbol{y})\dot{\boldsymbol{y}}\right] + 4\left[\boldsymbol{\Xi}^\top(\boldsymbol{y})\dot{\boldsymbol{y}}\right]_\times \mathbf{I}_{hub}\,\boldsymbol{\Xi}^\top(\boldsymbol{y})\dot{\boldsymbol{y}} \tag{2.34}$$

Magnetic Torques

A similar closed-form solution can be found when considering a rigid-body satellite with magnetorquers as given in Appendix E.2 (Eq. (E.7)). Since magnetic torques act externally on the satellite body, as can be seen by comparing the models in Appendix E, the NDI technique will simply lead to slight variation on Eq. (2.34):

$$\left[\boldsymbol{b}_{mag}^B\right]_\times^\top \boldsymbol{u} = \mathbf{I}_{hub}\,\boldsymbol{\Xi}^\top(\boldsymbol{y})\left[2\ddot{\boldsymbol{y}} - 2\dot{\boldsymbol{\Xi}}(\dot{\boldsymbol{y}})\boldsymbol{\Xi}^\top(\boldsymbol{y})\dot{\boldsymbol{y}}\right] + 4\left[\boldsymbol{\Xi}^\top(\boldsymbol{y})\dot{\boldsymbol{y}}\right]_\times \mathbf{I}_{hub}\,\boldsymbol{\Xi}^\top(\boldsymbol{y})\dot{\boldsymbol{y}} \tag{2.35}$$

A least-squares fit solution can be found for the input $\boldsymbol{u}$ using the MP pseudoinverse $[\cdot]^+$ (Eq. (A.19)) as there will exist no direct inverse of the skew-symmetric cross product matrix:

$$\boldsymbol{u} = -\left[\boldsymbol{b}_{mag}^B\right]_\times^+ \left(\mathbf{I}_{hub}\,\boldsymbol{\Xi}^\top(\boldsymbol{y})\left[2\ddot{\boldsymbol{y}} - 2\dot{\boldsymbol{\Xi}}(\dot{\boldsymbol{y}})\boldsymbol{\Xi}^\top(\boldsymbol{y})\dot{\boldsymbol{y}}\right] + 4\left[\boldsymbol{\Xi}^\top(\boldsymbol{y})\dot{\boldsymbol{y}}\right]_\times \mathbf{I}_{hub}\,\boldsymbol{\Xi}^\top(\boldsymbol{y})\dot{\boldsymbol{y}}\right) \tag{2.36}$$

The magnetic field vector in body axes ($\boldsymbol{b}_{mag}^B$) also still needs to be solved as this will not be constant, varying with the attitude of the satellite. It can be found by considering the

rotation of the magnetic field vector from body to inertial axes using a direction cosine matrix (Eq. (B.8)):

$$\boldsymbol{u} = -\left[\mathbf{C}^{B/I}(\boldsymbol{y})\boldsymbol{b}^{I}_{mag}\right]^{+}_{\times}\left(\mathbf{I}_{hub}\,\boldsymbol{\Xi}^{\top}(\boldsymbol{y})\left[2\ddot{\boldsymbol{y}} - 2\dot{\boldsymbol{\Xi}}(\dot{\boldsymbol{y}})\boldsymbol{\Xi}^{\top}(\boldsymbol{y})\dot{\boldsymbol{y}}\right] + 4\left[\boldsymbol{\Xi}^{\top}(\boldsymbol{y})\dot{\boldsymbol{y}}\right]_{\times}\mathbf{I}_{hub}\,\boldsymbol{\Xi}^{\top}(\boldsymbol{y})\dot{\boldsymbol{y}}\right) \tag{2.37}$$

Therefore, the input is fully defined in terms of the output and its derivatives. As was seen by the necessity of having to use a pseudoinverse for the skew-symmetric matrix, the NDI solution suffers the same input constraint issue seen with InvSim, as discussed in Section 2.2.3.

Reaction Wheels

Closed-form solutions were possible through NDI for a rigid-body satellite with external torque and magnetorquer inputs. The derivation when reaction wheels are introduced requires more substantial re-derivation. Starting again with the system dynamics and kinematics in nonlinear state space form (Appendix E):

$$\begin{aligned}\dot{\boldsymbol{x}} &= \boldsymbol{f}(\boldsymbol{x},\boldsymbol{u})\\ \begin{bmatrix}\dot{\boldsymbol{\omega}}_B\\ \dot{\boldsymbol{q}}^{B/I}\\ \dot{\boldsymbol{h}}_w\end{bmatrix} &= \begin{bmatrix}\mathbf{I}_{tot}^{-1}\left[-\mathbf{A}_w\boldsymbol{u} - [\boldsymbol{\omega}_B]_\times(\mathbf{I}_{tot}\boldsymbol{\omega}_B + \boldsymbol{h}_w)\right]\\ \frac{1}{2}\boldsymbol{\Xi}(\boldsymbol{q}^{B/I})\,\boldsymbol{\omega}_B\\ \mathbf{A}_w\boldsymbol{u}\end{bmatrix}\end{aligned} \tag{2.38}$$

Differentiating the output $\boldsymbol{y} = \boldsymbol{q}^{B/I}$ twice and substituting in the state space model gives:

$$\begin{aligned}\ddot{\boldsymbol{y}} &= \frac{1}{2}\dot{\boldsymbol{\Xi}}(\dot{\boldsymbol{q}}^{B/I})\boldsymbol{\omega}_B + \frac{1}{2}\boldsymbol{\Xi}(\boldsymbol{q}^{B/I})\dot{\boldsymbol{\omega}}_B\\ &= \frac{1}{2}\dot{\boldsymbol{\Xi}}(\dot{\boldsymbol{y}})\boldsymbol{\omega}_B + \frac{1}{2}\boldsymbol{\Xi}(\boldsymbol{y})\mathbf{I}_{tot}^{-1}\left[-\mathbf{A}_w\boldsymbol{u} - [\boldsymbol{\omega}_B]_\times\left(\mathbf{I}_{tot}\boldsymbol{\omega}_B + \boldsymbol{h}^{B}_{w/B}\right)\right]\end{aligned} \tag{2.39}$$

Rearranging for the input $\boldsymbol{u}$:

$$\boldsymbol{u} = -\mathbf{A}_w^{+}\left[\mathbf{I}_{tot}\boldsymbol{\Xi}^{\top}(\boldsymbol{y})\left(2\ddot{\boldsymbol{y}} - \dot{\boldsymbol{\Xi}}(\dot{\boldsymbol{y}})\boldsymbol{\omega}_B\right) - [\boldsymbol{\omega}_B]_\times(\mathbf{I}_{tot}\boldsymbol{\omega}_B + \boldsymbol{h}_w)\right] \tag{2.40}$$

While the body rate term $\boldsymbol{\omega}_B$ could be solved for using the inverse kinematics (Eq. (B.19)), as was the case with the previous two examples, the angular momentum of the reaction wheels $\boldsymbol{h}_w$ is not directly available. Instead, $\boldsymbol{h}_w$ can be found through numerical integration of the model (Eq. (2.38)), giving a semi-analytical inverse dynamic solution. The process of solving this semi-analytical solution can be seen in Fig. 2.22, with the integration completed using numerical methods such as Euler or RK4 (Appendix D.3).

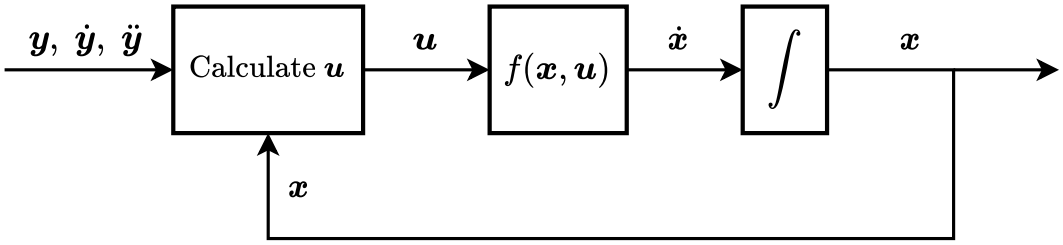


Figure 2.22: Semi-analytical NDI solution

Reaction Wheels with First-Order Torque Response

In all previous examples, the output was differentiated twice ($\nu = 2$ i.e. acceleration) to achieve the input as a function of the system's output. This was due to these dynamic systems having a relative degree of two. The addition of a first-order torque response to represent the reaction wheel's electro-mechanical system increases the relative degree of the overall system to three (Appendix E.3). In nonlinear state-space form, the dynamics can be given as:

$$\begin{aligned} \dot{\boldsymbol{x}} &= \boldsymbol{f}(\boldsymbol{x},\boldsymbol{u}) \\ \begin{bmatrix} \dot{\boldsymbol{\omega}}_B \\ \dot{\boldsymbol{q}}^{B/I} \\ \dot{\boldsymbol{h}}_w \\ \ddot{\boldsymbol{h}}_w \end{bmatrix} &= \begin{bmatrix} \mathbf{I}_{tot}^{-1}\left[-\dot{\boldsymbol{h}}_w - [\boldsymbol{\omega}_B]_\times \left(\mathbf{I}_{tot}\boldsymbol{\omega}_B + \boldsymbol{h}_w\right)\right] \\ \frac{1}{2}\boldsymbol{\Xi}(\boldsymbol{q}^{B/I})\ \boldsymbol{\omega}_B \\ \dot{\boldsymbol{h}}_w \\ \frac{\mathbf{A}_w\boldsymbol{u} - \dot{\boldsymbol{h}}_w^B}{T_w} \end{bmatrix} \end{aligned} \tag{2.41}$$

where T_w is the time constant of the reaction wheel torque. Starting with the second derivative of the output as seen before:

$$\ddot{\boldsymbol{y}} = \frac{1}{2}\dot{\boldsymbol{\Xi}}(\dot{\boldsymbol{y}})\boldsymbol{\omega}_B + \frac{1}{2}\boldsymbol{\Xi}(\boldsymbol{y})\dot{\boldsymbol{\omega}}_B \tag{2.42}$$

and then differentiating again:

$$\dddot{\boldsymbol{y}} = \frac{1}{2}\ddot{\boldsymbol{\Xi}}(\ddot{\boldsymbol{y}})\boldsymbol{\omega}_B + \dot{\boldsymbol{\Xi}}(\dot{\boldsymbol{y}})\dot{\boldsymbol{\omega}}_B + \frac{1}{2}\boldsymbol{\Xi}(\boldsymbol{y})\ddot{\boldsymbol{\omega}}_B \tag{2.43}$$

The body jerk $\ddot{\boldsymbol{\omega}}_B$ is found through differentiation of the relevant ODE in Eq. (2.41), remembering to consider transport theorem:

$$\begin{aligned} \ddot{\boldsymbol{\omega}}_B = \mathbf{I}_{tot}^{-1}\Big[-\ddot{\boldsymbol{h}}_w - [\dot{\boldsymbol{\omega}}_B]_\times \left(\mathbf{I}_{tot}\boldsymbol{\omega}_B + \boldsymbol{h}_w\right) - 2\,[\boldsymbol{\omega}_B]_\times \left(\mathbf{I}_{tot}\dot{\boldsymbol{\omega}}_B + \dot{\boldsymbol{h}}_w\right) \\ - [\boldsymbol{\omega}_B]_\times [\boldsymbol{\omega}_B]_\times \left(\mathbf{I}_{tot}\boldsymbol{\omega}_B + \boldsymbol{h}_w\right)\Big] \end{aligned} \tag{2.44}$$

which can be substituted into Eq. (2.43) along with the ODE for $\ddot{\boldsymbol{h}}_w$, found in Eq. (2.41), giving:

$$\ddot{\boldsymbol{y}} = \frac{1}{2}\ddot{\boldsymbol{\Xi}}(\ddot{\boldsymbol{y}})\boldsymbol{\omega}_B + \dot{\boldsymbol{\Xi}}(\dot{\boldsymbol{y}})\dot{\boldsymbol{\omega}}_B + \frac{1}{2}\boldsymbol{\Xi}(\boldsymbol{y})\mathbf{I}_{tot}^{-1}\left[-\frac{\mathbf{A}_w\boldsymbol{u} - \dot{\boldsymbol{h}}_w}{T_w} - [\dot{\boldsymbol{\omega}}_B]_\times(\mathbf{I}_{tot}\boldsymbol{\omega}_B + \boldsymbol{h}_w)\right.$$
$$\left. -2[\boldsymbol{\omega}_B]_\times\left(\mathbf{I}_{tot}\dot{\boldsymbol{\omega}}_B + \dot{\boldsymbol{h}}_w\right) - [\boldsymbol{\omega}_B]_\times[\boldsymbol{\omega}_B]_\times(\mathbf{I}_{tot}\boldsymbol{\omega}_B + \boldsymbol{h}_w)\right] \quad (2.45)$$

Finally, rearranging for the output $\boldsymbol{u}$ gives:

$$\boldsymbol{u} = -\mathbf{A}_w^+\left[T_w\mathbf{I}_{tot}\boldsymbol{\Xi}^\top(\boldsymbol{y})\left(2\ddot{\boldsymbol{y}} - \ddot{\boldsymbol{\Xi}}(\ddot{\boldsymbol{y}})\boldsymbol{\omega}_B - 2\dot{\boldsymbol{\Xi}}(\dot{\boldsymbol{y}})\dot{\boldsymbol{\omega}}_B\right) + T_w[\dot{\boldsymbol{\omega}}_B]_\times(\mathbf{I}_{tot}\boldsymbol{\omega}_B + \boldsymbol{h}_w)\right.$$
$$\left. +2T_w[\boldsymbol{\omega}_B]_\times\left(\mathbf{I}_{tot}\dot{\boldsymbol{\omega}}_B + \dot{\boldsymbol{h}}_w\right) + T_w[\boldsymbol{\omega}_B]_\times[\boldsymbol{\omega}_B]_\times(\mathbf{I}_{tot}\boldsymbol{\omega}_B + \boldsymbol{h}_w) - \dot{\boldsymbol{h}}_w\right] \quad (2.46)$$

where an expression for $\dot{\boldsymbol{h}}_w$ can be found from the solution of the previous system Eq. (2.40):

$$\dot{\boldsymbol{h}}_w = -\mathbf{I}_{tot}\boldsymbol{\Xi}^\top(\boldsymbol{y})\left(2\ddot{\boldsymbol{y}} - \dot{\boldsymbol{\Xi}}(\dot{\boldsymbol{y}})\boldsymbol{\omega}_B\right) - [\boldsymbol{\omega}_B]_\times(\mathbf{I}_{tot}\boldsymbol{\omega}_B + \boldsymbol{h}_w) \quad (2.47)$$

Therefore, a semi-analytical solution for a reaction wheel system with first-order torque dynamics has been found. The NDI solution can then be calculated using the process seen previously in Fig. 2.22.

2.4.2 | Comparison of Inverse Simulation and Nonlinear Dynamic Inversion (NDI)

The process of deriving the NDI solutions for each of the satellite mathematical models has proved to require extensive mathematical manipulation. Conversely, as was seen in Section 2.2, the InvSim did not require any extensive modification when the mathematical model being solved was changed.

It can be imagined that during the design of a satellite, a rigid-body mathematical model might at first be used to assess the dynamics. However, as the design of the overall system matures, more complex models may be implemented to investigate the dynamics in more detail, e.g. considering the flexibility of the satellite structure. InvSim would therefore be much more applicable for use in this iterative design process, providing a *ready-to-go* inverse dynamic solution as the design and model change. Use of NDI in the design process would be far less practical as the inverse solution would have to be re-derived every time a significant alteration is made to the design and associated mathematical model.

The inverse solutions obtained by both NDI and InvSim will now be compared quantitatively to assess their performance. For this comparison, the model parameters de-

tailed previously in Table 2.2, using the 3-wheel configuration, and InvSim hyperparameters in Table 2.4 will be used, unless otherwise mentioned. In each case, a 90-deg rest-to-rest slew manoeuvre about the body $\hat{\boldsymbol{x}}^B$ axis will be performed as before.

External Torques

First, the satellite mathematical model with external torque inputs will be inverted using both InvSim and NDI. A comparison of the input solutions is shown in Fig. 2.23, from which it can be seen that NDI and InvSim produce near identical results. Comparison of the output tracking and pointing error, as seen in Fig. 2.24, shows that during the manoeuvre, the NDI has a higher pointing error than the InvSim solution.

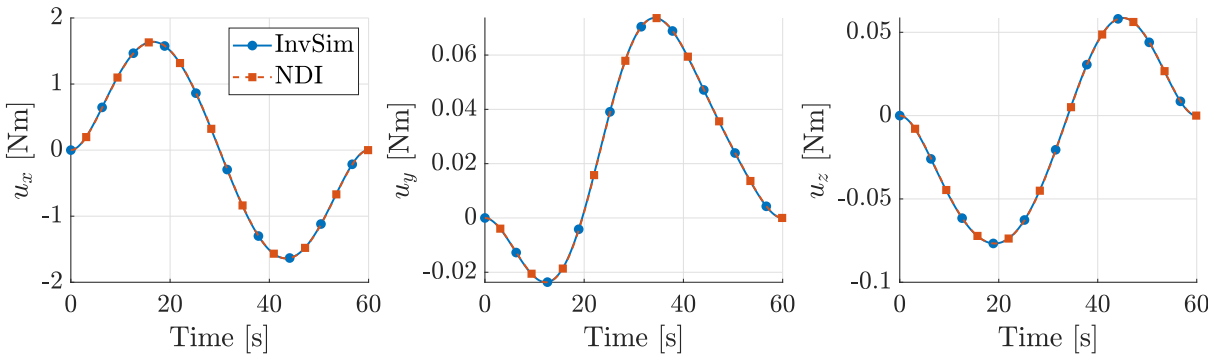


Figure 2.23: InvSim vs NDI: External torque system, comparison of input solution, InvSim $\nu = 1$, NDI $\nu = 2$

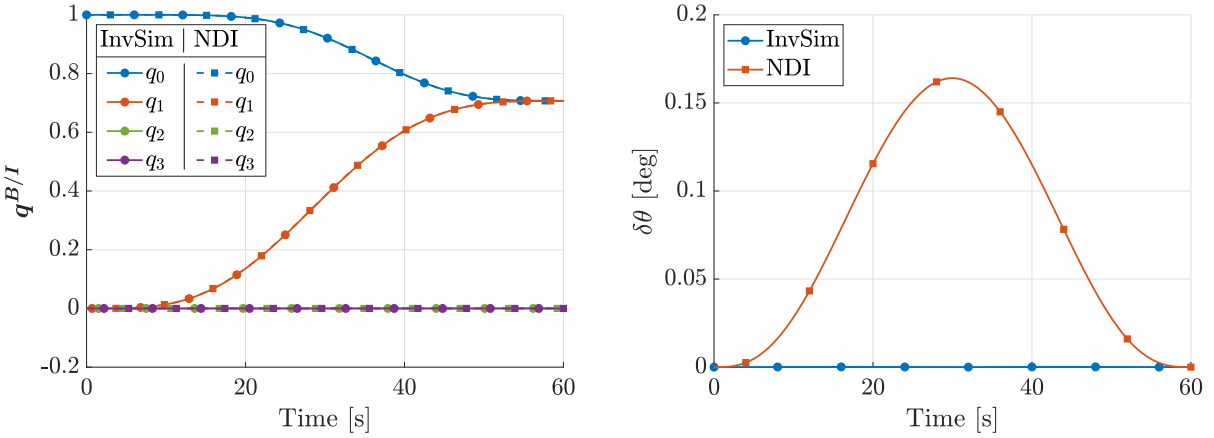


Figure 2.24: InvSim vs NDI: External torque system, comparison of output tracking, InvSim $\nu = 1$, NDI $\nu = 2$

It was shown previously in Section 2.2.2 that as the differential order of the desired output ν increased, more significant pointing errors would accumulate. The NDI solution was itself obtained through differentiation of the desired output position twice (Eq. (2.32)), which appears in the analytical input solution given in Eq. (2.33). The NDI solution is therefore effectively being driven with the second-order derivative of the

output, equivalent to InvSim using a desired output order of $\nu = 2$. The InvSim solution in Fig. 2.24 used a desired output derivative order of $\nu = 1$, which is lower than that used by the NDI and mostly accounts for the lower pointing error.

This can be confirmed by comparing the NDI solution to an InvSim solution which uses a desired output order of $\nu = 2$. In Fig. 2.25, the pointing errors in the InvSim and NDI are now similar as they each use the same order of output derivative to drive their solutions. This behaviour has also been reported previously in literature [158].

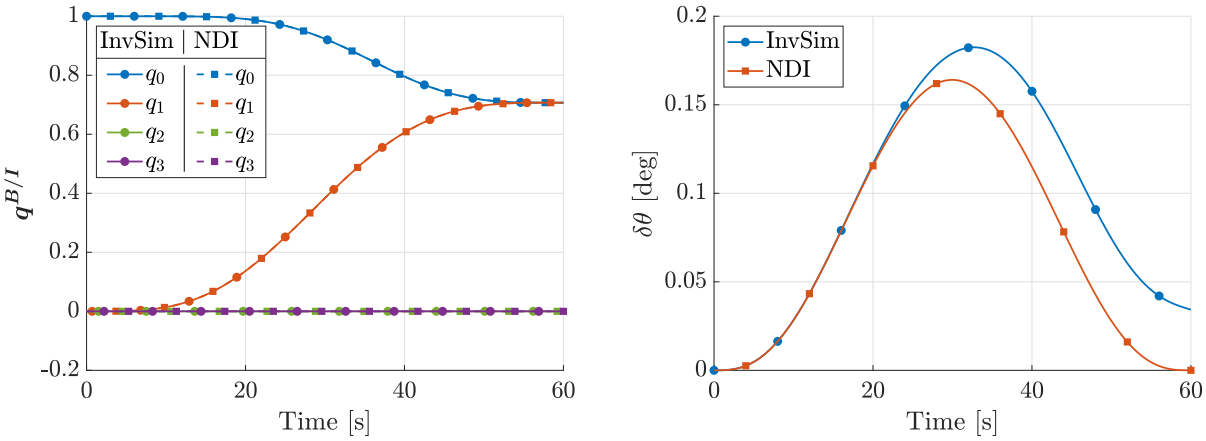


Figure 2.25: InvSim vs NDI: External torque system, comparison of output tracking, InvSim $\nu = 2$, NDI $\nu = 2$

Reaction Wheels

The NDI solution when reaction wheel dynamics are included in the model is also driven by the second derivative of the output ($\nu = 2$), as can be seen in Eq. (2.40). Therefore, similar behaviour with the pointing error is seen for the InvSim and NDI solutions in Fig. 2.26.

It was shown in Section 2.2.2 that the InvSim pointing error $\delta\theta$ decreased for lower values of the discretisation timestep T; this is also the case with NDI. In Fig. 2.27, the results of InvSim using a discretisation timestep of $T = 0.1$ s is compared to NDI with a lower discretisation timestep of $T = 0.01$ s. Through comparison with Fig. 2.26 it can be seen that the maximum pointing error $\delta\theta$ has reduced proportionally with the reduction in the discretisation timestep T. However, while the pointing error $\delta\theta$ for the NDI has been reduced significantly, it is still higher than InvSim. It seems that the differential order of the output ν used to drive the inverse solution has the most impact on accuracy. InvSim therefore has an advantage, as it can target any differential order ν, while NDI is limited by the relative order of the system input-output dynamics.

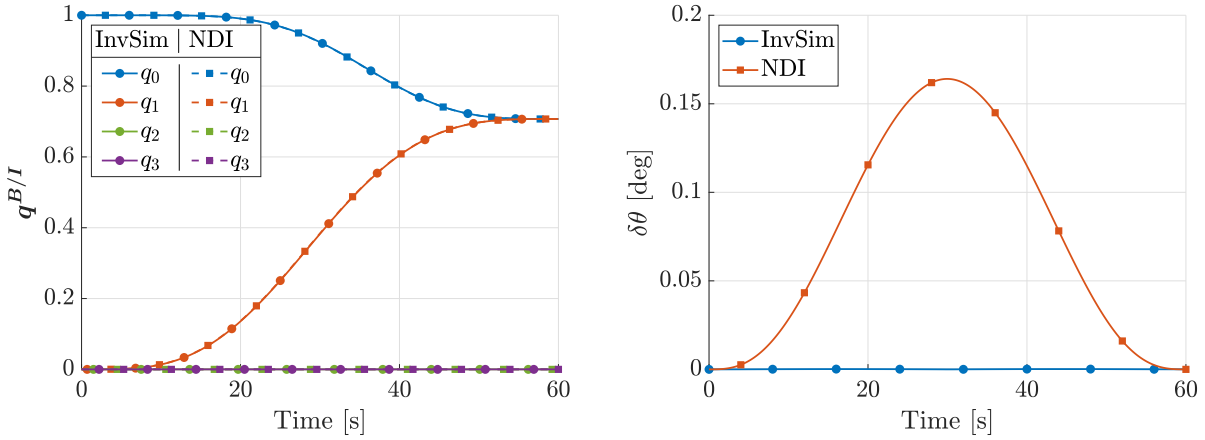


Figure 2.26: InvSim vs NDI: Reaction wheel system, comparison of output tracking, InvSim $\nu = 1$, NDI $\nu = 2$, $T = 0.1$ s

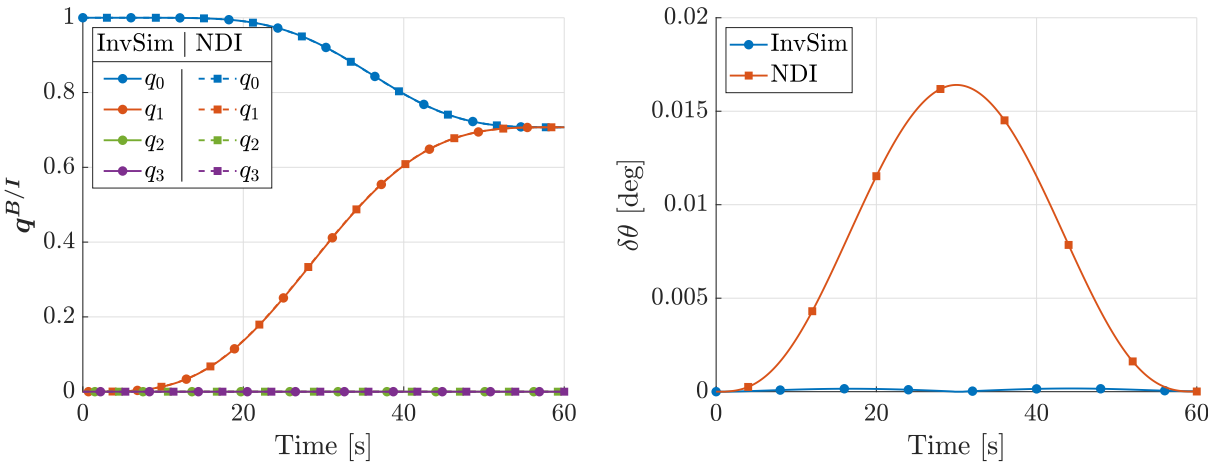


Figure 2.27: InvSim vs NDI: Reaction wheel system, comparison of output tracking, InvSim $\nu = 1$ $T = 0.1$ s, NDI $\nu = 2$ $T = 0.01$ s

Reaction Wheels with First-Order Dynamics

The introduction of the first-order torque response increases the overall order of the system dynamics, thus requiring further differentiation before the NDI solution in Eq. (2.46) could be found. The differential order of the output used to drive the NDI solution, therefore, increases by one to $\nu = 3$. Comparison of the NDI and InvSim in Fig. 2.28 shows that while the maximum pointing error has not increased significantly when compared to Fig. 2.26, there is now a significant steady-state error at the end of the slew manoeuvre.

Another point to consider is whether the additional actuator dynamics need to be taken into account, or whether the torque can just be assumed to act instantaneously. A comparison can be made between InvSim that takes into account the first-order torque response, and an NDI solution that assumes that the torque acts instantaneously, i.e. the

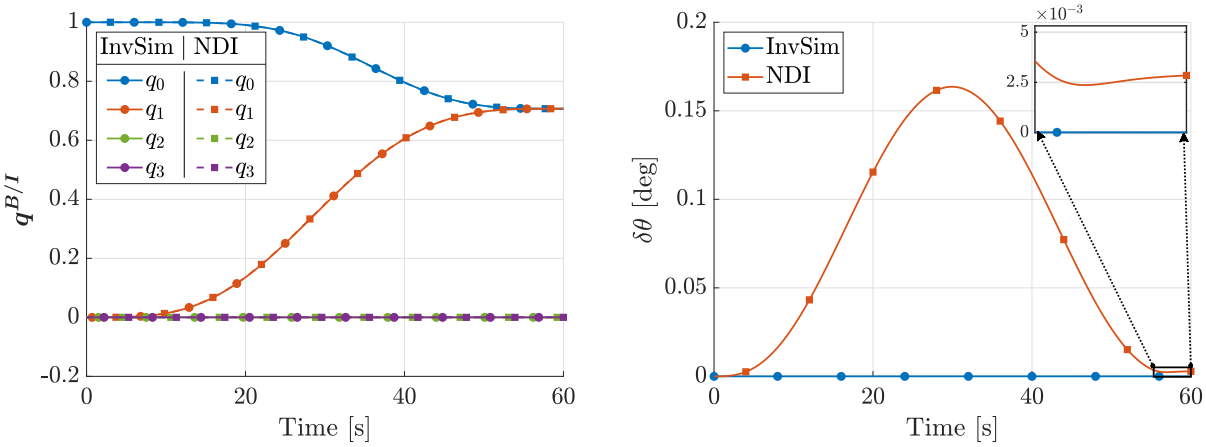


Figure 2.28: InvSim vs NDI: Reaction wheel system with first-order response, comparison of output tracking, InvSim $\nu = 1$, NDI $\nu = 3$ correct input solution Eq. (2.46)

solution found in Eq. (2.40). Each input solution is tested on a model with first-order torque response and plotted in Fig. 2.29. It can be seen that NDI now struggles to follow the slew manoeuvre closely, with the maximum pointing error exceeding 3 deg.

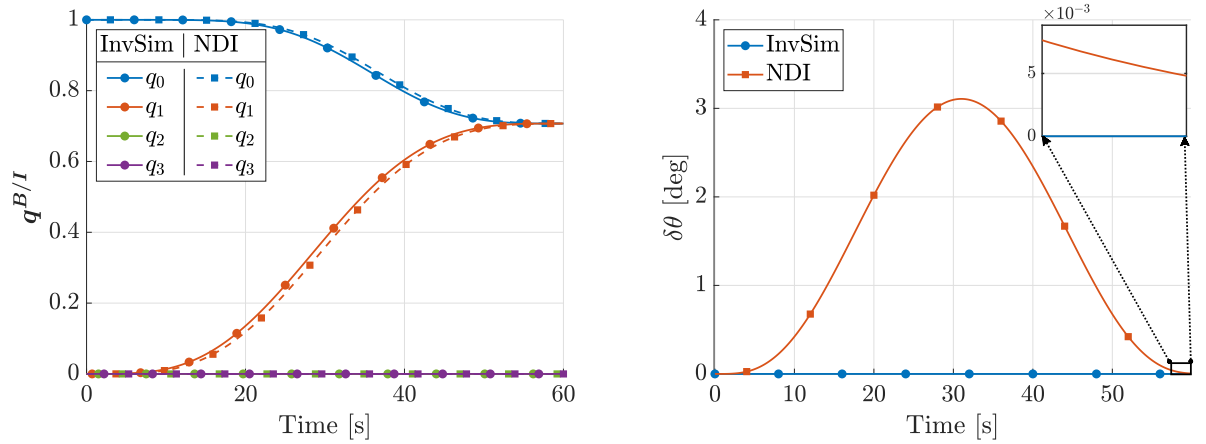
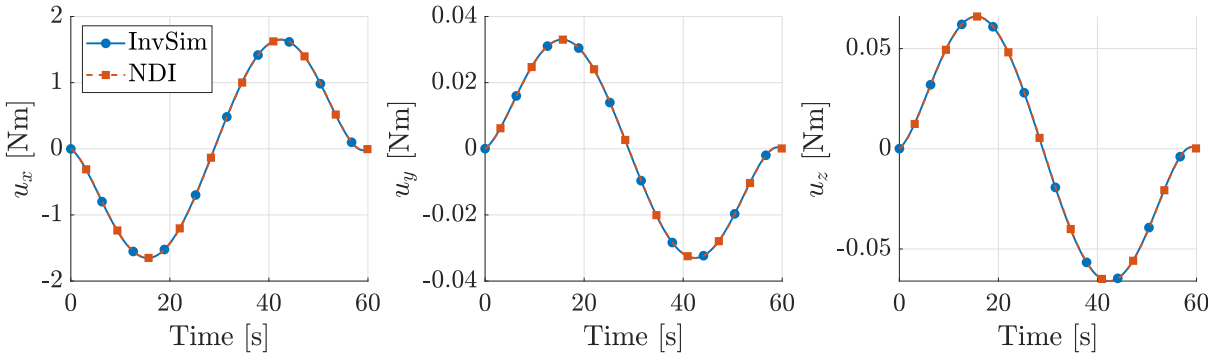


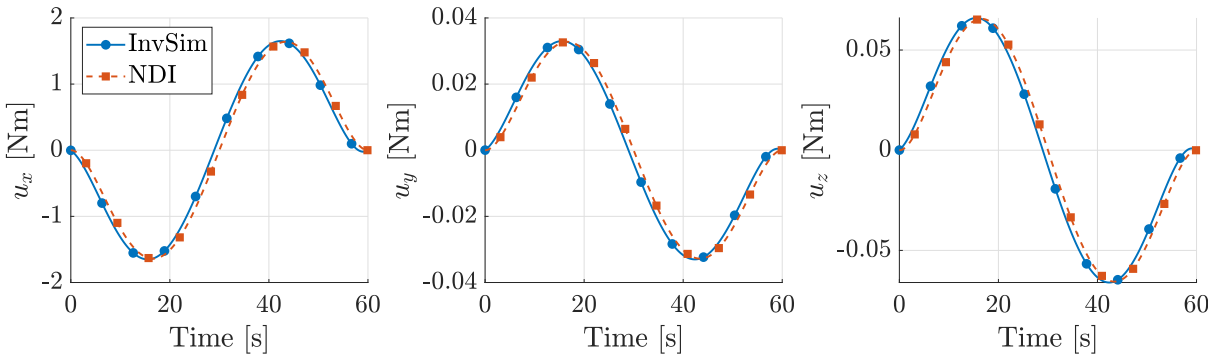
Figure 2.29: InvSim vs NDI: Reaction wheel system with first-order response, comparison of output tracking, InvSim $\nu = 1$, NDI $\nu = 2$ incorrect input solution Eq. (2.40) assuming instant torque

The correct and incorrect NDI input solutions, Eq. (2.46) and Eq. (2.40) respectively, are compared against the InvSim solution in Fig. 2.30. The correct and incorrect input solutions are offset due to the delay imparted by the first-order torque response. Re-derivation of the NDI solution is, therefore, necessary to ensure close tracking of the attitude slew manoeuvre.

This also illustrates that if an error is made in the mathematical derivation of the NDI solution, then accuracy can degrade significantly. InvSim, however, avoids the need for tedious mathematical derivations that could be subject to human error. As long as



(a) InvSim $\nu = 1$, NDI $\nu = 3$ correct input solution Eq. (2.46)



(b) InvSim $\nu = 1$, NDI $\nu = 2$ incorrect input solution Eq. (2.40) assuming instant torque

Figure 2.30: InvSim vs NDI: Reaction wheel system with first-order response, comparison of input solution

there exists a validated mathematical model with a stable forward simulation, InvSim can provide the inverse dynamic solution.

Magnetic Torques

As shown previously with InvSim in Section 2.2.2, the use of magnetic-only control poses issues when a non-diagonal inertia matrix is present and the attitude is fully constrained. In this scenario, the InvSim would simply fail to converge, evaluating the desired output as *unrealisable* by the system dynamics.

Instead, using a pseudoinverse (Eq. (2.37)), the NDI will converge to a least-squares-fit solution. This, coupled with the second-order derivative $\nu = 2$ of the output being used to drive the solution, leads to significant divergence from the desired output path, as can be seen in Fig. 2.31.

It could be said that this least-squares-fit solution is more desirable than the *no solution* given by the InvSim. However, this failure of the InvSim to converge can be seen as an advantage over NDI. The failure of the InvSim to converge tells the user that the given desired output for the current system dynamics is *unrealisable*, whereas the NDI simply

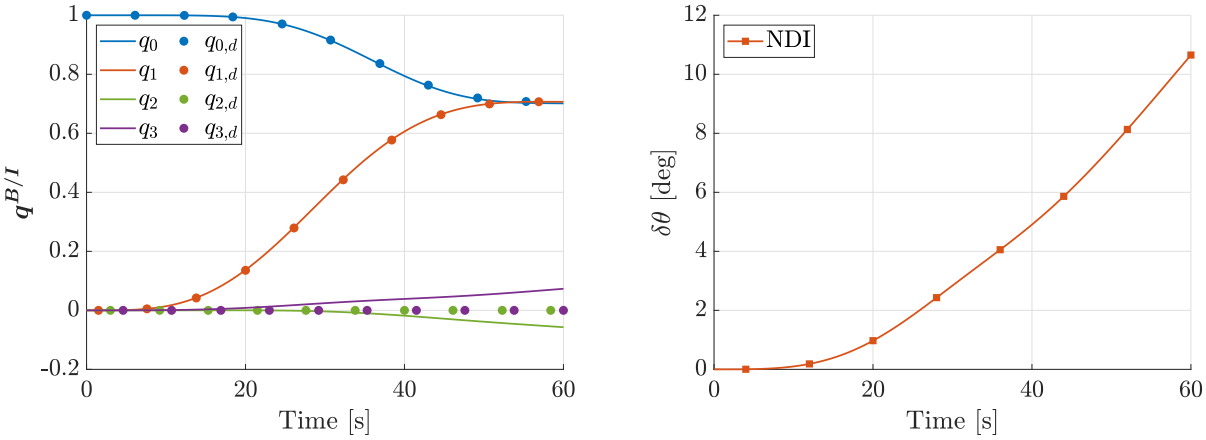


Figure 2.31: NDI: Magnetic control system, output tracking, non-diagonal inertia

attempts, and fails, to converge to a solution. It can be imagined that if the inverse dynamic solution is to be used within some larger optimisation problem, having the ability to determine a desired output to be *unrealisable* could be useful. The user can then make an informed decision to redesign the system or modify the path planning algorithm to give a realisable desired output path.

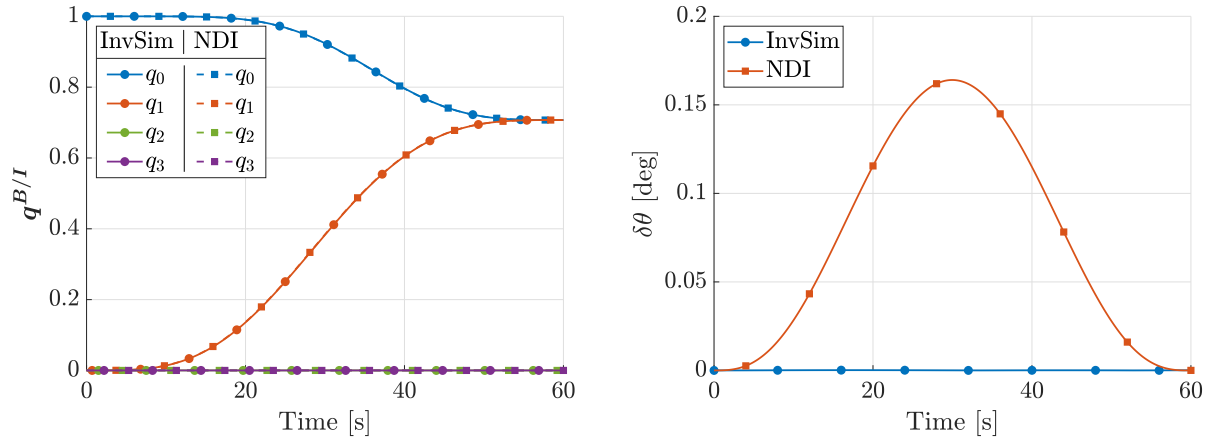


Figure 2.32: InvSim vs NDI: Magnetic control system, comparison of output tracking, diagonal inertia, InvSim using $\boldsymbol{u}_0 = \boldsymbol{u}[(k-1)T]$

Similarly to InvSim, it can be seen in Fig. 2.32 that the NDI solution closely follows the attitude slew path when a diagonal inertia matrix is used.

One of the InvSim hyperparameters that must be chosen is the initial input $\boldsymbol{u}_0$. Most commonly, this is assumed as the solution to the previous timestep $\boldsymbol{u}_0 = \boldsymbol{u}[(k-1)T]$, which, if the input solution is smooth and continuous, is a good assumption to make. However, this gives undesirable results in the case of magnetic-only control, as shown in Fig. 2.33. The InvSim, while converging to the desired output trajectory, produces a higher magnitude input solution than NDI. This is due to there being more than one

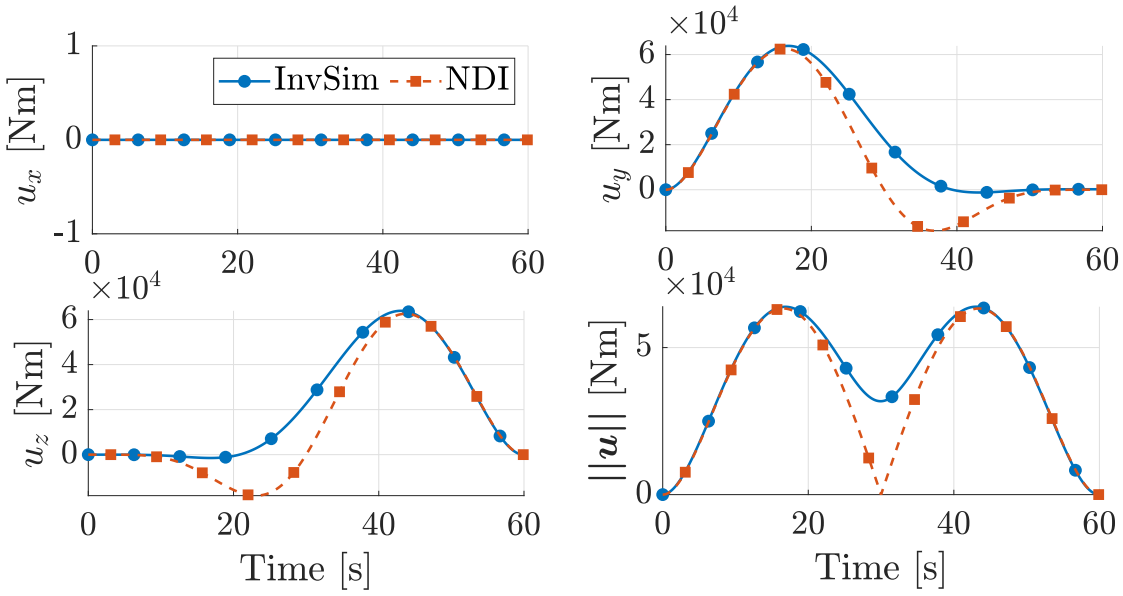


Figure 2.33: InvSim vs NDI: Magnetic control system, comparison of input solution, non-diagonal inertia, InvSim using $\boldsymbol{u}_0 = \boldsymbol{u}[(k-1)T]$

combination of u_y and u_z magnetic dipoles that produce the required torque on the satellite.

The InvSim will converge to the solution closest to the initial estimate, which, if set to zero $\boldsymbol{u}_0 = 0$ will result in the lower magnitude solution, as can be seen in Fig. 2.34. Alternatively, modifications can be made to the InvSim process to allow for additional criteria on the input solution to be defined through a suitable cost function. This approach will be investigated in a later chapter of this thesis. Conversely, the NDI will always converge to the minimum norm solution and does not require careful selection of an initial estimate $\boldsymbol{u}_0$.

Speed Comparison

The speed of the InvSim and NDI can also be compared. The NDI solutions for both the external torque and magnetic control inputs, Eq. (2.34) and Eq. (2.37) respectively, are both closed-form and fully analytical. They will therefore not require the model to be numerically integrated forward in time, with their solutions effectively being available instantaneously.

When reaction wheels are included in the model, the NDI solutions in Eqs. (2.40) and (2.46) become only semi-analytical, requiring forward numerical integration of the model as illustrated in Fig. 2.22. The overall speed of both the InvSim and NDI for each of the mathematical models is shown in Fig. 2.35 in terms of *total model calls*, as was the case in the hyperparameter study conducted earlier (Section 2.2.2). The NDI is

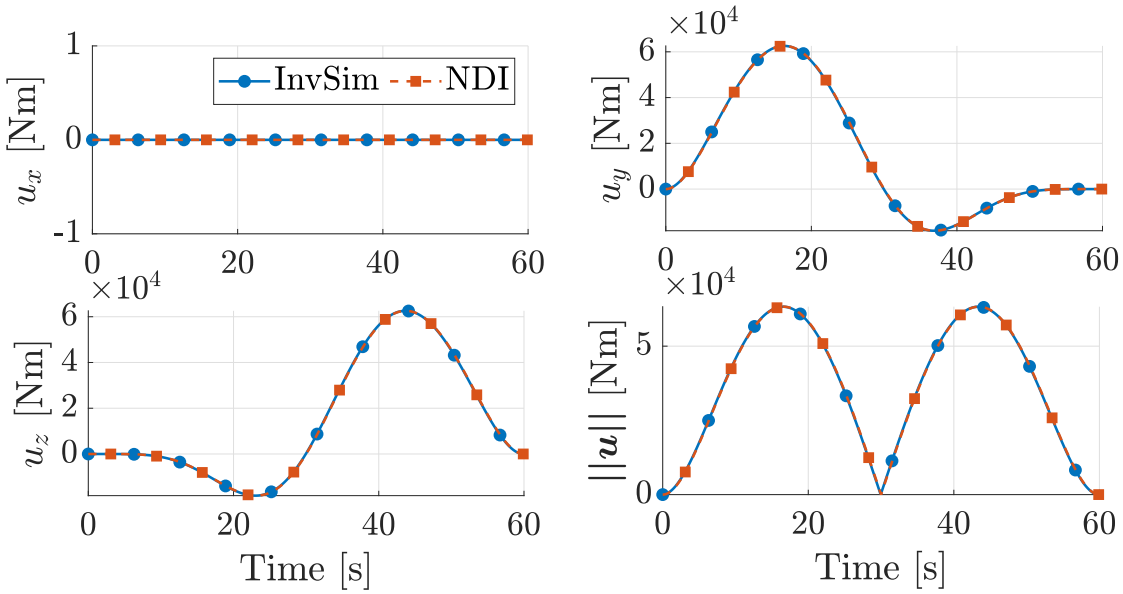


Figure 2.34: InvSim vs NDI: Magnetic control system, comparison of input solution, non-diagonal inertia, InvSim using $\boldsymbol{u}_0 = 0$

faster in every case, with the model only ever being integrated forward in time $\Delta t / T$ times per discretisation timestep T. The InvSim is slower due to its iterative nature, and the requirement to integrate the model forward many more times at each iteration to calculate the Jacobian.

Overall, InvSim provides a much more practical approach to solving the inverse dynamics of a system. This, however, comes at the cost of significantly greater computational effort than other approaches such as NDI, and in some cases requires particular care when selecting the initial input estimate.

2.5 | Summary

This chapter laid out much of the foundations upon which the rest of the thesis is built, outlining the specific InvSim algorithm used and demonstrating its application to satellite attitude dynamics in their simplest form, i.e. rigid body. A hyperparameter study revealed that both the differential order ν, and lower discretisation timestep T had the biggest impact on InvSim accuracy. Small discretisation timesteps T require additional computational effort, and thus, must be chosen to ensure a good balance between speed and accuracy. It was also determined that while stricter tolerances $\delta \dot{y}_{tol}^{(\nu)}$ are usually desirable, when set too low, the computational effort could increase without any improvements in accuracy.

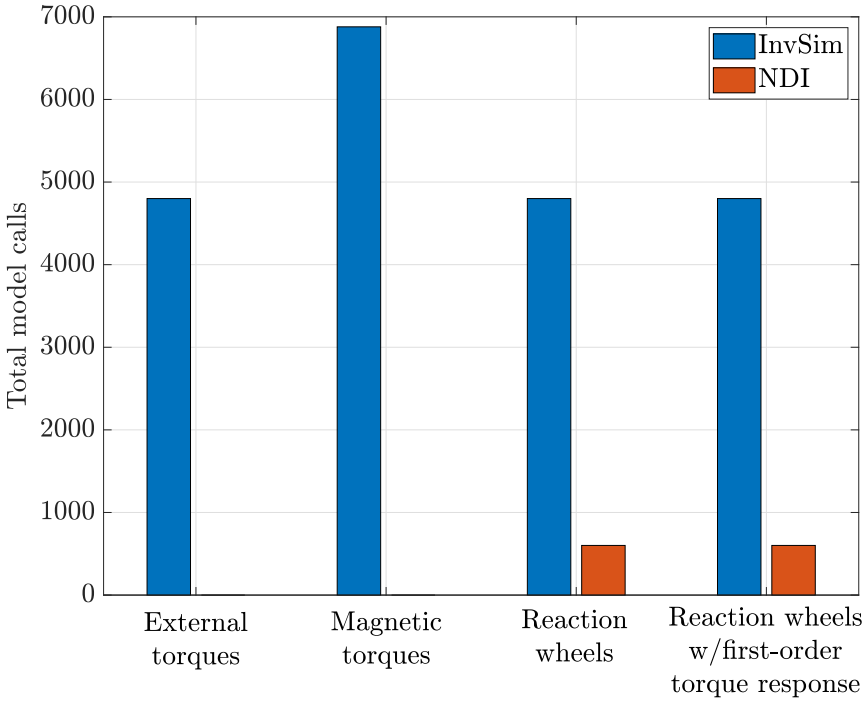


Figure 2.35: InvSim vs NDI: Speed comparison

Numerical stability issues associated with applying InvSim to satellite attitude dynamics were also addressed. Higher-order derivative outputs ν were found to improve robustness to Jacobian inaccuracies, and lower discretisation timesteps T. Several numerical stability issues in the case of magnetic-only control were also highlighted, including sensitivity to the initial estimate $\boldsymbol{u}_0$, and loss of control in the direction of the magnetic field (i.e. null space).

Compared to the alternative analytical NDI method, InvSim was found to be more practical, having the benefit of not requiring a re-derivation when the model is modified, and giving a more accurate solution; at the cost of higher computational power, and requiring an initial input estimate. InvSim also displayed the flexibility to solve for different combinations of outputs, allowing for various mission scenarios to be represented easily.

The insights into the InvSim algorithm, and its application to satellite attitude dynamics, as uncovered in this chapter, will be further utilised in subsequent chapters. In particular, these insights will inform the use of InvSim as part of a design framework for satellites with flexible structures. First, however, the next chapter applies InvSim within a model-based attitude control architecture.

Chapter 3 | Attitude Control of Satellites with Flexible Appendages using Inverse Simulation

THE basic application of InvSim to the attitude dynamics of satellites has now been demonstrated, including: consideration of the method's flexibility and practicality as a dynamic inversion tool, evaluation of its numerical performance, and investigation of its stability properties. However, the inclusion of the flexible modes of a satellite's structure has not thus far not been included in the models being solved using InvSim. One common configuration, as discussed in Section 1.1.1, is to attach deployable and lightweight, and thus flexible, appendages to a relatively rigid main hub to accommodate a variety of instrumentation and equipment. Sufficiently flexible appendages can induce significant disturbances on the satellite and impact the performance of attitude-tracking control schemes. The inverse dynamic solution provided by InvSim could conceivably be utilised to provide the required input torques for tracking a prescribed attitude manoeuvre, which, if a sufficiently accurate model is used, could be used in an InvSim-based attitude control architecture to compensate for the disturbances induced by flexible modes.

This chapter will therefore develop a multi-fidelity mathematical model of a satellite with flexible appendages, allowing for the effects of model fidelity and uncertainty on the InvSim-based controller to be investigated. The effect that inclusion of the flexible modes has on the performance and stability of the InvSim numerical process is also considered, as this could have significant implications when attempting to implement an InvSim-based attitude controller on real-time hardware. It is also suggested in this chapter that InvSim can be utilised in a path-planning scheme to reduce appendage flexing during slew manoeuvres. Finally, it is demonstrated that the proposed InvSim-based control architecture can make use of existing simulation frameworks.

3.1 | Development of a Multi-Fidelity Model

To facilitate later investigations into the effect of model fidelity on performance of an InvSim-based attitude controller, a suitable *multi-fidelity* mathematical model is developed. An example of a satellite comprising a main rigid hub to which two symmetric flexible appendages are attached can be seen in Fig. 3.1.

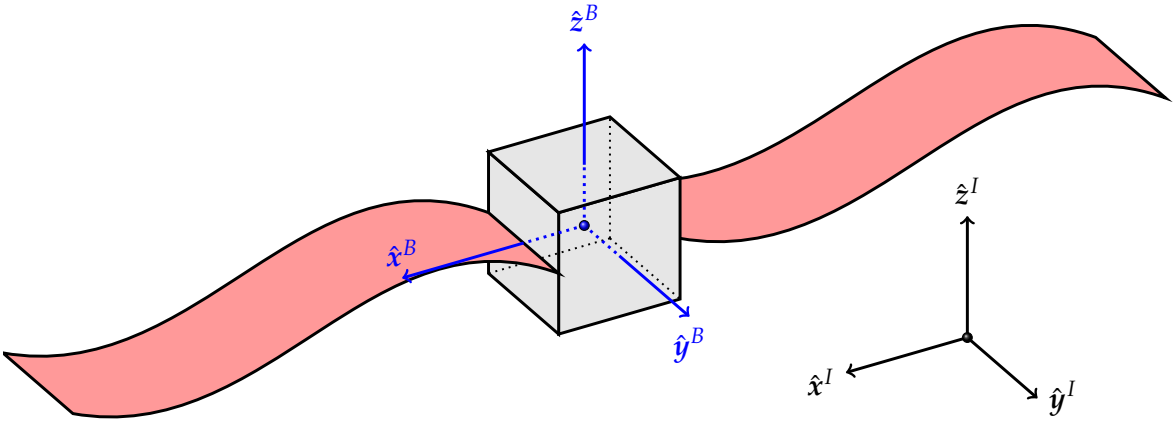


Figure 3.1: Satellite with two flexible appendages

3.1.1 | Hybrid Coordinate Approach

One of the most common approaches to model a satellite with such a configuration is to use what is known as the *Hybrid Coordinate* approach. This combines discrete coordinates to represent the dynamics of the rigid hub, i.e. body rate $\boldsymbol{\omega}_{B/I}^{B}$ and attitude quaternion $\boldsymbol{q}^{B/I}$, with *modal* coordinates for the flexible appendages. Using modal coordinates allows for the deformation in each appendage to be deconstructed into several flexible dynamical modes. For example, modelling each appendage as a beam with deflections only occurring about one axis yields a time-dependent function $w(l,t)$ describing the vertical deformation due to bending:

$$w(l,t) = \sum_{m=1}^{N_m} \phi_m(l)\eta_m(t) = \boldsymbol{\phi}^\top(l)\boldsymbol{\eta}(t) \tag{3.1}$$

where l is the position along the appendage as illustrated in Fig. 3.2, $\phi_m(l)$ is the modal shape, and $\eta_m(t)$ is the nondimensional modal coordinate of the m^{th} mode, and N_m is the number of modes considered. The summation can alternatively be performed using the dot product of vectors containing modal shapes and coordinates as seen in Eq. (3.1). The deformation is now spatially and temporally decoupled, allowing for the

velocity to be found with only the derivative of $\eta_m(t)$ required:

$$\dot{w}(l,t) = \sum_{m=1}^{N_m} \phi_m(l)\dot{\eta}_m(t) = \boldsymbol{\phi}^\top(l)\dot{\boldsymbol{\eta}}(t) \tag{3.2}$$

The Lagrangian

The set of ODEs that describes both the rigid-body and flexible dynamics, and the dynamic coupling between the two, can be obtained by considering the total energy in the system. First, the total kinetic energy $\mathcal{T}$ of a satellite with N_{ap} flexible appendages and a set of reaction wheels can be defined:

$$\begin{aligned}\mathcal{T} = & \underbrace{\frac{1}{2}{\boldsymbol{\omega}^B_{B/I}}^\top \mathbf{I}_{hub}\boldsymbol{\omega}^B_{B/I}}_{\text{kinetic energy of rigid hub}} + \underbrace{\frac{1}{2}\left(\boldsymbol{\omega}^B_{w/B}+\boldsymbol{\omega}^B_{B/I}\right)^\top \mathbf{I}_w \left(\boldsymbol{\omega}^B_{w/B}+\boldsymbol{\omega}^B_{B/I}\right)}_{\text{kinetic energy of reaction wheels}} + \\ & + \underbrace{\sum_{i=1}^{N_{ap}} \frac{1}{2}\rho_i \int_o^{L_i} {\dot{\boldsymbol{r}}^B_{i/B}}^\top \dot{\boldsymbol{r}}^B_{i/B} dl}_{\substack{\text{integral of kinetic energy of} \\ \text{each appendage along its length}}}\end{aligned} \tag{3.3}$$

where $\mathbf{I}_{hub}$ is the inertia of the main rigid hub, $\mathbf{I}_w$ is the inertia of the reaction-wheel set, $\boldsymbol{\omega}^B_{w/B}$ is the angular velocity of the wheel relative to the main hub, ρ_i is the mass per length density, and $\dot{\boldsymbol{r}}^B_{i/B}$ is the velocity of the infinitesimal mass element $\rho_i dl$ of the i^{th} appendage. The position of each mass element of an appendage can be given:

$$\boldsymbol{r}^B_{i/B} = (d_i + l)\hat{\mathbf{x}}^B_i + w_i(l,t)\hat{\mathbf{z}}^B_i \tag{3.4}$$

where d_i is the distance from the origin of the body frame (centre of the rigid hub) to the root of the i^{th} appendage as seen in Fig. 3.2. Taking the time derivative of Eq. (3.4), and remembering to consider transport theorem as defined in Appendix C (Eq. (C.8)), to give:

$$\dot{\boldsymbol{r}}^B_{i/B} = \dot{w}_i(l,t)\hat{\mathbf{z}} + \left[\boldsymbol{\omega}^B_{B/I}\right]_\times \left((d_i + l)\hat{\mathbf{x}}^B_i + w_i(l,t)\hat{\mathbf{z}}^B_i\right) \tag{3.5}$$

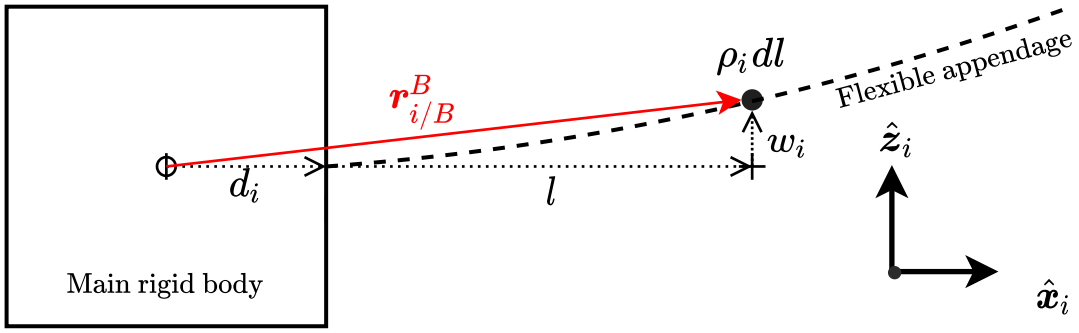


Figure 3.2: Position of mass $\rho_i dl$ along each appendage

The integral term in Eq. (3.3) can then be formed using Eq. (3.5) and substituting in Eq. (3.2):

$$\frac{1}{2}\rho_i \int_0^{L_i} {\dot{\boldsymbol{r}}^B_{i/B}}^\top \dot{\boldsymbol{r}}^B_{i/B} dl = \rho_i \int_0^{L_i} \left[\frac{1}{2}\dot{\boldsymbol{\eta}}_i(t)^\top \boldsymbol{\phi}_i(l)\boldsymbol{\phi}_i(l)^\top \dot{\boldsymbol{\eta}}_i(t) + \right.$$
$$+ \boldsymbol{\phi}_i(l)^\top \dot{\boldsymbol{\eta}}_i(t)(d_i+l)\hat{\boldsymbol{z}}_i^{B\top} \left[\hat{\boldsymbol{x}}_i^B\right]_\times^\top \boldsymbol{\omega}^B_{B/I} + \frac{1}{2}{\boldsymbol{\omega}^B_{B/I}}^\top \left(2\boldsymbol{\phi}_i^\top(l)\boldsymbol{\eta}_i(t)(d_i+l)\left[\hat{\boldsymbol{x}}_i^B\right]_\times \left[\hat{\boldsymbol{z}}_i^B\right]_\times^\top + \right.$$
$$\left. \left. + (d_i+l)^2 \left[\hat{\boldsymbol{x}}_i^B\right]_\times \left[\hat{\boldsymbol{x}}_i^B\right]_\times^\top + \boldsymbol{\eta}_i(t)^\top \boldsymbol{\phi}_i(l)\boldsymbol{\phi}_i(l)^\top \boldsymbol{\eta}_i(t) \left[\hat{\boldsymbol{z}}_i^B\right]_\times \left[\hat{\boldsymbol{z}}_i^B\right]_\times^\top \right) \boldsymbol{\omega}^B_{B/I} \right] dl \quad (3.6)$$

This can be simplified by considering the following definition of the mass matrix $\mathbf{M}_i$ for each appendage:

$$\mathbf{M}_i = \rho_i \int_0^{L_i} \boldsymbol{\phi}_i \boldsymbol{\phi}_i^\top dl \quad (3.7)$$

which includes the outer product of the modal shape vector $\boldsymbol{\phi}_i \boldsymbol{\phi}_i^\top$. Substituting the mass matrix $\mathbf{M}_i$ into Eq. (3.6) gives a final definition of the total kinetic energy of each appendage:

$$\frac{1}{2}\rho_i \int_0^{L_i} {\dot{\boldsymbol{r}}^B_{i/B}}^\top \dot{\boldsymbol{r}}^B_{i/B} dl = \underbrace{\frac{1}{2}\dot{\boldsymbol{\eta}}_i(t)^\top \mathbf{M}_i \dot{\boldsymbol{\eta}}_i(t)}_{\substack{\text{vibration energy} \\ \text{in the appendage}}} + \underbrace{\rho_i \int_0^{L_i} \boldsymbol{\phi}_i(l)^\top \dot{\boldsymbol{\eta}}_i(t)(d_i+l)(\hat{\boldsymbol{z}}_i^B)^\top \left[\hat{\boldsymbol{x}}_i^B\right]_\times^\top \boldsymbol{\omega}^B_{B/I} dl}_{\text{coupling between rigid hub and appendage}} +$$
$$+ \frac{1}{2}{\boldsymbol{\omega}^B_{B/I}}^\top \left[\underbrace{\rho_i \int_0^{L_i} 2\boldsymbol{\phi}_i^\top(l)\boldsymbol{\eta}_i(t)(d_i+l)\left[\hat{\boldsymbol{x}}_i^B\right]_\times \left[\hat{\boldsymbol{z}}_i^B\right]_\times^\top dl + \boldsymbol{\eta}_i(t)^\top \mathbf{M}_i \boldsymbol{\eta}_i(t) \left[\hat{\boldsymbol{z}}_i^B\right]_\times \left[\hat{\boldsymbol{z}}_i^B\right]_\times^\top}_{\text{change in inertia due to deformation of the appendage}} + \right.$$
$$\left. + \underbrace{\rho_i \left(d_i^2 L_i + d_i L_i^2 + \frac{L_i^3}{3} \right) \left[\hat{\boldsymbol{x}}_i^B\right]_\times \left[\hat{\boldsymbol{x}}_i^B\right]_\times^\top}_{\substack{\text{inertia of the appendage at rest}}} \right] \boldsymbol{\omega}^B_{B/I} \quad (3.8)$$

where L_i is the total length of the appendage. The final term in Eq. (3.8) effectively gives the contribution of the inertia of the appendage, both at rest and when deformed. If the effect of the appendage deformation on the overall inertia of the satellite is assumed to be negligible, then the total inertia of the satellite can be defined as:

$$\mathbf{I}_{tot} = \mathbf{I}_{hub} + \mathbf{I}_w + \sum_{i=1}^{N_{ap}} \rho_i \left(d_i^2 L_i + d_i L_i^2 + \frac{L_i^3}{3} \right) \left[\hat{\boldsymbol{x}}_i^B \right]_\times \left[\hat{\boldsymbol{x}}_i^B \right]_\times^\top \tag{3.9}$$

which, along with Eq. (3.8) can be substituted into Eq. (3.3) to give the total energy in the system:

$$\begin{aligned} \mathcal{T} = \frac{1}{2} {\boldsymbol{\omega}_{B/I}^B}^\top \mathbf{I}_{tot} \boldsymbol{\omega}_{B/I}^B + \frac{1}{2} {\boldsymbol{\omega}_{w/B}^B}^\top \mathbf{I}_w \boldsymbol{\omega}_{w/B}^B + {\boldsymbol{\omega}_{w/B}^B}^\top \mathbf{I}_w \boldsymbol{\omega}_{B/I}^B + \sum_{i=1}^{N_{ap}} \frac{1}{2} \dot{\boldsymbol{\eta}}_i(t)^\top \mathbf{M}_i \dot{\boldsymbol{\eta}}_i(t) + \\ + \rho_i \int_0^{L_i} \boldsymbol{\phi}_i(l)^\top \dot{\boldsymbol{\eta}}_i(t) (d_i + l) (\hat{\boldsymbol{z}}_i^B)^\top \left[\hat{\boldsymbol{x}}_i^B \right]_\times^\top \boldsymbol{\omega}_{B/I}^B dl \end{aligned} \tag{3.10}$$

Next, the total potential energy in the system needs to be defined. This will simply be the total potential energy stored in the appendages due to bending:

$$\mathcal{V} = \sum_{i=1}^{N_{ap}} (EI)_i \int_0^{L_i} \left(\frac{\partial^2 w_i(l,t)}{\partial l^2} \right)^2 dl \tag{3.11}$$

where $(EI)_i$ is the bending stiffness of the appendage, consisting of Young's modulus E, and the second moment of area I. Substituting in the definition for appendage deformation, Eq. (3.1), gives:

$$\mathcal{V} = \sum_{i=1}^{N_{ap}} (EI)_i \boldsymbol{\eta}_i^\top(t) \left[\int_0^{L_i} \left(\frac{\partial^2 \boldsymbol{\phi}_i(l)}{\partial l^2} \right) \left(\frac{\partial^2 \boldsymbol{\phi}_i(l)}{\partial l^2} \right)^\top dl \right] \boldsymbol{\eta}_i(t) \tag{3.12}$$

Similar to the mass matrix, a stiffness matrix for the appendage can be defined for simplification as:

$$\mathbf{K}_i = (EI)_i \int_0^{L_i} \left(\frac{\partial^2 \boldsymbol{\phi}_i(l)}{\partial l^2} \right) \left(\frac{\partial^2 \boldsymbol{\phi}_i(l)}{\partial l^2} \right)^\top dl \tag{3.13}$$

giving a condensed form of the total potential energy:

$$\mathcal{V} = \sum_{i=1}^{N_{ap}} \boldsymbol{\eta}_i^\top(t) \mathbf{K}_i \boldsymbol{\eta}_i(t) \tag{3.14}$$

Finally, with the aforementioned assumptions, the full Lagrangian of the system can be formed:

$$\mathcal{L} = \frac{1}{2}{\boldsymbol{\omega}_{B/I}^{B}}^{\top}\mathbf{I}_{tot}\boldsymbol{\omega}_{B/I}^{B} + \frac{1}{2}{\boldsymbol{\omega}_{w/B}^{B}}^{\top}\mathbf{I}_{w}\boldsymbol{\omega}_{w/B}^{B} + {\boldsymbol{\omega}_{w/B}^{B}}^{\top}\mathbf{I}_{w}\boldsymbol{\omega}_{B/I}^{B} + \sum_{i=1}^{N_{ap}}\frac{1}{2}\dot{\boldsymbol{\eta}}_i(t)^{\top}\mathbf{M}_i\dot{\boldsymbol{\eta}}_i(t) + \\ + \sum_{i=1}^{N_{ap}}\rho_i\int_0^{L_i}\boldsymbol{\phi}_i(l)^{\top}\dot{\boldsymbol{\eta}}_i(t)(d_i+l)(\hat{\boldsymbol{z}}_i^{B})^{\top}\left[\hat{\boldsymbol{x}}_i^{B}\right]_{\times}^{\top}\boldsymbol{\omega}_{B/I}^{B}dl - \sum_{i=1}^{N_{ap}}\boldsymbol{\eta}_i^{\top}(t)\mathbf{K}_i\boldsymbol{\eta}_i(t) \quad (3.15)$$

3.1.2 | Rigid-Body Dynamics

Assuming no external torques, the Euler-Lagrange equation, as defined in Appendix C (Eq. (C.7a)), can be applied to the rigid-body dynamics:

$$\frac{d}{dt}\left(\frac{\partial\mathcal{L}}{\partial\boldsymbol{\omega}_{B/I}^{B}}\right) = \frac{d}{dt}\left(\mathbf{I}_{tot}\boldsymbol{\omega}_{B/I}^{B} + \mathbf{I}_{w}\boldsymbol{\omega}_{w/B}^{B} + \left[\rho_i\int_0^{L_i}(d_i+l)\left[\hat{\boldsymbol{x}}_i^{B}\right]_{\times}\hat{\boldsymbol{z}}_i^{B}\boldsymbol{\phi}_i(l)^{\top}dl\right]\dot{\boldsymbol{\eta}}_i(t)\right) = 0 \quad (3.16)$$

Performing the time derivative, and remembering the transport theorem (Eq. (C.8)) once again, yields:

$$\mathbf{I}_{tot}\dot{\boldsymbol{\omega}}_{B/I}^{B} + \mathbf{I}_{w}\dot{\boldsymbol{\omega}}_{w/B}^{B} + \sum_{i=1}^{N_{ap}}\underbrace{\left[\rho_i\int_0^{L_i}(d_i+l)\left[\hat{\boldsymbol{x}}_i^{B}\right]_{\times}\hat{\boldsymbol{z}}_i^{B}\boldsymbol{\phi}_i(l)^{\top}dl\right]}_{\text{coupling between rigid-body and flexible appendage}}\ddot{\boldsymbol{\eta}}_i(t) + \\ + \left[\boldsymbol{\omega}_{B/I}^{B}\right]_{\times}\left(\mathbf{I}_{tot}\boldsymbol{\omega}_{B/I}^{B} + \mathbf{I}_{w}\boldsymbol{\omega}_{w/B}^{B} + \sum_{i=1}^{N_{ap}}\underbrace{\left[\rho_i\int_0^{L_i}(d_i+l)\left[\hat{\boldsymbol{x}}_i^{B}\right]_{\times}\hat{\boldsymbol{z}}_i^{B}\boldsymbol{\phi}_i(l)^{\top}dl\right]}_{\text{coupling between rigid-body and flexible appendage}}\dot{\boldsymbol{\eta}}_i(t)\right) = 0 \quad (3.17)$$

The coupling between the rigid-body and flexible appendages is determined by the time-invariant matrix $\boldsymbol{\Gamma}_i$ for each appendage:

$$\boldsymbol{\Gamma}_i^{\top} = \rho_i\int_0^{L_i}(d_i+l)\left[\hat{\boldsymbol{x}}_i^{B}\right]_{\times}\hat{\boldsymbol{z}}_i^{B}\boldsymbol{\phi}_i^{\top}(l)dl \quad (3.18)$$

and considering the input to the system, $\boldsymbol{u}$, to be the change in angular momentum of the reaction wheels:

$$\mathbf{I}_{w}\dot{\boldsymbol{\omega}}_{w/B}^{B} = \boldsymbol{u} \quad (3.19)$$

the ODE for the rigid-body dynamics can be given with some simplifications to the notation:

$$\mathbf{I}_{tot}\dot{\boldsymbol{\omega}}_B + \boldsymbol{u} + \sum_{i=1}^{N_{ap}} \boldsymbol{\Gamma}_i^\top \ddot{\boldsymbol{\eta}}_i(t) + [\boldsymbol{\omega}_B]_\times \left(\mathbf{I}_{tot}\boldsymbol{\omega}_B + \mathbf{I}_w\boldsymbol{\omega}_w + \sum_{i=1}^{N_{ap}} \boldsymbol{\Gamma}_i^\top \dot{\boldsymbol{\eta}}_i\right) = 0 \tag{3.20}$$

3.1.3 | Assumed Modes Model of a Flexible Beam Appendage

The dynamics of each flexible appendage can similarly be found through the Euler-Lagrange equation:

$$\frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{\boldsymbol{\eta}}_i(t)}\right) - \frac{\partial \mathcal{L}}{\partial \boldsymbol{\eta}_i(t)} = \frac{d}{dt}\left(\mathbf{M}_i\dot{\boldsymbol{\eta}}_i(t) + \boldsymbol{\Gamma}_i\boldsymbol{\omega}_{B/I}^B\right) + \mathbf{K}_i\boldsymbol{\eta}_i(t) = 0 \tag{3.21}$$

differentiation, and some simplification to the notation to give:

$$\mathbf{M}_i\ddot{\boldsymbol{\eta}}_i(t) + \boldsymbol{\Gamma}_i\dot{\boldsymbol{\omega}}_B + \mathbf{K}_i\boldsymbol{\eta}_i(t) = 0 \tag{3.22}$$

The developed ODEs in Eqs. (3.20) and (3.22) therefore represents the full coupled rigid-flexible dynamic behaviour of a satellite with flexible appendages. Comparing the form of the ODEs to those presented in literature [68, 103, 165–167] provides some verification for the model, and confirms its validity.

The final step for defining the modal dynamics is determining the modal shapes of the appendage, which in the case of a beam bending about one axis, can be defined as functions of the longitudinal position l:

$$\boldsymbol{\phi}_i(l) = \begin{bmatrix}\phi_{i,1}(l) & \phi_{i,2}(l) & \cdots & \phi_{i,m}(l) & \cdots & \phi_{i,N_m}(l)\end{bmatrix}^\top \tag{3.23}$$

and can be defined through an appropriate discretisation method, such as those described in Section 1.2.1. An assumed-modes approach is adopted for a simple, continuous, *Euler-Bernoulli* beam representation of the appendage. The modal shape of the m^{th} mode of the i^{th} appendage can be found analytically [168]:

$$\phi_{i,m}(l) = \left(\cosh\frac{\gamma_{i,m}l}{L_i} - \cos\frac{\gamma_{i,m}l}{L_i}\right) - \frac{\cosh\gamma_{i,m} + \cos\gamma_{i,m}}{\sinh\gamma_{i,m} + sin\gamma_{i,m}}\left(\sinh\frac{\gamma_{i,m}l}{L_i} - \sin\frac{\gamma_{i,m}l}{L_i}\right) \tag{3.24}$$

where $\gamma_{i,m}$ is the solution to the equation:

$$\cos\gamma_{i,m}\cosh\gamma_{i,m} + 1 = 0 \tag{3.25}$$

which garners an infinite number of solutions corresponding to the infinite number of natural modal frequencies that exist, $\omega_n^{i,m}$:

$$\omega_n^{i,m} = \gamma_{i,m}^2 \sqrt{\frac{(EI)_i}{\rho_i L_i^4}} \tag{3.26}$$

For a selected number of modes N_m, the modal shapes can be determined using Eq. (3.24), and Eq. (3.18) can be integrated to obtain $\mathbf{\Gamma}_i$. Thus, a full definition of the hybrid coordinate model for continuous *Euler-Beroulli* beam appendages using the assumed-modes approach has been completed.

