Generative and Regressive Approaches for Periodic Orbit-Based Spacecraft Mission Design in the Circular Restricted Three Body Problem

by

Jonah Pedra

Aerospace Supervisor:	J. De Teixeira da Encarnacao
Computer Science Supervisor:	D. Tax
Project Duration:	04/2025 - 01/2025
Faculties:	Aerospace Engineering/Computer Science
Course Codes:	AE5822/IN5000

TU Delft

Abstract

The circular restricted three-body problem (CR3BP) is a canonical example of deterministic chaos in dynamics, characterized by extreme sensitivity to initial conditions and the absence of a general analytical solution. The study of the CR3BP has long provided insight into the chaotic gravitational interactions that guide the design of complex spacecraft trajectories. This thesis evaluates emerging artificial intelligence regression and generative techniques as a means to overcome computational bottlenecks and provide new tools for exploring families of periodic orbits in a spacecraft mission design context.

The work begins with a detailed examination of the gravitational model, numerical formulations, and analytical structures that govern the CR3BP. Lagrange points, periodic motion, stability measures, and manifold structures are introduced with emphasis on their relevance to spacecraft mission design. A numerical pipeline is then introduced to compute periodic orbits and their manifolds, revealing the sensitivity of the system to numerical tolerances, time-of-flight selections, continuation strategies, and discretisation resolution.

A comprehensive review of current developments in numerical integration, learning-based trajectory prediction, and orbit generation then provides the foundation for this study within the context of current literature. From this review, two research objectives are formulated. The first explores whether generative models can reproduce physically consistent periodic orbits across multiple orbital families, and whether the latent structure of these generative models can accurately capture the bifurcationary relationship between orbital families. The second investigates whether regression-based surrogates can approximate unstable manifold propagation with sufficient accuracy to support mission-design workflows.

Several generative models are evaluated, including dense, convolution, and time-series variational autoencoders, in combination with a preliminary investigation into transformer-diffusion architectures. These models are assessed using two datasets of periodic orbits with differing levels of diversity, comprising 9 and 31 orbit families. Their ability to reconstruct orbits, reveal latent geometric structure, support latent-space arithmetic, and sample new orbits is evaluated. Although the generated orbits exhibit strong spatial and dynamical similarity to the reference trajectories, their raw outputs generally require post-processing via multiple-shooting differential correction to ensure physical validity. A loss formulation that explicitly incorporates the Jacobi constant is introduced, encouraging approximate conservation of energy within the system, and penalizing in-sequence variations in Jacobi.

Surrogate regression models are evaluated for the prediction of unstable manifold trajectories. Both deep neural networks and Kolmogorov-Arnold networks are explored with energy conservation (Jacobi) mechanisms. These surrogates are compared against traditional integrators for accuracy, energy preservation, and computational cost. Although surrogate models reproduce the qualitative behaviour of invariant manifolds, their accuracy remains several orders of magnitude above the desired threshold for mission design. Neural-scaling analyses demonstrate that achieving the required accuracy using current architectures would demand computational resources far exceeding those of numerical integrators.

Generative approaches hold promise for rapid exploration of periodic-orbit families and for augmenting mission-design strategies, whereas regression-based surrogates of invariant manifolds currently fall short of replacing integrators. Future research should focus on alternative prior formulations, expanded datasets, different decoder architectures, and further integrating physics-based constraints. These methods are readily transferable to other nonlinear systems by substituting the governing equations and conserved quantities, and contribute to the ongoing development of AI-based tools for dynamical systems analysis and trajectory design.

Index

Abstract **1**

Nomenclature **10**

1 Introduction **1**
1.1 Introduction . . . 1
1.2 Context: 3BP Variations . . . 2
1.2.1 Hierarchical 3BP (H3BP) . . . 2
1.2.2 Non-Hierarchical 3BP (NH3BP) . . . 2
1.2.3 Restricted 3BP (R3BP) . . . 2
1.2.4 Alternative 3BP Cases . . . 2
1.3 Context: Research Applications . . . 3
1.3.1 Planetary Migration . . . 3
1.3.2 Eccentricity Distributions . . . 3
1.3.3 Trajectory Optimization . . . 3

2 Dynamical Model **5**
2.1 Synodic Reference Frame . . . 5
2.2 Equations of Motion . . . 6
2.3 Jacobi Constant (C) . . . 7
2.4 Lagrange (Libration) Points . . . 7
2.5 Periodic Orbits . . . 8
2.6 State Transition Matrix and Monodromy Matrix . . . 9
2.7 Stability Index . . . 11
2.8 Invariant Manifolds . . . 11
2.9 Hetero/Homoclinic Connections . . . 13
2.10 Single/Multiple Shooting Differential Correctors . . . 15
2.11 Familial Continuation . . . 16
2.11.1 Natural Parameter Variation (NPV) Continuation . . . 17
2.11.2 Pseudo-Arc Length (PAL) Continuation . . . 17
2.11.3 Interpolation . . . 18
2.12 Transfer to Ephemeris-based High Fidelity (EHF) Model . . . 18
2.13 Bifurcation Theory . . . 19
2.13.1 CR3BP Bifurcation Types . . . 19
2.13.1.1 Bifurcations of the L_2 Families . . . 21
2.13.2 Broucke's Stability Diagram . . . 22

3 Mission Design Pipeline **24**
3.1 Numerical Representation of Periodic Orbits and their Manifolds . . . 24
3.2 Discretization and Computational Process . . . 25
3.2.1 Population of Periodic Orbit Abacus . . . 26
3.2.2 Population of Manifold Arc Set . . . 27
3.3 Transfer to EHF . . . 28

4 State Of The Art Developments **30**
4.1 Clean Simulation & Arbitrary Precision Integrators . . . 30
4.2 Neural Surrogates to Orbital Integration . . . 31
4.2.1 DNN Implementations . . . 31
4.2.2 HNN Implementations . . . 33
4.2.3 Hybrid Implementations . . . 34
4.2.4 Network Sensitivity . . . 35

4.3 Generative Architectures . . . 36
4.3.1 VAE-Based Trajectory Generation . . . 36
4.3.2 Latent Operations . . . 38
4.3.3 DNN-Based Familial Continuation . . . 39
4.4 Orbital Classification Techniques . . . 40
4.4.1 Algorithm-Based Classification . . . 40
4.4.2 Convolutional Neural Network Classification . . . 41
4.4.3 Classification of Boundary Regions . . . 42
4.5 Approaches with Research Potential . . . 43
4.5.1 (Multiplicative) Kolmogorov-Arnold Networks . . . 43
4.5.2 Transformers as Chaotic Control Mechanisms . . . 44
4.5.3 Alternatives VAE Architectures . . . 44
4.5.3.1 Dense-VAE . . . 44
4.5.3.2 Convolutional-VAE . . . 44
4.5.3.3 TimeVAE . . . 45
4.5.4 Time-Series Diffusion Models . . . 46
4.5.5 Large Kernel CNN's (LKCNN) . . . 46

5 Research Proposal 47
5.1 Research Objectives and Questions . . . 47
5.1.1 Research Objectives . . . 47
5.1.2 Research Questions . . . 48
5.1.3 Model Requirements . . . 48

6 Data and Numerical Benchmarking 51
6.1 Existing Periodic Orbit Data . . . 51
6.1.1 JPL Repository . . . 51
6.1.1.1 Bifurcation Characterization . . . 52
6.2 Numerical Integration . . . 56
6.3 Periodic Orbit Classification . . . 58
6.3.1 Clustering Algorithms . . . 58
6.4 Requirement Verification . . . 59
6.4.1 Trajectory Accuracy Requirements Verification . . . 60
6.4.2 Dataset Continuity Verification . . . 62
6.5 Computational Requirements of Numerical Schemes . . . 64
6.5.1 Manifold Arc Propagation . . . 64
6.5.2 Familial Interpolation Benchmarking and Sensitivity Analysis . . . 67
6.5.2.1 Interpolator Degree . . . 68
6.5.2.2 Continuity Tolerance Requirement . . . 69
6.5.3 Manifold Arc Set Population . . . 70

7 Periodic Orbit Generative Schemes 72
7.1 Data Pre-Processing & Normalization . . . 72
7.2 Neural Architectures and Loss Landscapes . . . 73
7.3 Hyperparameter Optimization . . . 74
7.4 Results . . . 75
7.4.1 Reconstruction of Periodic Orbits (Posterior Decoding) . . . 77
7.4.2 Latent Structure Verification . . . 79
7.4.3 Latent Operations . . . 80
7.4.4 Latent Prior Sampling . . . 83
7.4.4.1 MSDC Post-Processing . . . 84
7.4.4.2 Prior-Sample Clustering . . . 85
7.5 Other Architectures Considered: TransFusion . . . 86
7.6 Model Sensitivity . . . 86
7.6.1 MSDC Node Sampling/Noise Impact on Convergence Rates . . . 87
7.6.2 PINN Gradient Sensitivity . . . 89
7.7 Neural Surrogates of Invariant Manifolds . . . 92
7.7.1 Manifold Set Population Experimental Parameters . . . 93

7.7.2 Dataset Generation . . . 95
7.7.3 Regression Models . . . 96
7.7.3.1 DNN Implementation . . . 96
7.7.3.2 KAN Implementation . . . 97
7.7.3.3 Loss Landscape . . . 97
7.7.4 Results . . . 97

8 Neural Mission Design Pipeline, Future Research Directions 100
8.1 Proposed Mission Design Pipeline . . . 100
8.2 Future Research Topics . . . 102

9 Conclusion 105

References 108

A Supplemental Figures and Tables 112
A.1 Families of Periodic Orbits in JPL-A . . . 112
A.1.1 $L_{2,N}$ Halo . . . 112
A.1.2 L_2 Lyapunov . . . 114
A.1.3 L_2 Vertical . . . 115
A.1.4 L_2 Axial . . . 116
A.1.5 L_2 Dragonfly . . . 117
A.1.6 L_2 Butterfly . . . 118
A.2 Node Placement . . . 119
A.3 Efficiency and Accuracy of Numerical Integration Schemes . . . 120
A.4 Franz-Russel Database . . . 121
A.5 VAE Optimization Constants and Hyperparameter Search Space . . . 122
A.6 DNN & KAN Optimization Constants and Hyperparameter Search Space . . . 123

List of Figures

1.1 Example of a transfer trajectory based on invariant-manifolds between two periodic orbits ($L_{1,N}$ Halo, $L_{2,S}$ Halo) [13] [Modified] . . . 4

2.1 Geometry of the CR3BP in a rotating reference frame [31] . . . 5
2.2 Lagrange points as a function of μ, locations at $\mu = 0.01215$ . . . 8
2.3 $L_{2,N}$ Halo family, spatial representation, coloured by Jacobi Constant (C). . . . 9
2.4 Unit circle of the monodromy matrix [34] . . . 11
2.5 Process of applying invariant manifold perturbations for an $L_{2,S}$ Halo orbit. Applied perturbation magnitude is $\epsilon = 10^{-4}$. . . . 12
2.6 Example of heteroclinic and homoclinic connections between periodic orbits in the Sun-Jupiter system. Shaded regions represent forbidden zones determined by the specified Jacobi energy level, where the motion of M_3 is restricted due to lack of energy [37]. . . . 13
2.7 Manifold patching arcs between $L_{1,N}$ and $L_{2,S}$ Halo periodic orbits and manifolds [35] . 14
2.8 SSDC representation between actual (T) and desired trajectory ($\hat{T}$) . . . 15
2.9 Multiple shooting segmentation and convergence of a trajectory based on differential correction per segment [32]. . . . 16
2.10 Diagram Comparing Natural Parameter Variation (NPV) and Pseudo-Arc Length Continuation [36] . . . 17
2.11 Transition process of a CR3BP Halo orbit to an Ephemeris model. First differential correction for a number of orbital revolutions is applied, prior to pruning of edge cases [32] 19
2.12 Sample Bifurcation Diagram [34] . . . 20
2.13 Bifurcation Diagram of the Earth Moon System, nomenclature is not adopted [33] 21
2.14 L_2 Lyapunov (green) and $L_{2,S}$ Halo (blue) families. The red orbit denotes the bifurcating orbit [36] . . . 21
2.15 $L_{2,S}$ Halo (blue) and $L_{2,S}$ Butterfly (purple) families. The red orbit denotes the bifurcating orbit [36]. . . . 22
2.16 Broucke Stability Diagram [34] . . . 22
2.17 L_2 Halo (NRHO) Broucke stability diagram [34] . . . 23

3.1 Pipeline for Computing Periodic Orbits and Manifold Arc's . . . 25
3.2 Comparison of two representations of the $L_{2,S}$ Halo family. Green point is the L_2 point, whilst the blue point is the Moon (Moon not to scale). Coloring is in accordance with Jacobi Constant (C) . . . 27
3.3 Manifold arc population across 5 $L_{2,S}$ Halo periodic orbits for 4 [TU] with a resolution of $\Delta\xi = 0.02$. . . . 28

4.1 Comparison of two DNN approaches: (a) direct DNN approximation and (b) Connectome-inspired reservoir approximation of an orbital trajectory. . . . 32
4.2 Koopman-based deep-learning approaches for CR3BP dynamics. . . . 32
4.3 Relative (energy) error between HNN predictor, DNN predictor, and Numerical Integrator [48]. . . . 34
4.4 Schematic of a Hybrid DNN/HNN Integration Network [48]. . . . 35
4.5 HNN and DNN energy dissipation over time for various initialization conditions [Modified] [48]. . . . 36
4.6 Comparison of VAE generated orbits pre/post refinement algorithm, (a) shows the initially generated orbits with dynamic inconsistencies, (b) shows the refined orbits, corrected to satisfy physical constraints [20]. . . . 37
4.7 VAE latent space coloured by orbital family classes [20] [Modified]. . . . 38
4.8 Vector arithmetic for learned latent word embedding [58] . . . 39

4.9 Comparison of domain expansion and stability analysis for DNN-based periodic orbital family expansion [2] . . . 40
4.10 Receiver Operating Characteristic (ROC) curves of the tuned CNN Architectures [60] . 41
4.11 One-hot prediction results for stability boundary, with and without AL [61]. . . . 42
4.12 KAN vs DNN performance on three examples. KANs scale more efficiently and are more information-dense than DNNs [63]. . . . 43
4.13 Block Diagram of the Convolutional Network. (De)Convolutions are applied per channel/feature [56] . . . 45
4.14 Block diagram of TimeVAE. (De)Convolutions are applied per channel/feature [56] . . . 45

6.1 Broucke stability diagram of the NRHO subset of the $L_{2,N/S}$ Halo family, illustrating the locations of bifurcations (marked intersections). . . . 52
6.2 Broucke stability diagram illustrating tangent bifurcations of the L_2 Lyapunov family. . . 53
6.3 Node–edge graph illustrating the bifurcation relationships between orbit families in the JPL repository. Icon shapes denote bifurcation type. Red dotted line denotes JPL-A subset. . . . 54
6.4 Node–edge graph of the JPL-A subset, showing orbit families bifurcating from the L_2 point and their interconnections. Each bifurcation node is numbered. . . . 55
6.5 $L_{2,N}$ Halo, coloured by Jacobi in spatial phase space. Bifurcating orbits indicated. . . . 55
6.6 $L_{2,N}$ Dragonfly and Butterfly orbital families, coloured by Jacobi, bifurcation nodes indicated. . . . 56
6.7 Process Imbalance in Adaptive Time Step Integrators [73] . . . 57
6.8 Convergence of a noisy L_2 Lyapunov orbit to an outer circular orbit as a result of MSDC 58
6.9 Transition of $L_{2,S}$ Halo orbit from CR3BP to EHF. Left: 3D spatial phase space representation. Right: xy projection. . . . 60
6.10 State components of CR3BP and EHF transitioned $L_{2,S}$ Halo orbit over 10 revolutions . 61
6.11 Magnitude of transition error between position (Δr) and velocity (Δv) for a $L_{2,S}$ Halo orbit transitioning between the CR3BP and EHF . . . 61
6.12 JPL Database: Continuity Deficit Pre-/Post 10 SSDC iterations. . . . 62
6.13 Periodicity satisfaction by class for JPL Dataset: horizontal stacked bars show the fraction of orbits that were initially periodic (green), became periodic only after SSDC (blue), and remained non-periodic (red). N_i reports per class counts. . . . 63
6.14 Periodicity satisfaction by stability category: Stable [1, 1.25], Minor Instability (1.25, 2.5], Unstable (2.5, ∞). Stacks indicate initial (green), corrected-only (blue), and failed (red) percentages. N_i report per-class counts. . . . 63
6.15 Periodicity satisfaction by period bins: horizontal stacks show initial (green), corrected-only (blue), and failed (red) percentages per bin. N_i report per-class counts. . . . 64
6.16 Manifold divergence over time for varying $L_{2,S}$ Halo family members. . . . 65
6.17 Mean position error norm (top), velocity error norm (middle) norm, Jacobi constant error (bottom) at 10 [TU] for various integrators and integration tolerances. Red line denotes the accuracy requirements, $\Delta p \leq 1 \times 10^{-5}\,[\mathrm{LU}]$, $\Delta v \leq 1 \times 10^{-5}\,[\mathrm{LU/TU}]$, $\Delta C \leq 1\%$. . . 66
6.18 Comparison of the average number of FLOPs per TU for integration schemes without and with STM. . . . 67
6.19 $L_{2,N/S}$ Halo family interpolation with $\Delta C \approx 0.05$. Initial guesses for linear, quadratic and cubic interpolators marked (VY). . . . 68
6.20 Interpolation convergence cost and convergence rate of SSDC scheme for varying abacus resolutions and interpolator degrees. . . . 68
6.21 Convergence cost and rate as a function of abacus resolution (ΔC) and periodicity tolerance. . . . 69
6.22 Computational cost in FLOPs for interpolation of a parent orbit and propagation of a manifold arc, DOP853 (tol 10^{-13}). . . . 70
6.23 Ratio of computational cost between initial SSDC interpolation and Manifold arc propagation, DOP853 (tol 10^{-13}). . . . 71

7.1 Overview of the Data Processing, Training, and Inference Pipeline. . . . 72
7.2 VAE Dataset State Reconstruction Error . . . 75

7.3 Pareto fronts of generated sample state accuracy and computational effort (using parameter count as a surrogate) across different architectures (coloured) . . . 76
7.4 Pareto front per dataset (JPL, JPL-A) of state accuracy and computational effort . . . 77
7.5 State variable reconstructions across full period phase for sampled VAE architecture and sampled orbit . . . 78
7.6 Reconstruction error per state variable for each VAE architecture for a sampled orbit . . 78
7.7 Reconstructed orbital trajectories for representative samples from two periodic orbit families. . . . 79
7.8 UMAP reduction of the latent distribution of the validation set, highlighting bifurcating nodes. . . . 80
7.9 Cosine similarity between latent representation of node members . . . 81
7.10 Cosine similarity between vector differences of latent representations per node . . . 82
7.11 Transformation of latent representations of L_2 Halo along direction of period-doubling $\vec{d}_{2P}$ 82
7.12 Latent sampling, sequence and spatial representation. . . . 84
7.13 MSDC scheme demonstration for a L_2 Lyapunov orbit sampled from the prior. . . . 85
7.14 Clustering of decoded and post-processed solutions (DBSCAN, DTW). UMAP dimensionality reduction is applied for visualization. . . . 85
7.15 Comparison of initial phase shift behaviour for node placements outside (left) and inside (right) the special set $\{0, 0.5, 1.0\}$. . . . 87
7.16 Impact of initial node–phase selection on MSDC convergence rates. . . . 88
7.17 DTW similarity threshold used to determine trajectory correspondence. . . . 89
7.18 Impact of prediction errors on magnitude and ratio of L_{MSE} and L_C gradients . . . 91
7.19 Cosine similarity between L_{MSE} and L_C gradient updates per component (left) and overall (right) across a range of prediction errors ($||\epsilon||$). . . . 92
7.20 Mean distance ($\pm 1\sigma$, shaded) between subsequent manifold arcs for an arbitrary $L_{2,S}$ Halo orbit . . . 93
7.21 Propagation of manifold arcs for $t = 3$ (left), 5 (middle), 10 (right) [TU] for an arbitrary $L_{2,S}$ Halo orbit . . . 93
7.22 Segmentation division of the $L_{2,S}$ Halo family. The line and hatched region correspond to 2.375 TU and to orbits with periods shorter than 2.375 TU, respectively. . . . 94
7.23 Injective representation of the $L_{2,S}$ Halo family using the Jacobi constant. The green point marks $L_{2,S}$, and the blue point marks the Moon (not to scale). . . . 95
7.24 Architecture pattern used in DNN NAS. Each hidden layer applies Dense $\rightarrow$ Activation $\rightarrow$ Dropout. The final output layer omits activation and dropout to allow unrestricted (including negative) outputs. . . . 97
7.25 Accuracy vs. function evaluations for classical integrators (with/without STM), DNN, KAN. Red dashed line is accuracy requirement. Familial interpolation is cubic at abacus resolution 10^{-3}, 10 [TU] propagation for a single manifold arc. . . . 98
7.26 Log-Log linear neural scaling for invariant manifold surrogate models within explored architectures. . . . 99

8.1 Node–edge graph of the JPL-A subset derived from VAE-based clustering, used for verification against Figure 6.4. . . . 101
8.2 Demonstration of Latent Arithmetic for Spacecrat Mission Design . . . 101
8.3 Latent arithmetic applied to Halo Butterfly. . . . 102
8.4 Comparison of an assumed isotropic Gaussian prior (left) and trainable Riemannian latent prior (right) for the same data distribution (blue dots) [79]. . . . 103
8.5 Preliminary demonstration for normalization limited to the $L_{2,N/S}$ range, decoded samples 103
8.6 Periodic trajectory of an electron in an electromagnetic control system [80][Modified] . . 104

A.1 Projection of the $L_{2,N}$ Halo family. . . . 112
A.2 Broucke stability diagram for the $L_{2,N}$ Halo family. . . . 113
A.3 Zoomed Broucke stability diagram for the $L_{2,N}$ Halo family. . . . 113
A.4 Projection of the L_2 Lyapunov family. . . . 114
A.5 Broucke stability diagram for the L_2 Lyapunov family. . . . 114
A.6 Projection of the L_2 Vertical family. . . . 115

A.7 Broucke stability diagram for the L_2 Vertical family. 115
A.8 Projection of the L_2 Axial family. 116
A.9 Broucke stability diagram for the L_2 Axial family. 116
A.10 Projection of the L_2 Dragonfly family. 117
A.11 Broucke stability diagram for the L_2 Dragonfly family. 117
A.12 Projection of the L_2 Butterfly family. 118
A.13 Broucke stability diagram for the L_2 Butterfly family. 118
A.14 Zoomed Broucke stability diagram for the L_2 Butterfly family. 119
A.15 Subpar node placement, N = 4 nodes/segments 119
A.16 Non-subpar node placement, N = 5 nodes/segments 120
A.17 Position error norm at 10TU for various integrators and integration tolerances. Red line denotes positional accuracy requirement, $\Delta p < 1 \times 10^{-5}[\mathrm{LU}]$ 120
A.18 Velocity error norm at 10TU for various integrators and integration tolerances. Red line denotes velocity accuracy requirement, $\Delta v < 1 \times 10^{-5}\ [\mathrm{LU/TU}]$ 121
A.19 Jacobi constant error at 10TU for various integrators and integration tolerances. Red line denotes energy. Red line denotes energy preservation requirement, $\Delta C < 1\%$ . . 121

List of Tables

2.1 Unit conversion factors for the Earth-Moon CR3BP [27]. 6
2.2 Summary of stability characteristics based on the stability index ν. 11
2.3 Types of heteroclinic-like connections and associated manoeuvre requirements. 14
2.4 Bifurcation types, corresponding lines, and color transitions [34] 23

3.1 Percentage of Successful CR3BP - EHF Transitions by Family and Period Amount (Rev) [40] 29
3.2 Order of magnitude of external perturbations in dimensional and non-dimensional units [39, 41] 29

4.1 AUC value for the best model of each algorithm explored in the classification of stability regimes [59] 41

5.1 Summary of CR3BP to High-Fidelity (HF) Orbit Positional Differences for Earth-Moon Halo Families [70, 71] 49

6.1 Classification and original dataset count of periodic orbit families by associated branches or symmetry types [27] 51
6.2 Node Numbering Family and Type Overview 53
6.3 Summary of integrators in `scipy.integrate.solve_ivp`. 57
6.4 Comparison of time-series feature handling options for clustering/classification 59
6.5 Summary of available datasets. 64

7.1 Summary of the 12 NAS configurations. 74
7.2 VAE evaluation metrics 75
7.3 Comparison of JPL-A and JPL accuracy (MAE) metrics per architecture. Lowest values per dataset are bolded. 77
7.4 Neural Surrogate Formulation: Input/Output Parameters 94
7.5 Neural Surrogate Control Parameters and Sampling Rates 96
7.6 Summary of the four NAS configurations. 96
7.7 Evaluation metrics summary 96

A.1 Comparison of CR3BP Normalization Parameters: Franz-Russell vs JPL Databases . . 122
A.2 VAE optimization constants and hyperparameter search space, in combination with their sampling schemes (Bayesian opt). 122
A.3 DNN NAS constants and searched hyperparameters with their sampling schemes. . . . 123
A.4 KAN NAS constants and parameters with their sampling schemes. 123

Nomenclature

Abbreviations

Abbreviation	Definition
3BP	Three-Body Problem
BDF	Backward Differentiation Formula (implicit multistep integrator)
BRUTUS	Arbitrary-precision N-body integrator
CNN	Convolutional Neural Network
CNS	Clean Numerical Simulation
CR3BP	Circular Restricted Three-Body Problem
DBSCAN	Density-Based Spatial Clustering of Applications with Noise
DNN	Deep Neural Network
DOP853	8th-order Dormand–Prince Runge–Kutta method
ELBO	Evidence Lower Bound
ESA	European Space Agency
GA	Genetic Algorithm
GAN	Generative Adversarial Network
GPU	Graphics Processing Unit
HF	High-Fidelity (ephemerides) model
HNN	Hamiltonian Neural Network
HOC	Hybrid Optimal Control
H3BP	Hierarchical Three-Body Problem
ITN	Interplanetary Transport Network
JPL	Jet Propulsion Laboratory
KAN	(Multiplicative) Kolmogorov–Arnold Network
KL	Kullback–Leibler (divergence)
LKCNN	Large-Kernel Convolutional Neural Network
LNN	Lagrangian Neural Network
LTI	Linear Time-Invariant (system)
LSODA	Adams/BDF integrator with automatic switching
LU	Normalized length unit
MBH	Monotonic Basin Hopping
MSDC	Multiple Shooting Differential Correction
NASA	National Aeronautics and Space Administration
NH3BP	Non-Hierarchical Three-Body Problem
NRHO	Near-Rectilinear Halo Orbit
NPV	Natural Parameter Variation (continuation)
PAL	Pseudo-Arc Length (continuation)
PIDM	Physics-Informed Diffusion Model
PINN	Physics-Informed Neural Network
R3BP	Restricted Three-Body Problem
RK23	3(2)-order explicit Runge–Kutta method
RK45	5(4)-order explicit Runge–Kutta method
SI	International System of Units
SJS	Sun–Jupiter–Saturn (three-body setup)
SNOPT	Sparse Nonlinear Optimizer
SQP	Sequential Quadratic Programming

Abbreviation	Definition
SSDC	Single Shooting Differential Correction
STM	State Transition Matrix
TOF	Time of Flight
TU	Normalized time unit
UMAP	Uniform Manifold Approximation and Projection
VAE	Variational Autoencoder
AL	Active Learning
AMD	Angular Momentum Deficit
AUC	Area Under Curve
DT	Decision Tree
HGB	Histogram Gradient Boosting
LBFGS	Limited-memory BFGS optimizer
LightGBM	Light Gradient Boosting Machine
NAS	Neural Architecture Search
RF	Random Forest
ROC	Receiver Operating Characteristic
SEMpy	CR3BP propagation/analysis toolkit (Python)
TimeVAE	Time-series Variational Autoencoder
XGBoost	Extreme Gradient Boosting
E.O.M.	Equations of Motion

Symbols

Symbol	Definition	Unit
$A(t)$	Jacobian of partial derivatives of the E.O.M. w.r.t. the state	[—]
C	Jacobi constant	[—]
$C_{\min},\ C_{\max}$	Minimum/maximum Jacobi constant considered	[—]
d	Distance from P_1 to spacecraft (as defined in E.O.M.)	[LU]
E_r	Relative energy error	[—]
F_{env}	Average function-evaluation rate	[1/s]
G	Gravitational constant	[m^3 kg^{-1} s^{-2}]
H	Hamiltonian	[J]
I_6	6×6 identity matrix	[—]
L_i	Lagrange (libration) points, $i \in \{1, \ldots, 5\}$	[—]
M	Monodromy matrix (Φ over one period)	[—]
$m_1,\ m_2$	Masses of primary (P_1) and secondary (P_2) bodies	[kg]
$N,\ N_t$	Number of samples; number of time steps per sample	[—]
P	Orbital period	[TU]
$r_1,\ r_2$	Distances to P_1 and P_2 (used in U)	[LU]
r	Distance from P_2 to spacecraft (as defined in E.O.M.)	[LU]
T	Set of states along a trajectory interval	[—]
$T_{\max}$	Maximum manifold arc time of flight used	[TU]
$t_0,\ t_i,\ t_f,\ t_P$	Initial, sample, final, and one-period times	[TU]
$\Delta p,\ \Delta v$	Accuracy thresholds in position and velocity	[LU], [LU/TU]
Δs	PAL continuation step length	[—]
ΔV	Total velocity increment	[m/s]
$\Phi(t, t_0)$	State Transition Matrix (STM)	[—]
Φ_T	Set of STMs along a trajectory	[—]
U	Effective potential in the synodic frame	[—]

Symbol	Definition	Unit
V	Speed magnitude $\left(V^2 = \dot{x}^2 + \dot{y}^2 + \dot{z}^2\right)$	[LU/TU]
$\mathbf{X}$	State vector $[x,\, y,\, z,\, \dot{x},\, \dot{y},\, \dot{z}]^\top$	[LU; LU/TU]
$\mathbf{X}_i^S,\ \mathbf{X}_i^U$	Stable/unstable manifold ICs at point i	[as $\mathbf{X}$]
$\delta\mathbf{X}_0$	Initial state correction (SSDC/MSDC)	[as $\mathbf{X}$]
$x,\, y,\, z$	Position components in synodic frame	[LU]
$\dot{x},\, \dot{y},\, \dot{z}$	Velocity components in synodic frame	[LU/TU]
$\ddot{x},\, \ddot{y},\, \ddot{z}$	Acceleration components in synodic frame	[LU/TU^2]
$\vec{q}_i,\, \vec{p}_i$	Canonical position and momentum of body i	[m], [kg m/s]
ΔC	Difference in Jacobi constant	[—]
τ	Convergence tolerance	[—]
$\mathbf{v}_i^S,\ \mathbf{v}_i^U$	Stable/unstable eigenvectors at point i	[—]
$\mathbf{v}_i^{S_E},\ \mathbf{v}_i^{S_I}$	Stable exterior/interior eigenvectors at point i	[—]
$\mathbf{v}_i^{U_E},\ \mathbf{v}_i^{U_I}$	Unstable exterior/interior eigenvectors at point i	[—]
$\hat{\mathbf{v}}_i^{S_E},\ \hat{\mathbf{v}}_i^{S_I}$	Unit stable exterior/interior eigenvectors	[—]
$\hat{\mathbf{v}}_i^{U_E},\ \hat{\mathbf{v}}_i^{U_I}$	Unit unstable exterior/interior eigenvectors	[—]
$\hat{\mathbf{X}}$	Target/desired state	[as $\mathbf{X}$]
$\mathbf{X}_f$	State at final time	[as $\mathbf{X}$]
$\mathbf{X}_i^{S_E},\ \mathbf{X}_i^{S_I}$	Stable exterior/interior manifold ICs at point i	[as $\mathbf{X}$]
$\mathbf{X}_i^{U_E},\ \mathbf{X}_i^{U_I}$	Unstable exterior/interior manifold ICs at point i	[as $\mathbf{X}$]
z_i	Latent variable component i	[—]
$\alpha^{(\mathrm{Broucke})}$	Broucke stability parameter α	[—]
$\beta^{(\mathrm{Broucke})}$	Broucke stability parameter β	[—]
$\beta^{(\mathrm{VAE})}$	Weight on KL term in β-VAE objective	[—]
χ	Manifold branch label (stable/unstable, interior/exterior)	[—]
$\epsilon^{(\mathrm{pert})}$	Manifold perturbation magnitude	[—]
$\boldsymbol{\epsilon}^{(\mathrm{term})}$	Terminal state error vector	[as $\mathbf{X}$]
γ	Physics/PINN loss weight	[—]
κ	Class ID of parent orbit	[—]
λ	Eigenvalue of Φ or M	[—]
λ_{max}	Eigenvalue of largest magnitude	[—]
μ	Mass parameter $\mu = \frac{m_2}{m_1+m_2}$	[—]
ν	Stability index	[—]
$\theta,\, \phi$	Model parameters (e.g., VAE encoder/decoder)	[—]
ξ	Relative position along parent orbit	[—]

Orbit Families

Family Notation	Definition
L_1 Lyapunov	L1 Lyapunov orbit family
L_2 Lyapunov	L2 Lyapunov orbit family
L_3 Lyapunov	L3 Lyapunov orbit family
$L_{1,N}$ Halo	L1 Northern Halo orbit family
$L_{1,S}$ Halo	L1 Southern Halo orbit family
$L_{2,N}$ Halo	L2 Northern Halo orbit family
$L_{2,S}$ Halo	L2 Southern Halo orbit family
$L_{3,N}$ Halo	L3 Northern Halo orbit family
$L_{3,S}$ Halo	L3 Southern Halo orbit family
L_1 Vertical	L1 Vertical orbit family
L_2 Vertical	L2 Vertical orbit family
L_3 Vertical	L3 Vertical orbit family

Family Notation	Definition
L_1 Axial	L1 Axial orbit family
L_2 Axial	L2 Axial orbit family
L_3 Axial	L3 Axial orbit family
L_4 Long Period	L4 Long Period orbit family
L_5 Long Period	L5 Long Period orbit family
L_4 Short Period	L4 Short Period orbit family
L_5 Short Period	L5 Short Period orbit family
L_4 Vertical	L4 Vertical orbit family
L_5 Vertical	L5 Vertical orbit family
L_4 Axial	L4 Axial orbit family
L_5 Axial	L5 Axial orbit family
$L_{2,N}$ Butterfly	L2 Northern Butterfly orbit family
$L_{2,S}$ Butterfly	L2 Southern Butterfly orbit family
$L_{2,N}$ Dragonfly	L2 Northern Dragonfly orbit family
$L_{2,S}$ Dragonfly	L2 Southern Dragonfly orbit family
DRO	Distant Retrograde orbit family
DPO	Distant Prograde orbit family
LPO	Low Prograde orbit family

1

Introduction

1.1. Introduction

The gravitational three-body problem (3BP) has long served as an example of deterministic chaos in classical mechanics. This is due to inherent nonlinearity and sensitivity to initial conditions of the problem, with the general solution of the 3BP remaining analytically intractable, meaning it can only be solved through numerical integration. The progression of computing power in the last few decades has greatly improved knowledge of the both 3BP and other N-body systems [1, 2]. Numerical simulations can extend predictions over finite time periods, but due to the system's intrinsic sensitivity, small numerical errors can still rapidly propagate and grow exponentially, limiting the reliability of individual simulations [3]. Despite this, new studies have emerged focused on periodicity, long-term stability, orbital resonances, and scattering. Nonetheless, these studies remain largely hindered by the large amount of computational resources required to achieve the required accuracy levels [3, 4].