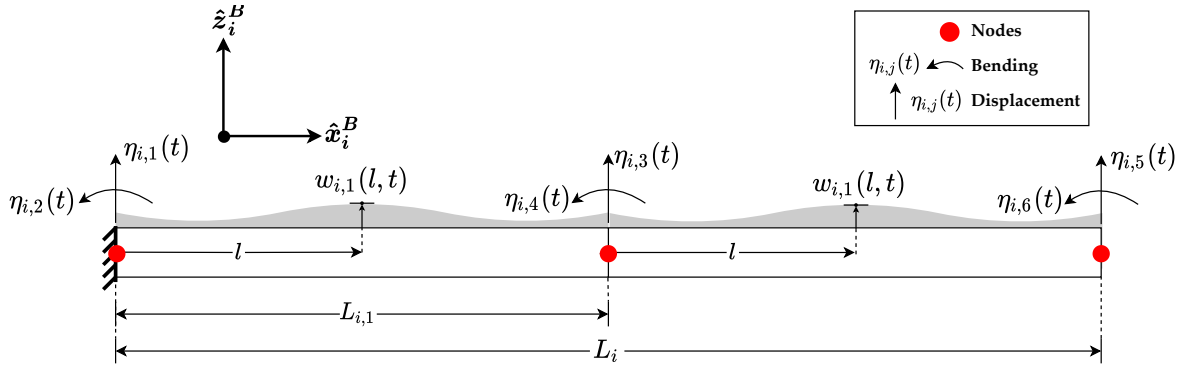


Figure 3.3: FEM appendage: two-element example [161]

3.1.4 | Finite Element Method (FEM) Model of a Flexible Beam Appendage

Later in this chapter, investigations into the effects of model fidelity and accuracy will be performed, therefore an alternative finite element method (FEM) discretisation approach is used to construct a representation of the flexible appendages. Finite beam elements will be used to create suitable mass ($\mathbf{M}_i$), stiffness ($\mathbf{K}_i$), and coupling matrices ($\mathbf{\Gamma}_i$) that can be used with the previously derived ODEs found in Eqs. (3.20) and (3.22). An example of a two-element beam can be seen in Fig. 3.3 with nodes attached to the ends of each finite element.

For the FEM model, the vector $\boldsymbol{\eta}_i$ will now correspond to the displacement or bending at each node seen in Fig. 3.3. Previously, in the continuous model, $\boldsymbol{\eta}_i$ corresponded to the contribution of each mode to the overall deflection in the appendage. The notation is used here interchangeably simply to maintain a consistent form of ODEs as shown in Eqs. (3.20) and (3.22).

With this modified definition of $\boldsymbol{\eta}_i$, the overall deflection across the j^{th} finite element of the i^{th} appendage can be given as [168]:

$$w_{i,j}(l,t) = \boldsymbol{\phi}_{i,j}(l)^\top \boldsymbol{\Lambda}_{i,j} \boldsymbol{\eta}_i(t) \tag{3.27}$$

where the transformation matrix $\boldsymbol{\Lambda}_{i,j}$ selects the nodal displacements and bending in $\boldsymbol{\eta}_i$ that affect the j^{th} element. An example of a two-element appendage, as seen in Fig. 3.3, is:

$$\boldsymbol{\Lambda}_{i,1} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \tag{3.28a}$$

$$\boldsymbol{\Lambda}_{i,2} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.28b}$$

The deflection across each element of the beam is assumed to be cubic, giving the shape function $\boldsymbol{\phi}(l)_{i,j}$ for each element [168]:

$$\boldsymbol{\phi}(l)_{i,j} = \begin{bmatrix} 1 - 3\left(\frac{l}{L_{i,j}}\right)^2 + 2\left(\frac{l}{L_{i,j}}\right)^3 \\ l - 2l_{i,j}\left(\frac{l}{L_{i,j}}\right)^2 + L_{i,j}\left(\frac{l}{L_{i,j}}\right)^3 \\ 3\left(\frac{l}{L_{i,j}}\right)^2 - 2\left(\frac{l}{L_{i,j}}\right)^3 \\ -L_{i,j}\left(\frac{l}{L_{i,j}}\right)^2 + L_{i,j}\left(\frac{l}{L_{i,j}}\right)^3 \end{bmatrix} \tag{3.29}$$

where $L_{i,j}$ is the length of each element as seen in Fig. 3.3:

$$L_{i,j} = \frac{L_i}{N_e} \tag{3.30}$$

with N_e being the number of finite elements for the appendage. Previously, with the continuous beam, the number of shape functions corresponded directly to the number of modes considered, with each defining the deflection across the entire length of the appendage. However, for the FEM model, only four shape functions will ever be needed. The deflection across the entire length of the appendage in this case will be a piecewise cubic interpolation between each node seen in Fig. 3.3.

The spring stiffness of each element $\mathbf{K}_{i,j}$ is defined as [168]:

$$\mathbf{K}_{i,j} = \frac{(EI)_i}{L_{i,j}^3} \begin{bmatrix} 12 & 6L_{i,j} & -12L_{i,j} & 6L_{i,j} \\ 6L_{i,j} & 4L_{i,j}^2 & -6L_{i,j} & 2L_{i,j}^2 \\ -12 & -6L_{i,j} & 12 & -6L_{i,j} \\ 6L_{i,j} & 2L_{i,j}^2 & -6L_{i,j} & L_{i,j}^2 \end{bmatrix} \tag{3.31}$$

with the element mass matrix $\mathbf{M}_{i,j}$ similarly [168]:

$$\mathbf{M}_{i,j} = \frac{\rho_i L_{i,j}}{420} \begin{bmatrix} 156 & 22L_{i,j} & 54 & -13L_{i,j} \\ 22L_{i,j} & 4L_{i,j}^2 & 13L_{i,j} & -3L_{i,j}^2 \\ 54 & 13L_{i,j} & 156 & -22L_{i,j} \\ -13L_{i,j} & -3L_{i,j}^2 & -22L_{i,j} & 4L_{i,j}^2 \end{bmatrix} \tag{3.32}$$

Using the transformation matrices $\mathbf{\Lambda}_{i,j}$ we can get the overall stiffness and mass matrices:

$$\mathbf{K}_i = \sum_{j=1}^{N_e} \mathbf{\Lambda}_{i,j}^\top \mathbf{K}_{i,j} \mathbf{\Lambda}_{i,j} \tag{3.33}$$

$$\mathbf{M}_i = \sum_{j=1}^{N_e} \mathbf{\Lambda}_{i,j}^\top \mathbf{M}_{i,j} \mathbf{\Lambda}_{i,j} \tag{3.34}$$

The cantilever boundary conditions can then be applied by simply constraining the first node's displacement and bending, as seen in Fig. 3.3, by deleting the first two rows and columns of $\mathbf{K}_i$ and $\mathbf{M}_i$. The first two columns of $\mathbf{\Lambda}_{i,j}$ are also deleted. Finally, the coupling between the rigid and flexible dynamics for the FEM model can be found by considering contributions from each finite element:

$$\mathbf{\Gamma}_i = \left(\rho_i \int_0^{L_{i,j}} \left[\sum_{j=1}^{N_e} \left(d_i + (j-1)L_{i,j} + l \right) \mathbf{\Lambda}_{i,j}^\top \right] \boldsymbol{\phi}(l) dl \right) \left[\hat{\boldsymbol{x}}_i^B \right]_\times \hat{\boldsymbol{z}}_i^B \tag{3.35}$$

Comparison to the Assumed Modes Model

The completed FEM model can then be deconstructed using the eigenvalue problem to determine the natural frequencies of the appendage:

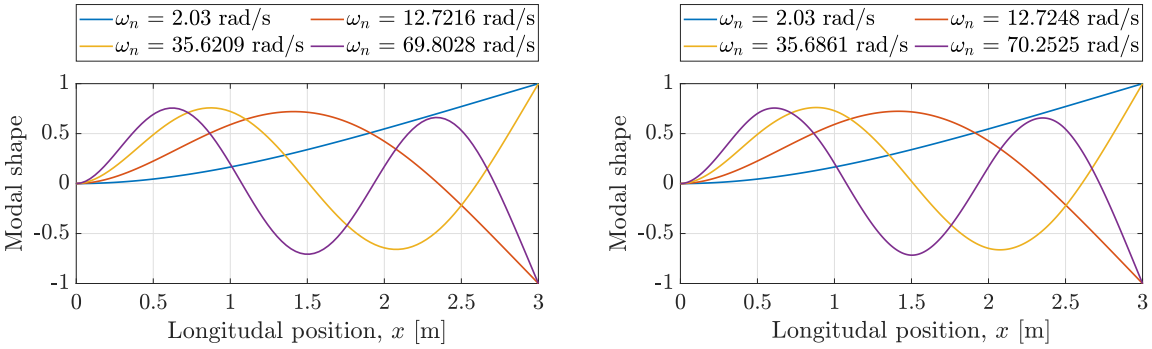
$$\lambda_{i,m} \boldsymbol{\eta}_{i,m} = -\mathbf{M}_i^{-1} \mathbf{K}_i \boldsymbol{\eta}_{i,m} \tag{3.36}$$

where $\boldsymbol{\eta}_{i,m}$ is the eigenvector, and $\lambda_{i,m}$ the eigenvalue corresponding to the m^{th} flexible mode the appendage. The natural frequency of each mode $\omega_n^{i,m}$ can then be found:

$$\omega_n^{i,m} = ||\sqrt{\lambda_{i,m}}|| \tag{3.37}$$

The corresponding eigenvectors $\boldsymbol{\eta}_{i,m}$ can then be substituted into Eq. (3.27) to find the shape of each mode across the entire appendage length L_i. Using the example parameters given in Table 3.1 the natural frequencies and normalised modal shapes of a continuous beam appendage, and a FEM beam appendage with six elements ($N_e = 6$),

can be compared as seen in Fig. 3.4. It can therefore be seen that the modal response of the FEM model has been verified against the analytical continuous model.



(a) Continuous beam **(b)** FEM beam $N_e = 6$

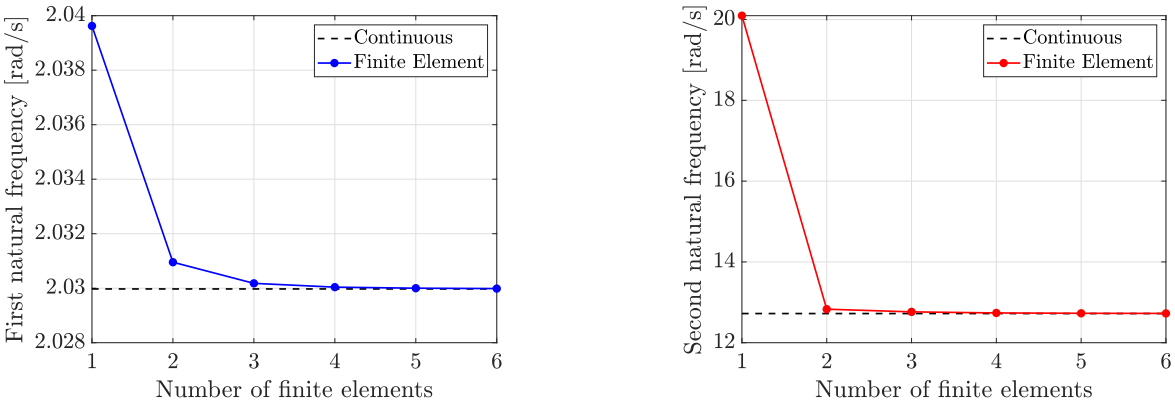
Figure 3.4: Comparison of first four normalised modal shapes of continuous and FEM beams [161]

As the number of finite elements used increases, the number of flexible modes captured by the model increases. The higher-order modes will however have much less impact on the attitude dynamics of the satellite and, therefore, special attention is paid to the convergence of the first two modes of the FEM model.

Table 3.1: Model parameters of a satellite with flexible appendages

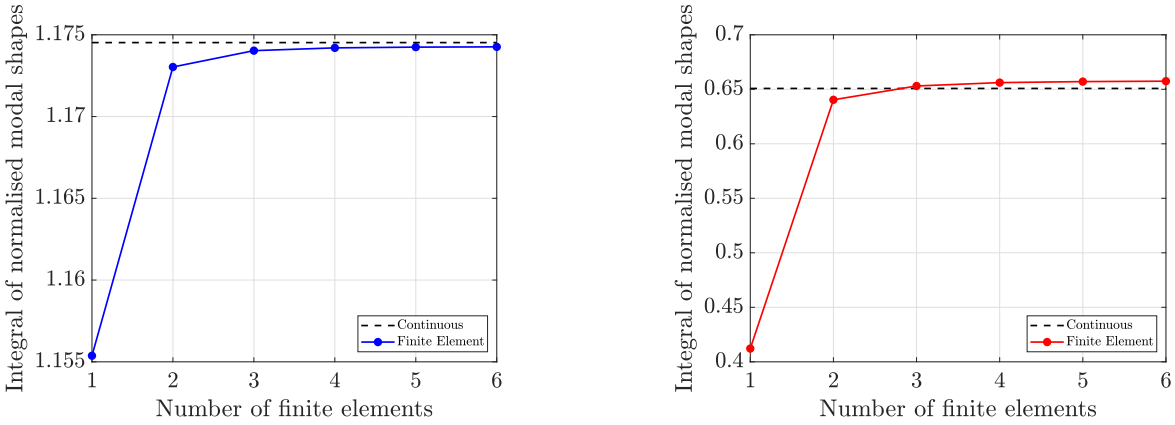
Parameter		**Value**
Number of appendages	N_{ap}	2
Appendage mass per unit length	ρ_i	33.3 kg/m
Appendage length	L_i	3 m
Appendage total mass	m_i	100 kg
Appendage root distance from centre of main hub	d_i	1 m
Main hub inertia	$\mathbf{I}_{hub}$	$\begin{bmatrix} 500 & -10 & -20 \\ -10 & 400 & -30 \\ -20 & -30 & 350 \end{bmatrix}$ kgm^2
Unit vectors of each appendage longitudinal axes	$[\hat{\boldsymbol{x}}_1^B \ \hat{\boldsymbol{x}}_2^B]$	$\begin{bmatrix} -1 & 1 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$
Unit vectors of appendage bending axes	$[\hat{\boldsymbol{y}}_1^B \ \hat{\boldsymbol{y}}_2^B]$	$\begin{bmatrix} 0 & 0 \\ 1 & -1 \\ 0 & 0 \end{bmatrix}$
Reaction wheel distribution matrix	$\mathbf{A}_w$	$\begin{bmatrix} 1 & 0 & 0 & 1/\sqrt{3} \\ 0 & 1 & 0 & 1/\sqrt{3} \\ 0 & 0 & 1 & 1/\sqrt{3} \end{bmatrix}$
Appendage bending stiffness	$(EI)_i$	900 Nm^2

The convergence of the natural frequencies of the first and second modes can be seen in Fig. 3.5. After the 3$^{\text{rd}}$ or 4$^{\text{th}}$ finite elements, there is relatively not much further



(a) First mode **(b)** Second mode

Figure 3.5: Natural frequency convergence of first two flexible modes [161]



(a) First mode **(b)** Second mode

Figure 3.6: Modal shape convergence of first two flexible modes [161]

improvement to the approximation of the first two modal frequencies. This will be worth considering when viewing the results of an InvSim-based controller developed later in this chapter.

Next, the first two modal shapes for both the continuous and FEM models are integrated across the appendage and compared in Fig. 3.6. The modal shapes will affect the coupling between the rigid and flexible modes and, therefore, may have an impact on the attitude-tracking performance. It can be seen again that after the 3^{rd} or 4^{th} element is added, the accuracy of the first two modal shapes does not change much. It can be seen, however, that the modal shapes do not converge as closely to the continuous model as the natural frequencies did. This is most likely due to the limitation of the assumed cubic shape of each element. It will be seen later in this chapter if this discrepancy has any detrimental effect on the developed InvSim-based controller. Before developing the InvSim-based attitude controller, the effect that including these flexible modes has on the numerical performance and stability of InvSim will be investigated.

3.2 | Inverse Simulation of Satellites with Flexible Appendages

At its heart, the integral InvSim algorithm, as is illustrated in Section 2.1, is simply a series of forward simulations being performed over the discretisation timestep T. This requires numerical integration of the governing ODEs, which in the case of the rigid-body satellite was performed using the popular RK4 explicit integration method (Appendix D.3). However, with the introduction of the appendage's flexible modes, numerical integration stability issues linked to the *stiffness* of the governing ODEs are encountered.

The definition of a *stiff* system can be rather ambiguous [59, p.67], however, one useful definition put forward by Lambert [169] is:

> *If a numerical method with a finite region of absolute stability, applied to a system with any initial conditions, is forced to use in a certain interval of integration a steplength which is excessively small in relation to the smoothness of the exact solution in that interval, then the system is said to be stiff in that interval*
>
> **- Lambert, 1991**

This effectively means that a dynamic system can be defined as stiff if the integration step size is required to be small enough to ensure numerical stability, rather than its size being dictated by the accuracy that the user desires. How stiff a system is will be defined by how fast its dynamics are, or rather, what the frequency of the highest frequency mode is.

The zero-frequency rigid-body dynamics of a satellite, as covered in Chapter 2, are by definition *non-stiff* and, therefore, could be solved using RK4. However, RK4 has a bounded stability region and can become numerically unstable for stiff systems, should the integration step size Δt be too large.

Implicit integration methods, such as the trapezoidal implicit Runge-Kutta (IRK) method (Appendix D.3), are however, known to provide some benefits over more straightforward explicit Runge-Kutta (ERK) methods in specific applications such as solving stiff systems [59, p.33]. The use of IRK methods can relieve the strict requirements on the integration step size Δt, with larger step sizes possibly improving the computational efficiency of InvSim for stiff systems.

The computational efficiency of InvSim will have significant implications when considering it for use within an attitude control architecture, or an optimisation problem, as will be shown later in this chapter. This section will therefore endeavour to investigate

the numerical stability of both explicit RK4 and implicit trapezoidal integration (Appendix D.3), before looking at the impact each has on the computational efficiency of the InvSim for satellites with flexible appendages.

3.2.1 | Numerical Integration Stability

Here, the stability regions of each integration method will be derived. Consider first the general form for a linear time-invariant (LTI) dynamical system with no inputs:

$$\dot{\boldsymbol{x}} = \mathbf{A}\boldsymbol{x} \tag{3.38}$$

where $\mathbf{A}$ is the system matrix, and $\dot{\boldsymbol{x}}$ and $\boldsymbol{x}$ are the state derivative and state vectors respectively. To simplify the stability analysis of each integration method, only one mode will be considered at a time. This can be done simply by considering the eigenvalue problem:

$$\mathbf{A}\boldsymbol{x}_i = \lambda_i \boldsymbol{x}_i \tag{3.39}$$

where λ_i and is the eigenvalue for the i^{th} mode with corresponding eigenvector $\boldsymbol{x}_i$ within the state space. Considering any arbitrary mode, the scalar state derivative can be given as [170]:

$$\dot{x} = \lambda x \ , \ x = x_0 \text{ when } t = 0 \tag{3.40}$$

which will have an analytical solution:

$$x(t) = x_0 e^{\lambda t} \tag{3.41}$$

Explicit Runge-Kutta 4$^{\text{th}}$ Order Integration (RK4)

Next consider the solution obtained by applying RK4 integration (Appendix D.3) to a scalar dynamic system such as that defined in Eq. (3.40):

$$x_{k+1} = x_k + \Delta t \left(\frac{1}{6}k_1 + \frac{1}{3}k_2 + \frac{1}{3}k_3 + \frac{1}{6}k_4 \right) \tag{3.42}$$

where Δt is the integration step size and each weighted element can be found through the application of Eq. (3.40):

$$k_1 = \lambda x_k \tag{3.43a}$$

$$k_2 = \lambda x_k + \frac{1}{2}\lambda \Delta t^2 x_k \tag{3.43b}$$

$$k_3 = \lambda x_k + \frac{1}{2}\lambda \Delta t^2 x_k + \frac{1}{4}\Delta t^2 \lambda^3 x_k \tag{3.43c}$$

$$k_4 = \lambda x_k + \lambda \Delta t^2 x_k + \frac{1}{2}\Delta t^2 \lambda^3 x_k + \frac{1}{4}\Delta t^3 \lambda^4 x_k \tag{3.43d}$$

Substituting Eq. (3.43) into Eq. (3.42) therefore gives the state transition:

$$x_{k+1} = \left(1 + \Delta t\lambda + \frac{1}{2}\Delta t^2\lambda^2 + \frac{1}{6}\Delta t^3\lambda^3 + \frac{1}{24}\Delta t^4\lambda^4\right) x_k = \phi(\lambda\Delta t) x_k \tag{3.44}$$

where $\phi(\lambda\Delta t)$ is the stability function:

$$\phi(\lambda\Delta t) = 1 + \lambda\Delta t + \frac{1}{2}(\lambda\Delta t)^2 + \frac{1}{6}(\lambda\Delta t)^3 + \frac{1}{24}(\lambda\Delta t)^4 \tag{3.45}$$

It can be seen that the stability function must satisfy $|\phi(\lambda\Delta t)| < 1$ for the solution to converge and avoid instability in the numerical integration process. Therefore the region of stability for RK4 can be defined on the complex $\lambda\Delta t$ plane as:

$$\left\{\lambda \in \mathbb{C}\ ,\ \Delta t \in \mathbb{R}\ ,\ \Delta t > 0\ ,\ \left|1 + (\lambda\Delta t) + \frac{1}{2}(\lambda\Delta t)^2 + \frac{1}{6}(\lambda\Delta t)^3 + \frac{1}{24}(\lambda\Delta t)^4\right| < 1\right\} \tag{3.46}$$

which can be seen plotted in Fig. 3.7. The integration method can be determined to be stable if when the eigenvalues λ_i of the system matrix $\mathbf{A}$ are multiplied by the chosen integration step size Δt, they all lie within the stability region found in Fig. 3.7 [170].

Implicit Trapezoidal Integration

A similar definition of the stability region of implicit trapezoidal integration can also be found through a similar process. The solution for a scalar dynamic system, such as that in Eq. (3.40), using trapezoidal integration (Appendix D.3) can be given as:

$$x_{k+1} = \boldsymbol{x}_k + \frac{1}{2}\Delta t\,(\dot{x}_k + \dot{x}_{k+1}) \tag{3.47}$$

Substituting Eq. (3.40) into Eq. (3.47) gives:

$$x_{k+1} = x_k + \frac{1}{2}\lambda\Delta t x_k + \frac{1}{2}\lambda\Delta t x_{k+1} \tag{3.48}$$

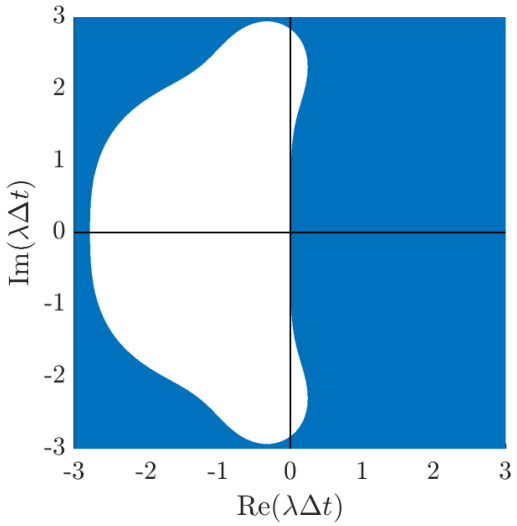


Figure 3.7: Stability region for explicit RK4 integration. Area of stability (white), and instability (shaded blue)

which can be solved through collection of x_{k+1} terms:

$$x_{k+1} = \frac{1 + \frac{1}{2}\lambda\Delta t}{1 - \frac{1}{2}\lambda\Delta t} x_k = \phi(\lambda\Delta t) x_k \tag{3.49}$$

where the stability function $\phi(\lambda\Delta t)$ is given as:

$$\phi(\lambda\Delta t) = \frac{1 + \frac{1}{2}\lambda\Delta t}{1 - \frac{1}{2}\lambda\Delta t} \tag{3.50}$$

which defines that stability region on the $\lambda\Delta t$ plane for implicit trapezoidal integration:

$$\left\{ \lambda \in \mathbb{C}\ ,\ \Delta t \in \mathbb{R}\ ,\ \Delta t > 0\ ,\ \left| \frac{1 + \frac{1}{2}\lambda\Delta t}{1 - \frac{1}{2}\lambda\Delta t} \right| < 1 \right\} \tag{3.51}$$

which can be seen plotted in Fig. 3.8. It can be seen that the stability region of implicit trapezoidal integration encompasses the entire left-hand side of the $\lambda\Delta t$ plane. Therefore, the numerical integration of any stable dynamic mode λ_i (i.e. lies on the left-hand side of the complex plane) using the implicit trapezoidal method will be stable for any $\Delta t > 0$.

Numerical Integration Stability for a Satellite with Flexible Appendages

To illustrate the significance of the stability regions and requirements for each integration method, the FEM appendage flexible satellite model (Section 3.1.4) will be used. It can be seen in Fig. 3.7 that the RK4 integration stability region only partially cov-

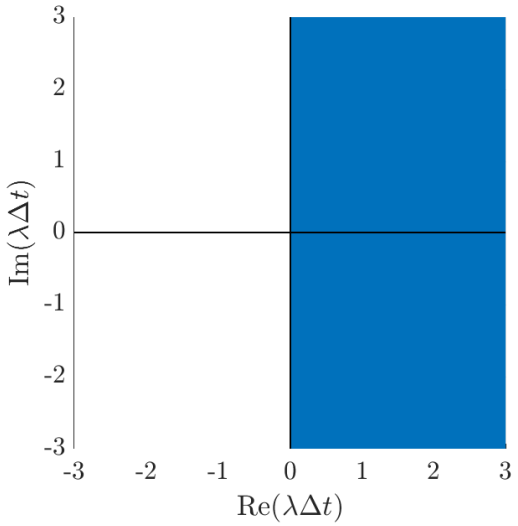


Figure 3.8: Stability region for implicit 2nd order runge-kutta (trapezoidal) integration. Area of stability (white), and instability (shaded)

ers the left-hand side of the complex plane, with high-frequency dynamics (poles far from the origin), causing instability in the numerical process. Reducing the integration step size Δt will bring the poles of the dynamic modes closer to the origin of Fig. 3.7. Higher-frequency dynamics can therefore be brought into the region of stability given sufficiently small Δt. A *rule of thumb* can be used to find a suitable integration step size Δt that brings the dynamic poles within the stability region on the $\lambda \Delta t$ plane:

$$\Delta t \approx \frac{1}{10} \frac{2\pi}{\max\limits_{i,m}(\omega_n^{i,m})} \tag{3.52}$$

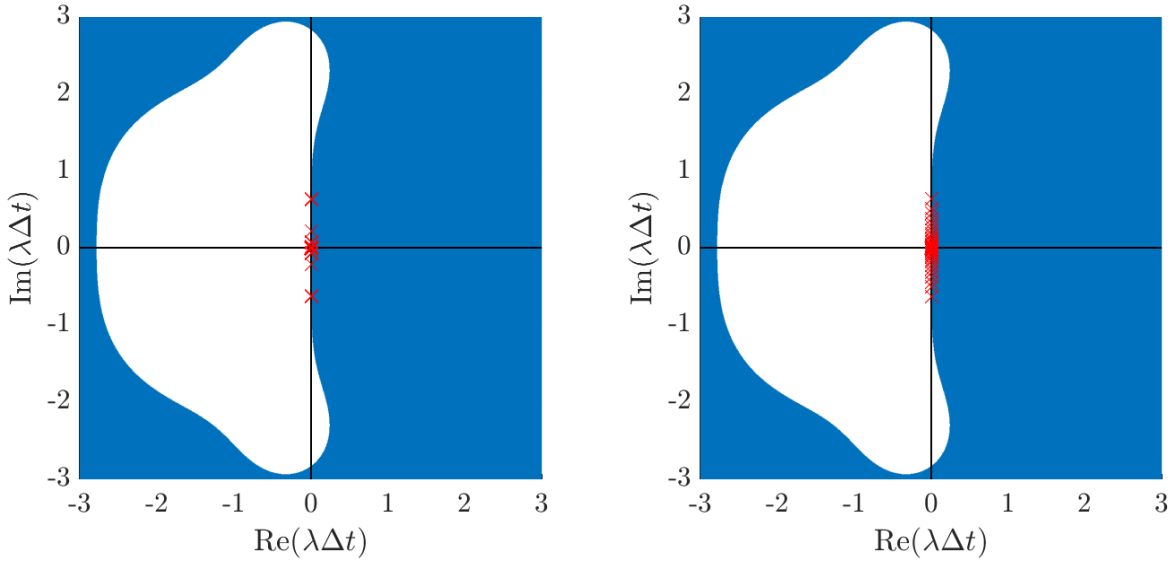
where $\omega_n^{i,m}$ is the natural frequency of the m^{th} mode of the i^{th} appendage of the FEM appendage model (Eq. (3.37)). Eq. (3.52) ensures that there are at least 10 integration steps within one period of the system's fastest dynamics. Consider a satellite with two flexible, FEM appendages defined by the parameters given previously in Table 3.1. However, instead of using 4 reaction wheels, as determined by the distribution matrix $\mathbf{A}_w$ in Table 3.1, a 3-wheel configuration is adopted:

$$\mathbf{A}_w = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.53}$$

which simplifies some of the analysis in this section.

The poles of the dynamics can be seen plotted on the $\lambda \Delta t$ plane, assuming an integration timestep for RK4, as per Eq. (3.52), for FEM appendages with $N_{el} = 2$ and $N_{el} = 10$

in Fig. 3.9. The poles of the FEM appendage model all lie along the imaginary axis, due to there being zero damping, and within the stability region, as shown in Fig. 3.9, confirming that RK4 numerical integration will be stable given a sufficiently small value of Δt.



(a) $N_{el} = 2$, $\Delta t = 5 \times 10^{-3}$ s **(b)** $N_{el} = 10$, $\Delta t = 1.8 \times 10^{-4}$ s

Figure 3.9: Flexible satellite poles plotted on the $\lambda\Delta t$ plane shown to lie within the stability region for explicit RK4 integration. Area of stability (white), and instability (shaded blue)

However, as the number of finite elements $N_{el} \to \infty$ the maximum frequency of the flexible modes $\max_{i,m}(\omega_n^{i,m}) \to \infty$, therefore requiring the integration step size for RK4 $\Delta t \to 0$ as per Eq. (3.52) to maintain numerical stability. Reducing the integration timestep Δt will lead to an increase in the total number of times the ODEs describing the dynamics are evaluated (ODE evaluations). This will have the effect of slowing down the forward simulation process, which will also have a knock-on effect on the InvSim speed.

As seen previously in Fig. 3.8, any stable flexible modes will fall within the stability region for implicit trapezoidal integration. Therefore, it can be deduced that any integration step size $\Delta t > 0$ will achieve numerical stability. It will still be desirable to choose a sufficiently small integration step size to provide accurate results; however, unlike RK4, the implicit trapezoidal integration is not constrained by the frequency of the fastest flexible modes. However, the implicit trapezoidal integration will need to be solved iteratively, as the solution to the next timestep x_{k+1} appears on both sides of the equation (Appendix D.3). Each iteration will require two ODE evaluations (Eq. (3.47)), with the number of iterations needed to converge upon a solution for x_{k+1} not being fixed. On the other hand, RK4 will only ever require four ODE evaluations (Eq. (3.42))

to calculate x_{k+1}. There is, therefore, a trade-off in terms of the total number of ODE evaluations, either using explicit RK4 integration with smaller Δt, or using iterative trapezoidal integration with larger Δt.

3.2.2 | Speed of Inverse Simulation

The trade-off between the use of ERK and iterative IRK numerical integration in terms of InvSim speed for stiff systems is now investigated. A FEM appendage flexible satellite model, defined in Section 3.1, is used with parameters defined previously in Table 3.1. Each InvSim will solve a 90-deg, 60-second, rest-to-rest, polynomial, SLERP slew manoeuvre about the body $\hat{\boldsymbol{y}}^B$ axis (Appendix B.4).

InvSim Speed for Different Model Fidelities

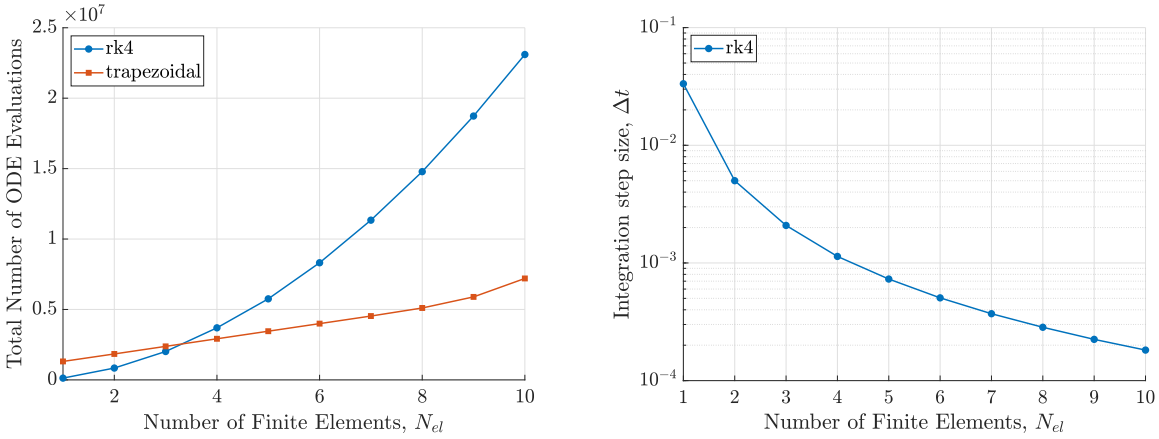
Increasing the model fidelity, i.e. the number of finite elements N_{el} in each appendage, will include more flexible modes of increasing frequency. The effect of this on the speed of InvSim when using RK4 explicit integration and trapezoidal implicit integration is now investigated.

In each case, a stable integration step size Δt can be determined for RK4 using Eq. (3.52) with the implicit trapezoidal integration always using a step size $\Delta t = 0.1$ s. The remaining InvSim parameters used can be found in Table 3.2. Additionally, the tolerance for the implicit trapezoidal integration is set to one order of magnitude lower than the InvSim tolerance ($\delta\dot{\boldsymbol{y}}^{(\nu)}_{tol}/10$) to ensure sufficient accuracy. The computational speed of the InvSim process is quantified as the total number of ODE evaluations performed, which can be seen for each integration method and a variety of finite elements per appendage N_{el} in Fig. 3.10a.

Table 3.2: InvSim numerical integration speed comparison: InvSim hyperparameters

Hyperparameter	Value
Discretisation timestep, T	0.1 s
NR tolerance, $\delta\dot{\boldsymbol{y}}^{(\nu)}_{tol}$	1×10^{-10}
Maximum NR iterations, n_{max}	30
Minimum finite difference perturbation, Δu_{min}	1 Nm
Desired output, $\dot{\boldsymbol{y}}^{(\nu)}_d$	$\dot{\boldsymbol{q}}^{(\nu)B/I}_{d,1:3}$
Desired output order, ν	1 (velocity)

When only one finite element is used to represent each flexible appendage ($N_{el} = 1$), the implicit trapezoidal integration proves to be slower than explicit RK4 integration. This is despite the integration step size Δt for RK4 (Fig. 3.10b) being smaller than trape-



(a) Number of ODE evaluations for each integration method

(b) Integration step size required by explicit RK4 integration for increasing number of finite elements

Figure 3.10: Effect of number of finite elements N_{el} on the speed of InvSim for explicit RK4 and implicit trapezoidal integration methods. $\Delta t = 0.1$ s for trapezoidal.

zoidal ($\Delta t = 0.1$ s), as the latter requires a significant number of iterations to converging upon $\boldsymbol{x}_{k+1}$. This remains the case until $N_{el} = 4$ finite elements are included in each appendage, where trapezoidal integration begins to become faster than RK4. The integration step size Δt (Fig. 3.10b) has to be reduced substantially for RK4 to maintain numerical stability, such that the total number of ODE evaluations exceeds that of the trapezoidal method. It can be seen in Fig. 3.10a that the number of ODE evaluations for trapezoidal integration still increases as the fidelity ($N_{el} \to \infty$). This is due to the increased complexity of the model requiring the implicit trapezoidal integration to perform more iterations before converging upon $\boldsymbol{x}_{k+1}$ within the prescribed tolerance ($\delta \dot{y}_{tol}^{(\nu)}/10$). However, the stringent requirement on the integration step size Δt for RK4 causes the ODE evaluations to grow much more rapidly than trapezoidal.

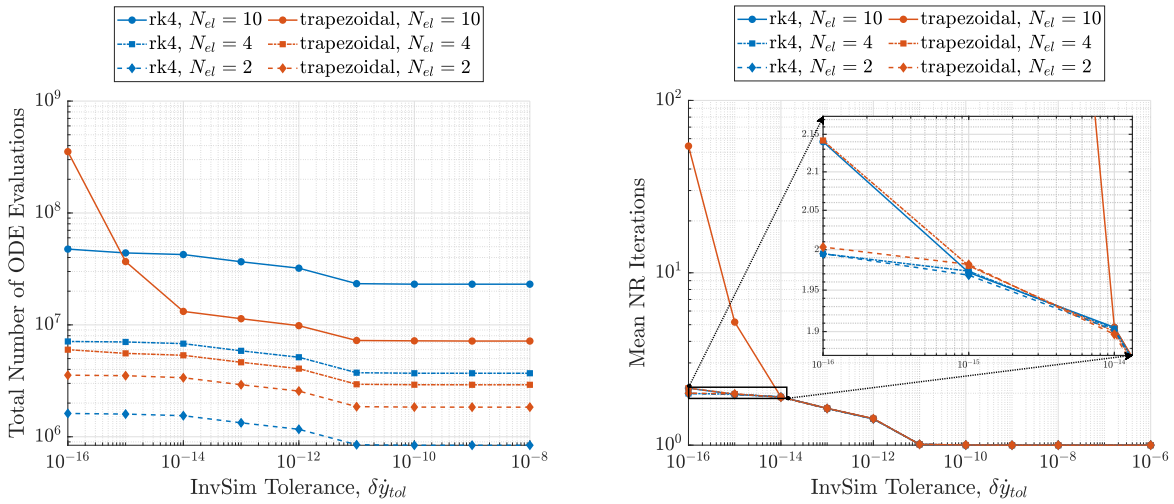
InvSim Speed for Different NR Tolerances

It is also important to consider the effect that the selected NR tolerance of the InvSim will have on the computational efficiency of each integration method. It can be hypothesised that as the NR tolerance becomes more stringent the RK4 integration, with its smaller and more accurate step size Δt, may require fewer iterations of the NR loop when compared to the trapezoidal method. Requiring more NR iterations, the trapezoidal method may become less computationally efficient than RK4 in cases where it previously outperformed it, as it did in Fig. 3.10a.

The total number of ODE evaluations when performing InvSim for FEM appendage models of fidelity $N_{el} = 2$, 4 and 10 can be seen in Fig. 3.11a for a variety of NR tolerance values. For the $N_{el} = 2$ and $N_{el} = 4$ cases, the trend is similar to what was seen

previously in Fig. 3.10a, with RK4 and trapezoidal integration being quicker in each case respectively. Fig. 3.11a also shows that these results are insensitive to the NR tolerance chosen, with the number of finite elements, or the *stiffness* of the system, being the only contributing factor.

Looking more closely at the case when $N_{el} = 4$, it can be seen that the speed differential between trapezoidal and RK4 narrows as the NR tolerance is reduced to 10^{-16}, with trapezoidal however, still performing best. Increasing the number of finite elements in each appendage further to $N_{el} = 10$ causes this behaviour to become more pronounced, with the total number of ODE evaluations for trapezoidal integration rapidly approaching and then exceeding that of RK4 as the NR tolerance approaches 10^{-16}. Considering Fig. 3.11b, it can be seen that this degradation in performance of trapezoidal integration at lower NR tolerances is directly linked to the mean number of NR iterations required by the InvSim. Similar trends were found during the hyperparameter study in Section 2.2.2; however, it is clear that the integration method used also has an influence.



(a) Number of ODE evaluations for each integration method

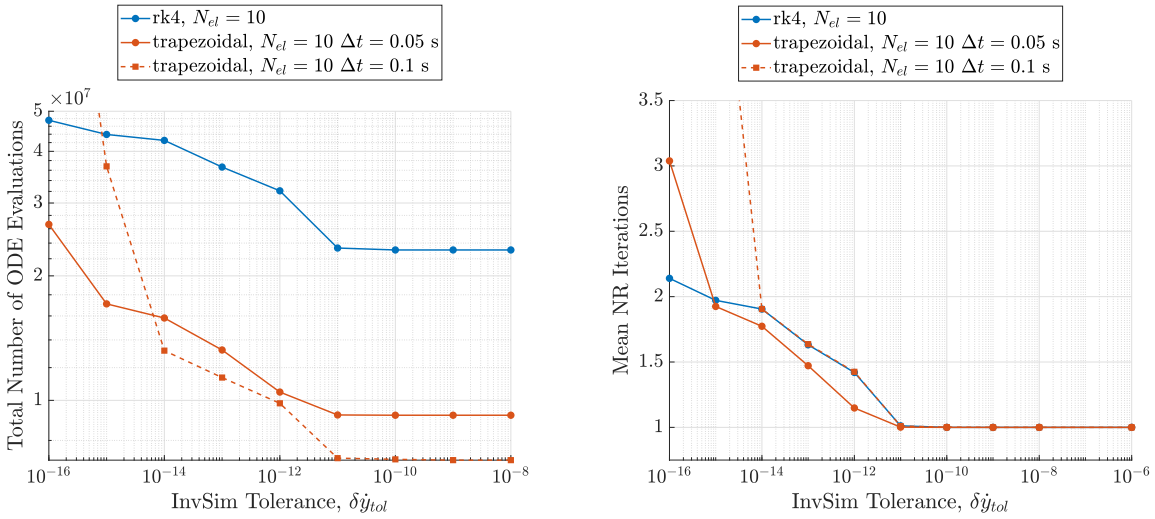
(b) Mean number of NR iterations per Invsim timestep

Figure 3.11: Effect of NR tolerance $\delta\dot{y}_{tol}^{(\nu)}$ on the speed of InvSim for explicit RK4 and implicit trapezoidal integration methods. $\Delta t = 0.1$ s for trapezoidal

The accuracy of any Runge-Kutta method of order r integrating over a fixed time, T in the case of the InvSim, will decay on the order of $\mathcal{O}(\Delta t^r)$ [170, p.8]. The error accumulated during integration over T for RK4 and trapezoidal (2^{nd} order) will decay at rates of $\mathcal{O}(\Delta t^4)$ and $\mathcal{O}(\Delta t^2)$ respectively. Importantly in the test case provided in Fig. 3.11 the step size of trapezoidal integration is fixed at $\Delta t = 0.1$ s while the step size of RK4 is reduced to ensure stability according to Eq. (3.52) when the number of finite elements is increased. This stringent requirement on RK4's integration step size Δt, therefore, has the added effect of improving accuracy, whereas in the case of trapezoidal integration

maintaining a large integration step size Δt, in the hopes of increasing InvSim speed, results in the accuracy remaining constant.

As the number of finite elements N_{el} in the model is increased, introducing higher frequency modes, smaller integration step sizes are needed to retain good accuracy. Therefore, for a high number of finite elements ($N_{el} = 10$), trapezoidal integration, with fixed Δt, will no longer provide sufficiently accurate results to meet ever-decreasing NR tolerances. This is illustrated in Fig. 3.11b where the mean number of NR iterations during the InvSim for trapezoidal integration solving a model with $N_{el} = 10$ increases rapidly as the NR tolerance is reduced below 10^{-14}. As discussed previously, this drastic increase in mean NR iterations will degrade overall InvSim speed.



(a) Number of ODE evaluations for each integration method

(b) Mean number of NR iterations per Invsim timestep

Figure 3.12: Effect of NR tolerance $\delta \dot{y}_{tol}^{(\nu)}$ on the speed of InvSim for explicit RK4 and implicit trapezoidal integration methods. Comparing $\Delta t = 0.1$ s and $\Delta t = 0.05$ s for trapezoidal integration for the $N_{el} = 10$ case

To ensure that trapezoidal integration can provide accurate InvSim solutions that meet stringent NR tolerances, it can be appealing to reduce the integration step size Δt. This leads to another balancing act whereby the step size Δt must be reduced significantly enough to provide sufficiently accurate results, while not being reduced so far as to require too many additional ODE evaluations and slow down the InvSim. An example of this balancing act working can be seen in Fig. 3.12 where the integration step size for trapezoidal integration is reduced to $\Delta t = 0.05$ s for the $N_{el} = 10$ case.

Considering Fig. 3.12b, it can be seen that the reduced integration step size Δt improves the accuracy of trapezoidal integration sufficiently enough to garner a significant improvement in mean NR iterations during the InvSim. This impacts the overall speed of

the InvSim, reducing the total number of ODE evaluations in Fig. 3.12a and remaining faster than RK4, even down to the smallest NR tolerance tested ($\delta\dot{y}_{tol}^{(\nu)} = 10^{-16}$). It can be seen however that for looser NR tolerances ($\delta\dot{y}_{tol}^{(\nu)} > 10^{-14}$) the decrease of the trapezoidal integration step size Δt decreases the speed of the InvSim despite having lower or equal mean NR iterations (Fig. 3.12b). For looser NR tolerances high accuracy of the numerical integration is not needed, allowing for the trapezoidal integration step size to be increased, reducing the number of integration steps within each InvSim discretisation timestep T, and therefore reducing the total number of ODE evaluations and increasing InvSim speed.

Therefore, it can be concluded that the appropriate choice of integration method will depend on both the NR tolerance required, and the stiffness of the dynamic system, i.e. highest frequency mode. Explicit RK4 integration will provide the fastest InvSim for relatively low stiffnesses, as the number of ODE evaluations per integration step is bounded, and the integration step size can be relatively large. Implicit trapezoidal integration can provide faster InvSim speeds when stiffer dynamic modes are present, however, its benefits can be limited should a very low NR tolerance be required. For most of the work in the rest of this chapter, RK4 integration will be used due to its simpler closed-form implementation, and since the speed of the InvSim was not a limiting factor. If, however, the InvSim was required to run in real-time for a highly stiff system, it can be seen that the use of an implicit method, such as trapezoidal integration, would be required. Alternatively, it might be found that a lower fidelity, less stiff model may provide a sufficiently accurate InvSim solution. This will be investigated in the next section of this chapter, but first, the InvSim-based attitude control architecture is developed.

3.3 | Feed-Forward Control Using Inverse Simulation

As discussed in Section 1.2.3, there have been many approaches to the attitude control of satellites with flexible appendages, such as the utilisation of low-jerk attitude manoeuvres to minimise excitation of disturbance inducing flexible modes, degrading attitude tracking performance and stability. Traditional attitude control methodologies can also be used to compensate for the disturbances induced by the flexing appendages, including PD control, SMC, and MPC. An alternative, but common, approach for improved tracking performance is to employ a model-based feedforward control element which can be added to an existing feedback controller [112, 113], and allow compensation of modelled disturbances [114]. Therefore, given a model that sufficiently describes the flexible dynamics of the appendages, it is suggested that the disturbances they induce during attitude-tracking manoeuvres can be compensated for.

This model-based feedforward term can be produced using a suitable model-inversion technique, commonly NDI [100, 171]. It has, however, already been demonstrated in Section 2.4 that deriving the inverse solution through an analytical approach such as NDI can be a labour-intensive task, requiring re-derivation whenever significant changes are made to the model, and can degrade significantly in accuracy when the relative order of the system input-output dynamics increases. At the cost of a greater computational burden, InvSim provides an alternative numerical model-inversion technique that could be used to produce a feedforward control element.

InvSim has been previously explored as a feedforward control element within a model following control architecture to improve output tracking performance of a helicopter, and container ship [146]. The work utilised the InvSim feedforward term with an H^∞ feedback controller, ensuring robustness to measurement noise and output disturbances. This section of the thesis will employ a similar hybrid control strategy, using PD instead of H^∞ as the feedback element, as it is commonly used within the satellite attitude control literature [81, 172].

This section will investigate the stability, performance, and effect of model fidelity and uncertainty on the proposed model-based InvSim feedforward, PD feedback hybrid control scheme; comparisons will be made with a traditional nonlinear PD attitude tracking control law. The model parameters used throughout this section to test each control method are defined previously in Table 3.1. First, however, both control architectures need to be properly defined.

3.3.1 | Attitude-Tracking Control Designs

Two control strategies are detailed herein: a traditional PD tracking controller with reference body acceleration feedforward and nonlinear cross-coupling compensation (Fig. 3.13), and the proposed hybrid control approach combining linear PD control with an InvSim feedforward element. First, the PD tracking controller will be derived with the body rate and attitude quaternion shown to be asymptotically stable, assuming the satellite is rigid [73]. Instability issues arising due to the excitation of the appendage's flexible modes will be investigated later.

Traditional PD Attitude-tracking Control Law

Letting $\boldsymbol{q}^{B/I} = \boldsymbol{q} = [q_0 \ \boldsymbol{q}_{1:3}]^\top$ a traditional PD tracking control law [172] can be defined with simplified notation:

$$\boldsymbol{u}_{PD} = -\mathbf{A}_w^+ \left[-k_p \delta \boldsymbol{q}_{1:3} - \mathbf{K_D} \delta \boldsymbol{\omega}_B + \mathbf{I}_{tot} \left(\dot{\boldsymbol{\omega}}_B^d - [\boldsymbol{\omega}_B]_\times \mathbf{I}_{tot} \boldsymbol{\omega}_B^d \right) + [\boldsymbol{\omega}_B]_\times \left(\mathbf{I}_{tot} \boldsymbol{\omega}_B + \mathbf{I}_w \boldsymbol{\omega}_w \right) \right] \tag{3.54}$$

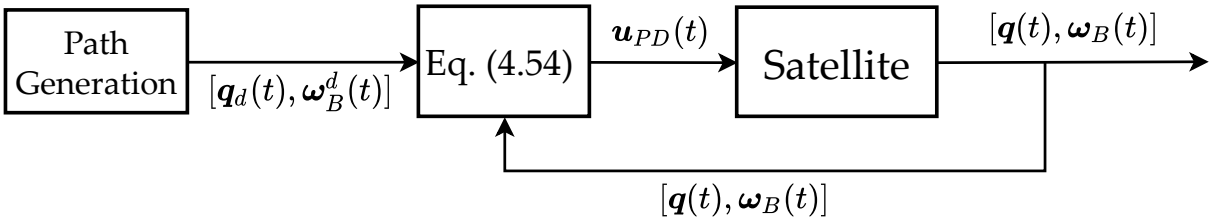


Figure 3.13: Traditional PD tracking control structure

where k_p is the scalar proportional gain, $\mathbf{K_D}$ is the derivative gain matrix, $\delta\boldsymbol{q}_{1:3}$ is the vector component of the error quaternion, and $\delta\boldsymbol{\omega}_B$ is the error in the body rate:

$$\delta\boldsymbol{\omega}_B = \boldsymbol{\omega}_B - \boldsymbol{\omega}_B^d \tag{3.55}$$

where the desired body rate, $\boldsymbol{\omega}_B^d$, and desired body acceleration, $\dot{\boldsymbol{\omega}}_B^d$, can be obtained from the desired quaternion attitude path:

$$\boldsymbol{\omega}_B^d = 2\begin{bmatrix} -\boldsymbol{q}_{1:3}^{d\top} \\ q_0^d\mathbf{I}_{3\times3} + \left[\boldsymbol{q}_{1:3}^d\right]_\times \end{bmatrix}^\top \dot{\boldsymbol{q}}_d \tag{3.56a}$$

$$\dot{\boldsymbol{\omega}}_B^d = 2\begin{bmatrix} -\dot{\boldsymbol{q}}_{1:3}^{d\top} \\ \dot{q}_0^d\mathbf{I}_{3\times3} + \left[\dot{\boldsymbol{q}}_{1:3}^d\right]_\times \end{bmatrix}^\top \dot{\boldsymbol{q}}_d + 2\begin{bmatrix} -\boldsymbol{q}_{1:3}^{d\top} \\ q_0^d\mathbf{I}_{3\times3} + \left[\boldsymbol{q}_{1:3}^d\right]_\times \end{bmatrix}^\top \ddot{\boldsymbol{q}}_d \tag{3.56b}$$

where $\boldsymbol{q}_d = [q_0^d \ \boldsymbol{q}_{1:3}^d]\top$ is the desired attitude quaterninon. Additionally, the body axis torques are mapped to each reaction wheel using the MP pseudoinverse (Eq. (A.19)) of the actuator distribution matrix $\mathbf{A}_w^+$. Next, the closed-loop, rigid-body dynamics are found in terms of body rate errors. First, the derivative of the body rate error is taken:

$$\delta\dot{\boldsymbol{\omega}}_B = \dot{\boldsymbol{\omega}}_B - \dot{\boldsymbol{\omega}}_B^d - \left[\boldsymbol{\omega}_B\right]_\times \dot{\boldsymbol{\omega}}_B^d \tag{3.57}$$

Rearranging and substituting into the dynamic ODE, defined in Appendix E.3 (Eq. (E.13)), along with the proposed control law (Eq. (3.54)), and cancelling out terms, gives the closed-loop error dynamics:

$$\delta\dot{\boldsymbol{\omega}}_B = \mathbf{I}_{tot}^{-1}\left[-k_p\delta\boldsymbol{q}_{1:3} - \mathbf{K_D}\delta\boldsymbol{\omega}_B\right] \tag{3.58}$$

Next, a candidate Lyapunov function similar to that in [81] is chosen, but in terms of quaternion and body rate error to ensure stability in the tracking scenario:

$$V = \frac{1}{4}\delta\boldsymbol{\omega}_B^\top\mathbf{I}_{tot}\delta\boldsymbol{\omega}_B + \frac{1}{2}k_p\delta\boldsymbol{q}_{1:3}^\top\delta\boldsymbol{q}_{1:3} + \frac{1}{2}k_p(1-\delta q_0)^2 \geq 0 \tag{3.59}$$

where $\delta q_0^{B/I}$ is the scalar component of the quaternion error. Next, the time derivative of the Lyapunov function is taken:

$$\dot{V} = \frac{1}{2}\delta\boldsymbol{\omega}_B^\top \mathbf{I}_{tot}\delta\dot{\boldsymbol{\omega}}_B + k_p\delta\boldsymbol{q}_{1:3}^\top\delta\dot{\boldsymbol{q}}_{1:3} - k_p(1-\delta q_0)\delta\dot{q}_0 \tag{3.60}$$

the quaternion kinematics defined in Appendix B.3 can be split in terms of the scalar and vector components:

$$\dot{\boldsymbol{q}}_{1:3} = \frac{1}{2}q_0\boldsymbol{\omega}_B + \frac{1}{2}\left[\boldsymbol{q}_{1:3}\right]_\times \boldsymbol{\omega}_B \tag{3.61a}$$

$$\dot{q}_0 = -\frac{1}{2}\boldsymbol{q}_{1:3}^\top\boldsymbol{\omega}_B \tag{3.61b}$$

The kinematics of the components of the error quaternion can then be found [81]:

$$\delta\dot{\boldsymbol{q}}_{1:3} = \frac{1}{2}\delta q_0\delta\boldsymbol{\omega}_B + \frac{1}{2}\left[\delta\boldsymbol{q}_{1:3}\right]_\times\left(\boldsymbol{\omega}_B^B + \boldsymbol{\omega}_B^d\right) \tag{3.62a}$$

$$\delta\dot{q}_0 = -\frac{1}{2}\delta\boldsymbol{q}_{1:3}^\top\delta\boldsymbol{\omega}_B \tag{3.62b}$$

substituting Eqs. (3.58), (3.62a) and (3.62b) into Eq. (3.60) gives:

$$\begin{aligned}\dot{V} = &\frac{1}{2}\delta\boldsymbol{\omega}_B^\top\left(-k_p\delta\boldsymbol{q}_{1:3} - \mathbf{K_D}\delta\boldsymbol{\omega}_B\right)\\ &+\frac{1}{2}k_p\delta\boldsymbol{q}_{1:3}^\top\left[\delta q_0\delta\boldsymbol{\omega}_B + \left[\delta\boldsymbol{q}_{1:3}\right]_\times\left(\boldsymbol{\omega}_B + \boldsymbol{\omega}_B^d\right)\right] + \frac{1}{2}k_p(1-\delta q_0)\delta\boldsymbol{q}_{1:3}^\top\delta\boldsymbol{\omega}_B\end{aligned} \tag{3.63}$$

collecting and cancelling terms:

$$\dot{V} = -\frac{1}{2}\delta\boldsymbol{\omega}_B^\top\mathbf{K_D}\delta\boldsymbol{\omega}_B < 0 \tag{3.64}$$

Therefore, stability is proven, provided that $\mathbf{K_D}$ is positive definite with the body rate error $\lim_{t\to\infty}\delta\boldsymbol{\omega}_B = 0$. Furthermore aymptotic stability can be proven through consideration of Eq. (3.58) where it can be seen that $\lim_{t\to\infty}\delta\boldsymbol{\omega}_B = 0$ only if $\lim_{t\to\infty}\delta\boldsymbol{q}_{1:3}^{B/I} = 0$ to reach the equilibrium $\delta\dot{\boldsymbol{\omega}}_B = 0$. Additonally $\lim_{t\to\infty}\delta q_0 = 1$ simply through the unit quaternion constraint.

Next, the roots of a linearised form of the closed-loop, rigid dynamics are found. This will allow for the PD control gains, k_p and $\mathbf{K_D}$, to be chosen to give a closed-loop response that does not excessively excite the flexible modes of the satellite, and thus ensure stability. Taking the closed-loop dynamics, Eq. (3.58), and linearising the quater-

nion error kinematics around $\delta \boldsymbol{q}_{1:3} = [1\ 0\ 0\ 0]^\top$:

$$\delta \boldsymbol{q}_{1:3} \approx \frac{1}{2}\delta \boldsymbol{\omega}_B \tag{3.65}$$

and then by considering each axis separately, and ignoring off-diagonal inertia terms, the closed-loop dynamics can be expressed as:

$$\begin{bmatrix} \delta \dot{q}_i \\ \\ \delta \dot{\omega}_i \end{bmatrix} = \begin{bmatrix} 0 & \frac{1}{2} \\ \\ -\frac{k_p}{I_i} & -\frac{K_{D,i}}{I_i} \end{bmatrix} \begin{bmatrix} \delta q_i \\ \\ \delta \omega_i \end{bmatrix} \tag{3.66}$$

Giving the closed-loop roots $\bar{\lambda}_i$:

$$\bar{\lambda}_i = -\frac{1}{2I_i}\left(K_{D,i} \pm \sqrt{K_{D,i}^2 - 2I_i k_p}\right) \tag{3.67}$$

which can be used to find the natural frequency, $\bar{\omega}_n^i$, damping ratio, $\bar{\zeta}_i$, and damped frequency $\bar{\omega}_\zeta^i$:

$$\bar{\omega}_n^i = ||\bar{\lambda}_i|| \tag{3.68a}$$

$$\bar{\zeta}_i = \cos(\angle\, \bar{\lambda}_i) \tag{3.68b}$$

$$\bar{\omega}_\zeta^i = \bar{\omega}_n^i \sqrt{1 - \bar{\zeta}_i^2} \tag{3.68c}$$

where $\angle$ is the argument, or angle, that the root makes with the real axis on the complex plane. Therefore, the PD control gains can be chosen to satisfy a desired closed-loop response. The results of this gain tuning will be seen later in this chapter.

Hybrid PD-Feedback, InvSim-Feedforward Control Law

The hybrid controller combines the linear PD feedback control elements with a feedforward *offline* InvSim control solution, as can be seen in Fig. 3.14. The full control law is given as:

$$\boldsymbol{u}_{hybrid} = \boldsymbol{u}_{InvSim\text{-}FF} + \mathbf{A}_w^+ \left[-k_p \delta \boldsymbol{q}_{1:3} - \mathbf{K_D} \delta \boldsymbol{\omega}_B\right] \tag{3.69}$$

The InvSim feedforward component is calculated before the manoeuvre is executed and should provide a nominal control effort to track the attitude slew manoeuvre, compensating for the nonlinear elements seen in the PD tracking control law (Eq. (3.54)), and for the flexible effects of the satellite appendages. The linear PD elements in Eq. (3.69) can provide some robustness to uncertainty in the model used by InvSim and also deal with external disturbances. Assuming that the InvSim is accurate and

provides the nominal control effort, the hybrid control law in Eq. (3.69) will give the approximated closed-loop error dynamics seen in Eq. (3.66). Therefore Eqs. (3.67) and (3.68a) to (3.68c) can be used to find suitable gains k_p and $\mathbf{K_D}$ for both the traditional PD tracking control law (Eq. (3.54)), and the hybrid control law (Eq. (3.69)).

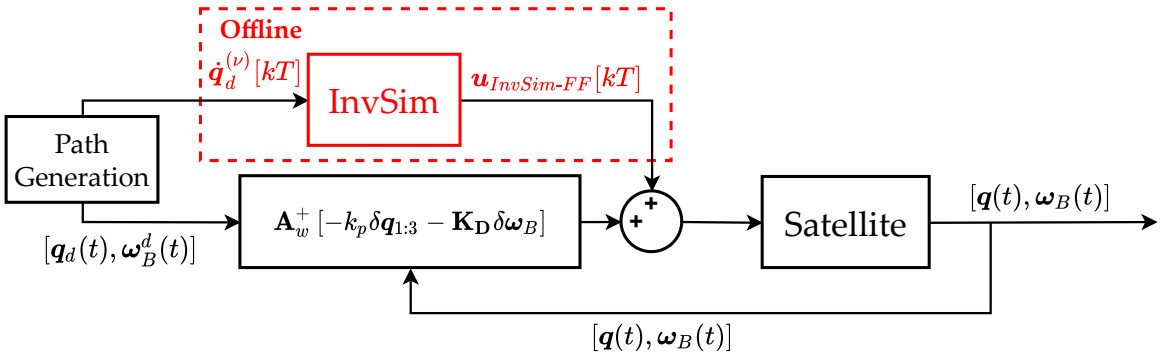


Figure 3.14: Hybrid control: Linear PD feedback with offline InvSim feedforward compensation

3.3.2 | Attitude-Tracking Control Stability

It has been shown that the PD tracking controller seen in Eq. (3.54) is capable of stabilising the system if it were assumed to be rigid. However, the system can be destabilised should the flexible modes of the satellite be excited sufficiently. Excessive excitation of the flexible modes can be avoided through the mitigation of jerk (the derivative of acceleration) during attitude manoeuvres. While several approaches exist, as discussed in Section 1.2.3, here it is found that the polynomial SLERP trajectories used thus far (Appendix B.4) already exhibit low jerk, with the downside of requiring higher maximum acceleration than a bang-bang manoeuvre of equal duration [96]; illustrated in Fig. 3.15. However, it is still possible for each control architecture to excite the flexible modes and destabilise the system if inappropriate PD gains are chosen.

Appropriate tuning of the PD gains can be accomplished by first considering the lowest frequency flexible mode, i.e. the first mode. In the case of the model parameters adopted in this section (Table 3.1), the first mode will have a natural frequency of approximately $\bar{\omega}_n = 2.03$ rad/s, as shown previously in Figs. 3.4 and 3.5a. The PD gains should therefore be tuned to give a closed-loop frequency response that is sufficiently lower than 2.03 rad/s to ensure that any excitation of the flexible modes is not amplified, and does not trigger instability. The consequences of not ensuring sufficient margin to the flexible modes can be seen in Fig. 3.16, which shows instability when using the hybrid controller to track a polynomial SLERP, with PD gains tuned for a damped closed-loop frequency of $\bar{\omega}_\zeta \approx 2$ rad/s and damping ratio $\bar{\zeta} \approx 0.7$. In this example, the system is completely known with the InvSim feedforward control element providing

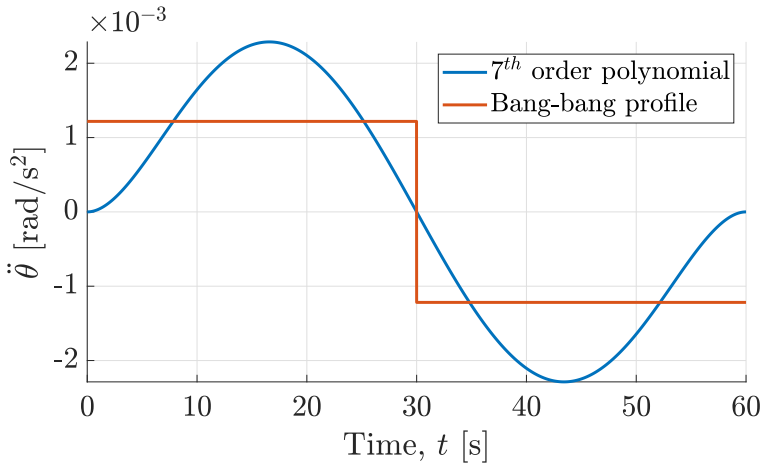


Figure 3.15: Comparison of the magnitude of a 7th order polynomial and bang-bang acceleration profile

good tracking by itself; however, with the addition of the fast PD control elements, the flexible modes have been excited sufficiently to induce instability. This is also the case for the PD tracking controller, as the proportional and derivative terms are the same.

If the PD controller is instead tuned to have a slower response, with damped natural frequency $\bar{\omega}_\zeta \approx 1$ rad/s and damping ratio $\bar{\zeta} \approx 0.7$, then stability can be regained as seen in Fig. 3.17. The PD gains could be tuned to be even slower, providing a larger margin to the flexible modes, however, as the attitude-tracking performance of both the PD tracking and the hybrid controllers need to be compared, tuning the PD gains to be too slow would give an unfair advantage to the hybrid approach. Therefore, for a fair comparison, the PD gains for both controllers are given in Table 3.3, giving the response for the hybrid controller as seen in Fig. 3.17.

Table 3.3: PD control gains, $\bar{\omega}_\zeta \approx 1$ rad/s, $\bar{\zeta} \approx 0.7$

k_p	3067
$\mathbf{K_D}$	diag(700,2500,2800)

The InvSim parameters used for the hybrid controller are shown in Table 3.4. It was highlighted in Section 3.2 that the numerical integration methods used within the InvSim algorithm require a growing number of ODE evaluations as the number of finite elements increases. This has the effect of slowing the speed at which the feed-forward term can be computed, resulting in an undesirable delay between the attitude command and execution of the manoeuvre. Whilst utilisation of implicit trapezoidal integration over explicit RK4 was shown in Section 3.2 to mitigate this loss in speed, it may still not be sufficient to provide a timely control signal, especially if the algorithm is deployed on real-time hardware where computational power is a limiting factor.

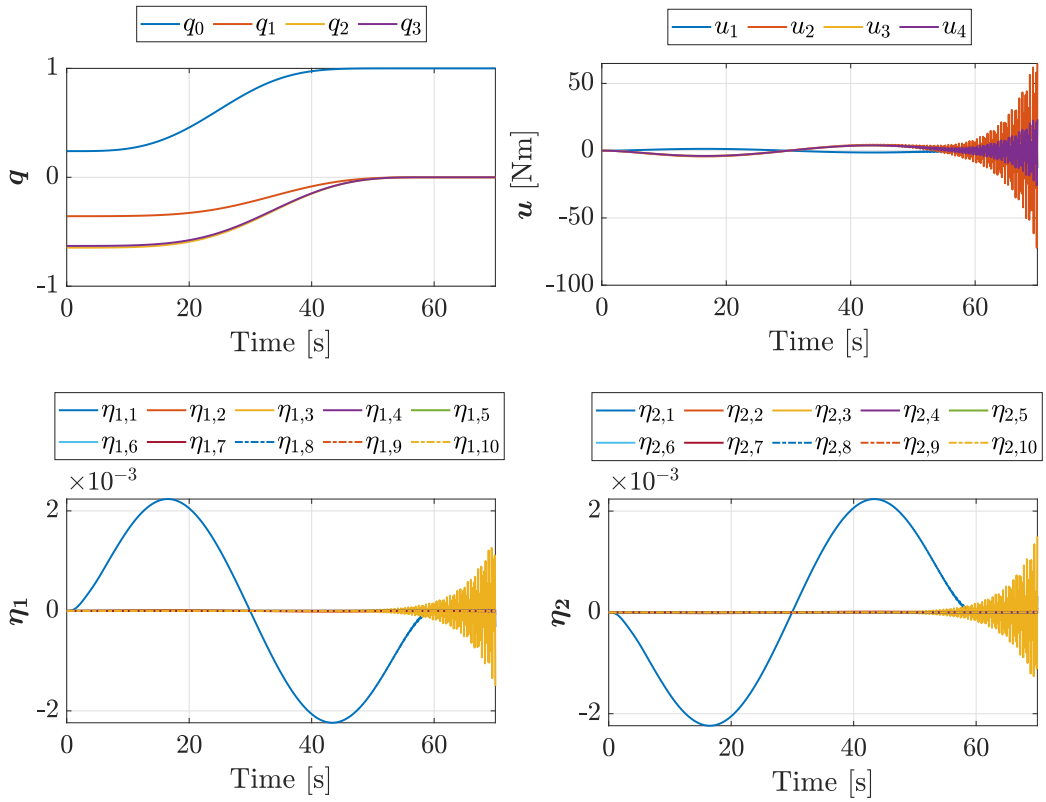


Figure 3.16: Hybrid control unstable response: $\bar{\omega}_\zeta \approx 2$ rad/s, $\bar{\zeta} \approx 0.7$

As can be seen in Fig. 3.5a, the first mode natural frequency does not converge much further after the 3rd or 4th finite element is added. Therefore, the level of model fidelity required for the model-based hybrid controller to provide sufficient attitude-pointing performance, whilst not requiring too much computational effort, should be investigated. It could be the case that a higher-fidelity model does not provide any significant improvement in attitude-tracking performance, and therefore, a simpler, more computationally efficient model may be used to provide the feedforward control signal more promptly.

Table 3.4: Hybrid controller: InvSim hyperparameters

Hyperparameter	**Value**
Discretisation timestep, T	0.1 s
NR tolerance, $\delta\dot{y}_{tol}^{(\nu)}$	1×10^{-16}
Maximum NR iterations, n_{max}	10
Minimum finite difference perturbation, Δu_{min}	1 Nm
Desired output, $\dot{\boldsymbol{y}}_d^{(\nu)}$	$\dot{\boldsymbol{q}}_{d,1:3}^{(\nu)B/I}$
Desired output order, ν	1 (velocity)

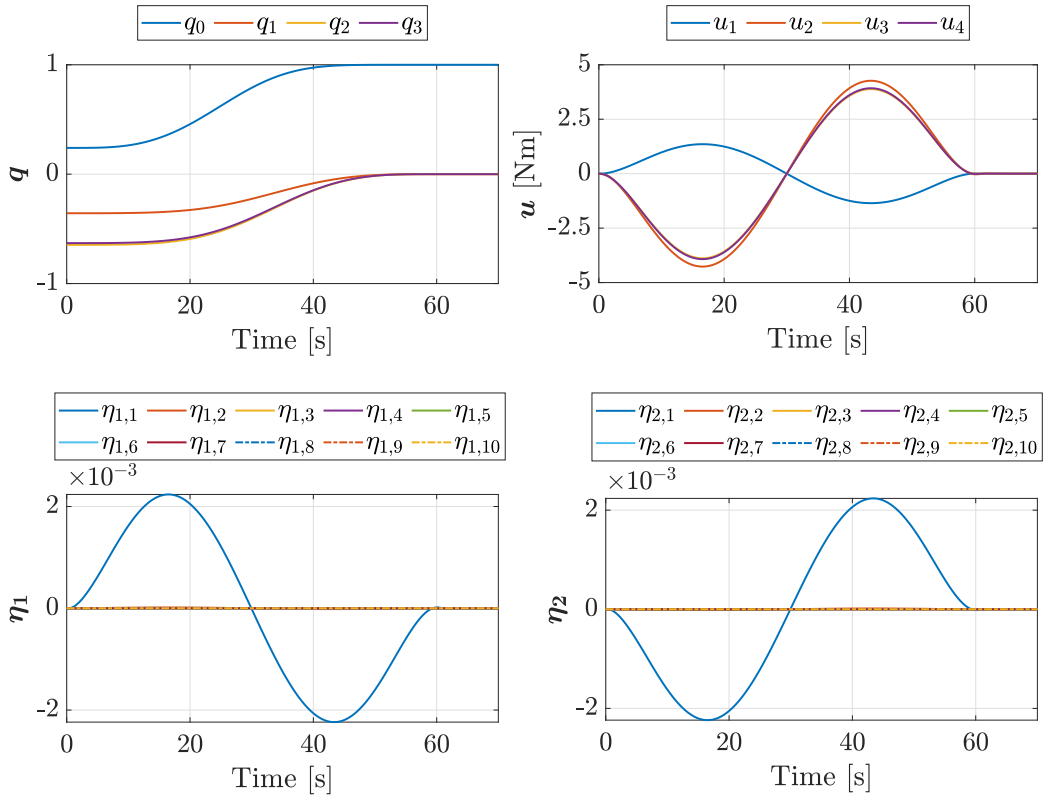


Figure 3.17: Hybrid control stable response: $\bar{\omega}_\zeta \approx 1$ rad/s, $\bar{\zeta} \approx 0.7$

3.3.3 | Effects of Model Fidelity and Uncertainty

The validity and performance of the model-based hybrid controller through comparisons to the traditional PD tracking controller are performed using sets of Monte-Carlo simulations and an in-depth sensitivity analysis. The continuous analytical beam appendages (Section 3.1.3) are used to provide a model which represents the *real* system, the InvSim used in the hybrid controller is configured to utilise different numbers of finite elements ($N_e = 1$ to $N_e = 6$). Additionally, a rigid-body model (Appendix E.3) will be used within the InvSim for the hybrid controller to more clearly see the impact of considering flexible dynamics within the control architecture.

Monte-Carlo Simulations

The first set of Monte-Carlo simulations is constructed by first generating 100 uniformly-random [173] quaternion SLERP polynomial slew manoeuvres. The attitude-tracking performance of each control architecture and configuration is quantified using the pointing error calculated using Eq. (B.14) in Appendix B.2. An additional set of Monte-Carlo simulations is run for the same 100 uniformly random manoeuvres, but in this case with the addition of uniform uncertainty applied to each of the *nominal* model parameters defined in Table 3.1. Each one of the 100 manoeuvres is run 100 times

for each controller, varying the uncertainty in the *real* model according to the ranges in Table 3.5. Therefore, 10,000 simulations are run for the traditional PD controller, and each hybrid control configuration.

Table 3.5: Monte-Carlo parameter uncertainties

Property	Uniform uncertainty
$\mathbf{I}_{hub}$	$\pm$ 10% scale $\pm$ 5deg misalignment
$[L_1\ L_2]$	$\pm$ 10% scale
$[m_1\ m_2]$	$\pm$ 10% scale
$[d_1\ d_2]$	$\pm$ 10% scale
$[EI_1\ EI_2]$	$\pm$ 10% scale
$[\hat{\boldsymbol{x}}_1^B\ \hat{\boldsymbol{x}}_2^B]$, $[\hat{\boldsymbol{z}}_1^B\ \hat{\boldsymbol{z}}_2^B]$	$\pm$ 5 deg misalignment
$\mathbf{A}_w$	$\pm$ 5 deg misalignment

The results of these Monte-Carlo simulations, both with and without parameter uncertainty, can be seen in Fig. 3.18 in the form of boxplots. The centre of each box shows the median mean pointing error for a specific controller. The upper and lower bounds of the boxes show the 75^{th} and 25^{th} percentiles, meaning that the box itself contains 50% of all of the results. The dotted whiskers show the absolute maximum and minimum mean pointing errors seen during the Monte-Carlo simulations for that specific controller configuration.

It can be seen in the case where no parameter uncertainty is present that the hybrid controller performs better than the PD tracking controller for any level of model fidelity. The inclusion of the flexible modes in the InvSim further improves tracking performance, giving an order of magnitude improvement over the traditional PD tracking controller. However, it is interesting to note that there does not seem to be any noticeable pointing accuracy gained by increasing the number of finite elements included in the hybrid controller's model. Instead, it seems that, in the absence of parameter uncertainty, a one-element FEM model is perfectly sufficient and increasing the number of elements will just add unnecessary computational cost. This is most likely because only the first few modes will have a significant impact on the rigid main hub, and therefore, it may be that the one-element model provides sufficient accuracy in the first few modes to give good attitude-tracking performance.

Next, it can be seen that when uniform parameter uncertainties are applied during the Monte-Carlo simulations in the ranges set out in Table 3.5, the performance advantage of the hybrid controller is almost completely lost. Therefore, it can be concluded that accurate knowledge of the system parameters is needed to achieve significant performance advantages when using the hybrid control approach.

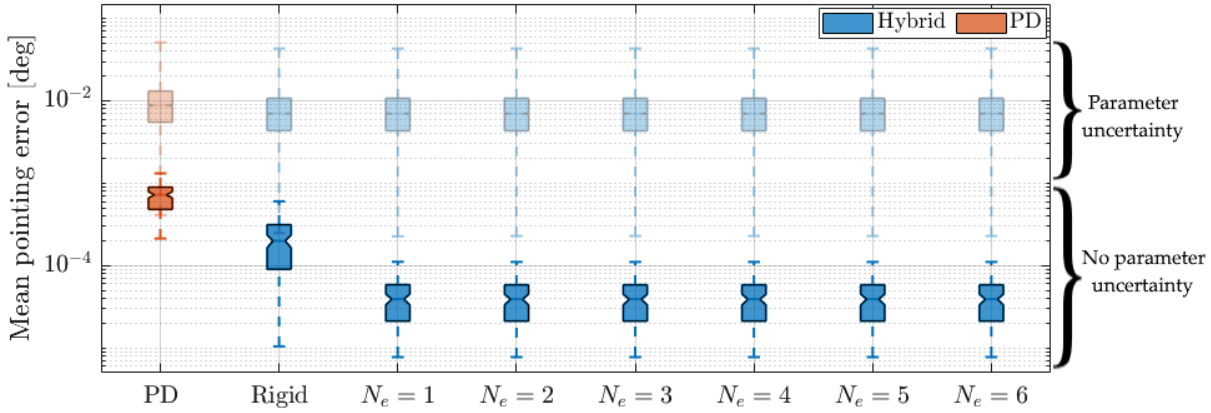


Figure 3.18: Mean pointing error Monte-Carlo results: with parameter uncertainty (top), and without parameter uncertainty (bottom)

Sensitivity Analysis

The effects of model parameter uncertainty on controller performance will be further investigated to determine which uncertainties degrade performance the most and ascertain the limits of the hybrid control approach. To this aim, a sensitivity analysis was performed on a hybrid controller that utilises 2 finite elements, a hybrid controller that utilises a rigid-body model (Appendix E.3), and the PD tracking controller. This will allow the effects of using the InvSim feedforward term and the effect of the model fidelity to be assessed independently.

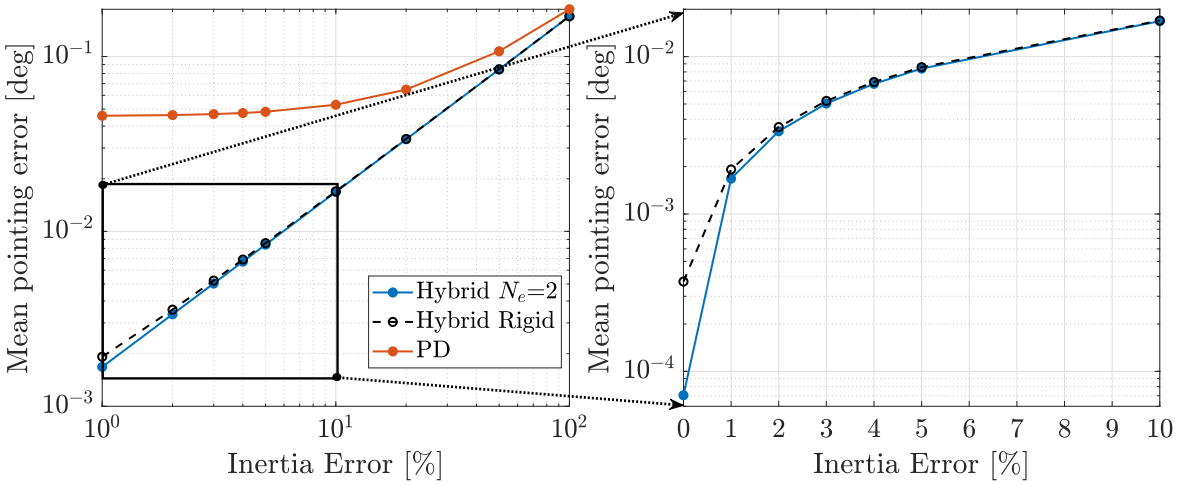


Figure 3.19: Sensitivity to scalar inertia uncertainty (including zoomed area)

First, the sensitivity of each controller to scalar uncertainty in the inertia tensor can be seen in Fig. 3.19, with an expanded close-up view on the right to highlight the difference between the hybrid controllers configured with rigid and FEM models. As the inertia uncertainty increases, both hybrid controllers approach the performance of the

PD tracking controller. Looking more closely, the performance of the FEM hybrid controller approaches that of the one utilising only a rigid-body model very quickly. Most of the performance advantage gained by including the flexible modes in the hybrid controller is lost within the first 1%. It can therefore be deduced that with an uncertain inertia tensor only, hybrid control will perform better than PD alone unless the inertia uncertainty is considerable. Additionally, including a FEM model with the hybrid controller will give improved performance over using only a rigid-body model if the inertia is well known, with an uncertainty of 1%.

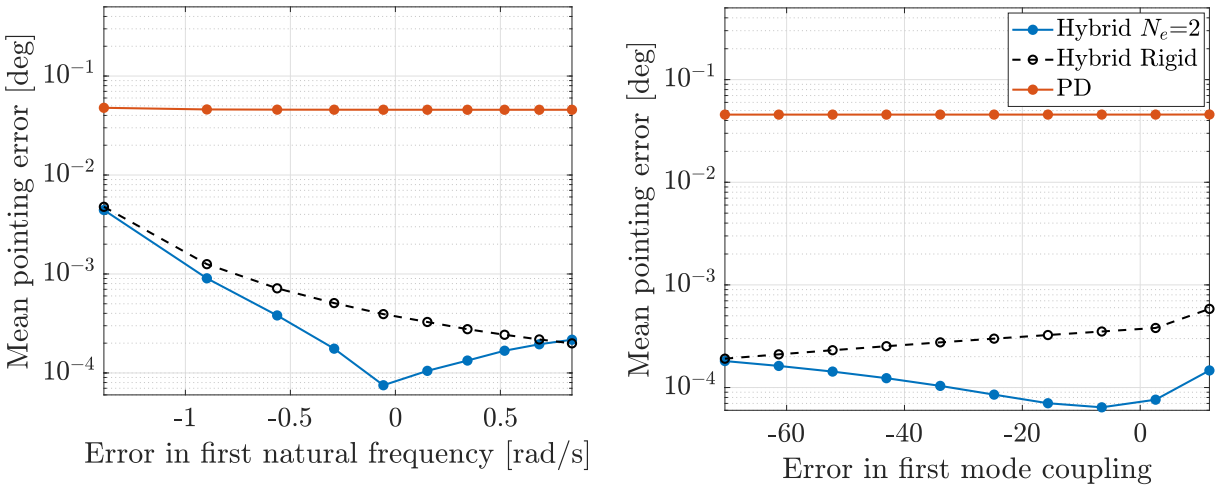


Figure 3.20: Sensitivity to errors in flexible dynamics

Next, assuming that the inertia of the satellite is known, the effects of uncertainty in the flexible modes can be seen in Fig. 3.20. First, it can be seen that as the coupling of the first flexible mode decreases, the performance of the FEM hybrid controller approaches that of the rigid hybrid controller. As the effect of the flexible modes on the dynamics of the rigid hub is diminished, their consideration in the controller design becomes inconsequential. Additionally, as the coupling increases, both hybrid controllers start to degrade in performance as there are additional torques on the main body due to flexing that is not being accounted for. Secondly, it can be seen that, as the uncertainty in the first natural frequency increases, the performance of the FEM hybrid controller approaches that of the rigid hybrid controller. As the flexible modes become uncertain, their inclusion in the hybrid controller becomes unnecessary and performs similarly to a hybrid controller that does not consider the flexible modes at all. It can be seen that in the case of uncertainty in the flexible modes only, both hybrid controllers maintain a significant performance advantage over the PD tracking controller. It can also be seen that, to ensure the best performance of the FEM hybrid controller, the uncertainty in the first mode's natural frequency needs to be maintained at less than 0.5 rad/s. Looking at Fig. 3.5a it can be seen that the one-element FEM model is well within this range,

which explains why not much improvement was gained when increasing the number of finite elements (Fig. 3.18).

An important note is that the ranges of uncertainties shown in Figs. 3.19 and 3.20 are limited by the stability of the controllers. Considering the inertia, Fig. 3.19, it can be seen that only positive inertia uncertainty was considered, i.e. when the inertia is greater than that assumed by the controllers. This is because, as the inertia decreases, the frequency response of the closed-loop dynamics increases (assuming the same PD control gains) according to Eqs. (3.67) and (3.68a) to (3.68c). The closed-loop frequency response will approach the natural frequencies of the flexible modes, eventually triggering instability, as seen in Fig. 3.16. It can therefore be seen that the PD gains may need to be retuned to give a slower response, and a larger margin in the first flexible mode, to ensure stability for a larger range of uncertainties in the inertia.

While it has been possible for the InvSim-based hybrid controller developed in this section to accurately track the prescribed attitude slew manoeuvres, it can be seen in Fig. 3.17 that the flexible modes of the appendage are still excited. Excessive bending of the satellite's appendages could lead to structural failures or adversely impact the instrumentation and sensors mounted on them. It may therefore be desirable to find alternative attitude slew paths that minimise appendage flexing, whilst still reaching the final desired orientation within a fixed time. The next section of this chapter will therefore develop an InvSim-based path planning process for finding such flex-minimising slew manoeuvres, which in turn could be followed using the InvSim-based hybrid attitude control scheme developed here.

3.4 | Attitude Path Planning using Inverse Simulation

As seen previously, the InvSim solution obtained is always contingent upon there being a defined, smooth, and continuous desired output to be tracked. Thus far, this thesis has utilised piecewise polynomials defined in Euler Angle space (Section 2.3), and polynomial rest-to-rest manoeuvres along SLERP paths between two quaternions (Appendix B.4). SLERPs are particularly useful as they give the shortest path between any two orientations of the satellite; however, for satellites with lightweight appendages, these shortest paths may, in some cases, cause excessive excitation of their flexible modes. In Section 3.3, this excitation was reduced through the low-jerk polynomial used to slew along the SLERP path, however, there is still no control over which axis is primarily being rotated about.

Depending on the configuration of the appendages, i.e. the orientations and stiffnesses of each, there may exist some alternative, longer attitude path which minimises flex-

ing of the appendages by avoiding rotations about axes that are most sensitive to the flexible modes. However, assuming a fixed-time manoeuvre, these longer paths will require larger accelerations and jerks and could, therefore, possibly increase appendage flexing. This presents an interesting optimisation problem to find a fixed-time attitude slew path that minimises appendage flexing. InvSim can be utilised to solve the inverse dynamics of any prospective attitude slew path, finding the required input, and additionally, the internal dynamics of the system will also be solved. This evaluation of the attitude path can then be used within an optimisation process to converge upon an attitude path that minimises appendage flexing, or optimises for some other performance criteria.

To minimise the parameter space that must be searched, a specific *shape* of the attitude path can be assumed. This will also ensure that the path is smooth and continuous, an important requirement for InvSim (Section 1.2.4), is properly constrained by the initial and final attitudes, and complies with other constraints such as the unit magnitude of the quaternion. In this section, spherical-quadrangle-interpolation (SQUAD) [174, 175] which utilises a series of SLERPs to generate a spline in quaternion space will be used, as they satisfy all of the properties detailed above.

3.4.1 | Attitude Path Shaping

Linear Cubic Splines

To demonstrate the basic principle behind SQUAD we can first look at linear cubic interpolation splines on a 2D plane. First we define a *linear* interpolation function lerp() between two arbitrary points in 2D space:

$$\text{lerp}(\boldsymbol{p}_0, \boldsymbol{p}_1, h) = \boldsymbol{p}_0(1-h) + \boldsymbol{p}_1 h \tag{3.70}$$

where $\boldsymbol{p}_0$ is the starting position, $\boldsymbol{p}_1$ is the final positon and $h \in [0,1]$ determines the distance along the interpolated path. By adding two extra position vectors, and then successively interpolating between them in an iterative fashion, a cubic spline can be formed as seen in Fig. 3.21. The first layer of interpolations, the points traversing the grey line in Fig. 3.21, can be calculated based on the initial position $\boldsymbol{p}_{00}$, final position $\boldsymbol{p}_{03}$ and the two shaping positions $\boldsymbol{p}_{01}$ and $\boldsymbol{p}_{02}$:

$$\boldsymbol{p}_{10} = \text{lerp}(\boldsymbol{p}_{00}, \boldsymbol{p}_{01}, h_k) \tag{3.71}$$

$$\boldsymbol{p}_{11} = \text{lerp}(\boldsymbol{p}_{01}, \boldsymbol{p}_{02}, h_k) \tag{3.72}$$

$$\boldsymbol{p}_{12} = \text{lerp}(\boldsymbol{p}_{02}, \boldsymbol{p}_{03}, h_k) \tag{3.73}$$

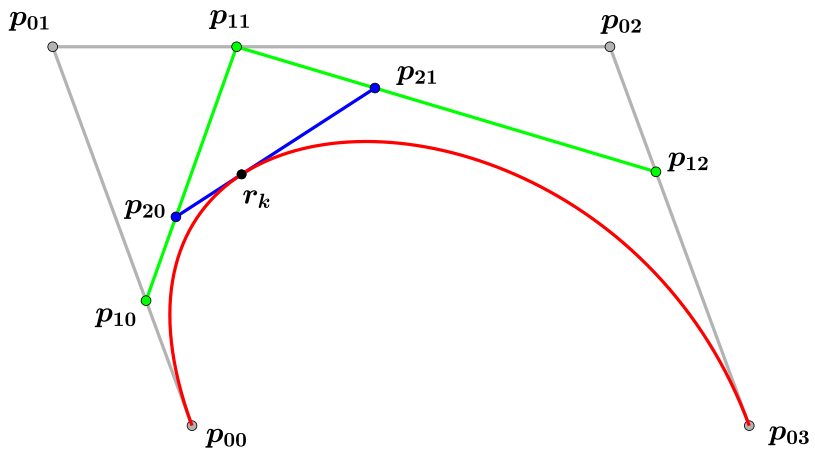


Figure 3.21: Linear interpolation on the x-y 2D plane

where h_k is the distance along the interpolation path at step k. These interpolated positions can then be used as the inputs to the next layer of interpolations:

$$\boldsymbol{p}_{20} = \text{lerp}(\boldsymbol{p}_{10}, \boldsymbol{p}_{11}, h_k) \tag{3.74}$$

$$\boldsymbol{p}_{21} = \text{lerp}(\boldsymbol{p}_{11}, \boldsymbol{p}_{12}, h_k) \tag{3.75}$$

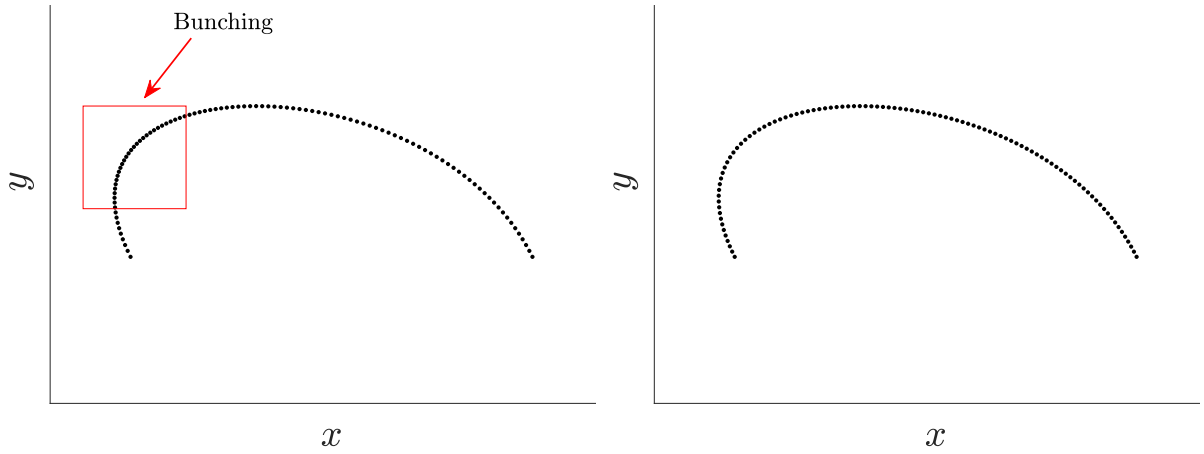
and finally the final position of the spline r_k:

$$\boldsymbol{r}_k = \text{lerp}(\boldsymbol{p}_{20}, \boldsymbol{p}_{21}, h_k) \tag{3.76}$$

The use of the quadrangle/cubic interpolated splines produces an ideal smooth, continuous path for feeding into the InvSim algorithm. However, there are some issues surrounding their use. Firstly, as the distance along the interpolation path h_k is increased linearly it is found that successive points begin to *bunch up*, as illustrated in Fig. 3.22a. When the radius of curvature is small, the points will be closer together, with the points moving further apart as the radius of curvature increases. As the distances between points on the path are non-uniform, its velocity profile will be warped.

The solution is to re-parameterise the spline in terms of the actual distance travelled. This can be done simply through use of a look-up table that provides a mapping of the parameter h_k to the real fractional distance travelled along the path. The results of this re-parameterisation can be seen in Fig. 3.22b where the points no longer bunch together, and in Fig. 3.23 where the distance between successive points is constant. The rest-to-rest manoeuvre along this path can therefore be generated through the use of

a polynomial to traverse along it, similar to the previously used SLERP manoeuvres (Appendix B.4).



(a) Original **(b)** Re-Parameterised

Figure 3.22: Discrete points of cubic linear 2D interpolation

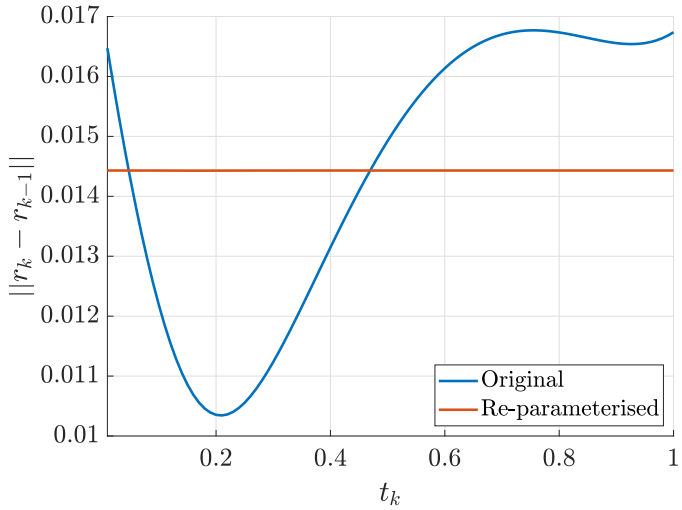


Figure 3.23: Distance between successive interpolated points (original and re-parameterised)

Spherical Cubic Splines (SQUAD)

Using the same principles as the linear cubic interpolated splines, we can also produce quaternion cubic interpolated splines (SQUAD) by simply replacing lerp() with slerp() (Appendix B.4). We re-define SLERP here in terms of an arbitrary stepping parameter h_k and arbitrary initial quaternion $\boldsymbol{q}_0$ and $\boldsymbol{q}_1$:

$$\text{slerp}(\boldsymbol{q_0}, \boldsymbol{q_1}, h_k) = \frac{\sin((1-h_k)\delta_0)}{\sin(\delta_0)}\boldsymbol{q}_0 + \frac{\sin(h_k\delta_0)}{\sin(\delta_0)}\boldsymbol{q}_1 \tag{3.77}$$

where δ_0 is given in Appendix B.4 (Eq. (B.21)). The sequence of interpolations is then performed in the same way as the linear case seen before, with $\boldsymbol{q}_i^{B/I}$ and $\boldsymbol{q}_f^{B/I}$ being the initial and final quaternions respectively, with $\boldsymbol{q}_a^{B/I}$ and $\boldsymbol{q}_b^{B/I}$ being the two shaping quaternions:

$$\boldsymbol{q}_{10}^{B/I} = \text{slerp}(\boldsymbol{q}_i^{B/I}, \boldsymbol{q}_a^{B/I}, h_k) \tag{3.78}$$
$$\boldsymbol{q}_{11}^{B/I} = \text{slerp}(\boldsymbol{q}_a^{B/I}, \boldsymbol{q}_b^{B/I}, h_k) \tag{3.79}$$
$$\boldsymbol{q}_{12}^{B/I} = \text{slerp}(\boldsymbol{q}_b^{B/I}, \boldsymbol{q}_f^{B/I}, h_k) \tag{3.80}$$

These interpolated positions can then be used as the inputs to the next layer of interpolations:

$$\boldsymbol{q}_{20}^{B/I} = \text{slerp}(\boldsymbol{q}_{10}^{B/I}, \boldsymbol{q}_{11}^{B/I}, h_k) \tag{3.81}$$
$$\boldsymbol{q}_{21}^{B/I} = \text{slerp}(\boldsymbol{q}_{11}^{B/I}, \boldsymbol{q}_{12}^{B/I}, h_k) \tag{3.82}$$

and finally, the final position of the quaternion spline q_k:

$$\boldsymbol{q}_k^{B/I} = \text{slerp}(\boldsymbol{q}_{20}^{B/I}, \boldsymbol{q}_{21}^{B/I}, h_k) \tag{3.83}$$

As before the spline needs to be re-parameterised in terms of the real fractional distance travelled through the use of a lookup table to allow for a polynomial profile, such as Eq. (B.22) in Appendix B.4, to give a smooth rest-to-rest manoeuvre along the quaternion spline path. As mentioned previously, it is common for InvSim to target derivatives of the output states to improve numerical stability. Therefore, the quaternion rate $\dot{\boldsymbol{q}}$ can be found by performing numerical differencing of Eq. (3.83). An example of some SQUAD quaternion spline paths can be seen in Fig. 3.24. It can be seen that the use of the additional shaping quaternions, $\boldsymbol{q}_{01}$ and $\boldsymbol{q}_{02}$, allows for more complex quaternion paths when compared to SLERP to be generated, which will give some additional degrees-of-freedom for the path planning optimisation detailed next.

3.4.2 | Attitude Path Optimisation

An attitude slew path can now be defined using SQUAD by an initial quaternion $\boldsymbol{q}_i^{B/I}$, final quaternion $\boldsymbol{q}_f^{B/I}$, and two shaping quaternions $\boldsymbol{q}_a^{B/I}$ and $\boldsymbol{q}_b^{B/I}$. Considering the case of a single axis pointing, the final rotation about the pointing axis $\theta_f^{\odot}$ can be considered unconstrained, as it will not affect that axis' orientation. For the demonstration in this section of the thesis, the pointing axis will be assumed to align with the body $\hat{\boldsymbol{x}}^B$ axis, such that the final quaternion can be defined through a Z-Y-X Euler rotation set

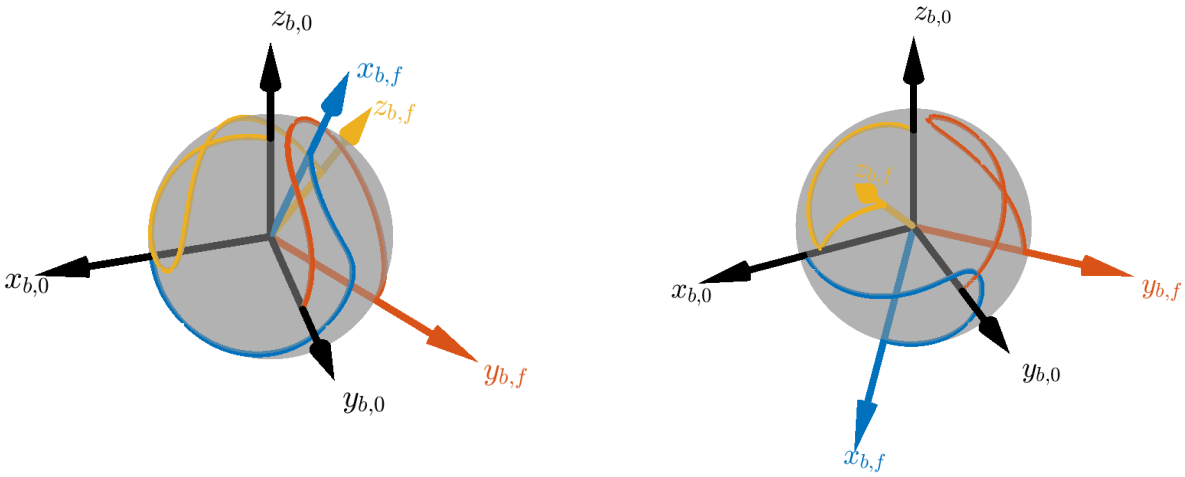


Figure 3.24: Projections of the body axes on a unit sphere for two example rest-to-rest SQUAD slew manoeuvres

(Appendix B):

$$\boldsymbol{\theta}_f = \begin{bmatrix} \theta_f^z & \theta_f^y & \theta_f^{\odot} \end{bmatrix}^\top \tag{3.84}$$

which can easily be converted into the corresponding quaternion $\boldsymbol{q}_f^{B/I}$ using quaternion multiplication as defined in Appendix B.2 (Eq. (B.10)). To avoid having to constrain the quaternion normal to unity during the optimisation process, the shaping quaternions $\boldsymbol{q}_a^{B/I}$ and $\boldsymbol{q}_b^{B/I}$ can also be defined in terms of the Z-Y-X Euler rotation sets $\boldsymbol{\theta}_a$ and $\boldsymbol{\theta}_b$ respectively:

$$\boldsymbol{\theta}_a = \begin{bmatrix} \theta_a^z & \theta_a^y & \theta_a^x \end{bmatrix}^\top \tag{3.85a}$$

$$\boldsymbol{\theta}_b = \begin{bmatrix} \theta_b^z & \theta_b^y & \theta_b^x \end{bmatrix}^\top \tag{3.85b}$$

Given that the slew manoeuvre has fixed time, the only free parameters available to shape the SQUAD path are $\theta_f^{\odot}$, $\boldsymbol{\theta}_a$, and $\boldsymbol{\theta}_b$, giving a small search space to be optimised over. A proposed path planning process utilising InvSim is illustrated in Fig. 3.25.

Given a prescribed attitude path to follow $\overset{(\nu)}{\boldsymbol{y}}(t) = \overset{(\nu)}{\boldsymbol{q}}^{B/I}(t)$ InvSim can be used to solve the internal dynamics of the satellite $\boldsymbol{x}(t)$. The performance of the satellite during a particular attitude manoeuvre can be quantified in terms of the solved full system dynamics $\boldsymbol{x}(t)$ using a cost function $s(\boldsymbol{x}(t))$. It can be seen in Fig. 3.25 that through the use of the SQUAD paths and InvSim, the cost function $s(\boldsymbol{x}(t))$ has been reposed as a function of the SQUAD's free parameters $s(\theta_f^{\odot}, \boldsymbol{\theta}_a, \boldsymbol{\theta}_b)$, which can be optimised to minimise

the cost and achieve best performance:

$$\min_{\boldsymbol{x}(t)} s(\boldsymbol{x}(t)) = \min_{\theta_f^{\odot}, \boldsymbol{\theta}_a, \boldsymbol{\theta}_b} s(\theta_f^{\odot}, \boldsymbol{\theta}_a, \boldsymbol{\theta}_b) \tag{3.86}$$

As seen in Fig. 3.25 the problem is initialised with the initial attitude quaternion $\boldsymbol{q}_i^{B/I}$, total time of the manoeuvre t_f, and the final attitude quaternion $\boldsymbol{q}_f^{B/I}$ with an arbitrary rotation about the pointing axis $\theta_f^{\odot}$. An initial SQUAD path is generated and the internal dynamics are solved through InvSim. The cost function can then be evaluated and utilised in a quasi-Newton (QN) optimisation method (Appendix D.1) which will provide updates to the free parameters of the SQUAD path $\theta_f^{\odot}$, $\boldsymbol{\theta}_a$, and $\boldsymbol{\theta}_b$. These newly updated parameters produce a new SQUAD path to be evaluated by the InvSim and the process continues iteratively for each step k until the first-order optimality (gradient) of the evaluated cost function meets some threshold.

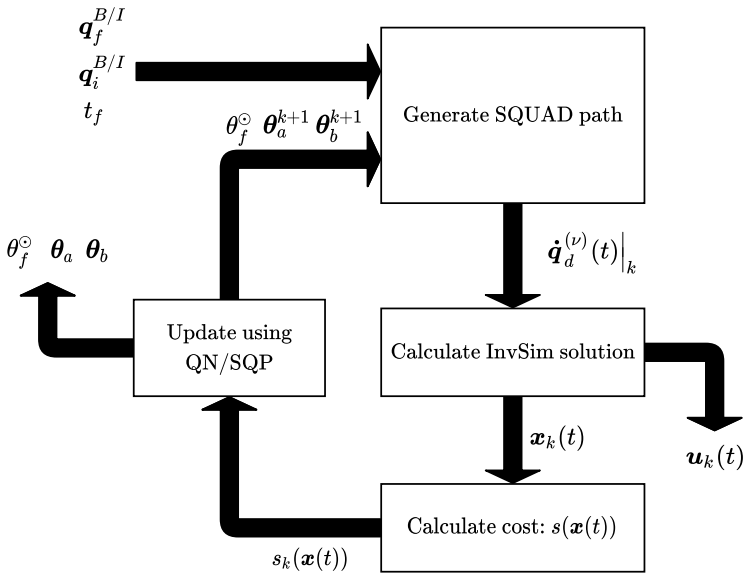


Figure 3.25: InvSim based quaternion path planning algorithm

3.4.3 | Minimisation of Appendage Flexing

While the path planning process illustrated in Fig. 3.25 optimises for any general performance criteria defined by $s(\boldsymbol{x}(t))$, this section will specifically look at finding attitude paths that minimise flexing in the satellite appendages. Minimisation of appendage flexing can be achieved through the cost function:

$$s(\boldsymbol{x}(t)) = \sum_{j}^{N_{ap}} \frac{1}{2} \int_0^{t_f} \boldsymbol{\eta}_j^{\top}(t) \boldsymbol{\eta}_j(t)\, dt \tag{3.87}$$

where $\boldsymbol{\eta}_j$ are the modal coordinates/displacements of the flexible modes of the j^{th} appendage as defined in Section 3.1. The model parameters used are as shown previously in Table 3.1 with the only modification being the use of 3 reaction wheels instead of 4 (Eq. (3.53)). Note that the flexible modes involve bending about the body $\hat{\boldsymbol{y}}^B$ axis as this will become prevalent later. The constraints imposed on the attitude path are given in Table 3.6, with the angle about the pointing axis $\theta_f^{\odot}$ initialised to 0 deg at the start of the optimisation process. To avoid issues of local minimia and ensure that a good solution is found, a multi-start approach is adopted. This is achieved by uniformly randomising the shaping Euler angles, $\boldsymbol{\theta}_a$ and $\boldsymbol{\theta}_b$, at the start of the optimisation process. This will give several locally minimal solutions from which the best-performing solutions can be selected.

Table 3.6: Attitude path planning constraints

Constraints		Value
Initial attitude quaternion	$\boldsymbol{q}_i^{B/I}$	$[1\ 0\ 0\ 0]^\top$
Final attitude Euler angles	$\boldsymbol{\theta}_f$	$[0\ 90\ \theta_f^{\odot}]^\top$ deg
Total manoeuvre time	t_f	60 s

Unconstrained Body Rate

Six multi-start optimisation runs are executed with varying initial shaping Euler angles $\boldsymbol{\theta}_a$ and $\boldsymbol{\theta}_b$; the convergence of the cost function $s(\boldsymbol{x}(t))$ for each run are shown in Fig. 3.26. It can be seen that some runs converge more quickly but to solutions that have a greater cost, or greater flexing of the appendages during the manoeuvre. Runs 3, 4, and 6 converge to costs several orders of magnitude lower than the rest. The solutions from runs 3 and 4 are very similar, therefore particular attention is paid to runs 3 and 6.

Table 3.7: Selected path planning results: Unconstrained body rate

Run #	Final angle about pointing axis, $\theta_f^{\odot}$ [deg]	Shaping Euler Angles [deg] $\boldsymbol{\theta}_a$	$\boldsymbol{\theta}_b$
3	180	[-61.2 18.6 61.2]$^\top$	[92 94.0 -92.0]$^\top$
6	180	[-15.2 90.0 164.8]$^\top$	[-89.6 89.2 89.6]$^\top$

The parameters converged upon by runs 3 and 6 are given in Table 3.7, with the resultant shaped attitude paths shown in Figs. 3.27b and 3.27c respectively; Fig. 3.27a giving the alternative, minimum-distance SLERP path. It can be seen that both solutions converged to the same final rotation about the pointing axis $\theta^{\odot}$, with the shaping Euler angles $\boldsymbol{\theta}_a$ and $\boldsymbol{\theta}_b$ resulting in paths that are approximate mirror images of each other.

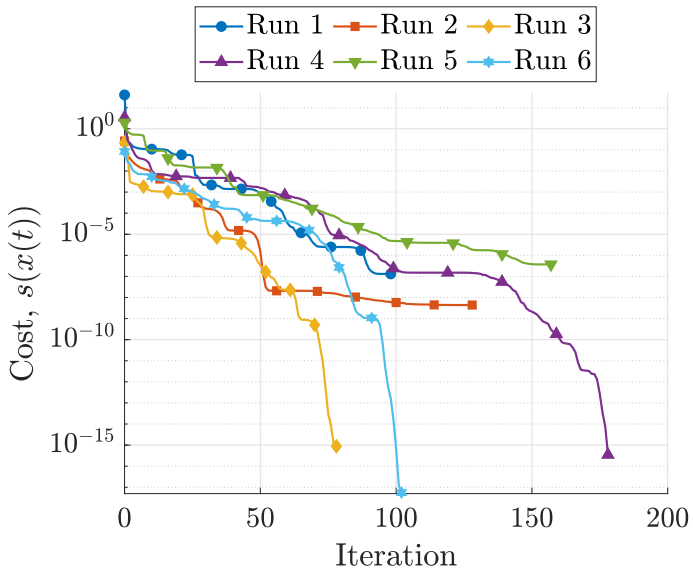
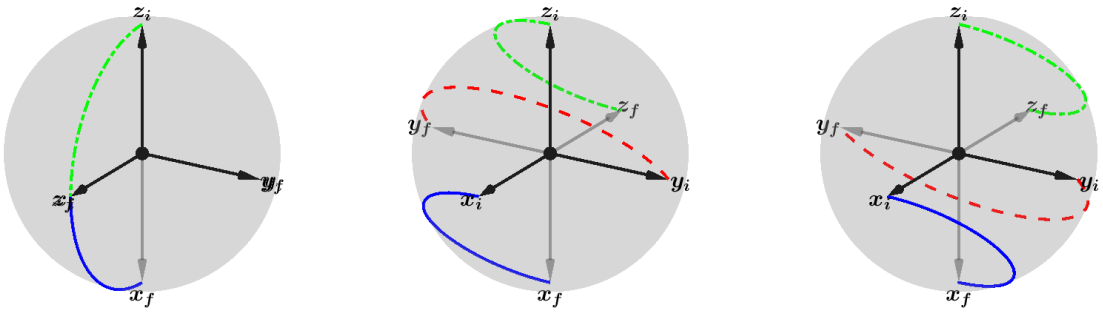


Figure 3.26: Convergence of the cost function $s(\boldsymbol{x}(t))$ (unconstrained body rate)

The displacement of the first modal coordinate for each appendage is shown in Fig. 3.28 for the solutions from run 3, run 6, and the SLERP path. It can be seen clearly that the flexing of the appendages has been reduced dramatically when adopting one of the two shaped SQUAD paths; seen in Fig. 3.29 to correlate to body rates about the body $\hat{\boldsymbol{y}}^B$ axis ω_y being avoided. This relates directly to the configuration of the appendages, as shown previously in Figs. 3.1 to 3.3, which bend about the body $\hat{\boldsymbol{y}}^B$ axis, and will therefore be excited by *jerk* (derivative of acceleration) about that axis.

The path planning algorithm, as illustrated in Fig. 3.25, is therefore able to converge to shaped attitude paths which minimise deformation in a satellite's flexible appendages for a fixed-time slew manoeuvre. The use of InvSim allowed for these shaped attitude paths to take into consideration the constraints imposed by the dynamics of the system, and to allow for each path to be evaluated in terms of the satellite's internal dynamics. As in many cases, the use of a bespoke attitude control algorithm could replace the role of the InvSim. However, in this case, it would pose two significant drawbacks. Firstly, the attitude controller would need to be designed for the specific satellite being considered, with any changes made to the design, such as appendage configuration, requiring retuning or redesign of the controller. Secondly, if a specific path is evaluated to perform badly, it could never be certain whether this is due to a badly designed controller, or if it is simply due to the inherent dynamics of the system. InvSim conversely provides a general solution, requiring no modifications when changes are made to the satellite design, and also removes the added complexity of bad controller design affecting the results.



(a) SLERP path

(b) Optimised SQUAD run 3 path

(c) Optimised SQUAD run 6 path

Figure 3.27: Paths of each body axis projected onto the unit sphere during slew manoeuvres (unconstrained body rate)

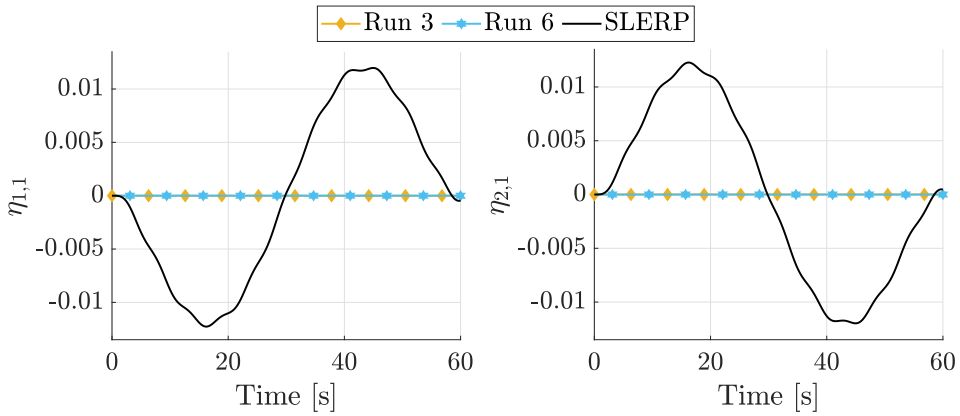


Figure 3.28: First model coordinate $\eta_{i,1}$ of each appendage (unconstrained body rate)

Currently, the use of the InvSim allows for the system's output dynamics to be constrained to the shaped attitude path, and for the internal dynamics to be evaluated, taken into account, and optimised for during the path planning process. It may, however, be that some of the attitude paths are infeasible due to limitations in the satellite's actuators, or limitations in the strength of its structure etc. Additional constraints can therefore be applied to the internal dynamics of the system by replacing the QN process in Fig. 3.25 with sequential-quadratic-programming (SQP), as detailed in Appendix D.2.

Constrained Body Rate

To illustrate the effect of applying additional constraints to the path planning process, restrictions will be placed on the magnitude of the body rate $||\omega_B||$. Including the body

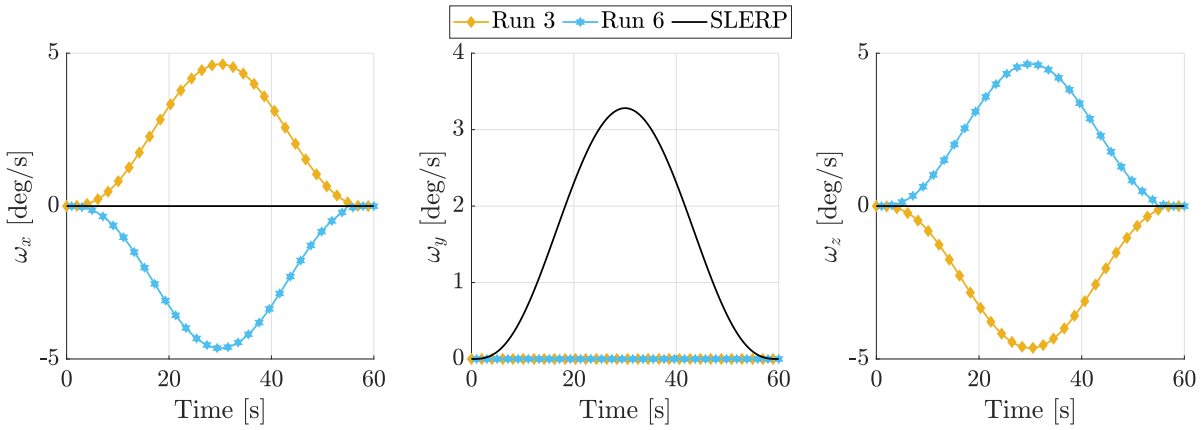


Figure 3.29: Satellite body rate $\boldsymbol{\omega}_B$ during each slew manoeuvre (unconstrained body rate)

rate constraint, the optimisation problem in Eq. (3.86) is modified to:

$$\min_{\theta_f^{\odot}, \boldsymbol{\theta}_a, \boldsymbol{\theta}_b} \; s(\boldsymbol{x}(t)) \tag{3.88a}$$

$$\text{subject to } ||\boldsymbol{\omega}_B|| < \omega_{max} \tag{3.88b}$$

where the cost function $s(\boldsymbol{x}(t))$ is again given as Eq. (3.87) to minimise appendage flexing. Using SQP within the path planning algorithm (Fig. 3.25), optimally-shaped attitude paths with maximum body rates of 6, 5, and 4 deg/s are found and presented in Table 3.8.

Table 3.8: Selected path planning results: Constrained body rate

ω_{max} [deg/s]	Final angle about pointing axis, $\theta_f^{\odot}$ [deg]	Shaping Euler Angles $\boldsymbol{\theta}_a$ [deg]	$\boldsymbol{\theta}_b$ [deg]
6	-143.2	[-167.5 177.0 133.3]$^\top$	[94.6 90.3 -71.8]$^\top$
5	-97.0	[-4.2 13.3 -53.0]$^\top$	[21.2 36.0 -73.5]$^\top$
4	-59.1	[169.6 152.3 147.2]$^\top$	[16.7 50.0 -45.2]$^\top$

It can be seen clearly in Fig. 3.30 that as the maximum body rate is decreased, the final cost converged upon increases. This corresponds to higher flexing in the appendages, as seen in Fig. 3.31, reducing the advantage of the shaped SQUAD path when compared to SLERP.

Considering Fig. 3.32, it can be seen that as the magnitude of the body rate $||\boldsymbol{\omega}_B||$ is reduced, the component about the body $\hat{\boldsymbol{y}}^B$ axis increases. With the appendages bending about the $\hat{\boldsymbol{y}}^B$ axis, this leads to an increase in appendage flexing as seen in Fig. 3.31. Looking at all three components of $\boldsymbol{\omega}_B$, the reduction in ω_{max} can be thought of as causing the shaped attitude path to converge towards the equivalent SLERP. The SLERP will always be the shortest path, with the lowest possible maximum body during the

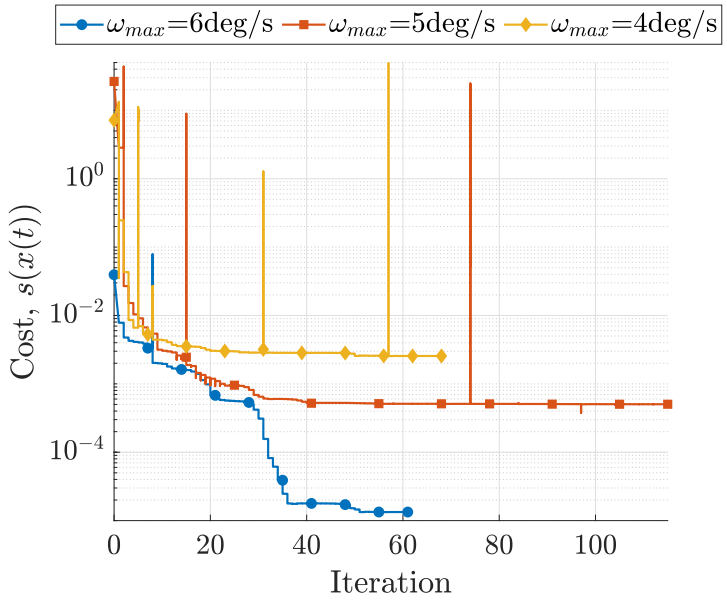


Figure 3.30: Convergence of the cost function $s(\boldsymbol{x}(t))$ (constrained body rate)

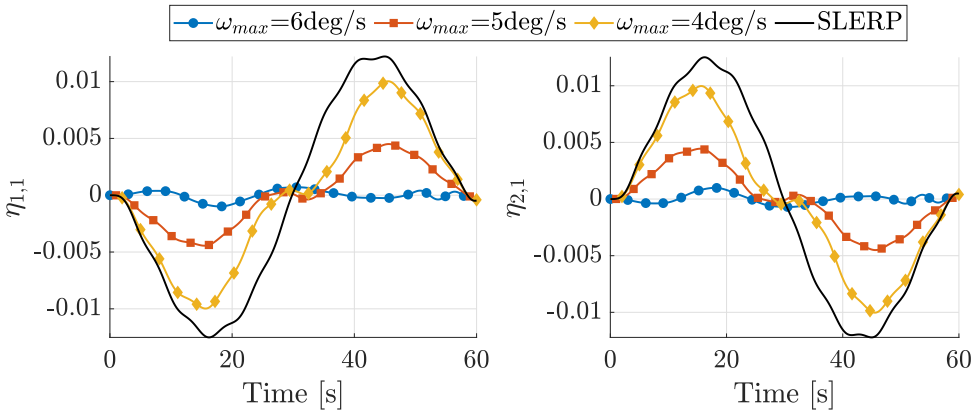


Figure 3.31: First model coordinate $\eta_{i,1}$ of each appendage (constrained body rate)

slew manoeuvre. Shaping of the paths will increase their overall length, and with the fixed-time constraint, cause an increase in the maximum body rate. Reducing the maximum body rate, therefore, forces the shaped path to become shorter and shorter, eventually closely resembling the SLERP.

The projections of these body rate-constrained shaped attitude paths can be seen in Fig. 3.33. It is clear that the reduction of the maximum body rate has limited the length of the attitude paths, and forced more rotation about the body $\hat{\boldsymbol{y}}^B$ axis, inducing greater flexing in the appendages.

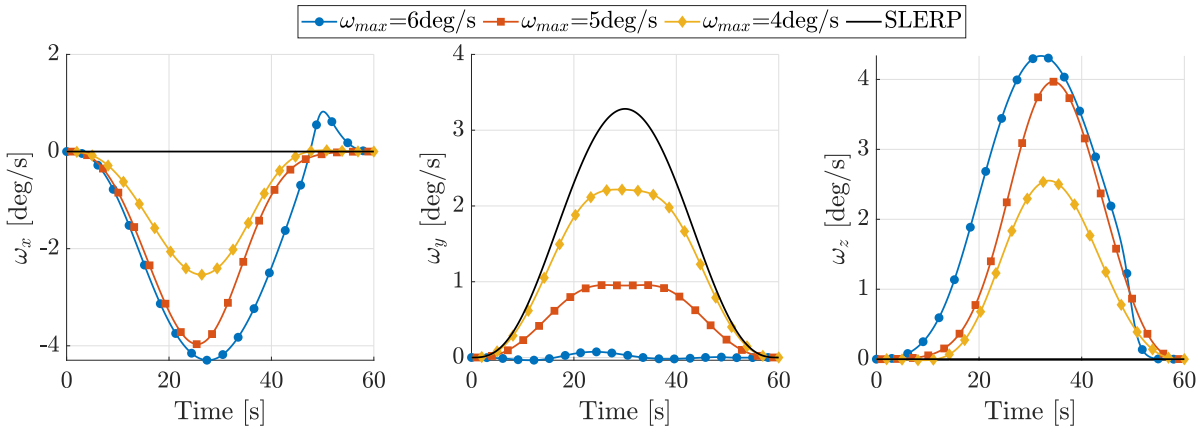
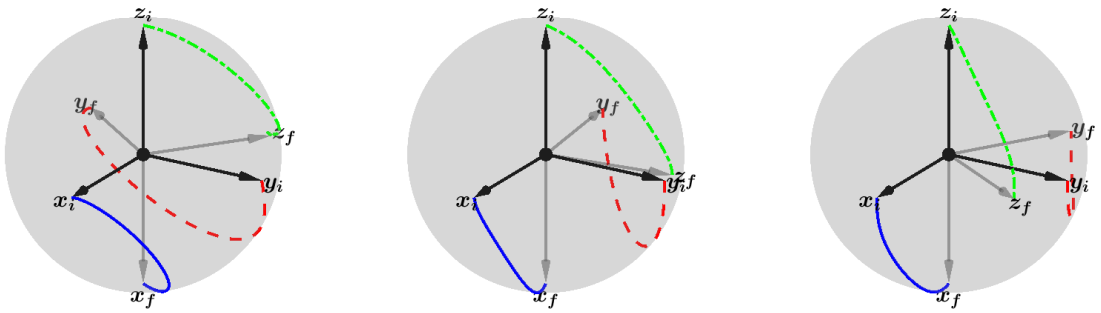


Figure 3.32: Satellite body rate ω_B during each slew manoeuvre (constrained body rate)



(a) Optimised SQUAD $\omega_{max} = 6$ deg/s

(b) Optimised SQUAD $\omega_{max} = 5$ deg/s

(c) Optimised SQUAD $\omega_{max} = 4$ deg/s

Figure 3.33: Paths of each body axis projected onto the unit sphere during slew manoeuvres (constrained body rate)

3.5 | Utilisation of Existing Simulation Frameworks

The hybrid controller developed in Section 3.3, and the InvSim-based attitude path planning in Section 3.4 rely on InvSim to produce the inverse dynamic solution using the model developed in Section 3.1. In the work presented thus far, the developed bespoke model and the InvSim algorithm have been implemented within *MATLAB*. While this approach has provided good results, deriving mathematical models from scratch can be labour-intensive. It may therefore be more attractive to apply InvSim to mathematical models already developed in existing simulation frameworks. Much emphasis has been made on the generality of the integral-based InvSim methodology used thus far in this thesis, which treats the mathematical model and its forward simulation as a *black-box*, only concerned with the effect that each input to the system has on the output. The integral-based InvSim algorithm should therefore be conducive to being

applied to any existing forward simulation framework, avoiding the need to re-derive and validate complex, bespoke models for use with InvSim.

One such simulation framework could be *MATLAB*'s *Simulink* and *Simscape* packages, which provide a graphical block diagram modelling language with a wide range of existing libraries. Another example specific to satellite attitude and orbital dynamics is the *Basilisk Astrodynamics Simulation Framework* developed at the University of Colorado, Boulder [176]. This section of the thesis will therefore investigate whether InvSim can be applied to the *Basilisk* simulation framework, and ascertain any complications or challenges associated with its implementation.

3.5.1 | Basilisk Astrodynamic Simulation Framework

The *Basilisk Astrodynamics Simulation Framework* is primarily written in *C++* to allow for fast and efficient execution, with users setting up and interacting with simulation scenarios using *Python* for ease of use [176]. Each *Basilisk* simulation comprises a set of *modules*, chosen by the user to add the required functionality for their needs. For example, modules can be added to include reaction wheel dynamics, SRP disturbances, an inertial measurement unit, and attitude control systems. The modules interact with each other through a standardised *messaging* system, whereby modules *publish* and *subscribe* to messages containing relevant data through a common message broker. This decouples the flow of input and output of data between modules, removing any explicit inter-module dependency and facilitating the modular nature of the overall simulation framework. This modularity should greatly complement the generality of the InvSim method.

To include the dynamics of the flexible appendages in *Basilisk*, the `nHingedRigidBodyStateEffector` module can be used [63]. This models the flexible modes by constructing each appendage as a series of lumped mass elements attached sequentially by torsional spring dampers. Then, including modules for reaction wheel dynamics and state measurements (`simpleNav`), a complete *Basilisk* simulation can be constructed as illustrated in Fig. 3.34. Note that the model is assumed to be *open-loop* with no attitude control included, instead, the *Basilisk* simulation simply supplies the mapping of an input time history $\boldsymbol{u}(t)$ to the full state time history $\boldsymbol{x}(t)$ through RK4 numerical integration. Additional processing is then required to map the full set of states to the output state/derivative of interest $\dot{\boldsymbol{y}}^{(\nu)} = \dot{\boldsymbol{g}}^{(\nu)}(\boldsymbol{x})$, which will occur after the *Basilisk* simulation.

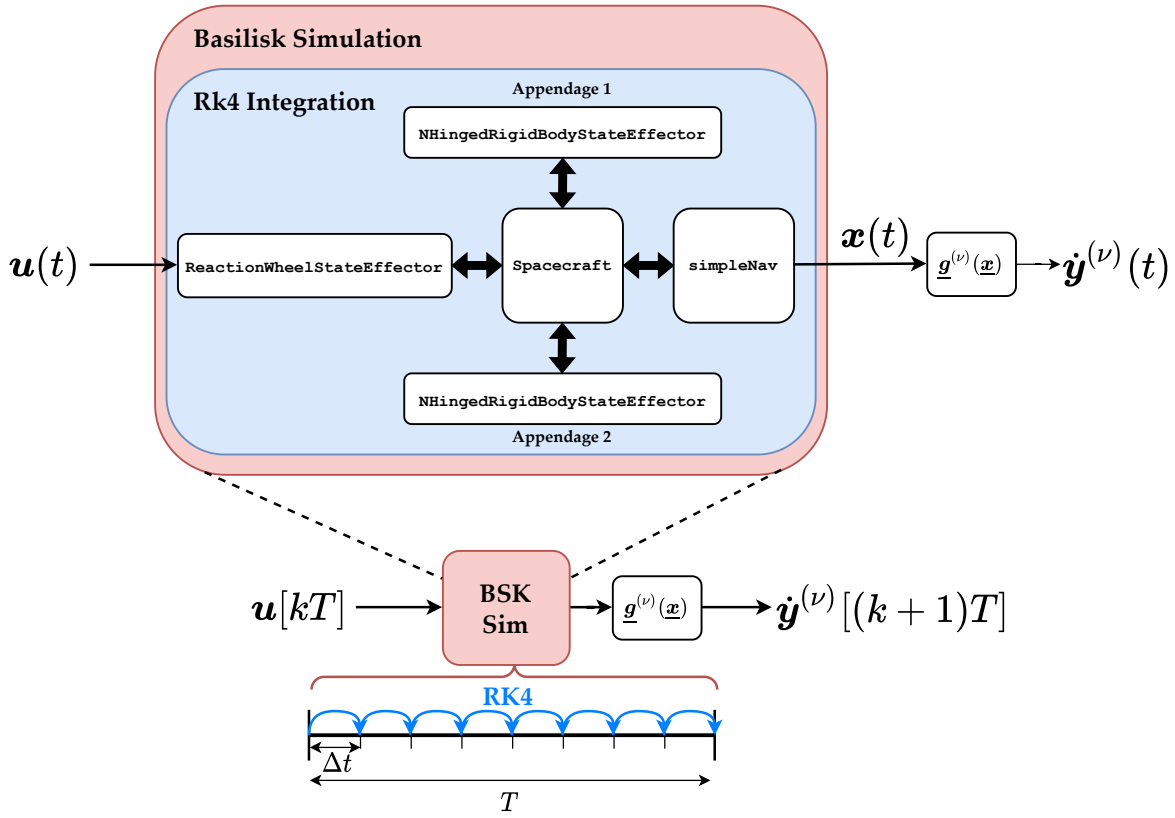


Figure 3.34: *Basilisk* model of a satellite with two flexible appendages

3.5.2 | Inverse Simulation Implimentation

To implement the integral InvSim algorithm, as detailed in Section 2.1, with the *Basilisk* simulation, illustrated in Fig. 3.34, the desired output/output derivative time history is discretised by the discretisation timestep T (Fig. 2.2). The effect of the input at the start of the timestep $\boldsymbol{u}[kT]$ on the output/output derivative at the end of the timestep $\dot{\boldsymbol{y}}^{(\nu)}[(k+1)T]$ can be obtained through a *Basilisk* simulation initialised with the current state of the system $\boldsymbol{x}[kT]$, and integrated over the timestep T using the integration timestep Δt as illustrated in Fig. 3.34.

The integral InvSim algorithm implementation with *Basilisk* is now fully detailed in Algorithm 2; however, the general approach should work with any existing simulation framework from which the output state of the system can be obtained given a prescribed input. Note that in line 9 of Algorithm 2, two *Basilisk* simulations are required per input to calculate the Jacobian partial derivatives through central differencing (Eq. (2.9)); the calculation of the Jacobian using *Basilisk* simulations is further illustrated in Fig. 3.35.

The InvSim algorithm (Algorithm 2) can therefore be implemented in Python, as this is the language used to interact with *Basilisk*, using the `numpy` package to perform all

Algorithm 2: Inverse Simulation with *Basilisk*

Generate timeseries of desired output/output derivative states $\dot{\boldsymbol{y}}_d^{(\nu)}(t)$;
Define an initial input estimate for the first step $\boldsymbol{u}_0$;
for $k = 1 \rightarrow \texttt{length}(\dot{\boldsymbol{y}}_d^{(\nu)}[kT])$ **do**
  Sample desired output/output derivative $\dot{\boldsymbol{y}}_d^{(\nu)}[(k+1)T]$;
  Initialise input estimate $\boldsymbol{u}_0[kT] = \begin{cases} \boldsymbol{u}[(k-1)T] & \text{for } k > 1 \\ \boldsymbol{u}_0 & \text{for } k = 1 \end{cases}$;
  Run forward simulation with initial input estimate
  $\dot{\boldsymbol{y}}^{(\nu)}[(k+1)T] \leftarrow$ **BSK Sim** $(\times 1)$;
  while $n < n_{max}$ **and** $\delta\dot{\boldsymbol{y}}^{(\nu)} > \delta\dot{\boldsymbol{y}}_{tol}^{(\nu)}$ **do**
    Calculate error $\delta\dot{\boldsymbol{y}}^{(\nu)} = \dot{\boldsymbol{y}}^{(\nu)}[(k+1)T] - \dot{\boldsymbol{y}}_d^{(\nu)}[(k+1)T]$;
    Calculate Jacobian $\mathbf{J}_{\boldsymbol{u}} \leftarrow$ **BSK Sim** $(\times 2N_u)$;
    Update input estimate $\boldsymbol{u}_n[kT] = \boldsymbol{u}_{n-1}[kT] - \mathbf{J}_{\boldsymbol{u}}^{-1}\delta\dot{\boldsymbol{y}}^{(\nu)}$;
    Run forward simulation with updated input solution
    $\dot{\boldsymbol{y}}^{(\nu)}[(k+1)T] \leftarrow$ **BSK Sim** $(\times 1)$;
    Update error to check tolerance $\delta\dot{\boldsymbol{y}}^{(\nu)} = \dot{\boldsymbol{y}}^{(\nu)}[(k+1)T] - \dot{\boldsymbol{y}}_d^{(\nu)}[(k+1)T]$;
  end
  Save input solution $\boldsymbol{u}[kT] \leftarrow \boldsymbol{u}_n[kT]$;
end

necessary mathematical operations, such as the direct inverse and MP pseudoinverse of the Jacobian. One complication found with this implementation, also found with *MATLAB*'s *Simscape* and *Simulink* frameworks, is that the forward simulation cannot easily be reset back to a previous state in time; this is required primarily for the calculation of the Jacobian and to facilitate the iterative nature of the NR loop. Instead, a new *Basilisk* simulation needs to be set up and initialised with the current state of the system $\boldsymbol{x}[kT]$, kept track of externally by the InvSim codebase. This greatly degrades the computational efficiency of the InvSim, due to the additional overhead of re-initialising *Basilisk* several times at each timestep. A possible solution, instead, would be to write the InvSim codebase in C++ and access the *Basilisk* simulation more directly in the backend, as opposed to the easier-to-use Python interface. This is, however, outside the scope of the current work, instead, this section simply provides a proof of concept as to the possibility of such an implementation of InvSim with an existing simulation framework.

Results of the *Basilisk* InvSim implementation for a simple rest-to-rest SLERP slew manoeuvre will be presented later in this section; however, first, proper modelling of

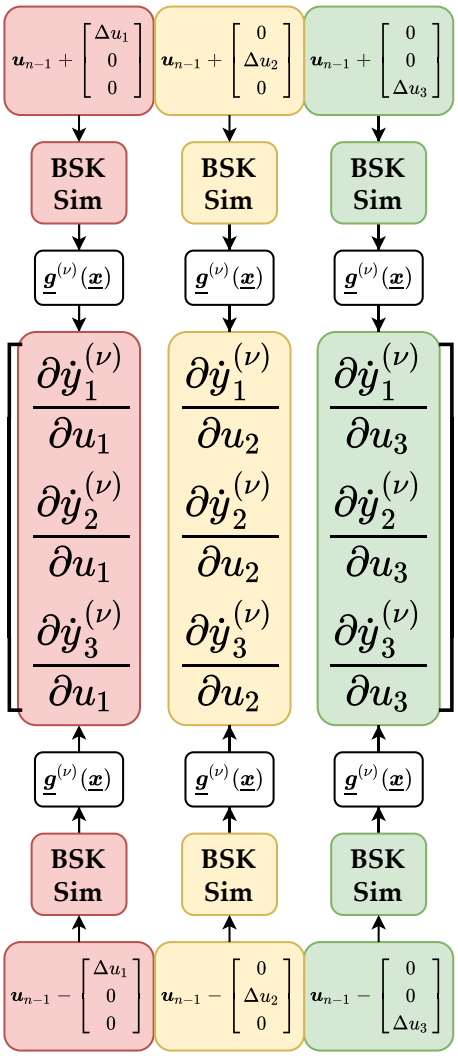


Figure 3.35: Jacobian calculation using *Basilisk*

the appendage stiffness using a lumped mass, torsional spring-damper, hinged model (`nHingedStateEffector`) must be completed.

3.5.3 | Lumped Element Appendage Modelling

The *lumped element model* considers each appendage as a series of point masses attached to a main rigid hub, connected in sequence by torsional spring-damper hinges. Each appendage will consist of N_{el} number of lumped-mass elements, with an illustrative example of an appendage with $N_{el} = 2$ elements shown in Fig. 3.36. Increasing the number of elements N_{el} in the appendage will allow for more flexible modes to be captured. As with the previous *hybrid coordinate model* (Section 3.1), the bending of each appendage is limited to one axis, defined as $\hat{\boldsymbol{y}}_{H_i}$ in this case. The position/deformation

of the appendage is defined in discrete coordinates, namely the angles ψ_i made by each mass m_i, its root hinge, and the $\hat{\boldsymbol{x}}_{H_i}$ axis.

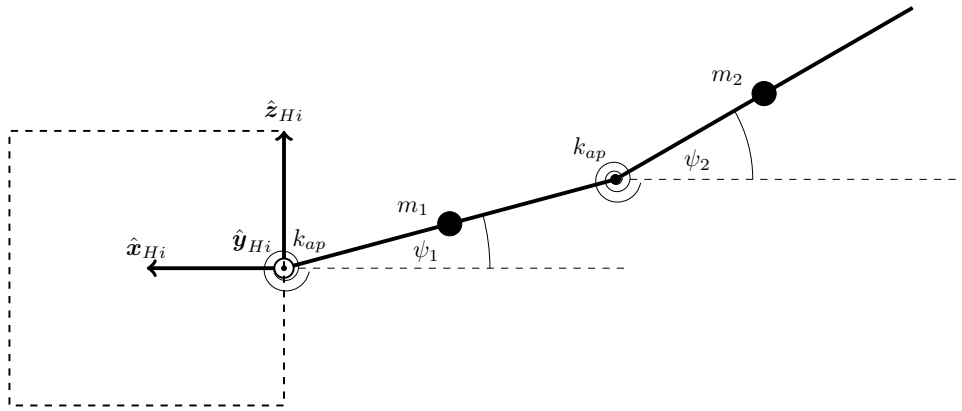


Figure 3.36: Example of an $N_{el} = 2$ lumped element flexible appendage

To achieve representative flexibility of the appendage using the lumped element approach, a suitable value for the spring constant k_{ap} of each hinge should be determined. First consider the potential energy $\mathcal{V}_{ap}$ of a continuous Euler-Bernoulli beam [177]:

$$\mathcal{V}_{ap} = \frac{1}{2}\int_0^{L_{ap}} EI\left(\frac{\delta^2 w}{\delta x^2}\right)^2 dx \tag{3.89}$$

where L_{ap} is the total length of the beam, and w is the deflection at position x along the beam.

The equivalent potential energy of the lumped element model, seen in Fig. 3.36, can be given in terms of the number of lumped mass elements used to represent the flexible appendage:

$$V_{ap} = \frac{1}{2}k_{ap}\sum_{i=1}^{N_{el}-1}(\psi_{i+1} - \psi_i)^2 \tag{3.90}$$

where V_{ap} is the potential energy of the lumped element flexible appendage, and ψ_i is the angle each segment of the beam makes with the horizontal plane, seen in Fig. 3.36.

Next, Eq. (3.89) is spatially discretised along the length of the beam, its x axis, by converting the integral to a finite sum:

$$\begin{aligned}\frac{1}{2}\int_0^{L_{ap}} EI\left(\frac{\delta^2 w}{\delta x^2}\right) dx &\approx \frac{1}{2}EI\lim_{\Delta x\to 0}\sum_{i=1}^{N_{el}-1}\left(\frac{\psi_{i+1}-\psi_i}{\Delta x}\right)^2\Delta x \\ &= \frac{1}{2}\lim_{\Delta x\to 0}\frac{EI}{\Delta x}\sum_{i=1}^{N_{el}-1}(\psi_{i+1}-\psi_i)^2\end{aligned} \tag{3.91}$$

where Δx is the length of each finite length of the appendage. By comparing Eqs. (3.90) and (3.91), substituting $\Delta x = L_{ap}/N_{el}$, and ignoring the limit, the continuous and discretised models can be equated:

$$\frac{1}{2}\frac{EIN}{L_{ap}}\sum_{i=1}^{N-1}(\psi_{i+1}-\psi_i)^2 = \frac{1}{2}k_{ap}\sum_{i=1}^{N-1}(\psi_{i+1}-\psi_i)^2 \tag{3.92}$$

It can therefore be deduced that an appropriate choice of the rotational spring constant is a function of Young's modulus E, second moment of area I, the total length of the appendage L_{ap}, and the number of lumped elements N_{el}:

$$k_{ap} = \frac{(EI)N_{el}}{L_{ap}} \tag{3.93}$$

This result is corroborated by Gambhir [178]. As seen in Eq. (3.91), the derivation of k_{ap} was based on a finite sum approximation of an integral which improves in accuracy as $\Delta x \to 0$ or as $N_{el} \to \infty$. To determine the appropriate value of N_{el} needed for an accurate lumped-parameter model, the potential energy of a beam with constant curvature κ can be compared for both the continuous and lumped-parameter models.

First, the curvature κ can be defined in continuous terms:

$$\kappa = \frac{\delta^2 w}{\delta x^2} \tag{3.94}$$

and discrete terms:

$$\kappa = \frac{\psi_{i+1}-\psi_i}{\Delta x} \tag{3.95}$$

which can then be substituted into Eqs. (3.89) and (3.90) respectively, giving the potential for a beam with constant curvature for the continuous Euler-Bernoulli beam appendage:

$$\mathcal{V}_{ap} = \frac{1}{2}(EI)L_{ap}\kappa^2 \tag{3.96}$$

and the lumped element case is also given with additional substitution of Eq. (3.93) and completing the summation:

$$\begin{aligned} V_{ap} &= \frac{1}{2}\frac{(EI)N_{el}}{L_{ap}}(N_{el}-1)\left(\kappa\frac{L_{ap}}{N}\right)^2 \\ &= \frac{1}{2}\frac{(N_{el}-1)}{N)el}(EI)L_{ap}\kappa^2 \end{aligned} \tag{3.97}$$

The relative potential energy $\tilde{V}$ of the lumped element model compared with the continuous Euler-Bernoulli beam can therefore be defined:

$$\tilde{V} = \frac{V_{ap}}{\mathcal{V}_{ap}} = \frac{N_{el} - 1}{N_{el}} \tag{3.98}$$

The results for $N_{el} = 2$ up to $N_{el} = 100$ can be seen plotted in Fig. 3.37. It can be seen that at $N_{el} = 2$ elements, the lumped element model has 50% of the potential energy when compared to the continuous Euler-Bernoulli beam. The accuracy of the lumped element model increases as $N_{el} \to \infty$ with $N_{el} = 5$ giving 80% accuracy, $N_{el} = 10$ giving 90% accuracy, and finally $N_{el} = 30$ gives 96.67% accuracy.

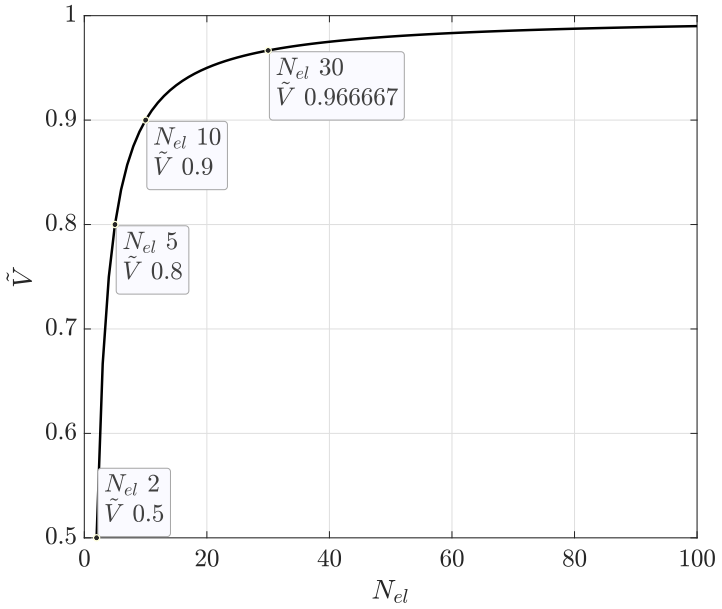


Figure 3.37: Relative potential energy of a lumped element beam relative to a continuous Euler-Bernoulli beam

3.5.4 | Results

To demonstrate the successful implementation of InvSim with the *Basilisk* software package, the solution to a 60-second, 90 deg, rest-to-rest slew manoeuvre about the body $\hat{\boldsymbol{y}}^B$ axis is presented here. The parameters of the *Basilisk* model with two `nHingedStateEffector` modules are configured to correspond closely to the model parameters defined previously in Table 3.1; however, an alternative bending stiffness of 300 Nm^2 is adopted.

The hyperparameters used by the InvSim to find the inverse dynamic solution to the *Basilisk* model are defined in Table 3.9. *Basilisk* natively uses Modified Rodrigues Parameters (MRPs) $\boldsymbol{p}^{B/I}$ to define the attitude, which can be converted to quaternions

Table 3.9: *Basilisk*: InvSim hyperparameters

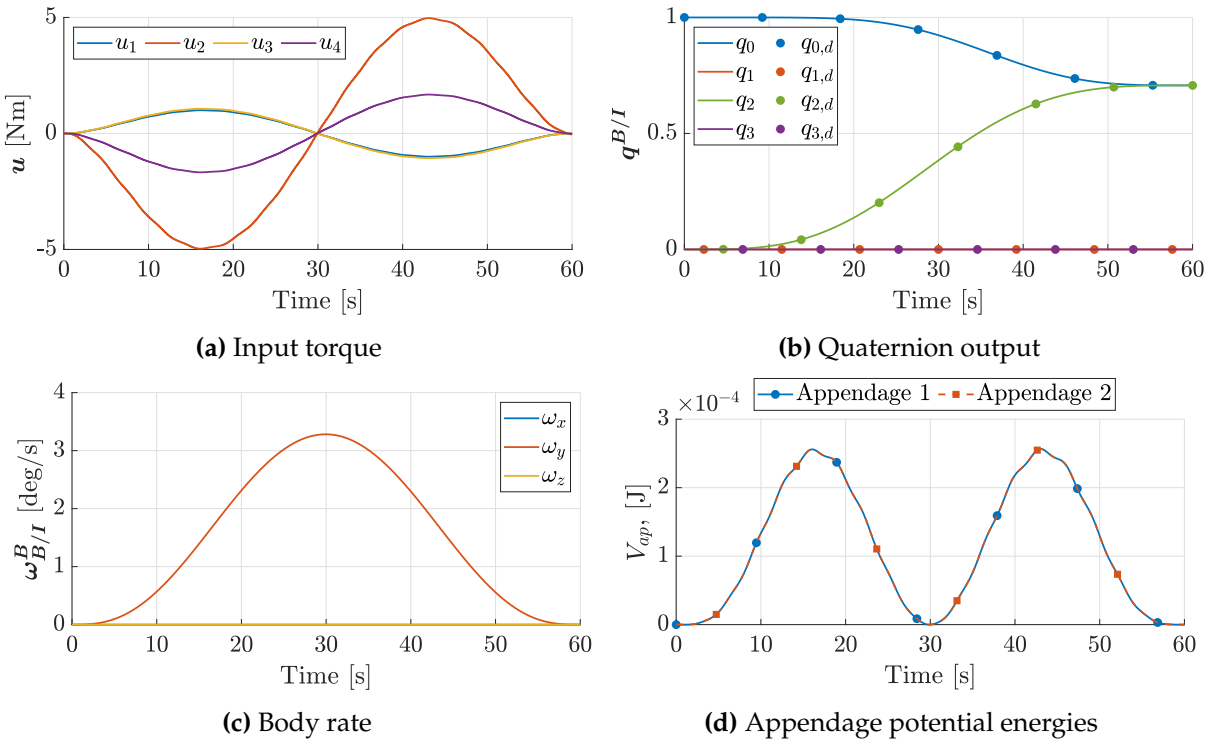
Hyperparameter	Value
Discretisation timestep, T	0.1 s
NR tolerance, $\delta\dot{y}_{tol}^{(\nu)}$	1×10^{-16}
Maximum NR iterations, n_{max}	10
Minimum finite difference perturbation, Δu_{min}	1 Nm
Desired output, $\dot{\boldsymbol{y}}_d^{(\nu)}$	$\dot{\boldsymbol{q}}_{d,1:3}^{(\nu)B/I}$
Desired output order, ν	1 (velocity)

using:

$$\boldsymbol{q}_0^{B/I} = \frac{1-||\boldsymbol{p}^{B/I}||^2}{1+||\boldsymbol{p}^{B/I}||^2} \tag{3.99a}$$

$$\boldsymbol{q}_{1:3}^{B/I} = \frac{2\boldsymbol{p}^{B/I}}{1+||\boldsymbol{p}^{B/I}||^2} \tag{3.99b}$$

with the ν^{th} order derivative of the vector component $\boldsymbol{q}_{1:3}^{B/I}$ being used as the desired output $\dot{\boldsymbol{y}}_d^{(\nu)}$.



(a) Input torque

(b) Quaternion output

(c) Body rate

(d) Appendage potential energies

Figure 3.38: InvSim solution from *Basilisk*

The results of the InvSim are presented in Fig. 3.38, with overall deformation across each appendage defined in terms of the total potential energy stored in the torsional

springs at each hinge. It can be seen that the 90 deg slew manoeuvre is successfully tracked (Fig. 3.38b) with the InvSim converging upon a smooth continuous input solution (Fig. 3.38a). As expected, given the orientation of the bending axes, the flexible modes of the appendages have been excited as illustrated in Fig. 3.38d. These results are also comparable to those seen with the hybrid-coordinate model used previously in this chapter (Fig. 3.17), verifying the validity of the *Basilisk* InvSim implementation.

The successful implementation of InvSim within an existing simulation framework (*Basilisk*) has proven the generality of the integral InvSim method and demonstrated the ability of InvSim to solve the dynamics of any pre-existing mathematical model, avoiding the need for redevelopment within some other environment such as *MATLAB*. Utilisation of existing simulation frameworks with InvSim could therefore provide a readily available model-based hybrid attitude control methodology (Section 3.3), and attitude path planning algorithm (Section 3.4).

The implementation of InvSim within an existing simulation framework, however, does not come without its challenges and caveats. It was found that much of the task of implementation was concerned with the interaction between the simulation framework and the InvSim, with the Jacobian requiring that many forward simulations be run. The open-source nature of *Basilisk* lends itself well to this task, however, it can be imagined that implementing InvSim with a framework that is not open-source, or does not have an extensive Application Programming Interface (API), would be a much more challenging endeavour.

3.6 | Summary

This chapter primarily focused on using InvSim to develop a model-based, hybrid, attitude control architecture for satellites with flexible appendages using InvSim. This involved first constructing a modular, multi-fidelity model for the InvSim to solve. It was identified that, depending on the stiffness of the system dynamics (i.e. level of fidelity), previously used RK4 numerical integration may need to be replaced with an alternative implicit method to ensure optimal InvSim speed and numerical stability. The InvSim feedforward control element was augmented with an additional PD feedback term, completing the hybrid control architecture. This feedback term was tuned using pole placement, ensuring a sufficient margin between the control bandwidth and the satellite's flexible modes. Combined with the use of a smooth polynomial slew trajectory, destabilising overexcitation of the flexible dynamics was avoided.

Compared to a traditional nonlinear PD tracking control law, the InvSim-based hybrid controller was shown to provide improved tracking accuracy using only a low-fidelity

model of the flexible dynamics, but proved highly sensitive to parametric model uncertainty. It was found that to achieve good attitude-tracking performance, uncertainties in the inertia and first modal frequency should be significantly small, $<1\%$ and < 0.5 rad/s respectively.

InvSim was also utilised to solve the inverse dynamics for an attitude path-planning algorithm, finding shaped slew manoeuvres that minimise appendage flexing. The algorithm uses the InvSim solution to assess the performance of potential attitude slew paths, informing which shaping parameters should be modified. The constrained optimisation method used also allows for additional constraints, such as maximum body rate or actuator saturation limits, to be implemented.