At its core, the 3BP describes how three gravitating bodies influence one another's motion, which is in turn foundational to understanding the formation of planetary and star systems, or for understanding the motion of a natural or artificial satellite in these systems. Observations and modelling of both circumbinary P-Type (orbiting two stars) and circumstellar S-type (orbiting one star of a binary) planets rely on precise 3BP modeling [5]. Additionally, understanding the dynamics of planetary formation is heavily reliant on such simulations, in particular for understanding both terrestrial planet and hot Jupiter planet formation [6, 7]. These investigations shape our understanding of the eccentricity distributions of planetary and stellar systems, with downstream implications for the dynamics and frequency of gravitational wave mergers [6, 8].

Beyond formation dynamics, 3BP research has played a central role in spacecraft mission design over the past several decades [9]. This research has enabled placement of spacecraft in gravitationally favourable locations where fuel consumption for station-keeping is minimized, with application already demonstrated for the James Webb Space Telescope, Hiten, Genesis, Moonrise, Whipple and Bepi-Colombo missions, amongst others [10, 9]. Invariant manifolds associated with unstable periodic orbits in the CR3BP act as natural low-energy transfer paths to and from periodic orbits, and accurate understanding of CR3BP dynamics is critical in identifying them, for example used in development of the Interplanetary Transport Network (ITN) [9, 11, 12, 13].

Recent advancements in artificial intelligence techniques offer powerful tools to facilitate simulation of these problems orders of magnitude faster than traditional numerical integration techniques [2, 14]. Notably, these come in the form of advancements in regression and generation schemes for time-series data. These advancements should facilitate more rapid exploration of structures in the solution space of the 3BP through reductions in computational cost, enabling mission design utilizing specific families of periodic orbits, regions of stability, and suitable transfer conditions with far greater efficiency.

1.2. Context: 3BP Variations

Understanding the 3BP in full requires consideration of the wide arrangement of scenarios it encompasses. Due to the problem's sensitivity to initial conditions and lack of a general analytical solution, numerous formulations and simplifications have been developed to better understand and simulate three-body dynamics. These formulations each capture distinct physical arrangements, geometric configurations, or dynamical behaviours, and are motivated by both practical applications and potential for theoretical exploration. The most notable variants are outlined as follows in Section 1.2.1 - 1.2.4.

1.2.1. Hierarchical 3BP (H3BP)

The Hierarchical Three-Body Problem (H3BP) contains configurations where two bodies of similar size form a close binary system while the third body is located at a relatively larger distance. This hierarchy allows simplifications by treating the inner binary and the distant third body as two separate two-body problems with perturbative interactions, decomposing the problem into an unperturbed part and a perturbing part [15]. These systems are furthermore also typically characterized by Kozai-Lidov oscillations, where the third body induces oscillations in the inclination and eccentricity of the inner binary [16]. In addition to this, these systems are much more common in nature than their non-hierarchical counterpart, as they tend to be more dynamically stable [15].

1.2.2. Non-Hierarchical 3BP (NH3BP)

In the Non-Hierarchical 3BP (NH3BP), all bodies have comparable masses and distances between them, leading to more chaotic dynamics. Classical investigation into the NH3BP yielded the information that these triples almost always decay into a set of binaries and an ejected body, or long term quasi-stability as a hierarchical pair [17]. This regime represents the most chaotic manifestation of the three-body problem and lacks effective regularization techniques, which necessitates reliance on direct numerical integration in most studies. Achieving the required accuracy in such large-scale simulations is particularly computationally demanding when compared to other 3BP variants due to the rate of growth of numerical errors during propagation [18]. The NH3BP is however frequently employed as a benchmark for testing and validating simulation algorithms due to this numerical complexity and sensitivity, making it an ideal test scenario for assessing the accuracy, stability, robustness, and performance of integration methods [19].

1.2.3. Restricted 3BP (R3BP)

The Restricted 3BP (R3BP) is used for the investigation of the motion of a negligible mass body relative to two significant primary masses. These two primaries are restricted to orbit around their common center of mass, whilst the third body is gravitationally influenced by the primaries. The circular restricted (CR3BP) variant is a special case of this, where the two massive bodies are restricted to move in a circular orbit. This problem is less complex to analyze than the unrestricted or non-hierarchical variant, and allows the equations of motion to be represented in a rotating frame. Furthermore, the existence of Lagrange points, or equilibrium points within the problem provide further avenues for research, particularly notable for their utilization in trajectory design [11, 13, 20]. In addition to this, the CR3BP has complex chaotic structures and regions, with periodic orbits possessing both stable and unstable manifolds. Exploitations of these manifolds enable highly efficient trajectory design [9, 11, 13].

1.2.4. Alternative 3BP Cases

Additional exploration of the 3BP in recent work incorporates the inclusion of non-newtonian terms. These terms can include relativistic terms, or terms to modelling dissipative forces such as radiation pressure, drag, or tidal forces. Interestingly, the introduction of such forces often acts to reduce the chaotic nature of the system, guiding it toward more stable or predictable long-term behaviour [21]. This is likely caused by these dissipative forces limiting the growth of orbital eccentricities, thereby dampening chaotic excursions and interactions, especially of application to formation dynamics of both stars and planets [21, 22].

1.3. Context: Research Applications

The study of 3BP dynamics has several implications beyond purely theoretical interest, ranging from exoplanet migration and eccentricity evolution, explored in Section 1.3.1 and 1.3.2, to spacecraft trajectory optimization, explored in Section 1.3.3

1.3.1. Planetary Migration

Traditional planetary formation models face several challenges arising from the observation of bodies whose sizes and locations are difficult to reconcile with in-situ formation theories. A prominent example is the existence of hot Jupiters, an exoplanet class characterized by close orbits and short orbital periods (< 10 days), which are not readily explained by conventional formation mechanisms [7]. Their existence challenges traditional models of planetary formation, as in-situ formation at such close distances to their star are inhibited by extreme conditions such as high temperature and low disk mass density. Various models have been proposed to understand their formation, such as high-eccentricity migration via Kozai-Lidov oscillations, planet-planet scattering, and secular chaos, which attributes migration to secular interactions redistributing the angular momentum deficit of a system. This in turn allows eccentricity and inclination variations over long timescales, however current simulations often experience truncation errors at ≈ 1 Gyr due to computational limitations, raising the question of how chaotic diffusion rates depend on planet mass/radius [7, 16].

Furthermore, explaining planet formation and stability in binary star systems is another prominent area of research. For planet formation within binary star systems, there are two commonly accepted orbital architectures, S-type orbits and P-type orbits. Extensive simulations have been performed to map out stability regions in binary systems for both orbital types, resulting in empirical formulas for the critical required value of the semi-major axis for which a P-type planets cannot maintain their stable orbit, or for which a S-type planets cannot remain bound to a single star [23, 24]. Planet–planet scattering outcomes (e.g. one planet being ejected vs. two planets remaining) can differ statistically in binaries versus single-star systems, however these statistics are inherently dependent on an ergodic phase space, which has been demonstrated to not always be the case [8, 23, 25].

1.3.2. Eccentricity Distributions

Apart from planetary migration, the dynamics within triple bodies systems (e.g. triple-star systems or black hole triples) has significant impact on gravitational wave sources. In dense clusters, binary systems can acquire a third body, forming a hierarchical triple. Kozai–Lidov oscillations in such a triple can drive the inner binary to extremely high eccentricity, causing it to merge via gravitational waves on much shorter timescales than it otherwise would [16, 8]. The third body thus trades angular momentum back and forth with the binary so that in some cycles the inner pair comes close enough to release a burst of gravitational waves, gradually tightening until coalescence, resulting in a merger that occurs on a highly eccentric orbit. These dynamics have been demonstrated to produce mergers/gravitational wave sources with properties matching LIGO/Virgo observations [16]. Recent investigations warrant a reassessment of the expected eccentricity distribution, due to the introduction of biases as a result of orbital regularities.

1.3.3. Trajectory Optimization

Space mission design has long leveraged the dynamics of three-body systems, particularly by investigating the inherent instabilities that arise in these regimes. Key applications, as highlighted upon in Section 1.1, include the use of periodic orbits about equilibrium points for observation platforms, station-keeping optimization for satellites, and leveraging invariant manifolds for low-fuel trajectory optimization [10]. A prominent example is the family of near-rectilinear halo orbits (NRHOs) about L_2, which form a subset of the $L_{2,N/S}$ Halo family and are currently being considered for the Lunar Gateway due to their favourable stability properties over months to years, resulting in minimal station-keeping requirements [26]. Furthermore, numerous other missions such as Hiten, Genesis and BepiColombo have utilized periodic three-body orbits as part of their trajectory design strategies [10].

As stated, transfer trajectory design can benefit from leveraging invariant manifolds for low-energy transfers between periodic orbits. This form of trajectory design is often formulated as a constrained single-objective optimization problem, minimizing the velocity increment (ΔV) required to patch tra-

jectories between periodic orbits. This however requires substantial mapping and collection of both periodic orbits and their associated manifolds for optimization purposes [27, 28]. Traditionally, manifold matching is performed through generation of a dense set of trajectories along the unstable manifold of the departure orbit and the stable manifold of the arrival orbit. These manifold trajectories are propagated forward and backward in time, respectively, to create a manifold surface in phase space. To find matches, current approaches use global search algorithms combined with local optimization refinements. Monotonic Basin Hopping (MBH) or Genetic Algorithms (GA) are used to explore the search space globally, prior to having local gradient-based solvers like Sequential Quadratic Programming (SQP) or SNOPT solver applied to refine the trajectory and enforce continuity and dynamical constraints [13, 29, 30]. An example of such an minimal fuel optimized transfer trajectory between $L_{1,N}$ and $L_{2,S}$ Halo orbits is presented in Figure 1.1. The trajectory departs the $L_{1,N}$ Halo orbit (black) via an initial burn (red star), follows its unstable invariant manifold (green), and then executes a second burn (blue star) to transition onto the stable invariant manifold (red) of the $L_{2,S}$ Halo orbit (black). A final insertion burn (green star) places the spacecraft into the target orbit.

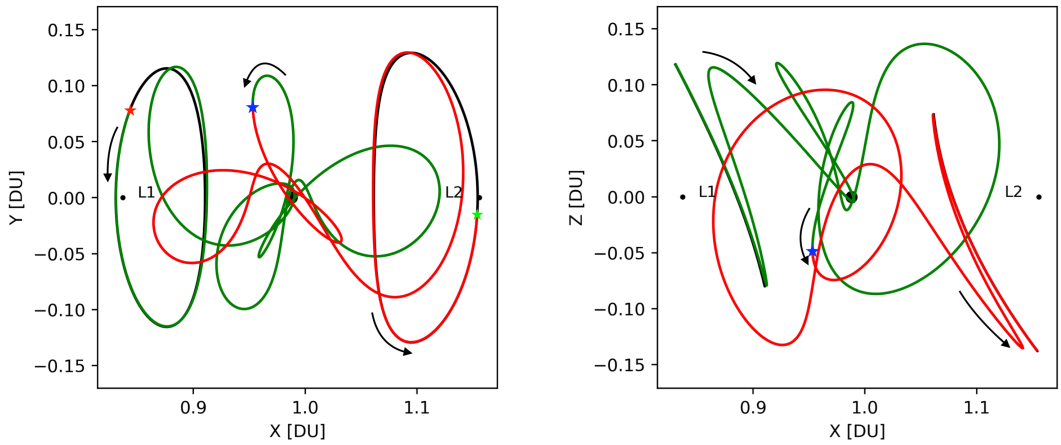


Figure 1.1: Example of a transfer trajectory based on invariant-manifolds between two periodic orbits ($L_{1,N}$ Halo, $L_{2,S}$ Halo) [13] [Modified]

Discovery of new periodic orbits, and orbital families within the CR3BP is thus incredibly useful for further expanding the available transfer options within mission design frameworks, whilst invariant manifolds offer low cost transfer between periodic orbits [11, 12, 13]. As a result, the discovery and exploitation of new periodic orbit families is not purely of theoretical interest, but constitutes an enabling technology for low-thrust, long-duration transfer trajectories. This shall be the focus of this investigation, with Chapter 2 presenting the dynamical model of the system and the key analytical and numerical tools used in its study. Chapter 4 surveys active research areas relevant to computationally efficient exploration of periodic orbits and manifold-based transfers. Chapter 5 then details the novel research proposed in this thesis, building upon the theoretical foundations and context established in the preceding chapters.

2

Dynamical Model

This chapter establishes the mathematical and dynamical framework within which the investigation is conducted. The CR3BP offers an idealized representation of spacecraft motion under the gravitational influence of two massive primaries, enabling the analysis of fundamental structures such as equilibrium points, periodic orbits, and their associated invariant manifolds. Notable examples of such systems for their relevance in spacecraft mission design include the Earth-Moon system (considered in this investigation), the Sun-Earth system, the Sun-Mars system, amongst others. First, the dynamical structure of the CR3BP is explored, prior to introducing key tools used in the analysis of the CR3BP, forming the dynamical and computational foundation for trajectory design developed in subsequent chapters.

2.1. Synodic Reference Frame

The equations of motion of the CR3BP are derived for a non-inertial rotating reference frame. This frame, known as the synodic reference frame, presented in Figure 2.1, has its origin at the barycenter of the two primary bodies (M_1, M_2) and rotates with the constant angular velocity (mean motion) of the primaries about that barycenter. The barycenter is the center of mass of a gravitationally interacting system, representing the point at which the weighted position vectors of all masses sum to zero when expressed relative to it. This point is inversely proportional to their mass ratio, such that each body traces an orbit about this common point. In the CR3BP, as the mass of the spacecraft is assumed to be negligible, this barycenter lies along the line connecting the two primaries bodies.

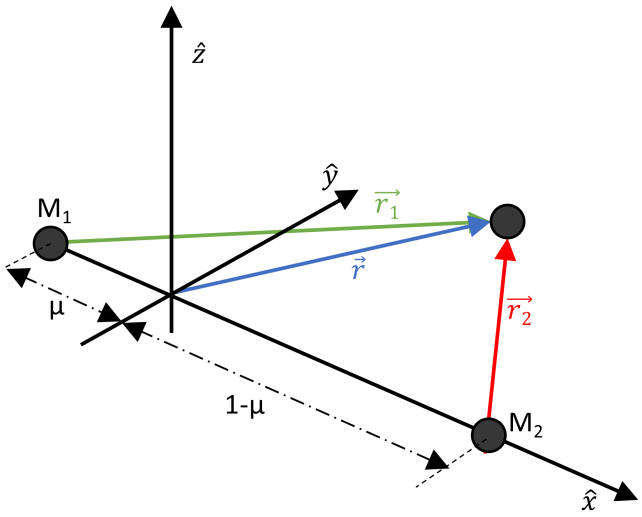


Figure 2.1: Geometry of the CR3BP in a rotating reference frame [31]

Adopting commonly used notation, the reference frame in Figure 2.1 is defined such that the x-axis always points from the larger body (M_1) to the smaller body (M_2), hence the line connecting them always lies along the x-axis. The z-axis is aligned with the angular momentum vector of the system, perpendicular to the orbital plane, while the y-axis completes the right-handed coordinate system. This definition results in a frame that rotates at a rate dictated by the relative motion of the two bodies in their orbits about the barycenter. As a result, if the bodies follow elliptical orbits, the rotation rate of this reference frame varies accordingly. In the case that both bodies move in circular orbits around their barycenter, such as that present in the CR3BP, the rotation rate is constant, simplifying the equations of motion and eventual frame transformations [32].

2.2. Equations of Motion

In the CR3BP, the motion of an infinitesimally small third body (spacecraft), M_3, is studied under the gravitational influence of two significantly more massive bodies, M_1 and M_2. M_1 and M_2 are defined as the bodies such that the masses $M_2 < M_1$. The dynamics of the spacecraft, M_3, are described by a set of second-order differential equations of motion (EOM) in the synodic reference frame [32]:

$$\ddot{x} - 2\dot{y} - x = -\frac{(1-\mu)(x+\mu)}{d^3} - \frac{\mu(x-1+\mu)}{r^3}, \tag{2.1}$$

$$\ddot{y} + 2\dot{x} - y = -\frac{(1-\mu)}{d^3}y - \frac{\mu}{r^3}y, \tag{2.2}$$

$$\ddot{z} = -\frac{(1-\mu)}{d^3}z - \frac{\mu}{r^3}z \tag{2.3}$$

Where r_1,r_2, and mass ratio μ are defined as:

$$r_1 = \sqrt{(x+\mu)^2 + y^2 + z^2}, \tag{2.4}$$

$$r_2 = \sqrt{(x-1+\mu)^2 + y^2 + z^2}, \tag{2.5}$$

$$\mu = \frac{M_2}{M_1 + M_2} \tag{2.6}$$

Where μ is used to scale the distance from the primaries M_1, M_2 to the barycenter in the synodic reference frame. $\mathbf{X}$ is then defined as the six-element state vector:

$$\mathbf{X} = \begin{bmatrix} x & y & z & \dot{x} & \dot{y} & \dot{z} \end{bmatrix}^{\mathrm{T}} = \begin{bmatrix} x & y & z & v_x & v_y & v_z \end{bmatrix}^{\mathrm{T}} \in \mathbb{R}^6 \tag{2.7}$$

The trajectory of any M_3 is deterministic in time if $\mathbf{X}$ is defined. The non-dimensional length unit ([LU]), is set to the distance between the two primaries, whilst the non-dimensional time unit ([TU]) is set such that the angular rate of the synodic frame (the mean motion of the two primaries about their barycenter) is 1. For the Earth-Moon system, the mass ratio $\mu \approx 0.01215$, results in the conversion factors between non-dimensional length/time units (LU/TU) and SI units presented in Table 2.1.

Table 2.1: Unit conversion factors for the Earth-Moon CR3BP [27].

Non-dimensional → SI	**Value**	**SI → Non-dimensional**	**Value**
LU → km	389703.26	km → LU	2.5661×10^{-6}
LU → m	3.89703×10^{8}	m → LU	2.5661×10^{-9}
TU → s	382981	s → TU	2.6111×10^{-6}
TU → days	4.4327	days → TU	0.22560

2.3. Jacobi Constant (C)

In the CR3BP, an integral of motion exists as the Jacobi constant (C), and acts as a conserved quantity in time (i.e. $\frac{\partial C}{\partial t} = 0$), and corresponds to the energy level of any M_3 in this system. The Jacobi constant is defined as:

$$C = 2U - V^2 \tag{2.8}$$

Where, U is the effective potential in the synodic frame, and V is the magnitude of the velocity;

$$U = \frac{1}{2}(x^2 + y^2) + \frac{1-\mu}{r_1} + \frac{\mu}{r_2} \tag{2.9}$$

$$V^2 = \dot{x}^2 + \dot{y}^2 + \dot{z}^2 = v_x^2 + v_y^2 + v_z^2 \tag{2.10}$$

The Jacobi constant of a spacecraft in the CR3BP is conserved as long as the spacecraft is influenced exclusively by the gravitational forces of the two primary bodies, and only varies if external perturbations are introduced. This allows C to serve as an efficient way to quantify numerical integration errors throughout propagation, as all changes in C can be attributed to numerical errors. The partial derivative with respect to each state is derived as follows:

$$\frac{\partial C}{\partial x} = 2x - \frac{2\left(1-\mu\right)\left(x+\mu\right)}{\left(\left(x+\mu\right)^2 + y^2 + z^2\right)^{3/2}} - \frac{2\mu\ \left(x-(1-\mu)\right)}{\left(\left(x-(1-\mu)\right)^2 + y^2 + z^2\right)^{3/2}} \tag{2.11}$$

$$\frac{\partial C}{\partial y} = 2y - \frac{2\left(1-\mu\right)y}{\left(\left(x+\mu\right)^2 + y^2 + z^2\right)^{3/2}} - \frac{2\mu\, y}{\left(\left(x-(1-\mu)\right)^2 + y^2 + z^2\right)^{3/2}} \tag{2.12}$$

$$\frac{\partial C}{\partial z} = -\frac{2\left(1-\mu\right)z}{\left(\left(x+\mu\right)^2 + y^2 + z^2\right)^{3/2}} - \frac{2\mu\, z}{\left(\left(x-(1-\mu)\right)^2 + y^2 + z^2\right)^{3/2}} \tag{2.13}$$

$$\frac{\partial C}{\partial v_x} = -2v_x \tag{2.14}$$

$$\frac{\partial C}{\partial v_y} = -2v_y \tag{2.15}$$

$$\frac{\partial C}{\partial v_z} = -2v_z \tag{2.16}$$

$$\frac{\partial C}{\partial t} = 0 \tag{2.17}$$

2.4. Lagrange (Libration) Points

For each value of the mass parameter (μ), the system of equations defined in Section 2.2, possesses 5 equilibrium points, or Lagrange/libration points [33]. Figure 2.2 shows the variation of these Lagrange points (coloured per point) with μ in x-y space, for systems with varying relative mass ratios between the two primaries (M_1, M_2). This in turn means for different CR3BP systems, such as the Sun-Earth or Earth-Moon system, the relative position of these points will differ. The specific locations of the Lagrange points for the Earth–Moon system are shown in the right sub-figure, with the corresponding value of μ intersection plane highlighted in the left sub-figure. L_1, L_2 and L_3 are co-linear with the x axis and both bodies, with L_1 between the primaries, with L_2 and $L3$ outside M_2 and M_1 respectively, with respect to the barycenter. L_4 and L_5 form an equilateral triangle bounded by the primaries with the coordinates described by [33]:

$$L_{5,4}(x, y) = (\frac{1}{2} - \mu, \pm\sqrt{\frac{3}{2}}) \tag{2.18}$$

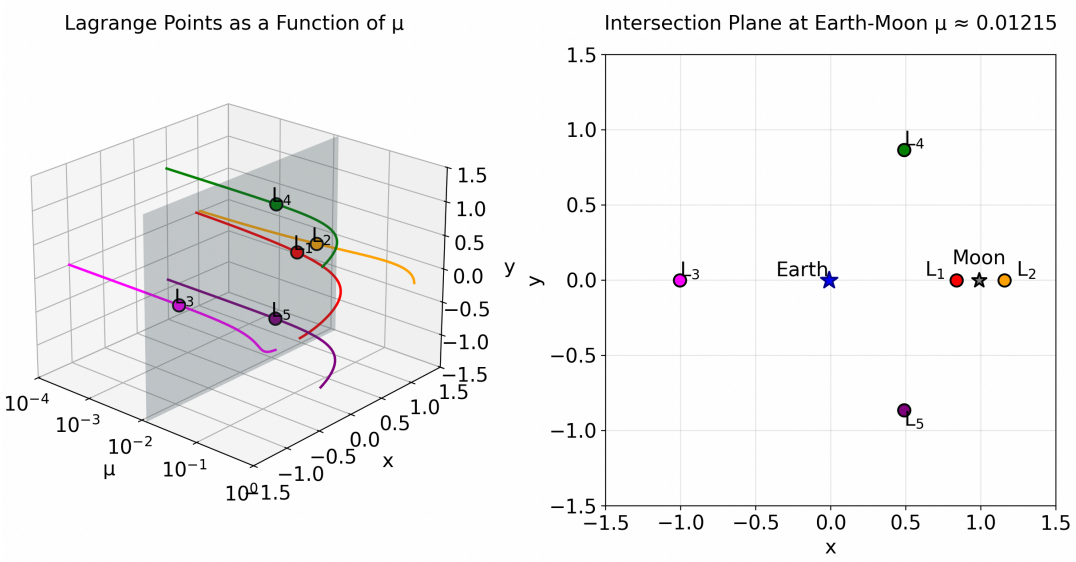


Figure 2.2: Lagrange points as a function of μ, locations at $\mu = 0.01215$

Evaluating the Lagrange points and their associated linear stability characteristics is central to CR3BP analysis, as families of periodic orbits emerge from the local dynamics in their vicinity and are subsequently traced using bifurcation and continuation methods, later detailed in Section 2.5 and Section 2.13. Furthermore, the dependence of Lagrange point locations on the mass parameter μ (as described by the left sub-figure in Figure 2.2 implies that periodic orbits differ between systems in both phase-space structure and family composition. Consequently, periodic orbits that exist in one system (e.g., Earth–Moon) do not necessarily exist in another (e.g., Sun–Mars). As such, while the methods explored in this investigation are applicable to other systems, the specific periodic orbits themselves are not.

2.5. Periodic Orbits

A periodic orbit is defined as a trajectory, or the propagation of a state $\mathbf{X}_0$ in time, such that after a discrete period (P), returns to its initial state, i.e. $\mathbf{X}_0 = \mathbf{X}_P$. There are an infinite amount of periodic solutions within the CR3BP, with numerous families described by continuous changes along a single or multiple natural parameters (e.g. state, stability, period) describing the orbit [34]. In practice, it is difficult to identify the true $\mathbf{X}_0$ due to numerical instabilities, as a result $\mathbf{X}_0$ is identified to a pre-defined tolerance (ϵ), often dictated by the rounding error of propagating the system. This state error is typically set to be within $\epsilon \leq 10^{-12}$ [35]. This lack of precise identification of $\mathbf{X}_0$ results in the periodic orbit becoming unbounded ($||\mathbf{X}_0 - \mathbf{X}_P|| \geq \epsilon$) after N periods due to not exactly pinpointing the true state due to the growth of numerical instabilities, where N is an arbitrarily large amount of periods as a result of the system tolerance.

Orbital families are periodic orbits that share similar shapes, behaviours, or dynamical characteristics. Each family is continuous, with one orbit smoothly transitioning into another within the same family by gradually changing parameters like the Jacobi (C) or state ($\mathbf{X}_0$). This gradual transition between orbits, and between families is explored through the process of familial continuation (Section 2.11) and bifurcation theory (Section 2.13). Furthermore, it should be noted that nomenclature adopts specific names for sub-families of periodic orbits. These sub-families are arbitrarily defined by mission designers for a range of orbits with specific desirable dynamical traits within a family, such as long term stability or position in phase space. An example of this is the Near Rectilinear Halo Orbit (NRHO) family, a subset of the L_2 Halo family characterized by highly elliptical, near-rectilinear motion that brings the orbiting body close to M_2, and a relatively high long term stability compared to other members of the L_2 Halo family [36].

The CR3BP features multiple axes of symmetry about which periodic orbits are found. Many families of periodic orbits exhibit symmetry themselves, often about specific planes such as the x–z or x–y plane. For instance, multiple families of orbits possess symmetry about the x-y plane, with their mirrored counterparts being valid solutions with a negative z position and velocity. The families that arise as a result of this type of symmetry are frequently known as *North* or *South* families of their respective type. Some periodic orbits also exhibit inherent symmetry, most frequently possessing symmetry about the x-z plane. This symmetry allows mission designers to only propagate half of the orbital period $(\frac{P}{2})$ for a complete characterization of the orbit, reducing computational effort.

There are various well known families that arise from bifurcations from the libration points, such as the Lyapunov, Halo and Vertical families, amongst others. Libration point orbits are among the most common types of orbits used in spacecraft mission design as a result of their dynamical characteristics [33]. The gradual transition between members in the $L_{2,N}$ Halo family is demonstrated in Figure 2.3, where various orbital configurations in both phase space and energy levels (as measured by the Jacobi constant) are presented as the evolution of a full orbit in space (x,y,z position) in time (closed loop, constant colouring by *C*). Family members that are close in phase space and have similar Jacobi constants C exhibit strong resemblance, whereas members located at opposite ends of the family (e.g., those with $C \approx 3.15$, yellow) differ substantially in their phase-space structure. Refer to Appendix A for three-dimensional spatial representations and projections of all periodic orbit families considered in this investigation, as described in Section 6.1.

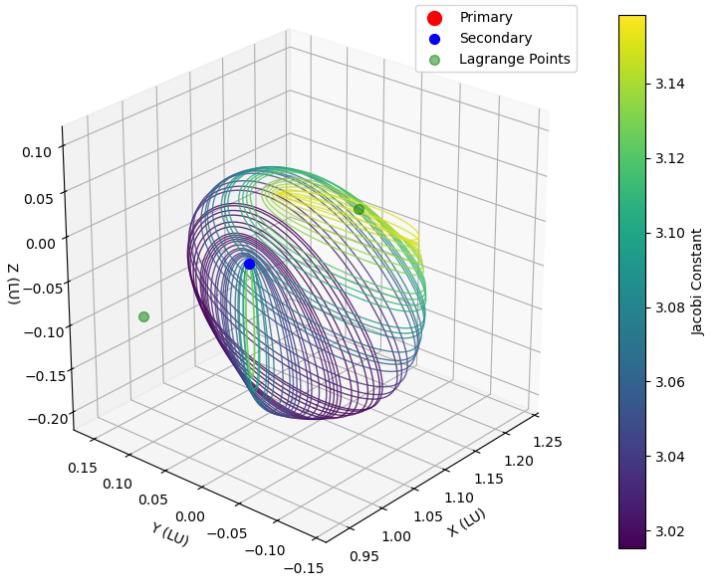


Figure 2.3: $L_{2,N}$ Halo family, spatial representation, coloured by Jacobi Constant (C).

2.6. State Transition Matrix and Monodromy Matrix

The State Transition Matrix (STM) is defined as a matrix Φ, and is used as a first order linearization of the EOM, relating two states in time during propagation. The STM (Φ) is a 6×6 matrix composed of the partial derivatives of the state:

$$\Phi(t,t_0) = \frac{\partial \mathbf{X}(t)}{\partial \mathbf{X}(t_0)} = \begin{bmatrix} \frac{\partial x(t)}{\partial x(t_0)} & \frac{\partial x(t)}{\partial y(t_0)} & \frac{\partial x(t)}{\partial z(t_0)} & \frac{\partial x(t)}{\partial \dot{x}(t_0)} & \frac{\partial x(t)}{\partial \dot{y}(t_0)} & \frac{\partial x(t)}{\partial \dot{z}(t_0)} \\ \frac{\partial y(t)}{\partial x(t_0)} & \frac{\partial y(t)}{\partial y(t_0)} & \frac{\partial y(t)}{\partial z(t_0)} & \frac{\partial y(t)}{\partial \dot{x}(t_0)} & \frac{\partial y(t)}{\partial \dot{y}(t_0)} & \frac{\partial y(t)}{\partial \dot{z}(t_0)} \\ \frac{\partial z(t)}{\partial x(t_0)} & \frac{\partial z(t)}{\partial y(t_0)} & \frac{\partial z(t)}{\partial z(t_0)} & \frac{\partial z(t)}{\partial \dot{x}(t_0)} & \frac{\partial z(t)}{\partial \dot{y}(t_0)} & \frac{\partial z(t)}{\partial \dot{z}(t_0)} \\ \frac{\partial \dot{x}(t)}{\partial x(t_0)} & \frac{\partial \dot{x}(t)}{\partial y(t_0)} & \frac{\partial \dot{x}(t)}{\partial z(t_0)} & \frac{\partial \dot{x}(t)}{\partial \dot{x}(t_0)} & \frac{\partial \dot{x}(t)}{\partial \dot{y}(t_0)} & \frac{\partial \dot{x}(t)}{\partial \dot{z}(t_0)} \\ \frac{\partial \dot{y}(t)}{\partial x(t_0)} & \frac{\partial \dot{y}(t)}{\partial y(t_0)} & \frac{\partial \dot{y}(t)}{\partial z(t_0)} & \frac{\partial \dot{y}(t)}{\partial \dot{x}(t_0)} & \frac{\partial \dot{y}(t)}{\partial \dot{y}(t_0)} & \frac{\partial \dot{y}(t)}{\partial \dot{z}(t_0)} \\ \frac{\partial \dot{z}(t)}{\partial x(t_0)} & \frac{\partial \dot{z}(t)}{\partial y(t_0)} & \frac{\partial \dot{z}(t)}{\partial z(t_0)} & \frac{\partial \dot{z}(t)}{\partial \dot{x}(t_0)} & \frac{\partial \dot{z}(t)}{\partial \dot{y}(t_0)} & \frac{\partial \dot{z}(t)}{\partial \dot{z}(t_0)} \end{bmatrix} \tag{2.19}$$

Where for t_0 and any other arbitrary point in time

$$\Phi(t_i, t_i) = I_6 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.20}$$

The STM is used to relate variations in the initial state, $\mathbf{X}(t_0)$, to variations to the state at time t, $\mathbf{X}(t)$, for a multitude of reasons. These reasons include, but are not limited to, information about the stability of an orbit (Section 2.7), information about the eigenvectors of an orbit (Section 2.8), differential correction strategies (Section 2.10), and detecting bifurcations (Section 2.13).

In order to be able to evaluate Φ in time, Φ is numerically integrated concurrently with the trajectory's dynamics in time. To propagate Φ in time, the following relationship is used:

$$\dot{\Phi}(t, t_0) = A(t)\Phi(t, t_0) \tag{2.21}$$

Where $A(t)$ is the Jacobian matrix of the partials of the equations of motion with respect to the states evaluated at time t;

$$A(t) = \frac{\partial \dot{\mathbf{X}}(t)}{\partial \mathbf{X}(t)} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ \frac{\partial \ddot{x}}{\partial x} & \frac{\partial \ddot{x}}{\partial y} & \frac{\partial \ddot{x}}{\partial z} & 0 & 2 & 0 \\ \frac{\partial \ddot{y}}{\partial x} & \frac{\partial \ddot{y}}{\partial y} & \frac{\partial \ddot{y}}{\partial z} & -2 & 0 & 0 \\ \frac{\partial \ddot{z}}{\partial x} & \frac{\partial \ddot{z}}{\partial y} & \frac{\partial \ddot{z}}{\partial z} & 0 & 0 & 0 \end{bmatrix} \tag{2.22}$$

In turn, the Monodromy matrix (M) is defined for periodic orbits in the CR3BP, as the state transition matrix propagated over an entire orbital period (P) for that orbit:

$$M = \Phi(t_0 + P, t_0) = \Phi(t_P, t_0) \tag{2.23}$$

As the system and its associated periodic orbits in the CR3BP are symplectic (geometry is conserved in phase space, meaning $M^\top J M = J$, where $J = \begin{bmatrix} 0 & I_3 \\ -I_3 & 0 \end{bmatrix}$), the eigenvalues of the state transition matrix occur in reciprocal pairs. An example of this is that if λ is an eigenvalue, then so is $\frac{1}{\lambda}$. Additionally, due to the existence of the Jacobi integral, a pair of eigenvalues always exists and is equal to unity (λ):

$$\text{spec}(\Phi) = \{1,\ 1,\ \lambda_1,\ \lambda_1^{-1},\ \lambda_2,\ \lambda_2^{-1}\} \tag{2.24}$$

There are thus two non-zero stable, and two non-zero unstable eigenvalues, along with their corresponding eigenvectors. The order of instability refers to the number of eigenvalue pairs of the monodromy matrix that lie outside the unit circle in the complex plane, see Figure 2.4 [34]. For example, a first-order unstable orbit is characterized by a single eigenvalue pair outside the unit circle, while the remaining pair lies inside it.

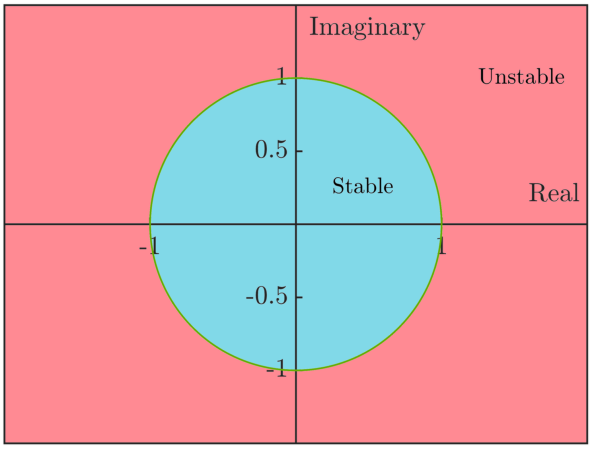


Figure 2.4: Unit circle of the monodromy matrix [34]

2.7. Stability Index

The eigenvalues of the monodromy matrix serve as the basis for a stability measure of a periodic orbit, referred to as the stability index (ν). This index exploits the property that the system's eigenvalues appear in reciprocal pairs and is defined as:

$$\nu = \frac{1}{2}\left(|\lambda_{\text{max}}| + \left|\frac{1}{\lambda_{\text{max}}}\right|\right) \tag{2.25}$$

A periodic orbit is generally considered unstable if $|\nu| > 1$; that is, the magnitude of the associated eigenvalue grows without bound as time progresses. This behaviour implies that any small perturbation to the orbit will cause the spacecraft to diverge from it over time. An orbit can be defined stable if $|\nu| < 1$, i.e., the magnitude of the eigenvector approaches zero as time goes to infinity, resulting in any perturbations being damped away, and the spacecraft returning to the exact same periodic orbit it was in. Practically, this does not occur, instead periodic orbits are typically neutrally stable, where $|\lambda| \approx 1$. In this scenario, the magnitude of the eigenvector does not change and corresponds to the center subspace, resulting in a shift from the original periodic orbit to a similar periodic orbit with a slightly different phase space if perturbations are introduced.