Finally, this chapter demonstrated that InvSim can be implemented within an existing simulation framework, provided its software allows interfacing with multiple forward simulations. An InvSim-based hybrid control architecture, such as that presented earlier in this chapter, can therefore be readily produced using pre-existing, pre-validated models, avoiding the need for re-derivation and re-validation. The next chapter will explore the use of InvSim, not for the direct control of satellites with flexible structures as covered in this chapter, but to aid in their design.

Chapter 4 | Design of Large Flexible Space Structures using Inverse Simulation

The integral-based InvSim algorithm, first demonstrated for application to satellite attitude dynamics in Chapter 2, was then used in the development of a hybrid, model-based, attitude control architecture for satellites with flexible appendages in Chapter 3; shown to be capable of improving attitude tracking performance, provided a sufficiently accurate model. Additionally, it was demonstrated that attitude slew paths that minimised appendage flexing could be found using an InvSim-based path planning algorithm. However, attitude control performance, and mitigation of structural deformation of a satellite are also influenced heavily by the underlying design of the satellite's structure, configuration, and even choice of attitude control actuators.

InvSim has a mature history in the investigation and assessment of vehicle dynamics as part of the design process, particularly in the field of rotorcraft dynamics [141], from which much of the development of InvSim originates. InvSim could therefore provide a useful framework for evaluating the full system dynamics of a satellite when performing a set of prescribed attitude manoeuvres required for a given mission scenario. This framework lends itself well to a "systems engineering" approach, allowing overall system requirements to inform the required attitude manoeuvrability, which subsequently can be used with InvSim to evaluate the dynamic performance of the current design configuration.

This chapter will focus on developing such a framework, providing a tool for the design of large gossamer space structures (LSSs). Specifically investigated will be the effect that different attitude control actuator types and configurations have on the structural deformation of LSSs during attitude slew manoeuvres. As discussed in Section 1.2.2, several types of actuators have been proposed for the attitude control of LSSs, including momentum exchange devices such as reaction wheels and control moment gyros, distributed magnetic torques, and utilisation of SRP torques. However, before such an

investigation can occur, a modular mathematical model capable of representing every actuator configuration of interest must be developed.

4.1 | Development of a Modular Mathematical Model

Several approaches exist to model the dynamics of LSSs, as detailed in Section 1.2.1. Here, however, the lumped-parameter approach is adopted, as it is the method most conducive to the modularity required for an InvSim-based design investigation. An LSS is therefore represented as a series of lumped point masses organised in a lattice structure. A Newtonian dynamic approach is adopted (Appendix C.1) with every aspect of the dynamics being deconstructed into forces that can be directly applied to each point mass of the LSS. Structural forces are represented by linear spring-dampers which connect the point masses, tuned to achieve the correct overall structural stiffness. The effect of each actuator type can be included by simply adding or removing their effective forces to the relevant point masses.

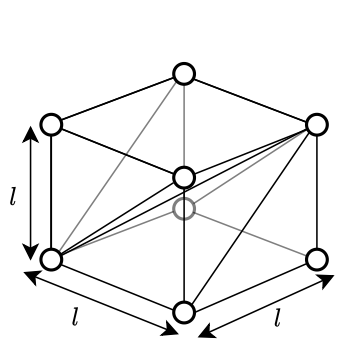
This approach results in a modular model where the physical properties of the LSS can be modified through the configuration of lattice shape, choice of physical connections between point masses, stiffness of each physical connection, mass of each point mass, and the actuators chosen. For the work in this chapter, however, investigations will be limited to planar LSSs with homogenous mass distribution and stiffness, similar to those studied by Robb *et al.* [66].

These planar LSSs consist of cubic unit elements with sidelength l; Fig. 4.1a shows the physical connections between each point mass. These cubic elements can be stacked in one direction to form a longer truss structure of width and height l, as seen in Fig. 4.1b, with those trusses being combined to give an overall planar LSS. An example of a $5\times5\times2$ mass element structure can be seen in Fig. 4.1c where N_x, N_y and N_z are the number of point masses in the $\hat{\boldsymbol{x}}^B$, $\hat{\boldsymbol{y}}^B$, and $\hat{\boldsymbol{z}}^B$ directions respectively.

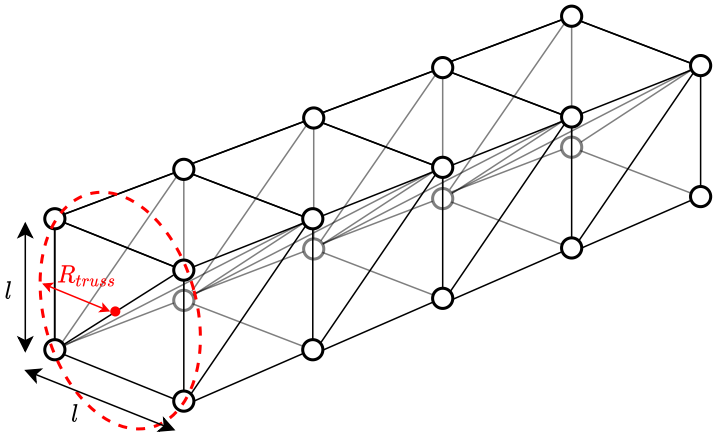
4.1.1 | Structural Forces

The definition of the structural forces is very general and not limited to the planar LSSs considered in this work. Each physical structural connection is represented by a linear spring-damper that applies forces on the two mass elements it is connected to, as seen in Fig. 4.2. The structural forces between two mass elements i and j can be given as:

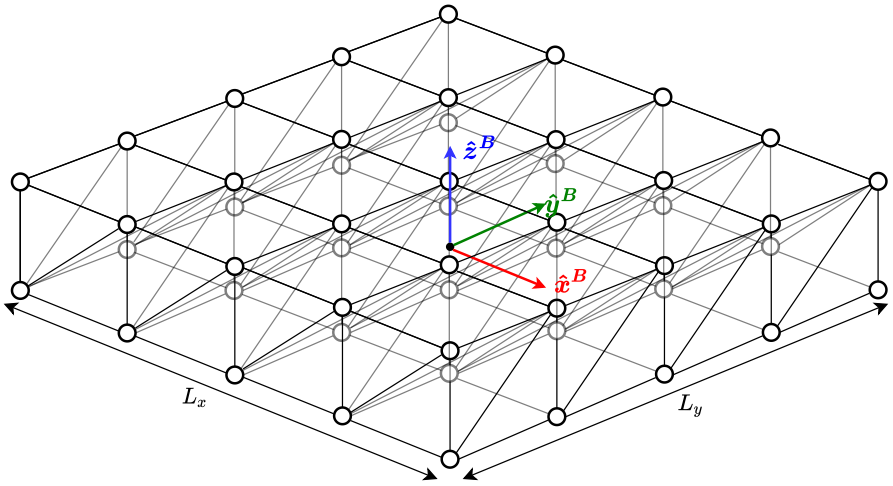
$$\boldsymbol{f}^B_{i,j} = -\boldsymbol{f}^B_{j,i} = k_{ij}\left(||\boldsymbol{r}^B_{j/i}|| - d^0_{j/i}\right)\hat{\boldsymbol{r}}^B_{j/i} + c_{ij}\left(\boldsymbol{v}^B_{j/i}\cdot\hat{\boldsymbol{r}}^B_{j/i}\right)\hat{\boldsymbol{r}}^B_{j/i} \tag{4.1}$$



(a) One cubic element of the LSS



(b) One truss element of the LSS, showing the radius of the cross-sectional area R_{truss}



(c) A 5×5×2 point mass LSS planar structure, comprised of 4 truss structures ($N_x = 5$, $N_y = 5$, $N_z = 2$)

Figure 4.1: Example of LSS planar structure using point-mass model, each connection between point masses produces structural forces as seen in Fig. 4.2 and Eq. (4.2)

.

where k_{ij} and c_{ij} are the spring and damping constants of the connection between mass i and j, $\boldsymbol{r}^B_{j/i}$ and $\boldsymbol{v}^B_{j/i}$ are the position and velocity of mass j relative to mass i measured in the body frame (B), and $d^0_{j/i}$ is the distance between mass i and j when no deformation in the structure is present. The total structural force $\boldsymbol{f}^B_{i,struct}$ acting on mass element i by all its connected neighbours, i.e. members of the group C_i, can therefore be given:

$$\boldsymbol{f}^B_{i,struct} = \sum_{j \in C_i} k_{ij} \left(||\boldsymbol{r}^B_{j/i}|| - d^0_{j/i} \right) \hat{\boldsymbol{r}}^B_{j/i} + c_{ij} \left(\boldsymbol{v}^B_{j/i} \cdot \hat{\boldsymbol{r}}^B_{j/i} \right) \hat{\boldsymbol{r}}^B_{j/i} \tag{4.2}$$

(a) Tension **(b)** Compression

Figure 4.2: Structural forces

Stiffness of a Planar LSS

The general planar LSSs used in this chapter, as seen in Fig. 4.1c, are assumed homogenous in the absence of any actuator mass, and therefore can be characterised by an overall bending stiffness about the longitudinal directions, $(EI)_{tot}$. Appropriate values for the individual spring constants k_{ij} need to be determined to achieve the characteristic bending stiffness of the full structure.

The full structure (Fig. 4.1c) can be decomposed into a series of trusses, each comprised of four longerons and a series of diagonal battens, as seen in Fig. 4.1b. The overall characteristic bending stiffness of one of these trusses is given as [179]:

$$(EI)_{truss} = 2(EA)_{truss}R^2_{truss} \tag{4.3}$$

where $(EA)_{truss}$ is the axial stiffness of the truss and R_{truss} is the radius of the circle enclosing the cross-section of the truss as seen in Fig. 4.1b.

Considering the full structure (Fig. 4.1c), it can be seen that there are 4 trusses connected side by side along each longitudinal axis. When considering the first truss by itself, the bending stiffness would be as seen in Eq. (4.3). However, the second truss will share half of its longerons and battens with the first, only contributing half the bending stiffness to the overall structure. Therefore, the total characteristic bending stiffness of the overall structure about each longitudinal axis will be:

$$(EI)_{total,x} = \left(1 + \frac{1}{2}(N_x - 2)\right)(EI)_{truss} \tag{4.4}$$

$$(EI)_{total,y} = \left(1 + \frac{1}{2}(N_y - 2)\right)(EI)_{truss} \tag{4.5}$$

where N_x and N_y are the number of point masses in the $\hat{\boldsymbol{x}}^B$ and $\hat{\boldsymbol{y}}^B$ directions respectively. The spring constant can now be derived simply from the axial stiffness of the trusses by considering the natural distance between the point masses at each end of the spring $d^0_{j/i}$:

$$k_{ij} = \frac{(EA)_{truss}}{d^0_{j/i}} \tag{4.6}$$

which can then be related to the bending stiffness of the truss using Eq. (4.3):

$$k_{ij} = \frac{(EI)_{truss}}{2R_{truss}^2 d_{j/i}^0} \tag{4.7}$$

Assuming the planar structure is square and homogenous, such as that seen in Fig. 4.1c, the spring constant k_{ij} of each spring can then be related directly to the characteristic total bending stiffness:

$$k_{ij} = \frac{(EI)_{tot}}{2\left(1 + \frac{1}{2}\left(N_{x,y} - 2\right)\right)\left(\frac{L_{x,y}}{N_{x,y} - 1}\right)^2 d_{j/i}^0} \tag{4.8}$$

where $L_{x,y}$ is the side length of the square planar structure, i.e. where $N_{x,y} = N_x = N_y$ and $L_{x,y} = L_x = L_y$. If the structure is non-square, the spring constants for the longerons and battens in each direction will differ.

The damping within the structure cannot easily be determined analytically, as the spring constants previously have been. For the most part, zero damping will be considered, or a very small value, to simply add extra stability to numerical simulations. One easy way to give a suitable damping constant for each connection in the structure is to use mass-proportional damping, or stiffness-proportional damping [180]. Stiffness-proportional damping can easily be implemented:

$$c_{ij} = \gamma k_{ij} \tag{4.9}$$

where γ is the constant of proportionality.

4.1.2 | Actuator Forces

Several types of actuators are employed in this study, including reaction-wheel sets (RWSs), axially embedded magnetorquers (MAGs) within structural elements, large current loopss (CLs) around the perimeter of the LSS, reflectivity control devices (RCDs), and moving masses (MMs). The dynamics of each actuator type are deconstructed into forces applied to the relevant point masses, and the changes to the mass of relevant point masses (MM actuator). The a^{th} actuator will be defined by the point masses it affects, i.e. a subset of point masses $\mathcal{A}_a$.

Reaction-Wheel Sets (RWS)

One of the simplest actuators considered for the attitude control of LSSs are sets of reaction wheels (Fig. 4.3) orientated along each cartesian body axis to give full 3-axis control.

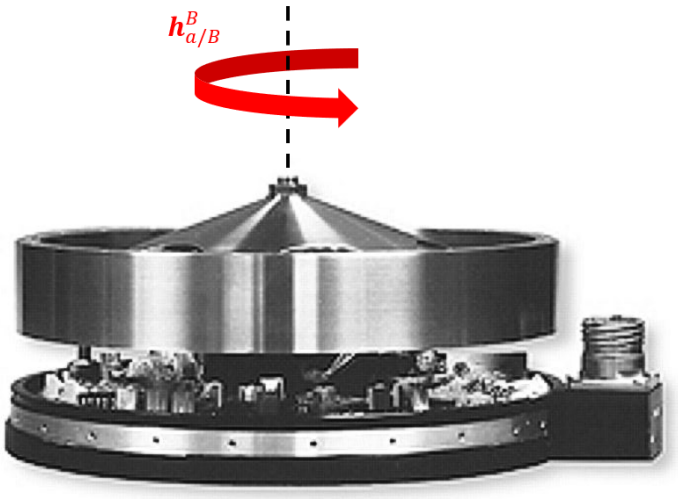


Figure 4.3: Example of a single reaction wheel (image courtesy of NASA)

In this model, an RWS actuator at a particular location in the LSS will apply forces to the closest 8 point masses surrounding it, equivalent to the effective torque they are producing (Fig. 4.4). This applies to the general planar LSSs used in this work, but will work with an LSS of any shape, provided that the RWS can be positioned between 8 point masses. The total effective torque produced by an RWS, $\boldsymbol{\tau}^B_a$, is given by:

$$\boldsymbol{\tau}^B_a = -\boldsymbol{u}_a - \boldsymbol{\omega}^B_{B/I} \times \boldsymbol{h}^B_{a/B} = -\boldsymbol{u}_a - \left[\boldsymbol{\omega}^B_{B/I}\right]_\times \boldsymbol{h}^B_{a/B} \tag{4.10}$$

where $\boldsymbol{u}_a$ is a general input vector for actuator a. In the case of a three-wheel RWS, this will be a three-element vector $\boldsymbol{u}_a \in \mathbb{R}^{3\times 1}$. The body rate of the LSS, $\boldsymbol{\omega}^B_{B/I}$, will be calculated in Section 4.1.3, and $\boldsymbol{h}^B_{a/B}$ is the angular momentum of the RWS relative to the body frame.

The effective forces associated with this reaction-wheel torque can be found by taking the cross product of the torque and the projection of the vector of the point mass relative to the actuator's position, $\boldsymbol{r}^B_{i/a}$, onto the plane defined by $\boldsymbol{\tau}_a$, as can be seen in Fig. 4.4. This gives:

$$\boldsymbol{f}^B_{i,a} = \frac{1}{8}\left[\boldsymbol{\tau}^B_a\right]_\times \underbrace{\left(\boldsymbol{r}^B_{i/a} - \left(\boldsymbol{r}^B_{i/a} \cdot \hat{\boldsymbol{\tau}}^B_a\right)\hat{\boldsymbol{\tau}}^B_a\right)}_{\text{projection}} \quad \text{for } i \in \mathcal{A}_a \tag{4.11}$$

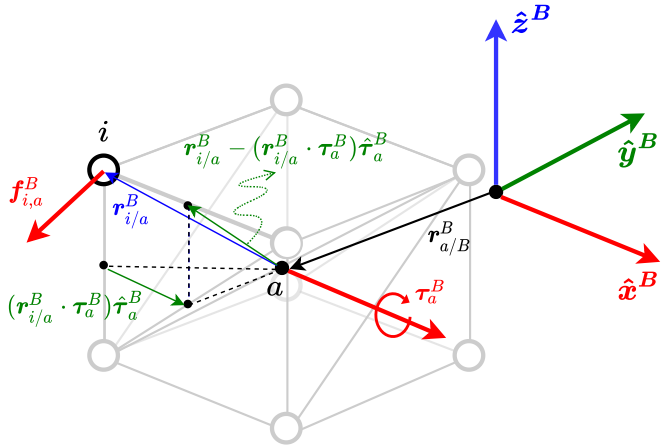


Figure 4.4: Reaction wheel torque converted to a force applied to the mass element i

where $\mathcal{A}_a$ is the group of mass elements associated with actuator a. In the case of an RWS the group $\mathcal{A}_a$ will always consist of 8 mass elements; however, this will vary for different actuator types. An additional differential equation is also required for the angular momentum of each RWS:

$$\dot{\boldsymbol{h}}^B_a = \boldsymbol{u}_a = \begin{bmatrix} u_x & u_y & u_z \end{bmatrix}^\top \tag{4.12}$$

where $[u_x \ u_y \ u_z]^\top$ are the RWS torques in each body frame direction. Assuming the RWS is centred within the 8 point masses, the mass can be equally distributed:

$$m_{i,a} = \frac{1}{8} m_a \ \text{ for } i \in \mathcal{A}_a \tag{4.13}$$

where $m_{i,a}$ is the mass contribution of actuator a to mass element i, and m_a, is the total mass of the a^{th} actuator, which in this case is an RWS.

Embedded Magnetorquers (MAG)

Utilisation of the interaction of magnetic dipoles in the spacecraft with the Earth's magnetic field can also be used to control the attitude of an LSS. Magnetic dipoles on a spacecraft are commonly generated by wrapping a conductor around some ferrous core material, such as that seen in Fig. 4.5. These magnetic coils could distributed throughout the structure of the LSS (Fig. 4.1), embedding them axially within each structural element (MAGs), as illustrated in Fig. 4.6.

In general, a magnetic torque can be defined as:

$$\boldsymbol{\tau}_{mag} = \boldsymbol{m}_d \times \boldsymbol{b}_{mag} \tag{4.14}$$

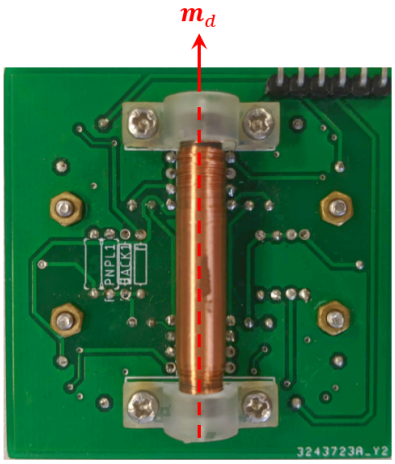


Figure 4.5: Example of a magnetic coil that can be embedded axially within an LSS structure [181]

where $\boldsymbol{\tau}_{mag}$ is the magnetic torque, $\boldsymbol{m}_d$ is the magnetic dipole, and $\boldsymbol{b}_{mag}$ is the local magnetic field vector. The cross product in Eq. (4.14) will limit the direction of torques that can be produced by the spacecraft to a plane (Appendix E); an issue that will become prevalent later.

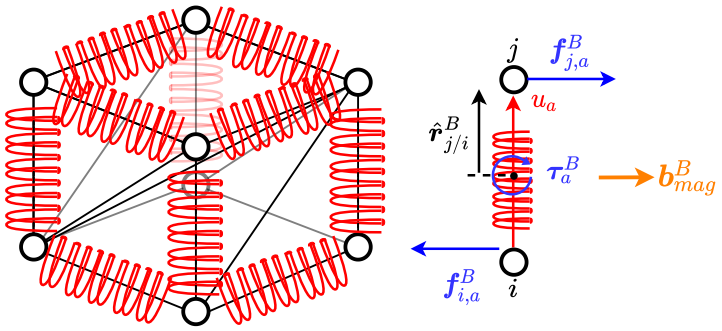


Figure 4.6: Embedded axial magnetorquer (MAG)

As illustrated in Fig. 4.6, the torque of each axially-embedded magnetorquer (MAG) can be found by considering the direction of the strut it is embedded within $\hat{\boldsymbol{r}}^B_{i/j}$:

$$\boldsymbol{\tau}^B_a = u_a \hat{\boldsymbol{r}}^B_{j/i} \times \boldsymbol{b}^B_{mag} \quad \text{for } i, j \in \mathcal{A}_a \tag{4.15}$$

where u_a is the input signal (the magnetic dipole moment) to the MAG. In the case of a MAG, the set of mass elements associated with $\mathcal{A}_a$ will only ever have two members, i.e. mass elements at each end of the MAG (i and j). The magnetic torques need to be deconstructed into the equivalent forces on the mass elements at each end of the MAG, as illustrated in Fig. 4.6, giving:

$$\boldsymbol{\tau}^B_a = \frac{1}{2}\boldsymbol{r}^B_{i/j} \times \boldsymbol{f}^B_{i,a} + \frac{1}{2}\boldsymbol{r}^B_{j/i} \times \boldsymbol{f}^B_{j,a} \tag{4.16}$$

Then, considering that $\boldsymbol{r}^B_{i/j} = -\boldsymbol{r}^B_{j/i}$ and $\boldsymbol{f}^B_{i,a} = -\boldsymbol{f}^B_{j,a}$, the forces applied to each point mass can be found using a MP pseudoinverse (Eq. (A.19)):

$$\boldsymbol{f}^B_{a,j} = -\boldsymbol{f}^B_{a,i} = \left[\boldsymbol{r}^B_{j/i}\right]^+_\times \boldsymbol{\tau}^B_a \tag{4.17}$$

The mass contribution of a MAG is simply split between the point masses on each side:

$$m_{i,a} = \frac{1}{2} m_a \quad \text{for } i \in \mathcal{A}_a \tag{4.18}$$

Large Current Loops (CL)

The size of the LSSs can be utilised to create larger current loops (CL) consisting of conductors wrapped around the perimeter of the structure (i.e. vacuum core), allowing higher magnetic dipoles than MAGs to be generated. The specific CLs used for the planar LSSs considered in this work can be seen for each axis in Fig. 4.7.

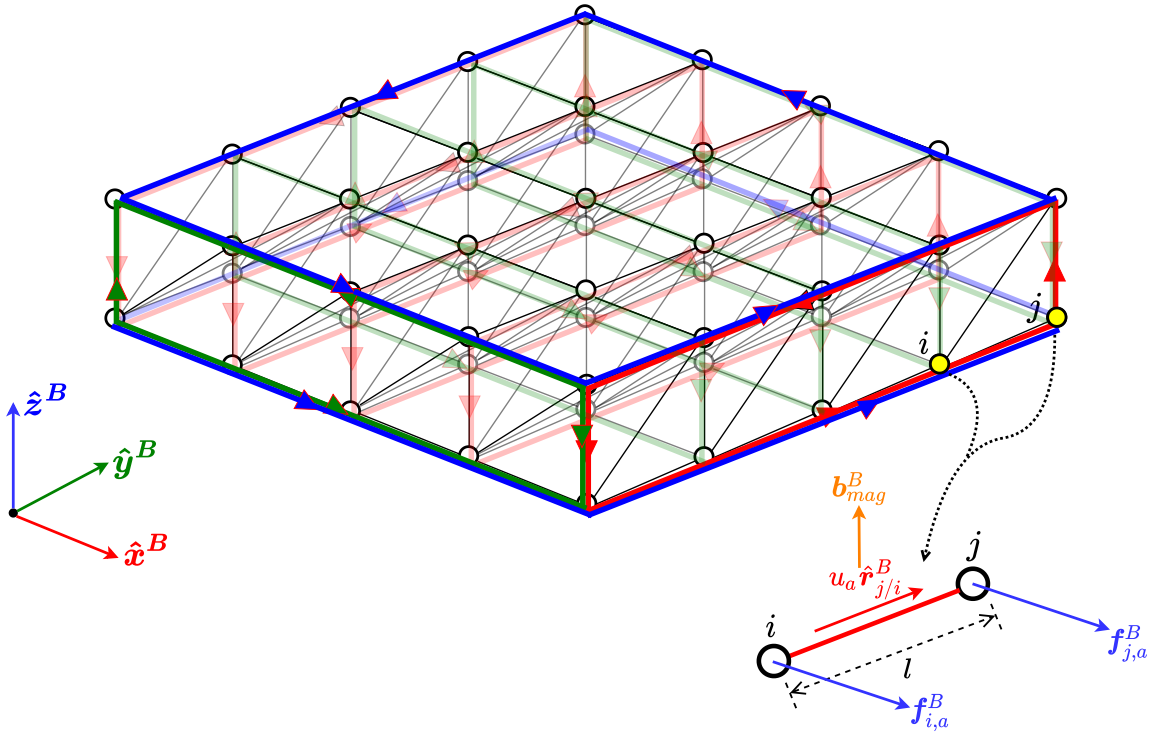


Figure 4.7: Large current loops (CL)

The torque produced by each CL can easily be deconstructed by considering the Lorenz forces produced by each small segment of the conductor between adjacent point masses:

$$\boldsymbol{f}^B_{i,a} + \boldsymbol{f}^B_{j,a} = u_a l \hat{\boldsymbol{r}}^B_{j/i} \times \boldsymbol{b}^B_{mag} \tag{4.19}$$

where $u_a \in \mathbb{R}^{1\times1}$, in this case, is the current flowing in the conductor times the number of loops passing between point mass i and j, and l is the length of the segment of the conductor, defined in Fig. 4.1a and assumed to be constant. The connection between the two masses i and j can be considered a subset $\mathcal{B}_k = \{i, j\}$ with the full set $\mathcal{A}_a$ defining the entire path of the CL actuator:

$$\mathcal{A}_a = \{\mathcal{B}_1,\ \mathcal{B}_2,\ \cdots,\ \mathcal{B}_k,\ \cdots\} \tag{4.20}$$

Then, considering that $\boldsymbol{f}^B_{i,a} = \boldsymbol{f}^B_{j,a}$ each force is given as:

$$\boldsymbol{f}^B_{i,a} = \boldsymbol{f}^B_{j,a} = \frac{1}{2} u_a\, l\, \hat{\boldsymbol{r}}^B_{j/i} \times \boldsymbol{b}^B_{mag} \ \text{ for } \mathcal{B}_k \in \mathcal{A}_a \tag{4.21}$$

The mass contribution of the CL actuator is defined as:

$$m_{i,a} = \frac{1}{N_{\mathcal{A}_a}} m_a \tag{4.22}$$

where $N_{\mathcal{A}_a}$ is the total number of unique point masses in the set $\mathcal{A}_a$, considering each connection $\mathcal{B}_k$ of the full CL path.

Reflectivity Control Devices (RCD)

SRP will not only induce a net force on the LSS but also induce torques if the centres of mass and pressure are offset. The realisation of this torque within the model can be achieved simply by considering the SRP force applied to each point mass individually. Assuming the top surface of the LSS, defined by the subset of point masses $\mathcal{S}$, is an ideal reflector, the force experienced by each point mass can be given [182]:

$$\boldsymbol{f}^I_i = -p_{sol} A_i \left(1 + \eta_{s,i}\right) \cos^2\theta_{srp} \hat{\boldsymbol{n}}^I \ \text{ for } i \in \mathcal{S} \tag{4.23}$$

where $\eta_{s,i} \in [0, 1]$ is a coefficient of reflectivity for each point mass, p_{sol} is the magnitude of the SRP, $\hat{\boldsymbol{n}}^I$ is a unit vector normal to the surface of the structure, and A_i is the effective area of each point mass:

$$A_i = \frac{A_{tot}}{N_{\mathcal{S}}} = \frac{L^2_{x,y}}{N_{\mathcal{S}}} \tag{4.24}$$

where $N_{\mathcal{S}}$ is the number of point masses in $\mathcal{S}$. The cosine of the angle between the surface normal and the direction of the sun, $\cos\theta_{srp}$, is given through the dot product:

$$\cos\theta_{srp} = \hat{\boldsymbol{n}}^I \cdot \hat{\boldsymbol{r}}^I_{sun/B} \tag{4.25}$$

where $\hat{\boldsymbol{r}}^I_{sun/B}$ is the unit vector of the sun relative to the LSS body measured in inertial axes, and $\hat{\boldsymbol{n}}^I$ will be defined later in Section 4.1.3.

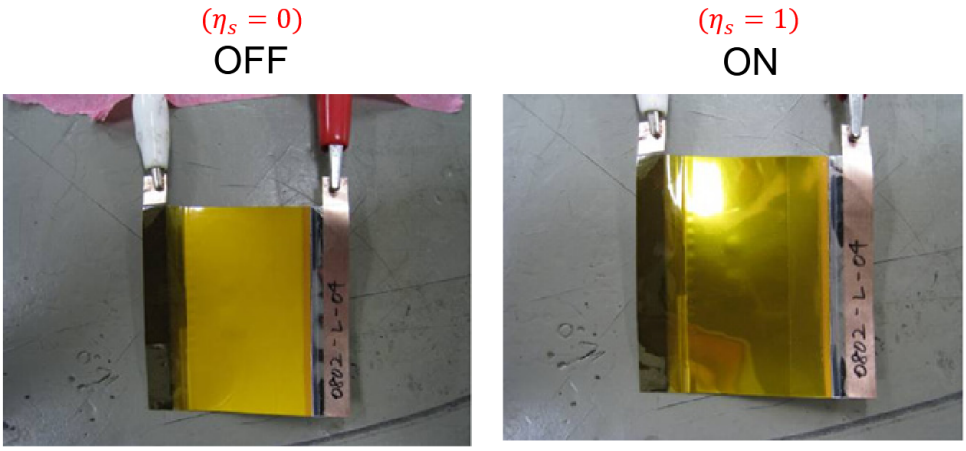
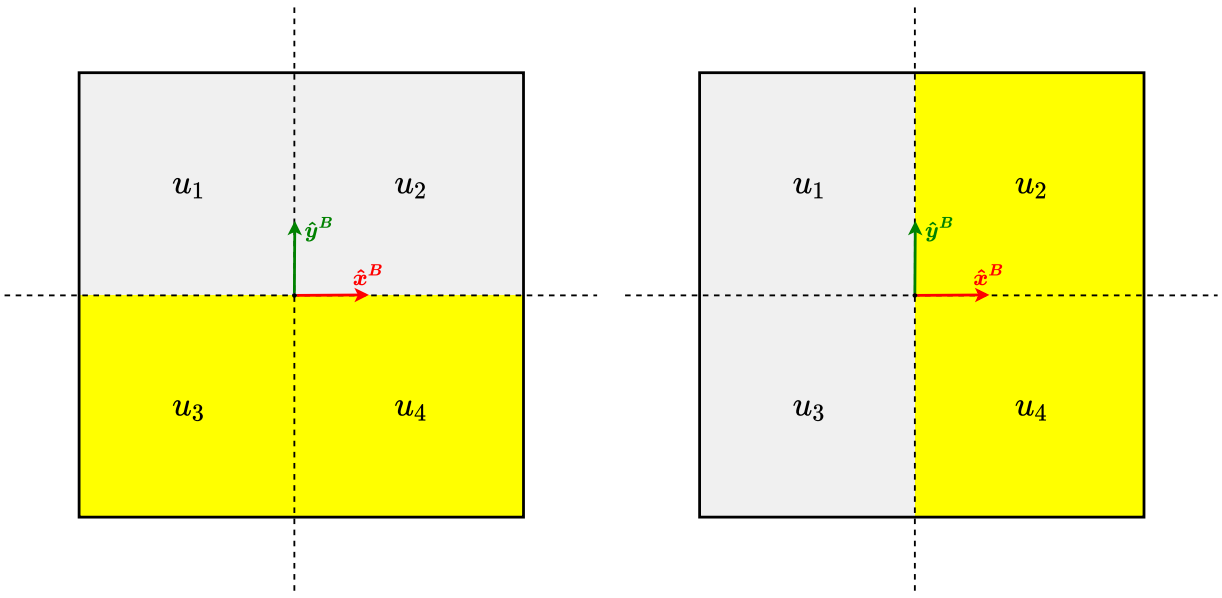


Figure 4.8: Example of an RCD patch used on the IKAROS solar sail [88]

One approach to offset the centres of mass and pressure is to make use of RCDs, thin film patches that can electrically modulate their reflectivity (Fig. 4.8). RCDs split the reflective surface of a solar sail, or solar reflector, into a grid pattern allowing for the reflectivity to be modulated and for the overall centre of pressure to be moved, a concept well described and investigated by Borggrafe *et al.* [89].



(a) Positive rotation about x-axis, $\boldsymbol{u}_a = [0\ 0\ 1\ 1]^\top$ **(b)** Positive rotation about y-axis, $\boldsymbol{u}_a = [0\ 1\ 0\ 1]^\top$

Figure 4.9: Reflective control device actuator (RCD) configuration for rotations about longitudinal axes. Showing fully reflecting (yellow), and fully absorbing (grey) patches

These RCD actuators are implemented in the modular model, assuming no mass, for planar LSSs by simply dividing the reflective surface into a suitable grid pattern where

each patch is defined by the subset of masses $\mathcal{A}_a \in \mathcal{S}$. Each patch of the grid is therefore considered a separate actuator with input $u_a \in [0, 1]$, where 0 is completely absorbing, and 1 is completely reflecting:

$$\eta_{s,i} = u_a \quad \text{for } i \in \mathcal{A}_a \tag{4.26}$$

The specific 2×2 RCD grid used for the LSSs in this chapter can be seen in Fig. 4.9, showing the inputs required for a positive rotation about each longitudinal axis.

Moving Mass (MM)

Another approach to generating a torque through SRP is to move the centre of mass by moving control/sliding masses within the LSS structure in a configuration such as that seen in Fig. 4.10 (MM).

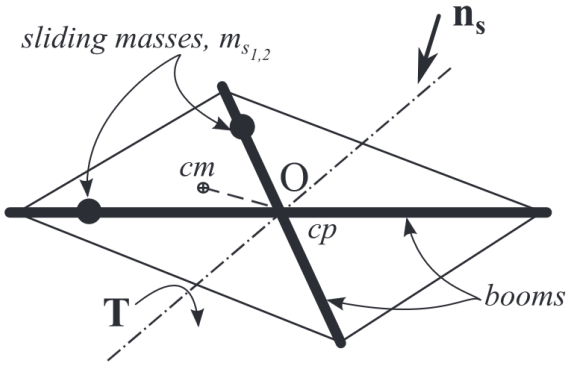


Figure 4.10: Example of a moving mass actuator configuration for a solar sail [183]. Showing the direction of the sun ($\boldsymbol{n}_s$), centre of mass (cm), centre of pressure (cp), and resultant torque $\mathbf{T}$

The MM actuator is realised within the modular model by modifying the mass of nearby point-mass elements to represent the movement of the control mass, avoiding the need to add a unique additional point mass to the system.

In the case of the planar LSSs considered in this work, the movement of the MM will be limited to the $\hat{\boldsymbol{x}}^B$-$\hat{\boldsymbol{y}}^B$ plane, as seen in Fig. 4.11. The input for an MM actuator will therefore be a two-element input vector, $\boldsymbol{u}_a \in \mathbb{R}^{2\times 1}$. The set of masses associated with the actuator $\mathcal{A}_a$ will change as the MM moves, always containing the 8 closest masses. The full position vector of the MM can be defined as:

$$\boldsymbol{r}^B_{a/B} = \begin{bmatrix} \boldsymbol{u}_a^\top & 0 \end{bmatrix}^\top = \begin{bmatrix} u_x & u_y & 0 \end{bmatrix}^\top \tag{4.27}$$

with the set $\mathcal{A}_a$ found from:

$$\mathcal{A}_a = \left\{ \arg\min_i {}_8 \; ||\boldsymbol{r}^B_{i/B} - \boldsymbol{r}^B_{a/B}|| \right\} \tag{4.28}$$

which returns the eightpoint masses closest to the current position of the control mass. The change in mass of each mass element, Δm_i, that considers the additional control mass m_a can be found using the simultaneous equations:

$$m_a = \sum_{i \in \mathcal{A}_a} \Delta m_i \tag{4.29a}$$

$$m_a \boldsymbol{r}^B_{a/B} = \sum_{i \in \mathcal{A}_a} \Delta m_i \boldsymbol{r}^B_{i/B} \tag{4.29b}$$

where $i \in \mathcal{A}_a$. The simultaneous equations in Eq. (4.29) can be formed into a system of equations to be solved:

$$\begin{bmatrix} 1 & 1 & \cdots & 1 \\ \boldsymbol{r}^B_{i/B} & \boldsymbol{r}^B_{j/B} & \cdots & \boldsymbol{r}^B_{k/B} \end{bmatrix} \begin{bmatrix} \Delta m_i \\ \Delta m_j \\ \vdots \\ \Delta m_k \end{bmatrix} - \begin{bmatrix} m_a \\ m_a \boldsymbol{r}^B_{a/B} \end{bmatrix} = 0 \;\text{ for }\; \{i, j, \cdots, k\} \in \mathcal{A}_a \tag{4.30}$$

which is in the form:

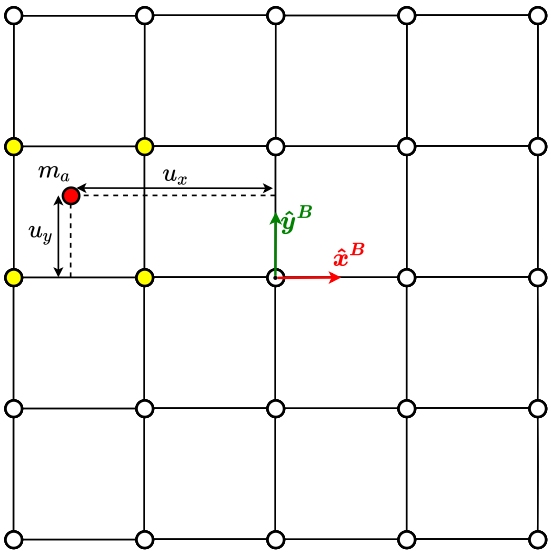
$$\mathbf{A}\boldsymbol{x} - \boldsymbol{b} = 0 \tag{4.31}$$

A least-square-fit to Eq. (4.30) can then be found with additional inequality constraints to ensure that the change in mass is only ever positive, since the movement of the control mass should never decrease the mass at any point in the LSS; though such solutions exist mathematically. This least-squares-fit problem takes the form:

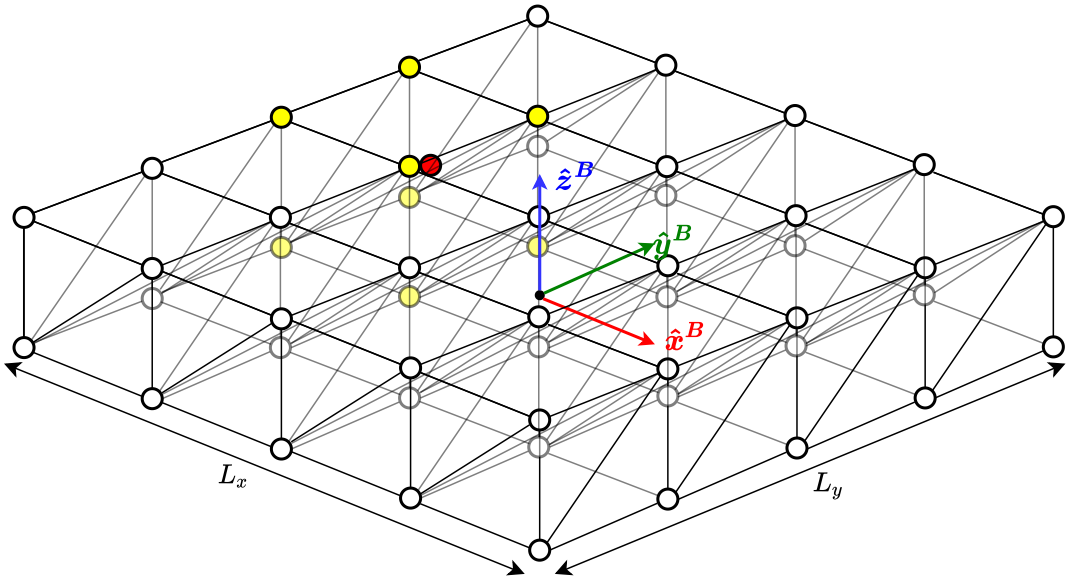
$$\begin{bmatrix} \Delta m_i \\ \Delta m_j \\ \vdots \\ \Delta m_k \end{bmatrix} = \arg\min_{\boldsymbol{x}} \frac{1}{2} ||\mathbf{A}\boldsymbol{x} - \boldsymbol{b}||^2 \;\text{ such that } 0 \leq x_i \leq m_a \tag{4.32}$$

which is solved using the *MATLAB* function `lsqlin()`. This gives the contribution of the MM to the point mass structure:

$$m_{i,a} = \Delta m_i \;\text{ for } i \in \mathcal{A}_a \tag{4.33}$$



(a) Top down view: Input vector $\boldsymbol{u}_a = [u_1\ u_2]^\top$ defines position of actuator mass in the x-y plane



(b) Isometric view

Figure 4.11: Moving mass actuator (MM): Actuator mass (red) and the affected 8 point-masses in the lattice (yellow)

4.1.3 | Dynamics and Kinematics

The total force enacted upon each mass by all of the actuators added to the model can be given by:

$$\boldsymbol{f}^B_{i,actuators} = \sum_a \boldsymbol{f}^B_{i,a} \quad \{\forall a : i \in \mathcal{A}_a\} \tag{4.34}$$

i.e. a summation of all actuator forces for which the mass element i belongs to the group $\mathcal{A}_a$ of mass elements associated with actuator a. The inertial accelerations of each point mass can therefore be found through a Newtonian approach (Appendix C.1):

$$\boldsymbol{a}^I_{i/I} = \mathbf{C}^{I/B}\left(\frac{\boldsymbol{f}^B_{i,struct} + \boldsymbol{f}^B_{i,actuators}}{m_i}\right) + \frac{\boldsymbol{f}^I_{i,srp}}{m_i} \tag{4.35}$$

where $\mathbf{C}^{I/B}$ is the direction cosine matrix (DCM) rotating vectors measured in the body frame (B) into the inertial frame (I), and m_i is the mass of each point mass which will be the sum of the structural and actuator masses:

$$m_i = m_{i,0} + \sum_a m_{i,a} \quad \{\forall a : i \in \mathcal{A}_a\} \tag{4.36}$$

where $m_{i,0}$ is the mass of point mass i due to the structure only, which for a homogenous structure such as that considered here will be:

$$m_{i,0} = \frac{\rho_{lss} L^2}{N_x N_y N_z} = \frac{\rho_{lss} L^2}{N} \tag{4.37}$$

where ρ_{lss} is the area mass density, and $N = N_x N_y N_z$ is the total number of point masses in the LSS structure. Considering the centroid of all point masses as the origin of the body frame, the inertial acceleration of the body frame $\boldsymbol{a}^I_{B/I}$ can be found:

$$\boldsymbol{a}^I_{B/I} = \frac{1}{N}\sum_{i=1}^{N} \boldsymbol{a}^I_{i/I} \tag{4.38}$$

The complete state vector that fully defines the system will therefore be:

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{r}^I_{B/I} & \boldsymbol{v}^I_{B/I} & \boldsymbol{r}^I_{i/B} & \cdots & \boldsymbol{r}^I_{N/B} & \boldsymbol{v}^I_{i/B} & \cdots & \boldsymbol{v}^I_{N/B} & \boldsymbol{h}^B_{a/B} & \cdots \end{bmatrix}^\top \tag{4.39}$$

where $\boldsymbol{h}^B_{a/B}$ are the angular momenta of the RWS if any exist in the model.

To complete the mathematical model, the kinematics of the LSS have to be defined to solve the dynamics, and to give relevant outputs for use in the InvSim. The attitude and body rates of the LSS are therefore required to be known. This would be trivial were the spacecraft rigid; however, defining the overall attitude of the LSS given the inertial

positions of the point masses representing it is non-trivial. Additionally, numerical differencing in the *time* domain to calculate the body rates should be avoided, as this can lead to instability in the InvSim process [124].

Best-Fit Attitude

The attitude of the LSS can be found through consideration of the active rotation matrix $\mathbf{R}^{B/I}$ which rotates the point masses representing the LSS from their original positions $\boldsymbol{r}^0_{i/B}$, to their current positions $\boldsymbol{r}^I_{i/B}$. This can be found through the following least-square-fit solution [184]:

$$\min_{\mathbf{R}^{B/I}} \left(\sum_{i=1}^{N} \left\| \boldsymbol{r}^I_{i/B} - \mathbf{R}^{B/I} \boldsymbol{r}^0_{i/B} \right\|^2 \right) \tag{4.40}$$

where $\boldsymbol{r}^0_{i/B}$ are point mass positions when no rotation (i.e frame B and I are aligned) or deformation is present. A solution to the least-square-fit problem can be found by first constructing the covariance matrix [184]:

$$\boldsymbol{\Sigma}_{att} = \sum_{i=1}^{N} \boldsymbol{r}^0_{i/B} \boldsymbol{r}^I_{i/B}{}^{\top} \tag{4.41}$$

which can be decomposed using the single-value-decomposition (SVD) [184]:

$$\boldsymbol{\Sigma}_{att} = \mathbf{U} \boldsymbol{\Lambda} \mathbf{V}^{\top} \tag{4.42}$$

where $\mathbf{U}$ and $\mathbf{V}$ are 3×3 matrices, and $\boldsymbol{\Lambda}$ is a 3×3 diagonal matrix. The rotation matrix can then be found:

$$\mathbf{R}^{B/I} = \mathbf{V}\mathbf{U}^{\top} \tag{4.43}$$

and the equivalent DCM can be given by the transpose:

$$\mathbf{C}^{B/I} = \left[\mathbf{R}^{B/I} \right]^{\top} \tag{4.44}$$

which can be decomposed further to give an attitude quaternion or any Euler-Angle set [81].

Best-Fit Body Rate

Similarly, a least-squares fit solution can be found for the overall body rate of the LSS $\boldsymbol{\omega}^B_{B/I}$. This can be found by first considering the definition of the velocity of a single

point mass in the rotating body frame:

$$\boldsymbol{v}^I_{i/B} = \dot{\boldsymbol{r}}^I_{i/B} + \boldsymbol{\omega}^I_{B/I} \times \boldsymbol{r}^I_{i/B} = \dot{\boldsymbol{r}}^I_{i/B} + \left[\boldsymbol{\omega}^I_{B/I}\right]_\times \boldsymbol{r}^I_{i/B} \tag{4.45}$$

where $\dot{\boldsymbol{r}}^I_{i/B}$ is the velocity of point mass i within the rotating body frame. Applying the transpose property of the skew-symmetric matrix, $[\boldsymbol{a}]_\times \boldsymbol{b} = [\boldsymbol{b}]^\top_\times \boldsymbol{a}$, and assuming $\dot{\boldsymbol{r}}^I_{i/B} = 0$ gives:

$$\boldsymbol{v}^I_{i/B} = \left[\boldsymbol{\omega}^I_{B/I}\right]_\times \boldsymbol{r}^I_{i/B} = \left[\boldsymbol{r}^I_{i/B}\right]^\top_\times \boldsymbol{\omega}^I_{B/I} \tag{4.46}$$

a system of equations can then be formed by applying Eq. (4.46) for all point masses and stacking the skew-symmetric matrices vertically:

$$\begin{bmatrix} \boldsymbol{v}^I_{1/B} \\ \boldsymbol{v}^I_{2/B} \\ \vdots \\ \boldsymbol{v}^I_{N/B} \end{bmatrix} = \begin{bmatrix} \left[\boldsymbol{r}^I_{1/B}\right]_\times \\ \left[\boldsymbol{r}^I_{2/B}\right]_\times \\ \vdots \\ \left[\boldsymbol{r}^I_{N/B}\right]_\times \end{bmatrix} \boldsymbol{\omega}^I_{B/I} \tag{4.47}$$

which can be solved using a MP pseudoinverse (Eq. (A.19)) to find the least squares fit for $\boldsymbol{\omega}^I_{B/I}$ that satisfies Eq. (4.47):

$$\boldsymbol{\omega}^I_{B/I} = \begin{bmatrix} \left[\boldsymbol{r}^I_{1/B}\right]_\times \\ \left[\boldsymbol{r}^I_{2/B}\right]_\times \\ \vdots \\ \left[\boldsymbol{r}^I_{N/B}\right]_\times \end{bmatrix}^+ \begin{bmatrix} \boldsymbol{v}^I_{1/B} \\ \boldsymbol{v}^I_{2/B} \\ \vdots \\ \boldsymbol{v}^I_{N/B} \end{bmatrix} \tag{4.48}$$

The MP pseudoinverse can be used in this case over the *MATLAB* `lsqlin()` function as there are no inequality constraints to consider. This can then be rotated into the body frame using the DCM calculated earlier:

$$\boldsymbol{\omega}^B_{B/I} = \mathbf{C}^{B/I} \boldsymbol{\omega}^I_{B/I} \tag{4.49}$$

giving the body rate in body axes (B) which is the usual convention. These body rates can also be used to get the quaternion or Euler-angle kinematics of the body frame (Appendix B.3).

Surface Deformation of a Planar LSS

One aspect of the LSS dynamics that could be of interest is the effect that different actuator types will have on the deformation of the structure. For example, in the case of an

orbital solar reflector, deformation of the reflector surface could degrade overall solar power delivery to ground stations, and should therefore be quantified by the model.

Here, the deflection angle across the reflective surface of the LSS, defined by the subset of point masses $\mathcal{S}$, will be calculated by comparing the local normal of the surface to the $\hat{\boldsymbol{z}}^B$ axis defined by the *best-fit* attitude. Determining the normals to the surface at each point mass's position is, however, nontrivial. One solution would be to fit a polynomial or piecewise polynomial surface to the set of point mass positions of the surface in question, then differentiate to obtain the normal. This approach, however, could be highly intensive computationally, especially when attempting to calculate the normals for every timestep of the simulation. Instead, a more approximate, computationally cheap alternative is desirable.

One such approach is to take a neighbourhood of the points closest to the point on the surface being evaluated and find a least squares fitting plane that minimises [185]:

$$\min_{\hat{\boldsymbol{n}}_i^B} \left\{ \sum_{j \in \mathcal{N}_i} \left[\left(\boldsymbol{r}_{j/B}^B - \boldsymbol{r}_{o/B}^B \right) \cdot \hat{\boldsymbol{n}}_i^B \right]^2 \right\} \ \forall i \in \mathcal{S} \tag{4.50}$$

where $\mathcal{N}_i$ is the group of six closest points to i, including i itself, $\mathcal{S}$ is the group of point masses that constitute the reflective upper surface of the LSS, and $\boldsymbol{r}_{o/B}^B$ is the centroid of that group of points. A neighbourhood of six masses was used to ensure good accuracy of the calculated normal while avoiding the loss of resolution that would come from the averaging effect of using a larger number of points. Considering the position of each point in $\mathcal{N}_i$ relative to the centroid $\boldsymbol{r}_{j/o}^B = \boldsymbol{r}_{j/B}^B - \boldsymbol{r}_{o/B}^B$, the covariance matrix can be found through the sum of the outer products:

$$\boldsymbol{\Sigma}_{norm} = \sum_{j \in \mathcal{N}_i} \boldsymbol{r}_{j/o}^B \left[\boldsymbol{r}_{j/o}^B \right]^\top \tag{4.51}$$

then solving the eigenvalue problem:

$$\lambda_i \hat{\boldsymbol{v}}_i^B = \boldsymbol{\Sigma}_{norm} \hat{\boldsymbol{v}}^B \tag{4.52}$$

three eignvalues $\lambda_{i,1}$, $\lambda_{i,2}$, $\lambda_{i,3}$, and three eigenvectors $\hat{\boldsymbol{v}}_{i,1}^B$, $\hat{\boldsymbol{v}}_{i,2}^B$, $\hat{\boldsymbol{v}}_{i,3}^B$ can be obtained. If the smallest of the eigenvalues is considered, such that $\lambda_{i,3} \leq \lambda_{i,2} \leq \lambda_{i,1}$, then the normal for each point mass will be:

$$\hat{\boldsymbol{n}}_i^B = \begin{cases} \hat{\boldsymbol{v}}_{i,3}^B & \text{if } \left(\hat{\boldsymbol{v}}_{i,3}^B \cdot \hat{\boldsymbol{z}}^B \right) > 0 \\ -\hat{\boldsymbol{v}}_{i,3}^B & \text{if } \left(\hat{\boldsymbol{v}}_{i,3}^B \cdot \hat{\boldsymbol{z}}^B \right) < 0 \end{cases} \quad \forall i \in \mathcal{S} \tag{4.53}$$

which can easily be rotated into the inertial frame:

$$\hat{\boldsymbol{n}}_i^I = \mathbf{C}^{I/B}\hat{\boldsymbol{n}}_i^B \tag{4.54}$$

The deflection angle $\delta\psi_i$ at each point on the surface $\mathcal{S}$ can be found by comparing the normal at that point $\hat{\boldsymbol{n}}_i^B$ with the direction of the body z-axis $\hat{\boldsymbol{z}}^B$:

$$\delta\psi_i = \cos^{-1}\left(\begin{bmatrix}0 & 0 & 1\end{bmatrix}\hat{\boldsymbol{n}}_i^B\right) \;\forall i \in \mathcal{S} \tag{4.55}$$

4.1.4 | Linearisation of Structural Dynamics

To inform the numerical integration stability regions defined previously in Section 3.2, and allow for modal analysis of the LSS structure, linearisation of the model must be performed. The structural dynamics of the model can be represented in LTI form:

$$\mathbf{M}\ddot{\boldsymbol{\delta r}}_{i/B}^B(t) + \mathbf{D}\dot{\boldsymbol{\delta r}}_{i/B}^B(t) + \mathbf{K}\boldsymbol{\delta r}_{i/B}^B(t) = 0 \tag{4.56}$$

where $\boldsymbol{\delta r}_{i/B}^B$ is the displacement of mass i from its natural undeformed location ($\boldsymbol{\delta r}_{i/B}^B = \boldsymbol{r}_{i/B}^B - \boldsymbol{r}_{i/B}^0$). Ignoring damping and taking the Laplace transform:

$$\mathbf{M}s^2\boldsymbol{\delta r}_{i/B}^B(s) + \mathbf{K}\boldsymbol{\delta r}_{i/B}^B(s) = 0 \tag{4.57}$$

where the mass matrix $\mathbf{M} = \text{diag}([m_1, m_2, \cdots, m_N])$ and the stiffness matrix can be formed:

$$\mathbf{K} = \begin{bmatrix}\mathbf{K}_{1,1} & \mathbf{K}_{1,2} & \cdots & \mathbf{K}_{1,N} \\ \mathbf{K}_{2,1} & \mathbf{K}_{2,2} & \cdots & \mathbf{K}_{1,N} \\ \vdots & & \ddots & \\ \mathbf{K}_{N,1} & \mathbf{K}_{N,2} & \cdots & \mathbf{K}_{N,N}\end{bmatrix} \tag{4.58}$$

which include the local 3×3 stiffness matrices along the diagonals:

$$\mathbf{K}_{i,i} = \sum_{j=\mathcal{C}_i} k_{ij}\,\hat{\boldsymbol{r}}_{j/i}\hat{\boldsymbol{r}}_{j/i}^\top \tag{4.59}$$

and the 3×3 stiffness matrices on the off-diagonals:

$$\mathbf{K}_{i,j} = \begin{cases} -k_{ij}\,\hat{\boldsymbol{r}}_{j/i}\hat{\boldsymbol{r}}_{j/i}^\top & \text{for} j \in \mathcal{C}_i \\ 0 & \text{otherwise}\end{cases} \tag{4.60}$$

where $\mathcal{C}_i$ is the set of mass elements connected to i, and $\hat{\boldsymbol{r}}_{j/i}$ is the unit vector of mass j relative to mass i. Letting $s^2 = \lambda$, and rearranging gives the eigenvalue problem:

$$\lambda \delta \boldsymbol{r}^B_{i/B}(s) = -\mathbf{M}^{-1}\mathbf{K}\delta \boldsymbol{r}^B_{i/B}(s) \tag{4.61}$$

$$\left(\lambda \mathbf{I} + \mathbf{M}^{-1}\mathbf{K}\right) \delta \boldsymbol{r}^B_{i/B}(s) = 0 \tag{4.62}$$

where λ is the eigenvalue with corresponding eigenvector $\delta \boldsymbol{r}^B_{i/B}(s)$. The trivial solution where $\delta \boldsymbol{r}^B_{i/B}(s) = 0$, which would correspond to no dynamic response, can be ignored, and each eigenvalue λ_i can be solved. There will be $3N$ roots since there are three degrees-of-freedom (DOF) for each point mass. These roots on the complex plane can then be found:

$$s_i = \sqrt{\lambda_i} \tag{4.63}$$

with the natural frequency given by the magnitude of the complex root:

$$\omega_i = ||s_i|| \tag{4.64}$$

A suitable maximum integration step size bound Δt_{max} to ensure the stability of the RK4 integration can then be found by first considering the period of the fastest dynamics:

$$T_{min} = \frac{2\pi}{\max_i(\omega_i)} \quad \text{for } i \in [1, 3N] \tag{4.65}$$

with the appropriate maximum integration step size chosen to ensure at least 10 integration steps within the period of fastest dynamics:

$$\Delta t_{max} \approx \frac{T_{min}}{10} \tag{4.66}$$

The eigenvalues λ_i, can also be substituted back into Eq. (4.62) to obtain the corresponding modal shapes of $\delta \boldsymbol{r}^B_{i/B}$.

Verification of Stiffness

The overall stiffness of the LSS has been modelled through linear springs with spring constants as calculated using Eq. (4.8). This will now be verified through a static test on the model by fixing one end of the structure as a cantilever and determining the maximum deflection under a distributed load, as seen in Fig. 4.12. The linearised dynamics, ignoring damping, with an applied load, can be given:

$$\mathbf{M}\ddot{\delta \boldsymbol{r}}^B_{i/B}(t) + \tilde{\mathbf{K}}\delta \boldsymbol{r}^B_{i/B}(t) = \boldsymbol{f}^B_{load} \tag{4.67}$$

where $\tilde{\mathbf{K}}$ is a modified stiffness matrix to ensure that the point masses on the cantilever end are constrained and fixed:

$$\tilde{\mathbf{K}}_{i,j} = \begin{cases} 0 & \text{for } i \in \text{cantilever} \\ \mathbf{K}_{i,j} & \text{otherwise} \end{cases} \tag{4.68}$$

and $\boldsymbol{f}^B_{load}$ is the series of load forces applied to each point mass:

$$\boldsymbol{f}^B_{load} = \begin{bmatrix} \boldsymbol{f}^B_{1,load} \\ \boldsymbol{f}^B_{2,load} \\ \vdots \\ \boldsymbol{f}^B_{N,load} \end{bmatrix} \tag{4.69}$$

Considering the deflection of the LSS about one of its longitudinal axes as being equivalent to the deflection of an Euler-Bernoulli beam, the maximum deflection under a distributed load p_z in the $\hat{\boldsymbol{z}}^B$ direction would be [186, p. 492]:

$$(\delta z)_{max} = -\frac{p_z L^4_{x,y}}{8(EI)_{tot}} \tag{4.70}$$

where $(\delta z)_{max}$ is the maximum deflection, and the distributed load p_z has units of force over length. The model can therefore be verified by applying load forces $\boldsymbol{f}^B_{i,load}$ to each of the point masses, as seen in Fig. 4.12, equivalent to the load p_z:

$$\boldsymbol{f}^B_{i,load} = \begin{cases} \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^\top & \text{for } i \in \text{cantilever} \\ \begin{bmatrix} 0 & 0 & -\dfrac{(N_{x,y}-1)p_z L_{x,y}}{N^2_{x,y} N_z} \end{bmatrix}^\top & \text{otherwise} \end{cases} \tag{4.71}$$

The linear model (Eq. (4.67)) can then be simulated forward in time until a steady state is reached, and the maximum deflection can be compared with an Euler-Bernoulli beam as given in Eq. (4.70).

Results of the maximum deflections for LSSs of various sizes, but with a constant distance between each point mass $l = 5$ m, can be seen in Table 4.1. As the number of masses along the longitudinal direction increases, the relative deflection approaches 1. With 10 lumped masses in the longitudinal direction (N_x), the discrepancy with Euler-Bernoulli beam theory is 13.7%, decreasing to just 4% when the number of masses is doubled. The modelled stiffness of the LSS structure has therefore been verified.

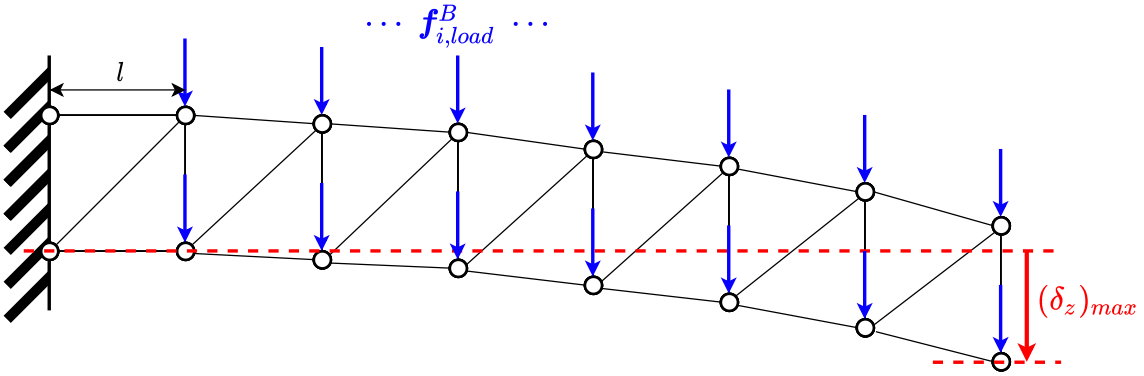


Figure 4.12: Static deflection of the model with distributed load (side view)

Table 4.1: Static deflection verification

$(EI)_{tot}$[Nm^2]	N_x	N_y	N_z	L [m]	$(\delta z)_{max}$ [m]		Relative $(\delta z)_{max}$
					Theory Eq. (4.70)	Model	
1×10^2	10	10	2	45	-0.51	-0.58	113.7%
1×10^2	20	20	2	95	-10.18	-10.59	104.0%
1×10^2	30	30	2	145	-55.26	-56.43	102.1%
1×10^3	10	10	2	45	-0.05	-0.058	113.7%
1×10^3	20	20	2	95	-1.02	-1.059	104.0%
1×10^3	30	30	2	145	-5.53	-5.64	102.1%
1×10^6	10	10	2	45	-5.13×10^{-5}	-5.83×10^{-5}	113.7%
1×10^6	20	20	2	95	-0.0010	-0.0011	104.0%
1×10^6	30	30	2	145	-0.0055	-0.0056	102.1%

4.1.5 | Actuator Mass Scaling Laws

An important point to consider is that the mass of each actuator will have a significant impact on the overall dynamic response of the LSS. To produce an equitable comparison, it is important to ensure that the mass of each type of actuator is representative of what could realistically be expected. Additionally, each actuator type will have a maximum size of LSS that can successfully perform the prescribed manoeuvre. This section uses the same methodology as Viale et al. [82], which derived actuator scaling laws specifically for large solar reflectors. Results provided here are similar with slight variations to reflect the use of the low-jerk polynomial rest-to-rest slew manoeuvres used previously in Chapters 2 and 3.

Consider first the maximum acceleration $\ddot{\theta}_{max}$ of an attitude slew manoeuvre about the longitudinal axes of the LSS $I_{xx,yy}$, giving the maximum required torque τ_{max}:

$$\tau_{max} = I_{xx,yy}\ddot{\theta}_{max} \tag{4.72}$$

with the maximum acceleration and velocity of the polynomial slew manoeuvre defined in Appendix B.4 (Eq. (B.22)) found to be:

$$\ddot{\theta}_{max} = \frac{85\sqrt{5}}{25}\frac{\theta_f}{t_f^2} \tag{4.73a}$$

$$\dot{\theta}_{max} = \frac{35\theta_f}{16t_f} \tag{4.73b}$$

and the inertia about the longitudinal axes of a square LSS is defined as:

$$I_{xx,yy} = \frac{1}{12}\rho_{lss}L^4 + I_a \tag{4.74}$$

where ρ_{lss} is the area mass density of the LSS, L is the side length of the LSS, and I_a is the contribution of inertia due to the actuator that is chosen. The maximum size of LSS that can be actuated by each actuator, and the mass of each actuator can therefore be defined by the duration of the manoeuvre t_f, the slew angle θ_f, and the areal mass density of the LSS structure ρ_{lss}.

Moving Mass (MM)

The inertia contribution of a MM actuator is [82]:

$$I_a = \rho_{lss}L^2r_{com}^2 + m_a\left(r_a - r_{com}\right)^2 \tag{4.75}$$

where the first term takes into account the parallel axis theorem for the movement of the centre-of-mass, and the second term is the inertia of the MM actuator. The centre of mass position r_{com} can be defined as:

$$r_{com} = \frac{m_a r_a}{m_a + \rho_{lss}L^2} \tag{4.76}$$

where m_a is the mass of the actuator and r_a is the position of the centre of mass. The maximum torque that can be produced by the MM actuator is [82]:

$$\tau_{max} = 2p_{sol}L^2r_{com}\cos^2(\phi_{sun}) \tag{4.77}$$

where ϕ_{sun} is the angle between the normal of the LSS surface and the sun. Substituting Eqs. (4.74) to (4.77) into Eq. (4.72) then assuming $r_a = L/4$ and ϕ_{sun} =45 deg we can solve for the mass of the MM actuator:

$$m_a = \frac{28\sqrt{5}L^3\rho_{lss}^2\theta_f}{25p_{sol}t_f^2 - 49\sqrt{5}L\rho_{lss}\theta_f} \tag{4.78}$$

The maximum possible side length can also be determined through consideration of the denominator of Eq. (4.78):

$$L_{max} = \frac{5\sqrt{5}p_{sol}t_f^2}{49\rho_{lss}\theta_f} \tag{4.79}$$

Reflectivity Control Devices (RCD)

It will be assumed that the variable reflectivity adds negligible mass to the LSS structure ($I_a = 0$), however, due to there being a limit on the magnitude of SRP that can be utilised to produce torque, there still is a constraint on the maximum size of the LSS. Assuming that the full area of the LSS acts as a variable reflector, the maximum torque that can be produced is given by [82]:

$$\tau_{max} = \frac{p_{sol}L^3\cos^2(\phi_{sun})}{8} \tag{4.80}$$

Assuming ϕ_{sun} =45 deg Eqs. (4.74) and (4.80) can be substituted into Eq. (4.72) which can be used to find a maximum side length:

$$L_{max} = \frac{5\sqrt{5}p_{sol}t_f^2}{112\rho_{lss}\theta_f} \tag{4.81}$$

Reaction-Wheel Sets (RWS)

Consider the torque at any point during the attitude slew manoeuvre being provided by one reaction wheel:

$$|\tau_w| = I_w|\dot{\omega}_w| \tag{4.82}$$

where $\dot{\omega}_w$ is the angular acceleration of the wheel. Considering Eq. (4.72) for any point in the slew manoeuvre and not necessarily at the maximum acceleration, and also Eq. (4.82), the integral can be formed taking into account that $I_a = I_w$ [82]:

$$\int I_w|\dot{\omega}_w|dt = \int \left(\frac{1}{12}\rho_{lss}L^4 + I_w\right)|\ddot{\theta}|dt \tag{4.83}$$

which once integrated and assuming $\omega_{w,max} \gg \dot{\theta}_{max}$ the inertia of the wheel can be expressed in terms of the maximum wheel speed and maximum angular rate of the slew:

$$I_w \approx \frac{\rho_{lss}L^4\dot{\theta}_{max}}{12\omega_{w,max}} \tag{4.84}$$

Assuming that the wheel is a uniform disk ($I_w = \frac{1}{2} m_w R_w^2$):

$$m_w = \frac{\rho_{lss} L^4 \dot{\theta}_{max}}{6 \omega_{w,max} R_w^2} \tag{4.85}$$

As can be seen, the wheel can increase in size to reduce the maximum speed of the wheel. This can be further constrained by considering the material strength of the wheel with the maximum stress of a wheel rotating about its centre $\sigma_{w,max}$ given as [82]:

$$\sigma_{w,max} = \frac{3 + \nu_w}{8} \rho_w \omega_w^2 R_w^2 \tag{4.86}$$

where ν_w is the Poisson's ratio of the wheel material and ρ_w is the volume density ($\mathrm{kg/m^3}$) of the wheel material. Rearranging Eq. (4.86) for ω_w and substituting in place of $\omega_{w,max}$ in Eq. (4.85) and substituting in Eq. (4.73b) gives:

$$m_w = \frac{1}{6} \sqrt{\frac{3 + \nu_w}{8}} \sqrt{\frac{\rho_w}{\sigma_{w,max}}} \frac{35 \rho_{lss} L^4 \theta_f}{16 t_f R_w} \tag{4.87}$$

To ensure 3-axis controllability of an LSS a 3-wheel set will be used, giving the total actuator mass:

$$m_a = 3 m_w \tag{4.88}$$

The maximum size of the LSS can be determined by considering the maximum available power to the LSS [82]:

$$P_{max} = \eta_e \eta_A L^2 P_0 cos(\phi_{sun}) \tag{4.89}$$

where η_e is the efficiency of the solar panels used to collect energy, η_A is the fraction of the LSS surface set aside for power production, and $P_0 = 1327 \mathrm{W/m^2}$ is the solar constant at 1AU. The power consumed by the reaction wheel is:

$$P_{w,max} = \tau_{w,max} \omega_w = \left(\frac{1}{12} \rho_{lss} L^4 + I_w \right) \ddot{\theta}_{max} \omega_w \tag{4.90}$$

Assuming that the inertia of the LSS is much larger than the reaction wheel and substituting in a solution from Eq. (4.86) for ω_w:

$$L_{max} = t_f \sqrt{\frac{10\sqrt{10}}{7} \sqrt{\frac{3 + \nu_w}{8}} \sqrt{\frac{\rho_w}{\sigma_{w,max}}} \frac{R_w \eta_A \eta_e P_0 cos(\phi_{sun})}{\rho_{lss} \theta_f}} \tag{4.91}$$

Table 4.2: Model parameters and manoeuvre requirements for LSS

Property	Variable	Value	Units	Note
LSS side lengths	L	10 105	m	
LSS truss width/cubic unit sidelength	l	2 5	m	
LSS characteristic total bending stiffness	$(EI)_{tot}$	1 100	Nm^2	
LSS damping constant of proportionality	γ	0.1	s^{-1}	Improves numerical stability
LSS structure area mass density	ρ_{lss}	10	$\mathrm{g/m}^2$	Robb *et al.* [66]
Orbital altitude	h	1000	km	Representative of solar reflector missions [33] and an altitude where SRP dominates over aerodynamic drag
Solar constant	P_0	1327	$\mathrm{W/m}^2$	At 1AU [33]
Solar radiation pressure	p_{sol}	4.42	µPa	P_0/c where c is the speed of light
Magnetic field strength	b_{mag}	25.9	µT	IGRF-13 sampled for 1st June 2024, given altitude and zero latitude and longitude
Reaction wheel radius	R_w	0.5	m	Chosen to fit easily within truss width
Reaction wheel density	ρ_w	7800	$\mathrm{kg/m}^3$	4340 steel. [82]
Max tensile strength of reaction wheel	$\sigma_{w,max}$	180	MPa	10% the tensile strength of 4340 steel [82]
Poissons ratio of reaction wheel	ν_w	0.3		Viale and McInnes [82]
Current loop conductor density	ρ_{cl}	8960	$\mathrm{kg/m}^3$	Rumble [187]
Current loop conductor resistivity	ζ_{cl}	16.78	nΩ m	Rumble [187]
Slew angle	θ_f	90	deg	Replicating the periodic slew manoeuvres executed by LighSail-2 [24]
Slew duration	t_f	947.67	s	15% of orbital period

Embedded Magnetorquers (MAG)

The mass of axially mounted MAGs within the struts of the LSS structure, as seen in Fig. 4.6, can be found by considering a typical mass ratio of 100 Am^2/kg [66]. The maximum required magnetic dipole needed is given by:

$$||\boldsymbol{m}_{mag,max}|| = \frac{\tau_{max}}{||\boldsymbol{b}_{mag}|| \sin(\phi_{mag})} \tag{4.92}$$

where ϕ_{mag} is the angle between the magnetic field and the magnetic dipole. Considering an even distribution of magnetorquers throughout the structure, and magnetorquers aligned in all three axes, the inertia contribution can be given as:

$$I_a = 3\left(\frac{1}{12} m_a L^2\right) = \frac{1}{4} m_a L^2 \tag{4.93}$$

Then, considering the mass ratio mentioned previously ($||\boldsymbol{m}_{mag,max}|| = 100 m_a$), and assuming $\phi_{mag} = 45$ deg, the mass of the magnetorquers can be given as:

$$m_a = \frac{7\sqrt{5} L^4 \rho_{lss} \theta_f}{1250\sqrt{2} b_{mag} t_f^2 - 21\sqrt{5} L^2 \theta_f} \tag{4.94}$$

where $b_{mag} = ||\boldsymbol{b}_{mag}||$. Again, the denominator can be solved to find the limit of the side length for this actuator:

$$L_{max} \approx 1.34 t_f \sqrt{\frac{b_{mag}}{\theta_f}} \tag{4.95}$$

Current Loop Magnetorquers (CL)

Assuming a CL running around the perimeter of the LSS, seen in Fig. 4.7, the maximum magnetic torque can be given:

$$\tau_{max} = ||\boldsymbol{m}_{mag,max}||\ ||\boldsymbol{b}_{mag}|| \sin(\phi_{mag}) = N_{cl} i_a b_{mag} L^2 \sin(\phi_{mag}) \tag{4.96}$$

where N_{cl} is the number of loops the CL consists of, i_a is the current running through each CL, L^2 is the area enclosed by the CL, and ϕ_{mag} is the angle between the magnetic field and the magnetic dipole. Assuming the mass of the CL magnetorquer is evenly distributed around the LSS:

$$I_a = \frac{1}{12} \rho_{lss} L^4 + \frac{1}{12} m_a L^2 \tag{4.97}$$

and equating the maximum magnetic torque to the required torque (Eqs. (4.72) and (4.96)), the term $N_{cl}i_a$ can be expressed [82]:

$$N_{cl}i_a = \frac{\ddot{\theta}_{max}\left(\rho_{lss}L^2 + m_a\right)}{12b_{mag}\sin(\phi_{mag})} \tag{4.98}$$

The power in the CL magnetorquer can be given:

$$P_{mag} = R_{cl}i_{max}^2 \tag{4.99}$$

where R_{cl} is the resistance in the CL is defined by its physical properties:

$$R_{cl} = \zeta_{cl}\frac{4N_{cl}L}{A_{cl}} \tag{4.100}$$

where ζ_{cl} is the resistivity of the CL conductor material, A_{cl} is the cross-sectional area of the conductor, and $4L$ is the length of one loop. This can be redefined in terms of the mass and density of the conductor by considering the dimensions of the conductor:

$$A_{cl} = \frac{m_a}{4N_{cl}L\rho_{cl}} \tag{4.101}$$

which when combined with Eq. (4.99) gives:

$$P_{mag} = \zeta_{cl}\frac{16L^2\rho_{cl}}{m_a}(N_{cl}i_a)^2 \tag{4.102}$$

Solving Eq. (4.102) for the $N_{cl}i_a$ term, substituting into Eq. (4.98), and solving for minimum m_a gives:

$$\begin{aligned} m_a = &\frac{2L^4\rho_{lss}\rho_{cl}\ddot{\theta}_{max}^2\zeta_{cl} - 9b_{mag}^2P_{mag}\sin^2(\phi_{mag})}{2L^2\rho_{cl}\ddot{\theta}_{max}^2\zeta_{cl}} \\ &- \frac{3b_{mag}\sin(\phi_{mag})\sqrt{9b_{mag}^2\sin^2(\phi_{mag})P_{mag}^2 - 4L^4\rho_{cl}\rho_{lss}\ddot{\theta}_{max}\zeta_{cl}}}{2L^2\rho_{cl}\ddot{\theta}_{max}^2\zeta_{cl}} \end{aligned} \tag{4.103}$$

The minimum power required can be found by considering that the term under the square root must be positive for the solution to the mass to remain real:

$$P_{mag,min} = \frac{4L^4\rho_{cl}\rho_{lss}\ddot{\theta}_{max}^2\zeta_{cl}}{9b_{mag}^2\sin^2(\phi_{mag})} \tag{4.104}$$

which when substituted back into Eq. (4.103) gives the mass required for minimum power:

$$m_a^{\text{min power}} = \arg\min_{m_a}(P_{mag}) = \rho_{lss} L^2 = m_{lss} \tag{4.105}$$

Therefore, to minimise power consumption, the mass of the CL will be equal to the structural mass of the LSS, doubling its overall mass [82]. It is also interesting to note that this is irrespective of the maximum angular acceleration of the manoeuvre.