Table 2.2: Summary of stability characteristics based on the stability index ν.

Stability Index ν	**Implication**
$\|\nu\| > 1$	The orbit is unstable. Perturbations grow over time, causing the spacecraft to diverge from the original periodic orbit.
$\|\nu\| < 1$	The orbit is stable. Perturbations are damped out over time, and the spacecraft returns to the original periodic orbit. Not observed in practice.
$\|\nu\| \approx 1$	The orbit is neutrally stable. Perturbations persist without growth or decay, causing the spacecraft to shift to a nearby periodic orbit with slightly different parameters.

2.8. Invariant Manifolds

Each unstable orbit has at least one stable and one unstable eigenvalue, each with an associated eigenvector. If a spacecraft on such an orbit is perturbed in the direction of an unstable eigenvector, it will diverge exponentially from its nominal path. Similarly, a spacecraft can follow a trajectory that approaches the orbit exponentially along the direction associated with the stable eigenvector. These two types of trajectories form the orbit's stable and unstable invariant manifolds, meaning that if whenever the system's state X is initially in the manifold, then all future (or past) trajectories stays in this manifold. As all manifolds intersect the original periodic orbit, and posses a similar energy level (as defined by C), they are largely applied in mission design for the purpose of low-energy transfers to and from these periodic orbits.

To generate the invariant manifolds of an unstable periodic orbit, it is necessary to understand the local stability properties at each point along the orbit. The most computationally efficient manner is if both M and Φ for each point in a periodic orbit have already been computed [34, 32]. The corresponding eigenvectors of the M can be projected via means of the Φ onto the specific point for the orbit in question $(t_0) \rightarrow (t_i)$, where $\mathbf{v_0}$ and $\mathbf{v_i}$ here corresponds to an eigenvector of the system at times t_0 and t_i respectively:

$$\mathbf{v}_i^S = \Phi(t_i, t_0)\, \mathbf{v}_0^S, \qquad \mathbf{v}_i^U = \Phi(t_i, t_0)\, \mathbf{v}_0^U \tag{2.26}$$

This projection however requires knowledge of the points for which X_0 and X_i should be stored in advance to avoid recalculating. In order to use the eigenvectors $\mathbf{v}_i^S$ (stable),$\mathbf{v}_i^U$ (unstable) to compute the invariant manifolds, a small perturbation ϵ is applied to the state of the orbit at $\mathbf{X}_i$, and the result is propagated in time. It is important to normalize the vectors to obtain the unit eigenvectors $\hat{\mathbf{v}}_i^S$ and $\hat{\mathbf{v}}_i^U$ that define the directions of the invariant manifolds, ensuring a consistent perturbation is applied to each state in the orbit. The final equations to produce the initial conditions for the stable ($\mathbf{X}_i^S$) and unstable manifolds ($\mathbf{X}_i^U$) at time t_i along the orbit, are thus:

$$\mathbf{X}_i^S = \mathbf{X}_i \pm \epsilon \frac{\mathbf{v}_i^S}{|\mathbf{v}_i^S|} = \mathbf{X}_i \pm \epsilon \hat{\mathbf{v}}_i^S, \qquad \mathbf{X}_i^U = \mathbf{X}_i \pm \epsilon \frac{\mathbf{v}_i^U}{|\mathbf{v}_i^U|} = \mathbf{X}_i \pm \epsilon \hat{\mathbf{v}}_i^U \tag{2.27}$$

Where the sign of the perturbation determines whether the trajectory belongs to the interior or exterior manifold. This direction (interior vs exterior) is arbitrarily defined, and for this purpose of this investigation the exterior manifold will be defined as those with a positive perturbation. A visualization of this perturbation process and the evolving dynamics is demonstrated below in Figure 2.5. The right figure demonstrates the perturbation (arrows; red: unstable exterior, green: stable exterior, orange: unstable interior, yellow: stable interior) that is applied to $\mathbf{X}_i$ (red point) in the direction of the eigenvectors to obtain the initial conditions of the manifolds $\mathbf{X}_i^{U,S}$ (black points). The propagation of these manifold conditions in time (4 [TU]) allow the approach/departure trajectories of the periodic orbit (blue) in the left sub-figure to be obtained (red: unstable exterior, green: stable exterior, orange: unstable interior, yellow: stable interior).

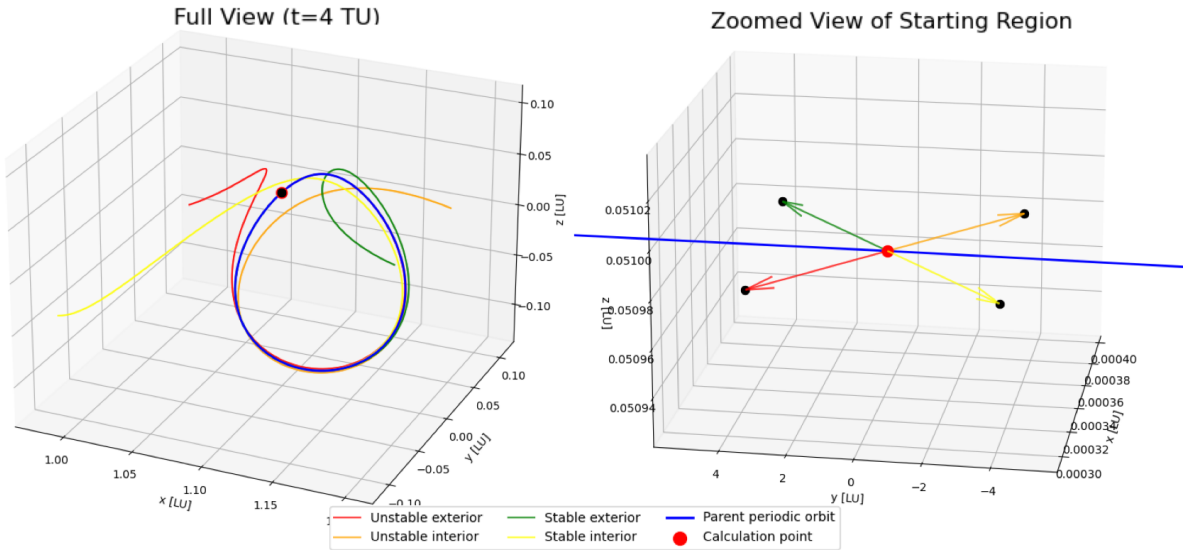
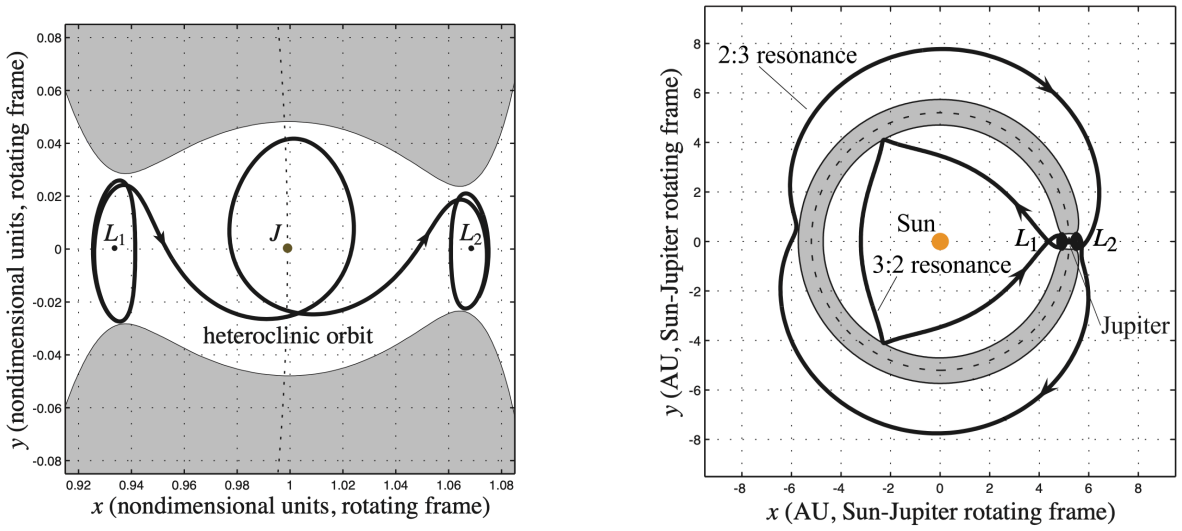


Figure 2.5: Process of applying invariant manifold perturbations for an $L_{2,S}$ Halo orbit. Applied perturbation magnitude is $\epsilon = 10^{-4}$.

To convert invariant manifolds to a practically usable trajectory in the mission design process, positional continuity is ensured in $\mathbf{X}$ with the parent periodic orbit. The motivation is the result of practical considerations, as these perturbations are used as approximations for high-thrust manoeuvres in real-world mission design, which would only result in a Δv in the velocity components of $\mathbf{X}$ [32]

2.9. Hetero/Homoclinic Connections

A common approach to construct a transfer between two periodic orbits using invariant manifolds involves the patching of unstable and stable manifold segments. For example, to transfer a spacecraft from periodic orbit A to periodic orbit B one would first compute the unstable manifold of the departure orbit. This would correspond to perturbing the orbit along its unstable eigenvector at various locations along the orbit, and integrating the trajectory forward in time. Simultaneously, the stable manifold of the target orbit is generated by perturbing along the stable eigenvector and integrating backward in time. These manifolds are then parsed for intersections, where a manifold arc from the departure orbit comes sufficiently close to an arc from the arrival orbit. When such an intersection is found, the resulting trajectory (departure via unstable manifold, approach via stable manifold) is referred to as a heteroclinic connection. An example of a heteroclinic connection is presented in Figure 2.6a, for the Sun-Jupiter system, with a connection departing on the unstable manifold of the L_1 periodic orbit and arriving at the stable manifold of the L_2 periodic orbit, facilitating a flyby of Jupiter (J).



(a) Heteroclinic connection (J: Jupiter)

(b) Homoclinic connection [Modified]

Figure 2.6: Example of heteroclinic and homoclinic connections between periodic orbits in the Sun-Jupiter system. Shaded regions represent forbidden zones determined by the specified Jacobi energy level, where the motion of M_3 is restricted due to lack of energy [37].

True heteroclinic connections are possible between periodic orbits that share the same Jacobi constant, as their corresponding invariant manifolds can naturally intersect in both position and velocity. If the difference in Jacobi constants is too large, it may be infeasible to find a true heteroclinic connection. False heteroclinic connections then become an attractive alternative, and are when the invariant manifolds of two periodic orbits intersect in position space, but not velocity. In this scenario, a high impulse Δv is applied to shift the spacecraft from the departing manifold to the arriving manifold [38]. If there is no intersection in position space, it is often necessary to introduce a patching arc. A patching arc is a (short) transfer segment that bridges the gap between the two manifold arcs. This manoeuvre enables the transition between manifolds but typically requires an additional Δv application moment (two burns required). Patching arcs may also be used strategically to reduce time of flight (TOF), trading energy efficiency (measured by Δv for faster transfers. An example of such a patched transfer is presented in Figure 2.7, where each change in colour corresponds to a burn point, and each coloured segment corresponds to a different coasting phase in the transfer trajectory. This approach is useful for transferring a spacecraft already stationed at a $L_{1,N}$ Halo orbit to an $L_{2,S}$, where the two orbits have different Jacobi constants and no true heteroclinic connection exists. Such transfers to alternative orbits can be highly relevant for mission design, for example when the target orbit offers improved long-term stability, or when the transfer trajectory itself provides a favourable viewing geometry that supports mission objectives.

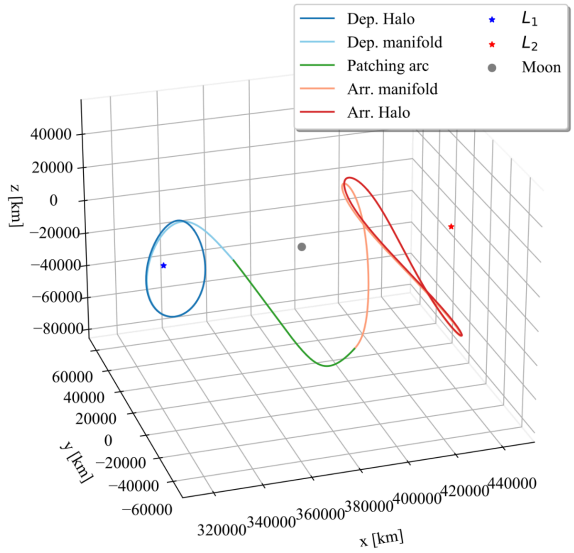


Figure 2.7: Manifold patching arcs between $L_{1,N}$ and $L_{2,S}$ Halo periodic orbits and manifolds [35]

The nomenclature for various types of heteroclinic-like connections considered in this work are summarized in Table 2.3. Each connection type differs in its geometric characteristics and the number of required burn moments ($N(\Delta v)$).

Table 2.3: Types of heteroclinic-like connections and associated manoeuvre requirements.

Connection Type	**Description**	$N(\Delta v)$ **Req.**
True Heteroclinic	Parent periodic orbits share the same Jacobi constant, allowing their stable and unstable invariant manifolds to naturally intersect in both position and velocity. These connections require no major Δv for manifold matching, as the transition occurs dynamically.	2 minor burns (entry into and exit from the manifold).
False Heteroclinic	Parent periodic orbits have different Jacobi constants. Their invariant manifolds intersect in position space only, not velocity. A corrective impulsive manoeuvre is required to shift the spacecraft from the departing manifold trajectory to the arriving manifold trajectory.	2 minor burns (entry/exit into the manifolds) + 1 additional burn between arcs.
Heteroclinic w/ patching arc	Parent periodic orbits have different Jacobi constants and their invariant manifolds do not intersect in either position or velocity. A patching arc is introduced to bridge the gap between the manifold arcs. Patching arcs may also be used to reduce TOF at the cost of higher Δv.	2 minor burns (entry/exit into the manifolds) + 2 burns for entering and exiting the patching arc.

A homoclinic connection is defined as a trajectory that departs from a periodic orbit and after a set TOF, returns to the same periodic orbit. More formally, it is a trajectory that lies in both the unstable and stable manifolds of the same orbit. It asymptotically departs along the unstable manifold and later re-enters along the stable manifold. From a mission design perspective, such connections can be exploited for low-energy loopings or for creating complex orbital paths that repeatedly interact with a reference orbit. Two examples of such trajectories (3:2, 2:3) are presented in Figure 2.6b as homoclinic connections that depart from and return to periodic orbits around Jupiter near the L_1 and L_2 points with 3:2 and 2:3 resonances during the transfer respectively. It should be noted that similar behaviours for homoclinic connections (i.e. True/False homoclinic connections, patching arcs) exist and should be inferred from the heteroclinic connection types and behaviour presented in Table 2.3.

One of the most important applications of hetero/homoclinic connections of invariant manifolds in mission design is in their application in seeding initial guesses for optimization purposes in order to increase the rate of convergence. The optimizer then marginally adjusts the initial conditions, segment

durations, and possibly inserts patching arcs to produce a trajectory that satisfies all required boundary conditions and minimizes/maximises a chosen objective function, typically a minimization problem for total Δv and time of flight (TOF). After optimization refinement, the resulting trajectory typically no longer exactly follows the invariant manifolds, even within the CR3BP.

2.10. Single/Multiple Shooting Differential Correctors

Multiple shooting differential correction (MSDC) algorithms divide the integration domain of a trajectory into several sub-intervals or segments. MSDC has various uses in the context of trajectory optimization in the CR3BP, most notably determining what adjustments to the initial trajectory are required to arrive at the desired converged state. Examples of eventual downstream applications include conversion to high-fidelity solutions (Section 2.12), familial continuation (Section 2.11), and trajectory optimization processes.

In MSDC, each segment is treated as an initial value problem, and the continuity of the full trajectory is enforced by constraining the end state of each segment to match the beginning state of the next. Single shooting differential correction (SSDC) is an equivalent algorithm where the initial state ($\mathbf{X}_0$) is propagated through the entire desired time domain (t_f), considering the entire time domain as a single segment, whilst in MSDC, intermediate nodes $\mathbf{X}_n$ are defined prior to correcting these states. In SSDC, to correct a deviation between a desired target state $\hat{\mathbf{X}}(t_f)$ and the propagated state $\mathbf{X}(t_f)$, a linear correction is applied to the initial state $\mathbf{X}(t_i)$ using the STM $\Phi(t_f, t_i)$, the linearized sensitivity of the final state with respect to changes in the initial state. To do this, the terminal state error must first be defined as:

$$\epsilon_{\mathbf{0}} = \mathbf{X}_0(t_f) - \hat{\mathbf{X}}_0(\hat{t}_f) \tag{2.28}$$

The differential correction to the initial condition is then computed as:

$$\delta\mathbf{X}_0 = -\Phi(t_f, t_i)^{-1} \cdot \epsilon \tag{2.29}$$

This provides a first-order estimate of how the initial condition should be adjusted to reduce the terminal state error:

$$\mathbf{X}_0^{(k+1)}(t_i) = \mathbf{X}_0^{(k)}(t_i) + \delta\mathbf{X}_0, \tag{2.30}$$

This is presented in Figure 2.8, where T is the set of all states propagated between t_i and t_f, Φ_T is the set of all Φ propagated between t_i and t_f, $\mathbf{X}(t_i)$ and $\mathbf{X}(t_f)$ are the initial and final states of the propagation, $\hat{\mathbf{X}}(t_i)$ and $\hat{\mathbf{X}}(t_f)$ are the desired initial and final states, and ϵ is the terminal state error (difference of the state vector at t_f).

$$T = \{\mathbf{X}(t) \mid t \in [t_i, t_f]\} \tag{2.31}$$

$$\Phi_T = \{\Phi(t_f, t_i) \mid t \in [t_i, t_f]\} \tag{2.32}$$

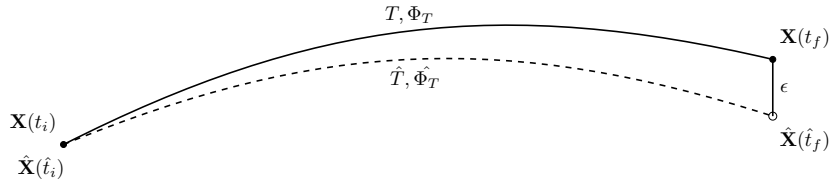


Figure 2.8: SSDC representation between actual (T) and desired trajectory ($\hat{T}$)

The update is typically applied iteratively until the terminal error $|\epsilon|$ falls below a specified threshold, which constitutes the complete iterative cycle of an SSDC scheme. Practical implementations often limit the number of correction iterations to avoid numerical instabilities, whilst ensuring the represented trajectory does not diverge too much from the desired properties.

Compared to SSDC, MSDC extends the implementation through constructing a large differential correction step for all nodes $\mathbf{X_n}$ and discontinuities (ϵ_n) in the trajectory such that:

$$\begin{bmatrix} -I & \Phi_1 & 0 & \cdots & 0 \\ 0 & -I & \Phi_2 & \cdots & 0 \\ \vdots & & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & -I & \Phi_{N-1} \end{bmatrix} \begin{bmatrix} \delta\mathbf{X}_1 \\ \delta\mathbf{X}_2 \\ \vdots \\ \delta\mathbf{X}_N \end{bmatrix} = - \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_{N-1} \end{bmatrix}. \tag{2.33}$$

This allows the adjustment per node to be calculated at once, improving stability of the correction scheme. In order to effectively implement MSDC algorithms requires propagating $\Phi(t, t_0)$ (STM) alongside the integration, as described in Section 2.6. This is then used to iteratively estimate the correction across all states needed in order to converge the trajectory within the terminal error. This is demonstrated by Figure 2.9, where the mismatch (defects, grey vertical line) between initial states of one segment (q_0^{i+1}), and final states of the previous segment (q_f^i) are iteratively corrected from the initial guess (forward integration, dashed) to the final true physically valid solution (solid).

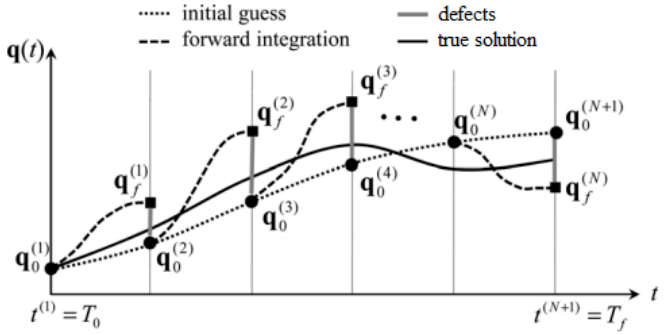


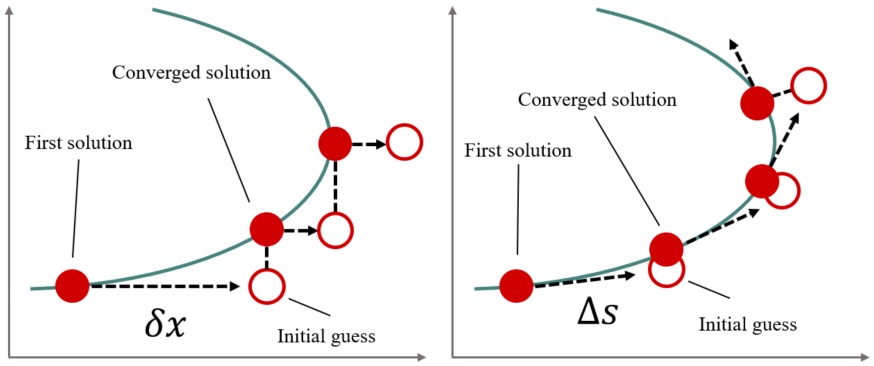
Figure 2.9: Multiple shooting segmentation and convergence of a trajectory based on differential correction per segment [32].

The amount of segments into which the original trajectory is divided is typically dependent on the situation being modelled. Periodic orbits typically require a higher amount of segmentation than transfer trajectories, as keeping the desired physical shape of the orbit is of greater relevance in these scenarios then for transfer trajectories, see Section 2.12. MSDC is used in a variety of contexts, such as targeting, trajectory optimization, or enforcing boundary conditions in two-point boundary value problems.

Two time variations of MSDC exist, fixed time correction and variable time correction. In the fixed-time formulation, the time interval between nodes is kept constant, and only the state vectors at each node are adjusted to satisfy continuity and boundary conditions. This approach is computationally simpler due to having less free variables, and is often used for this reason. Conversely, the variable-time formulation allows the time intervals between nodes to vary, introducing the segment durations as additional correction parameters. This added flexibility enables higher convergence rates due to relaxing model constraints, at the added cost of more computational effort [35]. Additionally, an extension of this method exists in ephemeris-based high fidelity models (Section 2.12), where variable-epoch correction is employed to account for epoch dependence of external perturbations.

2.11. Familial Continuation

Familial continuation methods are used to compute additional periodic orbits once a single member of a family has been identified. These methods involve gradually perturbing parameters such as the Jacobi (C), Period (P), and initial state ($\mathbf{X}_0$), and applying differential correction schemes to ensure periodicity with each change of initial parameters. While effective, this approach is inherently local, as taking large steps along the family often leads to poor initial guesses, preventing the differential correction scheme from converging. Consequently, familial continuation is typically performed using small incremental steps to preserve convergence stability and accurately trace the orbit family. An overview of the most commonly used familial continuation schemes are presented below in Section 2.11.1 and Section 2.11.2.



(a) Natural parameter continuation (b) Pseudo-arclength continuation

Figure 2.10: Diagram Comparing Natural Parameter Variation (NPV) and Pseudo-Arc Length Continuation [36]

2.11.1. Natural Parameter Variation (NPV) Continuation

Natural Parameter Variation (NPV) methods consist of perturbing the solution within a desired family by a single parameter. There are numerous options for this variation, such as a state in $\mathbf{X}_0$, Jacobi (C), or Period (P). This incremental change to the initial state is then propagated in time, prior to using a differential correction strategy to enforce periodicity, and aiming to converge $\mathbf{X}_P = \mathbf{X}_0$. This strategy is applied iteratively until the solution converges for a new periodic orbit with a stepped parameter. This strategy is then applied iteratively to generate new solutions, as is visible in Figure 2.10a, where the initial guess for a new solution is generated by the scheme prior to correction to a converged solution [36]. It is thus required to propagate the STM in order to apply the differential correction mechanism. The pseudocode for this method is presented in Algorithm 1:

Algorithm 1 Natural Parameter Variation (NPV) Method

Input: Initial periodic solution $\mathbf{X}_0^0$, perturbation step δx, maximum iterations N, tolerance τ
Choose parameter for variation (e.g., state in $\mathbf{X}_0$, Jacobi constant C, or period P)
for $k = 1$ to N **do**
    Apply perturbation: $\mathbf{X}_0^{(k)} = \mathbf{X}_0^{(k-1)} + \delta x$
    while $\|\mathbf{X}_P^{(k)} - \mathbf{X}_0^{(k)}\| > \tau$ **do** ▷ SSDC
        $\mathbf{X}_0^{(k)}$: $\delta \mathbf{X}_0^k = -\Phi(t_P, t_0)^{-1} \cdot \epsilon$ ▷ Update initial state
        Propagate $\mathbf{X}_0^{(k)}$, STM $\Phi(t_P, t_0)$ using E.O.M.
    end while
    Save converged periodic orbit for $\mathbf{X}_0^{(k)}$
end for
Output: Family of periodic orbits $\{\mathbf{X}_0^{(k)}\}$

2.11.2. Pseudo-Arc Length (PAL) Continuation

Pseudo-Arc Length (PAL) Continuation acts as an alternative to NPV, where instead of instead of perturbing a single parameter, all free-variables are updated simultaneously [34]. The update step, as denoted by Δs in Figure 2.10a is defined tangent to previous updates within the family by leveraging the Jacobian of updates to the family [36, 34]. This results in a more robust continuation method, often requiring less iterations to converge. An equivalent formulation updates the parameters using the difference between the two preceding solutions while adapting the step size Δs according to the local gradient [36]. The approach can be extended to higher-order predictor steps for increased accuracy, with the second-order implementation presented in Algorithm 2:

Algorithm 2 Two-Step Pseudo-Arc Length (PAL) Continuation

Input: Two initial family states $\mathbf{X}_0^0$, $\mathbf{X}_0^1$; step size Δs, maximum iterations N, tolerance τ
for $k = 2$ to N **do**
  Compute estimated tangent direction: $\mathbf{T}^{(k)} = \mathbf{X}_{k-1} - \mathbf{X}_{k-2}$
  Predict next solution $\mathbf{X}_0^{(k)} = \mathbf{X}_{k-1} + \Delta s \cdot \frac{\mathbf{T}^{(k)}}{\|\mathbf{T}^{(k)}\|}$
  while $\|\mathbf{X}_P^{(k)} - \mathbf{X}_0^{(k)}\| > \tau$ **do** ▷ SSDC
    $\delta\mathbf{X}_0^{(k)} = -\Phi(t_P, t_0)^{-1} \cdot \epsilon$ ▷ Update initial state
    Propagate $\mathbf{X}_0^{(k)}$, STM $\Phi(t_P, t_0)$ using E.O.M.
  end while
  Save converged periodic orbit for $\mathbf{X}_0^{(k)}$
end for
Output: Family of periodic orbits $\{\mathbf{X}_0^{(k)}\}$

2.11.3. Interpolation

Since NPV and PAL are extrapolation-based continuation methods, it is also possible to estimate parameters through interpolation. Interpolation is particularly useful when a dense family of periodic orbits exists, improving the initial estimate for the periodic orbit and reducing the amount of iterations required within the SSDC scheme. The order of interpolation (i.e. linear, cubic quadratic) used is dependent on both existing familial information and family shape. Interpolation has the added benefit of providing an estimate that has a much higher rate of convergence in differential correction schemes compared to familial continuation methods, reducing the amount of SSDC update iterations in Algorithm 3, and in turn computational effort:

Algorithm 3 Interpolation-Based Continuation

Input: Initial states of two converged periodic solutions $\mathbf{X}_0^{(k-1)}$, $\mathbf{X}_0^{(k+1)}$; interpolation fraction $0 \leq \alpha \leq 1$, maximum iterations N, tolerance τ
Compute initial guess via interpolation: $\mathbf{X}_0^k = (1-\alpha) \cdot \mathbf{X}_0^{(k-1)} + \alpha \cdot \mathbf{X}_0^{(k+1)}$
while $\|\mathbf{X}_P^{(k+\alpha)} - \mathbf{X}_0^{(k+\alpha)}\| > \tau$ **do** ▷ SSDC
  $\delta\mathbf{X}_0^{(k+\alpha)} = -\Phi(t_P, t_0)^{-1} \cdot \epsilon$ ▷ Update initial state
  Propagate $\mathbf{X}_0^{(k+\alpha)}$, STM $\Phi(t_P, t_0)$ using E.O.M.
end while
Save converged periodic orbit for $\mathbf{X}_0^{(k+\alpha)}$
Output: Interpolated periodic orbit $\mathbf{X}_0^{(k)}$

2.12. Transfer to Ephemeris-based High Fidelity (EHF) Model

Periodic orbits and transfer trajectories within the CR3BP are typically used as a preliminary approximation, prior to transition into a high-fidelity model. It is critical for accurate mission design at later stages to include these perturbations, such as solar radiation pressure, and gravitational effects from additional celestial bodies (e.g. ephemeris-based motion). Transfer is typically achieved by applying MSDC over a number of periods $n > 2$, where the trajectory is segmented into numerous arcs and continuity constraints are enforced between arc endpoints. By propagating each arc under the higher-fidelity dynamics and iteratively correcting the mismatches at segment boundaries, the method refines the initial CR3BP-based solution into one that is dynamically consistent within the more complete model. Figure 2.11 demonstrates this, where a CR3BP approximation of a L_2 Halo orbit (left) is differentially corrected to the EHF model (middle), prior to having extreme points pruned (right), resulting in a Quasi-Periodic orbit, the EHF analogue of a Periodic Orbit. Performed over a set amount of revolutions, such correction are able to obtain such a Quasi-Periodic orbit in the EHF model, however it is typical of these Quasi-Periodic orbit to become unbounded in the future due to the chaotic dynamics. Quasi-periodic orbits typically remained bounded by a torus in the vicinity of the original CR3BP periodic orbit [27, 32].

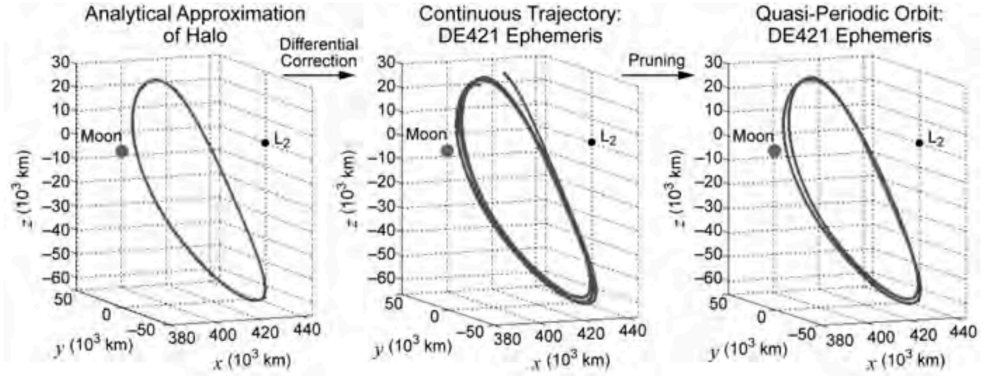


Figure 2.11: Transition process of a CR3BP Halo orbit to an Ephemeris model. First differential correction for a number of orbital revolutions is applied, prior to pruning of edge cases [32]

2.13. Bifurcation Theory

Bifurcation theory explores how the behaviour and structure of both a trajectory, and family of trajectories in a dynamical system governed by differential equations changes with the variation of initial conditions $\mathbf{X}$ and corresponding system parameters (e.g. C, P). It focuses on identifying critical parameter values at which the system undergoes a fundamental change in stability or in the number and type of equilibrium solutions and periodic orbits. In the context of the CR3BP dynamical system, bifurcation theory enables the identification of bifurcation points where new families of periodic solutions emerge, exhibiting behaviours distinct from those of the original family [34].

Bifurcation theory is reliant on analysis of the eigenvalues of the monodromy matrix (M), and uses this to both determine when bifurcations can occur, and classify the type of bifurcation that occurs within a family. A variation in the number of real, complex, or unit-magnitude eigenvalues indicates the occurrence of a bifurcation [32], as explored in Section 2.13.1.

2.13.1. CR3BP Bifurcation Types

Within CR3BP dynamics the following bifurcations types are identified, and are limited to those governed by the eigenvalues of the monodromy matrix [32, 34]. It should be noted that bifurcations that are not governed by eigenvalues do exist, and will be elaborated upon later in this investigation.

I. **Tangent Bifurcation:** Occurs when two nontrivial eigenvalues of the monodromy matrix cross unity, marking a shift in the order of instability (amount of eigenvectors inside (blue)/outside (red) the unit circle (green) in the imaginary plane in Figure 2.4) along the family:

$$\lambda_j = \frac{1}{\lambda_j} = +1, \tag{2.34}$$

   (a) **Cyclic Fold Bifurcation:** The orbits within a single periodic family change order of instability but do not intersect any other family. A cyclic fold occurs at an extremum of the Jacobi constant value.

   (b) **Pitchfork Bifurcation:** A change in stability along a family produces two new families, both possessing the same stability as the original family prior to the bifurcation.

   (c) **Transcritical Bifurcation:** A stable and an unstable family intersect, and at the intersection, the stability characteristics of the families are exchanged.

II. **Period-Multiplying Bifurcation:** Occurs when two nontrivial eigenvalues evolve as the $(m-1)^{th}$ complex roots of unity, with a multiplying factor $m > 2$:

$$\lambda_j, \frac{1}{\lambda_j} = \cos\left(\frac{2\pi}{m}\right) \pm i \sin\left(\frac{2\pi}{m}\right), \tag{2.35}$$

For example when m = 3:

$$\lambda_j, \frac{1}{\lambda_j} = \cos\left(\frac{2\pi}{3}\right) \pm \sin\left(\frac{2\pi}{3}\right) \tag{2.36}$$

$$\lambda_j, \frac{1}{\lambda_j} = -0.5 \pm 0.8660i \tag{2.37}$$

Period-multiplying bifurcations do not require eigenvalue intersection with the unit circle and may not lead to changes in orbital stability. The exception is the period-doubling bifurcation, as when $m = 2$, two nontrivial eigenvalues of the monodromy matrix intersect and depart from the unit circle along the negative real axis:

$$\lambda_j, \frac{1}{\lambda_j} = \cos\left(\frac{2\pi}{2}\right) \pm \sin\left(\frac{2\pi}{2}\right) = -1 \tag{2.38}$$

It should be noted that in order to apply PAL continuation schemes, the bifurcating orbit from the original minimal-period orbit must be integrated for $m \cdot P$ periods of the original bifurcating orbit [34].

III. **(Modified) Secondary Hopf Bifurcations:** Occurs when two eigenvalues intersect the unit circle and depart into the complex plane, but not at ± 1 on the real axis, resulting in a change of stability. A modified secondary Hopf bifurcation on the other hand occurs when two eigenvalues intersect on the real line and move into the complex plane (other than at ± 1), but without a stability changes. In some cases, secondary Hopf bifurcations lead to new periodic solutions, associated with an extra (vertical) dimension, but generally, they produce invariant tori surrounding the periodic orbit.

A bifurcation diagram is commonly used to graphically represent a bifurcation by plotting two varying non-zero parameters of an orbit family, as shown in Figure 2.12. In this example, the Jacobi constant is plotted against a non-zero coordinate at a symmetry plane (either x, z, or v_y), highlighting two possible bifurcation points as the transition (triangle) between orders of instability (the number of unstable eigenvectors pairs , teal: 0, blue: 1, magenta: 2) within a family, one of which leads to a new family corresponding to the dashed line [34].

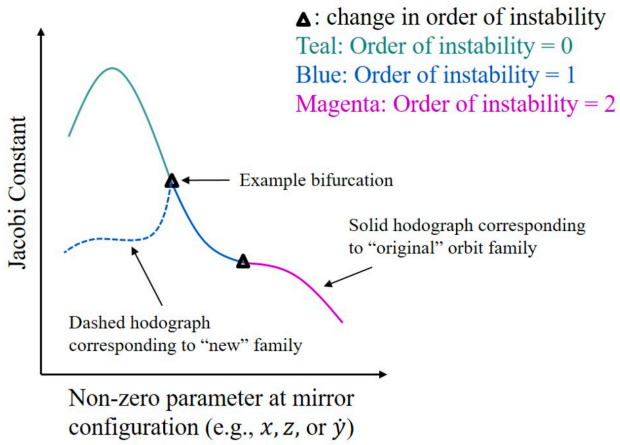


Figure 2.12: Sample Bifurcation Diagram [34]

Bifurcation diagrams that represents large amounts of families cannot always be created due to overlap in curves and parameter values between families due to the reduction to two parameters. In these scenarios, diagrams that solely present the branch points between families are used. Figure 2.13 presents such a diagram for the Earth-Moon system, where red cubes mark the libration points, small white spheres indicate bifurcation points, and red spheres mark termination points of the family (collision of the orbit with one of the primaries, specific to this diagram) [33]. It should be noted that this representation is not complete, and should not be treated as such, as the diagram is limited to families arising solely from tangent bifurcations. The exact bifurcating orbit within a family is often not determined with complete precision, but rather identified within an acceptable tolerance range [34, 36].

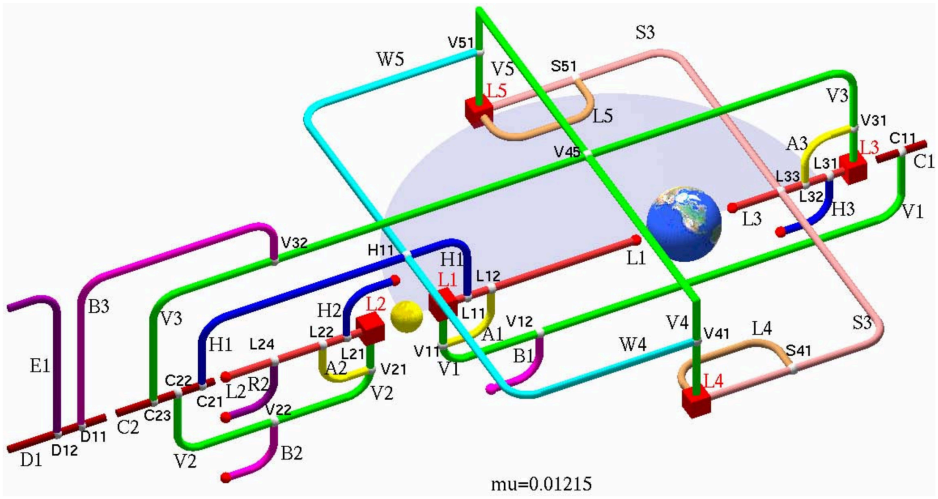
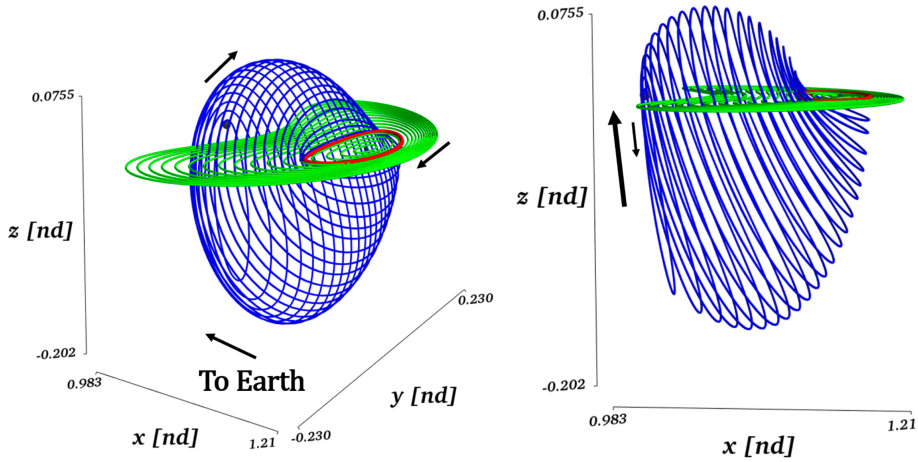


Figure 2.13: Bifurcation Diagram of the Earth Moon System, nomenclature is not adopted [33]

2.13.1.1. Bifurcations of the L_2 Families

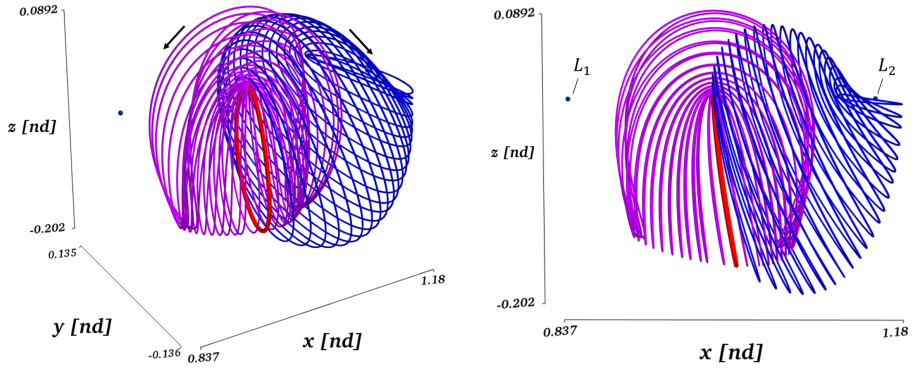
An example of a bifurcation between families occurs in the L_2 Halo family, which originates from a tangent pitchfork bifurcation along the L_2 Lyapunov family. The L_2 Lyapunov family originates itself as a perturbation based bifurcation from the L_2 libration point, as described in Section 2.4. At the bifurcation point (red orbit) between the $L_{2,S}$ Halo (blue) and L_2 Lyapunov (green), presented in Figure 2.14, both the L_2 Lyapunov and L_2 Halo orbits are planar, however at the bifurcating orbit the Halo family subsequently extends out of the plane in the $\pm$ z directions, corresponding to the North and South L_2 Halo families respectively. The bifurcating orbit (red orbit) in Figure 2.14 corresponds to the red branch point between L_2 and H_2 familes, (L_{21}) in Figure 2.13, denoting the L_2 Lyapunov and L_2 Halo orbit families respectively.



(a) (b) View along $+y$ axis.

Figure 2.14: L_2 Lyapunov (green) and $L_{2,S}$ Halo (blue) families. The red orbit denotes the bifurcating orbit [36]

Similarly, another example is presented in Figure 2.15 for the L_2 Butterfly family (purple), which bifurcates with a period-doubling bifurcation from a six-day L_2 Halo orbit (red), forming its own distinct family with unique dynamical characteristics when compared to the original L_2 Halo orbit. This bifurcation is not presented in Figure 2.13, as the figure is restricted to not include any period-multiplying bifurcations, however would occur along the H_2 branch near its termination point (red).



(a) (b) View along $+y$ axis.

Figure 2.15: $L_{2,S}$ Halo (blue) and $L_{2,S}$ Butterfly (purple) families. The red orbit denotes the bifurcating orbit [36].

2.13.2. Broucke's Stability Diagram

A Broucke's stability diagram serves as a powerful tool for analyzing where bifurcations arise within a family, in order to identifying bifurcation points after a family has been fully continued. In this approach, the nontrivial eigenvalues of the monodromy matrix for each member of the family are expressed in terms of two parameters, α and β:

$$\alpha = -\left(\lambda_1 + \frac{1}{\lambda_1} + \lambda_2 + \frac{1}{\lambda_2}\right), \qquad \beta = \frac{1}{2}\left(\alpha^2 - \left(\lambda_1^2 + \frac{1}{\lambda_1^2} + \lambda_2^2 + \frac{1}{\lambda_2^2}\right)\right) \tag{2.39}$$

This reduces the amount of parameters to be monitored from four to two, allowing them to be plotted as Figure 2.16 [34]. Each boundary (grey line) between regions is defined by critical eigenvalue conditions. Note the orange region is the only stable region in the figure as all nontrivial eigenvalues are within the stable region of the unit circle ($|\lambda| < 1$). The small unit circle within each region depicts the typical configuration of the four nontrivial eigenvalues for that region.

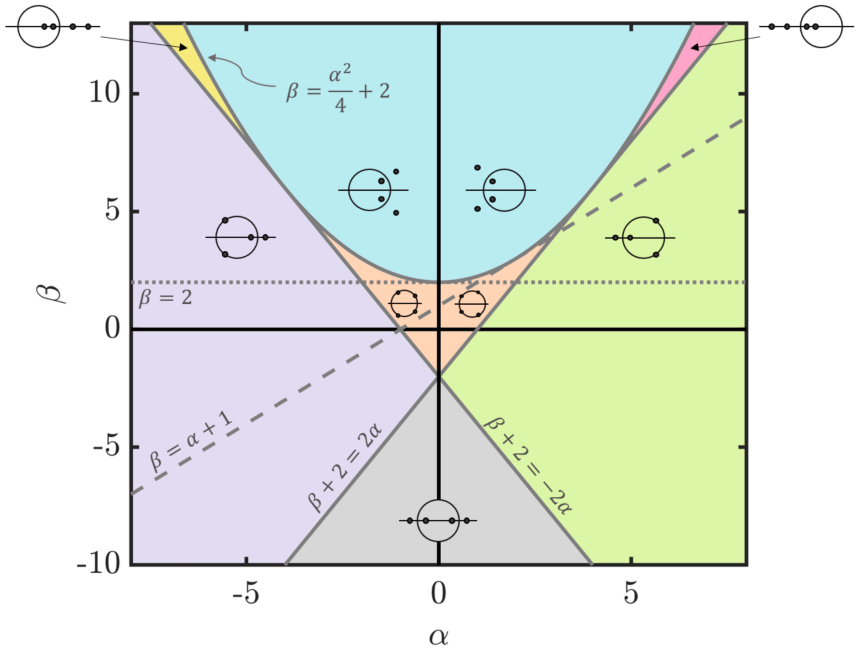


Figure 2.16: Broucke Stability Diagram [34]

Table 2.4: Bifurcation types, corresponding lines, and color transitions [34]

Bifurcation Type	Eqn. of Line Crossed	Color Transitions
Tangent	$\beta + 2 = -2\alpha$	Orange $\leftrightarrow$ Purple Green $\leftrightarrow$ Grey Yellow $\leftrightarrow$ Purple
Period-Doubling	$\beta + 2 = 2\alpha$	Orange $\leftrightarrow$ Green Purple $\leftrightarrow$ Grey Pink $\leftrightarrow$ Green
Secondary Hopf	$\beta = \frac{\alpha^2}{4} + 2$ $\alpha \in (-4, 4)$	Orange $\leftrightarrow$ Blue
Modified Secondary Hopf	$\beta = \frac{\alpha^2}{4} + 2$ $\alpha \in (-\infty, -4] \cup [4, \infty)$	Pink $\leftrightarrow$ Blue Yellow $\leftrightarrow$ Blue

A bifurcation can occur when the curve of the family crosses one of these boundaries when a family of orbits is plotted on the diagram, as demonstrated for the L_2 Halo family in Figure 2.17. The occurrence of the bifurcation is dependent on the conditions presented in Section 2.13.1, whilst the specific type of bifurcation is determined based on the boundary that has been crossed [32, 34]. Table 2.4 denotes which region crossing corresponds to the types of bifurcations identified in Section 2.13.1. The intersections (stars, colored) in Figure 2.17 correspond to the location of the bifurcating orbits, whilst the arrow corresponds to an increase of the perilune radius [34]. The two period doubling bifurcations (orange, red) correspond to the L_2 Butterfly and Dragonfly families, explored later in Section 6.1.1.

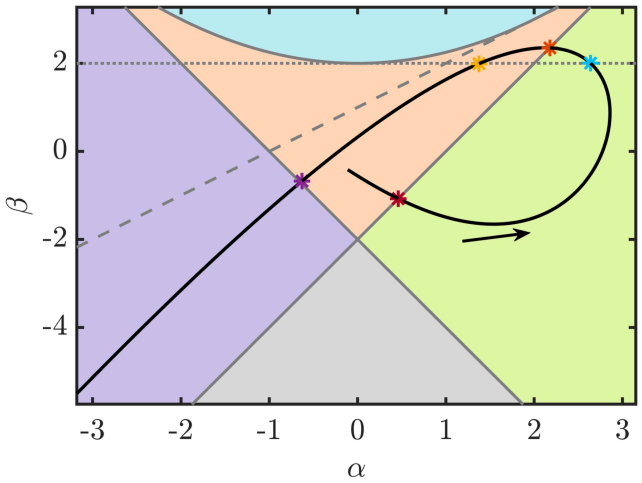


Figure 2.17: L_2 Halo (NRHO) Broucke stability diagram [34]

3

Mission Design Pipeline

Designing trajectories within the CR3BP typically follows a structured sequence of stages. The process begins with the identification and generation of periodic orbits that possess desirable dynamical properties such as favourable stability characteristics, appropriate Jacobi constants, suitable orbital periods, or advantageous spatial locations over time. Identification of suitable periodic orbits is followed by analysis of approach and departure orbits by means of invariant manifolds to determine costs (Δv, TOF) associated with approaching or departing these orbits, prior to transfer to EHF to assess full feasibility.

3.1. Numerical Representation of Periodic Orbits and their Manifolds

To formalize the mission design pipeline, set notation is introduced. Let $\mathcal{P}$ denote the set of all parent periodic orbits considered with desirable characteristics. Each element $P_i \in \mathcal{P}$ represents a single periodic orbit obtained through familial continuation or interpolation within a family. The first stage in mission design requires familial continuation of periodic orbit families or interpolating within those families to obtain a sufficiently dense set of candidate periodic orbits with desirable mission characteristics (e.g. stability, location). Once a suitable set of periodic orbits has been selected, the next step is computing transfer trajectories that enable a spacecraft to approach or depart that orbit. As stated in Section 2.9, this is achieved by computing the set of invariant manifolds for each periodic orbit. For each periodic orbit P_i, the associated set of invariant manifolds is defined as:

$$\mathcal{W}(\mathcal{P}_i) = \{\mathcal{W}^s(\mathcal{P}_i),\ \mathcal{W}^u(\mathcal{P}_i)\} \tag{3.1}$$

Where $\mathcal{W}^s(\mathcal{P}_i)$ and $\mathcal{W}^u(\mathcal{P}_i)$ denote the stable and unstable manifolds of $\mathcal{P}_i$ respectively. By joining the sets of manifolds associated with all periodic orbits in $\mathcal{P}$, the global set $\mathcal{W}$ is obtained representing the set of all invariant manifolds relevant to the mission design process:

$$\mathcal{W} = \bigcup_{\mathcal{P}_i \in \mathcal{P}} \mathcal{W}(\mathcal{P}_i) \tag{3.2}$$

As each family is continuous, the amount of selected periodic orbits ($P = |\mathcal{P}|$) for analysis within the set $\mathcal{P}$ is arbitrarily selected depending on the desired resolution of the analysis. Similarly, each invariant manifold is discretized into a finite number of manifold arcs ($A = |\mathcal{W}^i(\mathcal{P}_i)|$ for analytical purposes at a predetermined resolution. Each manifold arc is then further sampled over N time steps, such that a time-discretized representation of the flow is achieved. Consequently, the fully discretized size of the set of invariant manifolds for a set of periodic orbits is of size:

$$|\mathcal{W}| = P \cdot A \cdot N \tag{3.3}$$

3.2. Discretization and Computational Process

To understand the numerical process and computational burden behind obtaining the set $\mathcal{W}$, the pipeline in Figure 3.1 is introduced. The process is initialized with the population of the abacus to assess periodic orbit suitability, prior to population of the set of manifold arcs to assess transfer orbit suitability.

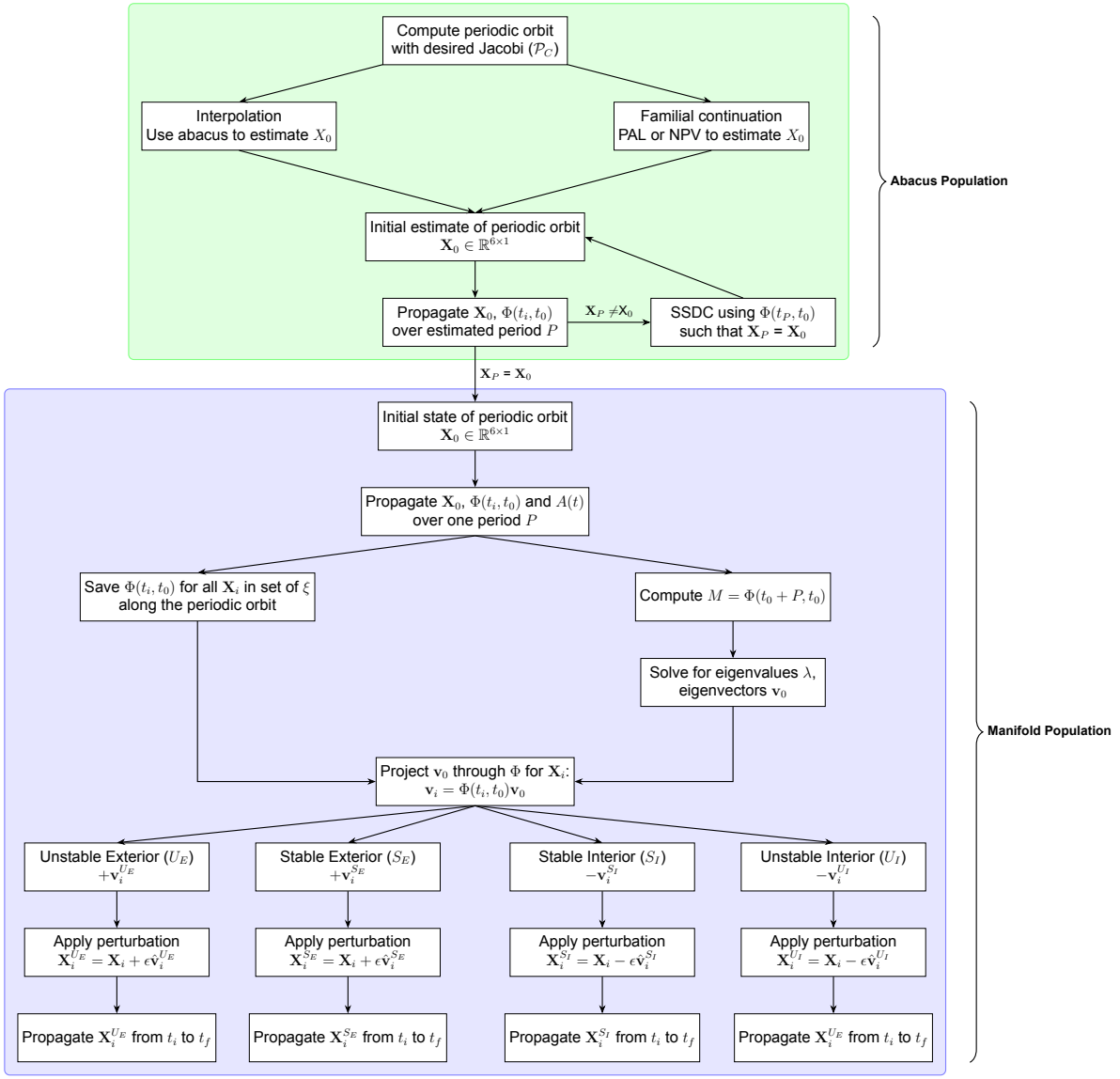


Figure 3.1: Pipeline for Computing Periodic Orbits and Manifold Arc's

Figure 3.1 provides a high-level overview of the computational workflow, separating the periodic-orbit abacus population stage (green) from the subsequent manifold-arc generation stage (blue). For clarity and reproducibility, the corresponding implementation-level procedure is summarized in Algorithm 4 as pseudo-code. These two stages are then described in detail in Section 3.2.1 and Section 3.2.2 respectively.

Algorithm 4 Periodic Orbit Abacus Population and Manifold Arc Generation

Part 1: Abacus Population (Periodic Orbit Computation)

if abacus is sufficiently populated near Jacobi value C **then**
  $\mathbf{X}_0 \leftarrow$ InterpolateAbacus(C) ▷ Algorithm 3
else
  $\mathbf{X}_0 \leftarrow$ ContinueFamily(C) ▷ PAL (Algorithm 2) or NPV (Algorithm 1)
end if
while $\|\mathbf{X}_P - \mathbf{X}_0\| > \tau$ **do** ▷ SSDC
  Propagate $\mathbf{X}_0$ over estimated period P to obtain $(\mathbf{X}_P, \Phi(t_P, t_0))$
  $\delta\mathbf{X}_0 \leftarrow -\Phi(t_P, t_0)^{-1}(\mathbf{X}_P - \mathbf{X}_0)$
  $\mathbf{X}_0 \leftarrow \mathbf{X}_0 + \delta\mathbf{X}_0$
end while
Save converged periodic orbit $\mathbf{X}_0$ to abacus

Part 2: Manifold Population (Eigenstructure, Perturbation, Propagation)

Propagate periodic orbit over period P to obtain states $\{\mathbf{X}_i\}, \Phi(t_i, t_0), \forall i \in \xi$
Compute monodromy matrix $M = \Phi(t_0 + P, t_0)$
Compute eigenstructure $(\lambda, \mathbf{v}_0)$ of M
Initialize manifold set $\mathcal{A} \leftarrow \emptyset$
for each manifold direction $d \in \{U_E, U_I, S_E, S_I\}$ **do**
  for each X_i with spacing $\Delta\xi$ **do**
    $\mathbf{v}_i \leftarrow \Phi(t_i, t_0)\mathbf{v}_0$ ▷ Projected eigenvector
    $\mathbf{X}_i^d \leftarrow \mathbf{X}_i + \epsilon\, s(d)\, \hat{\mathbf{v}}_i^d$ ▷ $s(d) \in \{+1, -1\}$ encodes exterior/interior
    $\mathrm{arc} \leftarrow$ PropagateArc$(\mathbf{X}_i^d, \mathrm{TOF})$
    $\mathcal{A} \leftarrow \mathcal{A} \cup \{\mathrm{arc}\}$
  end for
end for
return $\mathcal{A}$

3.2.1. Population of Periodic Orbit Abacus

The set of computed $\mathcal{P}$, adopting SEMpy nomenclature is referred to as an abacus [35]. For practical implementations in this work, the resolution of this familial abacus is chosen according to the desired spacing in Jacobi constant. For example, if a resolution of $\Delta C = 0.1$ is selected, then periodic orbits are sampled at Jacobi levels separated by increments of 0.1. For each desired value of C, a corresponding periodic orbit is computed, through either familial continuation or interpolation using SSDC and added to the set $\mathcal{P}$. In this way, the density of periodic orbits in the abacus is directly controlled by the chosen ΔC, allowing the mission designer to trade off between computational effort and the fidelity of the resulting orbit and manifold representation. It is important to emphasize that ΔC is defined as the spacing between consecutive discretely sampled orbits within the continuous orbital family, not the global distribution of Jacobi constants within the family, since multiple distinct periodic orbits may share the same Jacobi value.

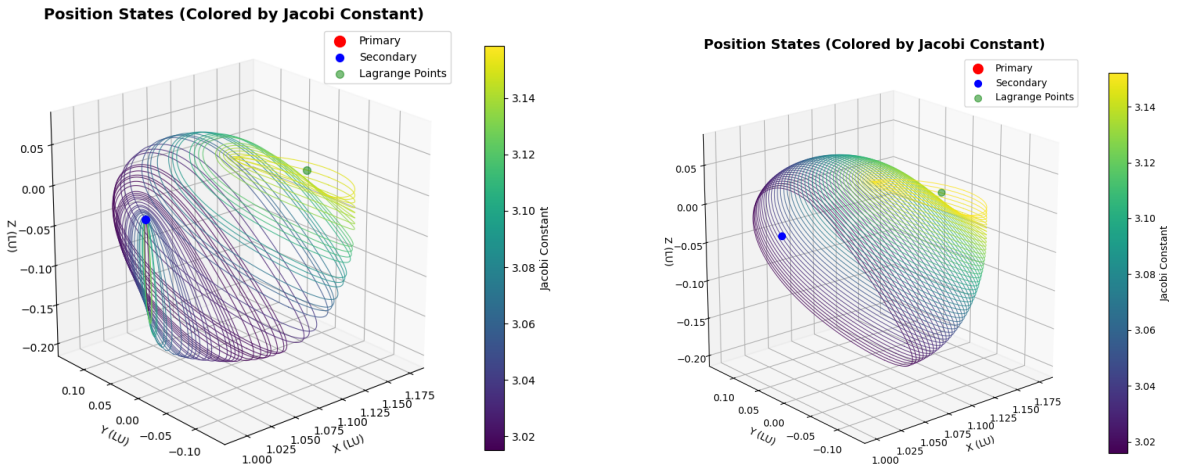
Given a chosen resolution ΔC, the total number of periodic orbits in the set $\mathcal{P}$ follows directly from the range of Jacobi constants of interest. If the family spans a Jacobi interval $[C_{\min}, C_{\max}]$, and orbits are sampled at uniform increments of ΔC, then the number of selected periodic orbits is:

$$P = |\mathcal{P}| \;=\; \left\lfloor \frac{C_{\max} - C_{\min}}{\Delta C} \right\rfloor + 1. \tag{3.4}$$

P is thus determined by the chosen resolution and the width of the Jacobi domain under consideration. A smaller ΔC yields a denser abacus, and therefore more periodic orbits, whilst inversely a larger ΔC results in a coarser representation and reduced computational load. Figure 3.2 demonstrates the difference between two storage methods for $\mathcal{P}$. Figure 3.2a stores the $L_{2,S}$ Halo family at a lower resolution ($\Delta C \approx 0.01$) than Figure 3.2b, which stores a subset of $L_{2,S}$ at a higher resolution ($\Delta C \approx$

0.005). Note the impact of choosing ΔC as the variable for control storage resolution is that orbits are unequally spaced in phase space. This is demonstrated by the unequal orbit spacing in phase space for $\Delta C \approx 0.01$ in Figure 3.2a, where each orbit's position is coloured by its associated C. Setting ΔC to be constant however does have the benefit of adequately resolving regions of rapid state space change within a family, such as regions near primaries or Lagrange points.

Figure 3.2a further illustrates that within the $L_{2,S}$ Halo family, a single Jacobi constant corresponds to two distinct periodic orbits. Specifically, both short and long-period orbits, those located closer in phase space to the secondary body and to the Lagrange point, respectively, may share similar Jacobi values yet different phase-space characteristics. Extended to other families, this means that a single Jacobi can be associated with 2 (or more) different orbits within a family, constituting a caveat of applying clustering algorithms based on Jacobi and initial state, explored in Section 6.3. If an injective representation in C of a family is desired (i.e. a value of C corresponds to single periodic orbit), the family must be further subdivided, demonstrated for the subset presented in Figure 3.2b, where each value of C solely corresponds to a single orbit, limiting the phase space.



(a) Full $L_{2,S}$ Halo family stored at $\Delta C \approx 0.01$

(b) Injective $L_{2,S}$ Halo subset family stored at $\Delta C \approx 0.005$).

Figure 3.2: Comparison of two representations of the $L_{2,S}$ Halo family. Green point is the L_2 point, whilst the blue point is the Moon (Moon not to scale). Coloring is in accordance with Jacobi Constant (C)

Assuming that a collection of representative periodic orbits for the family has already been generated through bifurcation analysis and stored in an abacus, the periodic orbit associated with a specified Jacobi constant $\mathcal{P}_C$ is obtained using either familial continuation methods or interpolation techniques, as detailed in Section 2.11, depending on the resolution and completeness of the existing abacus. Both methods require multiple SSDC iterations in order find $\mathcal{P}_C$, with the exact iteration amount dependent on the quality of the initial guess of X_0. This orbit is then appended to the abacus, such that $\mathcal{P}_{\text{new}} = \mathcal{P}_{\text{old}} \cup \mathcal{P}_C$. This process is presented as the green levels of Figure 3.1 as *"Abacus Population"*, and is repeated until the abacus is populated at the desired resolution. The specific process is detailed in Algorithm 4 in pseudo-code form.

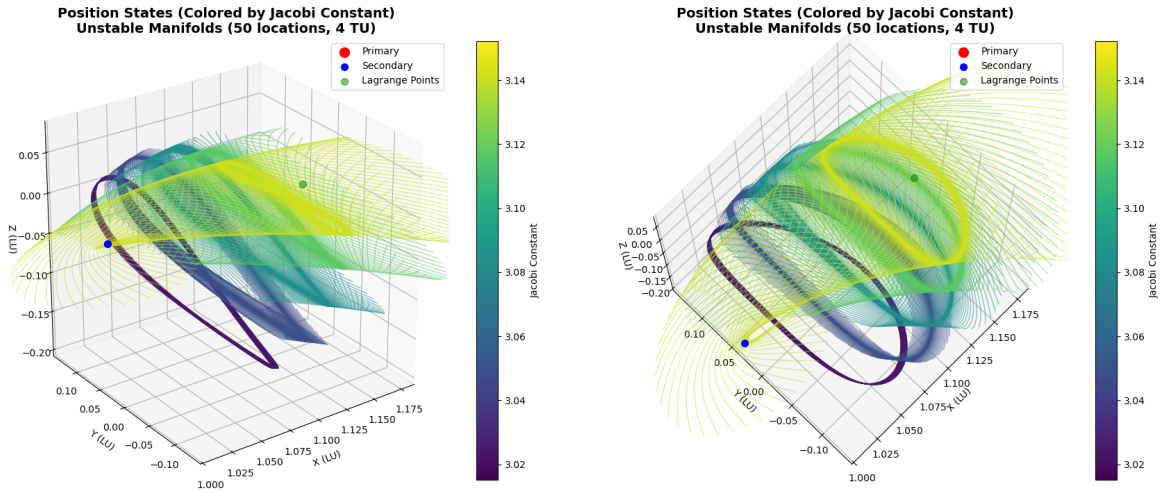
3.2.2. Population of Manifold Arc Set

In order to populate the set of manifold arcs, each orbit is propagated over its period (P) and using the STM (Φ), the associated eigenvectors of the monodromy matrix (M) are calculated. If knowledge about the specific phase ($\xi_i \in [0, 1]$) along the orbit for which the points of the manifold arc's propagation will occur (X_i) is available, the STM ($\Phi(t_i, t_0)$) relating this point to the initial coordinate X_0 of the orbit is saved. Similarly to the abacus, the resolution of these points $\Delta\xi$ is chosen depending on computational budget, and is inversely proportional to the amount of manifold arcs $A \propto \frac{1}{\Delta\xi}$:

$$A = |\mathcal{A}| = \frac{1}{\Delta\xi} \tag{3.5}$$

The perturbation corresponding to the desired manifold direction is then applied and the manifold arc is propagated for its desired TOF. This can be done with or without the STM, depending on whether or not SSDC needs to performed on the arc in question. This population process is presented as the blue levels of the pipeline in Figure 3.1 as *"Manifold Population"*.

An example of a populated manifold $\mathcal{A}$ in Figure 3.3 is presented in positional phase space for a 5 orbit subset of the$L_{2,S}$ Halo family, corresponding to an abacus stored at resolution $\Delta C \approx 0.025$. Each arc (line, coloured according to C of its parent orbit) is propagated for 4 [TU] with a manifold arc resolution of $\Delta\xi = 0.02$. The divergence rate of the manifold arcs varies across the family, reflecting differences in orbital stability. Darker blue trajectories (lower Jacobi constant) remain coherent for longer durations, whereas lighter yellow trajectories (higher Jacobi constant) diverge more rapidly. Consequently, an estimate of the stability characteristics of each orbit is required a priori to determine the appropriate propagation time needed for the corresponding manifold arcs, further increasing population difficulty.



(a) **(b)**

Figure 3.3: Manifold arc population across 5 $L_{2,S}$ Halo periodic orbits for 4 [TU] with a resolution of $\Delta\xi = 0.02$.

The tolerance τ is an important mission design requirement, as it controls the state mismatch between X_0 and X_P. This is especially important when propagating manifold arcs, as small errors in evaluation of $\Phi(t_0 + P, t_0)$ are compounded both by calculation of $X_i^{\mathcal{W}}$ and propagation of $X^{\mathcal{W}}$ in time to produce $\mathcal{W}(\mathcal{P})$. This compounding of errors is measured to produce accuracy discrepancies of $10^{-1}[-]$ for errors on the order of magnitude of $10^{5}[-]$ in $X_i^{\mathcal{W}}$ over a manifold arc propagation time of 25 [TU].

It is not common to fully populate these sets prior to optimization evaluation, as fully populating this set requires setting a resolution in advance, restricting the search space. Furthermore, storing the fully populated sets also becomes intractable for large amounts of orbits and manifold arcs [28]. Instead, ad-hoc computation during the optimization process is preferred, iteratively computing $\mathcal{P}_C$ and a manifold arc $\mathcal{A}$ for each iteration. This however reduces the amount of calculations that can be repurposed, as the set of Φ, saved for the start of each manifold arc, is not computed in advance, resulting in the iteration process inducing a higher additional cost for each propagation of Φ, $\mathcal{P}_C$, and $\mathcal{A}$.

3.3. Transfer to EHF

Once a periodic orbit and/or transfer trajectory is selected during the optimization process, it must be transitioned to the EHF model. As described in Section 2.12, this transition is performed using an MSDC correction scheme over a set number of revolutions. Successful transfer to the EHF is not guaranteed and depends strongly on the specific orbit or trajectory under consideration. In particular, highly unstable trajectories or periodic orbits may diverge rapidly when propagated in the EHF model, causing correction techniques such as multiple shooting to become ineffective or numerically unstable

beyond short propagation intervals [39, 40]. This behaviour is demonstrated in Table 3.1, where only the most stable families and the most stable members within those families, remain dynamically feasible after several revolutions.

Table 3.1: Percentage of Successful CR3BP - EHF Transitions by Family and Period Amount (Rev) [40]

Family	3 Rev.	5 Rev.	7 Rev.	9 Rev.
DRO	48.3%	40.3%	35.7%	26.2%
$L_{1,N}$ Halo	43.7%	34.8%	34.5%	32.3%
$L_{1,S}$ Halo	43.7%	34.8%	34.8%	31.7%
L_1 Lyapunov	34.8%	22.5%	18.5%	17.5%
$L_{2,N}$ Halo	49.5%	21.2%	17.5%	16.9%
$L_{2,S}$ Halo	49.5%	21.2%	17.5%	16.9%
L_2 Lyapunov	29.2%	16.9%	16.9%	16.6%
L_4 Axial	83.1%	68.3%	58.2%	50.5%
L_4 Vertical	100%	100%	100%	100%
L_5 Axial	100%	64.9%	55.7%	50.5%
L_5 Vertical	100%	100%	100%	100%

External perturbations in the cislunar system are dominated by the gravitational influence of the Sun, followed by radiation pressure and Jupiter, as demonstrated by the orders of magnitude of these accelerations presented in Table 3.2 [39, 40, 41]. Note that the radiation pressure in this case assumes an surface area between 10-20 $[m^2]$, using the cannonball model with a diffuse reflection parameter of 0.8, typical of a mid-size satellite.