The mass of the CL can be reduced by increasing the power used by the CL. The maximum power in Eq. (4.89) can be used as an upper bound, which can be substituted into Eq. (4.103) to find a limit to the size of the LSS:

$$L_{max} = \frac{3}{2} \frac{b_{mag} \sin(\phi_{mag}) \sqrt{\eta_A \eta_e P_0 \rho_{cl} \rho_{lss} \zeta_{cl} \cos(\phi_{sun})}}{\rho_{cl} \rho_{lss} \ddot{\theta}_{max} \zeta_{cl}} \tag{4.106}$$

The parameters of the modular model defined here and used throughout this chapter are given in Table 4.2 for reference.

A modular mathematical model of LSSs controlled using a variety of different actuator types has now been developed, including proper scaling to ensure fair comparisons of dynamic performance. An InvSim approach to inform the design process of LSSs will be developed in the following section of this chapter.

4.2 | Inverse Simulation for Satellite Design

Whilst this chapter focuses on the use of InvSim for the design of LSSs, namely the choice and configuration of attitude control actuators, the approach laid out in this section could in general be utilised in the design of satellites of any form, or any dynamic system where prescibed output trajectories must be followed. An illustration of the proposed use of InvSim within a system design process can be seen in Fig. 4.13. The user defines the mission requirements, which would then, in turn, inform the manoeuvres required to achieve those requirements, attitude manoeuvres in this case. For example, LightSail-2's mission was to demonstrate solar sailing technologies in Earth orbit [24] and slow the orbital decay of the spacecraft. This can be achieved by reorienting the sail by 90 deg twice per orbit, thus defining a required attitude manoeuvre that the system must be capable of performing. The user will then, through their engineering judgment, design a suitable satellite platform, which in the case of this chapter will be assumed to involve the use of an LSS. If a mathematical model of this design can be obtained, such as that detailed previously in Section 4.1, then InvSim can provide a solution to the full system dynamics for a required attitude manoeuvre. This gives

the user quantitative performance measures of the current design, informing its next iteration. This process could then be repeated iteratively until a design that meets the mission requirements, and gives good performance.

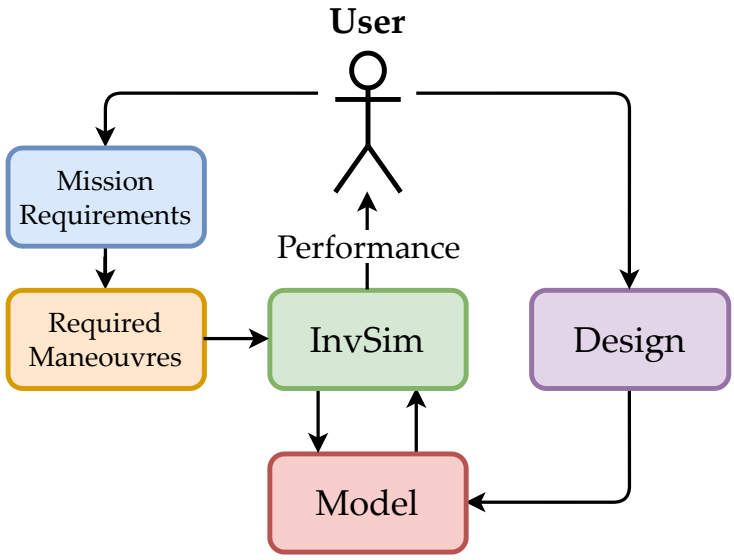


Figure 4.13: System design process with Inverse Simulation

4.2.1 | Inverse Simulation of Overactuated Systems

As discussed, this chapter is specifically concerned with investigating the performance of a variety of types and possible combinations of attitude control actuators, which may outperform any singular type of actuator. With a large number of actuators, and thus inputs to the system, the InvSim problem becomes overactuated, meaning that there could be more than one unique solution to the set of simultaneous equations defining the InvSim problem; this is referred to as the *redundant* InvSim problem. In Section 2.1 it was shown that the InvSim could utilise an MP pseudoinverse within the NR numerical method to solve redundant systems, converging upon the solution closest to the initial estimate $\boldsymbol{u}_0$. This approach was shown in Chapters 2 and 3 to be successful in solving the inverse dynamics of satellites with four reaction wheels.

Therefore, through careful selection of $\boldsymbol{u}_0$ a specific solution to the InvSim problem can be converged upon. Commonly, $\boldsymbol{u}_0$ is only chosen for the first timestep ($k = 0$), with the solution from the previous timestep $[(k-1)T]$ used as the initial estimate for each subsequent timestep $[kT]$. It is fair to assume that if the InvSim is given a smooth and continuous output trajectory to track, then the input will also be smooth and continuous, with the solution of $\boldsymbol{u}$ at each timestep being close to the previous. However, there can be issues associated with this approach, which will be discussed later in this chapter.

In some cases, choosing the initial estimate $\boldsymbol{u}_0$ to converge upon a specific solution may be non-trivial, and not obvious to the user. Additionally, there may be some constraints on the inputs that need to be considered, as some solutions may violate these constraints, while other possible solutions do not. It may therefore be beneficial to replace the set of simultaneous equations previously used to define the InvSim problem (Eq. (2.6)) with a constrained optimisation problem [188]:

$$\min_{\boldsymbol{u}[kT]} \quad s(\boldsymbol{u}[kT]) \tag{4.107a}$$

$$\text{subject to} \quad c_i(\boldsymbol{u}[kT]) = 0, \quad i \in \mathcal{E} \tag{4.107b}$$

$$c_i(\boldsymbol{u}[kT]) \leq 0, \quad i \in \mathcal{I} \tag{4.107c}$$

where $s(\boldsymbol{u}[kT])$ is a *cost function* that can be used to ensure convergence to a preferable input solution $\boldsymbol{u}[kT]$. The set of simultaneous equations (Eq. (2.6)) used to define the InvSim problem are implemented as equality constraints on the outputs of the system (Eq. (4.107b)):

$$c_i(\boldsymbol{u}[kT]) = \delta \overset{(\nu)}{\boldsymbol{y}}_i(\boldsymbol{u}[kT]), \quad i \in \mathcal{E} \tag{4.108}$$

with actuator limits/saturations considered as *inequality* constraints (Eq. (4.107c)). SQP will be utilised in this chapter to solve the redundant InvSim as a series of constrained optimisation problems [188]. Details of the SQP numerical method are detailed in Appendix D.2. Fig. 4.14 illustrates how SQP can be implemented and replace the usual NR method within the InvSim process with the full modified algorithm shown in Algorithm 3.

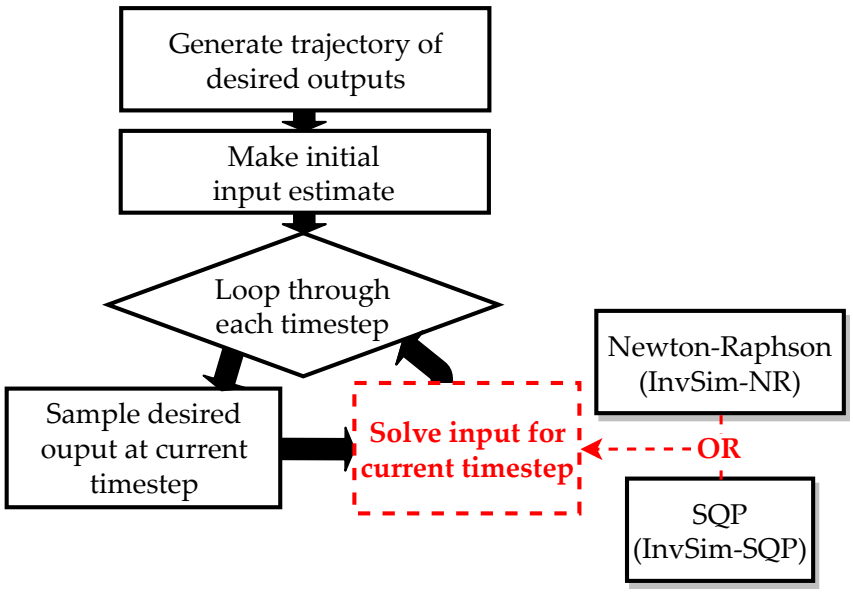


Figure 4.14: Inverse simulation using NR and SQP solvers

Algorithm 3: Integral Inverse Simulation with SQP solver (InvSim-SQP)

Generate timeseries of desired output/output derivative states $\dot{\boldsymbol{y}}_d^{(\nu)}(t)$;
Define an initial input estimate for the first step $\boldsymbol{u}_0$;
for $k = 1 \rightarrow$ `length`$(\dot{\boldsymbol{y}}_d^{(\nu)}[kT])$ **do**
    Sample desired output/output derivative $\dot{\boldsymbol{y}}_d^{(\nu)}[(k+1)T]$;
    Initialise input estimate $\boldsymbol{u}_0[kT] = \begin{cases} \boldsymbol{u}[(k-1)T] & \text{for } k > 1 \\ \boldsymbol{u}_0 & \text{for } k = 1 \end{cases}$;
    Form constrained optimisation problem (Eqs. (4.107) and (4.108));
    Find constrained optimal input $\boldsymbol{u}^*[kT]$ problem using SQP (Appendix D.2);
    Save input solution $\boldsymbol{u}[kT] \leftarrow \boldsymbol{u}^*[kT]$;
end

Two distinct InvSim algorithms can therefore be defined, one utilising a NR solver (InvSim-NR), and the other using an SQP solver instead (InvSim-SQP). The hyperparameters for each InvSim method used throughout this chapter are given in Table 4.3, including the minimum perturbation size Δu_{min} for each actuator used by InvSim-NR. Note that the *MATLAB* implementation of SQP `fmincon()` will terminate if the constraint tolerance is satisfied *and* either the cost, step, or first-order optimality tolerances are also met.

4.2.2 | Reaction Wheel-Set (RWS) Example

An example solution of InvSim-NR solving the inverse dynamics of an LSS with one RWS actuator performing a 90 deg slew manoeuvre about the body $\hat{\boldsymbol{x}}^B$ axis is detailed next. Considering the case of single-axis pointing requirements, as would be the case for missions such as LightSail-2 [24], the desired slew manoeuvre can be defined using two Euler angles only ($\dot{\boldsymbol{y}}_d^{(\nu)} = [\dot{\theta}_x^{(\nu)} \; \dot{\theta}_y^{(\nu)}]_d^\top$) as covered in Sections 2.3 and 3.4.

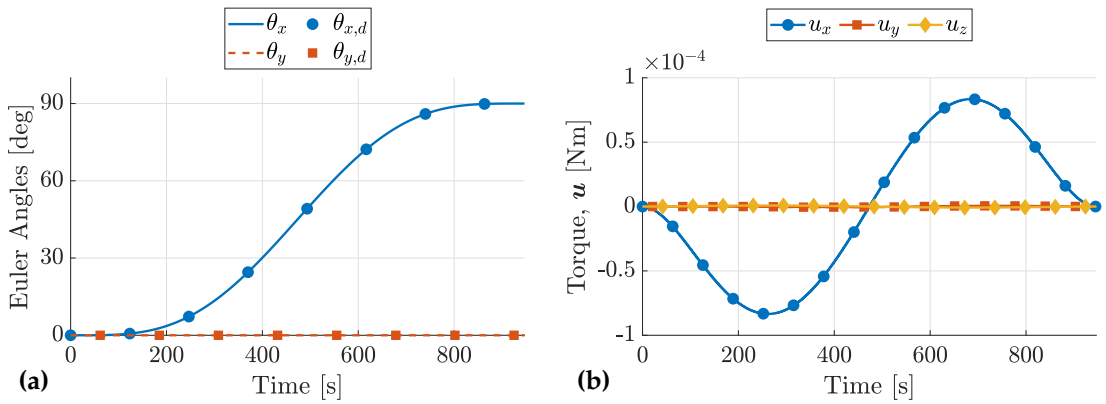


Figure 4.15: InvSim example solution: single centralised RWS, $L = 10$ m, $(EI)_{tot} = 1$ Nm^2. (a) outputs, (b) inputs.

Table 4.3: Large flexible space structures: InvSim hyperparameters

Hyperparameter		**Value**
Discretisation timestep, T		$10{\cdot}\Delta t$
Desired output, $\dot{\boldsymbol{y}}_d^{(\nu)}$		$[\dot{\theta}_x^{(\nu)}\ \dot{\theta}_y^{(\nu)}]_d^\top$
Desired output order, ν		1 (velocity)
Maximum NR/SQP iterations, n_{max}		100
InvSim-NR		
NR tolerance, $\delta\dot{y}_{tol}^{(\nu)}$		$1{\times}10^{-10}$
Minimum finite difference perturbation, Δu_{min}:	MM	$0.1{\cdot}L$ m
	RCD	0.25
	RWS	1 Nm
	MAG	10 Am^2
	CL	1 A
InvSim-SQP		
Constraint tolerance, $\delta\dot{y}_{tol}^{(\nu)}$		$1{\times}10^{-10}$
Cost tolerance, $s(\boldsymbol{u}_n)_{tol}$		$1{\times}10^{-10}$
Step tolerance, $\|\|\boldsymbol{u}_{n+1}-\boldsymbol{u}_n\|\|_{tol}$		$1{\times}10^{-10}$
First-order optimality, $\|\|\nabla_{\boldsymbol{u}}\mathcal{L}(\boldsymbol{u}_n,\boldsymbol{\lambda}_n,\boldsymbol{\sigma}_n)\|\|_{tol}$		$1{\times}10^{-10}$

The traditional NR solver can be utilised in this example as the InvSim problem is not redundant, and there is no need for constraints to be applied to the input torques since proper actuator scaling has already been performed (Section 4.1.5). The specific model parameters used can be found in Table 4.2, which for the purposes herein can be considered arbitrary, but will be physically reasoned later in Section 4.3. The InvSim-NR results for a small 10×10 m LSS with a total overall stiffness of $(EI)_{tot} = 1$ Nm^2 can therefore be found in Fig. 4.15. It can be seen that the 90-deg slew manoeuvre is successfully tracked (Fig. 4.15a), and a smooth continuous input torque signal is produced (Fig. 4.15b).

It is therefore clear that the basic integral InvSim-NR algorithm described in Section 2.1 and utilised throughout Chapters 2 and 3 is still perfectly capable of providing good results for more complex mathematical models, such as the one presented in this chapter (Section 4.1). It will be shown next that this is not the case for all actuator types, with particular numerical issues encountered when solving the dynamics of LSSs with magnetic torques, an issue previously discussed in Section 2.2.3.

4.2.3 | Numerical Issues Associated with Magnetic Torques (MAG and CL)

Section 2.2.3 previously detailed that the Jacobian calculated during the InvSim-NR method will always have a *null* space in the direction of the local magnetic field $\hat{\boldsymbol{b}}^B_{mag}$. The resultant Jacobian is therefore singular, does not have a direct inverse, and must therefore employ a MP pseudoinverse (Eq. (A.19)). It was shown that strict constraints on the attitude could result in no physical solution existing if torques were required outside the plane defined by $\hat{\boldsymbol{b}}^B_{mag}$, resulting in the InvSim-NR solution diverging to infinity. However, loosening the constraints was shown to allow for InvSim-NR to converge upon a physically realisable, stable input solution.

This behaviour can be shown to be present once again with the modular LSS model developed in this chapter, for both magnetic actuator types (MAG and CL). To demonstrate, the CL actuator will be employed, but similar results are found when using the MAG architecture. The minimum power/maximum mass CL configuration is assumed here, as this will minimise the surface area of the LSS dedicated to power generation, an important consideration for solar sailing and in-orbit solar reflecting applications. Model properties as detailed in Table 4.2 are again used.

Fig. 4.16 shows the InvSim-NR input solution (total magnetic dipole) in the case where both the $\hat{\boldsymbol{x}}^B$ and $\hat{\boldsymbol{y}}^B$ axes are constrained (dashed line), and when only the $\hat{\boldsymbol{x}}^B$ axis is constrained (solid line). As was seen in Section 2.2.3 when the attitude of the LSS is overconstrained, requiring some torque outwith the plane defined by $\hat{\boldsymbol{b}}^B_{mag}$, the input solution diverges to infinity and is therefore numerically unstable.

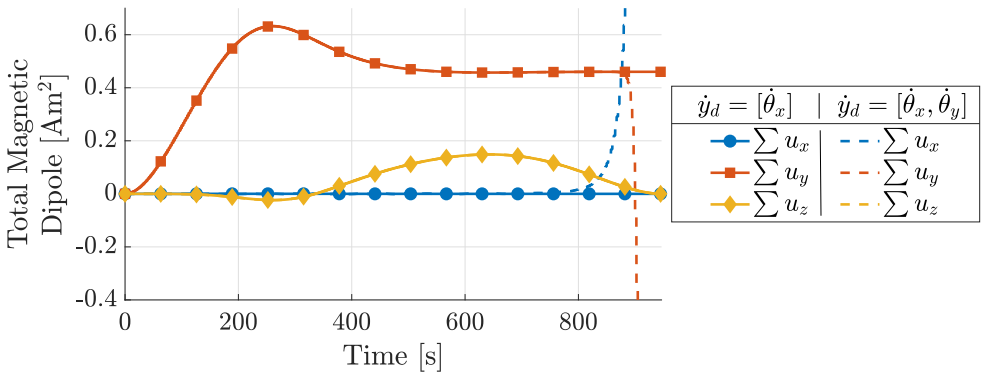


Figure 4.16: Current-loop (CL) actuator: Comparison of different axis constraints when using NR method (InvSim-NR)

This can be seen more clearly in Fig. 4.17, where the local magnetic field $\hat{\boldsymbol{b}}^B_{mag}$ is shown to rotate into the $\hat{\boldsymbol{y}}^B$ direction causing actuation about that axis to be progressively lost. Rotations about that axis are strictly constrained through $\dot{\boldsymbol{y}}^{(\nu)}_d$, causing exponential di-

vergence in the input solution as InvSim-NR attempts to correct any small divergences about the $\hat{\boldsymbol{y}}^B$ axis.

Loosening the constraints to $\dot{y}_d^{(\nu)} = \dot{\theta}_x^{(\nu)}$ ensures that a physically realisable solution can be converged upon, as is evident in Fig. 4.16 by the solid lines. Also, it is interesting to note that the output for the $\hat{\boldsymbol{y}}^B$ axis is still tracked, even without it being strictly constrained. This is due to the LSS effectively having no products of inertia, which would have induced additional unwanted body rates that could not be attenuated, as shown in Section 2.2.3. While it was shown in Section 2.2.3 that satellites with no products of inertia did not require loosening of the output constraints, in the case of the LSS it is still required to ensure stability. It can be deduced that numerical accuracy has a part to play, as while in practice, no additional unwanted body rates are induced, numerical errors can cause such body rates to appear, triggering instability.

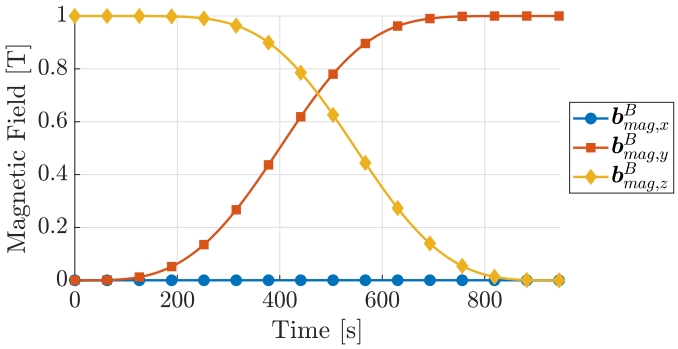


Figure 4.17: Magnetic field direction in body axes during 90-deg slew manoeuvre about the positive $\hat{\boldsymbol{x}}^B$ axis

While the single-axis constrained solution in Fig. 4.16 is stable, the steady state of the u_y input at the end of the manoeuvre is unexpectedly non-zero. This is similarly due to the loss of actuation about the $\hat{\boldsymbol{y}}^B$ axis seen in Fig. 4.17. As actuation about the $\hat{\boldsymbol{y}}^B$ axis is lost, the input u_y has less and less effect on the output, which is reflected by the associated column of the Jacobian used in the NR solver tending to zero. So far the InvSim-NR performed has used the input solution from the previous timestep $\boldsymbol{u}[(k-1)T]$ as the initial estimate $\boldsymbol{u}_0$ used to solve the current timestep $\boldsymbol{u}[kT]$. As elements of the Jacobian associated with u_y tend to zero, the NR method will stop applying any updates to u_y. This leads to u_y becoming stuck at a nonzero value (Fig. 4.16), and since deviation from that value does not affect the output, the InvSim-NR has no reason to modify it. This behaviour is not ideal as the nonzero steady-state magnetic dipole at the end of the manoeuvre is a waste of power and, therefore, not a realistic input solution that could be applied to a functional spacecraft.

As mentioned in Section 4.2.1, it is possible to set a specific initial estimate $\boldsymbol{u}_0$ instead of using the previous timestep's solution. This can allow for more control over the specific

InvSim-NR solution converged upon using the NR method, and will be employed here. Previously the initial estimate $\boldsymbol{u}_0$ was taken from the solution to the previous timestep ($\boldsymbol{u}_0 = \boldsymbol{u}[(k-1)T]$); instead, it is possible to converge upon a more suitable solution for magnetic actuator types by setting $\boldsymbol{u}_0 = 0$ at each and every timestep. A comparison between the two initial estimate approaches for the NR-based method (InvSim-NR) can be seen in Fig. 4.18a. The solution using zeroed initial estimates no longer has a steady state input at the end of the manoeuvre. The Jacobian column associated with u_y will still tend to zero, however, since the NR method starts with an estimate of $\boldsymbol{u}_0 = 0$ a more suitable solution of $u_y = 0$ can be converged upon.

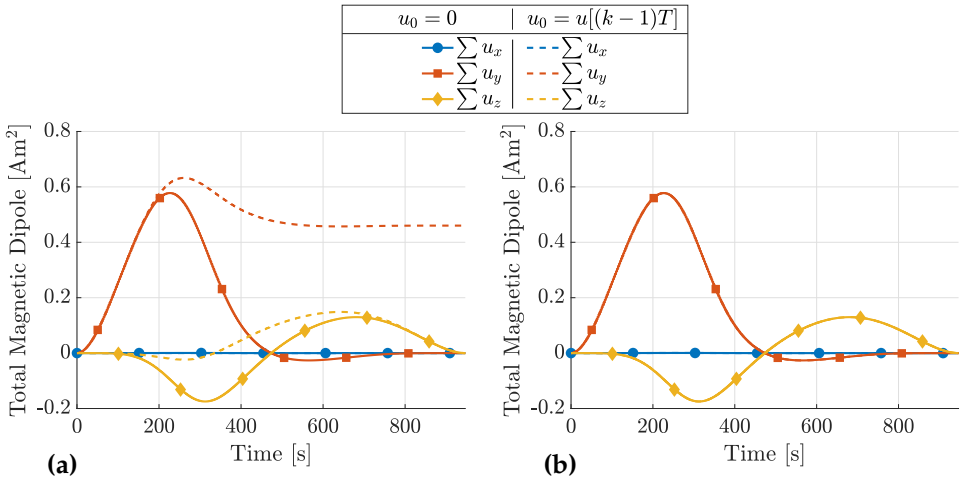


Figure 4.18: Comparison of effects of initial input estimate when using (a) InvSim-NR and (b) InvSim-SQP

Choosing an appropriate initial estimate for the NR solver may not always be trivial, despite achieving a suitable solution. Instead, the same solution found using NR with zeroed initial estimates can be converged upon using SQP and an appropriate cost function (InvSim-SQP). Choice of the cost function in many cases will be far more obvious than trying to select the initial estimate, and in many cases will be insensitive to the initial estimate chosen. A cost function $s(\cdot)$ to minimise the control effort during the slew manoeuvre is therefore defined:

$$s(\boldsymbol{u}[kT]) = \frac{1}{2}\boldsymbol{u}^\top[kT]\boldsymbol{u}[kT] \tag{4.109}$$

A comparison of using InvSim-SQP for different initial input estimates $\boldsymbol{u}_0$ is given in Fig. 4.18b. The solutions converged to are identical to those found when using InvSim-NR with zeroed initial estimates ($\boldsymbol{u}_0 = 0$). Additionally, it can be seen that the addition of the cost function (Eq. (4.109)) means that InvSim-SQP is insensitive to the initial estimate ($\boldsymbol{u}_0$) used.

This section primarily introduced the use of sequential-quadratic-programming (SQP) within the InvSim solution process for overactuated systems, with the effect of Jacobian null space, and the initial input estimate also explored. The benefits of the InvSim-SQP algorithm (Algorithm 3) within a system design framework will be illustrated in the next section of this chapter.

4.3 | An Investigation into Attitude Control Actuators for Large Flexible Space Structures

The process for utilising InvSim for the design of LSSs has now been established with the use of a suitable mathematical model, and InvSim methods (InvSim-NR and InvSim-SQP). This section will now present a full investigative case study to demonstrate the use of InvSim as a design tool, specifically comparing the performance of different attitude control actuator types and configurations. Of particular interest will be the deformation experienced by the structure while performing a prescribed manoeuvre; which is defined in Eq. (4.55), and can be averaged across the entire surface of the LSS using a root-mean-square (RMS):

$$\mathrm{RMS}(\delta\boldsymbol{\psi}) = \sqrt{\frac{\sum_{i\in\mathcal{S}} \delta\psi_i^2}{N_{\mathcal{S}}}} \tag{4.110}$$

where $N_{\mathcal{S}}$ is the number of point masses in the set $\mathcal{S}$ defining the top surface of the LSS.

The investigation in this section is therefore tasked to aid in the design of an LSS (of various sizes) which is capable of performing at least two 90 deg slew manoeuvres during each orbit. This is, as mentioned previously in Section 4.2, indicative of a spacecraft utilising solar sailing propulsion, such as LightSail-2 [24], which will require that its thrust vector be reorientated regularly to achieve a desired orbital trajectory. Again, as mentioned in Section 4.2.2, the pointing of a single axis ($\hat{\boldsymbol{z}}^B$) as would be required in this mission scenario, can be defined using two Euler angles ($\dot{\boldsymbol{y}}_d^{(\nu)} = [\dot{\theta}_x^{(\nu)}\ \dot{\theta}_y^{(\nu)}]_d^\top$). The LSS will be assumed to be in a circular orbit of altitude $h = 1,000$ km, which places it within a regime where SRP will dominate over aerodynamic drag, and is approximately representative of a solar reflector mission architecture [33]. These manoeuvre requirements and model parameters can be found in Table 4.2, including some additional reasoning for some of the values chosen.

4.3.1 Design of a Large Flexible Space Structure

Here, the baseline design for planar homogeneous LSSs is described. This will give the testbed upon which each actuator type and configuration can be tested. The maximum sizes of LSSs for each actuator type must be determined based on the actuator scaling laws described in Section 4.1.5, with the mass of each actuator also being found to allow for equitable comparisons to be made. The flexible modes of the LSSs can also be determined and ensured to lie within the stability regions of the numerical integration method (Section 3.2).

Actuator Mass Scaling

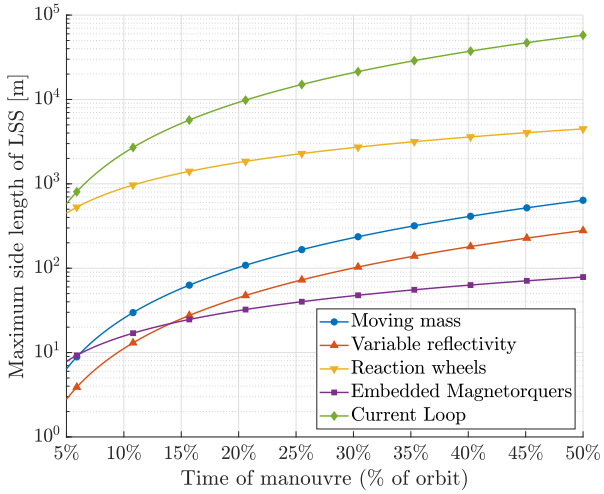
As was shown in Section 4.1.5 the maximum size of an LSS that can be actuated by a particular actuator will be dependant on the maximum angular acceleration $\ddot{\theta}$ and velocity $\dot{\theta}$ of the prescribed attitude slew manoeuvre. Given that the magnitude of the slew is already defined (90 deg), the maximum angular acceleration $\ddot{\theta}$ and velocity $\dot{\theta}$ of the manoeuvre can only be altered by varying the duration of the manoeuvre t_f.

The parameters of the LSS used are given in Table 4.2 with the angle of the sun to the reflective surface ϕ_{sun}, and the angle between the magnetic dipole of the LSS and the local magnetic field ϕ_{mag} both assumed to be 45 deg [82]. The maximum sidelength for each actuator can then be given in Fig. 4.19a as a function of the manoeuvre duration t_f in terms of the percentage of the 1,000-km-orbit's period.

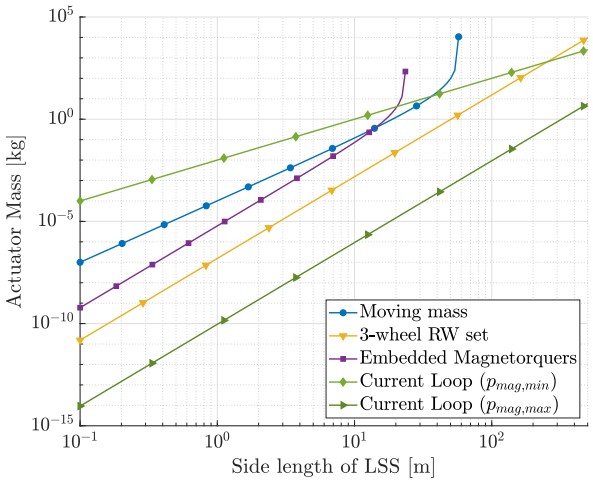
It can be seen that for a slew duration that is 15% of the orbital period, all actuator types are sufficient to actuate an LSS of sidelength 10 m. This will give a good test case for comparing each actuator type individually (Section 4.3.2). A slew duration of 15% orbital period also allows time for two 90 deg slew manoeuvres to be performed every orbit, with additional time for orbital manoeuvring, achieving a similar mission profile to LightSail-2 [24]. Therefore, the manoeuvre duration is chosen to be 947.67 s, as seen in Table 4.2. A larger LSS of side length 105 m can then be used to test combinations of different actuator types and configurations (Section 4.3.3).

The actuator masses for the two LSS side lengths can therefore be taken from Fig. 4.19b, which includes both the minimum and maximum power CL configurations. These values will be used to give a fair comparison between actuator types, as the total mass and mass distribution throughout the LSS will have a large impact on the overall dynamics of the structure.

Considering Fig. 4.19a in more detail, it can be seen that the most limited actuator types are the two SRP-based actuators, and the MAGs. The SRP-based actuators are simply



(a) Maximum side length of LSS for a polynomial slew manoeuvre at 1000 km altitude



(b) Mass of different actuator types for an attitude slew manoeuvre. Circular orbit at 1000 km altitude, with $t_f = 15\%$ of orbital period

Figure 4.19: Actuator scaling results. Assumes reflector and solar cells are oriented at 45 deg to the sun. Assumes magnetorquers orientated 45 deg to magnetic field $\boldsymbol{b}_{mag}$

limited by the available SRP, and the area-to-mass ratio ρ_{lss}. While increasing the size of the LSS will allow for more SRP to be obtained for torquing the LSS, the increased size will also increase the inertia, therefore, limiting the maximum angular acceleration. The MAGs, with a constant mass-to-torque ratio of 100 Am^2/kg, simply do not provide an efficient magnitude of torque for their mass. Instead, by utilising the full size of the LSS to produce large CLs, a much more efficient torque-to-mass ratio can be achieved. The large CLs therefore allow the actuation of much larger LSSs, on the kilometre scale (Fig. 4.19a). As expected, it can also be seen that reaction wheels allow for similarly large LSSs; however, as the size of the LSSs increases, the reaction wheels eventually require more mass than both CL actuator configurations (Fig. 4.19a). Many of these results mirror those found in previous work by Viale and McInnes [82] for orbital solar reflectors.

Structural Dynamics

The structural dynamics of the larger 105 m LSS can be shown through the linearisation and modal decomposition, completed previously in Section 4.1.4. Assuming no actuators have been added to the structure, and omitting the rigid-body modes, the first nine flexible modal shapes can be seen in Fig. 4.20 with associated frequencies for the two stiffnesses given in Table 4.4. Several modes have similar, or the same, frequencies due to the symmetries in the structure of the LSS. Knowledge of the flexible modes ensures that stable numerical integration can be ensured (Section 3.2). It has also been previously shown in Section 3.3.2 that consideration of the flexible modes is important

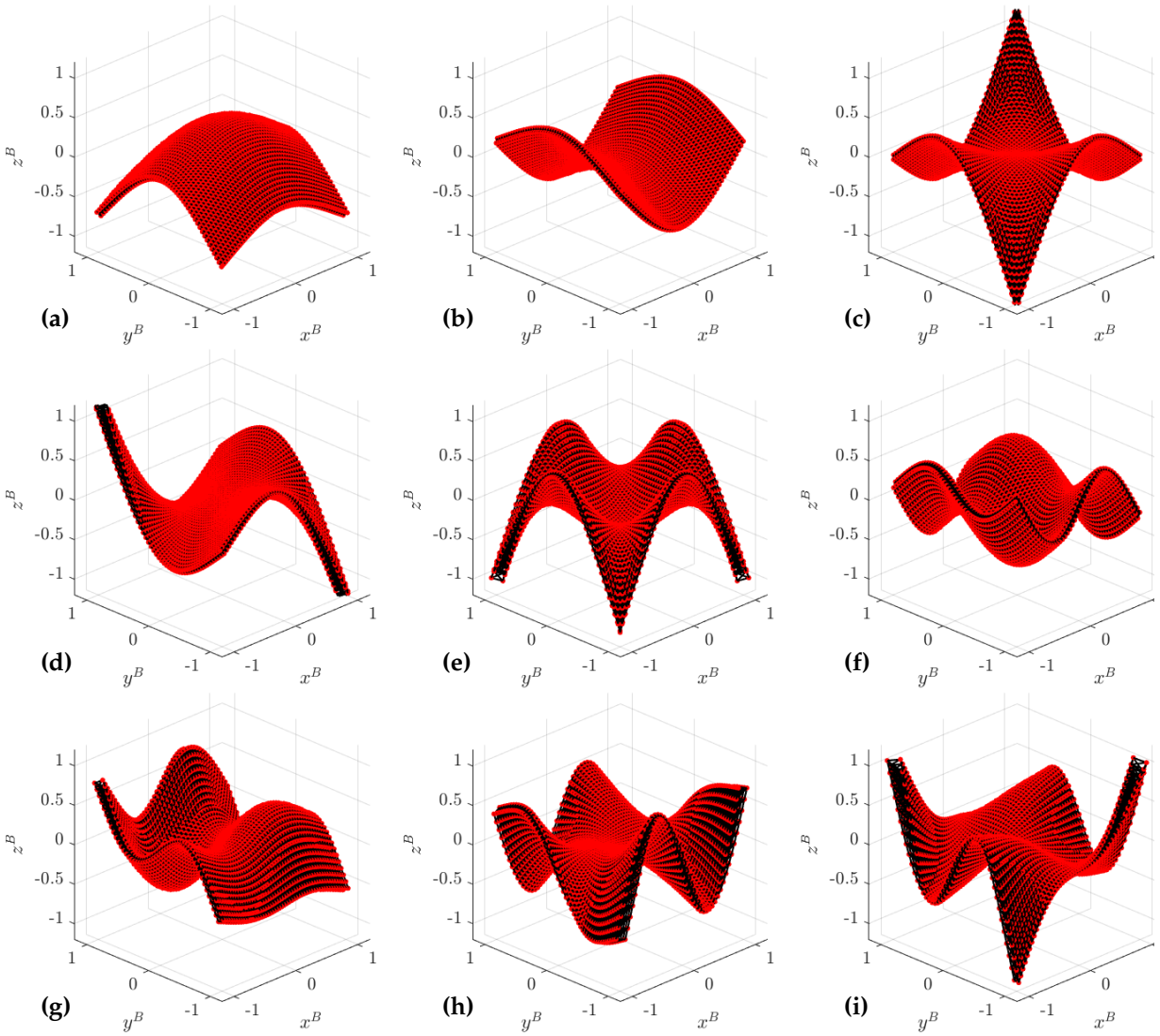


Figure 4.20: First nine flexible modal shapes of a planar homogenous LSS: $L = 105$ m (frequencies in Table 4.4)

during the design and tuning of the control algorithms to avoid their over-excitation and instability.

4.3.2 | Single Actuator Type Comparison

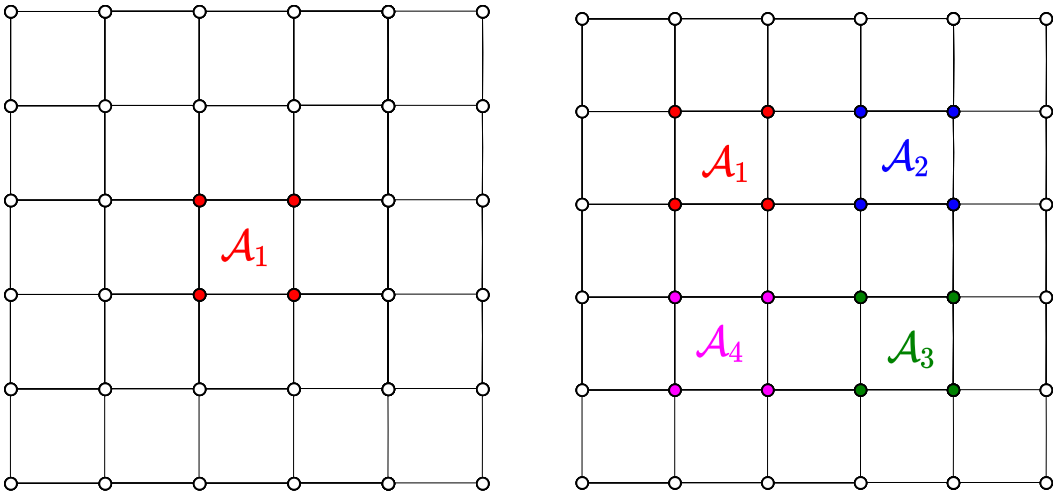
To compare the performance of each actuator type individually, the smaller ($L = 10$ m) LSS will be used. First, two reaction wheel configurations will be compared in detail before a full comparison of each actuator type; results given in terms of the deformation experienced by the LSS using Eq. (4.110).

Table 4.4: Modal frequencies (shapes in Fig. 4.20)

	Modal Frequency [m·rad/s]	
Mode	$EI = 1\ \text{Nm}^2$	$EI = 100\ \text{Nm}^2$
(a)	2.1	20.5
(b)	2.1	20.5
(c)	2.1	20.6
(e)	2.9	29.0
(f)	5.6	55.8
(g)	5.6	55.9
(h)	5.6	56.0
(i)	5.6	57.3

Reaction Wheel-Set (RWS) Configurations

The first configuration considered is the same used in Section 4.2.2, which consists of a single RWS placed centrally within the LSS structure, as illustrated in Fig. 4.21a; the mass of which can be found in Fig. 4.19b. Secondly, as shown by Robb *et al.* [66], it can be beneficial to use a distribution of smaller torques across the LSS instead of a single, large, centralised torque. Previously, this was only shown for magnetic torques [66], but here it will be shown to hold true for other actuator types such as reaction wheels. A configuration of four smaller RWSs, each with a quarter of the mass of the larger single RWS, can be seen in Fig. 4.21b and will be compared to the centralised configuration.



(a) Centralised 1 RWS configuration **(b)** Distributed 4 RWS configuration

Figure 4.21: RWS positions within LSS structure (top-down view)

The InvSim-NR method can be used to solve both RWS configurations, with detailed results for the single centralised RWS configuration previously shown in Fig. 4.15. A comparison with the distributed RWS configuration is presented in Fig. 4.22. Both configurations produce the same total torque (Fig. 4.22a), with the distributed configura-

tion effectively halving the deformation during the manoeuvre (Fig. 4.22c). The deformation across the LSS surface at the time of maximum RMS deformation can be seen for the centralised and distributed configurations in Fig. 4.22b and Fig. 4.22d respectively.

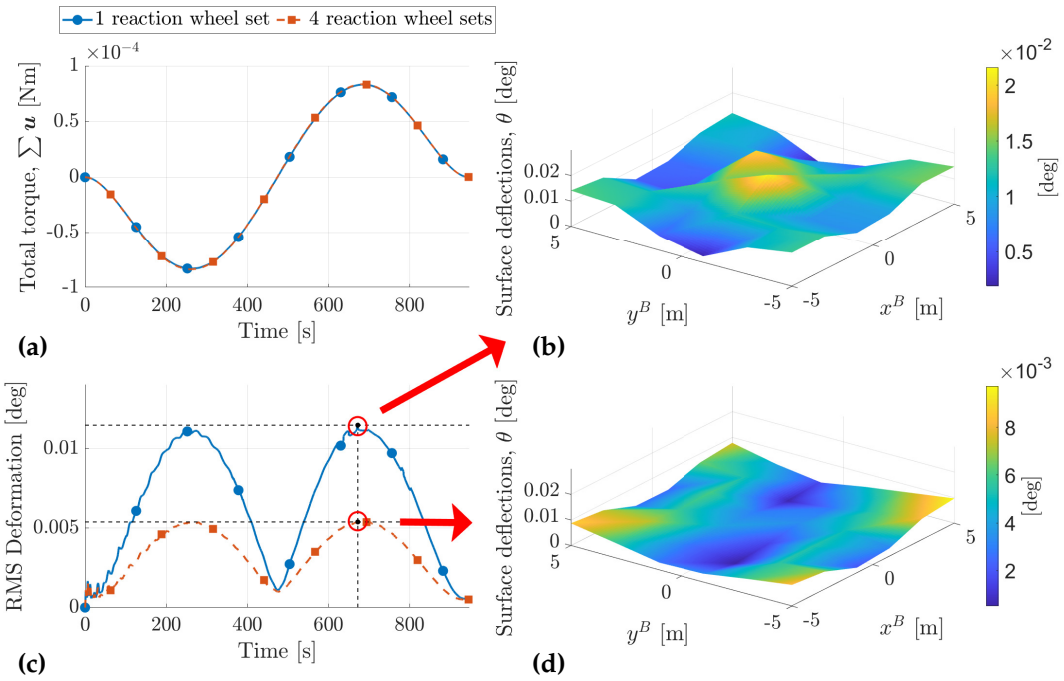


Figure 4.22: Comparison of LSS deformation for one central RWS vs 4 distributed RWS. $L = 10$ m, $(EI)_{tot} = 1\text{Nm}^2$. (a) total torque, (b) deformation across surface at time of max RMS deformation for the centralised configuration, (c) RMS deformation during the manoeuvre for both configurations, (d) deformation across surface at time of max RMS deformation for the distributed configuration

.

Full Comparison

The InvSim solution can be found for all actuator types using the InvSim-NR method with special consideration of the initial input estimate $\boldsymbol{u}_0$ and output constraints required for the magnetic actuators (MAG & CL), as discussed previously in Section 4.2.3. Thus, a comparison of the structural deformation for small LSSs ($L = 10$ m), of varying stiffness, performing a 90 deg slew for different actuator types is performed. This includes the single centralised RWS configuration (Fig. 4.21a), the distributed four RWS configuration (Fig. 4.21b), the MM actuator (Fig. 4.11), RCDs (Fig. 4.9), axial MAGs (Fig. 4.6), and large CLs (Fig. 4.7). The CL configuration is again assumed to be of minimum power/maximum mass. In this scenario, the actuator scaling laws found in Fig. 4.19b give the mass for each actuator type.

The results for a small, highly flexible ($(EI)_{tot} = 1$ Nm^2) LSS are given in Fig. 4.23. First, it is observed that the deformation response for the MM actuator is still diverging

at the end of the manoeuvre, giving the worst performance of all the actuators. Since the MM is substantial, approximately 13% of the total mass of the LSS in this case, and the structure is extremely flexible, the movement of the control mass excites the flexible modes dramatically.

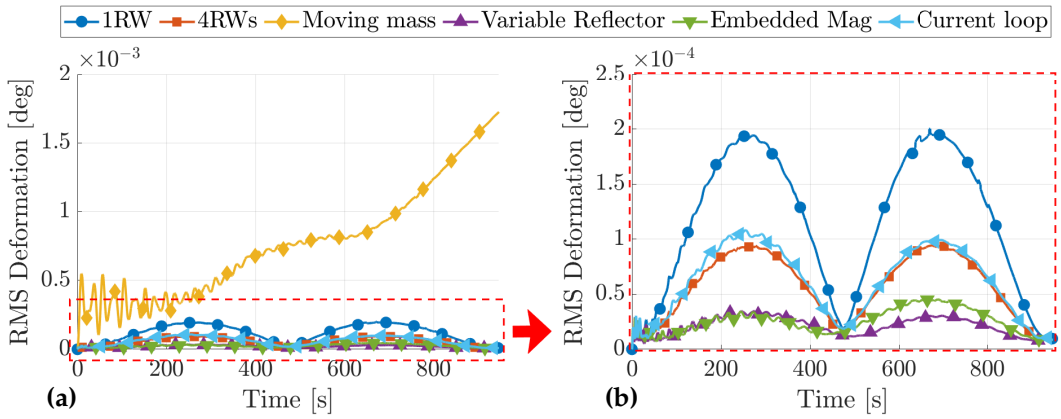


Figure 4.23: Compare deformation of LSS for different actuator types: $L = 10$ m, $(EI)_{tot} = 1$ Nm^2. (a) zoomed out, (b) zoomed in

Looking at Fig. 4.23b, the deformation of the other actuator types can be seen more clearly. After the MM, the next worst-performing actuator is the single centralised RWS; with the CL, and four-RWS configurations reducing the deformation in half. The four small RWSs also provide additional redundancy to the system, and therefore, while not decreasing the deformation as much as some other actuator types, may provide a good trade-off between ease of implementation, technological readiness, and minimisation of LSS deformation.

The best-performing actuators in terms of LSS deformation are the MAGs and RCD actuators. These actuators reduce the deformation by approximately 70% compared to the single centralised RWS. The RCD actuators, however, require significantly less mass than the MAGs, making them perhaps the most attractive choice for attitude control of LSSs.

Similar results and trends can be seen for a stiffer LSS ($(EI)_{tot} = 100$ Nm^2) of the same size, Fig. 4.24. Comparing the MM actuator to the other actuator types (Fig. 4.24a), it can be seen that there is comparatively less deformation for a stiffer structure. Therefore, the MM actuator is more applicable for stiffer LSSs, which will not be excited as significantly by the movement of the MM.

4.3.3 | Actuator Combinations for Larger Space Structures

While LSSs on a smaller scale (i.e. tens of meters) may be useful for some applications, many of the use cases outlined in Section 1.1.2 will require LSSs of much larger scales.

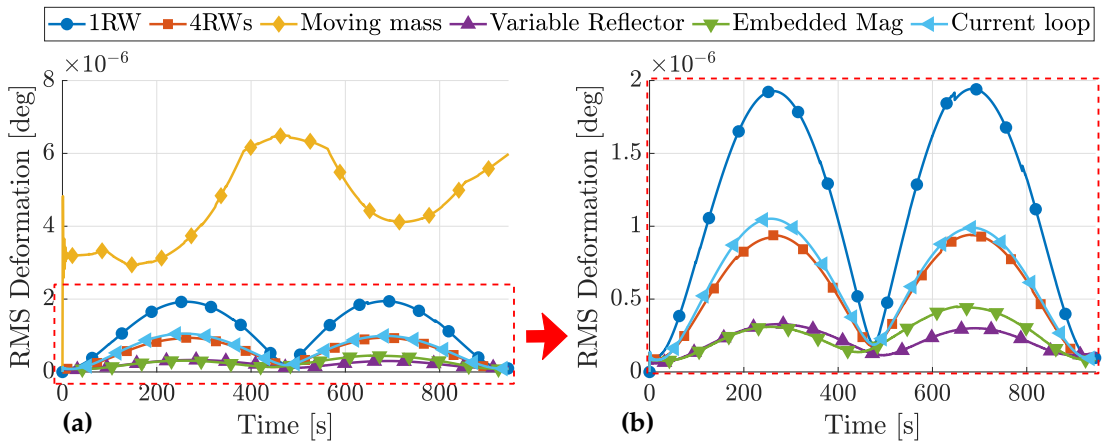


Figure 4.24: Compare deformation of LSS for different actuator types: $L = 10$ m, $(EI)_{tot} = 100$ Nm^2. (a) zoomed out, (b) zoomed in

However, as seen in Fig. 4.19a, several of the actuator types will not be able to provide sufficient actuation to perform the 90 deg slew manoeuvre in the prescribed time on their own. An LSS of sidelength $L = 105$ m and stiffness $(EI)_{tot} = 200$ Nm^2 will be used to investigate whether different actuator types can be combined to utilise the benefits, and minimise the limitations of each.

Limitations of RCDs

While it is clear from Fig. 4.19a that RCDs cannot provide sufficient torques for the prescribed slew manoeuvre, it can be further confirmed by performing an InvSim of a 2×2 RCD grid (Fig. 4.9). An unconstrained solution is found using InvSim-NR with the results shown in Fig. 4.25. While the output is successfully tracked (Fig. 4.25a), the input solution exceeds the physical bounds of $\eta_s \in [0, 1]$ (Fig. 4.25b).

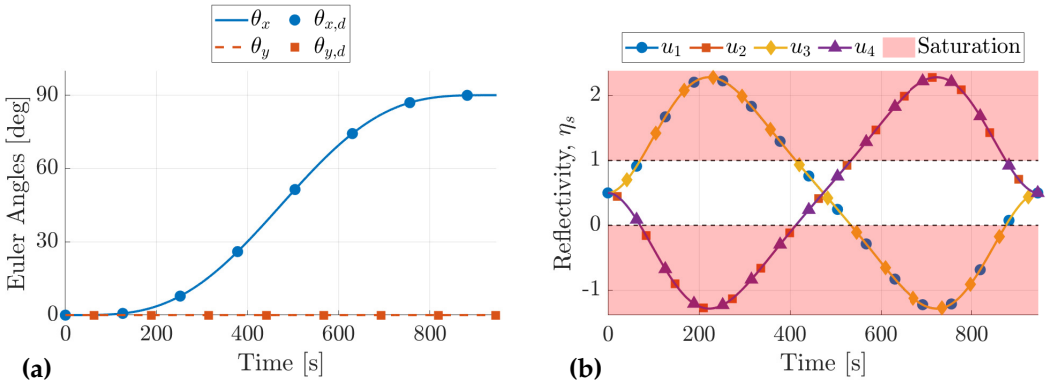


Figure 4.25: RCD actuation only: $L = 105$ m, $(EI)_{tot} = 200$ Nm^2. (a) outputs, (b) inputs

Limitations of RWSs

Once again, considering Fig. 4.19a, it can be seen that using a single centralised RWS (Fig. 4.21a) would provide sufficient actuation for the prescribed slew manoeuvre. The mass of the single centralised RWS for an LSS of sidelength $L = 105$ m is again defined by the scaling law found in Fig. 4.19b. The InvSim solution can then be found using InvSim-NR, with results in Fig. 4.26. While the output is successfully tracked (Fig. 4.26a), the deformation experienced during the manoeuvre is now significant and undesirable (Fig. 4.26c and Fig. 4.26d).

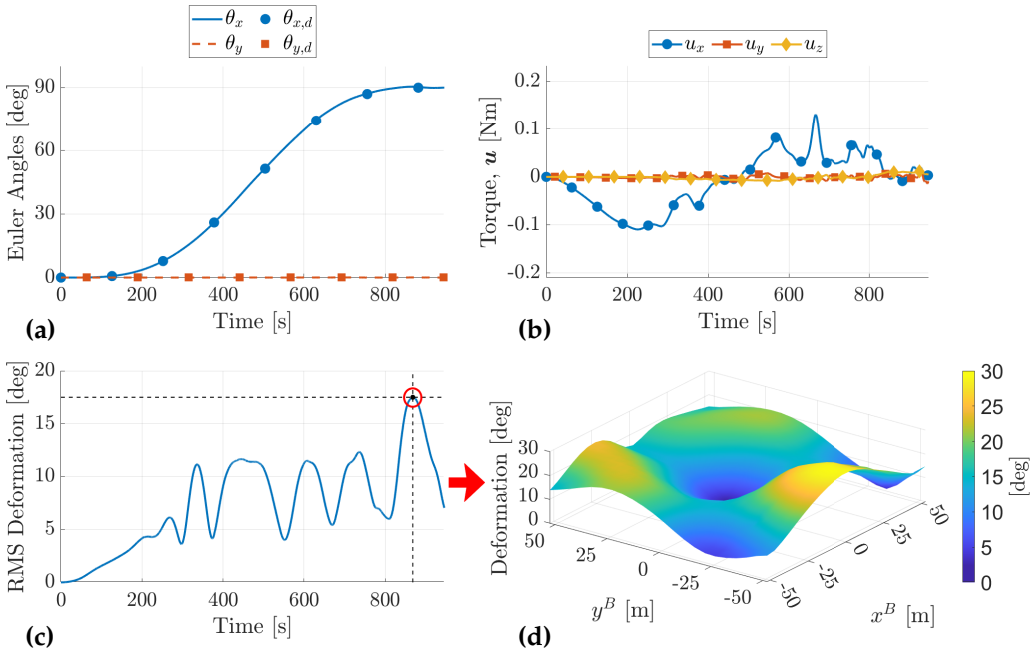


Figure 4.26: One RWS only: $L = 105$ m, $(EI)_{tot} = 200$ Nm^2. (a) outputs, (b) inputs, (c) RMS deformation during manoeuvre, (d) deformation across surface at time of max RMS deformation

Central RWS with RCDs

Combining both the single centralised RWS (Fig. 4.21a) and a 2× RCD grid (Fig. 4.9) may therefore allow for the weakness in each actuator type to be mitigated, while making the most of their strengths. The InvSim-SQP method can be employed to facilitate the combination of the two different actuator types. This allows for the actuator saturation of the RCD control devices to be enforced by implementing them as inequality constraints, and a suitable cost function can be chosen to achieve the desired behaviour. The objective is to slew the LSS while minimising deformation, and since the RWS induces greater deformation, its use should be minimised. Therefore, the cost function used previously (Eq. (4.109)) can be modified to minimise the RWS control effort:

$$s(\boldsymbol{u}[kT]) = \frac{1}{2}\boldsymbol{u}^\top[kT]\mathbf{G}\boldsymbol{u}[kT] \tag{4.111}$$

where **G** is a diagonal matrix which can be defined as:

$$\mathbf{G} = \text{diag}\left(\begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}\right) \tag{4.112}$$

which will penalise the RWS inputs to minimise their use, as can be seen by considering the full input vector $\boldsymbol{u}$ order:

$$\boldsymbol{u} = \begin{bmatrix} u_{rw,x} & u_{rw,y} & u_{rw,z} & u_{rcd,1} & u_{rcd,2} & u_{rcd,3} & u_{rcd,4} \end{bmatrix}^\top \tag{4.113}$$

where $\begin{bmatrix} u_{rw,x} & u_{rw,y} & u_{rw,z} \end{bmatrix}^\top$ and $\begin{bmatrix} u_{rcd,1} & u_{rcd,2} & u_{rcd,3} & u_{rcd,4} \end{bmatrix}^\top$ are RWS and RCD inputs respectively. Therefore, the cost function (Eq. (4.111)) will minimise the use of the RWS. The results of the InvSim for combined centralised RWS and RCD actuator can be seen in Fig. 4.27. The output is successfully tracked and the saturation limit of the RCD actuator is respected (Fig. 4.27b). Additionally, through comparison of Fig. 4.27c and Fig. 4.27d to Fig. 4.26c and Fig. 4.26d, it can be seen that the deformation experienced during the manoeuvre is reduced significantly. This is achieved by utilising SRP torques as much as possible, only applying RWS torques when the RCD actuator has reached saturation (Fig. 4.27b). InvSim-SQP is, therefore, able to converge to a solution for multiple actuator types, allocating control effort in such a way that deformation of the LSS can be minimised while still following the prescribed slew path.

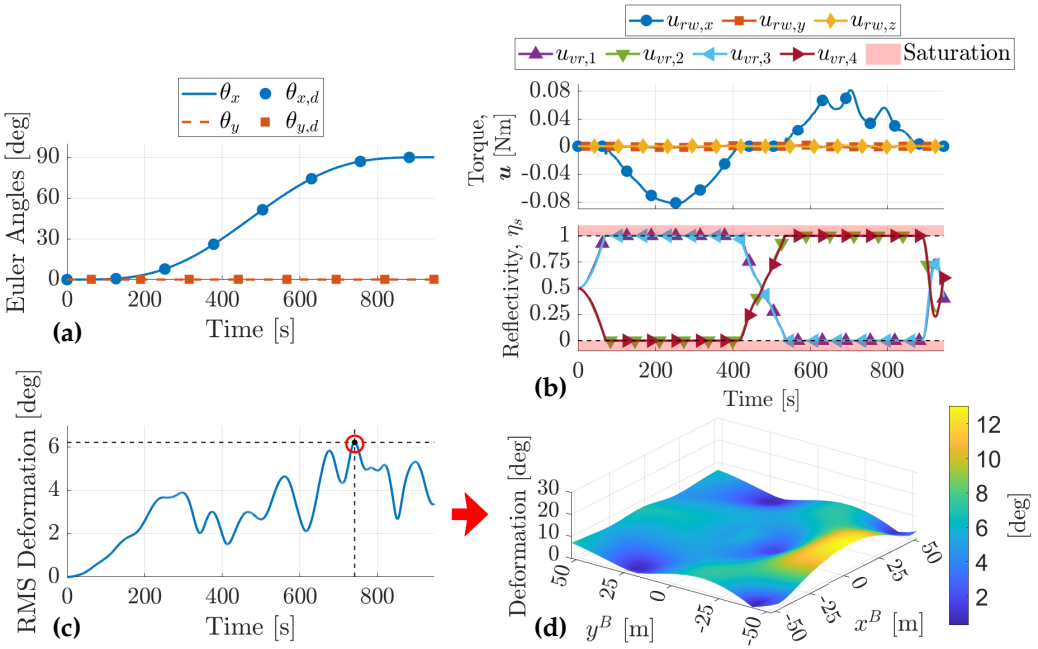


Figure 4.27: Combined single RWS and RCD actuator: $L = 105$ m, $(EI)_{tot} = 200$ Nm^2. (a) output, (b) input, (c) RMS deformation during manoeuvre, (d) deformation across surface at time of max RMS deformation

Central RWS with Distributed MAG

Another actuator type that exhibits low deformation in the LSS structure is the MAG architecture, as seen previously in Figs. 4.23 and 4.24. However, considering Figs. 4.19a and 4.19b, it can be seen that sufficient actuation from the MAGs is not possible for side length $L = 105$ m and for the prescribed slew manoeuvre. InvSim-SQP can be utilised once again, minimising the RWS torques when using a single centralised RWS (Fig. 4.21a) and MAGs (Fig. 4.6) in combination. This is achieved through the same cost function used previously, Eq. (4.111), with the diagonal matrix $\mathbf{G}$ modified:

$$\mathbf{G} = \text{diag}\left(\begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 \end{bmatrix}\right) \tag{4.114}$$

The MAGs oriented in each direction will, for simplicity, be mapped to the same input giving the three inputs $[u_{mag,x}\ u_{mag,y}\ u_{mag,z}]^\top$ and the full combined full input vector:

$$\boldsymbol{u} = \begin{bmatrix} u_{rw,x} & u_{rw,y} & u_{rw,z} & u_{mag,x} & u_{mag,y} & u_{mag,z} \end{bmatrix}^\top \tag{4.115}$$

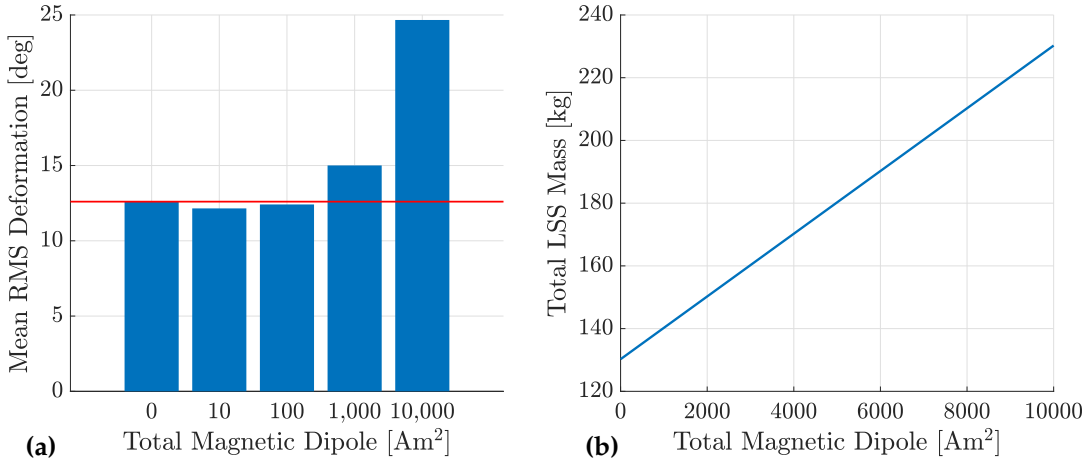


Figure 4.28: Mean RMS deformation for combined single RWS and MAG with varying total magnetic dipole: $L = 105$ m, $(EI)_{tot} = 200$ Nm^2. (a) mean RMS deformation, (b) total LSS mass

The InvSim-SQP solution is found for the combined single centralised RWS with MAGs of a varying total magnetic dipole; the mean RMS deformation during the slew manoeuvre is given in Fig. 4.28a for each case, including RWS only (marker in red). By introducing a small magnetic dipole actuation, a small reduction in the deformation can be achieved. However, increasing the total magnetic dipole to $> 1,000$ Am^2 causes the deformation in the structure appears to increase higher than in the RWS-only case.

As seen in Fig. 4.28b, this is due to the total mass of the LSS increasing rapidly as magnetic dipole actuation is added (100 Am^2/kg). For a given stiffness, $(EI)_{tot} = 200$

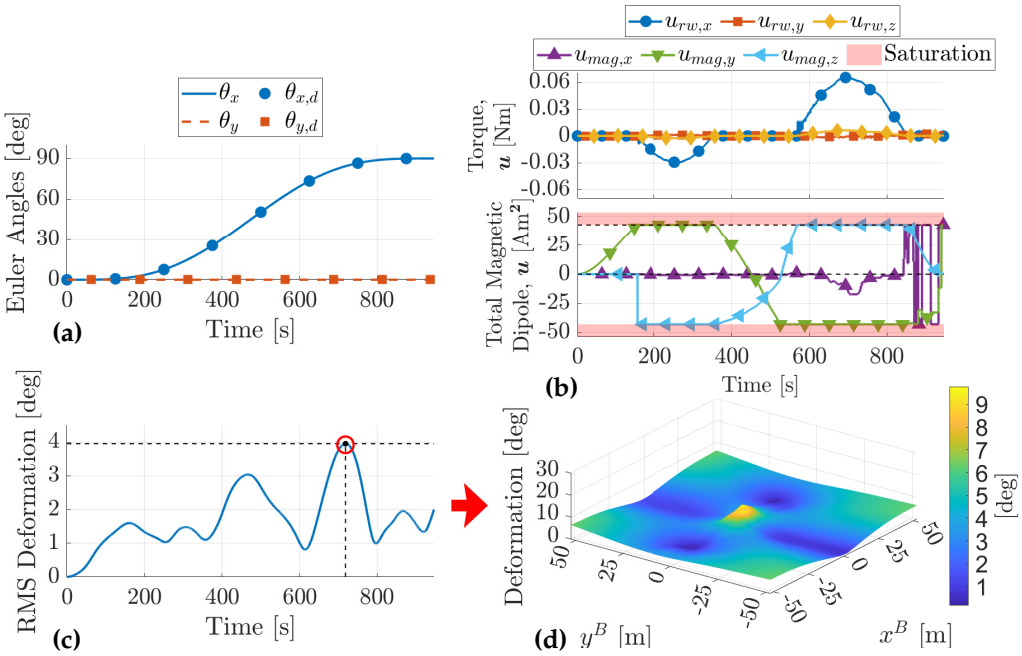


Figure 4.29: Combined single RWS and MAG configuration, total magnetic dipole 100 kAm^2, assuming no MAG mass: $L = 105$ m, $(EI)_{tot} = 200$ Nm^2. (a) output, (b) input, (c) RMS deformation during manoeuvre, (d) deformation across surface at time of max RMS deformation

Nm^2 in this case, an increase in the total mass will increase the total deformation of the structure. This is confirmed by considering Fig. 4.29 where the MAGs are assumed to have no mass and a total magnetic dipole of 100 kAm^2. The deformation in this case (Fig. 4.29c) is reduced significantly, comparable to the performance of the combined single RWS-acrcd architecture (Fig. 4.27c).

It was found through the mass scaling laws that, for larger LSSs, the CLs are more mass efficient (Fig. 4.19b) even in the case of the minimum power configuration. Therefore, the performance of using a CL actuator by itself will be compared to the combined actuator configurations outlined above. The InvSim results of a CL actuator, assuming a minimum power mass of $m_a = \rho_{lss} L^2$ [82], can be found in Fig. 4.30. This displays the least deformation of any configuration for the larger LSS (Fig. 4.30c and Fig. 4.30d) owing to its distribution of torques throughout the structure, and favourable mass scaling when compared to the MAGs.

Interestingly, as the scale of the LSS continues to increase, the mass of the CL becomes smaller than that of an RWS Fig. 4.19b. It can therefore be deduced that for larger, more flexible LSSs the CL actuator will become more and more advantageous, with minimal structural deformation and less mass than an RWS. As detailed by Viale and McInnes [82], the mass of the CL actuator can also be reduced further at the cost of greater power consumption. Using large CLs for larger-scale LSSs proves far superior to using the MAG configuration in terms of both mass efficiency and structural deformation.

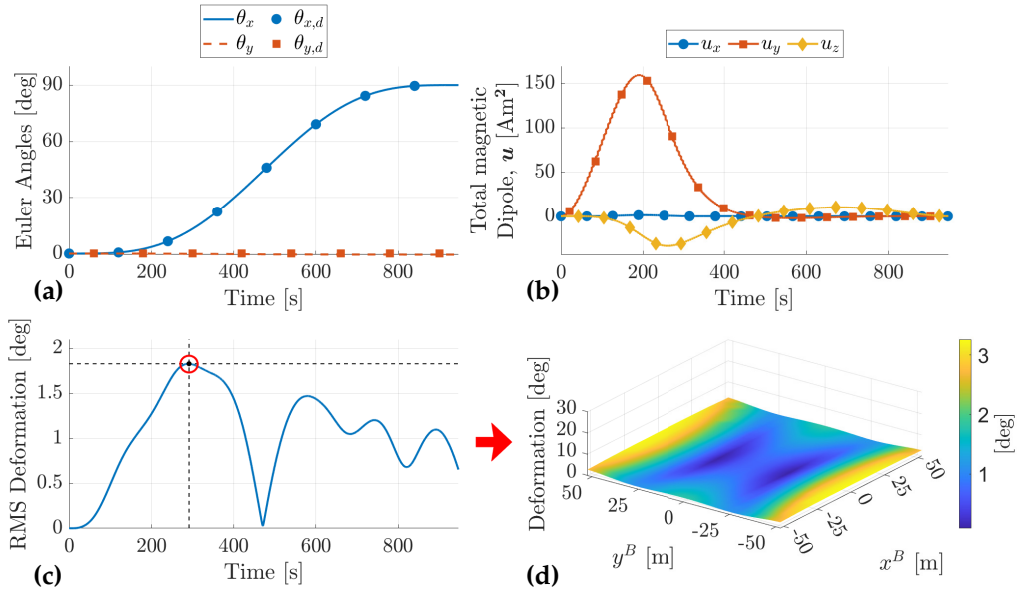


Figure 4.30: CL actuator only: $L = 105$ m, $(EI)_{tot} = 200$ Nm^2. (a) outputs, (b) inputs, (c) RMS deformation during manoeuvre, (d) deformation across surface at time of max RMS deformation

CLs also provide reduced deformation when compared to the combined central RWS, and RCD configuration. However, the combined central RWS and RCD configuration may still be more advantageous in terms of the reduced overall mass of the LSS, and lower power requirements. The CL configuration may also still require the inclusion of an RWS to add redundancy, which would allow for the size of the CLs to be reduced.

4.4 | Summary

This chapter demonstrated the use of InvSim as a tool to aid in the design of satellites with flexible structures. This was achieved through a case study, which investigated the performance of various attitude control actuator configurations for generic, gossamar LSSs. The InvSim-based design framework included a developed modular mathematical model, and actuator scaling laws to ensure a fair comparison between various actuator configurations.

The InvSim algorithm was also modified to better handle over-actuated systems, replacing the NR solver with a sequential-quadratic-programming (SQP) constrained local optimisation method, allowing for specific inverse solutions to be converged upon, without the need to carefully select the initial input estimate $\boldsymbol{u}_0$; a particular problem when utilising magnetic control only, as first observed in Chapter 2. The optimisation-based InvSim therefore allowed for combinations of different actuator types to be

utilised together intelligently, giving the best possible performance for each actuator combination, e.g. minimising deformation in the structure.

The results of the design investigation, illustrated in Table 4.5, found that, in general, a more even distribution of torques throughout the LSS structure would minimise deformation. However, for larger scale LSSs, due to limitations in the actuator scaling, a combination of a single centralised reaction-wheel set and distributed reflectivity control devices could produce a good compromise between structural deformation, total mass, and power consumption.

Table 4.5: Summary of attitude control actuator investigation

	Deformation	Maximum Size*	Mass *	Power [82]
Central RWS	High	>1km	Medium	Medium
Distributed RWS	Medium	>1km	Medium	Medium
Moving mass (MM)	Very High	<100m	Very High	Low
Reflectivity control device (RCD)	Low	<100m	Low	Low
Embedded magnetorquer (MAG)	Low	<100m	Very High	Very High
Current loops (CL)	Medium	>1km	High	High
Central RWS + RCD	Low	>1km	Medium	Medium

* Assuming LSS properties and maneouvre parameters in Table 4.2. Quantatative details in Fig. 4.19

It was also noted during this design investigation that as the number of actuators added to the system increased, the time taken to compute the InvSim solution increased considerably. As discussed previously, it was found that distributing more actuators across an LSS minimised deformation in the satellite's flexible structure. Investigating the dynamic behaviour of such distributed actuator configurations can therefore take prohibitively long to complete due to the limitations in the current general integral-based InvSim algorithm. The next chapter will therefore investigate one possible new, novel approach to improve the numerical performance of the integral-based InvSim algorithm.

Chapter 5 | A Faster Inverse Simulation Method

THIS thesis has presented several benefitial applications of InvSim to the dynamics of satellites with flexible structures. Chapter 3 demonstrated the use of InvSim for attitude path planning and control, and Chapter 4 used InvSim within the design process of LSSs. Whilst this demonstrates much promise for InvSim within the satellite sector, each use case requires that the InvSim solution be computed within a reasonable amount of time.

It has been discussed previously that the numerical efficiency of the InvSim algorithm will have a profound impact when attempting to implement the developed InvSim-based attitude control algorithm on real-time hardware. Additionally, a numerically inefficient InvSim could cause the path planning algorithm developed in Section 3.4 to take considerable time to generate desired attitude paths, particularly as the InvSim solution must be evaluated many times by the outer optimisation loop.

Several configurations investigated in Chapter 4 using InvSim consisted of large numbers of actuators, drastically increasing the computational complexity of the InvSim numerical process. With a sufficiently large number of actuators, the integral-based InvSim approach presented thus far becomes prohibitively computationally expensive to run. Without the ability to run many InvSims rapidly, its use as a tool for quick prototyping during initial design phases becomes limited.

Previously in Section 3.2 it was shown that the choice of integration method could allow for the speed of InvSim when solving stiff systems to be dramatically improved. This, however, does not address the issue associated with large numbers of inputs. Instead, the degradation in computational speed for systems with large numbers of inputs is linked to the calculation of the Jacobian required by the NR solver. As the number of inputs to the system increases, the size of the Jacobian also increases, requiring additional numerical integration of the model to obtain the additional partial derivatives through finite differencing. This chapter will therefore look to modify the existing InvSim method (Section 2.1), improving computational efficiency through faster calculation of the Jacobian. This novel new method will not only make designing systems

with large numbers of inputs more feasible, but will also have implications when deploying InvSim-based attitude path planning and control algorithms on real-time hardware.

5.1 | Modified Inverse Simulation Algorithm

Whether using central, forward, or backward differencing in the calculation of the Jacobian, the model needs to be evaluated and integrated many times. The number of times that the model is evaluated and integrated will depend on the following factors: (1) the duration of the timeseries to be solved, (2) the size of the integration timestep used, (3) the integration method being used (RK4, Euler etc), (4) the size of the timestep between successive input solutions (also known as the control timestep or discretisation timestep), (5) the number of iterations needed to solve each timestep of the InvSim, and (6) the number of times the model needs to be called to obtain the Jacobian for each iteration of the solver.

Points 1 to 4 above are somewhat independent of the InvSim method and are more related to the system being studied, therefore, not strictly considered here for improvement. Point 5 will also mostly be affected by the specific dynamics involved, with more nonlinear and complex systems requiring more iterations to solve each timestep. However, the accuracy of the calculated Jacobian used to update the input estimate will also have a significant impact on point 5. Therefore, any modifications to the calculation of the Jacobian that reduce its accuracy could slow down the InvSim process, taking more NR iterations to converge to the solution.

The approach proposed in this chapter aims to reduce the number of times that the model needs to be called during the calculation of the Jacobian (point 6 above) to speed up the integral InvSim approach. With the traditional integral InvSim approach, the number of times the model needs to be called to calculate the Jacobian will depend on whether central, forward, or backward differencing is used, and the number of inputs to the system. Central differencing will require the model to be called twice for every input: once to calculate the effect of a positive perturbation and once to calculate the effect of a negative perturbation in the input. Forward and backward differencing reduces the number of model evaluations to one per input, only applying perturbations in one direction; the tradeoff being a reduction in accuracy in the calculated partial derivatives.

Reduction of the number of times the model is evaluated has previously been achieved through the use of a method known as automatic differentiation [139]. An example of *forward mode* automatic differentiation is illustrated in Fig. 5.1, whereby the par-

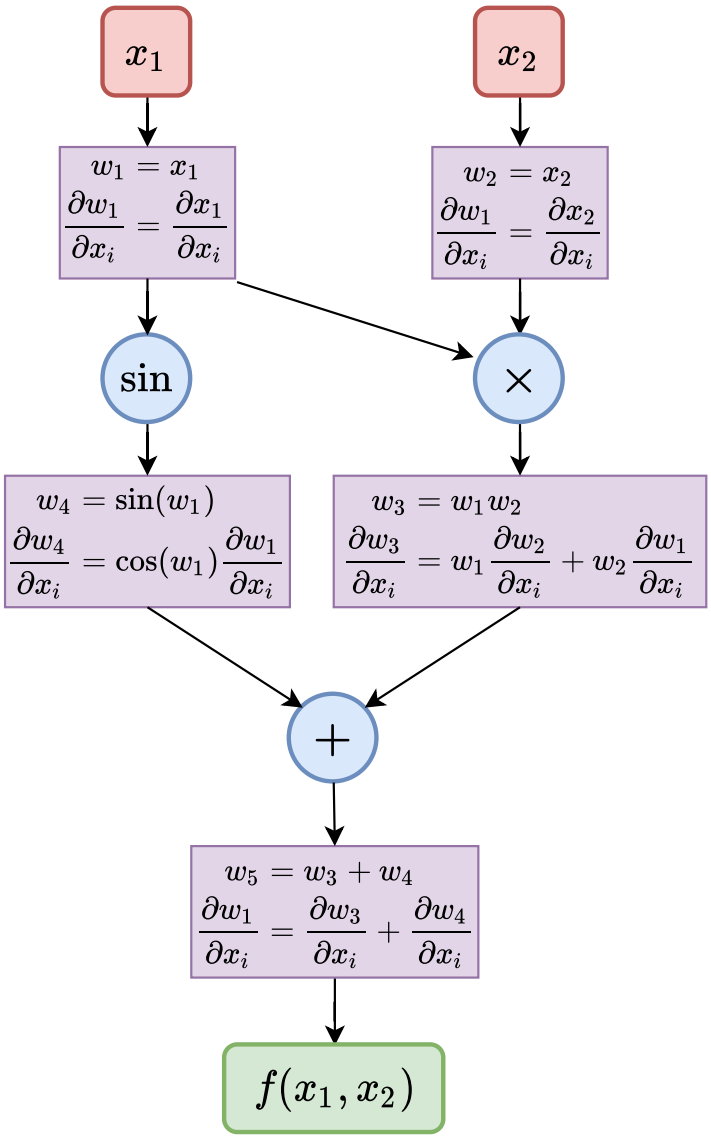


Figure 5.1: Automatic Differentiation example (forward mode). $f(x_1, x_2) = \sin(x_1) + x_1 x_2$

tial derivatives of the function $f(x_1, x_2)$ can be found through the use of chain rule. For example, when finding the partial derivative $\partial f / \partial x_2$ using forward automatic differentiation, the terms $\partial w_1 / \partial x_2$ and $\partial w_2 / \partial x_2$ at the first two nodes in Fig. 5.1 would become 0 and 1 respectively. Carrying through the values and propagating through subsequent nodes, the partial derivative of the function is found at the final node to be $\partial f / \partial x_2 = \partial w_5 / \partial x_2$.