Table 3.2: Order of magnitude of external perturbations in dimensional and non-dimensional units [39, 41]

External Perturbation	Order of Magnitude [m/s^2]	Order of Magnitude [-]
Sun (third-body gravity)	10^{-5}	10^{-2}
Radiation pressure (cannonball)	10^{-7}	10^{-5}
Jupiter (third-body gravity)	10^{-10}	10^{-8}

For the purpose of this investigation, when transitioning to the EHF model, only the Sun is included as a third-body perturbation, as it is the dominant external influence in the cislunar environment [39, 41]. This keeps the analysis spacecraft-invariant by avoiding assuming a specific spacecraft surface area. Spherical harmonic perturbations are disregarded because as influence on periodic orbits in the cislunar regime is typically negligible. This is due to the fact that periodic orbits typically lie far from the Earth, where higher-order gravity terms decay rapidly, and higher-order gravity terms from the Moon typically have a minimal effect compared to other perturbations [41].

4

State Of The Art Developments

Traditional numerical integration techniques, while highly developed, face limitations when used to propagate chaotic systems over long timescales, especially when accuracy is prioritized. Errors in energy conservation or angular momentum can accumulate, causing solutions to diverge rapidly from physical reality [42]. To counteract this, Section 4.1 explores high-precision approaches and their relevance to the study of chaotic systems. While these methods incur substantial computational overhead, they are critical tools for benchmarking, stability analysis, and distinguishing genuine physical chaos from numerical artifacts and chaos.

In recent years, advances in machine learning and artificial intelligence have opened new pathways for accelerating orbital computations. These techniques offer attractive alternatives as neural surrogates, or trained neural-network models that approximate the input–output behaviour of computationally expensive numerical procedures, to traditional numerical methods across several key CR3BP applications, with main focuses lying in acceleration of orbital integration, explored in Section 4.2, generation of new candidate periodic orbits, explored in Section 4.3, and classification of periodic orbits, explored in Section 4.4. Through integrating these AI techniques into the workflows for orbital integration, generation, and classification, it is thus possible to iterate faster and explore broader regions of parameter space.

4.1. Clean Simulation & Arbitrary Precision Integrators

Traditional numerical solvers for the 3BP tend to be very computationally expensive and require large amounts of processing time due to their chaotic nature. This results in simulations requiring either architecture that supports the required level of precision, resulting in tiny time-steps when orbits become chaotic, or ensemble methods to determine statistical trends across multiple solutions. Arbitrary precision N-body integrators, such as BRUTUS or Clean Numerical Simulation (CNS) Packages, have the ability to maintain a specified precision accurately throughout the simulation, albeit at significantly higher computational costs [3, 43]. An example of this is the BRUTUS integrator requiring 120 times as much computational resources and time compared to a conventional Hermite integrator at the time-step size boundary when the Hermite and BRUTUS solutions begin to diverge. These arbitrary precision integrators are, as a result, incredibly useful when an accurate ground truth is desired for either benchmarking, tuning, verification, or calibration. Thus, these types of integrators are critical when distinguishing numerical errors from physical chaos, evaluating the performance of the integration scheme, or precisely characterizing sensitivity to initial conditions.

Moreover, such high-precision solvers play a crucial role in validating the statistical assumptions underpinning ensemble simulations. While conventional integrators may yield diverging individual trajectories, studies using BRUTUS have shown that their statistical outputs remain accurate provided energy conservation is within acceptable bounds. These acceptable bounds have been determined to be simulations which violate energy conversation by more than 10%. In other words, if the total energy of the system decreases by more than 10% throughout the simulation, these simulations should be excluded

in order to not statistically alter the distribution or reliability of the ensemble [3]. Ensemble behaviour under these conditions has been demonstrated to be statistically meaningful, and can be used to determine output trends, such as escape velocity distributions, binding energies, system lifetimes, and outcome frequency. These arbitrary precision methods offer a definitive way to quantify when traditional simulations break down, and to determine whether observed trends are a result of the physical scenario and not the build up of numerical artifacts [3, 8].

4.2. Neural Surrogates to Orbital Integration

A key practical application of artificial intelligence in orbital propagation is the acceleration of computation. Faster state prediction enables scalable analysis, making propagation and evaluation tractable for large dataset sizes. Various machine learning techniques have been proposed in recent years to reduce this computational burden through the use of mechanisms such as Deep Neural Networks (DNN's), Physics-Informed Neural Networks (PINN's) and Hamiltonian Neural Networks (HNN's), in combination with alternative Hybrid Mechanisms, explored respectively in Section 4.2.1 - 4.2.4.

4.2.1. DNN Implementations

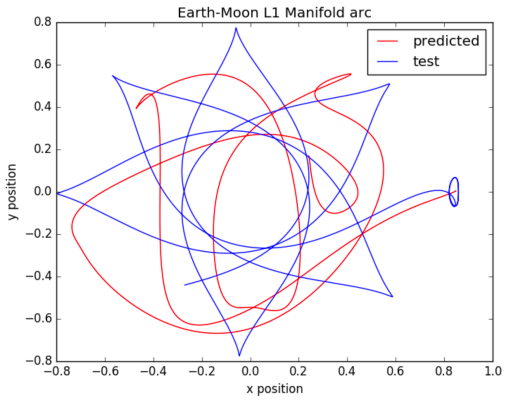
Early implementations of neural surrogates were centered around the direct applications of DNN's, which effectively demonstrated that DNN's both accurately and efficiently predict the positions of three gravitationally interacting bodies over short, bounded intervals, even in highly chaotic regimes such as those found within the CR3BP. This was demonstrated on a non-hierarchal setup with 3 equal masses with dimensionless units, with distances initialized within the dimensionless unit circle. As training data, 10^4 highly accurate BRUTUS simulations were used, comprising a 9900/100 train/test split, with each propagation being integrated over 10^5 [TU]. Such a setup was able to emulate the extreme sensitivity to initial conditions present within non-hierarchical simulations, with predicted trajectories closely matching that of the benchmark solutions in state space [14].

This DNN implementation nonetheless exhibits limitations that restrict its practical applicability, most notably its relatively poor preservation of energy compared to the benchmark solution, depspite proximity in state space. This energy preservation error is typically around 10^{-2} % , however during close interactions has the possibility to spike to 10^1 %, emphasizing the fact that such implementations have poor performance when modelling close interactions. This is overcome by including a projection layer in the DNN, which normalizes the loss of energy through optimizing weight terms for deviations from the initial x and y values, to decrease the relative energy error. This effectively reduces the relative energy error by two orders of magnitude to 10^{-5} %, largely attributed to the dampening of energy error spikes during close interactions.

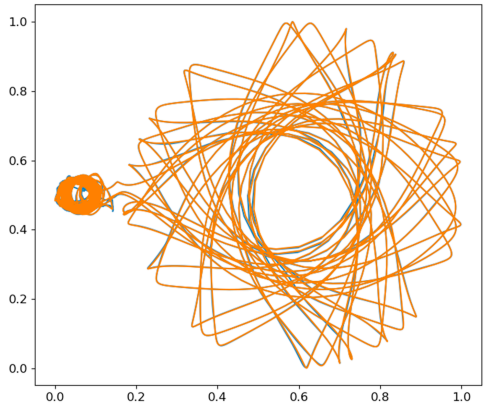
It should be noted that the DNN architecture demonstrated was relatively shallow, opting for 10 densely connected layers with 128 nodes per layer in combination with ReLu activations and optimization with the ADAM optimizer. This architecture was identified through systemically increasing layer and node count until the desired accuracy was reached [14]. This is indicative of tuning until performance metrics are met, possibly suggesting that there are significant performances increases that could be achieved through the inclusion of other mechanisms (such as the projection layer), with the potential for increased efficiency. Furthermore, the long-term stability of such DNNs has not been demonstrated, and the network is neither trained nor evaluated on known periodic orbits, which would provide a useful benchmark for quantifying approximation errors in CR3BP dynamics [14]. Furthermore, the DNN performs well only within the time window it was trained for, and it is highly likely that once extrapolated beyond, its prediction quality could deteriorate. This however is nonetheless suitable for surrogate models, so long the tested time range is equivalent to the regime in which it is trained.

Demonstrations of pure DNN within the CR3BP for both periodic orbits and manifold arcs have thus far been limited. Models with insufficient network capacity, such as architectures employing on the order of 10^2 neurons, have resulted in poor approximation accuracy, as illustrated by the discrepancy between predictions (red) and ground truth (test,blue) in state space in Figure 4.1a [28]. Conversely, high-capacity reservoir computing models employing on the order of $10^3 10^4$ neurons have demonstrated improved performance, achieving prediction accuracies on the order of 10^{-2} [LU] over short time horizons (≈ 1 [TU]). This behaviour is demonstrated by the close agreement between ground-truth state trajectories (blue) and model predictions (orange) shown in Figure 4.1b [44]. This reservoir computing

model had a highly specialized architecture based on the neural architecture of a fruit fly (*Drosophila*), allowing it to have a relative small neuron amount relative to its performance as a reservoir computing model [44]. Random Forest implementations have further proved to be more successful (measured by predictive accuracy), at the cost of more computationally expensive models [28].



(a) Direct DNN approximation of a manifold arc [28].



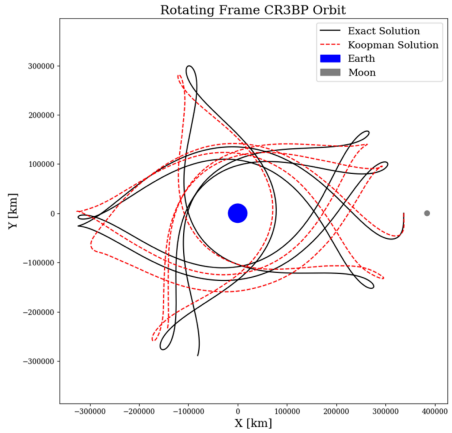
(b) *Drosophila*-inspired reservoir computing prediction of a CR3BP trajectory. Blue = prediction, orange = ground truth. [44]

Figure 4.1: Comparison of two DNN approaches: (a) direct DNN approximation and (b) Connectome-inspired reservoir approximation of an orbital trajectory.

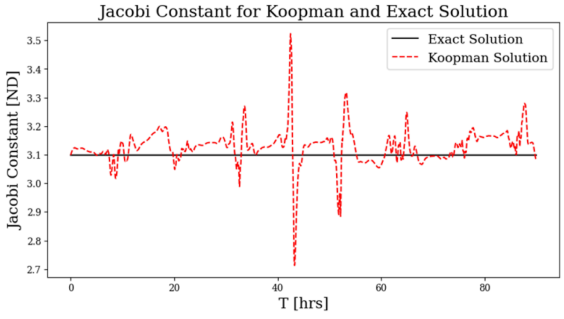
Alternative implementations leverage DNN's to discover Koopman operators that linearize the integration dynamics into Linear Time-Invariant (LTI) systems. The Koopman operator transforms a nonlinear dynamical system $x_{k+1} = F(x_k)$ into a linear system in an augmented observable space [45, 46]:

$$\Phi(x_{k+1}) = K\Phi(x_k) \tag{4.1}$$

Where $\Phi(\cdot)$ denotes a vector of learned observables and K is a linear operator that approximates the Koopman operator over this finite-dimensional space. It should be noted that the Koopman operator doesn't linearize the state space, instead the evolution of the observables are linearized [46]. This Koopman-based model (red, dashed) minimized energy errors more when compared to the standard DNN (black), with these energy errors remaining within a relative error of approximately 12.7% for the tested time-span of 20 [TU], presented over the full prediction horizon in Figure 4.2b. This is presented below in Figure 4.2a in turn presents the discrepancy between Koopman predictions (dashed,red) and the exact solution (solid, grey). This yields a substantial improvement in predictive accuracy over the pure DNN implementation (red, Figure 4.1a), as evidenced by the closer agreement of the Koopman-based state predictions with the benchmark solution (red dashed, Figure 4.2a), with further energy preservation implications later explored in Section 4.2.4.



(a) Predictions of Koopman-linearization DNN applied to periodic CR3BP dynamics [46].



(b) Relative Jacobi constant error over propagation using the Koopman-based DNN [46].

Figure 4.2: Koopman-based deep-learning approaches for CR3BP dynamics.

4.2.2. HNN Implementations

Recent investigation into the inclusion of HNN's in order to combat the loss of energy throughout propagation has yielded beneficial results. As explored upon in Section 4.2.1, DNN's for dynamic models often fail to generalize over long time spans and regions for which they have not been trained due to lack of generalization, resulting in state error accumulation and lack of energy conservation. HNN's have been demonstrated to address this by deriving dynamics using Hamilton's equations for an N-body system with an arbitrary fixed N:

$$H = \sum_{i=0}^{N-1} \frac{\|p_i\|^2}{2m_i} - G \sum_{i=0}^{N-2} m_i \sum_{j=i+1}^{N-1} \frac{m_j}{\|q_j - q_i\|} \tag{4.2}$$

Where H is the Hamiltonian, q, p represent the position and momentum respectively [47]. These scalar-valued functions encoding the total energy of the system are then encoded into the loss function [48, 49]:

$$\frac{dq}{dt} = \frac{\partial H}{\partial p}, \quad \frac{dp}{dt} = -\frac{\partial H}{\partial q}, \quad \mathcal{L}_{\text{HNN}} = \left\| \frac{\partial H}{\partial p} - \frac{dq}{dt} \right\|^2 + \left\| -\frac{\partial H}{\partial q} - \frac{dp}{dt} \right\|^2 \tag{4.3}$$

A typical DNN would thus predict the change in position and linear momentum of each body in an N-body system, whilst an HNN takes the same inputs to predict the Hamiltonian of the N-body system, which is in turn used to solve for the changes in position and linear momentum.

In order to effectively compare these networks to a numerical integrator scheme, they must be compared to a Hamiltonian based integrator, and thus for accurate comparison the symplectic Wisdom-Holman integrator is used as a benchmark Numerical integrator for comparison [48]. Figure 4.3 presents the predictions of a numerical integrator (left), in combination with HNN (middle) and DNN (left) for a Sun, Jupiter, Saturn (SJS) surrogate model. These predictions are presented for phase space (top), change in eccentricities per body (middle) and energy error (bottom) over the simulated/predicted time horizon tested. The performance of the HNN prediction, in terms of degradation of energy in time (bottom, middle), was again much more comparable to the numerical integrator (bottom, left) then the DNN (bottom, right). Notably, the temporal evolution of the energy error for the numerical integrator and the HNN remains bounded and qualitatively similar, whereas the DNN exhibits a clear growth in energy error. This behaviour indicates that the HNN-based surrogate better captures the underlying Hamiltonian structure of the system and is therefore capable of stable extrapolation over significantly longer timescales than a pure DNN.

This improved long-term fidelity comes at the cost of increased computational expense, as HNNs typically require approximately twice the training time and computational resources of comparably sized DNNs [47, 50]. Similarly to Section 4.2.1, the HNN implemented can still admit further improvements in terms of architecture and training strategies to improve model robustness/efficiency. Initial steps have been made, such as the introduction of a decaying learning rate for improved training stability. Nonetheless, energy conservation in multi-body systems has proven to be a useful tool for accelerating numerical integration, with significant speed-ups observed once the number of massless points (used in this case to represent asteroids) exceeds $N \geq 70$. This observation highlights the potential applicability of such models to the approximation of invariant manifold arcs, where similarly large ensembles of trajectories with shared underlying dynamics must be propagated efficiently.

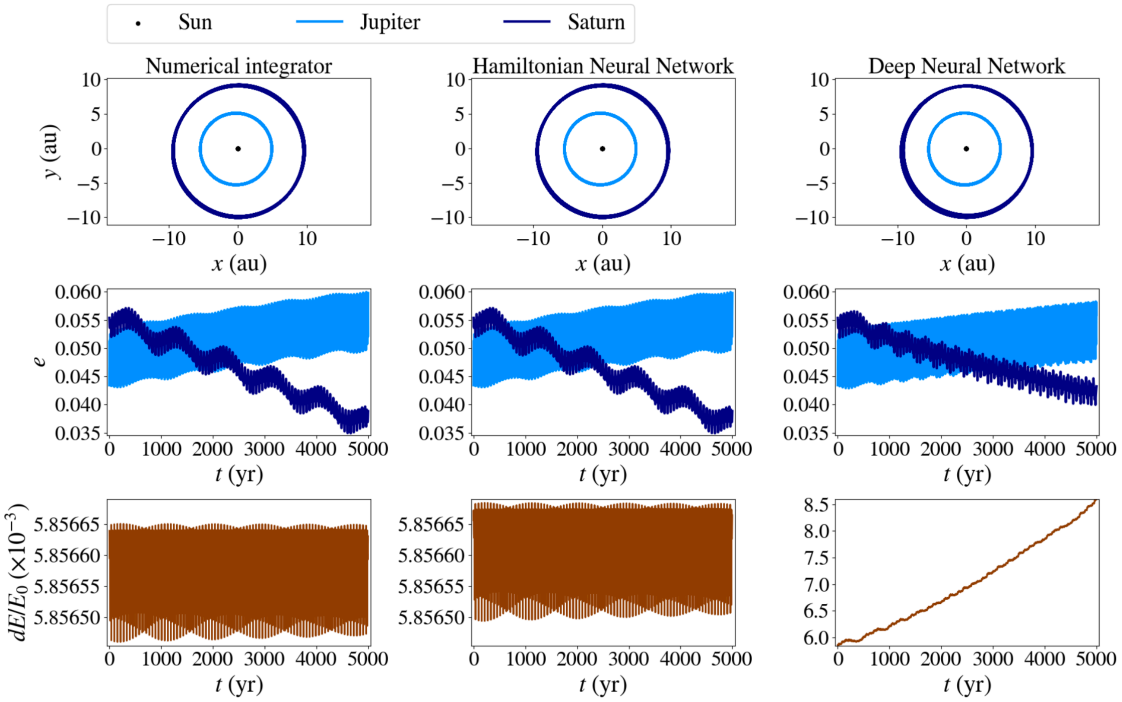


Figure 4.3: Relative (energy) error between HNN predictor, DNN predictor, and Numerical Integrator [48].

It is important to consider the drawbacks of including energy constraints in the loss function in HNNs as a result of the physics of the system. Due to the physical energy and symplectic structure constraints, they are sensitive to wide variations in input scale. An example of this is when propagating relatively smaller bodies within the system, as the order of magnitude difference between body masses results in the energy contribution of the smaller object being negligible [48]. This was demonstrated through an attempt to include a fourth negligible mass body to the system, resulting in training difficulties and significant performance impacts, requiring re-training of the model instead of adaptation, indicating a lack of robustness in these test cases. Furthermore, such systems are not able to model dissipative forces, thus requiring modification for the incorporation of external perturbations, should it be desired [47]. For investigation into the CR3BP, the Jacobi constant might be a more effective constraint for use than the Hamiltonian, due to its invariance in the rotating frame.

4.2.3. Hybrid Implementations

Well-engineered integrations of both machine learning and classical integration methods can be incredibly beneficial in making a robust system, opting for classic integration techniques when the neural network prediction is suspected to be subpar, such as during close encounter where a higher level of precision is necessary, or if the estimated error is deemed to be too large. This not only has the benefit of making the model more predictable, but also more interpretable, as each Hamiltonian/dynamical component is explicitly predicted. In these hybrid models, when the predicted error of the HNN/DNN is above a specified value, the system instead opts for the accelerations to be directly calculated via standard numerical integration methods. This is visible in Figure 4.4, where a NN is used to predict accelerations (a) at each time-step (blue) [48]. Predictions are then passed to a validation function (orange), which checks whether the change in acceleration between consecutive steps exceeds a predefined threshold, and if so, is replaced by numerical integration techniques for this time-step. This method prevents cascading errors throughout propagation, making the simulation more robust overall and acting as a safeguard to maintain accuracy over longer simulations. Naturally, this does once again come at the cost of increased overhead in terms of computational effort, whilst also increasing model complexity. Accurately tuning this threshold likely requires a ground truth reference, once again increasing computational effort, in order to find the lowest threshold that still gives acceptable accuracy while minimizing fallbacks to classical integration methods.

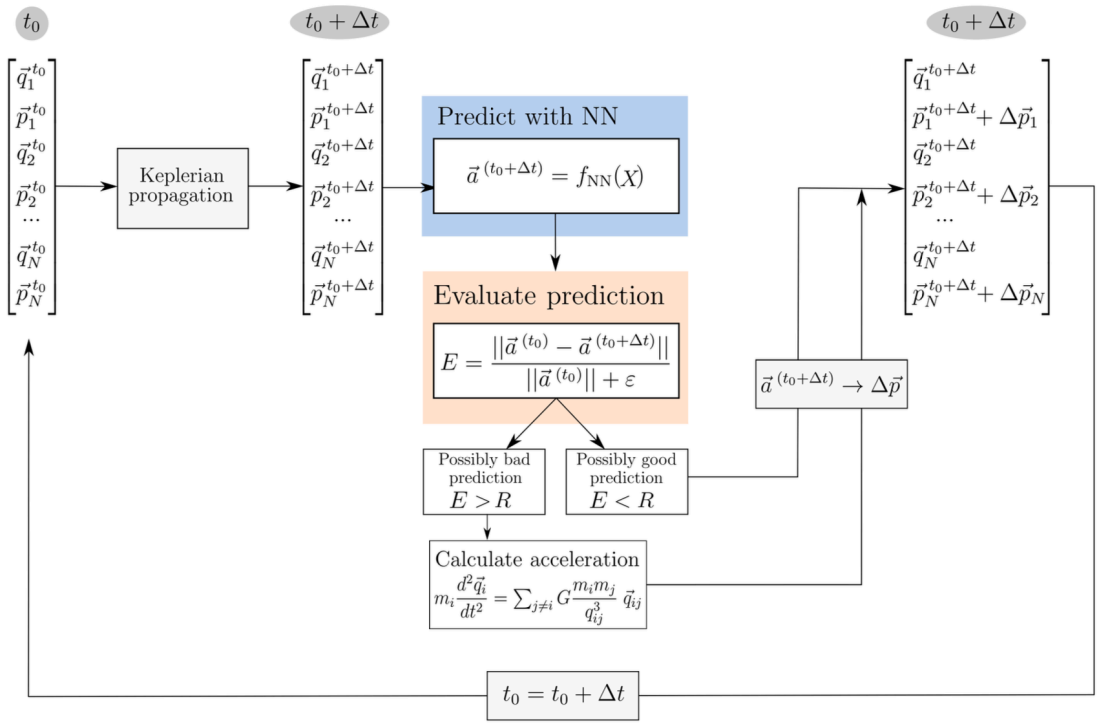


Figure 4.4: Schematic of a Hybrid DNN/HNN Integration Network [48].

Other hybrid solutions exist that use a second estimation function to determine the correction needed to refine the initial guess, demonstrated to be especially useful for estimating orbital perturbations. When such methods are compared to a standard SSDC correction, a measurable speed up of 10x was able to be achieved in terms of computational effort, whilst still maintaining an accuracy of 99.8% when tested on periodic trajectories with 10 revolutions [51].

Similarly, PINN's have demonstrated the potential to be used for perturbation estimation, both for unknown perturbations, and for direct estimation of perturbed accelerations based on known system states, helping stabilize diverging predictions. In particular, PINN's excel in reconstructing dynamics over short timescales, where trajectories remain locally stable [52]. In chaotic regimes however, PINNs tend to overfit, leading to poor generalization to test data or extrapolated time horizons, which suggests that they may be ill-suited for such systems [52].

4.2.4. Network Sensitivity

Both energy-preserving and standard neural surrogates (HNNs and DNNs, respectively) exhibit high sensitivity to configuration conditions, which strongly influences their ability to predict energy accurately over time. This is demonstrated in Figure 4.5, where for various initialization conditions of the NH3BP the energy dissipation over time are modelled [48]. In particular, the figure compares the temporal energy deviation of the neural surrogate (HNN: blue, DNN: red) predictions against the ground-truth solution across different random initialization seeds. This implementation does not possess the aforementioned hybrid architecture, and is limited to the pure DNN or HNN model, without any hyperparameter or architecture tuning, and is not a direct reference to the SJS implementation presented in Section 4.2.2. The standard DNN implementation exceeds this 10% energy exclusion condition at some point in the propagation for all cases, with some individual scenarios spiking to values of over 500%. Whilst the standard HNN implementation performed better then the DNN, the networks still violated the 10% energy conservation condition in approximately 20% of simulations. These energy losses typically occur in close encounters between bodies, highlighting an opportunity for the aforementioned hybridization technique.

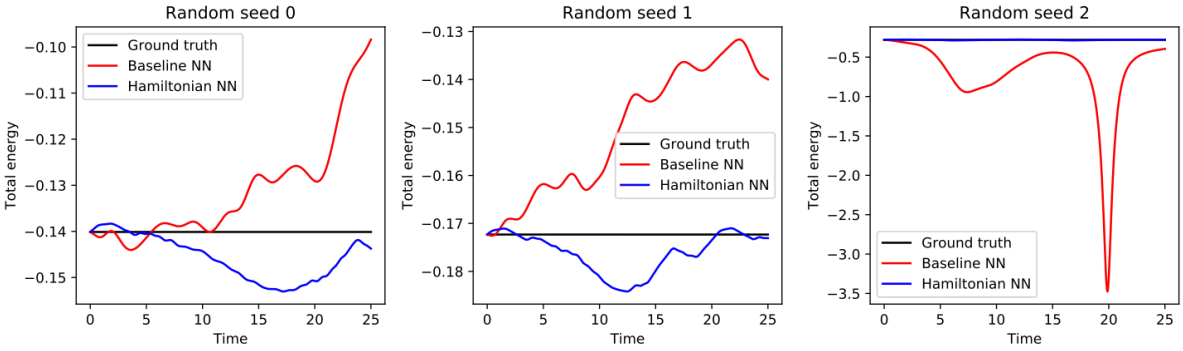


Figure 4.5: HNN and DNN energy dissipation over time for various initialization conditions [Modified] [48].

These results demonstrate that the observed energy violations are highly configuration dependent, rather than arising from a systematic model bias. This motivates the use of the ensemble learning methods highlighted in Section 4.1, for which multiple surrogates are evaluated and unreliable (transfer) trajectories are identified and discarded using an exclusion criterion. The 10% energy change exclusion condition [3] identified in this case serves as a diagnostic to maintaining statistical reliability when considering the behaviour of the ensemble, especially given the significant amount of violates demonstrated.

4.3. Generative Architectures

Generative methods have emerged as a promising new data synthesis and pattern modeling technique, with architectures such as generative adversarial networks (GAN's), variational auto-encoders (VAE's), and diffusion-based models exhibiting capabilities in generating high-quality time series data [53, 54, 55]. These models operate by learning data distributions and subsequently sampling from these learned representations to produce novel instances. Generative models can thus learn the underlying structures of known periodic orbits to generalize to new solutions, in an effort to accelerate the search for orbits with desired characteristics during the mission design process.

4.3.1. VAE-Based Trajectory Generation

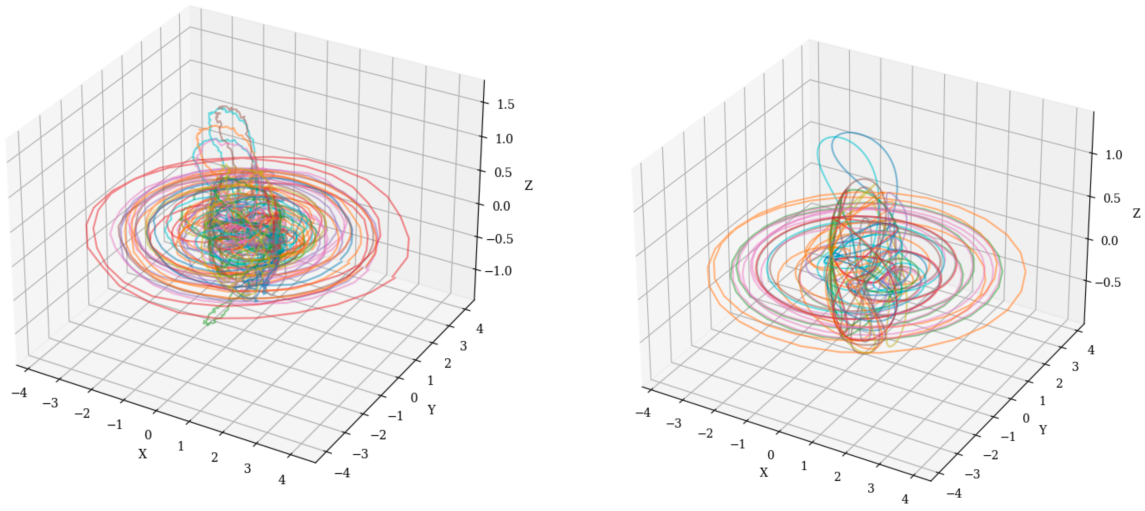
Current generative implementations in the CR3BP are limited to convolutional VAE architectures. VAEs are probabilistic generative neural networks that learn to compress data into a continuous probabilistic low dimensional latent space and then decode samples from that space to reconstruct or generate new data. In this context, the latent space is a learned low-dimensional representation in which each point encodes the full phase space of a trajectory. VAEs are trained by balancing reconstruction accuracy with a regularization term that shapes the latent distribution (typically toward a standard normal) through the following β-ELBO Loss:

$$L_{\theta,\phi} = \underbrace{-\lambda\, \mathbb{E}_{q_\phi(z|\mathbf{X})}\left[\log p_\theta(\mathbf{X}|z)\right]}_{L_{\text{MSE}}} + \underbrace{\beta\, D_{KL}(q_\phi(z|\mathbf{X}) \,\|\, p_\theta(z))}_{L_{\text{KL}}} \tag{4.4}$$

This loss function is expressed in probabilistic form, where the reconstruction component appears as an expected log-likelihood. In Equation 4.4, $L_{\theta,\phi}$ denotes the total loss of the VAE, parameterized by the decoder parameters θ and encoder parameters ϕ. The variable x represents the input data sample, while z denotes the corresponding latent variable. The encoder distribution $q_\phi(z|x)$ approximates the posterior distribution over the latent variables given the input, and $p_\theta(x|z)$ denotes the likelihood of reconstructing the input x from a latent sample z through the decoder. The expectation $\mathbb{E}_{q_\phi(z|x)}[\cdot]$ is taken with respect to the encoder's approximate posterior. The reconstruction term L_{MSE} is weighted by the scalar λ, which controls the relative contribution of the reconstruction loss. The Kullback–Leibler divergence $D_{KL}(\cdot\|\cdot)$ measures the discrepancy between the encoder distribution $q_\phi(z|x)$ and the assumed prior distribution $p_\theta(z)$, chosen as a standard normal distribution. This regularization term L_{KL} is weighted by the hyperparameter β, which governs the trade-off between reconstruction fidelity and latent space regularization [56].

The core of the implemented VAE generative pipeline is a convolutional architecture, tailored for time-series input [20]. A convolutional architecture is one that applies shared, localized convolutional filters along the temporal dimension of the trajectory data to capture local time-dependent patterns and correlations while reducing the number of trainable parameters. An exploration of identified applicable VAE architectures is presented in Section 4.5.3.

While the implemented VAE architecture generates samples that resemble true periodic orbits, they do not necessarily satisfy the physical constraints of CR3BP dynamics. This is demonstrated for 100 orbits generatively sampled from the latent space of a VAE, with each sample (coloured) plotted in spatial phase space in Figure 4.6 [20]. MSDC refinement is therefore typically used as a post-processing step in order to create a physically viable orbit [20]. Figure 4.6a presents the unrefined sampled orbits in spatial phase space, whilst Figure 4.6b presents the refined sampled orbits. In this example, an acceptable level of discontinuity (as described by Section 2.10) between MSDC segments is set to be $\tau_{MSDC} < 10^{-5}[-]$, whilst the computational cost of this refinement is not stated. A 46% sample convergence rate within 20 MSDC iterations is achieved [20]. The high noise levels in the unrefined samples (Figure 4.6a), and low MSDC convergence rate highlights both the need for the post-processing refinement method, and the sensitivity of MSDC to high noise levels in samples. Including physical information in the form of physics informed neural networks, in generative architectures has the potential to improve sample quality, yet remains to be explored in the CR3BP context [57].



(a) Unrefined VAE-generated orbits, spatial position

(b) Refined VAE-generated orbits, spatial position

Figure 4.6: Comparison of VAE generated orbits pre/post refinement algorithm, (a) shows the initially generated orbits with dynamic inconsistencies, (b) shows the refined orbits, corrected to satisfy physical constraints [20].

Note that all refined orbits are distinct from any seen during training, highlighting the VAE's potential to generalize to original dynamical solutions, despite only a 46% success rate. Classification of sampled orbits to families however remains to be explored. When presenting the 2-dimensional latent space of this VAE in Figure 4.7 distinct orbital families (coloured) are identifiable. Despite being trained without labels, orbits of similar types clustered naturally in the latent space, forming filament-like structures. This suggests the model had learned a interpretable internal representation of the orbital dynamics, demonstrating the potential for further utility in unsupervised orbit classification. This is of particular interest due to the inherent bifurcation relationship between families explored in Section 2.13.

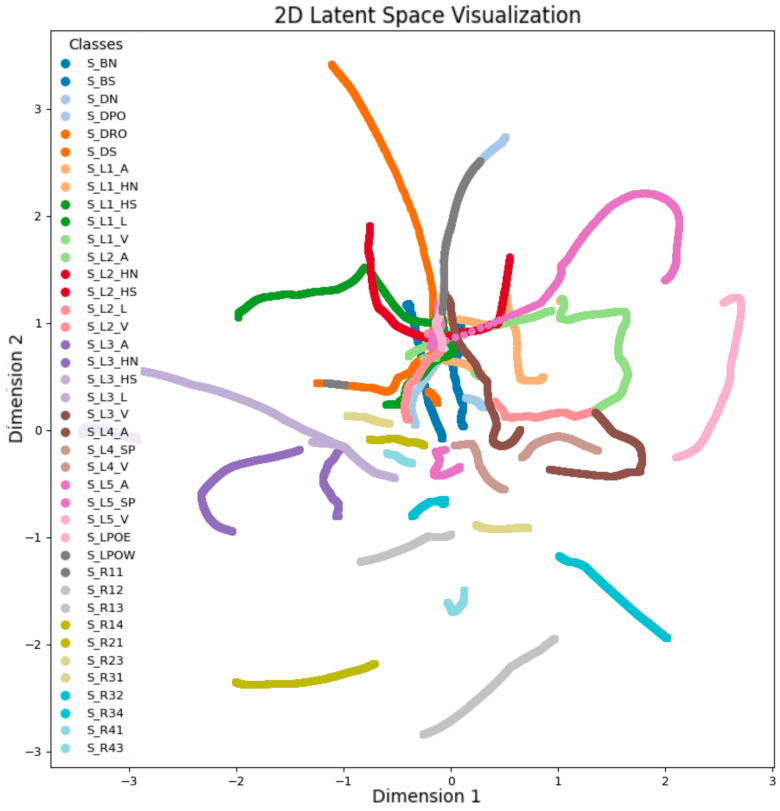


Figure 4.7: VAE latent space coloured by orbital family classes [20] [Modified].