Thus by deconstructing the mathematical model being solved by InvSim into its elementary mathematical operations (Fig. 5.1), the partial derivatives of each output $\boldsymbol{y}_j$ w.r.t each input $\boldsymbol{u}_i$ can be calculated automatically and used to form the Jacobian. Using automatic differentiation for InvSim [139] only requires that the model be evaluated twice per timestep as the values at each node (Fig. 5.1) can be saved to memory and reused for subsequent partial derivative at the same timestep. Automatic differentia-

tion also provides a more accurate Jacobian, as it avoids the truncation errors associated with numerical differencing.

However, implementation of automatic differentiation usually requires external software to modify the model's source code into a form that accommodates the chain rule approach. This could prove difficult in the case of a modular mathematical model, such as that used in Chapter 4, as the structure of the elementary mathematical operations is not fixed and can change depending on which modules (e.g. different actuators) are included. The process of generating the automatic differentiation code, therefore, introduces an additional step in the implementation of InvSim, requiring the code to be regenerated every time a modification is made to the underlying model. This leads to a loss in the overall generality of the integral InvSim method, one of its most significant advantages. Furthermore, for large complex models, the memory required to store the values at each node of the automatic differentiation process could become cumbersome, particularly if InvSim needs to be executed on embedded hardware for control applications (Section 3.3).

This chapter instead suggests replacing the conventional NR method with Broyden's method [189], similar to QN approaches used in numerical optimisation (Appendices D.1 and D.2). Broyden's method makes use of an estimated Jacobian to update the solution, instead of calculating the Jacobian explicitly. This reduces the number of model evaluations needed to one, regardless of the number of inputs to the system. Ideally, both the input estimate and the Jacobian should converge as the solver iterates, resulting in an adequate input solution. It can be intuitively seen, however, that the Jacobian estimate will be less accurate than calculating it explicitly, and may therefore require more solver iterations to finally converge. Therefore, the performance of InvSim with Broyden's method should be compared to traditional NR with numerical differencing, quantified by the total number of model evaluations needed for a full InvSim. First, however, the modified InvSim method utilising Broyden's approach is detailed.

5.1.1 | Broyden's Method for Faster Inverse Simulation

Recollect that the InvSim problem (Section 2.1) for a general nonlinear dynamic system:

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}) \tag{5.1}$$

$$\dot{\boldsymbol{y}} = \boldsymbol{g}(\boldsymbol{x}) \tag{5.2}$$

reduces to solving a system of equations defined by the ν^{th}-order derivative of the desired output $\overset{(\nu)}{\boldsymbol{y}}_d$:

$$\overset{(\nu)}{\delta \boldsymbol{y}} = \overset{(\nu)}{\boldsymbol{y}}[(k+1)T] - \overset{(\nu)}{\boldsymbol{y}}_d[(k+1)T] = \overset{(\nu)}{\boldsymbol{h}}(\boldsymbol{x}[kT], \boldsymbol{u}[kT]) - \overset{(\nu)}{\boldsymbol{y}}_d[(k+1)T] = 0 \tag{5.3}$$

where $\dot{\boldsymbol{h}}^{(\nu)}(\boldsymbol{x}[kT], \boldsymbol{u}[kT])$ is the forward simulation and mapping of the input $\boldsymbol{u}[kT]$ to the ν^{th}-order derivative of the output $\dot{\boldsymbol{y}}^{(\nu)}[(k+1)T]$, which is solved using the NR method (Appendix D.1):

$$\boldsymbol{u}_n[kT] = \boldsymbol{u}_{n-1}[kT] - \left[\mathbf{J}_{\boldsymbol{u}}(\dot{\boldsymbol{h}}^{(\nu)})\right]_n^{-1} \overset{(\nu)}{\delta \boldsymbol{y}}_n \tag{5.4}$$

with the partial derivatives of the Jacobian $\left[\mathbf{J}_{\boldsymbol{u}}(\dot{\boldsymbol{h}}^{(\nu)})\right]_n$ calculated through central, forward, or backward finite differencing:

$$\frac{\partial \dot{\boldsymbol{h}}^{(\nu)}\left(\boldsymbol{x}[kT], \boldsymbol{u}_{n-1}[kT]\right)}{\partial u_i} = \cdots$$

$$\cdots = \begin{cases} \dfrac{\dot{\boldsymbol{h}}^{(\nu)}\left(\boldsymbol{x}[kT], \boldsymbol{u}_{n-1}[kT] + \Delta u_i \hat{\boldsymbol{u}}_{\boldsymbol{i}}\right) - \dot{\boldsymbol{h}}^{(\nu)}\left(\boldsymbol{x}[kT], \boldsymbol{u}_{n-1}[kT] - \Delta u_i \hat{\boldsymbol{u}}_{\boldsymbol{i}}\right)}{2\Delta u_i} & \text{central} \\ \dfrac{\dot{\boldsymbol{h}}^{(\nu)}\left(\boldsymbol{x}[kT], \boldsymbol{u}_{n-1}[kT] + \Delta u_i \hat{\boldsymbol{u}}_{\boldsymbol{i}}\right) - \dot{\boldsymbol{h}}^{(\nu)}\left(\boldsymbol{x}[kT], \boldsymbol{u}_{n-1}[kT]\right)}{\Delta u_i} & \text{forward} \\ \dfrac{\dot{\boldsymbol{h}}^{(\nu)}\left(\boldsymbol{x}[kT], \boldsymbol{u}_{n-1}[kT]\right) - \dot{\boldsymbol{h}}^{(\nu)}\left(\boldsymbol{x}[kT], \boldsymbol{u}_{n-1}[kT] - \Delta u_i \hat{\boldsymbol{u}}_{\boldsymbol{i}}\right)}{\Delta u_i} & \text{backward} \end{cases} \tag{5.5}$$

where Δu_i is the of the perturbation applied to the i^{th} input, and $\hat{\boldsymbol{u}}_{\boldsymbol{i}}$ is the unit vector direction of the i^{th} input.

To improve the computational efficiency of the existing InvSim algorithm, Broyden's method [189] replaces the existing NR and finite differencing approach. Broyden's method makes use of an estimated Jacobian to solve sets of simultaneous nonlinear equations such as Eq. (5.3). Letting the Jacobian $\left[\mathbf{J}_{\boldsymbol{u}}(\dot{\boldsymbol{h}}^{(\nu)})\right]_n = \mathbf{J}_n$ for conciseness, the Brodyen Jacobian estimate update used in this chapter is defined:

$$\mathbf{J}_n = \mathbf{J}_{n-1} + \frac{\left(\partial \dot{\boldsymbol{y}}_{n-1}^{(\nu)} - \mathbf{J}_{n-1} \partial \boldsymbol{u}_{n-1}\right) \partial \boldsymbol{u}_{n-1}^{\top}}{\partial \boldsymbol{u}_{n-1}^{\top} \partial \boldsymbol{u}_{n-1}} \tag{5.6}$$

where $\partial \boldsymbol{u}_{n-1}$ is the input estimate update from the previous solver iteration:

$$\partial \boldsymbol{u}_{n-1} = \boldsymbol{u}_{n-1} - \boldsymbol{u}_{n-2} = \boldsymbol{u}_{n-1} - \mathbf{J}_{n-1}^{-1} \overset{(\nu)}{\delta \dot{\boldsymbol{y}}}_{n-1} \tag{5.7}$$

and where $\partial \dot{\boldsymbol{y}}_{n-1}^{(\nu)}$ is the change in Eq. (5.3) between updates caused by $\partial \boldsymbol{u}_{n-1}$:

$$\partial \dot{\boldsymbol{y}}_{n-1}^{(\nu)} = \delta \dot{\boldsymbol{y}}_{n-1}^{(\nu)} - \delta \dot{\boldsymbol{y}}_{n-2}^{(\nu)} = \dot{\boldsymbol{y}}_{n-1}^{(\nu)} - \dot{\boldsymbol{y}}_{n-2}^{(\nu)} \tag{5.8}$$

which does not necessarily depend on the change in the error $\delta \dot{\boldsymbol{y}}^{(\nu)}$, but simply the ν^{th}-order derivative of the output $\dot{\boldsymbol{y}}^{(\nu)}$ since the desired value is constant across each update.

The Jacobian can therefore be updated simply by taking the previous update in the input estimate and considering the effect it has on Eq. (5.3). The work by Broyden [189] defines several of these types of Jacobian updates. These can either attempt to minimise the norm of Eq. (5.3) as much as possible within one step, or simply attempt to find an update that reduces the norm. To simplify the method, a unitary step size is assumed, as used in a traditional NR update. This will produce a norm-reducing update that has been shown to perform better than the norm minimisation alternative [189].

Additionally, there is the choice of a full-step Jacobian update or an incremental Jacobian update. It was found that, if a reasonably good initial estimate of the solution is available, then a full-step update performs best when compared to a traditional NR method [189]. This assumption of a good initial estimate being available is particularly true for the application in InvSim. If the desired outputs are smooth and continuous, then it can be assumed that the solution to the previous timestep will give a good initial estimate for the solution to the next timestep, as they should be relatively close together provided the discretisation step size T is not too large.

The smoothness of the desired output also suggests that the system will be moving smoothly through the state and output spaces, and therefore, the true Jacobian will also be changing smoothly. The initial estimate of the Jacobian $\mathbf{J}_0$ can, therefore, be taken as the last estimate from the previous timestep. It was also found that Broyden updates to the Jacobian could be avoided until the second iteration of the solver ($n = 2$) is reached. The logic is that if the current estimate of the Jacobian from the previous timestep is sufficiently accurate to converge to an input solution $\boldsymbol{u}[kT]$ which satisfies the tolerance $\dot{y}_{tol}^{(\nu)}$, then there is no immediate need to call the model to update it.

An initial estimate for the Jacobian at the first solver iteration of the first timestep of the InvSim will still need to be provided. This can be done through the usual numerical central differencing. When discontinuities exist in the system dynamics or the desired output, both Broyden's method and the traditional numerical differencing approaches may fail to converge. Instead, derivative-free numerical methods such as search-based algorithms can be employed [128]. However, for most applications of InvSim, the system dynamics and desired output can be assumed to be continuous. The full modified

Algorithm 4: Modified Broyden-based Inverse Simulation

Generate timeseries of desired output/output derivative states $\dot{\boldsymbol{y}}_d^{(\nu)}(t)$;
Define an initial input estimate for the first step $\boldsymbol{u}_0$;
for $k = 1 \rightarrow \texttt{length}(\dot{\boldsymbol{y}}_d^{(\nu)}[kT])$ **do**
  Sample desired output/output derivative $\dot{\boldsymbol{y}}_d^{(\nu)}[(k+1)T]$;
  Initialise input estimate $\boldsymbol{u}_0[kT] = \begin{cases} \boldsymbol{u}[(k-1)T] & \text{for } k > 1 \\ \boldsymbol{u}_0 & \text{for } k = 1 \end{cases}$;
  Initialise Jacobian estimate
  $\mathbf{J}_0[kT] = \begin{cases} \mathbf{J}[(k-1)T] & \text{for } k > 1 \\ \text{Central finite differencing (Eq. (5.5))} & \text{for } k = 1 \end{cases}$;
  Run forward simulation with initial input estimate $\dot{\boldsymbol{h}}^{(\nu)}(\boldsymbol{x}[kT], \boldsymbol{u}_0[kT])$;
  while $n < n_{max}$ **and** $\delta\dot{\boldsymbol{y}}^{(\nu)} > \delta\dot{y}_{tol}^{(\nu)}$ **do**
    Calculate error $\delta\dot{\boldsymbol{y}}^{(\nu)}$ (Eq. (5.3));
    if $n > 1$ **then**
      Update Jacobian $\mathbf{J}_n[kT]$ using Broyden estimate (Eq. (5.6));
    end
    Update input estimate $\boldsymbol{u}_n[kT] = \boldsymbol{u}_{n-1}[kT] - \mathbf{J}_n^{-1}\delta\dot{\boldsymbol{y}}^{(\nu)}$;
    Run forward simulation with updated input solution $\dot{\boldsymbol{h}}^{(\nu)}(\boldsymbol{x}[kT], \boldsymbol{u}_n[kT])$;
    Update error to check tolerance $\delta\dot{\boldsymbol{y}}^{(\nu)}$ (Eq. (5.3));
  end
  Save input solution $\boldsymbol{u}[kT] \leftarrow \boldsymbol{u}_n[kT]$;
end

InvSim algorithm utilising Broyden's Jacobian estimate can be seen detailed in Algorithm 4.

The developed Broyden-based InvSim method must now be compared to traditional finite differencing approaches to validate its improvement in computational performance, whilst ensuring that solver tolerances can be met.

5.2 | Comparison to Finite Differencing Methods

The speed of the modified InvSim method will be compared against computing the Jacobian directly through central, forward, and backward finite differencing techniques. Computational speed is quantified in terms of the number of times the model is integrated over the discretisation timestep T. For this purpose, explicit RK4 numerical integration (Appendix D.3) is used, as it provides a fixed quantity of computational time thanks to its closed-form (Section 3.2). This fixed quantity of time was used previously

in Section 2.2.2, and is referred to as a *model call* to avoid confusion with Section 3.2, where the number of times the ODEs of the model are evaluated (ODE evaluations).

InvSim has been applied to many real-world dynamic systems such as rotorcraft [124, 141], fixed-wing aircraft [131, 143], planetary rovers [153, 156], and spacecraft [137, 161]. Therefore, to ensure that the generality of the modified InvSim method is conveyed, 3 dynamic systems, including a satellite with flexible appendages and reaction wheels (Section 3.1), are used to compare computational speed. The other two systems are generic, used in a previous work that compared InvSim and NDI [158]. These two generic systems allow for easy tractability and comparison to previously published results [158]. Both systems have four states, two outputs, and two inputs. The first is a control affine system (linear in inputs) with state derivatives [158]:

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}) = \begin{bmatrix} x_2 x_3 - x_1^2 \\ x_3 x_4 - x_2^2 \\ (x_1 + 1)u_1 - 0.1 x_1 x_3 \\ (x_2 + 1)u_2 - 0.1 x_2 x_4 \end{bmatrix} \tag{5.9}$$

Modifying the first system slightly gives a control non-affine system (nonlinear in inputs) [158]:

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}) = \begin{bmatrix} x_2 x_3 - x_1^2 \\ x_3 x_4 - x_2^2 \\ (x_1 + 1)u_1 + 10\ \mathrm{sgn}(u_2)(x_2 + 2)u_2^2 - 0.1 x_1 x_3 \\ (x_2 + 1)u_2 + 10\ \mathrm{sgn}(u_1)(x_1 + 3)u_1^2 - 0.1 x_2 x_4 \end{bmatrix} \tag{5.10}$$

where $\mathrm{sgn}(\cdot)$ is the sign function. The output and output derivative equations for both systems are identical [158]:

$$\boldsymbol{y} = \boldsymbol{g}(\boldsymbol{x}) = \begin{bmatrix} x_1 x_2 \\ x_2 \end{bmatrix} \tag{5.11a}$$

$$\dot{\boldsymbol{y}} = \dot{\boldsymbol{g}}(\boldsymbol{x}, \dot{\boldsymbol{x}}) = \begin{bmatrix} x_1 \dot{x}_2 + \dot{x}_1 x_2 \\ \dot{x}_2 \end{bmatrix} \tag{5.11b}$$

Specific functions for the desired output $\boldsymbol{y}_d(t)$ is given [158]:

$$\boldsymbol{y}_d(t) = \begin{bmatrix} 5.94 \times 10^{-5} \\ 6.6 \times 10^{-5} \end{bmatrix} t^5 - \begin{bmatrix} 1.485 \times 10^{-3} \\ 1.65 \times 10^{-3} \end{bmatrix} t^4 + \begin{bmatrix} 9.9 \times 10^{-3} \\ 1.1 \times 10^{-2} \end{bmatrix} t^3 + \begin{bmatrix} 0.01 \\ 0.1 \end{bmatrix} \tag{5.12}$$

where $t \in [0, 30]$ seconds. However, throughout this chapter, the first-order ($\nu = 1$) derivative of the desired output is used and can be found through differentiation of Eq. (5.12).

For the satellite with flexible appendages model, defined in Section 3.1, the properties used previously in Table 3.1 are adopted. The satellite is required to follow a rest-to-rest, polynomial, 60 second, 180 deg attitude slew manoeuvre. As with previous chapters, this is parameterised using SLERPs, giving the desired output in terms of the vector component of a unit quaternion:

$$\overset{(\nu)}{\boldsymbol{y}}_d(t) = \overset{(\nu)}{\boldsymbol{q}}^{B/I}_{d,1:3}(t) \tag{5.13}$$

The InvSim hyperparameters used to solve each dynamic system are in Table 5.1 where N_u is the number of inputs to the system, which will vary for the flexible satellite system as the number of reaction wheels used changes. The minimum perturbation size Δu_{min} is scaled to the speed of the system dynamics to ensure perturbations in the inputs result in significant perturbations in the outputs, thus avoiding similar differencing issues.

Table 5.1: Comparison of finite differencing and Broyden's method: InvSim hyperparameters

Hyperparameter	**Control affine system**	**Control non-affine system**	**Flexible satellite system**
Discretisation timestep, T [s]	0.01	0.01	0.1
Integration step size, Δt [s]	0.001	0.001	0.005
NR tolerance, $\delta \dot{y}^{(\nu)}_{tol}$	1×10^{-13}	1×10^{-13}	1×10^{-16}
Maximum NR iterations, n_{max}	200	200	200
Minimum finite difference perturbation, Δu_{min}	0.01	0.001	1 Nm
Initial input estimate, $\boldsymbol{u}_0$	$\mathbf{0}^{2\times1}$	$\mathbf{0}^{2\times1}$	$\mathbf{0}^{N_u\times1}$
Desired output, $\dot{\boldsymbol{y}}^{(\nu)}_d$	Eq. (5.12)	Eq. (5.12)	$\overset{(\nu)}{\dot{\boldsymbol{q}}}^{B/I}_{d,1:3}$
Desired output order, ν	1 (velocity)	1 (velocity)	1 (velocity)

5.2.1 | Control Affine System

The results of running the InvSim integral method algorithm to target a prescribed output derivative for a control affine system using different solver methods can be seen in Figs. 5.2 and 5.3, and Table 5.2. Figure 5.2 shows the tracking of the desired outputs and the resultant input solution. Table 5.2 gives the mean number of iterations per timestep for each solver method, the maximum error in the output, and the total number of times the model is called and integrated across the discretisation timestep. The

total number of model calls during the full InvSim is used to give a quantitative metric for the speed of the InvSim. Figure 5.3 shows the solver iterations needed to converge at each timestep of the InvSim, the cumulative number of model calls, and the mean error in the Jacobian at each timestep $|\delta\bar{\mathbf{J}}_{\mathbf{n}}|_k$. Since central differencing, as will be seen in the remaining results, always provides fastest convergence, it is assumed to provide the most accurate Jacobian. Therefore, the errors in the Jacobian are calculated relative to central differencing.

Table 5.2: Comparison of Jacobian calculation methods: Control affine system

Case	Mean solver iterations	Mean $\|\|\delta \boldsymbol{jy}\|\|$	Total model calls
Central differencing	1.3	3.3×10^{-15}	23,005 (100%)
Forward differencing	1.3	8.1×10^{-15}	15,003 (65.2%)
Backward differencing	1.3	4.3×10^{-15}	15,003 (65.2%)
Broyden's method	2.0	9.5×10^{-14}	8,910 (38.7%)

All Jacobian calculation approaches provided a good InvSim solution, with the desired output being tracked closely (Fig. 5.2a), and a smooth input solution being found (Fig. 5.2b). Table 5.2 shows that each solver is capable of converging within the tolerance set in Table 5.1. Considering the mean solver iterations in Table 5.2, and looking at Fig. 5.3b, it can be seen that all the finite differencing methods converge to the solution at the same rate. However, it can also be seen that while the number of iterations needed to converge is identical, the number of model calls required by forward and backward differencing is reduced by approximately 35%.

Table 5.2 shows that Broyden's method, on average, requires more iterations to solve each timestep than the finite differencing approaches. However, Fig. 5.3b shows that, for $k > 1000$, the number of iterations needed to converge to the solution reduces to one for all solver methods, corresponding with the system reaching a steady state (Fig. 5.2). Therefore, the InvSim solution at each timestep $k > 1000$ is simply identical to the solution at the previous timestep. This causes each solver to converge instantaneously, with the initial estimate of the input solution meeting the prescribed tolerance.

Looking at Fig. 5.3b before the steady state is reached, it can be seen that Broyden's method requires more iterations than every other method to converge. This corresponds with a larger error in the Jacobian when compared to the forward and backward differencing methods Fig. 5.3a. While Broyden's method requires more iterations to converge to a solution, this is offset substantially by requiring fewer model calls to obtain the Jacobian, making it the fastest solver method overall. When compared to the central differencing method, Brodyen's method provides an approximate 61% speed increase (Table 5.2).

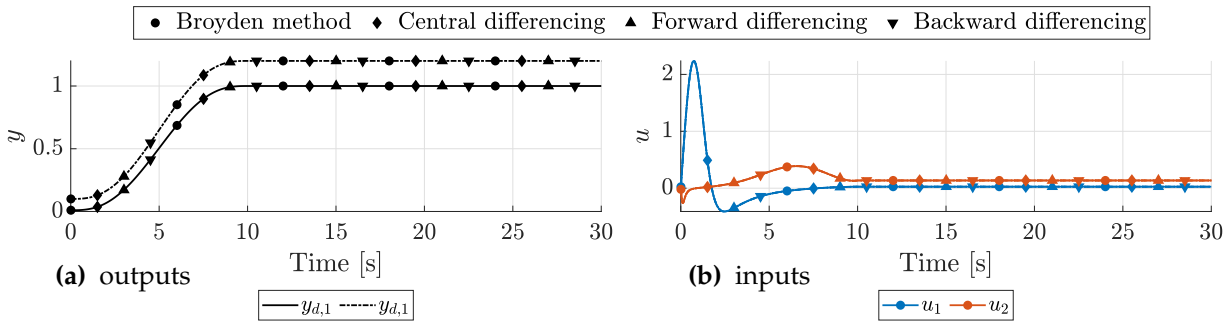
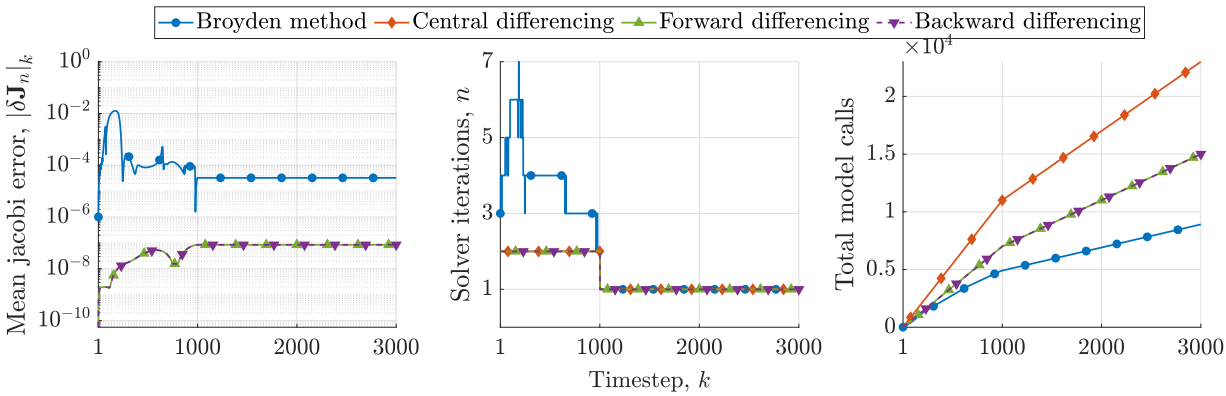


Figure 5.2: Control affine system InvSim solution

However, it could be argued that the InvSim should be terminated as soon as the steady state is reached ($k = 1000$), as beyond this point the input does not change. It can be seen in Fig. 5.3c that even at this point, Broyden's method has still used fewer model calls than the other methods, with a 55% speed increase when compared to central differencing.



(a) Mean Jacobian error at each timestep

(b) solver iterations for each timestep

(c) total cumulative model calls

Figure 5.3: Control affine system InvSim convergence

5.2.2 | Control Non-affine System

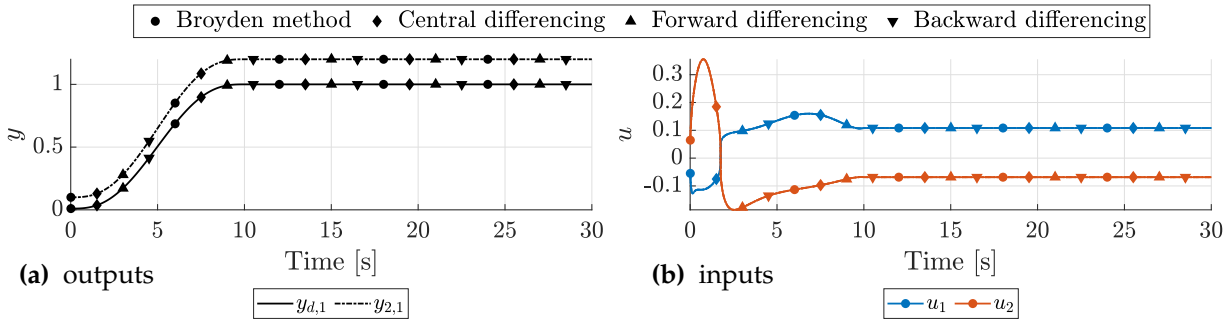
The InvSim solution for the control non-affine system can be found in Fig. 5.4. When comparing the total model calls and mean solver iterations of the non-affine system (Table 5.3) to the affine system (Table 5.2), a marked increase can be observed for each solver. This is due to the added nonlinear dynamics from the non-affine controls. It can intuitively be understood that any gradient-based solver such as NR or Broyden's method will require more iterations to converge to a solution, as Eq. (5.3) becomes more nonlinear.

With the additional nonlinear dynamics, the forward and backward finite differencing approaches produce an InvSim that requires more solver iterations to converge than both central differencing and Broyden's method (Table 5.3 & Fig. 5.5b). Examining

Table 5.3: Comparison of Jacobian calculation methods: Control-nonaffine system

Case	Mean solver iterations	Mean $\|\|\delta \dot{\boldsymbol{y}}\|\|$	Total model calls
Central differencing	1.7	9.6×10^{-14}	28,150 (100%)
Forward differencing	3.7	9.7×10^{-14}	36,504 (130%)
Backward differencing	3.8	9.4×10^{-14}	36,927 (131%)
Broyden's method	2.1	9.96×10^{-14}	9,387 (33.3%)

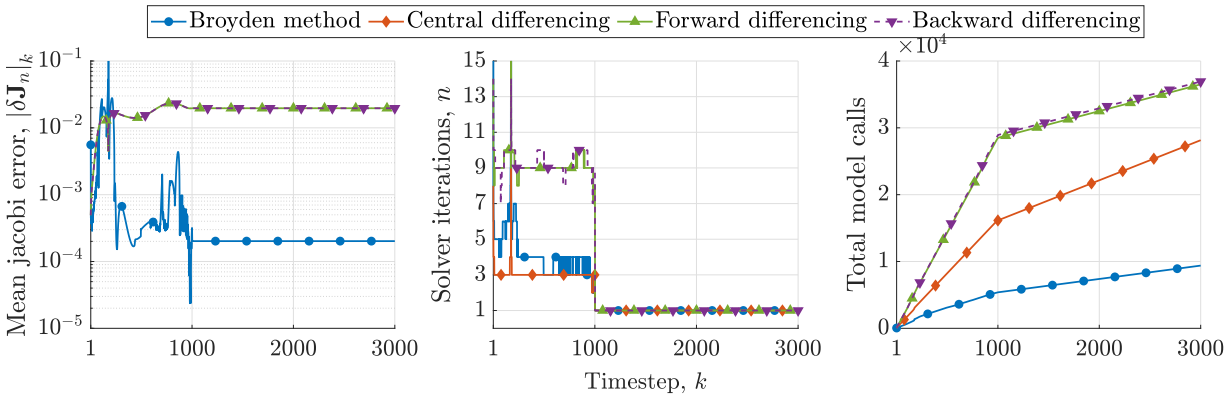
Fig. 5.5a, the forward and backward differencing methods produce Jacobians that are less accurate by several orders of magnitude when compared to the control affine case (Fig. 5.3a). Looking at Fig. 5.5a and Fig. 5.5b, a correlation can be seen between the number of solver iterations and the accuracy of the Jacobian, with Broyden's method obtaining a more accurate Jacobian, and thus fewer solver iterations than forward and backward differencing.



(a) outputs **(b)** inputs

Figure 5.4: Control non-affine system InvSim solution

Overall, even with the additional control non-affine dynamics, Broyden's method demonstrates the best performance with an approximate 67% speed increase (compared to central differencing) over the full InvSim, and a similar speed increase when the steady state is reached ($k = 1000$).



(a) Mean Jacobian error at each timestep **(b)** solver iterations for each timestep **(c)** total cumulative model calls

Figure 5.5: Control non-affine system InvSim convergence

5.2.3 | Flexible Satellite System

Results are shown in Table 5.4 for a flexible satellite with three and four reaction wheels, with the inputs and output responses seen in Fig. 5.6, and the convergence of the InvSims seen in Fig. 5.7. Overall trends are similar to the control affine system, with forward and backward differencing improving computational efficiency when compared to central differencing. Broyden's method again provides the fastest overall performance, despite requiring more solver iterations to converge.

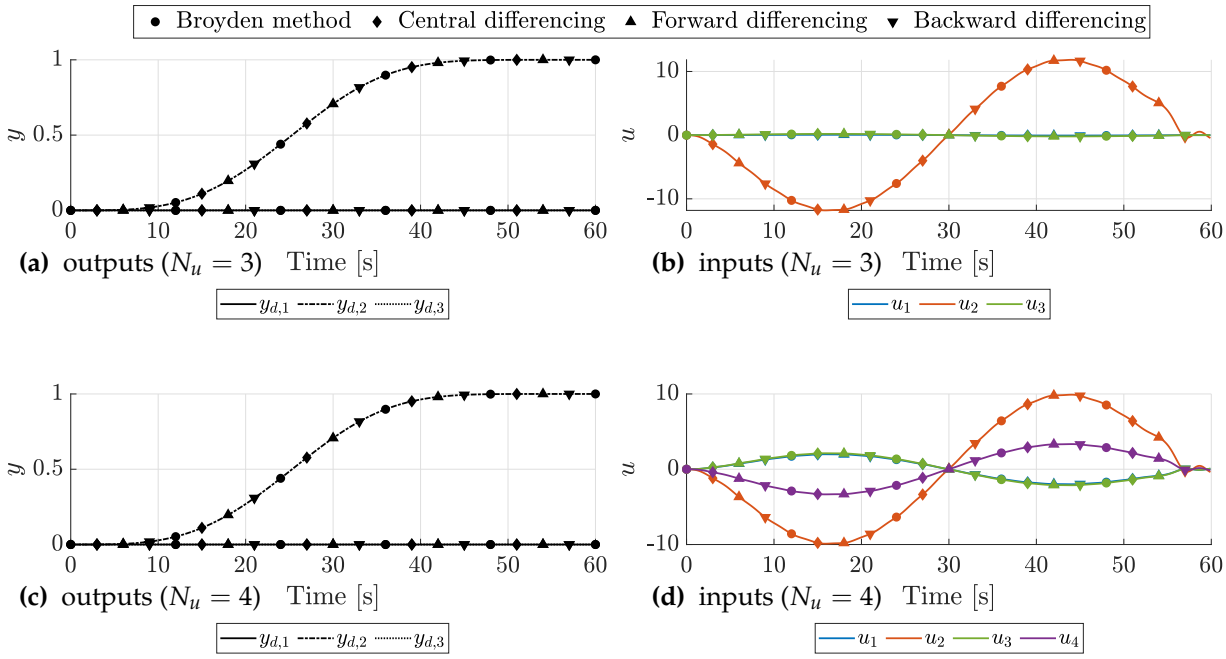
Table 5.4: Comparison of Jacobian calculation methods: Flexible satellite system

Case	Mean solver iterations	Mean $\|\|\delta\dot{\boldsymbol{y}}\|\|$	Total model calls
	3 reaction wheels, $N_u = 3$		
Central differencing	2.0	6.7×10^{-17}	8,930 (100%)
Forward differencing	2.7	9.8×10^{-17}	7,084 (79.3%)
Backward differencing	2.7	9.7×10^{-17}	7,080 (79.3%)
Broyden's method	5.2	9.998×10^{-17}	3,731* (41.7%)
	4 reaction wheels, $N_u = 4$		
Central differencing	2.0	6.6×10^{-17}	11,310 (100%)
Forward differencing	2.6	9.7×10^{-17}	8,420 (74.5%)
Backward differencing	2.6	9.9×10^{-17}	8,485 (75.0%)
Broyden's method	5.2	9.9×10^{-17}	3,753 (33.2%)

* Value erroneously reads 3,751 in [162]

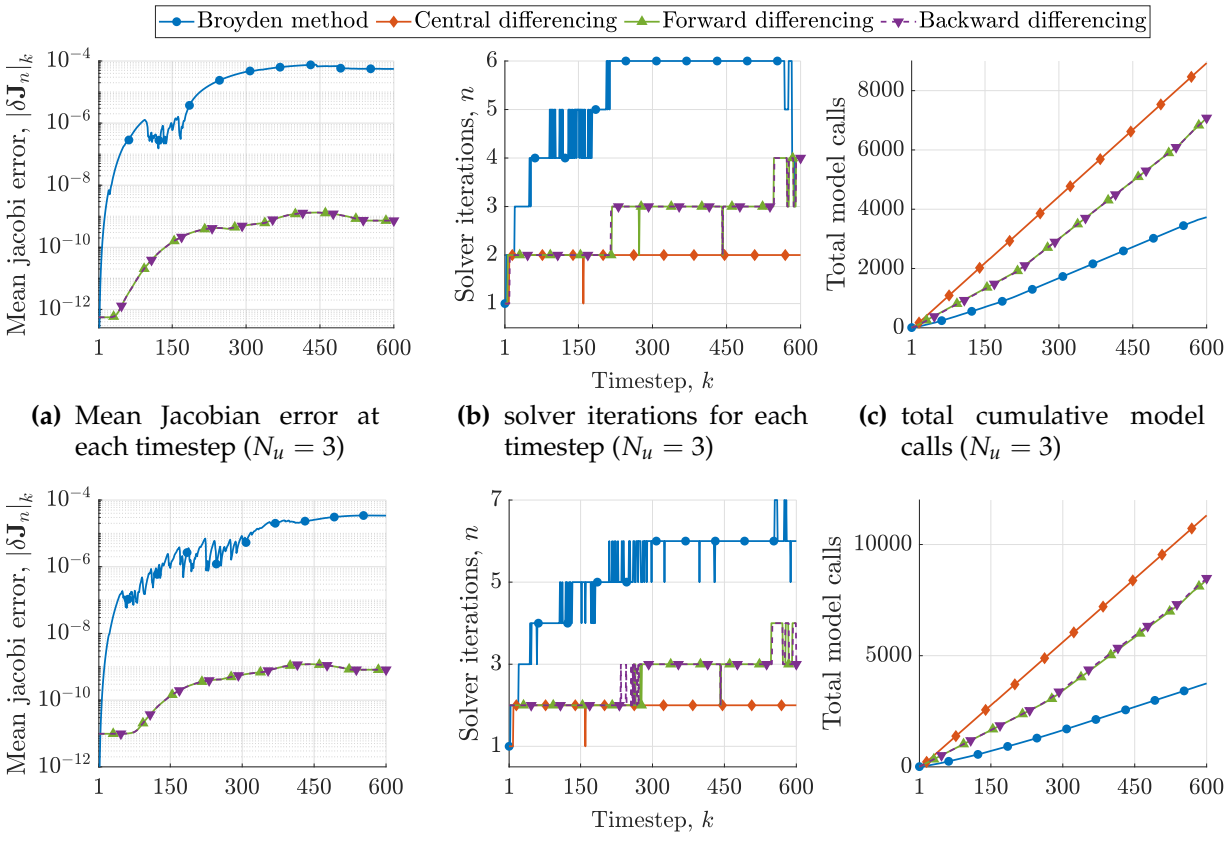
Broyden's method also maintains comparable performance with the addition of a fourth input, while finite differencing methods experience a substantial increase in total model calls (Table 5.4). With three inputs, Broyden's method shows a 58% speed increase over central differencing, improving further to 67% with four inputs. The performance advantage of Broyden's method grows as the number of inputs increases.

In Fig. 5.7, the Broyden Jacobian is less accurate than forward and backward differencing, similar to the control affine case. The error in the Broyden Jacobian starts small, diverges, and settles at around 10^{-4}; comparable to both the control affine (Fig. 5.3a), and non-affine systems (Fig. 5.5a). Broyden's method initialises the Jacobian using central differencing, starting with zero error. However, this highly accurate initialisation is evaluated around the initial input estimate (Table 5.1) and the system's initial state. For control affine and non-affine systems, the solution to the first timestep deviates from the initial estimate considerably, as can be seen in Fig. 5.3a. Consequently, the initialised Jacobian loses accuracy quickly, whereas the Broyden Jacobian error peaks initially and eventually settles at 10^{-4}. In the case of the flexible satellite, the solution at the first timestep is very close to the initial estimate, therefore, the Broyden Jacobain remains



(a) outputs ($N_u = 3$) **(b)** inputs ($N_u = 3$) **(c)** outputs ($N_u = 4$) **(d)** inputs ($N_u = 4$)

Figure 5.6: Flexible satellite system InvSim solutions



(a) Mean Jacobian error at each timestep ($N_u = 3$) **(b)** solver iterations for each timestep ($N_u = 3$) **(c)** total cumulative model calls ($N_u = 3$)

(d) Mean Jacobian error at each timestep ($N_u = 4$) **(e)** solver iterations for each timestep ($N_u = 4$) **(f)** total cumulative model calls ($N_u = 4$)

Figure 5.7: Flexible satellite system InvSim convergence

accurate for longer before diverging and coming to settle at around 10^{-4} (Fig. 5.7a). It is interesting to note that the behaviour of the Broyden Jacobian error with a good initial estimate resembles that of a first-order or critically damped second-order system. Conversely, with a poor initial estimate, the Broyden Jacobian error behaves like an underdamped second-order system.

The numerical performance of the developed Broyden-based InvSim method has been evaluated, with its superior computational efficiency demonstrated for a selection of dynamic systems. Next, the numerical stability of the new proposed method is investigated to ensure it remains robust across as many possible applications.

5.3 | Issues with Jacobian Null Space

It has been discussed previously in Chapters 2 and 4 that there are some numerical stability issues associated when the Jacobian used during the InvSim process has a null space, i.e. is singular. In most dynamic systems, this issue does not arise; however, in the case of magnetic-only control of a satellite, controllability in the direction of the local magnetic field $\hat{\boldsymbol{b}}^B_{mag}$ is lost. The use of the MP pseudoinverse (Eq. (A.19)) was shown to provide a sufficient InvSim solution, even in the case where a non-invertible singular Jacobian is present, provided the manoeuvre is physically realisable, i.e. not requiring torques in the direction of $\hat{\boldsymbol{b}}^B_{mag}$. It was further shown that loosening the output constraints in the direction of lost control could alleviate this numerical instability. It will be shown in this section, however, that this is no longer sufficient when using Broyden's method to provide faster InvSim.

5.3.1 | Orthogonality of the Broyden Update

When updating the Jacobian using Eq. (5.6) the only gradient information available for $\dot{\boldsymbol{h}}^{(\nu)}$ is in the direction of the previous input update $\delta \boldsymbol{u}_n$. This defines a direction $\hat{\boldsymbol{v}}$ orthogonal to $\delta \boldsymbol{u}_n$:

$$\hat{\boldsymbol{v}}^\top \delta \boldsymbol{u}_n = 0 \tag{5.14}$$

in which no new gradient information is available, thus:

$$\mathbf{J}_{n+1}\hat{\boldsymbol{v}} = \mathbf{J}_n \hat{\boldsymbol{v}} \tag{5.15}$$

This effectively means that the Jacobian is blind in the $\hat{\boldsymbol{v}}$ direction, simply assuming that gradients in that direction are the same from one iteration (n) to another ($n+1$). As has been seen in Section 5.2, this orthogonality of Broyden's method does not cause any numerical stability issues, with satisfactory InvSim solutions being converged upon.

However, in the case that the Jacobian itself has a null space, as is the case with magnetic-only control of a satellite, this can have significant numerical stability issues.

Table 5.5: Parameters of a rigid-body satellite with magnetic-only control test case

Parameter	Value
Inertia, $\mathbf{I}_{tot}$	diag(100,100,100) kgm^2
Magnetic field strength, $\|\|\boldsymbol{b}^I_{mag}\|\|$	25.9 µT
Magnetic field inertial direction, $\hat{\boldsymbol{b}}^I_{mag}$	$[0\ 0\ 1]^\top$
Initial quaternion, $\boldsymbol{q}^{B/I}_i$	$[1\ 0\ 0\ 0]^\top$
Final quaternion, $\boldsymbol{q}^{B/I}_f$	$[\frac{1}{\sqrt{2}}\ 0\ \frac{1}{\sqrt{2}}\ 0]^\top$
Slew duration, t_f	60 s

To illustrate, consider a rigid-body satellite with magnetorquers oriented in all three body axes (Appendix E.2) as a test case with parameters given in Table 5.5. The satellite is tasked with performing a 90 deg, 60 second, polynomial, SLERP slew manoeuvre about the positive body $\hat{\boldsymbol{y}}^B$ axis. The inertia matrix $\mathbf{I}_{tot}$ has no products of inertia (i.e. diagonal), and the magnetic field is orientated in the inertial $\hat{\boldsymbol{z}}^I$ direction. The direction of the magnetic field in body axes $\hat{\boldsymbol{b}}^B_{mag}$ during the prescribed slew manoeuvre is shown in Fig. 5.8. It can be seen that there should be no loss of control about the slewing axis, and therefore the prescribed slew manoeuvre is physically realisable without any loosening of the constraints. The InvSim hyperparameters for this section are given in Table 5.6.

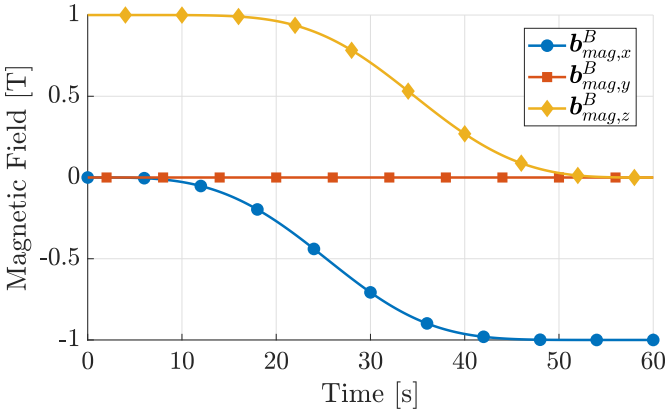


Figure 5.8: Magnetic field direction in body axes during 90-deg slew manoeuvre about the positive $\hat{\boldsymbol{y}}^B$ axis

Using the modified InvSim method developed in Section 5.1.1, and the traditional finite differencing-based InvSim methods, the input solutions in Fig. 5.9 are converged upon. It can be seen that the use of Brodyen's method results in the u_x magnetic dipole exponentially increasing and diverging from the other solutions for most of the manoeuvre.

Magnetic dipole in the u_z direction also fails to follow the other solutions and remains stuck at zero.

Table 5.6: Issues of Jacobian null space (Rigid-body with magnetic torques): InvSim hyperparameters

Hyperparameter	Value
Discretisation timestep, T	0.1 s
Integration step size, Δt	0.01 s
NR tolerance, $\delta \dot{y}_{tol}^{(\nu)}$	1×10^{-16}
Maximum NR iterations, n_{max}	200
Minimum finite difference perturbation, Δu_{min}	0.1 Am^2
Initial input estimate, $\boldsymbol{u}_0$	$\mathbf{0}^{3\times 1}$
Desired output order, ν	1 (velocity)
Desired output, $\dot{\boldsymbol{y}}_d^{(\nu)}$	$\dot{\boldsymbol{q}}_{d,1:3}^{(\nu)B/I}$

The modified, Broyden-based InvSim method initialises the Jacobian on the very first iteration using traditional central numerical differencing. This initial Jacobian is accurate, correctly including the null space in the $\hat{\boldsymbol{z}}^B$ direction as illustrated in Fig. 5.8. The numerical issue arises through the orthogonality of the Broyden update. It is already understood that updates to the input $\delta \boldsymbol{u}_n$ will be physically limited to the plane orthogonal to $\hat{\boldsymbol{b}}_{mag}^B$ and the corresponding null space of the Jacobian. Therefore, whilst the null space of the Jacobian is correctly orientated in the $\hat{\boldsymbol{z}}^B$ direction at the start of the InvSim, it will remain stuck there, failing to update and reflect the changing local magnetic field as the satellite rotates (Fig. 5.8). This explains why the u_z magnetic dipole remains stuck at zero, as the estimated Jacobian of the InvSim still determines that the null space, i.e. loss of actuation, is in the $\hat{\boldsymbol{z}}^B$ direction as it was at $t = 0$. Furthermore, the Jacobian continues to assume that only inputs in the u_x direction will impact the slew manoeuvre about the body $\hat{\boldsymbol{y}}^B$ axis.

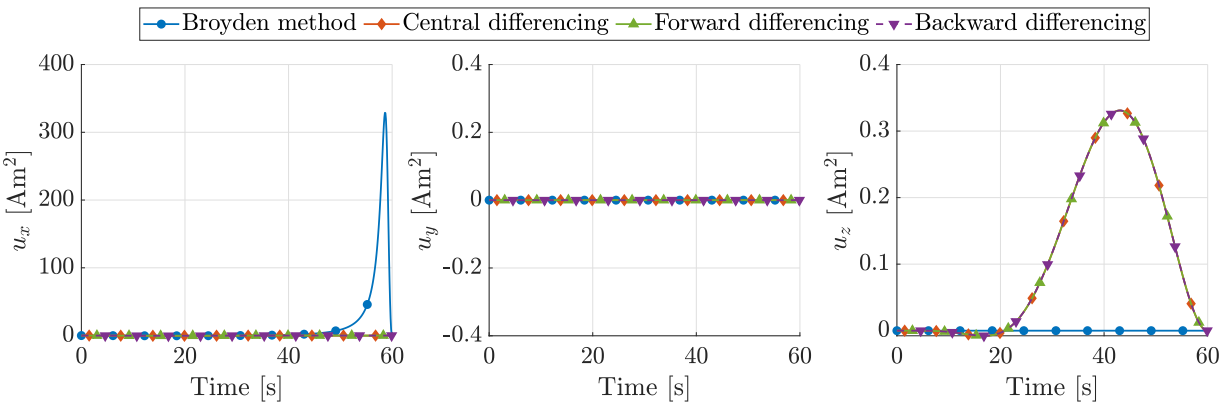


Figure 5.9: Rigid-body satellite with magnetic torque InvSim control solution

5.3.2 | Robust Broyden's Method for Faster Inverse Simulation

To improve the robustness of the modified InvSim method detailed in Section 5.1.1 and ensure that an adequate solution is converged upon, further modifications to the method are required. One simple approach may be to simply schedule a *full* finite difference update of the Jacobian at regular intervals during the InvSim process. This would ensure that the null space of the Jacobian updates in the correct direction at certain points in the InvSim and would hopefully avoid large divergences from the ideal input solution. This approach would, however, diminish much of the speed improvement achieved by the modified InvSim algorithm (Section 5.2), as each full Jacobian update would require $2N_u$ or N_u model calls, depending on which finite differencing method is used. Selecting the rate at which this full update occurs could allow for a balance between speed and accuracy to be struck, however, this is still far from ideal.

An alternative approach is to detect if a null space $\hat{\boldsymbol{v}}$ exists in the Jacobian, apply a single perturbation in that direction, and use the additional gradient information in the $\hat{\boldsymbol{v}}$ direction to perform an additional Broyden update to the Jacobian. The idea is to let the Jacobian *see* gradient information in the direction that it is currently *blind* in to ensure that it can *see* when the null space changes.

Detection of the existence of an uncontrollable null space $\hat{\boldsymbol{v}}$ can be achieved by simply considering the *rank* of the Jacobian. If $\texttt{rank}(\mathbf{J}_n) < N_y$, where N_y is the number of outputs of the system (also the number of rows in the Jacobian), then there could exist an uncontrollable null space $\hat{\boldsymbol{v}}$. Note that this process is synonymous with taking the row rank of the controllability matrix of an LTI system. The null space $\hat{\boldsymbol{v}}$ can then be found by utilising a singular value decomposition of $\mathbf{J}_n$, available in *MATLAB* using the `null()` function. An additional input perturbation $\Delta\boldsymbol{u}_v$ can be generated in the null space direction:

$$\Delta\boldsymbol{u}_v = T||\boldsymbol{u}_n||\hat{\boldsymbol{v}} \tag{5.16}$$

where $||\boldsymbol{u}_n||$ gives the magnitude of the current input estimate, scaled by the discretisation timestep size T, in the direction of the null space $\hat{\boldsymbol{v}}$. Integrating the model forward with this perturbation, and applying Eqs. (5.6) to (5.8) to update the Jacobian $\mathbf{J}_n$ allows for the null space to be updated and better represent the current state of the system. This allows for the changing local magnetic field direction in Fig. 5.9 to be taken into account, ensuring that a realisable input solution is converged upon. The modified InvSim algorithm utilising Brodyen's Jacobian estimate, with additional null space compensation for *robustness*, can be seen in full detail in Algorithm 5. The difference between the new *robust* algorithm (Algorithm 5) can be seen by comparing lines 10-17 with lines 10-12 in the original Broyden-based algorithm (Algorithm 4).

Algorithm 5: Robust Modified Broyden-based Inverse Simulation

Generate timeseries of desired output/output derivative states $\dot{\boldsymbol{y}}_d^{(\nu)}(t)$;
Define an initial input estimate for the first step $\boldsymbol{u}_0$;
for $k = 1 \rightarrow \texttt{length}(\dot{\boldsymbol{y}}_d^{(\nu)}[kT])$ **do**
  Sample desired output/output derivative $\dot{\boldsymbol{y}}_d^{(\nu)}[(k+1)T]$;
  Initialise input estimate $\boldsymbol{u}_0[kT] = \begin{cases} \boldsymbol{u}[(k-1)T] & \text{for } k > 1 \\ \boldsymbol{u}_0 & \text{for } k = 1 \end{cases}$;
  Initialise Jacobian estimate
  $\mathbf{J}_0[kT] = \begin{cases} \mathbf{J}[(k-1)T] & \text{for } k > 1 \\ \text{Central finite differencing (Eq. (5.5))} & \text{for } k = 1 \end{cases}$;
  Run forward simulation with initial input estimate $\dot{\boldsymbol{h}}^{(\nu)}(\boldsymbol{x}[kT], \boldsymbol{u}_0[kT])$;
  while $n < n_{max}$ **and** $\delta\dot{\boldsymbol{y}}^{(\nu)} > \delta\dot{y}_{tol}^{(\nu)}$ **do**
    Calculate error $\delta\dot{\boldsymbol{y}}^{(\nu)}$ (Eq. (5.3));
    if $n > 1$ **then**
      Use previous input update to update Jacobian $\mathbf{J}_n[kT]$ using Broyden estimate (Eq. (5.6));
      if null space $\hat{\boldsymbol{v}}$ exists **then**
        Run forward simulation with null space perturbation (Eq. (5.16)) $\dot{\boldsymbol{h}}^{(\nu)}(\boldsymbol{x}[kT], \boldsymbol{u}_n[kT] + \Delta\boldsymbol{u}_v)$;
        Calculate error $\delta\dot{\boldsymbol{y}}^{(\nu)}$ from perturbed output (Eq. (5.3));
        Update null space using Broyden Jacobian estimate update $\mathbf{J}_n[kT]$ (Eq. (5.6));
      end
    end
    Update input estimate $\boldsymbol{u}_n[kT] = \boldsymbol{u}_{n-1}[kT] - \mathbf{J}_{\boldsymbol{n}}^{-1}\delta\dot{\boldsymbol{y}}^{(\nu)}$;
    Run forward simulation with updated input solution $\dot{\boldsymbol{h}}^{(\nu)}(\boldsymbol{x}[kT], \boldsymbol{u}_n[kT])$;
    Update error to check tolerance $\delta\dot{\boldsymbol{y}}^{(\nu)}$ (Eq. (5.3));
  end
  Save input solution $\boldsymbol{u}[kT] \leftarrow \boldsymbol{u}_n[kT]$;
end

5.3.3 | Performance Comparison

To demonstrate the improved robustness of the modified Broyden-based InvSim method, the rigid-body satellite model with magnetic torques used in Section 5.3.1 is once again adopted with the model and test case parameters given in Table 5.5. The InvSim hyperparameters used are given previously in Table 5.6.

The InvSim input solutions using the improved algorithm in Algorithm 5 and the traditional finite differencing approaches are given in Fig. 5.10. The input solution obtained using the robust Broyden method now converges to the same input solution found using a Jacobian calculated through finite differencing. The instability issues associated with the null space have been compensated for, with the additional Broyden update allowing for the changing local magnetic field direction to be taken into account.

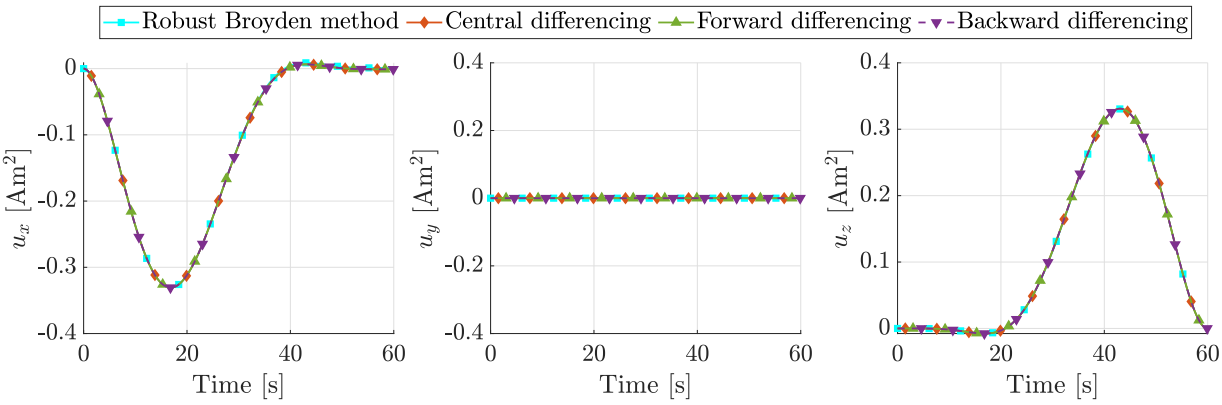


Figure 5.10: Rigid-body satellite with magnetic torque InvSim control solution with null space compensation

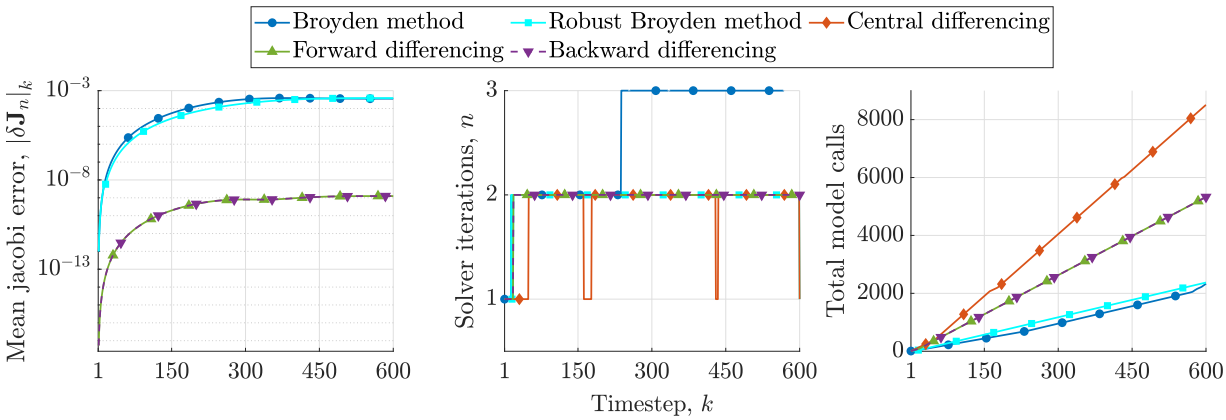
The computational speed of robust Broyden's method (Algorithm 5), Broyden's method (Algorithm 4), and the traditional finite differencing techniques are compared in Table 5.7 and Fig. 5.11. It can be seen that the trends are similar to those for the control affine and flexible satellite systems (Section 5.2). Forward and backward finite differencing improve the computational efficiency somewhat, however, both Broyden-based methods achieve the best performance, with a similar 72-73% speed increase when compared to traditional central differencing (Table 5.7). However, as shown previously in Fig. 5.9, the original Broyden-based InvSim algorithm (Algorithm 4) failed to converge to an ideal solution due to orthogonality of the Broyden update. Therefore, it can be seen that the robust Broyden InvSim algorithm (Algorithm 5) was capable of converging to a satisfactory input solution (Fig. 5.10), at the cost of a minimal reduction in computational speed, $\approx$0.5% when compared to central differencing.

The similarity in speed of the Broyden and robust Broyden methods is, at first, surprising since the robust method requires additional model calls to provide the gradient

Table 5.7: Comparison of Jacobian calculation methods: Rigid-body satellite with magnetic torque

Case	Mean solver iterations	Mean $\|\|\delta \boldsymbol{y}^{(\nu)}\|\|$	Total model calls
Central differencing	1.9	4.9×10^{-18}	8,503 (100%)
Forward differencing	2.0	3.8×10^{-18}	5,328 (62.7%)
Backward differencing	2.0	3.9×10^{-18}	5,328 (62.7%)
Broyden's method (Algorithm 4)	2.9	7.7×10^{-18}	2,334 (27.4%)
Robust Broyden's method (Algorithm 5)	2.0	1.6×10^{-17}	2,372 (27.9%)

information in the null space direction $\hat{\boldsymbol{v}}$. Considering Fig. 5.11a, it can be seen that the robust Broyden method produces, on average, a more accurate Jacobian thanks to its additional update in the null space direction $\hat{\boldsymbol{v}}$. This, as can be seen in Table 5.7 and Fig. 5.11b, leads to a reduction in the mean solver iterations for the robust Broyden method when compared to the original. The model calls required by the additional solver iterations of the original Broyden method (Algorithm 4) are, therefore, balanced by the model calls made by the robust Broyden method (Algorithm 5) when performing the additional null space Broyden update.



(a) Mean Jacobian error at each timestep

(b) solver iterations for each timestep

(c) total cumulative model calls

Figure 5.11: Rigid-body satellite with magnetic torque InvSim convergence with null space compensation

The robust Broyden InvSim algorithm (Algorithm 5) should therefore be adopted over the original (Algorithm 4) presented in Section 5.1.1 and published in Ref. [162]. In the case where no null space exists in the Jacobian, the robust method will effectively act identically to the original, not performing any additional Broyden updates. Therefore, the results and analysis presented in Section 5.2 can additionally be considered illustrative of the performance of the robust Broyden InvSim algorithm (Algorithm 5), and confirm that performance is consistent across multiple dynamic systems.

5.4 Summary

This chapter developed a new, novel, faster InvSim method through a simple, yet effective modification of the existing integral-based approach, only requiring a few lines of code in existing integral-based InvSim implementations to be changed. The modification is particularly effective in mitigating the degradation in computational speed that occurs when a large number of inputs/actuators are included, as seen during the design study in Chapter 4. However, the improved computational speed also has important implications when considering the implementation of an InvSim-based attitude controller, as developed in Chapter 3, on real-time hardware. The new method was tested on a satellite with flexible appendages, in addition to two arbitrary dynamics systems to demonstrate the wider applicability of the faster InvSim method.

The proposed method replaced the usual finite differencing used to calculate the Jacobian with a *Broyden* Jacobi estimation. Compared to central differencing, the new Broyden-based InvSim method improved computational speed by 58-67%. The additional iterations of the NR solver, caused by inaccuracies in the estimated Jacobian, are offset by only requiring one evaluation of the model per NR iteration. The relative numerical speed advantage of the new method increases as the number of inputs to the system increases, owing to the additional perturbations and model evaluations required by the finite differencing methods to populate the larger Jacobian.

The use of magnetic-only control was found to necessitate an additional modification to the Boyden-based InvSim method, mitigating instabilities caused by the Jacobian's null space. Despite the Jacobian null space requiring an additional model evaluation, the robust Broyden-based InvSim method still proves to be significantly faster than finite differencing; 72% faster than central differencing when solving the model of a rigid body satellite with magnetic-only control.

It can be seen, therefore, that the novel robust Broyden-based method provides a significant improvement in computational speed to integral-based InvSim, while still meeting strict error tolerances. The improved method would allow for the design investigation in Chapter 4 to evaluate configurations with large numbers of actuators within a reasonable amount of time, and improve the feasibility of implementing the InvSim-based hybrid attitude control architecture and path planning algorithm developed in Chapter 3 on real-time hardware.

Chapter 6 | Conclusions and Future Work

THIS thesis investigated the application of numerical Inverse Simulation (InvSim) techniques to the attitude control and design of satellites with flexible structures. Chapter 1 presented the background and motivation for the thesis, demonstrating the importance of accurate attitude tracking for various mission scenarios, illustrating the current and future uses of flexible structures in the construction of satellites, and highlighting problematic flexible dynamic behaviours that must be considered. Relevant historical and state-of-the-art literature was then presented, focusing on the potential use of InvSim as an attitude control method, which informed the central research question for this thesis:

Can Inverse Simulation improve the attitude control and design process of satellites with flexible structures, when compared to traditional techniques?

This chapter of the thesis will now summarise the thesis, considering whether this research question has been answered by the work presented in Chapters 2 to 5. Suggestions for potential future work will then be outlined before some final remarks are made.

6.1 | Conclusions

The research question for this thesis was developed in Section 1.3 into three main aims, which were in turn distilled into ten objectives to inform the direction of the thesis. This section of this chapter will consider each of the three main aims and how they contribute to answering the central research question.

6.1.1 | Aim 1

The first aim of this thesis was *to develop novel InvSim methodologies to investigate the attitude dynamics and control of satellites with flexible structures*. Completion of this aim indirectly contributes to answering the research question, as it uncovers specific problems

and considerations surrounding the numerical InvSim methods being employed, giving important context to the conclusions arrived at during completion of the remaining two aims.

It was demonstrated in Chapter 2 that the integral InvSim method was a highly practical, flexible and easy-to-implement methodology for solving the inverse dynamics of rigid body satellites with various attitude control actuators. Specific issues of numerical stability relating to magnetic-only control were discovered, some of which could be mitigated through careful selection of the initial input estimate, or by replacing the usual NR solver with a local optimisation method such as SQP, as demonstrated in Chapter 4. It was shown that the generality of the integral InvSim algorithm, while computationally more expensive, provides a more practical and accurate model-inversion technique than traditional analytical methods such as NDI. This allowed for the increasingly complex mathematical models derived throughout this thesis to be inverted without any modifications to the InvSim process itself.

A consideration specific to the application of InvSim to satellites with flexible structures is the relative stiffness of the dynamics. It was found in Chapter 3 that when only moderately stiff dynamics are exhibited, then explicit RK4 numerical integration provides the fastest InvSim. However, stiffer systems were found to require implicit numerical integration methods, such as implicit trapezoidal integration, to achieve a fast and stable InvSim solution. It was also demonstrated in Chapter 3 that the integral InvSim algorithm can be implemented within an existing simulation framework, but requires suitable interfacing with the forward simulation to work. Provided that this criterion is met, the integral InvSim algorithm can provide inversion of pre-existing, validated mathematical models, avoiding the need to recreate them in some other software environment such as *MATLAB* or *Python*.

In Chapter 4, it was concluded that replacing the NR solver with SQP was required to ensure convergence to a suitable InvSim solution when combinations of different actuator types are used together. The final contribution to the first aim was made in Chapter 5, which developed a faster InvSim methodology based on Broyden's method. It was concluded that this method could outperform all finite differencing approaches, and with additional modification, can be made robust to systems that exhibit partial loss-of-control, such as magnetic-only control. The method is particularly beneficial when the number of inputs to the system is large, ensuring a solution is converged upon within a reasonable amount of time.

Overall, it can be seen that the first aim was achieved, with contributions made across the thesis. It was shown that while the basic form of the integral InvSim algorithm can

be applied to the attitude dynamics of satellites with flexible structures, particular considerations in terms of computational efficiency and stability were required to account for the stiffness of the flexible dynamics, and satellites actuated with magnetic control only. Additionally, an improvement in numerical speed of the InvSim algorithm could be achieved using the developed robust Broyden-based InvSim method, particularly for systems with a large number of inputs. The improvement in computational speed would prove significantly advantageous when performing an extensive iterative design study, where the dynamics of different satellite designs must be evaluated promptly. Faster InvSim is also essential if the InvSim-based hybrid attitude control architecture, and path planning algorithm developed in this thesis are to be implemented on real-time hardware.

It can therefore be concluded that the developed InvSim methodologies presented in this thesis can provide a suitable numerical process for solving the inverse dynamics of satellites with flexible structures. The benefits of utilising InvSim for attitude control and design purposes have, however, yet to be established, but will be tackled by the remaining two aims.

6.1.2 | Aim 2

The second aim was *to develop an InvSim-based attitude control architecture, assessing its performance and practicality*. This aim was achieved for the most part within Chapter 3, which included the successful implementation of an InvSim-based feedforward term within a hybrid attitude control architecture.

When compared to a traditional nonlinear PD tracking controller, it was found that the addition of the InvSim feedforward term could improve attitude-tracking performance using only a low-fidelity model of the flexible dynamics, but was highly sensitive to uncertainties in the model. For example, uncertainties in the inertia and first modal frequency are required to be significantly small, $<1\%$ and < 0.5 rad/s respectively, to achieve improved pointing accuracy. This limitation is not unique to InvSim and has been observed by Iannelli et al. [107], who utilise model-predictive control (MPC) for attitude control of a flexible satellite. It is unlikely that these strict tolerances on model uncertainty can be achieved, particularly when considering that environmental effects will, over time, cause the dynamic behaviour of the system to change. Therefore, it can be concluded that to attain any meaningful and realistic improvement in tracking performance, the addition of an adaptive parameter estimation system is required to update the InvSim's model.

Deployment of the hybrid controller on real-time hardware is also limited by the high computational complexity of InvSim, but can be mitigated through the use of the faster InvSim method proposed in Chapter 5, or using techniques such as automatic differentiation [139] and machine learning [137]. Alternatively, as is suggested in Chapter 3, the InvSim solution could simply be computed offline before the manoeuvre is executed. The offline InvSim solution could then be used as the nominal control commands to be sent to the actuators, with an online PD element giving smaller commands to deal with disturbances and uncertainty in the model.

While it has been concluded that practical use of the proposed hybrid controller is somewhat limited, the calculated InvSim control solution could still be used as a preliminary control design. It would provide a temporary control solution that is readily available, and does not require retuning as the design of the system changes rapidly during early development. Then, once the final design is settled upon, a bespoke controller can be produced, with InvSim providing useful performance benchmarks and bandwidth requirements for a set of prescribed manoeuvres. Additionally, InvSim can be used within a path-planning algorithm to produce an improved attitude slew reference, which, if tracked by a suitable controller, minimises structural deformations.

Referring back to the central research question, it can be concluded that an InvSim-based attitude control architecture can provide improved attitude tracking performance over traditional methods for satellites with flexible structures. However, such a control architecture is limited by computational power, and a high sensitivity to model parameters. However, InvSim can additionally be used as a useful ready-to-go temporary control solution, and utilised within a path planning algorithm to produce improved attitude tacking manoeuvres, minimising excitation of flexible structural dynamics.

6.1.3 | Aim 3

The third and final aim was *to develop a framework that utilises InvSim for the design of satellites with flexible structures*. This was achieved in Chapter 4, which demonstrated use of InvSim, in combination with a modular mathematical model, to produce a design framework that could be used to compare the performance of different attitude control actuator types and configurations for LSSs. It was found that to obtain illustrative results for configurations which make use of different types of actuators, the SQP-based InvSim algorithm discussed previously must be used.

Carrying out the same investigation using traditional forward simulation methods would require a bespoke controller to be derived for each actuator configuration to

ensure the full closed-loop dynamics of the LSS can be evaluated when performing prescribed attitude manoeuvres. This would result in a highly laborious task if many designs need to be considered, and would result in a much slower iterative design process. Additionally, performance could be hampered by a poorly tuned controller if forward simulation were adopted, possibly contaminating the results of the investigation. Analytical model-inversion techniques, such as NDI considered in Chapter 2, would suffer from even more substantial labour when deriving the inverse dynamic solution for multiple design iterations. Conversely, the generality of the InvSim solution process allows for the inverse dynamics to be readily solved, provided an up-to-date model is available.

Relating these conclusions to the original research question, it can be surmised that the integral-based InvSim method, with an SQP solver, in combination with a modular mathematical model, can provide an improved design framework when compared to traditional numerical forward simulation and analytical dynamic inversion techniques. This is particularly the case when requirements for the manoeuvrability of the satellite are known during the preliminary stages of a satellite's design, e.g. the repetitive 90 deg slew manoeuvres required by an orbit raising solar sail, with InvSim being well suited to a systems engineering, requirements capture approach. Additionally, the faster, robust Broyden-based InvSim method developed in Chapter 5 greatly improves the speed at which different designs can be evaluated, allowing for a greater breadth of parameters and configurations to be investigated. The improved InvSim method also ensures that designs that require a large number of actuator inputs do not take prohibitively long to solve using InvSim.

6.2 | Future Work

The research covered in this thesis does have its limitations, therefore, this section details future directions of work that could be considered. Much has been discussed about the high computational complexity of InvSim posing a potential obstacle to practical implementation on a satellite's onboard hardware. Therefore, future work should investigate the problem of implementing the InvSim algorithm and InvSim-based controllers onto such hardware, determining requirements on memory, computational time, and power consumption. The application of the faster InvSim method developed in Chapter 5, machine learning [137], and automatic differentiation approaches [139] on hardware, would also prove an interesting area of investigation; the latter possibly being limited by the amount of memory available. However, if through appropriate modifications and choice of control frequency, the InvSim algorithm could be assured

to run faster than real-time, then InvSim could be used as a full-state feedback controller, hence avoiding the need for the additional PD feedback term used in Chapter 3.

Future testing of the developed InvSim-based attitude controller on a real orbiting satellite will not be feasible at first, due to the low technology readiness level. Instead, a good practical test of the InvSim-based controller could be completed using a satellite suspended on an air bearing in a lab, as illustrated in Fig. 6.1. This would allow for the investigation into the effects of model fidelity and uncertainty on an InvSim-based controller, presented in Chapter 3, to be reproduced and validated. This testbed arrangement could also allow for the introduction of an adaptive parameter estimator to mitigate sensitivity of InvSim-based feedforward control to model uncertainty.

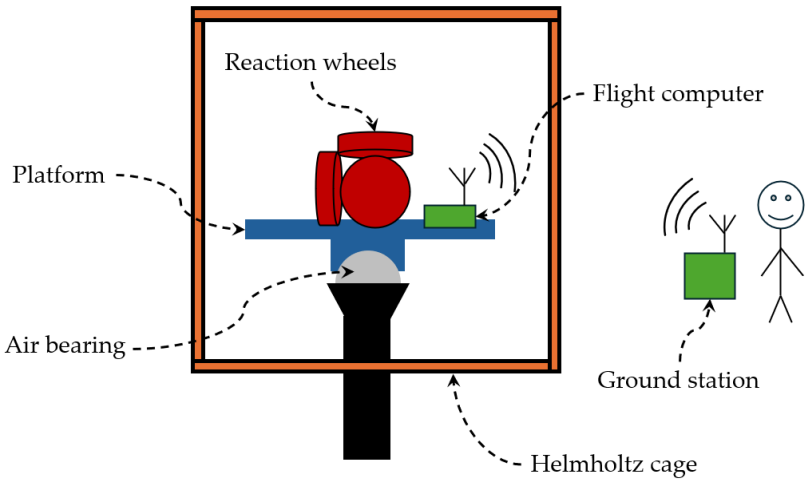


Figure 6.1: Future experimental setup for testing real-time hardware: Simulated satellite platform on an air bearing, equipped with 3 reaction wheels, and an onboard flight computer. Helmholtz cage allows for a simulated magnetic field to test magnetorquer configurations.

The application of InvSim as a tool for designing satellites with flexible structures is also an area filled with opportunities for future work. Currently, the approach has only been demonstrated in a simple conceptual design of generic gossamer LSSs. Future work applying the InvSim approach to design a real satellite would involve the additional step of model validation, once real prototypes of the design have been manufactured. For example, practical testing could reveal discrepancies in the real frequency response of the satellite's flexible structure, potentially leaving the InvSim-based design of the satellite flawed. Validation of the model using experimental data would allow for the InvSim-based study to be re-run, and the satellite's design updated to mitigate any initial flaws.

This thesis has focused on the use of InvSim to characterise and compensate the effects that flexible structures have on the dynamics of satellites; however, the method could

also be utilised to solve the inverse dynamics of many other phenomena commonly experienced by satellites. This could include internal disturbances such as fuel slosh, and external effects such as aerodynamic and gravity gradient torques. While magnetic and solar radiation pressure torques have been considered for active control of satellites in this thesis, they can also induce unwanted disturbances, which, if modelled correctly, could also be characterised and compensated for using InvSim.

Additionally, InvSim could be used to solve the inverse coupled attitude-orbital dynamics of a satellite for various mission scenarios. While this has already been explored by Zhou et al. [137] for rendezvous and docking, scenarios such as the capturing of uncooperative space debris and formation flying could also prove fruitful avenues of investigation. For InvSim to be applicable in each of these cases, a good mathematical model describing the coupled dynamics and an a priori output trajectory of the satellite's position and attitude will need to be known. Definition of a realisable coupled attitude-orbital trajectory for the InvSim to solve may prove particularly difficult. In this case, a model-predictive receding horizon framework [135–137] will be required to ensure the InvSim can converge to an inverse solution that is realisable by the dynamics of the system, and satisfies all constraints.

6.3 Final Remarks

This thesis set out to investigate the potential benefits and limitations of applying numerical Inverse Simulation (InvSim) techniques to the attitude control and design of satellites with flexible structures. The opening chapter highlighted the importance of accurate and stable pointing of a satellite's instrumentation to achieve its mission goals. Due to the mass and form factor limitations of launch vehicles, and the complexity of current and future satellite concepts, flexible structural dynamics are unavoidable. Several examples of the issues induced by these problematic flexible dynamics were also presented, including Explorer I, Alouette I, and the Hubble Space Telescope. It is therefore clear that attitude control strategies and algorithms must take these flexibility effects into account during their design.

The first step in designing these control systems is to understand the specific dynamic behaviour of the satellite, which can be achieved through appropriate mathematical modelling. This thesis included detailed derivations of a multi-fidelity model of satellites with flexible appendages, and a modular LSS model that can facilitate a range of actuators; derivations of the rigid body models used are also included in Appendix E.

Chapter 1 detailed a wide variety of historical and state-of-the-art attitude control methodologies applied to satellites with flexible structures. Many attitude control ap-

proaches for satellites with flexible structures are similar to those usually applied to rigid satellites, with some modifications to account for the flexible dynamics. One such approach identified was inverse control, commonly derived through an analytical method known as nonlinear dynamic inversion (NDI). Whilst this has been proven in literature to provide good results, it is easy to imagine that the task of analytically deriving the inverse solution could become prohibitively laborious. This is particularly true if significant changes need to be made to the satellite design, and its representative model, requiring re-derivation of the inverse solution.