4.3.2. Latent Operations

Latent variable models such as VAEs learn a low-dimensional manifold that captures the principal generative factors underlying the dataset. Within this manifold, nearby latent vectors correspond to similar data samples, and directions in latent space often encode meaningful modes of variation. Because the decoder is approximately smooth with respect to its latent inputs, simple linear operations in the latent space induce structured and predictable transformations in the reconstructed output. For example, latent interpolation between two encoded vectors is performed by linearly interpolating their latent vectors:

$$\vec{z}_\alpha = (1-\alpha)\,\vec{z}_1 + \alpha\,\vec{z}_2, \qquad \alpha \in [0,1] \tag{4.5}$$

In the context of trajectories, this has the potential to producing a smooth transition between the corresponding decoded sequences, compared to state space interpolation. Similarly, latent vector arithmetic allows the extraction and reapplication of specific attributes. Given a reference example ($\vec{z}_{\text{ref}}$) and an example exhibiting a desired attribute ($\vec{z}_{\text{attr}}$,) the latent direction associated with that attribute ($\vec{d}_{\text{attr}}$) is approximated and applied to a new latent representation ($\vec{z}_{\text{new}}$):

$$\vec{d}_{\text{attr}} = \vec{z}_{\text{attr}} - \vec{z}_{\text{ref}}, \qquad \vec{z}_{\text{new}} \approx \vec{z}_{\text{base}} + \vec{d}_{\text{attr}}. \tag{4.6}$$

Furthermore, small perturbations in latent space allow local exploration in the phase space, even though the corresponding transformations in the original state space may be highly nonlinear:

$$\vec{z}' = \vec{z} + \vec{\delta}. \tag{4.7}$$

This reflects the fact that the latent manifold provides a locally linearized representation of the underlying dynamics, making it an effective domain for interpolation, exploration, and controlled transformation of information. An example of this is demonstrated for words embeddings in natural language processing

in Figure 4.8. Here, a specific vector embedding per word is learned by the model. The learned latent vector representation results in similar words being grouped together, such as woman/man ($\vec{z}$). Sampling in this area would result in words that are similar $\vec{z}'$, such as boy/girl. The $\vec{d}_{\text{attr}}$ corresponding to royalty can be learned as the difference between woman/man and queen/king, in turn allows vector arithmetic to be applied for highly nonlinear transformations. This vector arithmetic can then be used to determine what the embedding for prince/princess to be using only $\vec{z}$ and $\vec{d}_{\text{attr}}$. Current literature does not apply vector arithmetic to either astrodynamic nor other nonlinear dynamic analysis contexts, indicating a promising avenue for exploration.

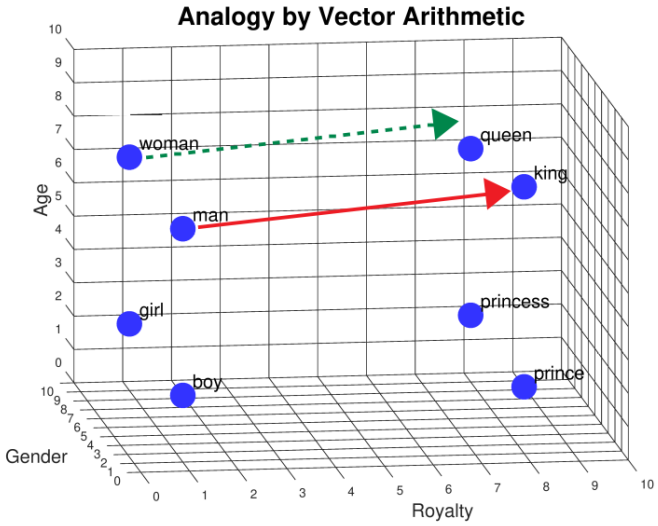


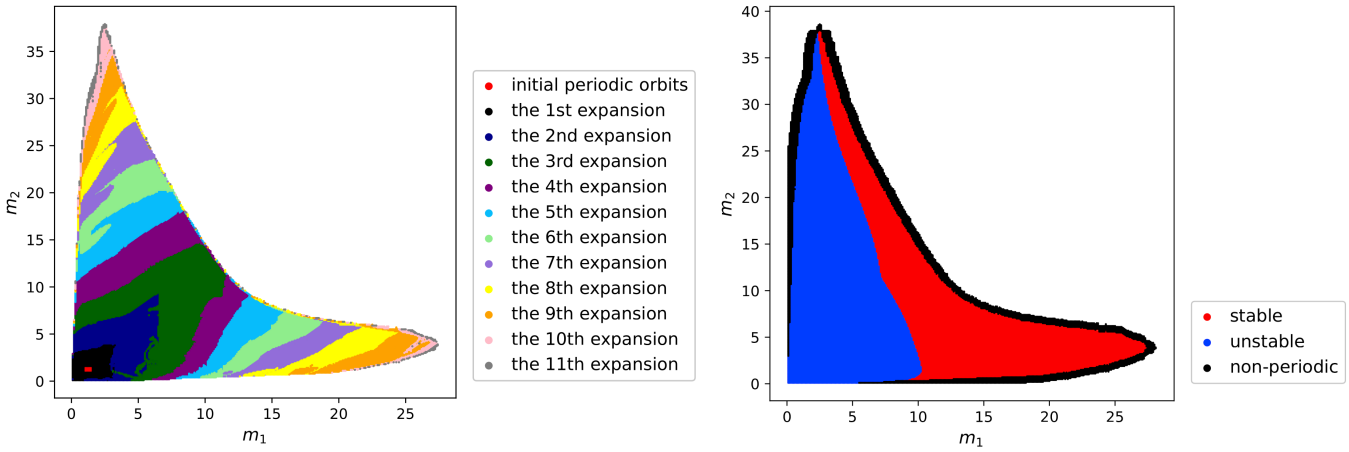
Figure 4.8: Vector arithmetic for learned latent word embedding [58]

4.3.3. DNN-Based Familial Continuation

The ability of DNN's to provide initial guesses for orbits within the same family based off an initial periodic guess is also demonstrated as an alternative to typical familial continuation exploration methods. This is explored for periodic orbits in the co-planar Newtonian NH3BP in a rotating reference frame. After an initial discovery of a periodic orbit, a DNN is demonstrated at inference time to predict other orbits within the family, before having these conditions confirmed/corrected through numerical simulation and SSDC methods (Section 2.10) [2]. The new orbits that satisfy the periodicity threshold ($||\mathbf{X}_P - \mathbf{X}_0|| < 10^{-10}$) are deemed to be validated and appended to the training set, prior to retraining on this expanded dataset. This DNN is trained to learn the mapping from possible mass parameters (m_1, m_2) to the initial conditions (x_1, v_1, v_2) and period T of the orbits required to be both periodic, and in the same family as the discovered orbit(s), whilst assuming m_3 to be constant at the unit mass $m_3 = 1$ [2].

This process is repeated over several extrapolation cycles, with each iteration expanding the search domain, and was demonstrated to generate 29,150 new orbits (all points) from a single initial guess (red dot). This is presented in Figure 4.9a, where the initial periodic orbit (red dot) is used, in combination with the DNN to iteratively continue the family. With each iteration (color) of the full DNN training/test cycle, new initial conditions (point in m_1, m_2) for periodic orbits are predicted.

The precision of these DNN predictions is validated by comparing predictions to initial conditions refined using numerical familial continuation techniques. The stability landscape of these predicted initial conditions is presented in Figure 4.9b. Of the generated initial conditions, 46.6 % were determined to be stable (red), 53.3% were determined to be unstable (blue), and 16.4% were determined to be non-periodic (black) [2]. This method aims to reduce the number of iterations required in SSDC through once again providing high-quality initial guesses, however no comparison to methods such as PAL or NPV (Section 2.11) is made.



(a) Domain Expansion Cycles

(b) Validity of Generated Solutions

Figure 4.9: Comparison of domain expansion and stability analysis for DNN-based periodic orbital family expansion [2]

It is also however important to be critical of such implementations. At its core, this model does not truly generate new solutions, instead acting as a regression-based interpolator/extrapolator across a discretely sampled parameter space defined by the mass ratios (m_1, m_2) within a specific family, for which the search domain is gradually increased with each expansion cycle. This in effect mirrors a guided search algorithm, or an algorithm more akin to a greedy search strategy or adaptive grid refinement, whilst the efficiency of such an implementation is not quantified. In addition to this, no parameters have been set to how different an orbit has to be to qualify as a newly discovered orbit (such as resolution within a family). This suggests that the quantity of newly discovered orbits (29,150) is more representative of the network interpolating between the known families rather than uncovering new structures. A small mass difference on the scale of the finite precision metric used could potentially qualify as a new orbit.

4.4. Orbital Classification Techniques

Machine learning tools have the potential to sort through vast amounts of trajectories and support high-level studies of the dynamical topology of periodic orbits in the CR3BP. Such tools are used to recognize different orbital regimes, classes, periods, or designate the stability of the proposed system, with the possibility of revealing emergent/underlying structures and dynamics. A particular future application is for identifying periodic orbits that bifurcate into new families, or the classification of bifurcation relationships between families.

4.4.1. Algorithm-Based Classification

Typical supervised ML-classification frameworks have been recently explored and have been proven to be promising in classifying stability regimes. For a test for the CR3BP, classifying stable orbital regions of the test particle around the least massive body, these classification frameworks demonstrated the ability to be highly accurate, whilst computationally efficient. Performance of 5 different algorithms were tested; XGBoost, LightGBM, Histogram Gradient Boosting, Random Forest and Decision Tree [59]. Stability in this setup was qualitatively defined as not being ejected or experiencing a collision with another body. Interestingly, only the initial conditions of the simulation were used for classification, with no supplemental time-series data of the propagation [59].

Table 4.1: AUC value for the best model of each algorithm explored in the classification of stability regimes [59]

Model	AUC
XGBoost	0.9978
LightGBM	0.9975
Histogram Gradient Boosting	0.9974
Random Forest	0.9957
Decision Tree	0.9554

All models explored achieved high accuracy scores, measured as AUC, presented in Table 4.1. High accuracy refers to AUC values exceeding 0.95, which are generally considered indicative of excellent classification performance in machine-learning contexts. XGBoost excelled particularly with an AUC value of 0.9978. Furthermore, this model also had a similar affinity for false positive and false negative, suggesting a robust model [59]. Values reported are however anomalously high, as performance degradation near boundaries of the stability regime is already noted for this system [59]. Further generalization testing is necessary to determine the applicability of these classification algorithms to more chaotic systems, explored further in Section 4.4.3. It could be possible that these models fail to generalize to a more chaotic system with boundary regions that are more difficult to resolve.

4.4.2. Convolutional Neural Network Classification

Alternative classification methods have been demonstrated through the use of Convolutional Neural Networks (CNN's) for the NH3BP [60]. Here, for a proxy for dynamical instability, they choose stability to be defined as if during its evolution, the semi-major axis ratio ($\frac{a_1}{a_2}$, inner to outer orbit ratio) deviates by more than 15% from its initial value. This stability criterion does not require the full disruption of the system (a body being ejected). Instead, it detects irreversible instabilities prior to the eventual ejection. Furthermore, it should be noted that the network was solely trained on the first 0.5% of the integration life sequence, for the purpose of training a robust network that quickly identifies unstable conditions without needing the entire trajectory to be propagated [60]. Two independent network architectures are tested, with architecture A having independent channel convolution layers per body prior to merging in the final layer, with 96,000 parameters. Architecture B is tested as having a multi-channel convolution layer in order to capture interdependency between bodies, with 59,000 parameters. Architecture A had 96,000 parameters whilst architecture B had 59,000. Each network had the same input parameters.

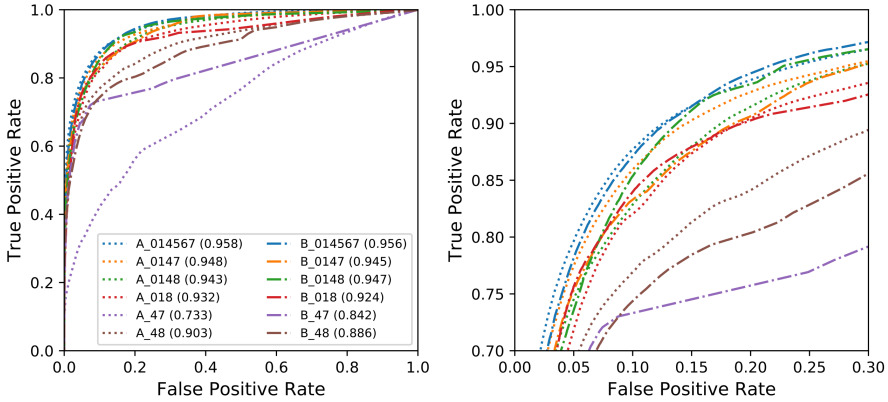


Figure 4.10: Receiver Operating Characteristic (ROC) curves of the tuned CNN Architectures [60]

Figure 4.10 presents the Receiver Operating Characteristic (ROC) curve, or the trade off between True Postive Rate and False Positve Rate, of the tuned CNN architectures. The best performing architecture (A-014567, blue dotted) is able to obtain an AUC of 95.8%. Names of each curve refer to which channels (variables in state $\mathbf{X}$), were included in the network architecture. Furthermore, it should be noted that architecture B was able to achieve similar performance with only 61.5% of the parameter amount that architecture A has, suggesting multi-channel convolution layers to capture dependencies on other bodies is a more efficient architecture for the system [60]. Architectures with reduced channel amounts are also demonstrated to have similar performances, providing the insight that not all channels

are required for effective convolution. In addition to this, the stability metric selected, while statistically motivated, is arbitrary and not based on the eigenvalue evolution of the periodic orbit.

4.4.3. Classification of Boundary Regions

There are limitations to when machine learning based classification techniques can be applied. An example of this would be boundary regions in the 3BP, where models experience poor generalization and low convergence rates. Boundary regions between regions in the 3BP are highly fractal in nature due to the chaotic dynamics of the system. An example of the difficulty to resolve such regions can be demonstrated for Active Learning (AL), where an AL DNN was implemented to predict one-hot encoding for stability for the restricted Sitnikov 3BP [61].

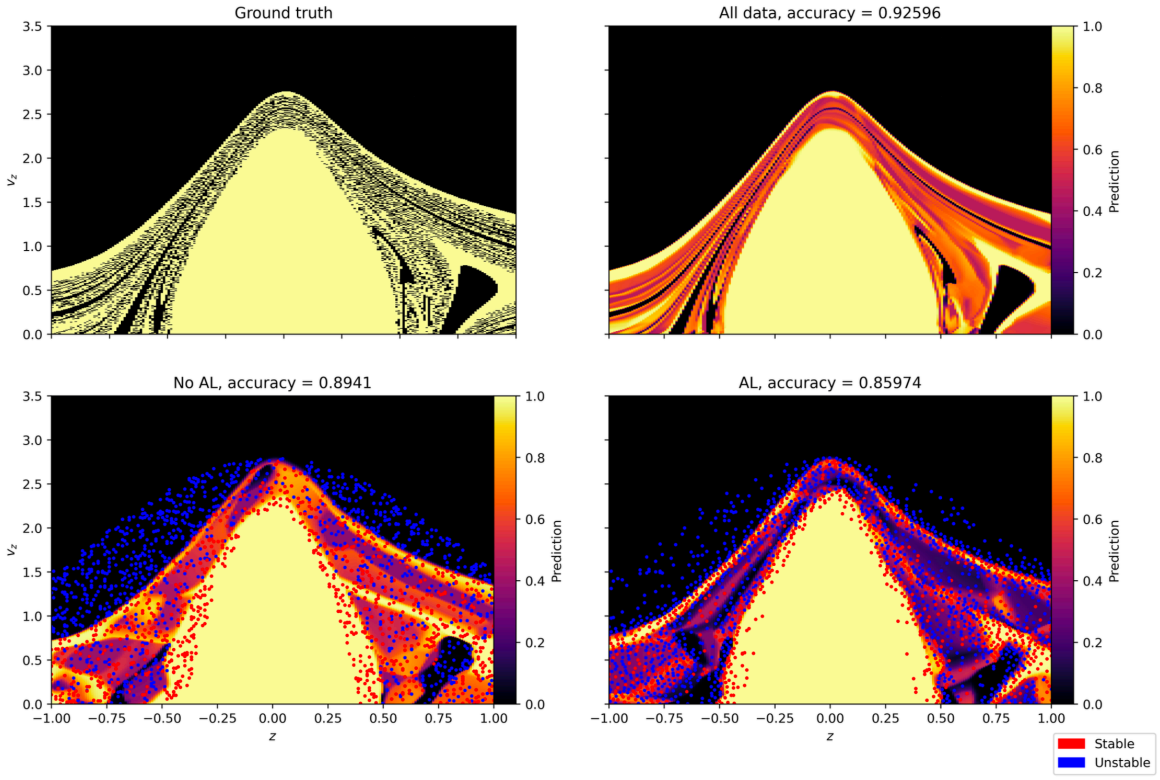


Figure 4.11: One-hot prediction results for stability boundary, with and without AL [61].

Contrary to the assumption that AL should reduce the number of training samples needed for effective generalization of the DNN, it was found that the AL frameworks not only failed to improve model performance but also resulted in worse predictive accuracy than random sampling, all whilst being computationally more expensive. This behaviour is illustrated in Figure 4.11, which presents a Poincaré map, a plot of a subspace of the state variables (z, v_z), of the dynamical system showing the distribution of sampled states and the corresponding classification outcomes in phase space. The top left figure presents the ground truth of the system, where yellow is a stable orbit, and black is an unstable orbit. All other sub figures (top right, bottom) present the prediction confidence of the DNN for stability (continuously coloured, 1: stable, 0: unstable). Exposing the DNN to the full training dataset (top right) allowed it to make predictions with a model accuracy of 0.92596. When reducing the amount of training samples (points, red, blue) in the set, AL reduced the model accuracy from 0.8941 to 0.8597, a reduction in accuracy of 10% [61]. This degradation in performance is attributed to the fractal geometry of the stability boundary in the problem investigated, implying that arbitrarily small perturbations in initial conditions drastically change the system's stability. This research thus yields the insight that when search space includes large fractal regions, AL is counterproductive, supporting the idea that the effectiveness of AL is context-dependent [61, 62]. For these fractal regions, alternative sampling techniques, or hybrid classic techniques should thus be used to effectively resolve these boundaries.

4.5. Approaches with Research Potential

In addition to developing an understanding of previously implemented models, it is also key to identify new techniques that have not yet been applied to the CR3BP for either regression, orbital generation, or classification purposes.

4.5.1. (Multiplicative) Kolmogorov-Arnold Networks

Kolmogorov–Arnold Networks (KANs) are a recent development, proposed as an alternative to traditional DNNs. Inspired by the Kolmogorov–Arnold representation theorem (Equation 4.8), namely that any continuous multivariate function can be expressed as a finite sum of outer functions (Φ_q), each applied to a sum of univariate functions ($\varphi_{q,p}$) [63].

$$f(x_1, x_2, \ldots, x_n) = \sum_{q=1}^{2n+1} \Phi_q \left(\sum_{p=1}^{n} \varphi_{q,p}(x_p) \right) \tag{4.8}$$

Whilst DNNs place fixed nonlinear activation functions (σ) on neurons and learn linear weights (W_i) on edges, KANs nodes perform summation, and the edges carry learnable activation functions (Φ), each parameterized as a spline [63]:

$$\text{DNN}(\mathbf{x}) = (W_3 \circ \sigma_2 \circ W_2 \circ \sigma_1 \circ W_1)(\mathbf{x}) \quad \Big| \quad \text{KAN}(\mathbf{x}) = (\Phi_3 \circ \Phi_2 \circ \Phi_1)(\mathbf{x}) \tag{4.9}$$

This allows KANs to directly learn univariate nonlinear functions along edges, making them much more efficient and information dense then DNN's. Whilst KAN parameter amounts scale much quicker, $O(N^2LG)$ for a network of depth L, width N, and spline order/point count G, compared to $O(N^2L)$ for DNN's, they typically require a much smaller layer amount then DNN's, saving parameter amount and increasing generalizability. This is demonstrated in Figure 4.12, where the tested KAN models (blue) exhibit a steeper scaling behaviour than their MLP counterparts (yellow, green, red, purple). Specifically, the KAN error scales approximately as $\mathcal{O}(N^{-4})$ (red, dashed), compared to $\mathcal{O}(N^{-0.04}) - \mathcal{O}(N^{-4})$ for MLPs (black, dashed), indicating a higher-order polynomial improvement in scaling efficiency compared to parameter count (N) across various test scenarios (left, middle, right) [63].

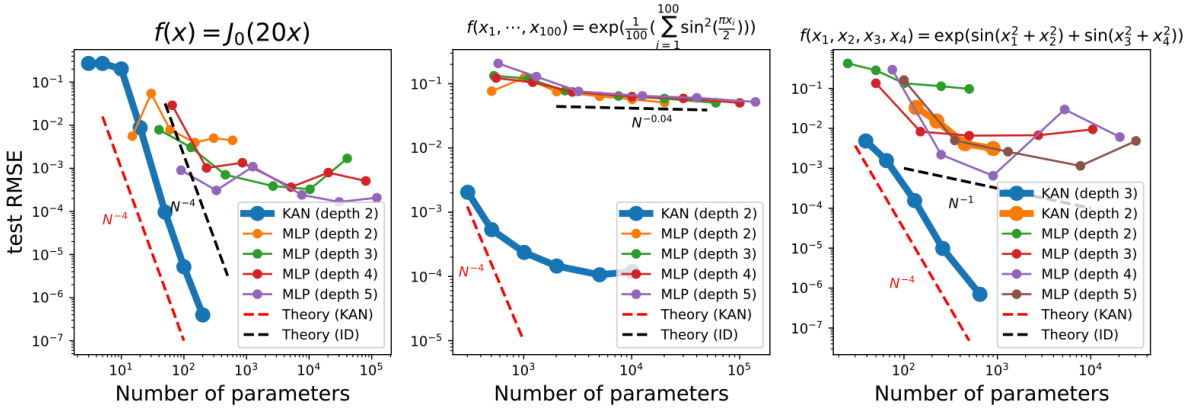


Figure 4.12: KAN vs DNN performance on three examples. KANs scale more efficiently and are more information-dense than DNNs [63].

Multiplicative Kolmogorov–Arnold Networks (MultKAN) extend the original KAN framework by explicitly introducing multiplicative operations between intermediate nodes, which makes it easier to capture nonlinear interactions such as products of variables that commonly arise in physical systems. This extension allows the network to directly represent terms like $x_1 \cdot x_2$ or higher-order nonlinearities without forcing them to emerge through combinations of learned univariate functions. Such networks have been demonstrated to be powerful for physics-informed spline fitting tasks, allowing even higher information density. MultKAN's have already been demonstrated to be able to extract physical dynamics and governing ODE/PDE equations directly from noisy measurement data in multiple physics contexts, such as the Duffing oscillator, Van der Pol oscillator (stiff ODE), Burger's equation, and Bouc-Wen model,

amongst others [64]. Such networks could allow much more information dense/low parameter count implementations for use in regression models, especially in nonlinear systems such as the CR3BP.

4.5.2. Transformers as Chaotic Control Mechanisms

Transformers have been a major focus on research in the past few years with application to chaotic time-series data. The self-attention mechanisms present within transformers has been demonstrated to allow for long range time dependency modeling. Primary uses have been to predict chaotic time-series data for estimating safety functions to control the investigated parameter, such as for control mechanisms in mechanization or power grid systems [65, 66, 67]. Specific applications have as of yet been limited to control mechanisms for chaotic time-series data [65]. This could have multiple practical implementations, such as controlling error within a bounded margin per step evaluation, similar to the method explored in Section 4.2.3. An alternative implementation could be in the form of direct application for correction mechanisms such as a supplement/alternative to SSDC, through learning nonlinear dependencies between distant states in time-series data.

In combination with this, limited spaceflight-dynamic oriented implementations have been demonstrated for other network types, such as Lagrangian Neural Networks (LNN's) [50]. These architectures, unlike HNN approaches, are not restricted to canonical coordinates, allowing for direct learning of energy conservation dynamics. This could provide the benefit of overcoming constraints in HNN's due to the relatively small energy contribution of the spacecraft to the total energy of the system. This scenario is analogous to the modeling of small charged particles in dominant magnetic fields, for which LNN performance has already been demonstrated [50]. Nonetheless, whilst only control-based implementations have been demonstrated, application of transformer architecture types for predictive capabilities remains to be explored in the CR3BP context.

4.5.3. Alternatives VAE Architectures

Beyond standard (dense) VAE formulations, a range of alternative VAE architectures have been proposed to better accommodate structured time-series data. These variants differ primarily in how temporal correlations, structure, and behaviour are encoded and reconstructed, for example through the use of convolutional operators (seen in Section 4.3.1), explicit decomposition of trends and seasonal components, or architectural inductive biases tailored to sequential data. Such design choices are particularly relevant in the context of the CR3BP, where orbital trajectories exhibit periodicity and multi-frequency structure. Several VAE architectures with demonstrated relevance to time-series modeling are introduced below and discussed with respect to their applicability for representing periodic orbits and trajectory data in the CR3BP.

4.5.3.1. Dense-VAE

The naive implementation of a VAE is denoted as Dense-VAE, in which all channels (features in $\mathbf{X}$) are flattened into a single vector and passed through a sequence of fully connected (linear) layers, with their depth, number of nodes, and activation functions determined via hyperparameter tuning. This culminates in two linear heads that produce the mean and log-variance of the latent variables. The decoder mirrors the encoder's structure but reconstructs features only in the final layer. Since the decoder's output can be any real valued number in a CR3BP context, its final layer cannot posses any activation function (e.g. ReLU) and instead must use an dense output layer.

4.5.3.2. Convolutional-VAE

As touched upon in Section 4.3.1, convolutional VAE's perform strided convolution for each channel (feature in $\mathbf{X}$). Each layer applies temporal kernels across all channels simultaneously, using strided convolutional layers (yellow) to progressively reduce the sequence length with each layer, as presented in Figure 4.13. After the last convolution, the output is flattened into a single vector and sent to two separate linear layers that produce the mean and log-variance of the latent variables, represented by the normal Gaussian distribution. Once again, the decoder inverts this process, with a dense layer first expanding the latent vector back into a tensor matching the encoder's last convolutional output. This is then passed through a mirrored sequence of deconvolutional layers (blue) to restore the original sequence $\hat{\mathbf{X}}$.

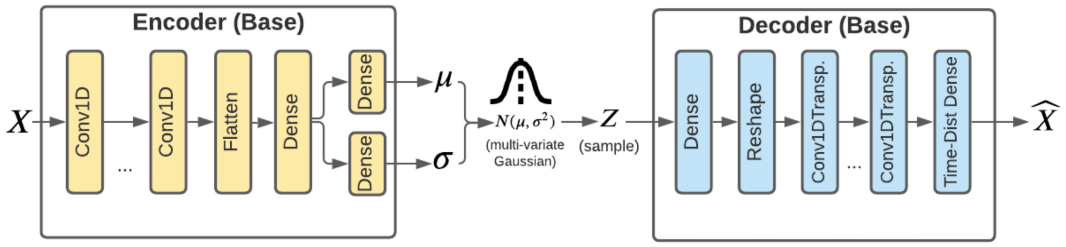


Figure 4.13: Block Diagram of the Convolutional Network. (De)Convolutions are applied per channel/feature [56]

4.5.3.3. TimeVAE

A new VAE architecture, known as TimeVAE, is an architecture specifically developed for the generation and modeling of multivariate time-series data, with the objective of incorporating domain-relevant temporal structure directly into the generative process. In the TimeVAE implementation, presented in Figure 4.14, the encoder (yellow) functions similarly to the one in the convolutional VAE. TimeVAE instead extends the standard convolutional VAE formulation by modifying the decoder (blue, green) to explicitly decompose reconstructed sequences into multiple components. The components are compromised of a level term (constant offset), a trend term (polynomial) and and one or more seasonality terms (sinusoidal periodic term). Rather than reconstructing each time step independently from the latent vector, the TimeVAE decoder predicts the parameters governing these components, which are then combined to form the final time-series output $\hat{\mathbf{X}}$ [56].

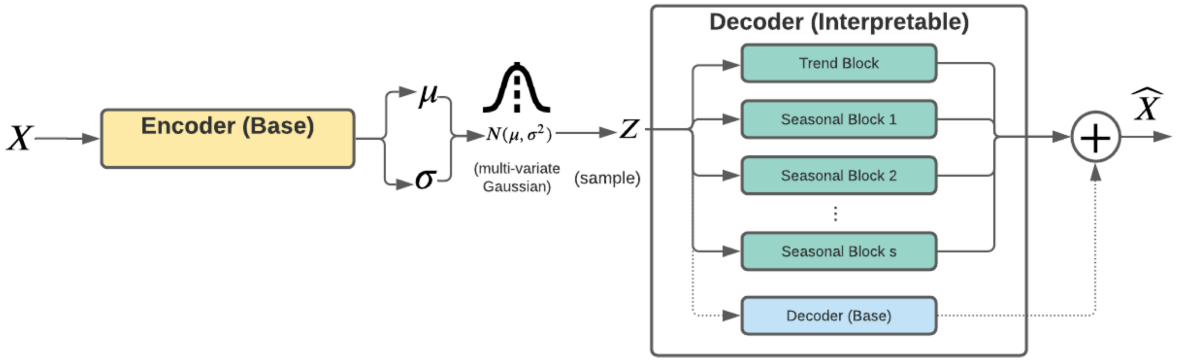


Figure 4.14: Block diagram of TimeVAE. (De)Convolutions are applied per channel/feature [56]

Each seasonality block can thus be used to independently model a different phase of the periodic orbit, which is then summed with other seasonality blocks, serving as a fourier-esque decomposition of the time series. For illustration, consider a time series of length 100. Suppose such a model employs three seasonality heads with configurations (4, 25), (5, 20), and (10, 10). The first head (4, 25) generates four seasonal values, each applied over 25 consecutive time steps, partitioning the sequence into four equal phases (steps 0–24, 25–49, 50–74, and 75–99). The second head (5, 20) produces five seasonal values, each spanning 20 steps, yielding five phases that tile the sequence twice (0–19, 20–39, 40–59, 60–79, and 80–99). The third head (10, 10) yields ten seasonal values of duration 10 steps each, repeating ten times over the full sequence. Each head learns its set of seasonal parameters from the latent representation, potentially enabling the model to capture periodic behaviour at multiple temporal scales in the context of the CR3BP, ranging from coarse orbital phases to fine-grained periodic fluctuations. The outputs of all seasonality heads are then combined via element-wise summation and (optionally) integrated with the level and trend components to form the reconstructed time series.

This multi-scale decomposition has the potential to supports representing periodic structures with overlapping frequencies, in the context of the CR3BP this is to allow separate modelling of the gravitational influences of the primary and the secondary. The trend block is not considered of relevance to this investigation, as this would imply a underlying change in the energy levels (Jacobi) of the orbit, which is not possible in the CR3BP. This could in future implementations be used to model external perturbations

(e.g. constant low-thrust, solar radiation pressure).

4.5.4. Time-Series Diffusion Models

The use of diffusion models in time-series forecasting has also grown in recent years, offering an alternative to traditional frameworks such as VAEs and Generative Adversarial Networks (GANs). At their core, diffusion models simulate a process whereby input data is incrementally perturbed into Gaussian noise through a forward diffusion process, before being reconstructed via a learned reverse process. Whilst having already been applied to descent trajectory generation, ballistic transfer trajectories, and other stochastic trajectory mechanisms, limited application for chaotic trajectory mechanisms have been demonstrated, such as those in the CR3BP [68, 54, 53]. Evaluation across multiple chaotic time-series forecasting models has demonstrated the applicability for these models to handle long-range dependencies, especially for generation of comparable full-length time-series data formats. Furthermore, recent research into Physics-Informed Diffusion Models (PIDM's) have proved effective at integrating physical laws into the generative process, enabling improved accuracy, stability, and generalizability in complex dynamical systems, however has not yet been demonstrated in an astrodynamic context [57]. Despite this, diffusion models remain computationally intensive due to their iterative reverse process. Training costs can be high when compared to VAE's, particularly for long time-series, as a result of each prediction requiring multiple denoising steps.

4.5.5. Large Kernel CNN's (LKCNN)

Recent developments within the CNN space, dubbed Large Kernel Convolutional Neural Networks (LKCNN's) due to the networks being characterized by kernels that are large relative to the input size, have proved to be particularly useful in the classification of time-series data inputs. Not only have these networks proved to be more accurate in chaotic time-series classification tasks, but are also more tractable due to their architectural simplicity [69]. In all test cases, stability metric reconstruction through LKCNN outperformed typical methods, where performance was attributed to the large receptive fields of the LKCNNs, which could capture long time-sequence periodic activations. Whilst this has been proven to be highly successful in the classification of periodicity themselves, it has yet to be demonstrated for classification within periodic data, or for quasi-periodic time-series data, such as orbits in higher fidelity models. Nonetheless, this provides insight for an extension to the research developed in Section 4.4.2.

5

Research Proposal

The current computational cost of exploring transfer trajectories of periodic orbits hinders quick iteration during mission design. Current models struggle with energy conservation, physical validity, and generalization across different CR3BP regimes. Such challenging regimes arise both within a single orbital family, where dynamical properties can vary sharply along the family, and across multiple orbital families, where distinct stability and phase space dynamical structures must be navigated. This extra computational burden inhibits research into mission design where these dynamics are utilized, thereby hindering the practical deployment of such missions. Determining how to both effectively and efficiently deploy methods that navigate these regimes is therefore critical for enabling mission design. This in turn also establishes a benchmark for applying architectures to less chaotic dynamical systems [19]. Overcoming generalization issues and computational bottlenecks is essential not only for theoretical understanding, but for practical application in mission planning and execution. In combination with this, there remains critical research gaps in exploring suitable periodic orbits for long-term mission use. Current exploration schemes either rely on grid-search or limited bifurcation theory based explorations. Generation schemes often require costly post-processing to produce physically valid orbits.

This research will focus on adapting and improving methods relating to the discovery of new periodic orbits, while also exploring practical applications of these orbits in spacecraft trajectory optimization. In particular, the goal is to develop and benchmark the efficiency of frameworks that can generate, validate, and classify dynamically feasible orbits within chaotic gravitational regimes, namely the CR3BP. Reducing the burden traditionally associated with periodic orbit discovery should enable the design of low-energy trajectories for use in future space missions, whilst also providing insights into the underlying dynamical structures that govern motion in chaotic gravitational systems. Specific research objectives and questions to address these shortcomings are outlined below in Section 5.1, in combination with their motivation [19].

5.1. Research Objectives and Questions

This research aims to improve the simulation, understanding, and design of trajectories in highly-chaotic gravitational systems through the development of an AI-based framework. The overarching research questions and objectives are twofold and are outlined below. Supporting sub-questions that are designed to guide the investigation with specific technical challenges within this domain are stated for each question.

5.1.1. Research Objectives

To discover, validate, and characterize periodic orbits in chaotic gravitational systems using a unified AI-assisted methodology, and to analyse the stability and bifurcation structures of these orbits.

OBJ.1. Quantify the accuracy and efficiency of neural network surrogate models compared to traditional numerical integrators for simulating invariant manifold arcs.

OBJ.2. Develop a physics-informed generative modeling framework capable of producing dynamically feasible periodic orbits in the CR3BP.

5.1.2. Research Questions

RQ1. **To what extent can different VAE types, both with and without energy conservation mechanisms, be adapted for periodic orbit generation for spacecraft mission design in the CR3BP?**

Classical orbit discovery methods are computationally demanding and often limited to known families. AI-based generative models offer an alternative for discovering new orbits, but their reliability and generalizability, need further exploration. Currently implemented methods leave room for improvement, most notably in their tendency to generate dynamically infeasible orbits that require extensive post-processing refinement. Reducing this refinement step by including energy conservation mechanisms could significantly improve both sampling efficiency and applicability to trajectory planning and understanding of dynamics.

RQ1.1. To what extent can different generative model architectures (dense, convolutional, timeVAE's, and TransFusion) reduce the level of post-processing refinement required to generate dynamically valid periodic orbits?

RQ1.2. What is the comparative efficiency of generative models vs classical continuation methods for in family interpolation in the CR3BP?

RQ1.3. Can latent sampling facilitate extrapolation to unseen (bifurcated) families?

RQ1.4. To what extent does including physical (energy) constraints within generative models improve orbit feasibility and computational efficiency?

RQ1.5. Can learned latent representations approximate physical bifurcation relationships between orbital families?

RQ1.6. Can vector arithmetic be applied to latent space representations for both mission design and classification purposes?