Inverse Simulation (InvSim) was instead identified as an alternative numerical technique for model-inversion. A review of InvSim methodologies and applications revealed that InvSim has already been applied as a prospective control methodology in disciplines such as rotorcraft, planetary rovers, and amphibious vehicles. This thesis, therefore, aimed to identify whether InvSim could be used as an alternative to NDI, to produce a practical, beneficial attitude control architecture for satellites with flexible structures. The conclusions of this thesis highlighted that indeed InvSim can be used within an attitude control architecture, improving tracking performance provided that a sufficiently accurate model can be obtained. Additionally, it was shown that InvSim can be used within a path planning algorithm to generate attitude slew paths which minimise excitation of flexible structural dynamics.

It was identified from the literature that one of the most common applications of InvSim was to study the dynamics and performance of aircraft, which could be used to inform the initial conceptual design process. This thesis therefore demonstrated that InvSim could be used as a tool in the preliminary design of a generic flexible large space structure (LSS), helping to find the best configuration of attitude control actuators that minimised structural deformation. This thesis concluded that, as in other aerospace fields, InvSim can be a beneficial tool in the design process of satellites with flexible structures, particularly when the manoeuvres that the satellite design will be required to perform are known ahead of time. This thesis also demonstrated that this application of InvSim is aided by the ability to implement the integral-based InvSim algorithm within existing simulation frameworks, making use of pre-existing, validated models.

Some points that should be considered when applying InvSim, specifically to solving the inverse dynamics of satellites with flexible structures, were also identified through the course of this thesis. The stiff dynamics inherent to the satellite's flexible structures introduced the need to consider the most efficient numerical integration method to use. Additionally, when only magnetic actuators are available to provide torque to the satellite, numerical stability issues can arise, but can be mitigated if specific steps are taken. A final contribution of this thesis was the development of a new, faster

InvSim method to deal with the drastic drop in computational efficiency experienced when a large number of inputs/actuators are present. It is envisaged that this new method could be utilised and applied in other fields, as the approach requires only minor modification of the most popular integral-based InvSim method.

It is hoped that the work and conclusions reached in this thesis may contribute to the continued development of useful methods and strategies for attitude control of satellites with flexible structures. Many of the limitations in computational power restricting the practical implementation of InvSim as an attitude control solution may begin to disappear as technologies develop, which, in combination with faster InvSim algorithms, such as that developed in Chapter 5, could lead to faster-than-realtime computation; leading the way for full-state feedback InvSim controllers at some point in the future. InvSim also presents an opportunity to simplify the future attitude control design pipeline by providing a temporary, ready-to-go control solution during the preliminary design phase. Future control design pipelines could use InvSim to provide a useful benchmark to compare the final attitude control design against. Use of InvSim in the preliminary design process of attitude control systems and actuator configuration could provide immediate benefits to industry, facilitating rapid prototyping and reducing developmental costs. Finally, it is hoped that this thesis has contributed to the identification of some of the specific considerations that are necessary when applying InvSim to satellites with flexible structures, and that the development of a new faster InvSim method may be beneficial across multiple disciplines.

Appendix A | Frames, Vectors and Matrices

The notation, definition of vectors, matrices, and reference frames are given in more detail in this appendix than afforded in the nomenclature for additional clarity.

A.1 | Notation

The notation of a vector are defined as an italic, bold, underlined, lower-case letter. Additionally, matrices are defined as bold, upper-case letters. Examples of both can be seen:

$$\underset{\text{Vector}}{\boldsymbol{\underline{a}}}\overset{\text{Measurement Frame}}{{}^{B}_{i/j}} \quad \text{(A.1a)} \qquad \underset{\text{Matrix}}{\mathbf{A}} \quad \text{(A.1b)}$$

i relative to j

In many cases, vectors have a defined *measurement frame* shown in the vector's superscript. Vectors such as position, velocity, and acceleration are always defined about some point, or the origin of some frame. For example, the vector in Eq. (A.1a) is measured in the B frame, and gives the position of point i relative to point j.

Vectors and matrices can be transposed, which will be denoted with a superscript $\square^{\top}$:

$$\boldsymbol{a} = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} a_1 & a_2 & a_3 \end{bmatrix}^{\top} \tag{A.2}$$

The *cross-product* of two vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ can be reformed as a simple matrix multiplication through the use of a *skew-symmetric cross-product matrix* $[\boldsymbol{a}]_{\times}$:

$$\boldsymbol{a} \times \boldsymbol{b} = [\boldsymbol{a}]_{\times} \boldsymbol{b} \tag{A.3}$$

with the skew-symmetric cross-product matrix defined as:

$$[\boldsymbol{a}]_\times = \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix} \tag{A.4}$$

There are some useful properties of *skew-symmetric cross-product matrix* which can be utilised to simplify matrix algebra. For example, the order of matrix multiplication can be modified:

$$[\boldsymbol{a}]_\times \boldsymbol{b} = -[\boldsymbol{b}]_\times \boldsymbol{a} = [\boldsymbol{b}]_\times^\top \boldsymbol{a} \tag{A.5}$$

Utilising the transpose $[\cdot]_\times^\top$ can avoid the need for sign changes when reordering a cross product, avoiding confusion during the derivation of mathematical models.

A.2 | Derivatives

The derivative of a vector with respect to (w.r.t.) time can be denoted using *Newton's* notation:

$$\frac{d\boldsymbol{a}}{dt} = \dot{\boldsymbol{a}} \tag{A.6}$$

with the second and third derivatives being:

$$\frac{d^2\boldsymbol{a}}{dt^2} = \ddot{\boldsymbol{a}} \tag{A.7a}$$

$$\frac{d^3\boldsymbol{a}}{dt^3} = \dddot{\boldsymbol{a}} \tag{A.7b}$$

This notation can be generalised for the ν^{th}-order time derivative:

$$\frac{d^\nu \boldsymbol{a}}{dt^\nu} = \overset{(\nu)}{\dot{\boldsymbol{a}}} = \dot{\boldsymbol{a}}^{(\nu)} \tag{A.8}$$

where $\square^{(\nu)}$ and $\overset{(\nu)}{\dot{\square}}$ are used interchangeably to accommodate other notations and save space. The derivative of a scalar a w.r.t a vector $\boldsymbol{b} = [b_1 \;\; b_2 \;\; \cdots \;\; b_N]^\top$ will give the *gradient* ∇:

$$\frac{da}{d\boldsymbol{b}} = \nabla_{\boldsymbol{b}} a = \begin{bmatrix} \dfrac{\partial a}{\partial b_1} \\ \dfrac{\partial a}{\partial b_2} \\ \vdots \\ \dfrac{\partial a}{\partial b_N} \end{bmatrix} \tag{A.9}$$

The derivative of a vector $\boldsymbol{a} = [a_1 \; a_2 \; \cdots \; a_M]^\top$ w.r.t. another vector $\boldsymbol{b} = [b_1 \; b_2 \; \cdots \; b_N]^\top$ will result in a matrix, commonly known as the *Jacobian*:

$$\mathbf{J}_{\boldsymbol{b}}(\boldsymbol{a}) = [\nabla_{\boldsymbol{b}} \boldsymbol{a}] = \begin{bmatrix} \nabla_{\boldsymbol{b}}^\top a_1 \\ \nabla_{\boldsymbol{b}}^\top a_2 \\ \vdots \\ \nabla_{\boldsymbol{b}}^\top a_M \end{bmatrix} = \begin{bmatrix} \frac{\partial \boldsymbol{a}}{\partial b_1} & \frac{\partial \boldsymbol{a}}{\partial b_2} & \cdots & \frac{\partial \boldsymbol{a}}{\partial b_N} \end{bmatrix} = \begin{bmatrix} \frac{\partial a_1}{\partial b_1} & \frac{\partial a_1}{\partial b_2} & \cdots & \frac{\partial a_1}{\partial b_N} \\ \frac{\partial a_2}{\partial b_1} & \frac{\partial a_2}{\partial b_2} & \cdots & \frac{\partial a_2}{\partial b_N} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial a_M}{\partial b_1} & \frac{\partial a_M}{\partial b_2} & \cdots & \frac{\partial a_M}{\partial b_N} \end{bmatrix} \tag{A.10}$$

The *Hessian*, a matrix of second-order partial derivatives, of a scalar a can be given as the transpose of the *Jacobian* of the *gradient*:

$$\mathbf{H}_{\boldsymbol{b}}(a) = \left[\mathbf{J}_{\nabla_{\boldsymbol{b}} a}\right]^\top = \begin{bmatrix} \frac{\partial \nabla_{\boldsymbol{b}} a}{\partial b_1} & \frac{\partial \nabla_{\boldsymbol{b}} a}{\partial b_2} & \cdots & \frac{\partial \nabla_{\boldsymbol{b}} a}{\partial b_N} \end{bmatrix}^\top = \begin{bmatrix} \frac{\partial^2 a}{\partial b_1^2} & \frac{\partial^2 a}{\partial b_1 \partial b_2} & \cdots & \frac{\partial^2 a}{\partial b_1 \partial b_N} \\ \frac{\partial^2 a}{\partial b_2 \partial b_1} & \frac{\partial^2 a}{\partial b_2^2} & \cdots & \frac{\partial^2 a}{\partial b_2 \partial b_N} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 a}{\partial b_N \partial b_1} & \frac{\partial^2 a}{\partial b_N \partial b_2} & \cdots & \frac{\partial^2 a}{\partial b_N^2} \end{bmatrix} \tag{A.11}$$

A.3 | Rotations

A DCM $\mathbf{C}$ transforms a vector from one measurement frame to another:

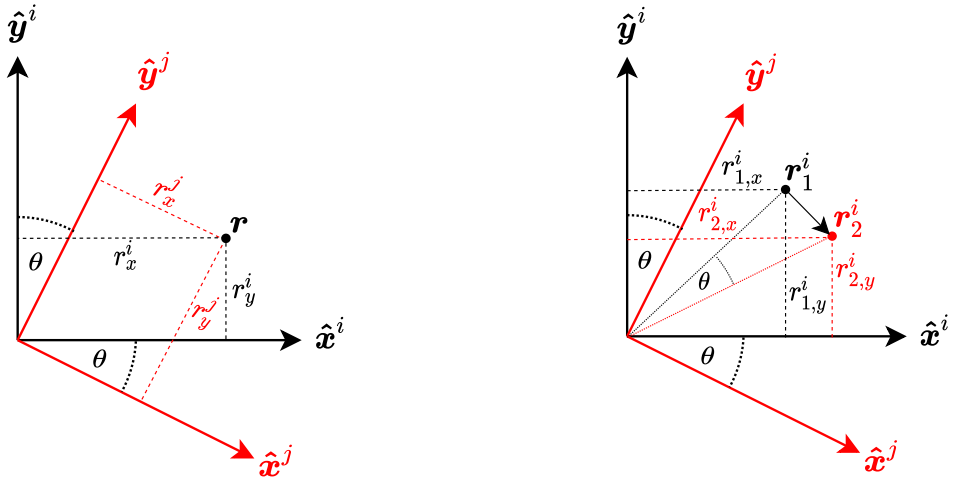
$$\boldsymbol{a}_{i/j}^B = \mathbf{C}^{B/I} \boldsymbol{a}_{i/j}^I \tag{A.12}$$

where the DCM $\mathbf{C}^{B/I}$ rotates the vector $\boldsymbol{a}_{i/j}$ from the I frame to the B frame.

Conversely, a vector can be rotated by an *active* rotation matrix $\mathbf{R}$. This rotates a vector within a measurement frame to a new vector within the same measurement frame. A comparison of the effect coordinate transformations (*inactive* rotations) and *active* rotations have on a vector can be seen in Fig. A.1.

A DCM and an *active* rotation matrix are transposes of each other:

$$\mathbf{C}^{B/I} = [\mathbf{R}^{B/I}]^\top \tag{A.13}$$



(a) Coordinate transformation using DCM: $\boldsymbol{r}^j = \mathbf{C}^{j/i}\boldsymbol{r}^i$

(b) Active rotation of vector: $\boldsymbol{r}_2^i = \mathbf{R}^{j/i}\boldsymbol{r}_1^i$

Figure A.1: Comparison between a coordinate transformation and active rotation

A.4 | Inverse Matrices

A common matrix operation is the *inverse* $\mathbf{A}^{-1}$ such that:

$$\mathbf{A}\mathbf{A}^{-1} = \mathbf{I} \tag{A.14}$$

where $\mathbf{I}$ is an *identity* matrix. The inverse matrix is commonly used when solving a system of equations of the form:

$$\mathbf{A}\boldsymbol{x} = \boldsymbol{y} \tag{A.15}$$

which will have the solution:

$$\boldsymbol{x} = \mathbf{A}^{-1}\boldsymbol{y} \tag{A.16}$$

The matrix *inverse* can be found directly when the number of rows and columns of $\mathbf{A}$ are identical, i.e. when $\boldsymbol{x}$ and $\boldsymbol{y}$ are of the same length. This effectively constitutes a system of equations with the same number of unknowns as there are equations, thus, there is one unique solution to Eq. (A.15).

However, in the case where $\mathbf{A}$ is non-square, i.e. when $\boldsymbol{x}$ and $\boldsymbol{y}$ are of differing lengths, a direct *inverse* is not available and there is not one unique solution to Eq. (A.15). Instead, a *pseudoinverse* can be defined to find a particular solution to Eq. (A.15). One common such pseudoinverse is the MP pseudoinverse [190], denoted: $\mathbf{A}^+$. It is defined as a pseudoinverse that will give a solution for $\boldsymbol{x}$ such that the Euclidean norm $||\mathbf{A}\boldsymbol{x} - \boldsymbol{y}||$ is minimised, giving solution for $\boldsymbol{x}$:

$$\boldsymbol{x} = \mathbf{A}^+\boldsymbol{y} \tag{A.17}$$

Using Eq. (A.17) the condition for the MP pseudoinverse can be found:

$$\arg\min_{\mathbf{A}^+} ||\mathbf{A}(\mathbf{A}^+\boldsymbol{y}) - \boldsymbol{y}|| = ||(\mathbf{A}\mathbf{A}^+ - \mathbf{I})\boldsymbol{y}|| \tag{A.18}$$

This is general and can be applied to problems with complex value solutions, $\{\mathbf{A}, \boldsymbol{x}, \boldsymbol{y}\} \in \mathbb{C}$, however, in the limited case where all values in the problem are real, $\{\mathbf{A}, \boldsymbol{x}, \boldsymbol{y}\} \in \mathbb{R}$, then we get two particular solutions for the MP pseudoinverse [190]:

$$\mathbf{A}^+ = \begin{cases} \mathbf{A}^\top(\mathbf{A}\mathbf{A}^\top)^{-1} & \text{when } (\mathbf{A}\mathbf{A}^\top) \text{ is invertible} \\ (\mathbf{A}^\top\mathbf{A})^{-1}\mathbf{A}^\top & \text{when } (\mathbf{A}^\top\mathbf{A}) \text{ is invertible} \end{cases} \tag{A.19}$$

This will give a solution to Eq. (A.15), or a least-squares fit (closest possible) solution in the case of no exact solution.

Appendix B | Attitude Representations

The attitude of a spacecraft or satellite can be defined using several methods, each defining the rotation of vectors measured in some inertial frame I to the spacecraft's body frame B. For example, the DCMs defined in Appendix A.3 can be used for this purpose:

$$\boldsymbol{r}^B_{i/j} = \mathbf{C}^{B/I}\boldsymbol{r}^I_{i/j} \tag{B.1}$$

where the DCM $\mathbf{C}^{B/I}$ represents the attitude of the spacecraft, $\boldsymbol{r}^I_{i/j}$ is a position vector defined in inertial axes, and $\boldsymbol{r}^B_{i/j}$ is the same vector now measured in body axes.

B.1 | Euler Angles

Alternatively, any orientation or attitude can be defined through three rotations about several intermediate axes. These families of rotations are known as *Euler-Angles* with the common Z-Y-X ordered rotation sequence seen in Fig. B.1.

Each Euler rotation occurs in sequence, rotating about the new body axes after each rotation. This Z-Y-X Euler rotation sequence can used to give the equivalent DCM:

$$\begin{aligned}
\mathbf{C}^{B/I}(\theta_z,\theta_y,\theta_x) &= \mathbf{C}^{B/2}\left(\hat{\boldsymbol{x}}^{B/2},\theta_x\right)\mathbf{C}^{2/1}\left(\hat{\boldsymbol{y}}^{2/1},\theta_y\right)\mathbf{C}^{1/I}\left(\hat{\boldsymbol{z}}^{1/I},\theta_z\right)\\
&= \begin{bmatrix}1 & 0 & 0\\ 0 & \cos\theta_x & \sin\theta_x\\ 0 & -\sin\theta_x & \cos\theta_x\end{bmatrix}\begin{bmatrix}\cos\theta_y & 0 & -\sin\theta_y\\ 0 & 1 & 0\\ \sin\theta_y & 0 & \cos\theta_y\end{bmatrix}\begin{bmatrix}\cos\theta_z & \sin\theta_z & 0\\ -\sin\theta_z & \cos\theta_z & 0\\ 0 & 0 & 1\end{bmatrix}\\
&= \begin{bmatrix}\cos\theta_y\cos\theta_z & \cos\theta_y\sin\theta_z & -\sin\theta_y\\ -\cos\theta_x\sin\theta_z+\sin\theta_x\sin\theta_y\cos\theta_z & \cos\theta_x\cos\theta_z+\sin\theta_x\sin\theta_y\sin\theta_z & \sin\theta_x\cos\theta_y\\ \sin\theta_x\sin\theta_z+\cos\theta_x\sin\theta_y\cos\theta_z & -\sin\theta_x\cos\theta_z+\cos\theta_x\sin\theta_y\sin\theta_z & \cos\theta_x\cos\theta_y\end{bmatrix}
\end{aligned} \tag{B.2}$$

where $\mathbf{C}^{1/I}$, $\mathbf{C}^{2/1}$, $\mathbf{C}^{B/2}$ are the rotations between the intermediate axes sets seen in Fig. B.1, with the superscripts in $\hat{\boldsymbol{x}}^{B/2}$, $\hat{\boldsymbol{y}}^{2/1}$, and $\hat{\boldsymbol{z}}^{1/I}$ reflecting the fact that the axes of rotation between successive intermediate frames are coincident.

Alternative rotation sequences, such as the X-Y-Z Euler rotation sequence, can be used with the equivalent DCM:

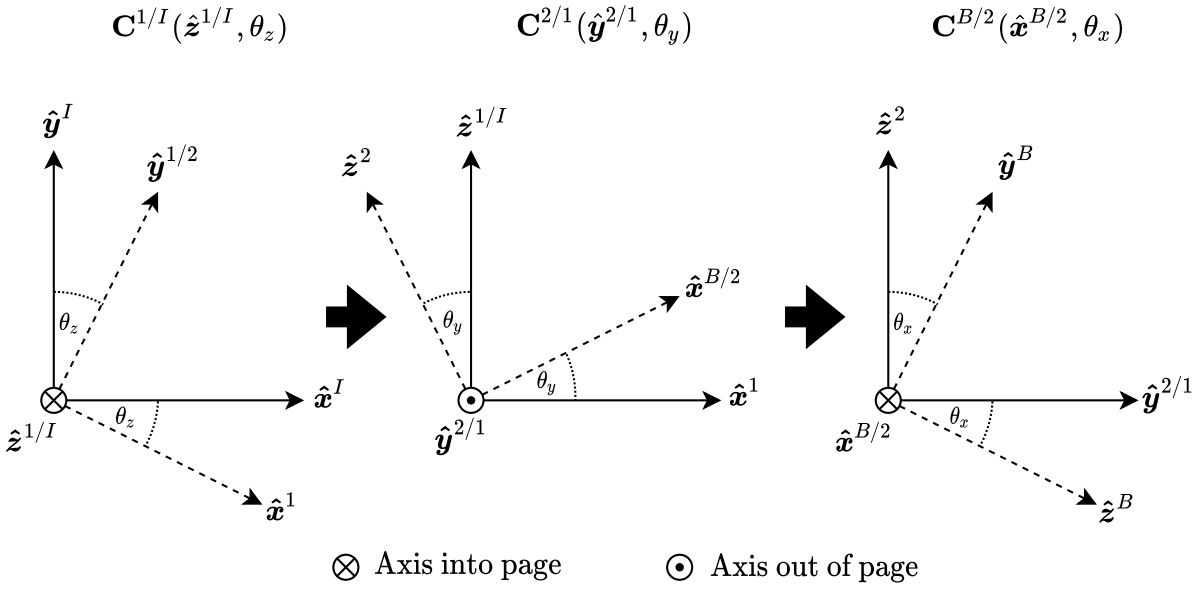


Figure B.1: *Z-Y-X* Euler rotation sequence

$$
\begin{aligned}
\mathbf{C}^{B/I}(\theta_x, \theta_y, \theta_z) &= \mathbf{C}^{B/2}\left(\hat{\boldsymbol{z}}^{B/2}, \theta_z\right)\mathbf{C}^{2/1}\left(\hat{\boldsymbol{y}}^{2/1}, \theta_y\right)\mathbf{C}^{1/I}\left(\hat{\boldsymbol{x}}^{1/I}, \theta_x\right) \\
&= \begin{bmatrix} \cos\theta_z & \sin\theta_z & 0 \\ -\sin\theta_z & \cos\theta_z & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} \cos\theta_y & 0 & -\sin\theta_y \\ 0 & 1 & 0 \\ \sin\theta_y & 0 & \cos\theta_y \end{bmatrix}\begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta_x & \sin\theta_x \\ 0 & -\sin\theta_x & \cos\theta_x \end{bmatrix} \\
&= \begin{bmatrix} \cos\theta_y\cos\theta_z & \cos\theta_x\sin\theta_z + \sin\theta_x\sin\theta_y\cos\theta_z & \sin\theta_x\sin\theta_z - \cos\theta_x\sin\theta_y\cos\theta_z \\ -\cos\theta_y\sin\theta_z & \cos\theta_x\cos\theta_z - \sin\theta_x\sin\theta_y\sin\theta_z & \sin\theta_x\cos\theta_z + \cos\theta_x\sin\theta_y\sin\theta_z \\ \sin\theta_y & -\sin\theta_x\cos\theta_y & \cos\theta_x\cos\theta_y \end{bmatrix}
\end{aligned}
\tag{B.3}
$$

In some cases, the full orientation of the spacecraft may not need to be constrained, but instead, only pointing of a boresight, i.e. a body-fixed axis, is required. This can be represented conveniently through two *Euler-Angle* rotations. For example, assuming the body $\hat{\boldsymbol{z}}^B$ as the boresight, its direction can be fully defined by a *X-Y* or *Y-X* Euler rotation sequence. This works since any rotation about the boresight, $\hat{\boldsymbol{z}}^B$ axis, will not change its direction. This is useful if one wishes to avoid over-constraining the attitude of the spacecraft.

Another issue when using *Euler-Angles* is an issue known as *gimbal-lock*. If the second Euler rotation angle $\theta_y = \frac{\pi}{2}$ then the attitude DCM for a *Z-Y-X* rotation sequence reduces to:

$$\mathbf{C}^{B/I}\left(\theta_z, \pm\frac{\pi}{2}, \theta_x\right) = \begin{bmatrix} 0 & 0 & \pm 1 \\ -\sin(\theta_z \pm \theta_x) & \cos(\theta_z \pm \theta_x) & 0 \\ \mp\cos(\theta_z \pm \theta_x) & \mp\sin(\theta_z \pm \theta_x) & 0 \end{bmatrix} \tag{B.4}$$

It can be seen, therefore, that a degree of freedom has been lost; this is due to the second rotation θ_y causing the first and third rotation axes to become colinear.

B.2 | Quaternions

To avoid the issue of gimbal lock, an alternative attitude representation can be used, such as the *quaternion*. The attitude quaternion rotating a vector from the inertial to the body frame is defined with four elements:

$$\boldsymbol{q}^{B/I} = \begin{bmatrix} q_0 \\ \boldsymbol{q}_{1:3} \end{bmatrix} \tag{B.5}$$

where q_0 is the scalar quaternion element, and $\boldsymbol{q}_{1:3}$ is the three element vector component:

$$\boldsymbol{q}_{1:3} = \begin{bmatrix} q_1 \\ q_2 \\ q_3 \end{bmatrix} \tag{B.6}$$

The form used here is the *scalar first* form implemented in *MATLAB*, a primary tool used in this thesis. Every quaternion is additionally constrained to have a norm of one, $||\boldsymbol{q}|| = 1$, and is equivalent to a rotation about some axis $\hat{\boldsymbol{e}}$ and by angle θ:

$$\boldsymbol{q}(\hat{\boldsymbol{e}}, \theta) = \begin{bmatrix} \cos(\theta/2) \\ \hat{\boldsymbol{e}}\sin(\theta/2) \end{bmatrix} \tag{B.7}$$

with the equivalent attitude DCM being given by:

$$\mathbf{C}(\boldsymbol{q}) = \begin{bmatrix} q_1^2 - q_2^2 - q_3^2 + q_0^2 & 2(q_1q_2 + q_3q_0) & 2(q_1q_3 - q_2q_0) \\ 2(q_2q_1 - q_3q_0) & -q_1^2 + q_2^2 - q_3^2 + q_0^2 & 2(q_2q_3 + q_1q_0) \\ 2(q_3q_1 + q_2q_0) & 2(q_3q_2 - q_1q_0) & -q_1^2 - q_2^2 + q_3^2 + q_0^2 \end{bmatrix} \tag{B.8}$$

Multiplication of two quaternions $\overline{\boldsymbol{q}}$ and $\boldsymbol{q}$ gives a third quaternion that is a combination of both rotations:

$$\overline{\boldsymbol{q}} \otimes \boldsymbol{q} = \begin{bmatrix} \bar{q}_0 q_0 - \overline{\boldsymbol{q}}_{1:3}^{\top} \boldsymbol{q}_{1:3} \\ q_0 \overline{\boldsymbol{q}}_{1:3} + \bar{q}_0 \boldsymbol{q}_{1:3} - \overline{\boldsymbol{q}}_{1:3} \times \boldsymbol{q}_{1:3} \end{bmatrix} \tag{B.9}$$

The quaternion defining the attitude of a spacecraft can therefore be defined in terms of the Z-Y-X Euler Angle rotation set defined previously:

$$\boldsymbol{q}^{B/I} = \begin{bmatrix} \cos(\theta_x/2) \\ \sin(\theta_x/2) \\ 0 \\ 0 \end{bmatrix} \otimes \begin{bmatrix} \cos(\theta_y/2) \\ 0 \\ \sin(\theta_y/2) \\ 0 \end{bmatrix} \otimes \begin{bmatrix} \cos(\theta_z/2) \\ 0 \\ 0 \\ \sin(\theta_z/2) \end{bmatrix} \tag{B.10}$$

while ensuring a right-to-left multiplication order, as is the case with matrix multiplication. To rotate a position vector $\boldsymbol{r}_{i/j}^{I}$ into body axes using the quaternion $\boldsymbol{q}^{B/I}$ we can perform the quaternion multiplication:

$$\begin{bmatrix} 0 \\ \boldsymbol{r}_{i/j}^{B} \end{bmatrix} = \boldsymbol{q}^{B/I} \otimes \begin{bmatrix} 0 \\ \boldsymbol{r}_{i/j}^{I} \end{bmatrix} \otimes \boldsymbol{q}^{*\,B/I} \tag{B.11}$$

where $\boldsymbol{q}^*$ is the quaternion conjugate:

$$\boldsymbol{q}^* = \begin{bmatrix} q_0 \\ \boldsymbol{q}_{1:3} \end{bmatrix}^* = \begin{bmatrix} q_0 \\ -\boldsymbol{q}_{1:3} \end{bmatrix} \tag{B.12}$$

It is also easy to find the shortest rotational angle between two quaternions. This will become very useful when trying to determine a satellite's pointing error, the angle between some desired attitude (defined by quaternion $\boldsymbol{q}_d$) and the actual attitude ($\boldsymbol{q}$). The *error* quaternion $\boldsymbol{q}_e$ can therefore be calculated:

$$\boldsymbol{q}_e = \boldsymbol{q}_d \otimes \boldsymbol{q}^* \tag{B.13}$$

with the pointing error $\delta\theta$ being extracted:

$$\delta\theta = 2\cos^{-1}(q_{e,0}) = 2\cos^{-1}(\boldsymbol{q}_d \cdot \boldsymbol{q}) = 2\cos^{-1}\left(\boldsymbol{q}_d{}^{\top} \boldsymbol{q}\right) \tag{B.14}$$

where $q_{e,0}$ is the scalar component of the quaternion error $\boldsymbol{q}_e$.

B.3 | Kinematics

Finally, the attitude kinematics must be defined for a given body-fixed angular velocity, or body rate, $\boldsymbol{\omega}_{B/I}^{B} = \begin{bmatrix}\omega_x & \omega_y & \omega_z\end{bmatrix}^\top$. Firstly for the *Z-Y-X* Euler rotation sequence the kinematics are:

$$\begin{bmatrix}\dot{\theta}_x \\ \dot{\theta}_y \\ \dot{\theta}_z\end{bmatrix} = \begin{bmatrix}1 & \tan\theta_y \sin\theta_x & \tan\theta_y \cos\theta_x \\ 0 & \cos\theta_x & -\sin\theta_x \\ 0 & \sec\theta_y \sin\theta_x & \sec\theta_y \cos\theta_x\end{bmatrix} \begin{bmatrix}\omega_x \\ \omega_y \\ \omega_z\end{bmatrix} \tag{B.15}$$

and similarly the kinematics for the *X-Y-Z* Euler rotation sequence:

$$\begin{bmatrix}\dot{\theta}_x \\ \dot{\theta}_y \\ \dot{\theta}_z\end{bmatrix} = \begin{bmatrix}\cos\theta_z \sec\theta_y & -\sin\theta_z \sec\theta_y & 0 \\ \sin\theta_z & \cos\theta_z & 0 \\ -\tan\theta_y \cos\theta_z & \tan\theta_y \sin\theta_z & 1\end{bmatrix} \begin{bmatrix}\omega_x \\ \omega_y \\ \omega_z\end{bmatrix} \tag{B.16}$$

The quaternion kinematics can be given using a matrix defined in terms of the body rate:

$$\dot{\boldsymbol{q}}^{B/I} = \frac{1}{2}\boldsymbol{\Omega}(\boldsymbol{\omega}_{B/I}^{B})\, \boldsymbol{q}^{B/I} = \frac{1}{2}\begin{bmatrix}\mathbf{0}^{3\times 3} & -\boldsymbol{\omega}_{B/I}^{B}{}^\top \\ \boldsymbol{\omega}_{B/I}^{B} & \left[\boldsymbol{\omega}_{B/I}^{B}\right]_\times^\top\end{bmatrix} \boldsymbol{q}^{B/I} \tag{B.17}$$

or alternatively in terms of the quaternion:

$$\dot{\boldsymbol{q}}^{B/I} = \frac{1}{2}\boldsymbol{\Xi}(\boldsymbol{q}^{B/I})\, \boldsymbol{\omega}_{B/I}^{B} = \frac{1}{2}\begin{bmatrix}-\boldsymbol{q}_{1:3}^{B/I}{}^\top \\ q_0^{B/I}\mathbf{I}^{3\times 3} + \left[\boldsymbol{q}_{1:3}^{B/I}\right]_\times\end{bmatrix} \boldsymbol{\omega}_{B/I}^{B} \tag{B.18}$$

where $\mathbf{I}^{3\times 3}$ is a 3×3 identity matrix. Furthermore, the inverse kinematics can be given:

$$\boldsymbol{\omega}_{B/I}^{B} = 2\boldsymbol{\Xi}^\top(\boldsymbol{q}^{B/I})\dot{\boldsymbol{q}}^{B/I} = 2\begin{bmatrix}-\boldsymbol{q}_{1:3}^{B/I} & q_0^{B/I}\mathbf{I}^{3\times 3} + \left[\boldsymbol{q}_{1:3}^{B/I}\right]_\times^\top\end{bmatrix} \dot{\boldsymbol{q}}^{B/I} \tag{B.19}$$

B.4 | Spherical-Linear-Interpolation (SLERP)

As shown in Appendix B.2, quaternions have many useful mathematical properties, such as having no singularities and being more concise than other attitude representations, e.g. DCMs. It is therefore useful to define fixed-time rest-to-rest slew manoeuvres as a timeseries of attitude quaternions. This can be achieved by considering SLERP [174], a linear interpolation within quaternion space:

$$\boldsymbol{q}(t) = \frac{\sin((1-h(t))\delta_0)}{\sin(\delta_0)}\boldsymbol{q}_i + \frac{\sin(h(t)\delta_0)}{\sin(\delta_0)}\boldsymbol{q}_f \tag{B.20}$$

where δ_0 is half the angle between the initial quaternion, $\boldsymbol{q}_i$, and final quaternion, $\boldsymbol{q}_f$, given by:

$$\delta_0 = \cos^{-1}(\boldsymbol{q}_i \cdot \boldsymbol{q}_f) \tag{B.21}$$

The function $h(t)$ defines the non-dimensional distance travelled along the SLERP path at time t. If $h(t)$ were simply a straight line, then Eq. (B.20) would give a quaternion path of constant velocity. Instead, to achieve a rest-to-rest manoeuvre, $h(t)$ can be chosen to be a polynomial. A 7^{th} order polynomial is chosen specifically in this case due to it producing a significantly smooth manoeuvre, being continuous at higher-order derivatives (e.g. jerk):

$$h(t) = a_4 t^7 + a_3 t^6 + a_2 t^5 + a_1 t^4 \tag{B.22}$$

with the coefficients calculated from the duration of the manoeuvre t_f:

$$a_4 = \frac{-20}{t_f^7} \quad \text{(B.23a)} \qquad a_3 = \frac{70}{t_f^6} \quad \text{(B.23b)} \qquad a_2 = \frac{-84}{t_f^5} \quad \text{(B.23c)} \qquad a_1 = \frac{35}{t_f^4} \quad \text{(B.23d)}$$

The first derivative, i.e. velocity, of the SLERP path can be found:

$$\dot{\boldsymbol{q}}(t) = \frac{-\dot{h}(t)\delta_0 \cos((1-h(t))\delta_0)}{\sin(\delta_0)}\boldsymbol{q}_i + \frac{\dot{h}(t)\delta_0 \cos(h(t)\delta_0)}{\sin(\delta_0)}\boldsymbol{q}_f \tag{B.24}$$

and the second derivative, i.e. acceleration, similarly:

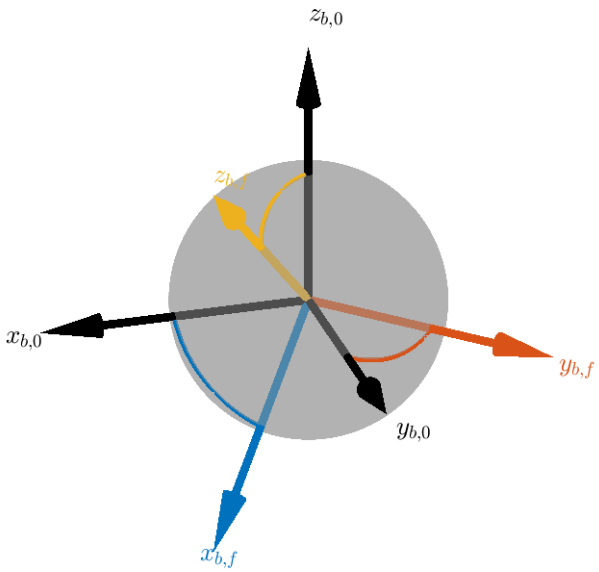
$$\begin{aligned}\ddot{\boldsymbol{q}}(t) = {} & \frac{-\ddot{h}(t)\delta_0 \cos((1-h(t))\delta_0) - \dot{h}(t)^2\delta_0^2 \sin((1-h(t))\delta_0)}{\sin(\delta_0)}\boldsymbol{q}_i \\ & + \frac{\ddot{h}(t)\delta_0 \cos(h(t)\delta_0) - \dot{h}(t)^2\delta_0^2 \sin(h(t)\delta_0)}{\sin(\delta_0)}\boldsymbol{q}_f\end{aligned} \tag{B.25}$$

with the first-order and second-order derivatives of the polynomial $h(t)$ are defined as:

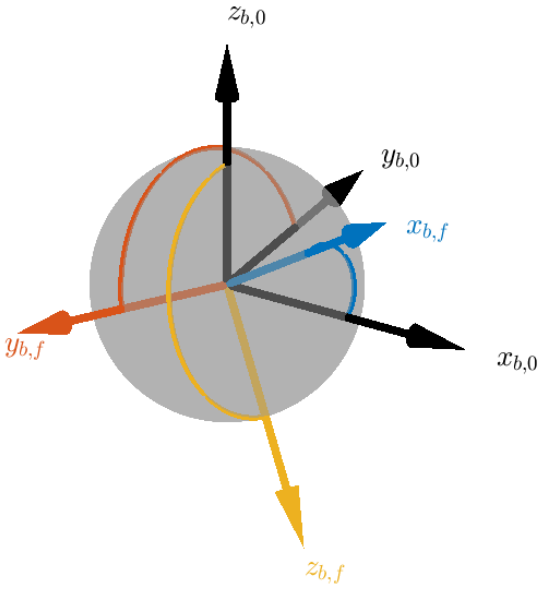
$$\dot{h}(t) = 7a_4 t^6 + 6a_3 t^5 + 5a_2 t^4 + 4a_1 t^3 \tag{B.26a}$$

$$\ddot{h}(t) = 42a_4 t^5 + 30a_3 t^4 + 20a_2 t^3 + 12a_1 t^2 \tag{B.26b}$$

Examples of both small and large SLERP manoeuvres can be seen in Fig. B.2. Each of the satellite's body axes is projected on a unit sphere to show its movement during the SLERP manoeuvre.



(a) Small slew



(b) Large slew

Figure B.2: Projections of the body axes on a unit sphere during rest-to-rest SLERP slew manoeuvres

Appendix C | Dynamics

This appendix details the methods used in this thesis for deriving the sets of ODEs that describe the dynamics of satellites with flexible structures. These ODEs can then be solved numerically integrated using methods outlined later in Appendix D.3, and used during the InvSim process to solve the inverse dynamics. There are several approaches for deriving these ODEs, as discussed in Section 1.2.1; however, in this thesis, Newtonian and Lagrangian dynamics will be primarily used.

C.1 | Newtonian

The acceleration of a system of particles will be related to the sum of *external* and *internal* forces through *Newton's second law* [69, p.5]:

$$m_i \ddot{\boldsymbol{r}}^I_{i/I} = \sum_j \boldsymbol{f}^I_{j,i} + \sum_k \boldsymbol{f}^I_{ext,k} \tag{C.1}$$

where $\ddot{\boldsymbol{r}}^I_{i/I}$ is the inertial acceleration of particle i, m_i is the mass of the particle, $\boldsymbol{f}^I_{j,i}$ is the internal force of particle j acting on particle i, and $\boldsymbol{f}^I_{ext,k}$ is the k^{th} external force acting on particle i. Summing Eq. (C.1) for all particles in the system:

$$\sum_i m_i \ddot{\boldsymbol{r}}^I_{i/I} = \sum_i \left(\sum_j \boldsymbol{f}^I_{j,i} \boldsymbol{f}^I_{j,i} \right) + \sum_i \left(\sum_k \boldsymbol{f}^I_{ext,k} \right) \tag{C.2}$$

where $\boldsymbol{f}^I_{i,i}$ will clearly be zero. Additionally, according to *Newton's third law*, each internal force will have an equal and opposite reaction force:

$$\boldsymbol{f}^I_{j,i} = -\boldsymbol{f}^I_{i,j} \tag{C.3}$$

therefore, the summation of all *internal* forces will be zero, as each *internal* force is cancelled out by its reaction. Defining the centre of mass position of the system:

$$\boldsymbol{r}^I_{CoM} = \frac{\sum m_i \boldsymbol{r}^I_i}{\sum m_i} = \frac{\sum m_i \boldsymbol{r}^I_i}{m_{tot}} \tag{C.4}$$

with corresponding velocity and acceleration:

$$\dot{\boldsymbol{r}}^I_{CoM} = \frac{\sum m_i \dot{\boldsymbol{r}}^I_i}{m_{tot}} \tag{C.5a}$$

$$\ddot{\boldsymbol{r}}^I_{CoM} = \frac{\sum m_i \ddot{\boldsymbol{r}}^I_i}{m_{tot}} \tag{C.5b}$$

Substituting Eq. (C.5b) into Eq. (C.2) gives:

$$m_{tot}\ddot{\boldsymbol{r}}^I_{CoM} = \sum_i \left(\sum_k \boldsymbol{f}^I_{ext,k} \right) \tag{C.6}$$

While simple and tractable, Newtonian dynamics does require that all the *internal forces* be resolved to solve the dynamics of each particle within the system, which for a complex system can become tedious.

C.2 | Lagrangian

Instead, *Euler-Lagrangian* approach can be used to derive the differential equations by considering the energy in the system, rather than the forces [69, p.45]:

$$\frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{q}_i}\right) - \frac{\partial \mathcal{L}}{\partial q_i} + \frac{\partial \mathcal{D}}{\partial \dot{q}_i} = Q_i \tag{C.7a}$$

$$\mathcal{L} = \mathcal{T} - \mathcal{V} \tag{C.7b}$$

where q_i is a *generalised* coordinate in this case (could be a position, angle etc.), Q_i is a *generalised* force (could be a force, torque etc.), $\mathcal{L}$ is the lagrangian, $\mathcal{T}$ is total kinetic energy in the system, $\mathcal{V}$ is the total potential energy in the system, and $\mathcal{D}$ is the dissipative function to deal with velocity proportional, non-conservative forces such as damping which remove energy from the system.

Performing the partial derivatives, some substitution, and rearranging can usually yield a set of ODEs that will form an adequate model. In the case of a rotating system, such as a spinning satellite, some additional care needs to be taken to ensure the inclusion of *transport theorem*; i.e. considering the rotation of the frame when performing differentiation w.r.t time. For a vector relative to the rotating body frame B, measured in an arbitrary frame X, the rotational velocity of the body frame $\boldsymbol{\omega}^X_{B/I}$ relative to inertial space is taken into account through the cross product:

$$\frac{d\boldsymbol{r}^X_{i/B}}{dt} = \dot{\boldsymbol{r}}^X_{i/B} + \boldsymbol{\omega}^X_{B/I} \times \boldsymbol{r}^X_{i/B} = \dot{\boldsymbol{r}}^X_{i/B} + \left[\boldsymbol{\omega}^X_{B/I}\right]_\times \boldsymbol{r}^X_{i/B} \tag{C.8}$$

Appendix D | Numerical Methods

This thesis utilises several numerical methods throughout. Primarily numerical integration methods for solving ODEs, and numerical methods for finding the roots or optimum of a function. These are all essential to the numerical InvSim techniques employed in this thesis, and so are described in this appendix in some detail.

D.1 | Newton's Method

Newton's method or the *NR* method is an iterative numerical process which aims to find the roots of a function $f(x)$:

$$f(x) = 0 \tag{D.1}$$

The function $f(x)$ can be approximated using the *Taylor series*, an infinite sum of weighted derivatives about some point x_0:

$$f(x) \approx f(x_0) + \frac{df(x_0)}{dx}(x - x_0) + \frac{1}{2}\frac{d^2f(x_0)}{dx^2}(x - x_0)^2 + \cdots \frac{1}{k!}\frac{d^kf(x_0)}{dx^k}(x - x_0)^k + \cdots \tag{D.2}$$

As the number of terms in the Taylor series increases, the accuracy of the approximation around x_0 improves. Newton's method for finding a function's roots (Eq. (D.1)) can be derived by forming a first-order approximation of the function, using the first two terms of the Taylor series (Eq. (D.2)), and setting it to zero:

$$f(x_0) + \frac{df(x_0)}{dx}(x - x_0) = 0 \tag{D.3}$$

The solution/root of this approximation of the function can then be found:

$$x = x_0 - \left(\frac{df(x_0)}{dx}\right)^{-1} f(x_0) \tag{D.4}$$

However, in most cases, this first-order approximation will not be sufficient to provide an accurate solution for x, especially in the case where the true solution is far from x_0. However, Eq. (D.4) will provide a solution x that is closer to the true solution than x_0. Therefore, Eq. (D.4) can be applied iteratively to give an updated estimate of the

solution x_{n+1} from a previous estimate x_n:

$$x_{n+1} = x_n - \left(\frac{df(x_n)}{dx}\right)^{-1} f(x_n) \tag{D.5}$$

Application of the iterative NR method for solving for the root of a simple polynomial is illustrated in Fig. D.1. One point to consider is that if multiple solutions to $f(x) = 0$ exist, then the solution converged upon will entirely depend on the initial estimate x_0.

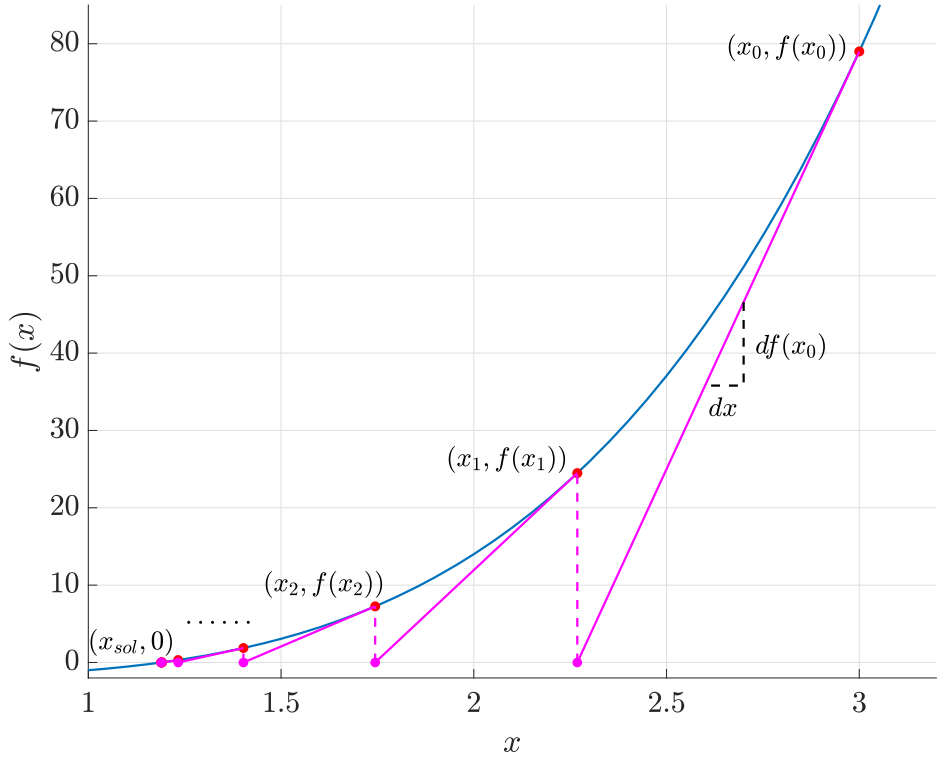


Figure D.1: Solving for a root of $f(x)$ using Newton-Raphson (NR) method. $f(x) = x^4 - 2$

D.1.1 | Systems of Equations

The NR method can be expanded to solve multivariate systems of equations $\boldsymbol{f}(\boldsymbol{x}) = 0$. The first-order Taylor series for such a system of equations can be given as:

$$\boldsymbol{f}(\boldsymbol{x}) \approx \boldsymbol{f}(\boldsymbol{x}_0) + [\nabla_{\boldsymbol{x}} \boldsymbol{f}(\boldsymbol{x}_0)] (\boldsymbol{x} - \boldsymbol{x}_0) \tag{D.6}$$

Applying $\boldsymbol{f}(\boldsymbol{x}) = 0$ and introducing the iterations $\boldsymbol{x} = \boldsymbol{x}_{n+1}$ and $\boldsymbol{x}_0 = \boldsymbol{x}_n$ gives the NR update:

$$\boldsymbol{x}_{n+1} = \boldsymbol{x}_n - [\nabla_{\boldsymbol{x}} \boldsymbol{f}(\boldsymbol{x}_n)]^{-1} \boldsymbol{f}(\boldsymbol{x}_n) = \boldsymbol{x}_n - \mathbf{J}_{\boldsymbol{x}}^{-1} (f(\boldsymbol{x}_n)) \boldsymbol{f}(\boldsymbol{x}_n) \tag{D.7}$$

where $\mathbf{J}_f^{-1}(\boldsymbol{x}_0)$ is the inverse of the Jacobian of partial derivatives of each equation $f_i()$ w.r.t to each independent variable x_j (Eq. (A.10)):

$$\mathbf{J}_{\boldsymbol{x}}\left(f(\boldsymbol{x}_0)\right) = \begin{bmatrix} \frac{\partial f_1(\boldsymbol{x}_0)}{\partial x_1} & \frac{\partial f_1(\boldsymbol{x}_0)}{\partial x_2} & \cdots & \frac{\partial f_1(\boldsymbol{x}_0)}{\partial x_{N_x}} \\ \frac{\partial f_2(\boldsymbol{x}_0)}{\partial x_1} & \frac{\partial f_2(\boldsymbol{x}_0)}{\partial x_2} & \cdots & \frac{\partial f_2(\boldsymbol{x}_0)}{\partial x_{N_x}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_{N_f}(\boldsymbol{x}_0)}{\partial x_1} & \frac{\partial f_{N_f}(\boldsymbol{x}_0)}{\partial x_2} & \cdots & \frac{\partial f_{N_f}(\boldsymbol{x}_0)}{\partial x_{N_x}} \end{bmatrix} \tag{D.8}$$

When the number of equations and independent variables are equal ($N_f = N_x$), then the Jacobian $\mathbf{J}_{\boldsymbol{x}}\left(f(\boldsymbol{x}_0)\right)$ will be square and directly invertible. In the case where the Jacobian is non-square ($N_f \neq N_x$) a direct inverse will not exist, and instead a MP pseudoinverse of the Jacobian $\mathbf{J}_{\boldsymbol{x}}^{+}\left(f(\boldsymbol{x}_n)\right)$ can be used (Eq. (A.19)):

$$\boldsymbol{x}_{n+1} = \boldsymbol{x}_n - \mathbf{J}_{\boldsymbol{x}}^{+}\left(f(\boldsymbol{x}_n)\right)\boldsymbol{f}(\boldsymbol{x}_n) \tag{D.9}$$

For the case where the system of equations are overdetermined ($N_f > N_x$) then there will exist multiple solutions, with the MP pseudoinverse causing the NR method to converge to the solution that minimises $||\boldsymbol{x}_{n+1} - \boldsymbol{x}_n||$. The solution found will therefore be sensitive to the initial estimate $\boldsymbol{x}_0$ used at the start of the NR method.

When the system of equations is underdetermined ($N_f < N_x$), then there may be no exact solutions. Use of the MP pseudoinverse may converge to a suitable solution that satisfies $\boldsymbol{f}(\boldsymbol{x}) = 0$, or may only converge to a least-squares fit solution $||\boldsymbol{f}(\boldsymbol{x})||^2$.

D.1.2 | Optimistation

Numerical methods can also be used to find the optimum $\boldsymbol{x}^*$ that minimises a scalar cost function $f(\boldsymbol{x})$:

$$\boldsymbol{x}^* = \arg\min_{\boldsymbol{x}} f(\boldsymbol{x}) \tag{D.10}$$

One of the most intuitive approaches is known as *gradient descent*, whereby steps in $\boldsymbol{x}$ are taken *down* the natural contours of the function $f(\boldsymbol{x})$ until the minimum is eventually reached. The update for such a gradient descent approach can be given [191]:

$$\boldsymbol{x}_{n+1} = \boldsymbol{x}_n - \gamma \frac{\nabla_{\boldsymbol{x}} f(\boldsymbol{x}_n)}{||\nabla_{\boldsymbol{x}} f(\boldsymbol{x}_n)||} \tag{D.11}$$

where γ is a specified step size to take in the *downhill* direction. The gradient descent approach, while simple, can struggle to converge quickly, with a poor choice of step

size γ causing iterations to zig-zag along the contours. The gradient descent method can be improved through the addition of a *linesearch* subroutine to find the best step size γ to take. However, an alternative approach that also takes into account the function's curvature in addition to the gradient can be used.

The curvature of a 1D scalar function can be taken into account by approximating it as a second-order Taylor series (quadratic) taken from Eq. (D.2):

$$f(x) \approx f(x_0) + \frac{df(x_0)}{dx}(x - x_0) + \frac{1}{2}\frac{d^2 f(x_0)}{dx^2}(x - x_0)^2 \tag{D.12}$$

which can be expanded for an N dimensional scalar valued function:

$$f(\boldsymbol{x}) \approx f(\boldsymbol{x}_0) + \nabla_{\boldsymbol{x}}^{\top} f(\boldsymbol{x}_0)\boldsymbol{s}_x + \frac{1}{2}\boldsymbol{s}_x^{\top}\mathbf{H}_{\boldsymbol{x}}\left(f(\boldsymbol{x}_0)\right)\boldsymbol{s}_x \tag{D.13}$$

where $\mathbf{H}_{\boldsymbol{x}}\left(f(\boldsymbol{x}_0)\right)$ is the hessian of the function:

$$\mathbf{H}_{\boldsymbol{x}}\left(f(\boldsymbol{x}_0)\right) = \begin{bmatrix} \frac{\partial^2 f(\boldsymbol{x}_0)}{\partial x_1^2} & \frac{\partial^2 f(\boldsymbol{x}_0)}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f(\boldsymbol{x}_0)}{\partial x_1 \partial x_N} \\ \frac{\partial^2 f(\boldsymbol{x}_0)}{\partial x_2 \partial x_1} & \frac{\partial^2 f(\boldsymbol{x}_0)}{\partial x_2^2} & \cdots & \frac{\partial^2 f(\boldsymbol{x}_0)}{\partial x_2 \partial x_N} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f(\boldsymbol{x}_0)}{\partial x_N \partial x_1} & \frac{\partial^2 f(\boldsymbol{x}_0)}{\partial x_N \partial x_2} & \cdots & \frac{\partial^2 f(\boldsymbol{x}_0)}{\partial x_N^2} \end{bmatrix} \tag{D.14}$$

and $\boldsymbol{s}_x$ is the step taken from $\boldsymbol{x}_0$:

$$\boldsymbol{s}_x = \boldsymbol{x} - \boldsymbol{x}_0 \tag{D.15}$$

The minimum of $f(\boldsymbol{x})$ will occur where the gradient of $f(\boldsymbol{x})$ w.r.t the step taken $\boldsymbol{s}_x$ is zero. Therefore, Newton's method can be applied to find the solution to $\nabla_{\boldsymbol{s}_x} f(\boldsymbol{x}) = 0$:

$$\nabla_{\boldsymbol{s}_x} f(\boldsymbol{x}) = \frac{d}{d\boldsymbol{s}_x}\left(f(\boldsymbol{x}_0) + \nabla_{\boldsymbol{x}}^{\top} f(\boldsymbol{x}_0)\boldsymbol{s}_x + \frac{1}{2}\boldsymbol{p}_x^{\top}\mathbf{H}_{\boldsymbol{x}}\left(f(\boldsymbol{x}_0)\right)\boldsymbol{s}_x\right) = 0 \tag{D.16}$$

which gives:

$$\nabla_{\boldsymbol{x}} f(\boldsymbol{x}_0) + \mathbf{H}_{\boldsymbol{x}}\left(f(\boldsymbol{x}_0)\right)\boldsymbol{s}_x = 0 \tag{D.17}$$

Then applying Eq. (D.17) iteratively, $\boldsymbol{s}_x = \boldsymbol{x}_{n+1} - \boldsymbol{x}_n$, gives the update:

$$\boldsymbol{x}_{n+1} = \boldsymbol{x}_n - \mathbf{H}_{\boldsymbol{x}}^{-1}\left(f(\boldsymbol{x}_n)\right)\ \nabla_{\boldsymbol{x}} f(\boldsymbol{x}_n) \tag{D.18}$$

A comparison between gradient descent Eq. (D.11) and Newton's method Eq. (D.18), minimising a Rosenbrock function, can be seen in Fig. D.2. Newton's method converges

to the optimal solution in far fewer iterations through its utilisation of second-order curvature information. Effectively, the function being optimised has been estimated by fitting a quadratic (Eq. (D.13)), with the update (Eq. (D.18)) stepping into the minimum of the fitted quadratic. Recalculating the Hessian and gradient around the new iterate gives a new quadratic estimation of the function, and the process can repeat.

In many cases, however, the gradient (first-order) and Hessian (second-order) will not be directly available and must therefore be calculated numerically. The Hessian is particularly computationally expensive to numerically calculate and is, therefore, in many cases, estimated to improve computational efficiency. This gives rise to so-called *QN* methods such as Broyden–Fletcher–Goldfarb–Shanno (BFGS) which leverage the calculated first-order gradient information from each iteration to build an approximation of the Hessian without needing to calculate it directly at each step [191]. QN methods such as BFGS are popular and have been implemented in *MATLAB's* `fminunc()` function as part of their optimisation toolbox.

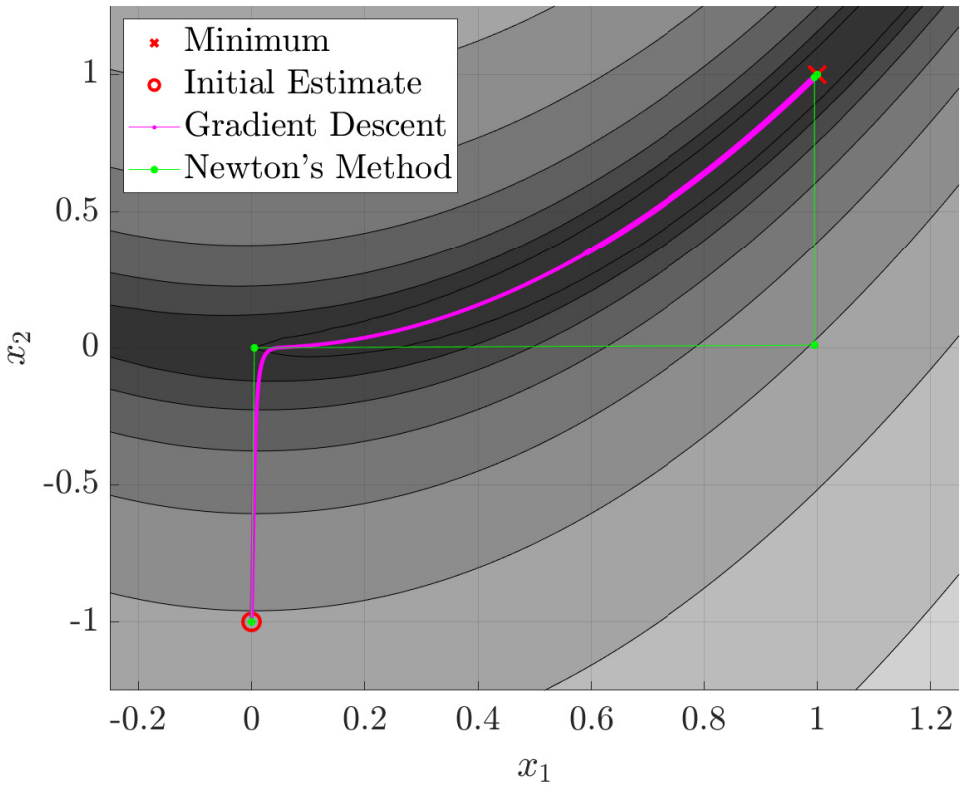


Figure D.2: Finding minimum of $f(x_1, x_2)$ using gradient descent and Newton's method. Rosenbrock function: $f(x_1, x_2) = (1 - x_1)^2 + 100(x_2 - x_1^2)^2$

Comparing the Newton update for optimisation seen in Eq. (D.18) to that for solving a system of equations, Eq. (D.7), some similarities can be seen. Eq. (D.7) updates towards

a solution to $\boldsymbol{f}(\boldsymbol{x}) = 0$ while Eq. (D.18) uses the same update to converge to a solution to $\nabla_{\boldsymbol{x}} f(\boldsymbol{x}) = 0$. This can be seen clearly by substituting $\boldsymbol{f}(\boldsymbol{x}) = \nabla_{\boldsymbol{x}} f(\boldsymbol{x})$ into Eq. (D.7).

The terms Newton's method and Newton-Raphson (NR) method are effectively two names for the same update used in both Eq. (D.7) and Eq. (D.18). However, to avoid confusion in this thesis, the NR method refers to solving the roots of a system of equations (Eq. (D.7)), while Newton's method (or QN method) refers to the application within the optimisation of some scalar-valued function (Eq. (D.18)).

D.2 | Sequential-Quadratic-Programming (SQP)

In Appendix D.1 it was shown that Newton's method, or QN methods, could provide an efficient approach to solving unconstrained optimisation problems by minimising a cost function $f(\boldsymbol{x})$. However, in many engineering problems, there can exist constraints that need to be imposed on the design variable $\boldsymbol{x}$. A common numerical method for solving constrained optimisation problems is *SQP* [191, 192], which will be detailed herein.

D.2.1 | Equality Constraints

First, consider an optimisation problem with equality constraints only:

$$\min_{\boldsymbol{x}} \quad f(\boldsymbol{x}) \tag{D.19a}$$

$$\text{subject to} \quad c_i(\boldsymbol{x}) = 0 \tag{D.19b}$$

where $c_i(\boldsymbol{x})$ is a set of equality constraint functions. An example of an equality constraint imposed on the Rosenbrock function is illustrated in Fig. D.3. It can be seen that at the constrained minimum, the gradient of the cost function and the equality constraint are *colinear*. It can be understood that for the next iterate $\boldsymbol{x}_{n+1}$ to remain on the constraint $c(\boldsymbol{x}) = 0$ the step $\boldsymbol{s}_x$ must be perpendicular/orthogonal to the gradient of that constraint $\nabla_{\boldsymbol{x}} c(\boldsymbol{x})$. The step $\boldsymbol{s}_x$ should minimise the cost function by moving down its gradient $\nabla_{\boldsymbol{x}} f(\boldsymbol{x})$, however, if colinear with the gradient of the constraint $\nabla_{\boldsymbol{x}} c(\boldsymbol{x})$ there will be no valid step $\boldsymbol{s}_x$ that both decrease the cost $f(\boldsymbol{x})$ and remains on the constraint $c(\boldsymbol{x})$.

Therefore at the constrained minimum $\boldsymbol{x}^*$ the gradient of the cost function will be equal to the gradient of the constraint multiplied by some linear scalar $\nabla_{\boldsymbol{x}} f(\boldsymbol{x}) = \lambda \nabla_{\boldsymbol{x}} c(\boldsymbol{x})$. Expanding this understanding to higher dimensions, the gradients of multiple constraints will form a hyper-plane to which any valid step $\boldsymbol{s}_x$ must be orthogonal. At the constrained minimum, $\boldsymbol{x}^*$ the gradient of the cost function $\nabla_{\boldsymbol{x}} f(\boldsymbol{x})$ will lie completely

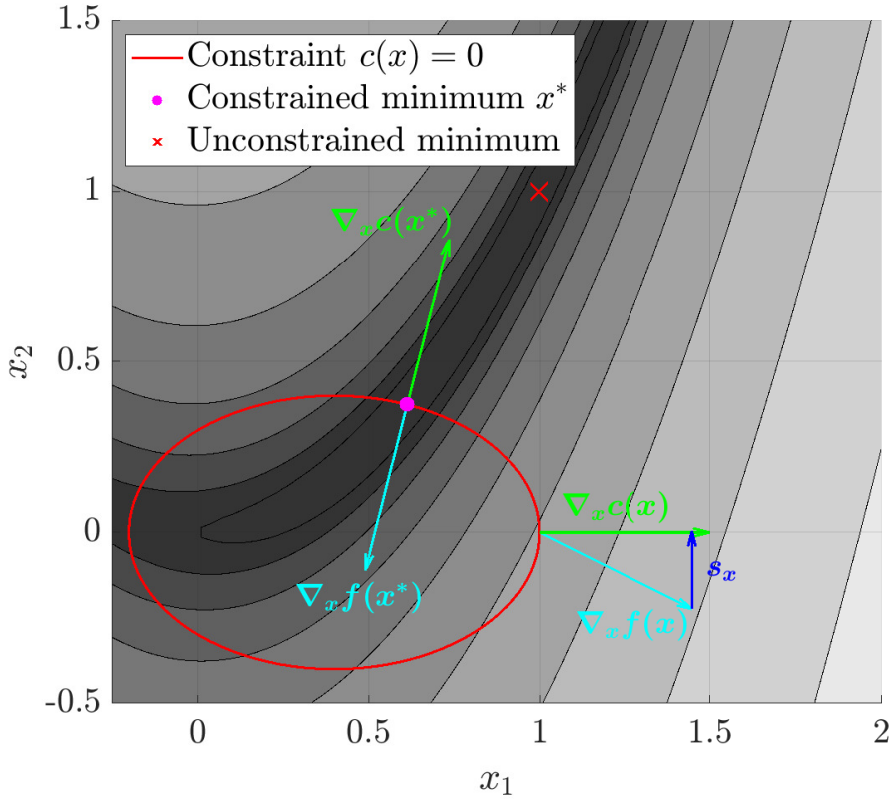


Figure D.3: Example of an equality constraint $c(x_1, x_2) = 2.78(x_1 - 0.4)^2 + 6.25x_2^2 - 1$. Rosenbrock function: $f(x_1, x_2) = (1 - x_1)^2 + 100(x_2 - x_1^2)^2$

within this plane and is, therefore, some linear combination of the gradients of the constraints $c_i(\boldsymbol{x})$ [191]:

$$\nabla_x f(\boldsymbol{x}^*) = -\sum_i \lambda_i \nabla_x c_i(\boldsymbol{x}^*) \tag{D.20}$$

where λ_i are known as *Lagrangian multipliers*. The equality-constrained optimisation problem defined in Eq. (D.19) is therefore solved by forming an augmented cost function known as the *Lagrangian*, not to be confused with the Lagrangian used in Lagrangian dynamics (Eq. (C.7b)) [191]:

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda}) = f(\boldsymbol{x}) + \sum_i \lambda_i c_i(\boldsymbol{x}) = f(\boldsymbol{x}) + \boldsymbol{\lambda}^\top \boldsymbol{c}(\boldsymbol{x}) \tag{D.21}$$

where $\boldsymbol{\lambda} = [\lambda_1 \ \lambda_2 \ \cdots \ \lambda_i]^\top$ is a vector of lagrangian multipliers, and $\boldsymbol{c}(\boldsymbol{x}) = [c_1(\boldsymbol{x}) \ c_2(\boldsymbol{x}) \ \cdots \ c_i(\boldsymbol{x})]^\top$ is a vector of all equality constraints. The constrained optimisation problem defined in Eq. (D.19) can now be posed as the unconstrained optimisation of the Lagrangian:

$$\min_{\boldsymbol{x},\boldsymbol{\lambda}} \mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda}) \tag{D.22}$$

where the Lagrangian multipliers $\boldsymbol{\lambda}$ are now independent variables that must be optimised for in addition to $\boldsymbol{x}$. Taking the partial derivatives of Eq. (D.21) and setting them to zero, the first-order Karush–Kuhn–Tucker (KKT) conditions can be formed as a system of equations [191]:

$$\begin{bmatrix} \nabla_{\boldsymbol{x}}\mathcal{L} \\ \nabla_{\boldsymbol{\lambda}}\mathcal{L} \end{bmatrix} = \begin{bmatrix} \nabla_{\boldsymbol{x}} f(\boldsymbol{x}) + [\nabla_{\boldsymbol{x}}\boldsymbol{c}(\boldsymbol{x})]^\top \boldsymbol{\lambda} \\ \boldsymbol{c}(\boldsymbol{x}) \end{bmatrix} = 0 \tag{D.23}$$

where $[\nabla_{\boldsymbol{x}}\boldsymbol{c}(\boldsymbol{x})]$ is the Jacobian (Eq. (A.10)) of the constraints w.r.t $\boldsymbol{x}$. It can be seen that the top row of Eq. (D.23) gives the same condition for optimality that was demonstrated in Fig. D.3 and Eq. (D.20), and the bottom row of Eq. (D.23) re-states that the equality constraints must be satisfied. Newton's method can now be applied to find the roots of the KKT conditions by first approximating Eq. (D.23) using a first-order Taylor series (Eq. (D.6)):

$$\underbrace{\begin{bmatrix} \nabla_{\boldsymbol{x}} f(\boldsymbol{x}) + [\nabla_{\boldsymbol{x}}\boldsymbol{c}(\boldsymbol{x})]^\top \boldsymbol{\lambda} \\ \boldsymbol{c}(\boldsymbol{x}) \end{bmatrix}}_{f(\boldsymbol{x}_0)} + \underbrace{\begin{bmatrix} [\nabla^2_{\boldsymbol{xx}}\mathcal{L}] & [\nabla_{\boldsymbol{x}}\boldsymbol{c}(\boldsymbol{x})]^\top \\ [\nabla_{\boldsymbol{x}}\boldsymbol{c}(\boldsymbol{x})] & 0 \end{bmatrix}}_{[\nabla_{\boldsymbol{x}}\boldsymbol{f}(\boldsymbol{x}_0)]} \underbrace{\begin{bmatrix} \boldsymbol{s}_x \\ \boldsymbol{s}_\lambda \end{bmatrix}}_{\boldsymbol{x}-\boldsymbol{x}_0} = 0 \tag{D.24}$$

where $\boldsymbol{s}_x$ and $\boldsymbol{s}_\lambda$ are steps in $\boldsymbol{x}$ and λ respectively, and $\left[\nabla^2_{\boldsymbol{xx}}\mathcal{L}\right]$ is the Hessian (Eq. (A.11)) of the Lagrangian $\mathbf{H}_{\boldsymbol{x}}(\mathcal{L})$. Solving Eq. (D.24) for $\boldsymbol{s}_x$ and $\boldsymbol{s}_\lambda$ the updates can be applied to form new iterates:

$$\begin{bmatrix} \boldsymbol{x}_{n+1} \\ \boldsymbol{\lambda}_{n+1} \end{bmatrix} = \begin{bmatrix} \boldsymbol{x}_n \\ \boldsymbol{\lambda}_n \end{bmatrix} + \begin{bmatrix} \boldsymbol{s}_x \\ \boldsymbol{s}_\lambda \end{bmatrix} \tag{D.25}$$

through iteratively estimating the cost function $f(\boldsymbol{x})$ as a quadratic and stepping into its minimum. The same solution can be found by re-framing the problem as a series of Sequential *quadratic problems (QPs)*, each of the form:

$$\min_{\boldsymbol{s}_x} \frac{1}{2}\boldsymbol{s}_x^\top \mathbf{Q}\boldsymbol{s}_x + \boldsymbol{q}^\top \boldsymbol{s}_x \tag{D.26a}$$

$$\text{subject to } \mathbf{A}\boldsymbol{s}_x + \boldsymbol{b} = 0 \tag{D.26b}$$

for which the solution can be found using:

$$\begin{bmatrix} \mathbf{Q} & \mathbf{A}^\top \\ \mathbf{A} & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{s}_x \\ \boldsymbol{s}_\lambda \end{bmatrix} + \begin{bmatrix} \boldsymbol{q} \\ \boldsymbol{b} \end{bmatrix} = 0 \tag{D.27}$$

where the QP is formed from the original Lagrangian (Eq. (D.21)) of the equality-constrained optimisation problem:

$$\mathbf{Q} = \left[\nabla^2_{xx}\mathcal{L}(\boldsymbol{x}_n, \boldsymbol{\lambda}_n)\right] = \mathbf{H}_{\boldsymbol{x}}\left\{\mathcal{L}(\boldsymbol{x}_n, \boldsymbol{\lambda}_n)\right\} \tag{D.28a}$$

$$\mathbf{A} = \left[\nabla_{\boldsymbol{x}}\boldsymbol{c}(\boldsymbol{x})\right] = \mathbf{J}_{\boldsymbol{x}}\left\{\boldsymbol{c}(\boldsymbol{x}_n)\right\} \tag{D.28b}$$

$$\boldsymbol{q} = \nabla_{\boldsymbol{x}}\mathcal{L}(\boldsymbol{x}_n, \boldsymbol{\lambda}_n) \tag{D.28c}$$

$$\boldsymbol{b} = \boldsymbol{c}(\boldsymbol{x}_n) \tag{D.28d}$$

The Lagrangian is therefore estimated at each iteration $\boldsymbol{x}_n$ as a quadratic, and the constraints are represented as linear approximations, i.e. second-order and first-order Taylor series, respectively.

D.2.2 | Inequality Constraints

Many optimisation problems, however, consist of a combination of both equality and *inequality* constraints and can be defined generally as:

$$\min_{\boldsymbol{x}} \quad f(\boldsymbol{x}) \tag{D.29a}$$

$$\text{subject to} \quad c_i(\boldsymbol{x}) = 0, \quad i \in \mathcal{E} \tag{D.29b}$$

$$c_i(\boldsymbol{x}) \leq 0, \quad i \in \mathcal{I} \tag{D.29c}$$

where $\mathcal{E}$ and $\mathcal{I}$ are the sets of equality and inequality constraints respectively. Previously, the equality constraints were implemented by adding them to a Lagrangian (Eq. (D.21)) with associated Lagrangian multipliers, which was then minimised. If all the inequality constraints are included in the Lagrangian, with associated Lagrangian multipliers σ_i, then the first KKT condition in Eq. (D.23) can be given as:

$$\nabla_{\boldsymbol{x}} f(\boldsymbol{x}^*) + \sum_{i\in\mathcal{E}} \lambda_i \nabla_{\boldsymbol{x}} c_i(\boldsymbol{x}^*) + \sum_{i\in\mathcal{I}} \sigma_i \nabla_{\boldsymbol{x}} c_i(\boldsymbol{x}^*) = 0 \tag{D.30}$$

If the optimal solution $\boldsymbol{x}^*$ lies on the boundary of all the inequality constraints, then they effectively act the same as equality constraints ($c_i(\boldsymbol{x}) = 0$), such as that illustrated in Fig. D.3. Eq. (D.20) would therefore imply that the KKT condition in Eq. (D.30) is satisfied. However, if the optimal solution $\boldsymbol{x}^*$ is within an inequality constraint ($c_i(\boldsymbol{x}) < 0$), then its gradient will not contribute to the linear combination in Eq. (D.20), thus requiring that its associated Lagrangian multiplier $\sigma_i = 0$. This rationale leads to the additional KKT condition known as *complimentary slackness* [191]:

$$\sigma_i(\boldsymbol{x}^*)c_i(\boldsymbol{x}^*) = 0 \quad i \in \mathcal{I} \tag{D.31}$$

which implies that inequality constraints at the optimal solution $\boldsymbol{x}^*$ are either *active* ($c_i(\boldsymbol{x}^*) = 0$) or *inactive* ($\sigma_i(\boldsymbol{x}^*) = 0$). It is, however, impossible to know *a priori* whether any particular inequality constraint is active or inactive at the optimal solution.

A concept known as the *active-set* [191, 192] can be implemented to allow SQP to solve optimisation problems with inequality constraints. A subset of the inequality constraints $\mathcal{I}$, which are considered to be active at any point during the SQP process, is defined as the *working-set* $\mathcal{W}$. The inequality constraints within the working-set are considered active, and therefore can be implemented in a Lagrangian similar to equality constraints:

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\sigma}) = f(\boldsymbol{x}) + \sum_{i \in \mathcal{E}} \lambda_i c_i(\boldsymbol{x}) + \sum_{i \in \mathcal{W}} \sigma_i c_i(\boldsymbol{x}) = f(\boldsymbol{x}) + \boldsymbol{\lambda}^\top \boldsymbol{c}_{eq}(\boldsymbol{x}) + \boldsymbol{\sigma}^\top \boldsymbol{c}_w(\boldsymbol{x}) \tag{D.32}$$

where $\boldsymbol{c}_w$ is a vector containing the current working set of inequality constraints. A QP can be used to solve this surrogate problem:

$$\min_{\boldsymbol{s}_x} \quad \frac{1}{2} \boldsymbol{s}_x^\top \mathbf{Q} \boldsymbol{s}_x + \boldsymbol{q}^\top \boldsymbol{s}_x \tag{D.33a}$$

$$\text{subject to} \quad \mathbf{A}\boldsymbol{s}_x + \boldsymbol{b} = 0 \tag{D.33b}$$

$$\mathbf{C}_w \boldsymbol{s}_x + \boldsymbol{d}_w = 0 \tag{D.33c}$$

where $\mathbf{C}_w$ and $\boldsymbol{d}_w$ are used to construct equality constraints from the working-set of inequality constraints. A solution to this QP can be found:

$$\begin{bmatrix} \mathbf{Q} & \mathbf{A}^\top & \mathbf{C}_w^\top \\ \mathbf{A} & 0 & 0 \\ \mathbf{C}_w & 0 & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{s}_x \\ \boldsymbol{s}_\lambda \\ \boldsymbol{s}_\sigma \end{bmatrix} + \begin{bmatrix} \boldsymbol{q} \\ \boldsymbol{b} \\ \boldsymbol{d}_w \end{bmatrix} = 0 \tag{D.34}$$

The QP is formed from Eq. (D.32):

$$\mathbf{Q} = \left[\nabla^2_{\boldsymbol{x}\boldsymbol{x}} \mathcal{L}(\boldsymbol{x}_n, \boldsymbol{\lambda}_n, \boldsymbol{\sigma}_n) \right] = \mathbf{H}_{\boldsymbol{x}} \left\{ \mathcal{L}(\boldsymbol{x}_n, \boldsymbol{\lambda}_n, \boldsymbol{\sigma}_n) \right\} \tag{D.35a}$$

$$\mathbf{A} = \left[\nabla_{\boldsymbol{x}} \boldsymbol{c}_{eq}(\boldsymbol{x}) \right] = \mathbf{J}_{\boldsymbol{x}} \left\{ \boldsymbol{c}_{eq}(\boldsymbol{x}_n) \right\} \tag{D.35b}$$

$$\mathbf{C}_w = \left[\nabla_{\boldsymbol{x}} \boldsymbol{c}_w(\boldsymbol{x}) \right] = \mathbf{J}_{\boldsymbol{x}} \left\{ \boldsymbol{c}_w(\boldsymbol{x}_n) \right\} \tag{D.35c}$$

$$\boldsymbol{q} = \nabla_{\boldsymbol{x}} \mathcal{L}(\boldsymbol{x}_n, \boldsymbol{\lambda}_n, \boldsymbol{\sigma}_n) \tag{D.35d}$$

$$\boldsymbol{b} = \boldsymbol{c}_{eq}(\boldsymbol{x}_n) \tag{D.35e}$$

$$\boldsymbol{d}_w = \boldsymbol{c}_w(\boldsymbol{x}_n) \tag{D.35f}$$

If the step $\boldsymbol{s}_x$ obtained from the solution to the QP would violate any of the inactive inequality constraints ($i \notin \mathcal{W}$) then the step must be scaled to place the new iterate

$\boldsymbol{x}_{n+1}$ on its boundary. This inequality is then considered active, added to the working set $\mathcal{W}$, a new QP is formed (Eqs. (D.33) and (D.35)), and solved (Eq. (D.34)).