RQ2. **To what extent can DNN's, with and without energy conservation mechanisms, and KAN's be used for computationally efficient regression of manifold arcs?**

Simulation in chaotic systems like the CR3BP is computationally expensive and highly sensitive to errors. Naive models tend to accumulate error through in-abacus interpolation and propagation in time. There is a need to evaluate whether AI models can match or surpass classical methods in both computational efficiency and stability for long-term propagation tasks, particularly for invariant manifold regression tasks. Neural surrogates have been demonstrated to have sufficient accuracy with complex architectures [44, 48], however analysis of whether or not numerical systems are more efficient at the required accuracy level remains to be understood.

Energy conservation is critical for reliable simulation in systems where energy is preserved. Existing AI approaches often ignore or approximate conservation laws, leading to physically invalid results. Determining how energy conservation errors will in turn have an impact on the feasibility of periodic orbits, and transfers between them is thus essential for their application.

RQ2.1. What is the associated accuracy of DNN's and KAN's with and without energy preservation mechanisms (PINN-losses), in terms of spatial, dynamical and energy errors for the regression of invariant manifold transfer trajectories?

RQ2.2. How does the inclusion of energy preservation mechanisms facilitate the accuracy and computational efficiency of DNN regression techniques, in the context of invariant manifold approximation?

5.1.3. Model Requirements

In order to guarantee the reliability of orbital predictions and to enable comparison with existing numerical solutions, the following model requirements are outlined. These requirements ensure that model predictions adhere to physical constraints and maintain accuracy within the predictable horizon of the system. Literature sets a threshold for trajectory positional accuracy in the CR3BP model on the order

of magnitude of $\Delta p < 1 \times 10^{-5}$ [LU], arising from state differences for Halo-orbits over 10 full periods after transition to an EHF model [70, 71].

$$\Delta p < 10 \text{ [km]} \approx 2.565 \times 10^{-5} \text{[LU]} \approx 1 \times 10^{-5} \text{ [LU]} \tag{5.1}$$

Trajectory velocity accuracy requirements are typically adapted on the same order of magnitude, resulting in the following preliminary trajectory accuracy requirements:

$$\Delta p < 1 \times 10^{-5} \text{ [LU]}, \quad \Delta v < 1 \times 10^{-5} \text{ [LU/TU]} \tag{5.2}$$

These discrepancies vary widely across orbit families and manifold arcs due to differences in dynamical sensitivity, with differences on the order of multiple orders of magnitude between periodic orbits already visible within the same family. This is demonstrated for the $L_{2,N/S}$ Halo family, where an overview of the typical discrepancies in position between the CR3BP and EHF models from literature is summarized in Table 5.1.

Table 5.1: Summary of CR3BP to High-Fidelity (HF) Orbit Positional Differences for Earth-Moon Halo Families [70, 71]

(Sub)Family (Period Range)	**Positional Difference**
(Near-Rectilinear) Halo ($6.5 < P < 7.0$ [days])	$\Delta p < 10$ [km] initial difference; ~ 100 [km] drift in long-term propagation.
(Long) Halo ($6.5 < P < 7.0$ [days])	$\Delta p < [100$ km] initial difference; ~ 1000 [km] drift in long-term propagation.
(Short) Halo (< 6 [days])	No bounded analog in EHF model.

State-of-the-art orbit determination capabilities in the cislunar region however are only able to provide position accuracy on the order of kilometres and velocity accuracy on the order of centimetres per second, which in non-dimensional units correspond to the same order of magnitude (10^{-5} [-]) [34, 39, 41]. This means that while the EHF model must satisfy this state-accuracy requirement, the CR3BP trajectory analysis only needs to meet accuracy levels sufficient to maximize transition rates to the EHF, allowing for some relaxation of these requirements, if transfer to the EHF is desired. Convergence in literature has been demonstrated for CR3BP state accuracy levels relaxed to the order of magnitude of 10^{-3} [-], however this is highly dependent on the specific trajectory [41, 40]. Nonetheless, optimal convergence rates to the EHF are achieved with state accuracy levels on the order of magnitude of 10^{-5} [-].

REQ.1: $\Delta C < 1\%$
The energy of simulated spacecraft, measured by the Jacobi, must not vary by more than 1% over the course of the propagation. This ensures results remain suitable for ensemble statistical analyses, and physical validity of the simulation. [3]

REQ.2: $\Delta p < 1 \times 10^{-5}$ [LU]
The maximum allowable position error is set to 1×10^{-5} LU. This threshold is derived from the CR3BP-EHF transition margins reported for the Earth–Moon system, with an added conservative margin to ensure reliable model-to-model continuity.

REQ.3: $\Delta v < 1 \times 10^{-5}$ [LU/TU]
The maximum allowable velocity error is set to 1×10^{-5} LU/TU, ensuring dynamical consistency with the position accuracy target and preserving trajectory integrity during CR3BP–EHF transitions.

REQ.4: $\mathcal{C}_{\text{surrogate}} < \mathcal{C}_{\text{simulation}}$
The total computational cost ($\mathcal{C}$) of performing regression-based predictions must remain lower than the cost of performing the equivalent numerical simulation at inference time. This ensures that the surrogate model provides a tangible efficiency advantage without compromising the established accuracy requirements. This shall be measured in FLOPs.

REQ.5: $\tau_{SSDC} = \|\mathbf{X}_P - \mathbf{X}_0\| < 10^{-10}$
The final state $\mathbf{X}_P$ of a periodic orbit after one full period must match the initial state $\mathbf{X}_0$ within a tolerance of 10^{-10} in dimensionless units, within a single numerical integration scheme [35]. This periodicity constraint ensures that numerical propagation preserves the closed nature of the orbit and maintains consistency for subsequent manifold generation or continuation processes.

REQ.6: $\tau_{MSDC} = \|\mathbf{X}_0^{i+1} - \mathbf{X}_f^i\| < 10^{-8}$
The discontinuity between segments for a converged MSDC scheme must be no greater than 10^{-8} in dimensionless units, within a single numerical integration scheme [35]. This continuity constraint is relaxed compared to the SSDC scheme, allowing for greater numerical stability across a larger node amount than the SSDC implementation.

6

Data and Numerical Benchmarking

6.1. Existing Periodic Orbit Data

As the discovery of new periodic orbits is an inherently complex task, it is necessary to evaluate repositories for the CR3BP for both their completeness and interpretability, such as the JPL Repository (Section 6.1.1). Software such as AUTO is specifically developed to explore relationships between periodic solutions, facilitating automated familial continuation, bifurcations, and subsequent familial continuation of child families that arise from bifurcations [72].

6.1.1. JPL Repository

JPL maintains a repository of initial conditions for 772,744 periodic orbits across 7 three-body systems and 43 distinct orbital families [27]. Only the Earth-Moon system subset will be explored in this investigation, split over the 31 classes presented below in Table 6.1, organized by their associated libration points or symmetry types.

Table 6.1: Classification and original dataset count of periodic orbit families by associated branches or symmetry types [27]

Family Name	Branch / Type	Orbit Amount (Branch/Type)
Lyapunov	L_1, L_2, L_3	3108 (L_1), 4298 (L_2), 5498 (L_3)
Halo	$L_{1,N}$, $L_{2,N}$, $L_{3,N}$, $L_{1,S}$, $L_{2,S}$, $L_{3,S}$	5731 ($L_{1,N}$), 1535 ($L_{2,N}$), 6175 ($L_{3,N}$), 5731 ($L_{1,S}$), 1535 ($L_{2,S}$), 6175 ($L_{3,S}$)
Vertical	L_1, L_2, L_3	6670 (L_1), 9651 (L_2), 6999 (L_3)
Axial	L_1, L_2, L_3	2498 (L_1), 2198 (L_2), 6999 (L_3)
Long Period	L_4, L_5	4703 (L_4), 4703 (L_5)
Short Period	L_4, L_5	6999 (L_4), 6999 (L_5)
Vertical	L_4, L_5	9172 (L_4), 9172 (L_5)
Axial	L_4, L_5	7003 (L_4), 7003 (L_5)
Butterfly	$L_{2,N}$, $L_{2,S}$	3238 ($L_{2,N}$), 3238 ($L_{2,S}$)
Dragonfly	$L_{2,N}$, $L_{2,S}$	1409 ($L_{2,N}$), 1409 ($L_{2,S}$)
Distant Retrograde	N/A	10998
Distant Prograde	N/A	5168
Low Prograde	East, West	2695 (East), 2726 (West)

It should be noted that this database is limited to a number of orbital families identified through the familial continuation and bifurcation methods stated above in Section 2.13, acting as a skeleton from which other families can be discovered [27, 33]. Practically, this means that for every family present in the dataset, not all children (families that arise from bifurcations) are included in the dataset. The repository contains information about the initial state ($\mathbf{X_0}$) of each periodic orbit, in combination with the period ($\mathbf{P}$) in order to facilitate propagation from $\mathbf{X_0}$ to $\mathbf{X_P}$:

$$\mathbf{X}_0 = [x_0,\ y_0,\ z_0,\ \dot{x}_0,\ \dot{y}_0,\ \dot{z}_0], \qquad \mathbf{P} = P \tag{6.1}$$

6.1.1.1. Bifurcation Characterization

The bifurcationary relationships between these families can be characterized through a Broucke stability diagram analysis, as explained in Section 2.13.2. These results are additionally used for validation purposes in comparison to literature [33, 34]. As an example, the NRHO subset of the $L_{2,N/S}$ Halo family is presented in Figure 6.1. In this case, the NRHO family exhibits five distinct bifurcation types: three quadrupling (red pentagon), one tripling (purple star), one tangent (blue circle), and two period-doubling bifurcations (orange diamond) , listed in order of first appearance in decreasing Jacobi constant. Within the context of the dataset, the two period-doubling bifurcations are of particular interest, corresponding respectively to the butterfly and dragonfly families (again, in order of decreasing Jacobi constant). This type of analysis can be applied to each family to both identify the relevant bifurcating orbits and to construct a node–edge graph representing a feature-independent bifurcation diagram.

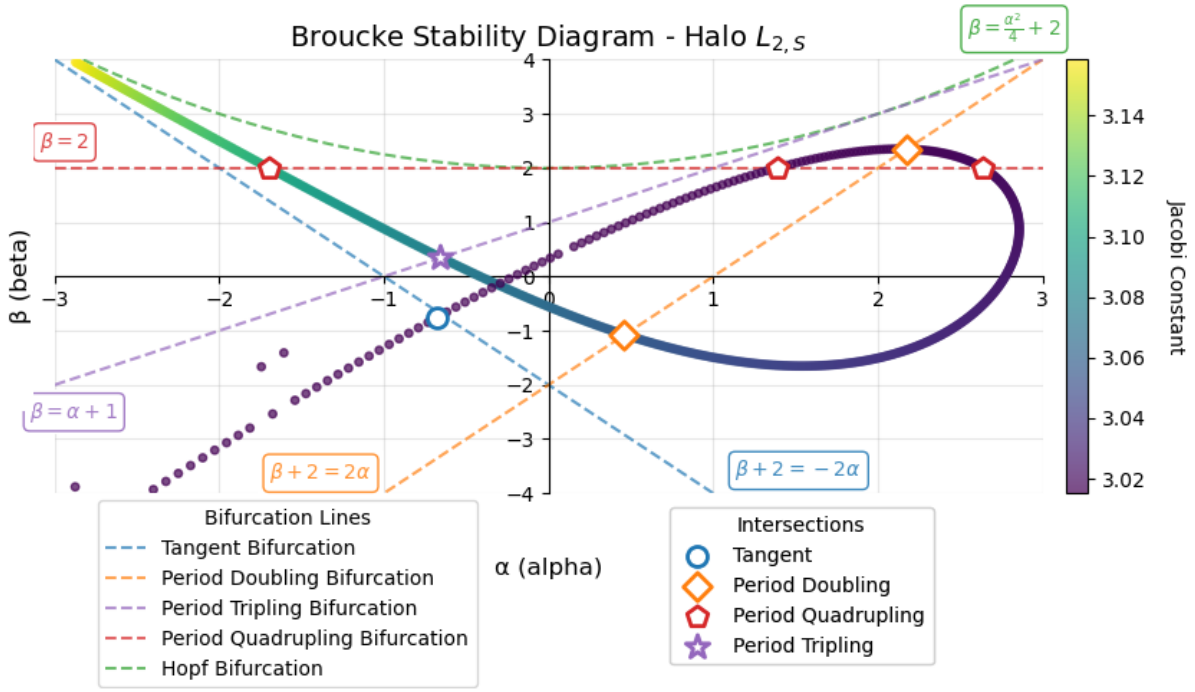


Figure 6.1: Broucke stability diagram of the NRHO subset of the $L_{2,N/S}$ Halo family, illustrating the locations of bifurcations (marked intersections).

Similar figures for all families in the dataset are found in Appendix A, whilst comparison to Figure 2.17 is used for validation purposes [34]. Note that Figure 2.17 contains a smaller subset of NRHO orbits, yet preserves the same overall family structure and bifurcation locations. The consistency between these two figures validates the bifurcation structure shown in Figure 6.1. Three points (purple) appear as outliers relative to the remainder of the family and are likely the result of numerical instabilities rather than true dynamical deviations. A similar method is demonstrated for tangent bifurcations in Figure 6.2 arising from the L_2 Lyapunov family (blue). In this case, the Lyapunov family crosses the tangent boundary ($\beta + 2 = -2\alpha$, dashed) at two distinct points (white dots), which give rise to the $L_{2,N/S}$ Halo (green) and L_2 Axial (pink) families. Note a third white dot identifies another tangent bifurcation from the L_2 Axial family, however this diagram is chosen to limit the families to those arising from bifurcations from the L_2 Lyapunov family.

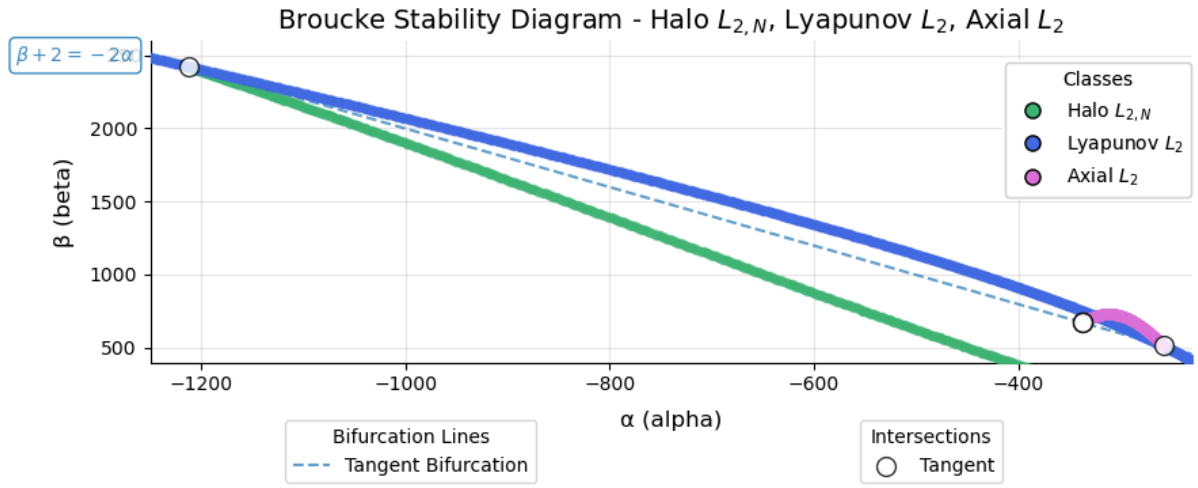


Figure 6.2: Broucke stability diagram illustrating tangent bifurcations of the L_2 Lyapunov family.

A node–edge graph representing these families and their bifurcations is presented in Figure 6.3. The icon shape indicates the type of bifurcation observed, while smooth node (green hexagon) connections signify that as an orbit family grows or shrinks, it transitions smoothly into either the inner (turquoise) or outer (teal) circular family. The inner family is circular about Earth, whereas the outer family is circular about the Earth–Moon system. These circular families are excluded from the dataset, since their dynamics are largely Keplerian and therefore of limited relevance to the non-Keplerian regimes of interest.

A subset of data is selected as JPL-A, consisting of the families that bifurcate from the L_2 point, and all subsequent child families that bifurcate from those. This subset is denoted as the red dotted line in Figure 6.3 The corresponding node–edge graph for this subset is presented in Figure 6.4, along with node numbering to denote each bifurcation, presented in Table 6.2. These families are selected for evaluation purposes due to their frequent use in mission design and their distinctive (continuous) bifurcation relationships. Further note that this node-edge representation will later be used for verification and validation purposes in Section 7.4.2.

Table 6.2: Node Numbering Family and Type Overview

Number	Families	Type
I	L_2 Axial, L_2 Lyapunov	Tangent
II	L_2 Axial, L_2 Vertical	Tangent
III	$L_{2,N}$ Butterfly, $L_{2,N}$ Halo	Period-doubling
IV	$L_{2,N}$ Butterfly	Terminal
V	$L_{2,S}$ Butterfly, $L_{2,S}$ Halo	Period-doubling
VI	$L_{2,S}$ Butterfly	Terminal
VII	$L_{2,N}$ Dragonfly, $L_{2,N}$ Halo	Period-doubling
VIII	$L_{2,N}$ Dragonfly	Terminal
IX	$L_{2,S}$ Dragonfly, $L_{2,S}$ Halo	Period-doubling
X	$L_{2,S}$ Dragonfly	Terminal
XI	$L_{2,N}$ Halo, $L_{2,S}$ Halo, $L_{2,N}$ Lyapunov	Tangent
XII	$L_{2,N}$ Halo	Terminal
XIII	$L_{2,S}$ Halo	Terminal
XIV	L_2 Lyapunov, L_2 Vertical	Lagrange (Tangent)
XV	L_2 Lyapunov	Terminal
XVI	L_2 Vertical	Terminal

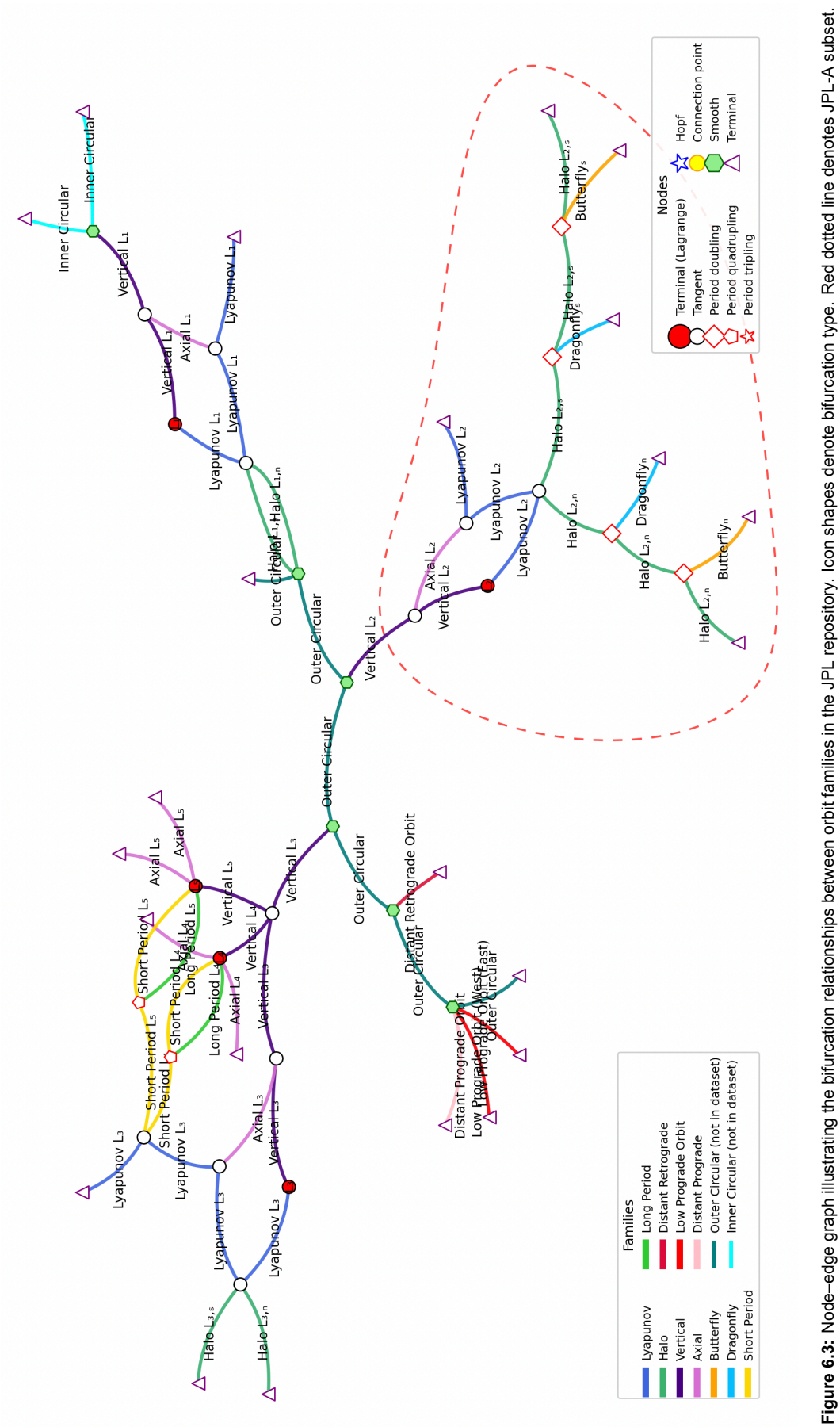


Figure 6.3: Node–edge graph illustrating the bifurcation relationships between orbit families in the JPL repository. Icon shapes denote bifurcation type. Red dotted line denotes JPL-A subset.

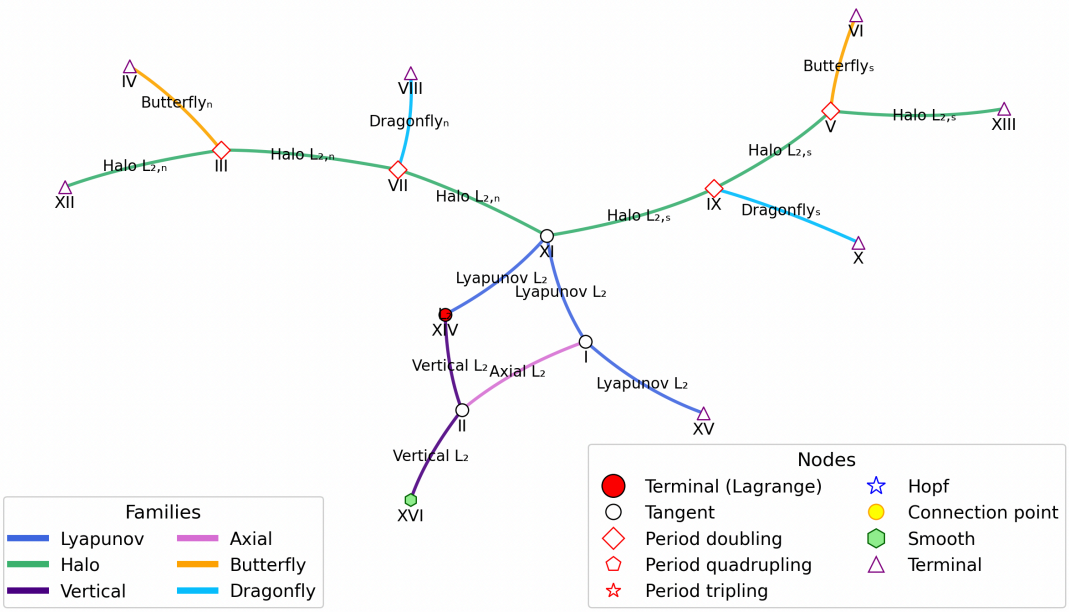


Figure 6.4: Node–edge graph of the JPL-A subset, showing orbit families bifurcating from the L_2 point and their interconnections. Each bifurcation node is numbered.

Note that for each node, a periodic orbit exists in both families corresponding to that individual orbit, i.e. III-$L_{2,N}$ Halo and III-$L_{2,N}$ Butterfly are two individual orbits present in each family. This is demonstrated for the period-doubling bifurcations from the $L_{2,N}$ Halo family in Figure 6.5, where the location and associated name of each bifurcating orbit is presented in positional phase space for the $L_{2,N}$ Halo Family. As stated in Table 6.2, Node XI corresponds to the tangent bifurcation from the Lyapunov family, and VII & III to the period doubling bifurcations to the Dragonfly family, and Butterfly family respectively. XII is a terminal node, with no bifurcating structure that extend from it. Note that for both the period doubling bifurcations, nodes III and VII, a visually similar orbits with approximately twice the period appears in the $L_{2,N}$ Butterfly and $L_{2,N}$ Dragonfly family respectively, marked in Figure 6.6a and Figure 6.6b respectively.

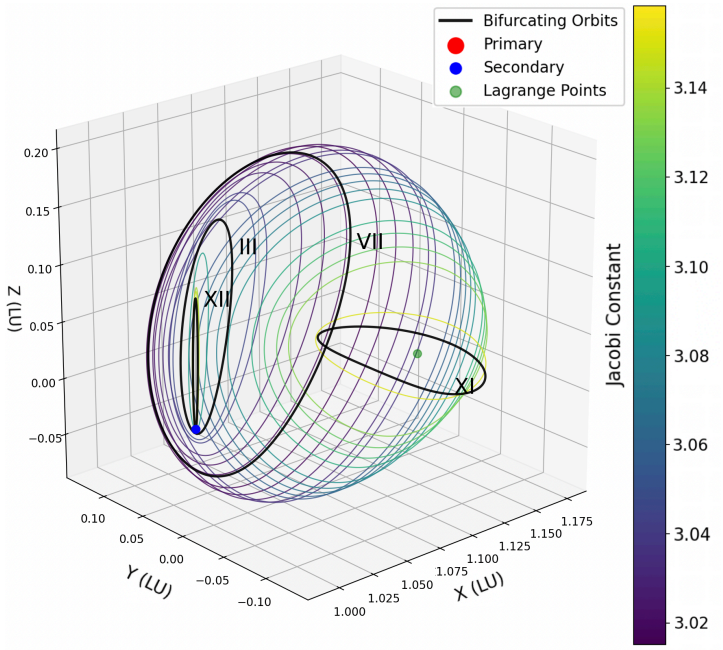
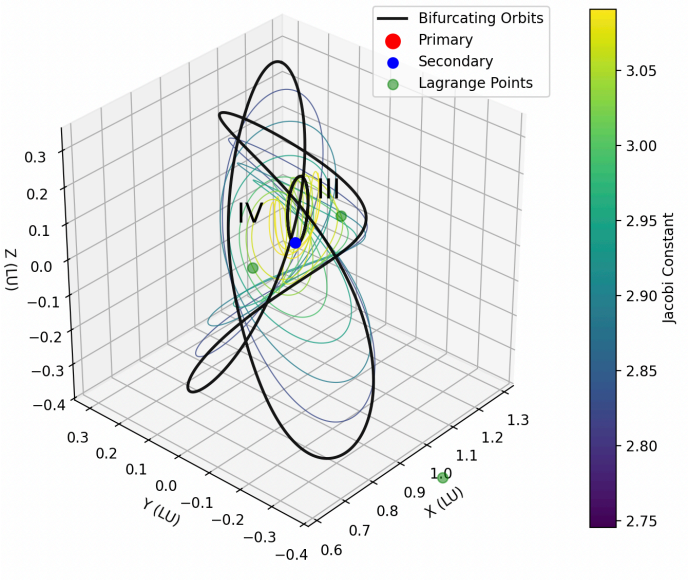
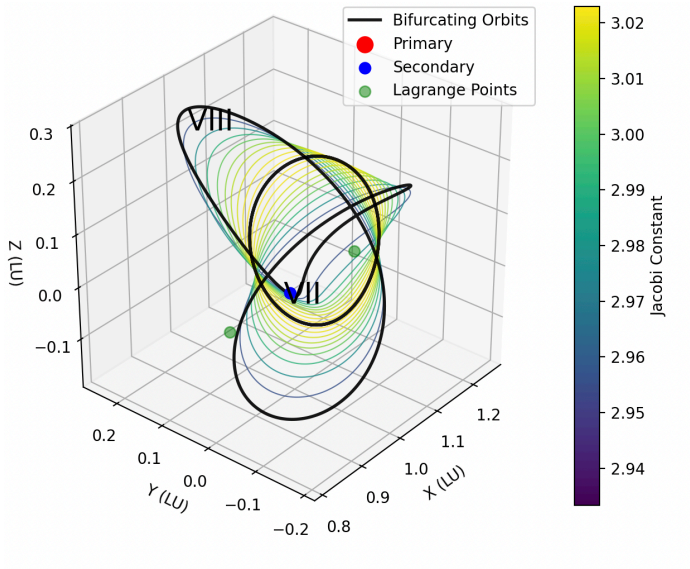


Figure 6.5: $L_{2,N}$ Halo, coloured by Jacobi in spatial phase space. Bifurcating orbits indicated.

In phase space, tangent bifurcation orbits are identical to their parent orbits, however this is not the case for period-multiplying bifurcations. At the bifurcation point, the resulting orbits initially appear similar because the multiple passes of the trajectory remain in close proximity, but they gradually diverge as the family is continued. This behaviour is illustrated in Figure 6.6a and Figure 6.6b, where at the bifurcation points (III and VII, respectively), the two passes of the orbit appear close to one another, however these passes progressively separate in phase space from with continued variation of the Jacobi constant along the family, until their most extreme separation in phase space near the termination nodes IV, VIII respectively.



(a) $L_{2,N}$ Butterfly family coloured by Jacobi in spatial phase space.



(b) $L_{2,N}$ Dragonfly family coloured by Jacobi in spatial phase space.

Figure 6.6: $L_{2,N}$ Dragonfly and Butterfly orbital families, coloured by Jacobi, bifurcation nodes indicated.

Further note edge length is not representative of location within family, as edges are set to be equidistant for the purpose of bifurcation plotting in Figure 6.4. This is evident in Figure 6.5, where the location and associated name of each bifurcating orbit is presented in position space for the $L_{2,N}$ Halo Family. Note the larger difference in phase space between orbits VII and XI compared to XII and II, yet these edge are selected to have equal distance in Figure 6.4. Edge distance is further selected to be constant, as the number of orbits along edges is arbitrary due to the families themselves being continuous, and would thus be dependent on the ΔC at which all families are resolved. Similar figures for each family in JPL-A are presented in Appendix A, in combination with their orthographic projection about each plane.

6.2. Numerical Integration

The integrators provided by the SciPy package are used in this investigation due to their frequent adoption in scientific computing, ease of implementation and flexibility. An overview of integrators available within this package is found below in Table 6.3. It should be noted that all the integrators listed in Table 6.3 are adaptive time-step methods, i.e. the solver dynamically adjusts the size of each time step based on the local behaviour of the solution.

Adaptive integrators estimate the local truncation error at each step and compare it to a specified tolerance. If the estimated error exceeds the allowed tolerance, the step is rejected and recalculated with a smaller step size. Adaptive integrators are considered in this investigation due to their ability to balance computational efficiency and solution accuracy through using larger steps in smooth regions to speed up computation, and smaller steps in regions with rapid changes to maintain accuracy and stability. This is especially of relevance to the CR3BP due to the underlying dynamics being nonlinear/chaotic. The solution thus changes gradually in some regions and rapidly in others, making uniform time-stepping inefficient or unstable.

Table 6.3: Summary of integrators in `scipy.integrate.solve_ivp`.

Method	Integrator Type	Order	Adaptive Step Size
RK23	Explicit Runge-Kutta	3(2)	Yes
RK45	Explicit Runge-Kutta	5(4)	Yes
Radau	Implicit Runge-Kutta	5	Yes
BDF	Implicit Multistep (BDF)	Variable	Yes
LSODA	Adams / BDF (Auto-selected)	Variable	Yes
DOP853	Explicit Runge-Kutta	8(5,3)	Yes

To quantify the numerical effort involved, the number of floating-point operations (FLOP) is used as the primary metric in this investigation. A FLOP represents a single arithmetic operation performed on floating-point numbers such as an addition, subtraction, multiplication, or division. FLOPs provide a hardware-agnostic measure of computational cost, capturing all operations required to evaluate the system's equations of motion, including those arising from internal sub-steps, error estimation, correction steps, and any rejected steps within adaptive integration schemes.

An important distinction is that while adaptive time-step integrators are effective for maintaining accuracy and efficiency, they do not scale well to massive parallelization (e.g., 100+ concurrent processes on GPUs). Adaptive methods introduce computational overhead beyond the core FLOPs, with each step involving estimating the local truncation error, performing tolerance checks, and potentially recalculating rejected steps. This results in bottlenecks when the behaviour of the numerical integration scheme varies over different regions, due to the need for processor synchronization [73].

This effect is illustrated in Figure 6.7, where processors P_0 and P_1 result in a bottleneck being created for the remaining processors. Each point in the figure represents a single integration step executed by an individual process within an adaptive time-stepping scheme. In this example, $P_0$and P_1require repeated step refinements to satisfy the error tolerance, whereas the other processors complete their steps with fewer iterations. As a result, processors P_2 to P_{15} must idle while waiting for synchronization before advancing to the next integration step, thereby reducing overall computational throughput. This behaviour is especially relevant to the CR3BP, where close-encounter dynamics require smaller time-steps for accurate resolution.

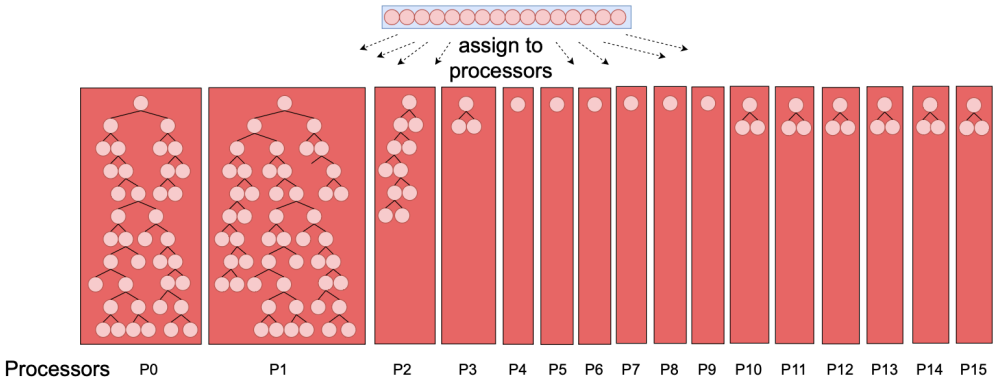


Figure 6.7: Process Imbalance in Adaptive Time Step Integrators [73]

Whilst several adaptive numerical integration schemes for GPU deployment exist, such as PAGANI, these integration schemes are typically reliant on partitioning the integration domain into spatial regions for parallel processing. This approach is not directly applicable to trajectory integration, as it would assume the evolution of the trajectory is known a priori [73]. These implementations are more suitable for finite element methods, where the space is divided into regions. An example of this would be simulating particle swarms, where gravitational interactions or density distributions can be computed over discrete spatial cells [73, 74].

"Embarrassingly parallel" implementations of numerical integrators are typically limited to fixed time-step schemes. This limitation is closely tied to the Single Instruction, Multiple Data (SIMD) nature

of GPU architectures, requiring that the same instruction be applied uniformly across multiple data points/within each tensor core. As a result, to achieve comparable accuracy in systems with rapidly changing dynamics (such as close encounters with the bodies), the time step must be reduced. This significantly increases the number of required integration steps and leads to high computational cost, despite the ease of parallelization.

6.3. Periodic Orbit Classification

Clustering of periodic orbits is necessary to evaluate both convergence rates of MSDC schemes (MSDC verification) and to detect new families (family clustering). Under noisy conditions, the MSDC pipeline may converge to an unintended periodic orbit rather than the target solution. This is demonstrated for a noisy planar L_2 Lyapunov orbit (grey, dotted) in Figure 6.8, where noise in the sub-sampled nodes (yellow, square) resulted in convergence of the MSDC scheme to a circular orbit (blue) with different characteristics than desired. This has been measured to occur in $\approx 3\%$ of all corrections across a variety of different experimental schemes, detailed in Section 7.6.1.