Previously it was shown that for equality constraints, or active inequality constraints in this case too, the feasible directions for $\boldsymbol{s}_x$ were limited to directions orthogonal to the gradients of the constraints $\nabla_{\boldsymbol{x}} c_i(\boldsymbol{x})$. This led to the condition in Eq. (D.20) that at the optimal solution $\boldsymbol{x}^*$, the gradients of the cost function and constraints must be parallel. Furthermore, in the case of the active inequality constraints, each of the constraint gradients must be pointing in the same direction as the cost function, otherwise, there exist feasible directions for $\boldsymbol{s}_x$ which decrease the cost function $f(\boldsymbol{x})$ while still satisfying $c_i(\boldsymbol{x}) \leq 0$. This leads to the final KKT condition:

$$\sigma_i(\boldsymbol{x}^*) \geq 0 \quad i \in \mathcal{I} \tag{D.36}$$

Therefore, during the solution process of the QP, if $\sigma_i < 0$, then the associated constraint in the working set has *loosened*, meaning that there now exists some cost-decreasing feasible directions for $\boldsymbol{s}_x$ and should be made inactive. This is accomplished simply by removing the loosened inequality constraint from the working-set $\mathcal{W}$, reforming the QP (Eqs. (D.33) and (D.35)), and solving it (Eq. (D.34)).

This process of modifying the working set and solving equality-constrained QPs continues until the solution satisfies all KKT conditions. The iterate $\boldsymbol{x}_{n+1}$, associated Lagrangian multipliers $\boldsymbol{\lambda}_{n+1}$ and $\boldsymbol{\sigma}_{n+1}$, and working set $\mathcal{W}$ are then accepted as the solution to the current inequality constrained QP. Using Eqs. (D.33) to (D.35), a new inequality-constrained QP can be formed using the working set from the previously solved inequality-constrained QP. Iteratively solving inequality-constrained QPs continues until the solution satisfies the KKT conditions of the original nonlinear problem defined in Eq. (D.29). The SQP implementation used in this thesis is the *MATLAB* `fmincon()` function [193], using the in-built *active-set* algorithm specifically. This implementation also includes a *line-search* subroutine to ensure no local minima are stepped over, and a BFGS Hessian estimate to improve computational efficiency.

D.3 | Runge-Kutta Numerical Integration

The mathematical models derived through the methods outlined in Appendix C will be in the form of ODEs. In the case of some LTI systems, application of the Laplace transform and inverse transform can allow for these ODEs to be solved [194, p.93] for periodic and discontinuous inputs. More complex and nonlinear ODEs with arbitrary inputs can instead be solved through numerical integration techniques, i.e. forward simulation. Many numerical integration approaches exist [59], however, one of the

most commonly utilised classes of numerical integration methods are the *Runge-Kutta* methods. Runge-Kutta methods are characterised by a *weighted* sum of multiple model evaluations occurring within one timestep, which are utilised to match the Taylor series expansion of the exact solution [169, p.153]. The number of model evaluations a specific Runge-Kutta method utilises is called its *order*, with higher-order methods being more accurate but requiring additional model evaluations. For example, forward and backward Euler methods are first-order Runge-Kutta methods [194, p.193-194].

To illustrate the application of Runge-Kutta numerical integration methods, assume that the state derivative of the dynamic system is a function of time and the current state of the system:

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(t, \boldsymbol{x}) \tag{D.37}$$

where $\boldsymbol{x}$ and $\dot{\boldsymbol{x}}$ are the state and state derivative respectively, t is time, and $\boldsymbol{f}()$ is the set of ODEs describing the system dynamics. The general form of an r^{th} order Runge-Kutta method can be given as [194, p.194]:

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \Delta t \sum_{i=1}^{r} b_i \boldsymbol{k}_i \tag{D.38}$$

where $\boldsymbol{x}_{k+1}$ is the state of the dynamic system at the next timestep, $\boldsymbol{x}_k$ is the state of the system at the current timestep, Δt is the integration step size, $\boldsymbol{k}_i$ are intermediate model evaluations with b_i associated weights.

D.3.1 | Explicit Runge-Kutta Methods (ERK)

ERK methods are closed-form, with each model evaluation $\boldsymbol{k}_i$ only dependent on the current state $\boldsymbol{x}_{k+1}$ and previous model evaluations. The i^{th} model evaluation of an ERK method can be given as:

$$\boldsymbol{k}_i = \boldsymbol{f}\left(t_k + c_i \Delta t, \boldsymbol{x}_k + \Delta t \sum_{j=1}^{i-1} a_{ij} \boldsymbol{k}_j\right) \tag{D.39}$$

where c_i and a_{ij} are additional weights applied to the timestep and previous model evaluations respectively. The weights b_i, c_i and a_{ij} for an ERK method can be expressed concisely within a *Butcher* table:

$$\begin{array}{c|cccccc} 0 & & & & & & \\ c_2 & a_{21} & & & & & \\ c_3 & a_{31} & a_{32} & & & & \\ \vdots & \vdots & & \ddots & & & \\ c_r & a_{r1} & a_{r2} & \cdots & a_{r,r-1} & & \\ \hline & b_1 & b_2 & \cdots & b_{r-1} & b_r \end{array} \tag{D.40}$$

A property of ERK methods is that they produce triangular *butcher* tables with $a_{ij} = 0$ for $j \geq i$. One of the most common ERK methods is the RK4, commonly referred to simply as *the Runge-Kutta method*, and has the *butcher* table [194, p.195]:

$$\begin{array}{c|cccc} 0 & & & & \\ \frac{1}{2} & \frac{1}{2} & & & \\ \frac{1}{2} & 0 & \frac{1}{2} & & \\ 1 & 0 & 0 & 1 & \\ \hline & \frac{1}{6} & \frac{1}{3} & \frac{1}{3} & \frac{1}{6} \end{array} \tag{D.41}$$

giving the four intermediate model evaluations:

$$\boldsymbol{k}_1 = \boldsymbol{f}\left(t_k, \boldsymbol{x}_k\right) \tag{D.42a}$$

$$\boldsymbol{k}_2 = \boldsymbol{f}\left(t_k + \frac{1}{2}\Delta t, \boldsymbol{x}_k + \frac{1}{2}\Delta t \boldsymbol{k}_1\right) \tag{D.42b}$$

$$\boldsymbol{k}_3 = \boldsymbol{f}\left(t_k + \frac{1}{2}\Delta t, \boldsymbol{x}_k + \frac{1}{2}\Delta t \boldsymbol{k}_2\right) \tag{D.42c}$$

$$\boldsymbol{k}_4 = \boldsymbol{f}\left(t_k + \Delta t, \boldsymbol{x}_k + \Delta t \boldsymbol{k}_3\right) \tag{D.42d}$$

which can be substituted into Eq. (D.38) to obtain the numerical integration solution for a single timestep:

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \Delta t \left(\frac{1}{6}\boldsymbol{k}_1 + \frac{1}{3}\boldsymbol{k}_2 + \frac{1}{3}\boldsymbol{k}_3 + \frac{1}{6}\boldsymbol{k}_4\right) \tag{D.43}$$

ERK methods are very simple to implement due to their closed-form solution. This is, however, not the case with *implicit* Runge-Kutta methods.

D.3.2 | Implicit Runge-Kutta Methods (IRK)

Similar to ERK methods, IRK methods make multiple model evaluations within one timestep as given below:

$$\boldsymbol{k}_i = \boldsymbol{f}\left(t_k + c_i\Delta t, \boldsymbol{x}_k + \Delta t \sum_{j=1}^{r} a_{ij}\boldsymbol{k}_j\right) \tag{D.44}$$

which will depend on the current state $\boldsymbol{x}_k$ as before, but are not limited to depending only on previous intermediate model evaluations. IRK methods will therefore exhibit square *butcher* tables as can be seen below:

$$\begin{array}{c|cccc} c_1 & a_{11} & a_{12} & \cdots & a_{1r} \\ c_2 & a_{21} & a_{22} & \cdots & a_{2r} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ c_r & a_{r1} & a_{r2} & \cdots & a_{rr} \\ \hline & b_1 & b_2 & \cdots & b_r \end{array} \tag{D.45}$$

The specific *butcher* table for the implicit trapezoidal method is given below [194, p.194]:

$$\begin{array}{c|cc} 0 & 0 & 0 \\ 1 & \frac{1}{2} & \frac{1}{2} \\ \hline & \frac{1}{2} & \frac{1}{2} \end{array} \tag{D.46}$$

with the intermediate model evaluations given as:

$$\boldsymbol{k}_1 = \boldsymbol{f}\left(t_k, \boldsymbol{x}_k\right) \tag{D.47a}$$

$$\boldsymbol{k}_2 = \boldsymbol{f}\left(t_k + \Delta t, \boldsymbol{x}_k + \frac{1}{2}\Delta t\left(\boldsymbol{k}_1 + \boldsymbol{k}_2\right)\right) \tag{D.47b}$$

resulting in the trapezoidal method state update equation:

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \frac{1}{2}\Delta t\left(\boldsymbol{f}\left(t_k, \boldsymbol{x}_k\right) + \boldsymbol{f}\left(t_{k+1}, \boldsymbol{x}_{k+1}\right)\right) \tag{D.48}$$

It can be seen from Eq. (D.48) that the state at the next timestep $\boldsymbol{x}_{k+1}$, which is unknown, appears on both sides of the equation. Therefore, no closed-form solution exists, and an iterative numerical solution process must be employed. This can be achieved by first defining a residual function $\delta\boldsymbol{x}_{k+1}$:

$$\delta\boldsymbol{x}_{k+1} = \boldsymbol{x}_{k+1} - \boldsymbol{x}_k - \frac{1}{2}\Delta t\left(\boldsymbol{f}\left(t_k, \boldsymbol{x}_k\right) + \boldsymbol{f}\left(t_{k+1}, \boldsymbol{x}_{k+1}\right)\right) \tag{D.49}$$

with the numerical integration solution $\boldsymbol{x}_{k+1}$ being found such that $\delta\boldsymbol{x}_{k+1} = 0$. This can be achieved using the NR numerical method, as laid out in Appendix D.1.1, however, for simplicity, the *MATLAB* function `fsolve()` is used in this thesis, which utilises the *trust region dogleg* algorithm.

Appendix E | Derivation of Rigid-Body Mathematical Models

The dynamics of rigid bodies, both torque-free and with external torques, are well known and can be found in Euler's rotational equations [81, p.84]. However, when momentum exchange devices, such as reaction wheels, are utilised for attitude control, additional internal dynamics must be considered. Therefore, for completeness, the rigid-body dynamics of satellites with various actuator inputs are derived in this appendix from first principles using the *Lagrangian* dynamics approach (Appendix C.2) herein.

E.1 | External Torques

The kinetic and potential energies, $\mathcal{T}$ and $\mathcal{V}$ respectively, of a rotating rigid-body can be defined in vector form:

$$\mathcal{T} = \frac{1}{2}{\boldsymbol{\omega}^B_{B/I}}^\top \mathbf{I}_{tot} \boldsymbol{\omega}^B_{B/I} \tag{E.1a}$$

$$\mathcal{V} = 0 \tag{E.1b}$$

where $\mathbf{I}_{tot}$ is the total inertia of the rigid-body, and $\boldsymbol{\omega}^B_{B/I}$ is the angular velocity of the satellite's body frame (B) relative to the inertial frame (I). The Lagrangian can then be formed:

$$\mathcal{L} = \frac{1}{2}{\boldsymbol{\omega}^B_{B/I}}^\top \mathbf{I}_{tot} \boldsymbol{\omega}^B_{B/I} \tag{E.2}$$

which can be used in the following *Euler-Lagrange* equation to obtain the differential equation of the rigid-body dynamics:

$$\frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \boldsymbol{\omega}^B_{B/I}}\right) = \boldsymbol{\tau}^B_{ext} \tag{E.3}$$

where $\boldsymbol{\tau}_{ext}$ is the external torque applied to the rigid-body. Finding the partial derivative, the ODE can then be found while remembering the transport theorem (Ap-

pendix C.2):

$$\frac{d}{dt}\left(\mathbf{I}_{tot}\boldsymbol{\omega}_{B/I}^{B}\right) = \cancelto{0}{\dot{\mathbf{I}}_{tot}\boldsymbol{\omega}_{B/I}^{B}} + \mathbf{I}_{hub}\dot{\boldsymbol{\omega}}_{B/I}^{B} + \left[\boldsymbol{\omega}_{B/I}^{B}\right]_{\times}\left(\mathbf{I}_{tot}\boldsymbol{\omega}_{B/I}^{B}\right) = \boldsymbol{\tau}_{ext}^{B} \tag{E.4}$$

where the derivative of the inertia $\dot{\mathbf{I}}_{tot}$ can be assumed zero for a rigid-body. Letting the input to the system $\boldsymbol{u}$ be the external torque applied to the rigid-body $\boldsymbol{\tau}_{ext}^{B}$, and simplifying notation, the well-defined set of *Euler's rotational equations* in vector form can be given:

$$\mathbf{I}_{tot}\dot{\boldsymbol{\omega}}_{B} + \left[\boldsymbol{\omega}_{B}\right]_{\times}\left(\mathbf{I}_{tot}\boldsymbol{\omega}_{B}\right) = \boldsymbol{u} \tag{E.5}$$

Eq. (E.5) describes the rigid-body dynamics for a general external torque. In many cases, it may be more prevalent to include the dynamics of the specific actuator being used by the satellite.

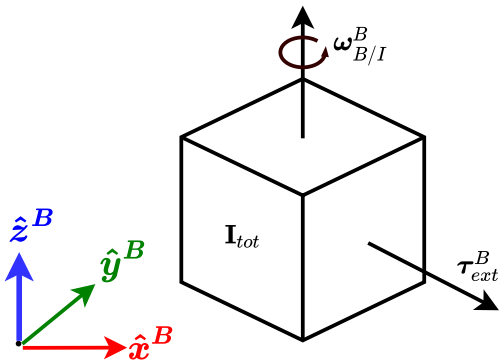


Figure E.1: Rigid-body satellite with external torques

E.2 | Magnetic Torques

A magnetorquer produces an external torque on the satellite by interacting with the local magnetic field, usually limiting use to LEO. Considering the input to the system $\boldsymbol{u}$ to be the magnetic dipole, the torque produced will therefore be:

$$\boldsymbol{\tau}_{ext}^{B} = \boldsymbol{u} \times \boldsymbol{b}_{mag}^{B} = \left[\boldsymbol{b}_{mag}^{B}\right]_{\times}^{\top}\boldsymbol{u} \tag{E.6}$$

which can be substituted into *Euler's rotational equations* Eq. (E.5) to give:

$$\mathbf{I}_{tot}\dot{\boldsymbol{\omega}}_{B} + \left[\boldsymbol{\omega}_{B}\right]_{\times}\left(\mathbf{I}_{tot}\boldsymbol{\omega}_{B}\right) = \left[\boldsymbol{b}_{mag}^{B}\right]_{\times}^{\top}\boldsymbol{u} \tag{E.7}$$

where it can be seen by the cross product $[\boldsymbol{b}_{mag}^{B}]_{\times}^{\top}$ that actuation is lost about the direction $\hat{\boldsymbol{b}}_{mag}^{B}$. The torques produced by a magnetorquer will always be limited to a plane defined by the direction of the magnetic field $\hat{\boldsymbol{b}}_{mag}^{B}$ as shown in Fig. E.2.

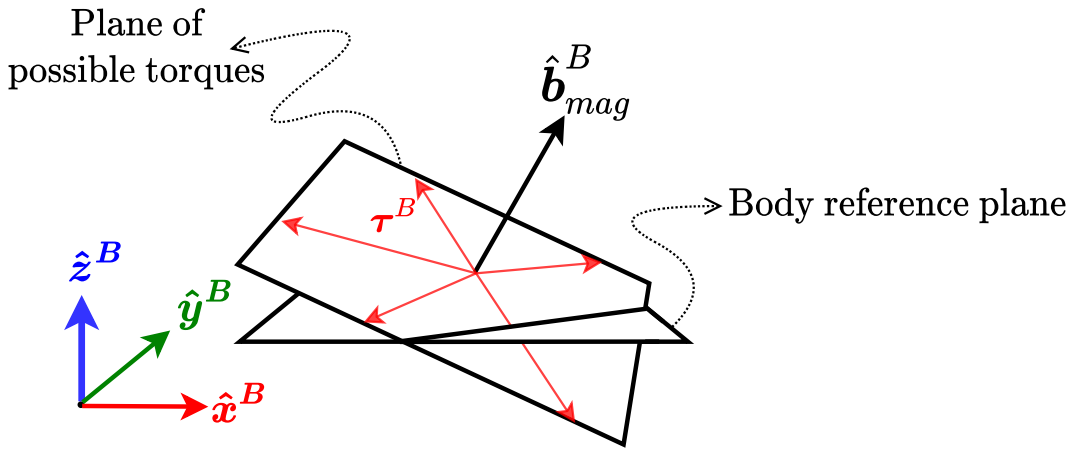


Figure E.2: Magnetorquer: plane of possible torques

E.3 | Reaction Wheels

One of the most common attitude control actuators, for satellites of all sizes, are reaction wheels, as illustrated in Fig. E.3. Angular momentum is exchanged between the reaction wheels and the satellite hub to produce torques and rotate the satellite. There will also be additional cross-coupling between the angular momentum of the satellite hub and reaction wheels.

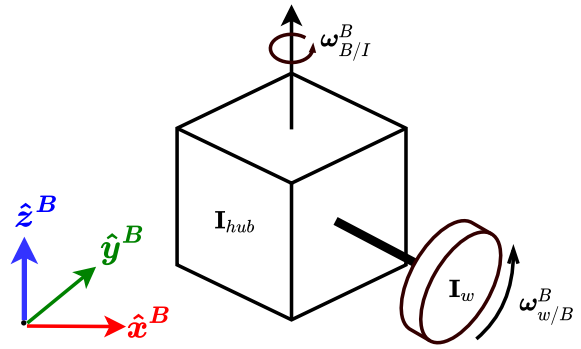


Figure E.3: Rigid-body satellite with reaction wheels

To properly capture the full dynamics of a satellite with reaction-wheel actuators, a suitable *Lagrangian* can be formed:

$$\mathcal{L} = \mathcal{T} - \mathcal{V} = \underbrace{\frac{1}{2}{\omega^B_{B/I}}^\top \mathbf{I}_{hub}\omega^B_{B/I}}_{\substack{\text{rotational energy of} \\ \text{just the main hub}}} + \underbrace{\frac{1}{2}\left(\omega^B_{w/B}+\omega^B_{B/I}\right)^\top \mathbf{I}_w\left(\omega^B_{w/B}+\omega^B_{B/I}\right)}_{\text{total rotational kinetic energy of reaction wheels}} \tag{E.8}$$

$$= \frac{1}{2}{\omega^B_{B/I}}^\top \mathbf{I}_{hub}\omega^B_{B/I} + \frac{1}{2}{\omega^B_{w/B}}^\top \mathbf{I}_w\omega^B_{w/B} + {\omega^B_{w/B}}^\top \mathbf{I}_w\omega^B_{B/I}$$

where $\mathbf{I}_{hub}$ is the inertia of the main body of the satellite, $\boldsymbol{\omega}^B_{w/B}$ is the angular velocity of the reaction wheel-set relative to the main hub, and $\mathbf{I}_w$ is the inertia of the reaction wheel-set. The total inertia $\mathbf{I}_{tot}$ can also be formed as:

$$\mathbf{I}_{tot} = \mathbf{I}_{hub} + \mathbf{I}_w \tag{E.9}$$

Forming and completing the *Euler-Lagrange* equation for the rigid-body dynamics gives:

$$\frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \boldsymbol{\omega}^B_{B/I}}\right) = \mathbf{I}_{tot}\dot{\boldsymbol{\omega}}^B_{B/I} + \mathbf{I}_w\dot{\boldsymbol{\omega}}^B_{w/B} + \left[\boldsymbol{\omega}^B_{B/I}\right]_\times \left(\mathbf{I}_{tot}\boldsymbol{\omega}^B_{B/I} + \mathbf{I}_w\boldsymbol{\omega}^B_{w/B}\right) = 0 \tag{E.10}$$

To simplify, the angular momentum of the reaction wheels $\boldsymbol{h}^B_{w/B}$ can be defined as:

$$\boldsymbol{h}^B_{w/B} = \mathbf{I}_w\boldsymbol{\omega}^B_{w/B} \tag{E.11}$$

and since the reaction wheels are controlled, the change in their angular momentum will be caused by an input control torque $\boldsymbol{u}$:

$$\mathbf{I}_w\dot{\boldsymbol{\omega}}^B_{w/B} = \dot{\boldsymbol{h}}^B_{w/B} = \boldsymbol{u} \tag{E.12}$$

It is common for more than three reaction wheels, one per axis, to be used to ensure redundancy. An actuator distribution matrix $\mathbf{A}_w$ can be added to the model to allow for N_u number of reaction wheels to be represented in the model. Simplifying some of the notation, this gives the modified *Euler rotational equations*:

$$\mathbf{I}_{tot}\dot{\boldsymbol{\omega}}_B + [\boldsymbol{\omega}_B]_\times \left(\mathbf{I}_{tot}\boldsymbol{\omega}_B + \mathbf{A}_w\boldsymbol{h}_w\right) = -\mathbf{A}_w\boldsymbol{u} \tag{E.13}$$

where the input $\boldsymbol{u}$ will be a $N_u \times 1$ vector of torques applied to each reaction wheel, and $\boldsymbol{h}_w$ will be a $N_u \times 1$ vector containing the angular momenta of each wheel. The combined inertia of all the reaction wheels $\mathbf{I}_w$ can be found using:

$$\mathbf{I}_w = \mathbf{A}_w\mathbf{I}_w^{\text{diag}}\mathbf{A}_w^\top \tag{E.14}$$

where $\mathbf{I}_w^{\text{diag}}$ is a diagonal matrix of each individual reaction wheel's inertia $\mathbf{I}_w^{\text{diag}} = \text{diag}(I_1, \cdots, I_{N_u})$.

E.3.1 | First-Order Actuator Response

The reaction wheel torques $\dot{\boldsymbol{h}}^B_w$ in Eq. (E.12) are assumed to occur instantaneously when the input $\boldsymbol{u}$ is applied. In many cases, this is a fair assumption; however, increased

accuracy can be achieved by modelling the reaction wheel's electro-mechanical system. One simple approach is assuming that the torque response acts as a first-order system [195], which in simplified notation is given as:

$$\ddot{\boldsymbol{h}}_w = \frac{\boldsymbol{u} - \dot{\boldsymbol{h}}_w}{T_w} \tag{E.15}$$

where $\ddot{\boldsymbol{h}}_w$ is the derivative of the reaction wheel torque, and T_w is the time constant of the torque response. The *Euler rotational equation* will therefore no longer be directly related to the input $\boldsymbol{u}$:

$$\mathbf{I}_{tot}\dot{\boldsymbol{\omega}}_B + [\boldsymbol{\omega}_B]_\times (\mathbf{I}_{tot}\boldsymbol{\omega}_B + \mathbf{A}_w \boldsymbol{h}_w) = -\mathbf{A}_w \dot{\boldsymbol{h}}_w \tag{E.16}$$

which effectively increases the overall differential order of the system by one.

Bibliography

[1] D. B. Yaden, J. Iwry, K. J. Slack, J. C. Eichstaedt, Y. Zhao, G. E. Vaillant, and A. B. Newberg, "The overview effect: awe and self-transcendent experience in space flight.," *Psychology of Consciousness: Theory, Research, and Practice*, vol. 3, no. 1, pp. 1–11, 2016. DOI: 10.1037/cns0000086.

[2] K. Whealan George, "The Economic Impacts of the Commercial Space Industry," *Space Policy*, vol. 47, pp. 181–186, 2019. DOI: 10.1016/j.spacepol.2018.12.003.

[3] G. Denis, D. Alary, X. Pasco, N. Pisot, D. Texier, and S. Toulza, "From new space to big space: How commercial space dream is becoming a reality," *Acta Astronautica*, vol. 166, pp. 431–443, 2020. DOI: 10.1016/j.actaastro.2019.08.031.

[4] J. Yang, P. Gong, R. Fu, M. Zhang, J. Chen, S. Liang, B. Xu, J. Shi, and R. Dickinson, "The role of satellite remote sensing in climate change studies," *Nature Climate Change*, vol. 3, no. 10, pp. 875–883, 2013. DOI: 10.1038/nclimate1908.

[5] K. Thangavel, D. Spiller, R. Sabatini, S. Amici, S. T. Sasidharan, H. Fayek, and P. Marzocca, *Autonomous Satellite Wildfire Detection Using Hyperspectral Imagery and Neural Networks: A Case Study on Australian Wildfire*, 2023. DOI: 10.3390/rs15030720.

[6] F. Kogan, W. Guo, and W. Yang, "Drought and food security prediction from NOAA new generation of operational satellites," *Geomatics, Natural Hazards and Risk*, vol. 10, no. 1, pp. 651–666, 2019. DOI: 10.1080/19475705.2018.1541257.

[7] L. Karthikeyan, I. Chawla, and A. K. Mishra, "A review of remote sensing applications in agriculture for food security: Crop growth and yield, irrigation, and crop losses," *Journal of Hydrology*, vol. 586, p. 124 905, 2020. DOI: 10.1016/j.jhydrol.2020.124905.

[8] H. Jones, "The recent large reduction in space launch cost," in *48th International Conference on Environmental Systems*, Albuquerque, NM, USA, 2018, pp. 2018–81. [Online]. Available: http://hdl.handle.net/2346/74082.

[9] R. E. Hodges, N. Chahat, D. J. Hoppe, and J. D. Vacchione, "A Deployable High-Gain Antenna Bound for Mars: Developing a new folded-panel reflectarray for

the first CubeSat mission to Mars," *IEEE Antennas and Propagation Magazine*, vol. 59, no. 2, pp. 39–49, 2017. DOI: 10.1109/MAP.2017.2655561.

[10] H. Heidt, J. Puig-Suari, A. Moore, S. Nakasuka, and R. Twiggs, "CubeSat: A new generation of picosatellite for education and industry low-cost space experimentation," in *14th Annual AIAA/USU Conference on Small Satellites*, Logan, UT, USA, 2000, SSC00-V–5. [Online]. Available: https://digitalcommons.usu.edu/smallsat/2000/All2000/32/.

[11] A. Poghosyan and A. Golkar, "CubeSat evolution: Analyzing CubeSat capabilities for conducting science missions," *Progress in Aerospace Sciences*, vol. 88, pp. 59–83, 2017. DOI: 10.1016/j.paerosci.2016.11.002.

[12] D. Selva and D. Krejci, "A survey and assessment of the capabilities of Cubesats for Earth observation," *Acta Astronautica*, vol. 74, pp. 50–68, 2012. DOI: 10.1016/j.actaastro.2011.12.014.

[13] J. F. Sauder, C. A. Gebara, and M. Arya, "A Survey of CubeSat Deployable Structures: The First Decade," in *AIAA Scitech 2021 Forum*, American Institute of Aeronautics and Astronautics, Jan. 2021. DOI: 10.2514/6.2021-1704.

[14] J. Champagne, S. Hansen, T. Newswander, and B. Crowther, "CubeSat image resolution capabilities with deployable optics and current imaging technology," in *28th Annual AIAA/USU Conference on Small Satellites*, Logan, UT, USA, 2014, SSC14-VII–2. [Online]. Available: https://digitalcommons.usu.edu/smallsat/2014/AdvTechII/10/.

[15] M. E. Dearborn, G. P. Andersen, O. Asmolova, R. L. Balthazor, M. G. McHarg, H. C. Nelson, T. S. Quiller, G. R. Wilson, T. J. Harvey, and T. W. Murphey, "A deployable membrane telescope payload for cubesats," *JoSS*, vol. 3, no. 1, pp. 253–264, 2014. [Online]. Available: https://jossonline.com/papers9dcc.html.

[16] S. Abulgasem, F. Tubbal, R. Raad, P. I. Theoharis, S. Lu, and S. Iranmanesh, "Antenna Designs for CubeSats: A Review," *IEEE Access*, vol. 9, pp. 45 289–45 324, 2021. DOI: 10.1109/ACCESS.2021.3066632.

[17] F. Santoni, F. Piergentili, S. Donati, M. Perelli, A. Negri, and M. Marino, "An innovative deployable solar panel system for Cubesats," *Acta Astronautica*, vol. 95, pp. 210–217, 2014. DOI: 10.1016/j.actaastro.2013.11.011.

[18] A. Solís-Santomé, G. Urriolagoitia-Sosa, B. Romero-Ángeles, C. R. Torres-San Miguel, J. J. Hernández-Gómez, I. Medina-Sánchez, C. Couder-Castañeda, J. I. Grageda-Arellano, and G. Urriolagoitia-Calderón, "Conceptual design and finite element method validation of a new type of self-locking hinge for deploy-

able CubeSat solar panels," *Advances in Mechanical Engineering*, vol. 11, no. 1, Jan. 2019. DOI: 10.1177/1687814018823116.

[19] R. L. Heacock, "The Voyager Spacecraft," *Proceedings of the Institution of Mechanical Engineers*, vol. 194, no. 1, pp. 211–224, Jun. 1980. DOI: 10.1243/PIME_PROC_1980_194_026_02.

[20] G. Gibbs and S. Sachdev, "Canada and the International Space Station program: Overview and status," *Acta Astronautica*, vol. 51, no. 1, pp. 591–600, 2002. DOI: 10.1016/S0094-5765(02)00077-2.

[21] D. Li, L. Zhong, W. Zhu, Z. Xu, Q. Tang, and W. Zhan, "A Survey of Space Robotic Technologies for On-Orbit Assembly," *Space: Science & Technology*, Aug. 2022. DOI: 10.34133/2022/9849170.

[22] W. Zhang, F. Li, J. Li, and Q. Cheng, "Review of On-Orbit Robotic Arm Active Debris Capture Removal Methods," *Aerospace*, vol. 10, no. 1, 2023. DOI: 10.3390/aerospace10010013.

[23] Y. Tsuda, O. Mori, R. Funase, H. Sawada, T. Yamamoto, T. Saiki, T. Endo, K. Yonekura, H. Hoshino, and J. Kawaguchi, "Achievement of IKAROS — Japanese deep space solar sail demonstration mission," *Acta Astronautica*, 7th IAA Symposium on Realistic Advanced Scientific Space MissionsAosta, Italy, July 2011, vol. 82, no. 2, pp. 183–188, 2013. DOI: 10.1016/j.actaastro.2012.03.032.

[24] D. A. Spencer, B. Betts, J. M. Bellardo, A. Diaz, B. Plante, and J. R. Mansell, "The LightSail 2 solar sailing technology demonstration," *Advances in Space Research*, vol. 67, no. 9, pp. 2878–2889, 2021. DOI: 10.1016/j.asr.2020.06.029.

[25] L. McNutt, L. Johnson, P. Kahn, J. Castillo-Rogez, and A. Frick, "Near-Earth Asteroid (NEA) Scout," in *AIAA SPACE 2014 Conference and Exposition*, American Institute of Aeronautics and Astronautics, Aug. 2014. DOI: doi:10.2514/6.2014-4435.

[26] L. Johnson, J. Everett, D. McKenzie, D. Tyler, D. Wallace, J. Newmark, D. Turse, M. Cannella, J. Wilson, and M. Feldman, "The NASA Solar Cruiser Mission-Solar Sail Propulsion Enabling Heliophysics Missions," in *36th Annual Small Satellite Conference*, 2022. [Online]. Available: https://digitalcommons.usu.edu/smallsat/2022/all2022/144/.

[27] P. E. Glaser, "Power from the sun: Its future," *Science*, vol. 162, no. 3856, pp. 857–861, 1968. DOI: 10.1126/science.162.3856.857.

[28] J. C. Mankins, "A fresh look at space solar power: New architectures, concepts and technologies," *Acta Astronautica*, vol. 41, no. 4, pp. 347–359, 1997. DOI: 10.1016/S0094-5765(98)00075-7.

[29] J. C. Mankins, "A Technical Overview of the "SunTower" Solar Power Satellite Concept," *Acta Astronautica*, vol. 50, no. 6, pp. 369–377, 2002. DOI: 10.1016/S0094-5765(01)00167-9.

[30] W. Seboldt, M. Klimke, M. Leipold, and N. Hanowski, "European Sail Tower SPS concept," *Acta Astronautica*, vol. 48, no. 5, pp. 785–792, 2001. DOI: 10.1016/S0094-5765(01)00046-7.

[31] Y. Yang, Y. Zhang, B. Duan, D. Wang, and X. Li, "A novel design project for space solar power station (SSPS-OMEGA)," *Acta Astronautica*, vol. 121, pp. 51–58, 2016. DOI: 10.1016/j.actaastro.2015.12.029.

[32] X. Chen, R. Yuan, J. Zhu, C. Liu, J. Li, and Q. Li, "Conceptual design and orbit-attitude coupled analyses of free-drift solar power satellite," *Advances in Space Research*, vol. 71, no. 12, pp. 5413–5430, Aug. 2023. DOI: 10.1016/j.asr.2023.02.039.

[33] A. Viale, O. Çelik, T. Oderinwale, L. Sulbhewar, and C. R. McInnes, "A reference architecture for orbiting solar reflectors to enhance terrestrial solar power plant output," *Advances in Space Research*, 2023. DOI: 10.1016/j.asr.2023.05.037.

[34] K. W. Billman, W. P. Gilbreath, and S. W. Bowen, "Introductory assessment of orbiting reflections for terrestrial power generation," NASA, Tech. Rep., 1977. [Online]. Available: https://ntrs.nasa.gov/citations/19790014444.

[35] L. M. Fraas, G. A. Landis, and A. Palisoc, "Mirror satellites in polar orbit beaming sunlight to terrestrial solar fields at dawn and dusk," in *39th IEEE Photovoltaic Specialists Conference (PVSC)*, 2013, pp. 2764–2769. DOI: 10.1109/PVSC.2013.6745046.

[36] W. M. Robbins Jr, "The feasibility of an orbiting 1500-meter radiotelescope," Tech. Rep., 1967. [Online]. Available: https://ntrs.nasa.gov/citations/19670018291.

[37] L. J. DeLucas, "International space station," *Acta Astronautica*, vol. 38, no. 4, pp. 613–619, 1996. DOI: 10.1016/0094-5765(96)00056-2.

[38] S. Fuller, E. Lehnhardt, C. Zaid, and K. Halloran, "Gateway program status and overview," *Journal of Space Safety Engineering*, vol. 9, no. 4, pp. 625–628, 2022. DOI: 10.1016/j.jsse.2022.07.008.

[39] L. Puig, A. Barton, and N. Rando, "A review on large deployable structures for astrophysics missions," *Acta Astronautica*, vol. 67, no. 1, pp. 12–26, 2010. DOI: 10.1016/j.actaastro.2010.02.021.

[40] J.-C. Angevain, A. Ihle, G. Rodrigues, and J. Santiago-Prowald, "Large Deployable Spaceborne Reflector Antennas in Europe: Progress Status and Perspec-

tives," in *13th European Conference on Antennas and Propagation (EuCAP)*, 2019, pp. 1–5. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8739491.

[41] R. E. Freeland and G. Bilyeu, "In-step inflatable antenna experiment," *Acta Astronautica*, vol. 30, pp. 29–40, 1993. DOI: 10.1016/0094-5765(93)90098-H.

[42] M. Schenk, A. D. Viquerat, K. A. Seffen, and S. D. Guest, "Review of Inflatable Booms for Deployable Space Structures: Packing and Rigidization," *Journal of Spacecraft and Rockets*, vol. 51, no. 3, pp. 762–778, Apr. 2014. DOI: 10.2514/1.A32598.

[43] R. P. Hoyt, J. Cushing, G. Jimmerson, J. Slostad, R. Dyer, and S. Alvarado, "SpiderFab: Process for on-orbit construction of kilometer-scale apertures," Tethers Unlimited, Inc., Tech. Rep., 2016. [Online]. Available: https://ntrs.nasa.gov/citations/20190001156.

[44] W. Doggett, J. Dorsey, T. Jones, M. Mikulas, J. Teter, and D. Paddock, "TriTruss: A New and Novel Structural Concept Enabling Modular Space Telescopes and Space Platforms," in *70th International Astronautical Congress*, Washington D.C., United States, 2019. [Online]. Available: https://ntrs.nasa.gov/citations/20200010330.

[45] Z. Xue, J. Liu, C. Wu, and Y. Tong, "Review of in-space assembly technologies," *Chinese Journal of Aeronautics*, vol. 34, no. 11, pp. 21–47, 2021. DOI: 10.1016/j.cja.2020.09.043.

[46] R. P. Hoyt, "SpiderFab: An Architecture for Self-Fabricating Space Systems," in *AIAA SPACE 2013 Conference and Exposition*, American Institute of Aeronautics and Astronautics, Sep. 2013. DOI: 10.2514/6.2013-5509.

[47] R. P. Hoyt, J. Cushing, J. Slostad, and G. Jimmerson, "TRUSSELATOR: On-Orbit Fabrication of High-Performance Composite Truss Structures," in *AIAA SPACE 2014 Conference and Exposition*, American Institute of Aeronautics and Astronautics, Aug. 2014. DOI: 10.2514/6.2014-4337.

[48] B. Levedahl, R. P. Hoyt, T. Silagy, J. Gorges, N. Britton, and J. Slostad, "Trusselator Technology for In-Situ Fabrication of Solar Array Support Structures," in *AIAA 2018 Spacecraft Structures Conference*, American Institute of Aeronautics and Astronautics, Jan. 2018. DOI: 10.2514/6.2018-2203.

[49] S. C. Joshi and A. A. Sheikh, "3D printing in aerospace and its long-term sustainability," *Virtual and Physical Prototyping*, vol. 10, no. 4, pp. 175–185, Oct. 2015. DOI: 10.1080/17452759.2015.1111519.

[50] P. G. Hastie, G. Bailet, and C. R. McInnes, "On-Orbit Manufacturing of Large Space Structures Using Solid Foams," in *72nd International Astronautical Congress*, Dubai, UAE, 2021. [Online]. Available: https://www.proceedings.com/62594.html.

[51] V. J. Modi, "Attitude Dynamics of Satellites with Flexible Appendages- A Brief Review," *Journal of Spacecraft and Rockets*, vol. 11, no. 11, pp. 743–751, 1974. DOI: 10.2514/3.62172.

[52] J. Deyst, "A survey of structural flexibility effects on spacecraft control systems," in *7th Aerospace Sciences Meeting*, 1969. DOI: 10.2514/6.1969-116.

[53] W. C. Pilkington, "Vehicle Motions as Inferred from Radio-Signal-Strength Records," Tech. Rep., 1958. [Online]. Available: https://ntrs.nasa.gov/citations/19650073999.

[54] B. Etkin and P. C. Hughes, "Explanation of the anomalous spin behavior of satellites with long, flexible antennae.," *Journal of Spacecraft and Rockets*, vol. 4, no. 9, pp. 1139–1145, Sep. 1967. DOI: 10.2514/3.29042.

[55] C. J. Burrows, J. A. Holtzman, S. M. Faber, P. Y. Bely, H. Hasan, C. R. Lynds, and D. Schroeder, "The imaging performance of the Hubble Space Telescope," *The Astrophysical Journal*, vol. 369, pp. L21–L25, 1991. [Online]. Available: https://adsabs.harvard.edu/full/1991ApJ...369L..21B/L000021.000.html.

[56] P. Y. Bely, O. L. Lupie, and J. L. Hershey, "Line-of-sight jitter of the Hubble Space Telescope," in *Proc.SPIE*, vol. 1945, Nov. 1993, pp. 55–61. DOI: 10.1117/12.158800.

[57] S. M. Anandakrishnan, C. T. Connor, S. Lee, E. Shade, J. Sills, J. R. Maly, and S. C. Pendleton, "Hubble Space Telescope solar array damper for improving control system stability," in *2000 IEEE Aerospace Conference*, vol. 4, 2000, pp. 261–276. DOI: 10.1109/AERO.2000.878438.

[58] *Servicing Mission 3B (SM3B)*. [Online]. Available: https://science.nasa.gov/mission/hubble/observatory/missions-to-hubble/servicing-mission-3b/#hds-sidebar-nav-3.

[59] F. E. Cellier and E. Kofman, *Continuous system simulation*, English. New York: Springer Science+Business Media Inc., 2006, ISBN: 0-387-26102-8.

[60] I. Moore, L. Sulbhewar, O. Çelik, and C. R. McInnes, "The Effects of Pointing Error Sources on Energy Delivery from Orbiting Solar Reflectors," *Acta Astronautica*, vol. 222, pp. 563–572, 2024. DOI: 10.1016/j.actaastro.2024.06.020.

[61] E. Stoneking, "Newton-Euler dynamic equations of motion for a multi-body spacecraft," in *AIAA Guidance, Navigation and Control Conference and Exhibit*, Hilton Head, South Carolina, 2007, pp. 2007–6441. DOI: 10.2514/6.2007-6441.

[62] S. Shahriari, S. Azadi, and M. M. Moghaddam, "An accurate and simple model for flexible satellites for three-dimensional studies," *Journal of Mechanical Science and Technology*, vol. 24, no. 6, pp. 1319–1327, 2010. DOI: 10.1007/s12206-010-0329-0.

[63] C. Allard, H. Schaub, and S. Piggott, "General Hinged Rigid-Body Dynamics Approximating First-Order Spacecraft Solar Panel Flexing," *Journal of Spacecraft and Rockets*, vol. 55, no. 5, pp. 1291–1299, 2018. DOI: 10.2514/1.A34125.

[64] G. Fracchia, J. D. Biggs, and M. Ceriotti, "Analytical low-jerk reorientation maneuvers for multi-body spacecraft structures," *Acta Astronautica*, vol. 178, pp. 1–14, 2021. DOI: 10.1016/j.actaastro.2020.08.020.

[65] S. Zhang, Y. Zhou, and S. Cai, "Fractional-Order PD Attitude Control for a Type of Spacecraft with Flexible Appendages," *Fractal and Fractional*, vol. 6, no. 10, 2022. DOI: 10.3390/fractalfract6100601.

[66] B. Robb, M. McRobb, G. Bailet, J. Beeley, and C. R. McInnes, "Distributed magnetic attitude control for large space structures," *Acta Astronautica*, vol. 198, pp. 587–605, 2022. DOI: 10.1016/j.actaastro.2022.06.045.

[67] R. Benvenuto, S. Salvi, and M. Lavagna, "Dynamics analysis and GNC design of flexible systems for space debris active removal," *Acta Astronautica*, vol. 110, pp. 247–265, 2015. DOI: 10.1016/j.actaastro.2015.01.014.

[68] S. Di Gennaro, "Output stabilization of flexible spacecraft with active vibration suppression," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 39, no. 3, pp. 747–759, 2003. DOI: 10.1109/TAES.2003.1238733.

[69] H. Goldstein, C. Poole, and J. Safko, *Classical Mechanics*, 3rd Ed. Addison-Wesley, 2002, ISBN: 978-0-201-65702-9.

[70] V. J. Modi and A. M. Ibrahim, "A general formulation for librational dynamics of spacecraft with deploying appendages," *Journal of Guidance, Control, and Dynamics*, vol. 7, no. 5, pp. 563–569, Sep. 1984. DOI: 10.2514/3.19895.

[71] H. Bang, C.-K. Ha, and J. Hyoung Kim, "Flexible spacecraft attitude maneuver by application of sliding mode control," *Acta Astronautica*, vol. 57, no. 11, pp. 841–850, 2005. DOI: 10.1016/j.actaastro.2005.04.009.

[72] Q. Li, X. Ma, and T. Wang, "Reduced model for flexible solar sail dynamics," *Journal of Spacecraft and Rockets*, vol. 48, no. 3, pp. 446–453, 2011. DOI: 10.2514/1.48789.

[73] D. P. Madonna, M. Pontani, and P. Gasbarri, "Nonlinear attitude maneuvering of a flexible spacecraft for space debris tracking and collision avoidance," *Acta Astronautica*, 2023. DOI: 10.1016/j.actaastro.2023.03.043.

[74] F. Nicassio, D. Fattizzo, M. Giannuzzi, G. Scarselli, and G. Avanzini, "Attitude dynamics and control of a large flexible space structure by means of a minimum complexity model," *Acta Astronautica*, 2022. DOI: 10.1016/j.actaastro.2022.05.047.

[75] F. Angeletti, P. Gasbarri, and M. Sabatini, "Optimal design and robust analysis of a net of active devices for micro-vibration control of an on-orbit large space antenna," *Acta Astronautica*, vol. 164, pp. 241–253, 2019. DOI: 10.1016/j.actaastro.2019.07.028.

[76] L. F. Shampine and M. W. Reichelt, "The matlab ode suite," *SIAM journal on scientific computing*, vol. 18, no. 1, 1997. DOI: 10.1137/S1064827594276424.

[77] P. Virtanen *et al.*, "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python," *Nature Methods*, vol. 17, pp. 261–272, 2020. DOI: 10.1038/s41592-019-0686-2.

[78] S. Miller, T. Soares, Y. Weddingen, and J. Wendlandt, "Modeling flexible bodies with simscape multibody software," *Mathworks*, 2017. [Online]. Available: https://uk.mathworks.com/campaigns/offers/model-flexible-bodies.html.

[79] P. W. Kenneally, S. Piggott, and H. Schaub, "Basilisk: A Flexible, Scalable and Modular Astrodynamics Simulation Framework," *Journal of Aerospace Information Systems*, vol. 17, no. 9, pp. 496–507, May 2020. DOI: 10.2514/1.I010762.

[80] G. Bascom and H. Schaub, "Modular Dynamic Modeling of Hinged Solar Panel Deployments," in *AAS/AIAA Astrodynamics Specialist Conference*, Charlotte, North Carolina, 2022, pp. 22–725.

[81] F. L. Markley and J. L. Crassidis, *Fundamentals of spacecraft attitude determination and control*. New York, NY: Springer New York, 2014, pp. 1–486, ISBN: 978-1-493-90802-8.

[82] A. Viale and C. R. McInnes, "Attitude control actuator scaling laws for orbiting solar reflectors," *Advances in Space Research*, vol. 71, no. 1, pp. 604–623, 2023. DOI: 10.1016/j.asr.2022.10.015.

[83] I. M. Da Fonseca, P. M. Bainum, and A. R. Da Silva, "Structural control interaction for an LSS attitude control system using thrusters and reaction wheels," *Acta Astronautica*, vol. 60, no. 10-11, pp. 865–872, 2007. DOI: 10.1016/j.actaastro.2006.11.008.

[84] B. Robb, M. McRobb, and C. McInnes, "Magnetic Attitude Control of Gossamer Spacecraft using a 3D-printed, Electrically Conducting Support Structure," in *AIAA Scitech 2020 Forum*, American Institute of Aeronautics and Astronautics, Jan. 2020. DOI: 10.2514/6.2020-0714.

[85] B. Wie, "Solar Sail Attitude Control and Dynamics, Part Two," *Journal of Guidance, Control, and Dynamics*, vol. 27, no. 4, pp. 536–544, Jul. 2004. DOI: 10.2514/1.11133.

[86] A. Bolle and C. Circi, "Solar sail attitude control through in-plane moving masses," *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, vol. 222, no. 1, pp. 81–94, 2008. DOI: 10.1243/09544100JAERO223.

[87] Y. Tsuda, "An Attitude Control Strategy For Spinning Solar Sail," in *17th IFAC Symposium on Automatic Control in Aerospace*, vol. 40, 2007, pp. 401–406. DOI: 10.3182/20070625-5-FR-2916.00069.

[88] R. Funase, Y. Shirasawa, Y. Mimasu, O. Mori, Y. Tsuda, T. Saiki, and J. Kawaguchi, "On-orbit verification of fuel-free attitude control system for spinning solar sail utilizing solar radiation pressure," *Advances in Space Research*, vol. 48, no. 11, pp. 1740–1746, 2011. DOI: 10.1016/j.asr.2011.02.022.

[89] A. Borggrafe, J. Heiligers, M. Ceriotti, and C. McInnes, "Attitude control of large gossamer spacecraft using surface reflectivity modulation," in *65th International Astronautical Congress*, 2014, IAC-14–C1.3.4. [Online]. Available: https://pureportal.strath.ac.uk/en/publications/attitude-control-of-large-gossamer-spacecraft-using-surface-refle.

[90] J. D. Biggs and A. Negri, "Orbit-attitude control in a circular restricted three-body problem using distributed reflectivity devices," *Journal of Guidance, Control, and Dynamics*, vol. 42, no. 12, pp. 2712–2721, 2019. DOI: 10.2514/1.G004493.

[91] C. Gao, W. Wang, J. Yuan, and L. Guo, "Hybrid Sail Displaced Orbits Around L2 Point in the Elliptic Earth-Moon System," *Journal of Guidance, Control, and Dynamics*, vol. 42, no. 2, pp. 416–424, Nov. 2018. DOI: 10.2514/1.G003748.

[92] W. Wang, H. Baoyin, G. Mengali, and A. A. Quarta, "Solar sail cooperative formation flying around L2-type artificial equilibrium points," *Acta Astronautica*, vol. 169, pp. 224–239, 2020. DOI: 10.1016/j.actaastro.2019.10.028.

[93] J. Mu, S. Gong, and J. Li, "Coupled Control of Reflectivity Modulated Solar Sail for GeoSail Formation Flying," *Journal of Guidance, Control, and Dynamics*, vol. 38, no. 4, pp. 740–751, Apr. 2014. DOI: 10.2514/1.G000117.

[94] J. M. Longuski, J. J. Guzmán, and J. E. Prussing, "Bounded Control Problems," in *Optimal Control with Aerospace Applications*, J. M. Longuski, J. J. Guzmán, and J. E. Prussing, Eds., New York, NY: Springer, 2014, pp. 175–191, ISBN: 978-1-4614-8945-0.

[95] J. J. Kim and B. Agrawal, "Experiments on Jerk-Limited Slew Maneuvers of a Flexible Spacecraft," in *Guidance, Navigation, and Control Conference and Exhibit*, AIAA, Aug. 2006. DOI: 10.2514/6.2006-6187.

[96] M. Marshall and S. Pellegrino, "Slew Maneuver Constraints for Agile Flexible Spacecraft," in *AIAA SCITECH 2023 Forum*, Jan. 2023. DOI: 10.2514/6.2023-1883.

[97] J.-J. E. Slotine and W. Li, *Applied nonlinear control*. Englewood Cliffs, New Jersey: Prentice Hall Inc, Simon & Schuster, 1991, ISBN: 0-13-040890-5.

[98] C. R. Mclnnes, "Satellite attitude slew manoeuvres using inverse control," *The Aeronautical Journal*, vol. 102, no. 1015, pp. 259–266, 1998. DOI: 10.1017/S0001924000065295.

[99] S. N. Singh, "Flexible spacecraft maneuver: Inverse attitude control and modal stabilization," *Acta Astronautica*, vol. 17, no. 1, pp. 1–9, 1988. DOI: 10.1016/0094-5765(88)90122-1.

[100] E. Weerdt, E.-J. Kampen, D. Gemert, Q. P. Chu, and J. A. Mulder, "Adaptive Nonlinear Dynamic Inversion for Spacecraft Attitude Control with Fuel Sloshing," in *Guidance, Navigation and Control Conference and Exhibit*, AIAA, Aug. 2008. DOI: 10.2514/6.2008-7162.

[101] B. Xiao, Q. Hu, and Y. Zhang, "Adaptive Sliding Mode Fault Tolerant Attitude Tracking Control for Flexible Spacecraft Under Actuator Saturation," *IEEE Transactions on Control Systems Technology*, vol. 20, no. 6, pp. 1605–1612, 2012. DOI: 10.1109/TCST.2011.2169796.

[102] P. Guan, X.-J. Liu, and J.-Z. Liu, "Adaptive fuzzy sliding mode control for flexible satellite," *Engineering Applications of Artificial Intelligence*, vol. 18, no. 4, pp. 451–459, 2005. DOI: 10.1016/j.engappai.2004.11.003.

[103] Q. Hu, "Robust adaptive sliding mode attitude maneuvering and vibration damping of three-axis-stabilized flexible spacecraft with actuator saturation limits," *Nonlinear Dynamics*, vol. 55, no. 4, pp. 301–321, 2009. DOI: 10.1007/s11071-008-9363-1.

[104] D. J. Inman, "Active modal control for smart structures," *Philosophical Transactions of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, vol. 359, no. 1778, pp. 205–219, 2001. DOI: 10.1098/rsta.2000.0721.

[105] Z.-c. Qiu, X.-m. Zhang, H.-x. Wu, and H.-h. Zhang, "Optimal placement and active vibration control for piezoelectric smart flexible cantilever plate," *Journal of Sound and Vibration*, vol. 301, no. 3, pp. 521–543, 2007. DOI: 10.1016/j.jsv.2006.10.018.

[106] L. Grüne and J. Pannek, *Nonlinear model predictive control: theory and algorithms*, English. London;New York; Springer, 2011, ISBN: 978-0-85729-500-2.

[107] P. Iannelli, F. Angeletti, and P. Gasbarri, "A model predictive control for attitude stabilization and spin control of a spacecraft with a flexible rotating payload," *Acta Astronautica*, 2022. DOI: 10.1016/j.actaastro.2022.07.024.

[108] M. TayyebTaher and S. M. Esmaeilzadeh, "Model predictive control of attitude maneuver of a geostationary flexible satellite based on genetic algorithm," *Advances in Space Research*, vol. 60, no. 1, pp. 57–64, 2017. DOI: 10.1016/j.asr.2017.03.013.

[109] G. Song and B. N. Agrawal, "Vibration suppression of flexible spacecraft during attitude control," *Acta Astronautica*, vol. 49, no. 2, pp. 73–83, 2001. DOI: 10.1016/S0094-5765(00)00163-6.

[110] H. Liu, L. Guo, and Y. Zhang, "An anti-disturbance PD control scheme for attitude control and stabilization of flexible spacecrafts," *Nonlinear Dynamics*, vol. 67, no. 3, pp. 2081–2088, 2012. DOI: 10.1007/s11071-011-0130-3.

[111] B. Baghi, M. Kabganian, R. Nadafi, and E. Arabi, "Three-axis attitude stabilization of a flexible satellite using non-linear PD controller," *Transactions of the Institute of Measurement and Control*, vol. 40, no. 2, pp. 591–605, 2018. DOI: 10.1177/0142331216663616.

[112] J. Wen and K. Kreutz-Delgado, "The attitude control problem," *IEEE Transactions on Automatic Control*, vol. 36, no. 10, pp. 1148–1162, 1991. DOI: 10.1109/9.90228.

[113] G. C. Goodwin, S. F. Graebe, M. E. Salgado, *et al.*, *Control system design*, 1st. Valparaíso, Chile: Pearson, 2000, ISBN: 978-0139586538.

[114] J. Wan and J. Yu, "High Precision Satellite Attitude Control Based on Feedforward Compensation," in *6th World Congress on Intelligent Control and Automation*, vol. 2, 2006, pp. 6261–6264. DOI: 10.1109/WCICA.2006.1714287.

[115] N. Singer and W. Seering, "Design and comparison of command shaping methods for controlling residual vibration," in *1989 IEEE International Conference on*

Robotics and Automation, Los Alamitos, CA, USA: IEEE Computer Society, May 1989, pp. 888–893. DOI: 10.1109/ROBOT.1989.100094.

[116] A. Tzes and S. Yurkovich, "An adaptive input shaping control scheme for vibration suppression in slewing flexible structures," *IEEE Transactions on Control Systems Technology*, vol. 1, no. 2, pp. 114–121, 1993. DOI: 10.1109/87.238404.

[117] M. Romano, B. N. Agrawal, and F. Bernelli-Zazzera, "Experiments on Command Shaping Control of a Manipulator with Flexible Links," *Journal of Guidance, Control, and Dynamics*, vol. 25, no. 2, pp. 232–239, Mar. 2002. DOI: 10.2514/2.4903.

[118] W. Singhose and J. Vaughan, "Reducing Vibration by Digital Filtering and Input Shaping," *IEEE Transactions on Control Systems Technology*, vol. 19, no. 6, pp. 1410–1420, 2011. DOI: 10.1109/TCST.2010.2093135.

[119] R. T. Jones, "A Simplified Application of the Method of Operators to the Calculation of Disturbed Motions of an Airplane," Tech. Rep., 1936. [Online]. Available: https://ntrs.nasa.gov/citations/19930091636.

[120] J. Bardeen and W. H. Brattain, "Physical Principles Involved in Transistor Action," *Phys. Rev.*, vol. 75, no. 8, pp. 1208–1225, Apr. 1949. DOI: 10.1103/PhysRev.75.1208.

[121] W. H. Dutton, E. M. Rogers, and S.-H. Jun, "Diffusion and Social Impacts of Personal Computers," *Communication Research*, vol. 14, no. 2, pp. 219–250, Apr. 1987. DOI: 10.1177/009365087014002005.

[122] O. Kato and I. Sugiura, "An interpretation of airplane general motion and contol as inverse problem," *Journal of Guidance, Control, and Dynamics*, vol. 9, no. 2, pp. 198–204, Mar. 1986. DOI: 10.2514/3.20090.

[123] D. Thomson and R. Bradley, "Development and verification of an algorithm for helicopter inverse simulation," *Vertica*, vol. 14, no. 2, pp. 185–200, 1990. [Online]. Available: https://eprints.gla.ac.uk/117720/.

[124] R. A. Hess and C. Gao, "Generalized algorithm for inverse simulation applied to helicopter maneuvering flight," *Journal of the American Helicopter Society*, vol. 38, no. 4, pp. 3–15, 1993. DOI: 10.4050/jahs.38.4.3.

[125] S. Rutherford and D. G. Thomson, "Improved methodology for inverse simulation," *The Aeronautical Journal (1968)*, vol. 100, no. 993, pp. 79–86, 1996. DOI: 10.1017/S0001924000067348.

[126] R. A. Hess, C. Gao, and S. H. Wang, "Generalized technique for inverse simulation applied to aircraft maneuvers," *Journal of Guidance, Control, and Dynamics*, vol. 14, no. 5, pp. 920–926, Sep. 1991. DOI: 10.2514/3.20732.

[127] D. J. Murray-Smith, "A Review of Inverse Simulation Methods and Their Application," *International Journal of Modelling and Simulation*, vol. 34, no. 3, pp. 120–125, Jan. 2014. DOI: 10.2316/Journal.205.2014.3.205-5906.

[128] L. Lu, D. J. Murray-Smith, and D. G. Thomson, "Issues of numerical accuracy and stability in inverse simulation," *Simulation Modelling Practice and Theory*, vol. 16, no. 9, pp. 1350–1364, 2008. DOI: 10.1016/j.simpat.2008.07.003.

[129] K.-C. Lin, "Comment on "Generalized Technique for Inverse Simulation Applied to Aircraft Maneuvers"," *Journal of Guidance, Control, and Dynamics*, vol. 16, no. 6, pp. 1196–1197, Nov. 1993. DOI: 10.2514/3.56617.

[130] S. R. K. Shilliday, E. W. McGookin, and D. G. Thomson, "Rover Inverse Simulation Using a Segmented Model Approach," *IFAC-PapersOnLine*, vol. 56, no. 2, pp. 6820–6825, 2023. DOI: 10.1016/j.ifacol.2023.10.395.

[131] G. de Matteis, L. M. de Socio, and A. Leonessa, "Solution of aircraft inverse problems by local optimization," *Journal of Guidance, Control, and Dynamics*, vol. 18, no. 3, pp. 567–571, May 1995. DOI: 10.2514/3.21424.

[132] S. Lee and Y. Kim, "Time-domain finite element method for inverse problem of aircraft maneuvers," *Journal of Guidance, Control, and Dynamics*, vol. 20, no. 1, pp. 97–103, 1997. DOI: 10.2514/2.4000.

[133] M. Borri, C. L. Bottasso, and F. Montelaghi, "Numerical Approach to Inverse Flight Dynamics," *Journal of Guidance, Control, and Dynamics*, vol. 20, no. 4, pp. 742–747, Jul. 1997. DOI: 10.2514/2.4106.

[134] R. Celi, "Optimization-Based Inverse Simulation of a Helicopter Slalom Maneuver," *Journal of Guidance, Control, and Dynamics*, vol. 23, no. 2, pp. 289–297, Mar. 2000. DOI: 10.2514/2.4521.

[135] M. Bagiev, D. G. Thomson, D. Anderson, and D. Murray-Smith, "Predictive inverse simulation of helicopters in aggressive manoeuvring flight," *The Aeronautical Journal*, vol. 116, no. 1175, pp. 87–98, 2012. DOI: 10.1017/S0001924000006631.

[136] G. Avanzini, D. Thomson, and A. Torasso, "Model predictive control architecture for rotorcraft inverse simulation," *Journal of Guidance, Control, and Dynamics*, vol. 36, no. 1, pp. 207–217, Dec. 2013. DOI: 10.2514/1.56563.

[137] W. Zhou, H. Wang, G. Tang, and S. Guo, "Inverse simulation system for manual-controlled rendezvous and docking based on artificial neural network," *Advances in Space Research*, vol. 58, no. 6, pp. 938–949, 2016. DOI: 10.1016/j.asr.2016.05.039.

[138] G. Avanzini, G. de Matteis, and L. M. de Socio, "Two-Timescale-Integration Method for Inverse Simulation," *Journal of Guidance, Control, and Dynamics*, vol. 22, no. 3, pp. 395–401, May 1999. DOI: 10.2514/2.4410.

[139] Y. Yuan, D. Thomson, and D. Anderson, "Faster-than-realtime inverse simulation method for tiltrotor handling qualities investigation," *Aerospace Science and Technology*, vol. 124, p. 107 516, 2022. DOI: 10.1016/j.ast.2022.107516.

[140] D. Thomson and R. Bradley, "Inverse simulation as a tool for flight dynamics research-Principles and applications," *Progress in Aerospace Sciences*, vol. 42, no. 3, 2006. DOI: 10.1016/j.paerosci.2006.07.002.

[141] D. G. Thomson and R. Bradley, "The use of inverse simulation for conceptual design," in *16th European Rotorcraft Forum*, Glasgow, 1990. [Online]. Available: https://eprints.gla.ac.uk/117721/.

[142] D. G. Thomson and R. Bradley, "The principles and practical application of helicopter inverse simulation," *Simulation Practice and Theory*, vol. 6, no. 1, pp. 47–70, 1998. DOI: 10.1016/S0928-4869(97)00012-8.

[143] C. Gao and R. A. Hess, "Inverse simulation of large-amplitude aircraft maneuvers," *Journal of Guidance, Control, and Dynamics*, vol. 16, no. 4, pp. 733–737, Jul. 1993. DOI: 10.2514/3.21074.

[144] L. Lu, D. Murray-Smith, and E. McGookin, "Applications of Inverse Simulation within the Model-Following Control Structure," 2006. [Online]. Available: https://www.researchgate.net/publication/44241466_Applications_of_Inverse_Simulation_within_the_Model-Following_Control_Structure.

[145] L. Lu, D. J. Murray-Smith, and D. G. Thomson, "Sensitivity-Analysis Method for Inverse Simulation Application," *Journal of Guidance, Control, and Dynamics*, vol. 30, no. 1, pp. 114–1215, Jan. 2007. DOI: 10.2514/1.20722.

[146] L. Lu, D. J. Murray-Smith, and E. W. McGookin, "Investigation of inverse simulation for design of feedforward controllers," *Mathematical and Computer Modelling of Dynamical Systems*, vol. 13, no. 5, pp. 437–454, Oct. 2007. DOI: 10.1080/13873950701344023.

[147] H. Jafari and F. Shahmiri, "Examination of Quadrotor Inverse Simulation Problem Using Trust-Region Dogleg Solution Method," 2018. [Online]. Available: https://jast.ias.ir/article_106251.html.

[148] Y. Yuan, D. Thomson, and D. Anderson, "Manoeuverability Assessment for Urban Air Mobility Using the Improved Inverse Simulation," in *AIAA SCITECH 2023 Forum*, American Institute of Aeronautics and Astronautics, Jan. 2023. DOI: 10.2514/6.2023-2206.

[149] M. G. Gallup, "Inverse Simulation of Autogyros," *Journal of the American Helicopter Society*, vol. 68, no. 4, pp. 42 001–42 010, 2023. DOI: 10.4050/JAHS.68.042001.

[150] D. Murray-Smith, L. Lu, and E. McGookin, "Applications of inverse simulation to a nonlinear model of an underwater vehicle," in *Summer Simulation Multi-Conference*, Edinburgh, 2008. [Online]. Available: https://eprints.gla.ac.uk/5042/.

[151] D. J. Murray-Smith, "Inverse simulation and analysis of underwater vehicle dynamics using feedback principles," *Mathematical and Computer Modelling of Dynamical Systems*, vol. 20, no. 1, pp. 45–65, Jan. 2014. DOI: 10.1080/13873954.2013.805146.

[152] K. J. Worrall, D. G. Thomson, E. W. McGookin, and T. Flessa, "Autonomous Planetary Rover Control using Inverse Simulation," in *13th Symposium on Advanced Space Technologies in Robotics and Automation*, 2015. [Online]. Available: https://eprints.gla.ac.uk/106312/.

[153] K. Worrall, D. Thomson, and E. Mcgookin, "Application of inverse simulation to a wheeled mobile robot," in *6th International Conference on Automation, Robotics and Applications*, 2015, pp. 155–160. DOI: 10.1109/ICARA.2015.7081140.

[154] T. Flessa, E. McGookin, and D. Thomson, "Numerical stability of inverse simulation algorithms applied to planetary rover navigation," in *24th Mediterranean Conference on Control and Automation*, 2016, pp. 901–906. DOI: 10.1109/MED.2016.7535960.

[155] T. Flessa, "Analysis of inverse simulation algorithms with an application to planetary rover guidance and control," Ph.D. dissertation, University of Glasgow, 2021. [Online]. Available: https://theses.gla.ac.uk/82863/.

[156] S. R. K. Shilliday, E. W. McGookin, and D. G. Thomson, "A comparison of forward and inverse simulation methods for fault detection on a rover," in *8th International Conference on Control, Decision and Information Technologies*, vol. 1, 2022, pp. 1432–1437. DOI: 10.1109/CoDIT55151.2022.9804106.

[157] R. Bradley, G. D. Padfield, D. J. Murray-Smith, and D. G. Thomson, "Validation of helicopter mathematical models," *Transactions of the Institute of Measurement and Control*, vol. 12, no. 4, pp. 186–196, Oct. 1990. DOI: 10.1177/014233129001200405.

[158] M. L. Ireland, T. Flessa, D. Thomson, and E. McGookin, "Comparison of nonlinear dynamic inversion and inverse simulation," *Journal of Guidance, Control, and Dynamics*, vol. 40, no. 12, pp. 3304–3309, 2017. DOI: 10.2514/1.G002875.

[159] R. Gordon, K. Worrall, and M. Ceriotti, "Attitude Control of a Nanosatellite using Inverse Simulation," in *72nd International Astronautical Congress*, Dubai, UAE, 2021, IAC–21, C1, 8, 12, x64025. [Online]. Available: https://eprints.gla.ac.uk/257797/.

[160] R. Gordon, K. Worrall, and M. Ceriotti, "Attitude Control of Satellites with Flexible Appendages Using Inverse Simulation," in *73rd International Astronautical Congress*, Paris, France, 2022, IAC-22–C1.1.6. [Online]. Available: https://eprints.gla.ac.uk/278365/.

[161] R. Gordon, M. Ceriotti, and K. Worrall, "Effects of model fidelity and uncertainty on a model-based attitude controller for satellites with flexible appendages," *Acta Astronautica*, 2023. DOI: 10.1016/j.actaastro.2023.10.011.

[162] R. Gordon, K. Worrall, and M. Ceriotti, "Broyden's Method for Faster Inverse Simulation," *Journal of Guidance, Control, and Dynamics*, vol. 47, no. 9, pp. 1939–1945, 2024. DOI: 10.2514/1.G007975.

[163] R. Gordon, M. Ceriotti, and K. Worrall, "Investigation of attitude control actuators for large flexible space structures using inverse simulation," *Advances in Space Research*, vol. 75, no. 2, pp. 2062–2087, 2025. DOI: 10.1016/j.asr.2024.11.010.

[164] C. R. Mclnnes, "Large angle slew maneuvers with autonomous sun vector avoidance," *Journal of Guidance, Control, and Dynamics*, vol. 17, no. 4, pp. 875–877, Jul. 1994. DOI: 10.2514/3.21283.

[165] B. Xiao, Q.-L. Hu, and G. Ma, "Adaptive sliding mode backstepping control for attitude tracking of flexible spacecraft under input saturation and singularity," *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, vol. 224, no. 2, pp. 199–214, Feb. 2010. DOI: 10.1243/09544100JAERO668.

[166] L. Cao, B. Xiao, and M. Golestani, "Robust fixed-time attitude stabilization control of flexible spacecraft with actuator uncertainty," *Nonlinear Dynamics*, vol. 100, no. 3, pp. 2505–2519, 2020. DOI: 10.1007/s11071-020-05596-5.

[167] Y. Shen, H. Shi, Z. Fan, and X. Song, "The hybrid attitude maneuver control of the three-axis stable liquid-filled spacecraft with large-deformation appendage," *International Journal of Dynamics and Control*, 2023, ISSN: 2195-2698. DOI: 10.1007/s40435-023-01204-7.

[168] S. S. Rao, *Vibration of continuous systems*, 2nd Ed. John Wiley & Sons, Ltd, 2019, ISBN: 9781119424147.

[169] J. D. Lambert *et al.*, *Numerical methods for ordinary differential systems*. Wiley New York, 1991, vol. 146.

[170] A. Iserles, *A first course in the numerical analysis of differential equations*, English, 2nd. Cambridge;New York; Cambridge University Press, 2009, ISBN: 978-0-521-73490-5.

[171] S. Tafazoli, "On Attitude Recovery of Spacecraft using Nonlinear Control," Ph.D. dissertation, 2005. DOI: 10.48550/arXiv.2012.02593.

[172] H. Schaub and J. L. Junkins, *Nonlinear Spacecraft Stability and Control*. 2018, pp. 387–518, ISBN: 978-1-62410-521-0.

[173] K. Shoemake, "Uniform Random Rotations," in *Graphics Gems III (IBM Version)*, D. KIRK, Ed., San Francisco: Morgan Kaufmann, 1992, pp. 124–132. DOI: 10.1016/B978-0-08-050755-2.50036-1.

[174] K. Shoemake, "Animating Rotation with Quaternion Curves," in *12th Annual Conference on Computer Graphics and Interactive Techniques*, New York, NY, USA: Association for Computing Machinery, 1985, pp. 245–254. DOI: 10.1145/325334.325242.

[175] E. B. Dam, M. Koch, and M. Lillholm, "Quaternions, interpolation and animation," Department of Computer Science, University of Copenhagen, Tech. Rep., 1998. [Online]. Available: https://api.semanticscholar.org/CorpusID:115207743.

[176] C. Allard, M. D. Ramos, H. Schaub, P. Kenneally, and S. Piggott, "Modular Software Architecture for Fully Coupled Spacecraft Simulations," *Journal of Aerospace Information Systems*, vol. 15, no. 12, pp. 670–683, Oct. 2018. DOI: 10.2514/1.I010653.

[177] P. Mann, "Near-Equilibrium Oscillations," in *Lagrangian and Hamiltonian Dynamics*, Oxford University Press, Jun. 2018, ch. 12, p. 146, ISBN: 9780198822370.

[178] M. L. Gambhir, *Stability analysis and design of structures*. Springer Berlin Heidelberg, 2004, ISBN: 978-3-540-20784-9.

[179] T. Murphey, "Symbolic Equations for the Stiffness and Strength of Straight Longeron Trusses," in *47th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, American Institute of Aeronautics and Astronautics, May 2006, pp. 2006–2123. DOI: 10.2514/6.2006-2123.

[180] J. Humar, *Dynamics of structures*, 3rd Ed. Boca Raton, Florida: CRC press, Taylor & Francis Group, 2012, ISBN: 978-0-203-11256-4.

[181] B. Robb, "The dynamics and control of large space structures with distributed actuation," Ph.D. dissertation, University of Glasgow, 2023.

[182] C. R. McInnes, *Solar Sailing: Technology, Dynamics and Mission Applications*. Springer-Verlag Berlin Heidelberg GmbH, 1999, ISBN: 978-1-85233-102-3.

[183] B. Fu, E. Sperber, and F. Eke, "Solar sail technology—A state of the art review," *Progress in Aerospace Sciences*, vol. 86, pp. 1–19, 2016. DOI: 10.1016/j.paerosci.2016.07.001. (visited on 03/26/2025).

[184] K. S. Arun, T. S. Huang, and S. D. Blostein, "Least-Squares Fitting of Two 3-D Point Sets," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-9, no. 5, pp. 698–700, 1987. DOI: 10.1109/TPAMI.1987.4767965.

[185] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle, "Surface Reconstruction from Unorganized Points," in *Proceedings of the 19th Annual Conference on Computer Graphics and Interactive Techniques*, New York, NY, USA: Association for Computing Machinery, 1992, pp. 71–78. DOI: 10.1145/133994.134011.

[186] J. Gere and B. Goodno, *Mechanics of Materials*, Brief Edit. Cenage Learning, 2012, ISBN: 978-1-111-13602-4.

[187] J. R. Rumble, *CRC Handbook of Chemistry and Physics (Internet Version)*, Boca Raton, Florida, 2023.

[188] G. Avanzini and G. de Matteis, "A Local Optimization Technique for Attitude Motion Tracking Using Control Moment Gyroscopes," *The Journal of the Astronautical Sciences*, vol. 50, no. 2, pp. 213–229, 2002. DOI: 10.1007/BF03546263.

[189] C. G. Broyden, "A class of methods for solving nonlinear simultaneous equations," *Mathematics of computation*, vol. 19, no. 92, pp. 577–593, 1965. DOI: 10.1090/S0025-5718-1965-0198670-6.

[190] J. C. A. Barata and M. S. Hussein, "The Moore–Penrose Pseudoinverse: A Tutorial Review of the Theory," *Brazilian Journal of Physics*, vol. 42, no. 1, pp. 146–165, 2012. DOI: 10.1007/s13538-011-0052-z.

[191] J. R. R. A. Martins and A. Ning, *Engineering design optimization*. Cambridge University Press, 2021, ISBN: 9781108833417.

[192] J. Nocedal and S. J. Wright, *Numerical optimization*. Springer, 2006, ISBN: 0-387-30303-0.

[193] *Optimization toolbox version: 9.4 (r2022b)*, Natick, Massachusetts, United States, 2022. [Online]. Available: https://www.mathworks.com.

[194] M. Hermann and M. Saravi, *A first course in ordinary differential equations: analytical and numerical methods*, English. New Delhi: Springer, 2014, ISBN: 978-81-322-1834-0.

[195] R. Kristiansen and D. Hagen, "Modelling of Actuator Dynamics for Spacecraft Attitude Control," *Journal of Guidance, Control, and Dynamics*, vol. 32, no. 3, pp. 1022–1025, May 2009. DOI: 10.2514/1.42574.