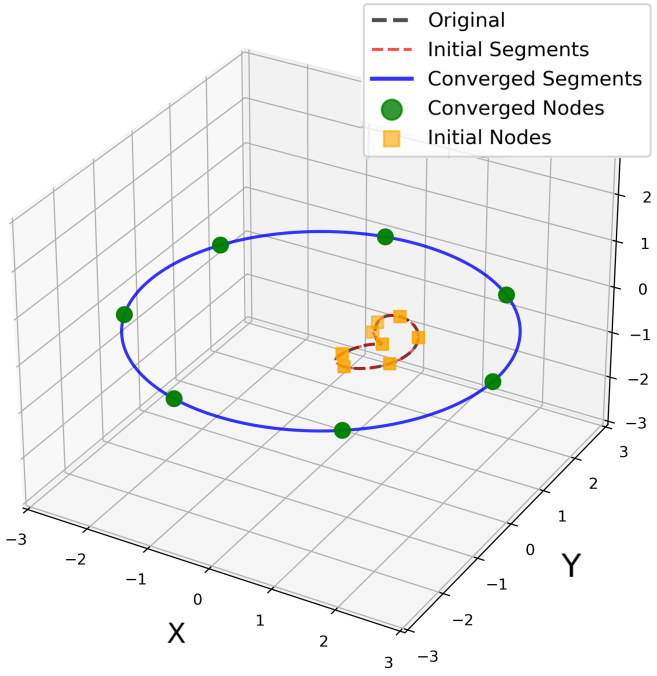


Figure 6.8: Convergence of a noisy L_2 Lyapunov orbit to an outer circular orbit as a result of MSDC

6.3.1. Clustering Algorithms

Current literature implements Density-Based Spatial Clustering of Applications with Noise (DBSCAN), chosen for its ability to find clusters of arbitrary shape without knowing cluster count in advance [72]. Clustering is applied using a single normalized initial condition (defined at a specific symmetry plane crossing, such as the x-y or x–z-plane), stability indices, symmetry type, and number of revolutions. This approach produces a large number of orbit families (20-30%) with low clustering confidence(< 0.75) in unlabelled datasets. The reduced confidence is likely due to two factors: initial conditions are not unique, since periodic orbits can intersect the same plane multiple times (see period-multiplying bifurcations, Table 2.4), and proximity in initial-condition space does not always reflect similarity in full phase space, causing nearby points to belong to different families. Including stability indices, revolution count and plane tangency conditions helps mitigate this issue, but does not entirely resolve it, demonstrated by the high amount of low confidence clusters. Various options to resolve this issue specifically for time series data are considered, along with their relevance, presented in Table 6.4

Table 6.4: Comparison of time-series feature handling options for clustering/classification

Option	Description	Pros	Cons	Clustering Feature Prep Complexity
Flatten feature vectors	All $6 \times N$ time steps are flattened into one long vector for clustering or classification.	– Captures full joint temporal evolution across all features. – Easily compatible with standard machine learning and clustering algorithms.	– Very high dimensional, leading to redundancy and noise, and class collapse – Requires subsampling (e.g., reducing from 100 to 10 timesteps) to remain tractable.	$\mathcal{O}(N)$
FFT feature compression	A Fast Fourier Transform (FFT) is applied to each feature's time sequence to reduce it to a set of frequency and amplitude pairs for classification.	– Provides strong dimensionality reduction. – Phase information is no longer considered. – Highlights periodicity and spectral content within the data.	– Frequency scaling becomes inconsistent due to variable real-time durations across sequences. – Noise in samples will result in new frequencies being added to feature list – Requires normalization and careful selection of frequency bins.	$\mathcal{O}(N \log N)$
DTW	Dynamic Time Warping (DTW) is used to measure shape-based similarity between temporal sequences across all features.	– Captures shape-based similarity even with temporal shifts. – Can handle multi-feature sequences through feature-wise DTW matrices or combined metrics. – Focuses on the trajectory's overall shape rather than absolute timing.	– Computationally expensive for large datasets. – Penalizes amplitude differences even when physically irrelevant, in turn requiring tuning of feature weighting and normalization across sequences.	$\mathcal{O}(M^2N^2)$
CDTW	Cyclic Dynamic Time Warping aligns sequences under all possible cyclic rotations to account for phase shifts in periodic motion.	– Phase-invariant comparisons, DTW metric not influenced by phase shift. – Preserves shape-based similarity while allowing temporal misalignment and starting-point variation.	– Naive implementation requires evaluating all rotations, increasing computational cost by another order of magnitude. – Assumes underlying periodicity, may misrepresent non-cyclic movements. All DTW drawbacks still present.	$\mathcal{O}(M^2N^3)$, reducible to $\mathcal{O}(M^2N^2 + N \log N)$ with FFT-based shift estimation CITE CYCLIC DTW

Dynamic Time Warping (DTW) is selected for its ability to measure similarity between time series based on their overall shape, allowing sequences that are out of phase or evolve at different temporal rates to be accurately aligned and compared. However, DTW is inherently sensitive to amplitude variations between sequences. To mitigate this, an additional normalization step is applied after transforming the data back to physical units. Each sample is proportionally scaled to the [0,1] range independently for position and velocity, based on the maximum Euclidean norm of all time points within that sample per position/velocity. All remaining points are scaled relative to this maximum, ensuring comparable trajectory amplitudes and allowing DTW to emphasize shape similarity rather than magnitude. This normalization is relevant for family clustering, whereas it is unnecessary for MSDC verification, since amplitude variations are small and MSDC converges to a physically similar trajectory regardless. Phase alignment is enforced during data pre-processing to eliminate the need for CDTW, as detailed in Section 7.1.

6.4. Requirement Verification

Orbits investigated within the CR3BP are employed in preliminary mission design prior to transition to an EHF model. Consequently, the accuracy requirements of the CR3BP formulation arise from the level of resolution needed to ensure a precise and efficient transition between the CR3BP and EHF representations. In practice, this implies that analyses performed within the CR3BP must achieve sufficient precision such that optimal transition rates in the converged EHF solution can be guaranteed. Should EHF convergence not be obtainable, optimal convergence to the CR3BP should be guaranteed. The accuracy requirements ($\Delta p < 10^{-5}[LU], \Delta v < 10^{-5}[LU/TU]$) within the CR3BP are therefore verified in the following section, as the overall accuracy of the computed or sampled trajectory compared to numerical benchmarks.

The second set of accuracy requirements, referred to as τ, specifies the acceptable level of discontinuity between propagated segments within the trajectory. Ideally, this inter-segment discrepancy should approach the order of numerical round-off or truncation errors in both the CR3BP and EHF models. However, because an additional correction step occurs during the transition from the CR3BP to the EHF formulation, such precision is not always required in the CR3BP domain, particularly when a large number of segments are employed, also in order to facilitate numerical stability. This discontinuity can be interpreted as the inherent noise level between CR3BP segments. These continuity accuracy

requirements, detailed in Section 7.6.1, provide a means of determining whether a noisy CR3BP trajectory must first be converted into a dynamically consistent CR3BP orbit before being transferred to the EHF model, or whether the noise levels in the CR3BP will result in an acceptable convergence rate to the EHF model.

6.4.1. Trajectory Accuracy Requirements Verification

To assess this accuracy requirement, the state difference is evaluated between CR3BP periodic orbits and their EHF-based counterparts for the orbit families included in the dataset. This is done by applying MSDC correction schemes for the $L_{2,S}$ Halo family, identified in Section 3.3 as yielding the lowest convergence rate, in order to provide a worse case-scenario estimate. This is done through the use of SEMpy [35]. Note that many families are unable to converge reliably beyond three revolutions, however, the associated state-difference error is expected to remain on the same orders of magnitude over an extended number of periods [40]. This process is illustrated for a randomly sampled $L_{2,S}$ Halo orbit propagated over nine revolutions in Figure 6.9, Figure 6.10, and Figure 6.11. Figure 6.9 presents the transition from the CR3BP (blue) to the EHF model (red), presented in both three-dimensional space (left) and an xy-plane projection (right). The pronounced variations in the x and y directions, particularly evident in the planar projection(right), arise from the dominant influence of solar perturbations within the ecliptic plane.

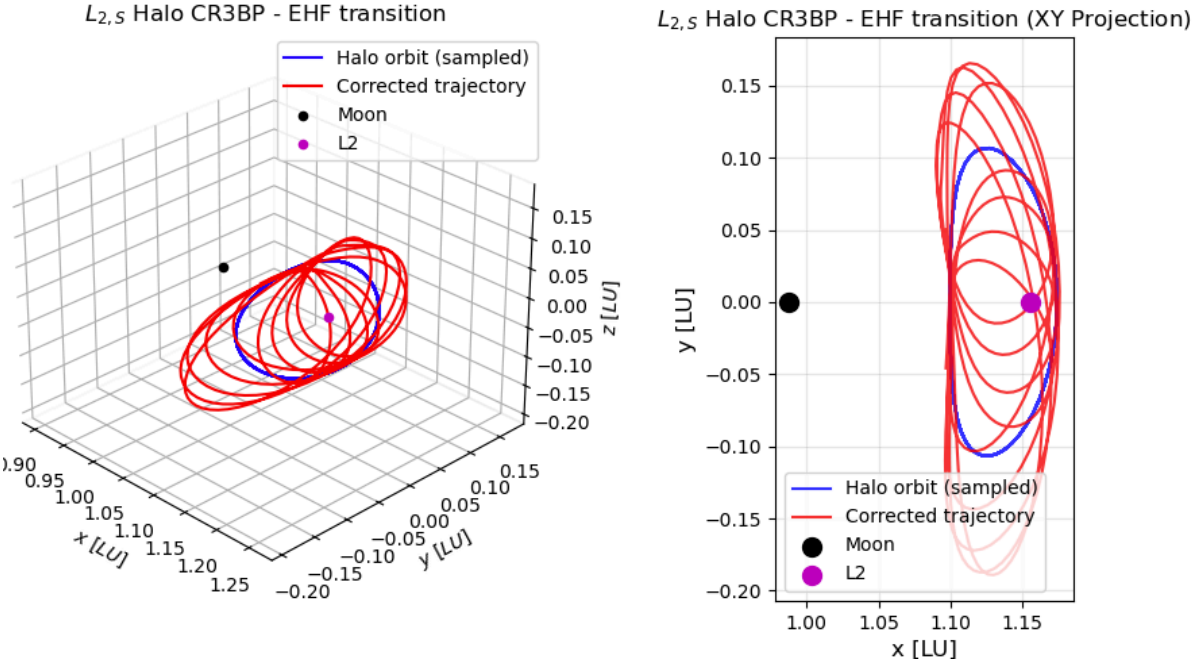


Figure 6.9: Transition of $L_{2,S}$ Halo orbit from CR3BP to EHF. Left: 3D spatial phase space representation. Right: xy projection.

Figure 6.10 presents the magnitude of each component over nine revolutions, for both the original CR3BP (blue) case, and the EHF (red) case. This figure confirms that the largest relative magnitude of the difference between the CR3BP and EHF orbit is dominated by the y (left, middle) and v_x (right, top) components.

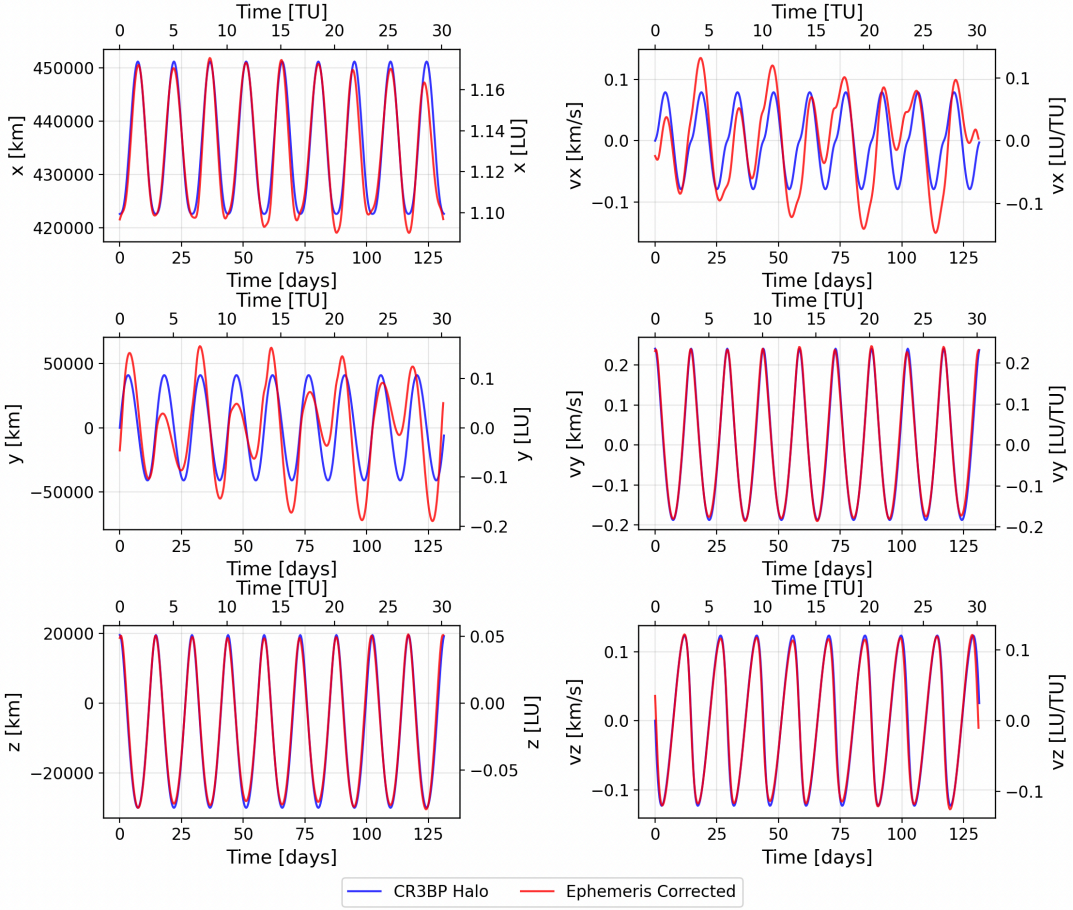


Figure 6.10: State components of CR3BP and EHF transitioned $L_{2,S}$ Halo orbit over 10 revolutions

Figure 6.11 presents the relative magnitude of this error for both position (Δr) and velocity (Δv). Note that the Δr stated exceeds expected values, reaching a peak $|\Delta r|$ of $\approx 1.25 \cdot 10^{-2}[LU]$, and $\approx 1 \cdot 10^{-2}[LU/TU]$ for $|\Delta v|$. This contradicts the literature presented in Table 5.1, with positional error measured on 1-1.5 orders of magnitude more then limits stated for $L_{2,N/S}$ Halo orbits (1000 [km]). Future relaxation of these accuracy requirements could be implemented such that the conversion rate between CR3BP and EHF are maximized. Nonetheless the 10^{-5} stated accuracy remains a viable target in the EHF frame itself, as this remains dictated by current orbit determination capabilities in the cislunar system.

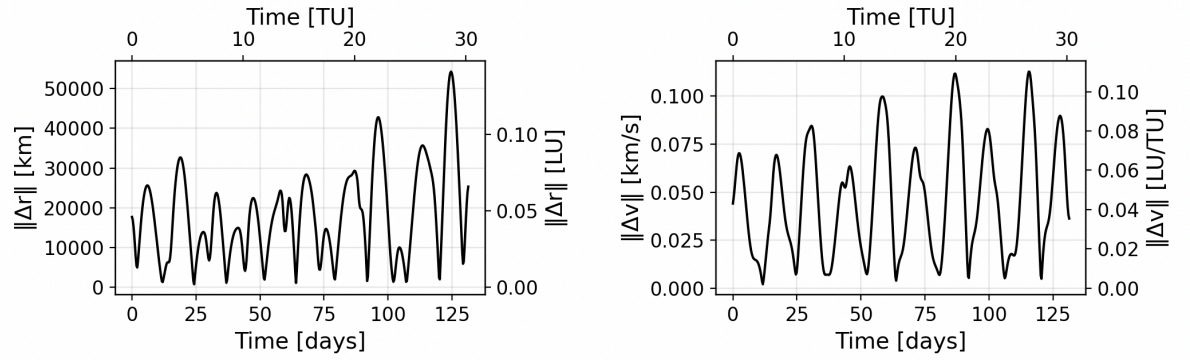


Figure 6.11: Magnitude of transition error between position (Δr) and velocity (Δv) for a $L_{2,S}$ Halo orbit transitioning between the CR3BP and EHF

6.4.2. Dataset Continuity Verification

The JPL dataset is structured as pairs of initial conditions and their corresponding orbital periods. This analysis evaluates the continuity properties of these datasets, specifically assessing whether the original conditions satisfy the desired periodicity requirements at the levels of 10^{-8} [-], 10^{-10} [-], and 10^{-12} [-]. This is done for the purpose of verifying the dataset accuracy to be within the $10^{-10}[-]$ value selected in literature, and sensitivity to this continuity tolerance.

To perform this assessment, each initial condition is numerically propagated over one orbital period using the propagation settings defined in the numerical integration framework. The deficit is then defined as the state vector difference between the initial and final conditions, $\epsilon = \mathbf{X}_0 - \mathbf{X}_P < tol$, where $\mathbf{X}_0$ denotes the initial state and $\mathbf{X}_P$ the state after one period. Subsequently, the SSDC scheme described in Section 2.10 is applied to iteratively refine the initial conditions and reduce the continuity deficit to within the specified tolerance levels. This process is repeated until the correction magnitude falls below the prescribed tolerance or the maximum iteration limit is reached, with the limit of SSDC iterations set to 10.

The results for the JPL database are presented in Figure 6.12, for tolerances levels of 10^{-8} [-], 10^{-10} [-] and 10^{-12} [-] (top, middle, bottom subplots), before and after 10 SSDC iterations respectively (top, bottom row per subplot). For the tolerance level commonly adopted in the literature (10^{-10}, dashed line, central subplot), the selected integration scheme yields convergence for approximately 90.5% (green, top row, central subplot) of the database. After applying SSDC correction, this proportion increases to 97% (green, bottom row, central subplot). This threshold is considered acceptable for the purposes of this investigation, as it allows excluding the remaining orbits that do not satisfy the tolerance criterion without significantly affecting the overall data quantity or distribution. In contrast, tightening the tolerance to 10^{-12} (dashed line, bottom subplot) results in only 32.4% (green, top row, bottom subplot) of the dataset meeting the requirement, while relaxing it to 10^{-8} (dashed line, top subplot) renders the criterion too permissive, with nearly all data (98.1%, green, top row, top subplot) satisfying the threshold.

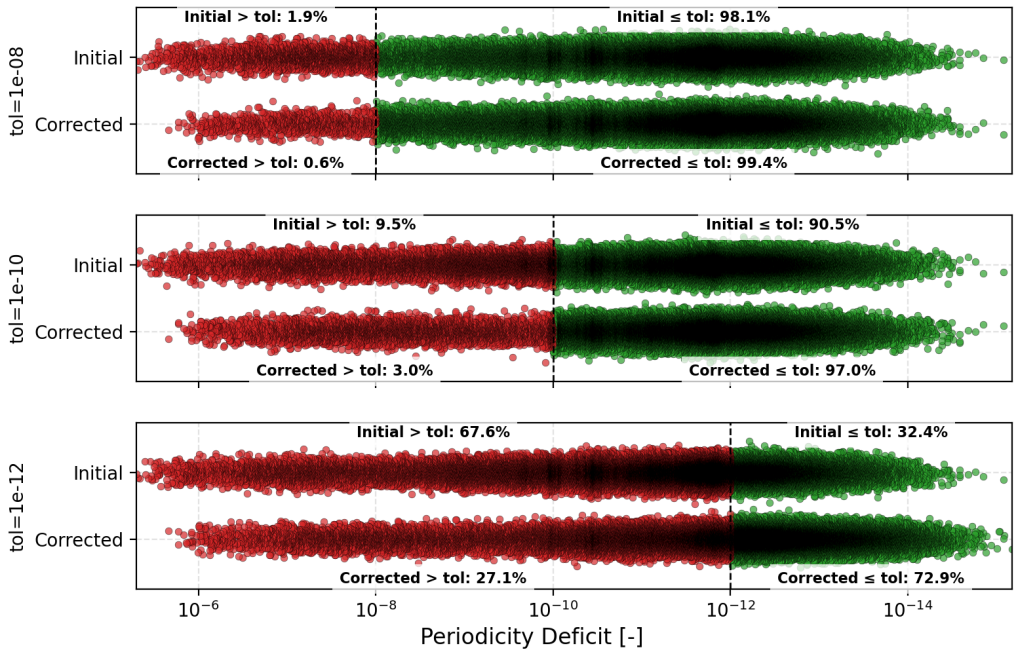


Figure 6.12: JPL Database: Continuity Deficit Pre-/Post 10 SSDC iterations.

The majority of orbits that failed to satisfy the periodicity constraint are concentrated within three families: the L_2 Lyapunov, Distant Prograde Orbit (DPO), and Low Prograde Orbit (LPO) classes, which exhibit post-SSDC failure rates of approximately 42%, 34.5%, and 28.4%, respectively. This behaviour is illustrated in Figure 6.13, which reports per-class satisfaction of the periodicity tolerance of 10^{-10} [–], distinguishing orbits that satisfy the constraint initially (green), only after correction (blue), and those

that fail to converge (red). All of these families are planar, suggesting that numerical instabilities in the more unstable members of such families more readily lead to violations of the continuity requirement.

This observation is further supported by Figure 6.14, which presents the same periodicity outcomes—initially satisfied (green), satisfied only after correction (blue), and failed to converge (red)—categorized according to stability class. Unstable orbits exhibit the highest likelihood of violating the continuity constraint, with approximately 7.5% failing to converge. Similarly, Figure 6.15 reports the same classification organized by orbital period. A pronounced increase in non-convergent cases (red) is observed for orbits with periods between 7 and 10 [TU], indicating that this regime is particularly susceptible to discontinuities.

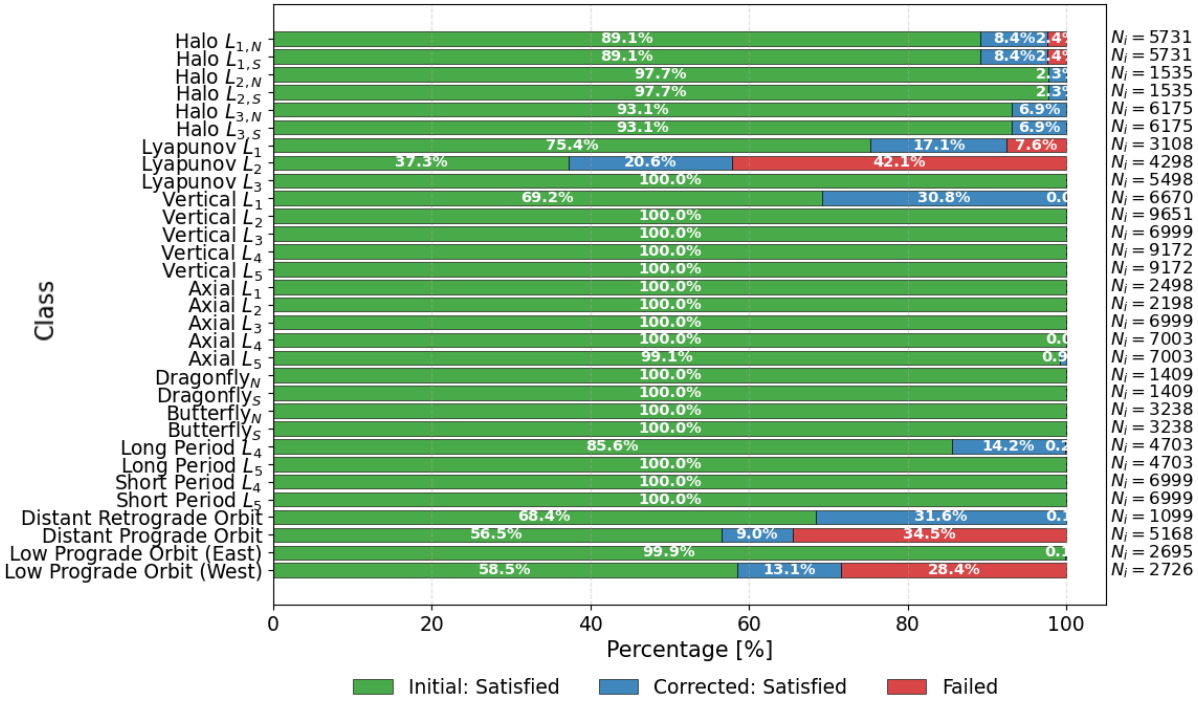


Figure 6.13: Periodicity satisfaction by class for JPL Dataset: horizontal stacked bars show the fraction of orbits that were initially periodic (green), became periodic only after SSDC (blue), and remained non-periodic (red). N_i reports per class counts.

The JPL database does not provide explicit continuity values. Assuming the source data are of high quality, the observed residuals are therefore attributed to sensitivities inherent to the dynamical propagation regime. To ensure consistency within the adopted numerical propagation framework, trajectories that fail to meet the prescribed tolerance are discarded, thereby enforcing the continuity requirement of $\tau_{\text{SSDC}} \leq 10^{-10}$ [–] within the applied scheme. Consequently, all orbits that violate this constraint (indicated in red in the preceding figures) are excluded from subsequent analyses.

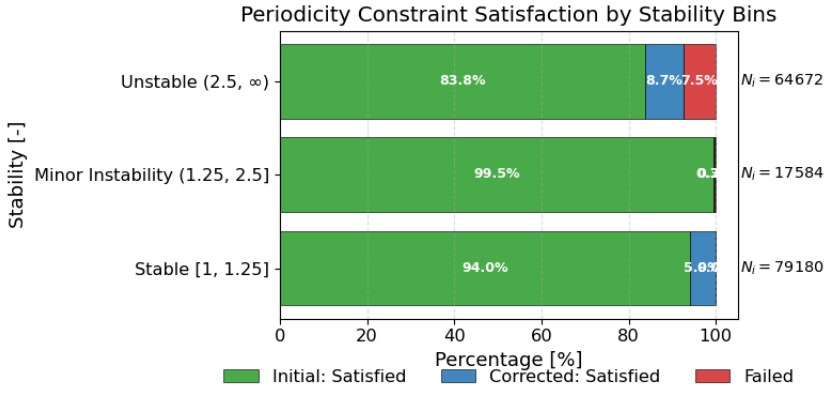


Figure 6.14: Periodicity satisfaction by stability category: Stable [1, 1.25], Minor Instability (1.25, 2.5], Unstable (2.5, ∞). Stacks indicate initial (green), corrected-only (blue), and failed (red) percentages. N_i report per-class counts.

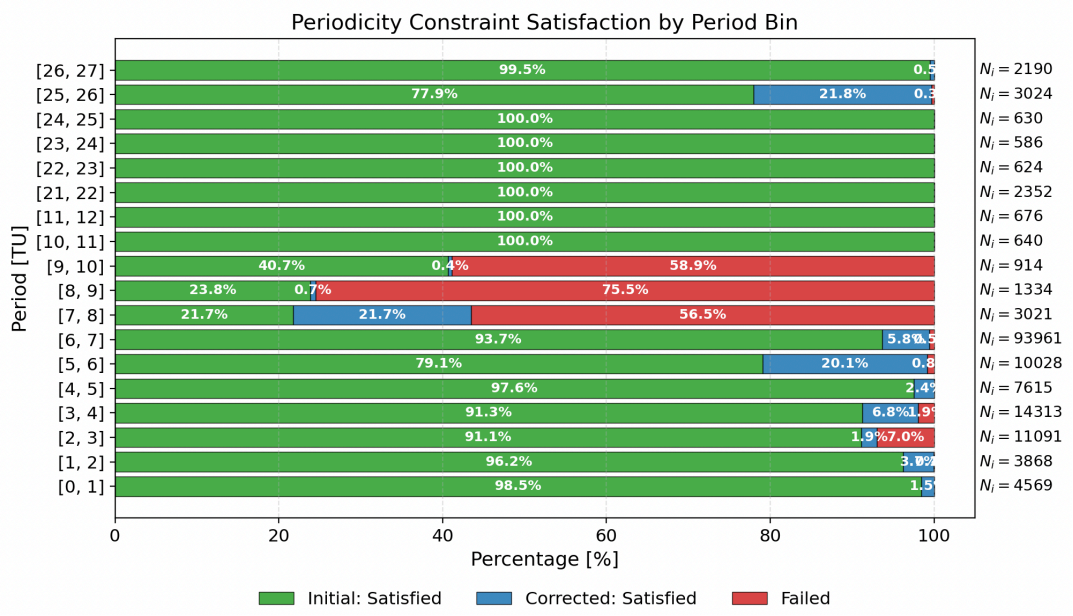


Figure 6.15: Periodicity satisfaction by period bins: horizontal stacks show initial (green), corrected-only (blue), and failed (red) percentages per bin. N_i report per-class counts.

An overview of the final datasets used across the various training scenarios is provided in Table 6.5. A nominal sequence length of $N(\mathbf{X}) = 100$ is selected to balance computational efficiency and storage requirements; however, this choice will be revisited during sensitivity analysis and hyperparameter tuning. More details regarding specific class makeup can be found in Appendix A.

Table 6.5: Summary of available datasets.

Dataset	Class Amount	Size (Count)	Size @ N=100 (.npz MB)
JPL	31	156544	802
JPL-A	9	26700	129.2

6.5. Computational Requirements of Numerical Schemes

To quantify the cost of conventional numerical integration schemes, two scenarios are considered, populating a familial abacus (Section 3.2.1), followed by populating a set of manifold arcs (Section 3.2.2). This is explored across a range of integration schemes presented in Table 6.3. Manifold arc population further serves as a benchmark for accuracy of numerical propagation schemes, in order to assess integrator performance in both chaotic (close interactions with the primaries) and non-chaotic phases.

The benchmarking process consists of two parts. First, the initial states of representative manifold arcs are propagated forward in time to evaluate integrator accuracy and speed. Second, interpolation costs are measured for different abacus resolutions. The total cost of populating a manifold-arc set is then computed as the sum of the propagation and interpolation costs, assuming ad-hoc orbit and arc location iteration during future optimization processes (Section 3.2.2).

6.5.1. Manifold Arc Propagation

The initial states (X_i) of 100 manifold arcs are propagated with and without the STM ($\Phi(t_i, t_0)$) for 100 randomly selected unstable periodic orbits(i.e. only orbits that posses manifolds). Furthermore, it is important to note that the results reflect only the propagating the manifolds from their initial states, computing manifold directions themselves is not included in these evaluations (evaluating Φ, solving M, etc.). Each manifold arc is propagated for 10 [TU], selected due to the majority of optimal (min Δv, min TOF) transfers occurring in this region [34, 41]. Furthermore, transfer to the EHF over periods longer than 10 [TU] becomes more difficult, with transfer rates dropping below 20% [39, 40]. In combination with this, this length of time allows the manifold arcs to have both phases of sensitive dynamics (close

gravitational encounters) and unsensitive dynamics (excursions). A propagation time of 10 [TU] additionally facilitates the divergence of manifold arcs from one another over a range of periodic orbits and associated stability values. This behaviour is illustrated in Figure 6.16, which shows the mean separation (solid line) and associated variance (shaded region) between successive manifold arcs with an initial manifold arc separation of 0.01 [LU] (along the parent periodic orbit), as they diverge over time. The divergence of these arcs over time depends on both the Jacobi constant (color-coded) and the inherent stability of the orbit family. Orbits with higher Jacobi values (yellow) diverge earlier in time than their lower-Jacobi counterparts (purple), but also exhibit more variability in distance between arcs, as indicated by their higher variance (shaded region). Notably, the manifold arcs exhibit a temporary convergence around $t = 4$ [TU], as indicated by a reduction in the separation between successive arcs. This behaviour is driven by their close approach to the Moon, where enhanced gravitational interactions draw the trajectories toward a common region of phase space. Following this encounter, the arcs diverge rapidly, reaching separations on the order of 10^{-1} to 1 [LU] by $t = 10$ [TU].

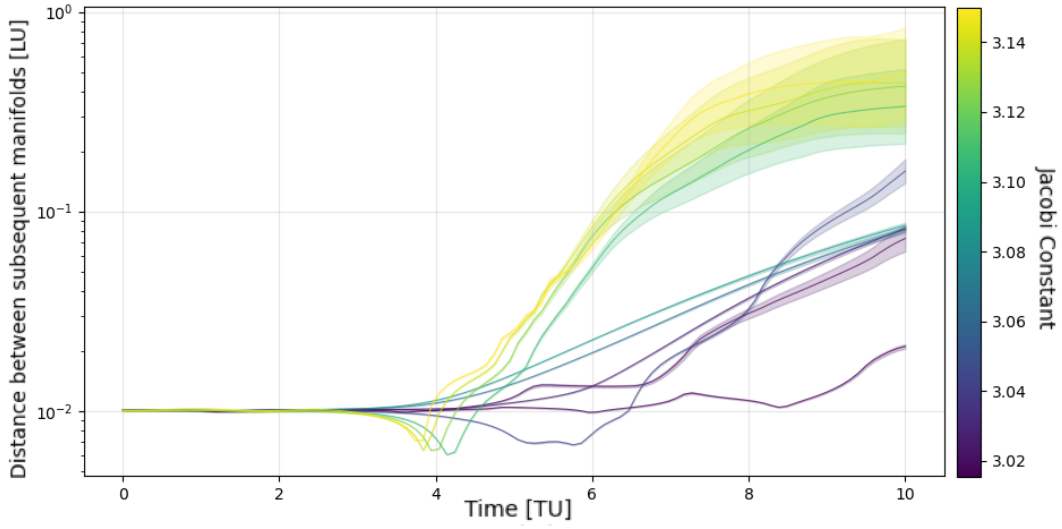


Figure 6.16: Manifold divergence over time for varying $L_{2,S}$ Halo family members.

Integration tolerances (atol, rtol) ranging from 10^{-6} [-] to 10^{-12} [-] are evaluated across the integrators listed in Table 6.3. Typically, the benchmark is set at the tolerance where the combined truncation and round-off error (the total numerical error) is minimized, however for the purpose of this investigation a benchmark is established using the DOP853 integrator (highest-order method considered) at a tolerance of 10^{-13} [-] (tightest tolerance evaluated). This choice reflects the fact that the point at which truncation and round-off errors balance varies across different periodic orbits and manifold arcs. Absolute tolerance and relative tolerance for adaptive integrations schemes are set to the same order of magnitude, with no minimum/maximum time-step set. Position, velocity and Jacobi constant error are selected as evaluations metrics at the end of the propagation to evaluate accuracy and energy conservation requirements. Position and velocity are critical for assessing the accuracy of the trajectory, whilst the Jacobi constant is used to measure energy conservation, as described in Section 5.1.3.

Figure 6.17 presents the the mean position (top), velocity (middle), and Jacobi (bottom) errors, and corresponding accuracy requirements (red lines, dashes are invalid region), for the manifold arc propagation case described above with a propagation time of 10 [TU]. A more detailed version including variances of each measurement is presented in Appendix A in Figure A.17 (position error), Figure A.18 (velocity error), Figure A.19 (Jacobi error). These figures summarize the relative computational efficiency of each numerical integrator, expressed as the number of floating-point operations (FLOPs) required to propagate the manifold arc (x-axis), as well as the resulting propagation accuracy at the final time, measured in terms of mean position, velocity, or Jacobi error. For each numerical integration scheme, tolerance levels from 10^{-6} [-] to 10^{-12} [-] correspond to the data points from left to right, with the associated variance indicating the spread of measurements about the mean. In addition, the average FLOPs per second measured during benchmarking is used to construct a secondary x-axis at the top of each figure, based on single-core performance of an *Apple M4 CPU*.

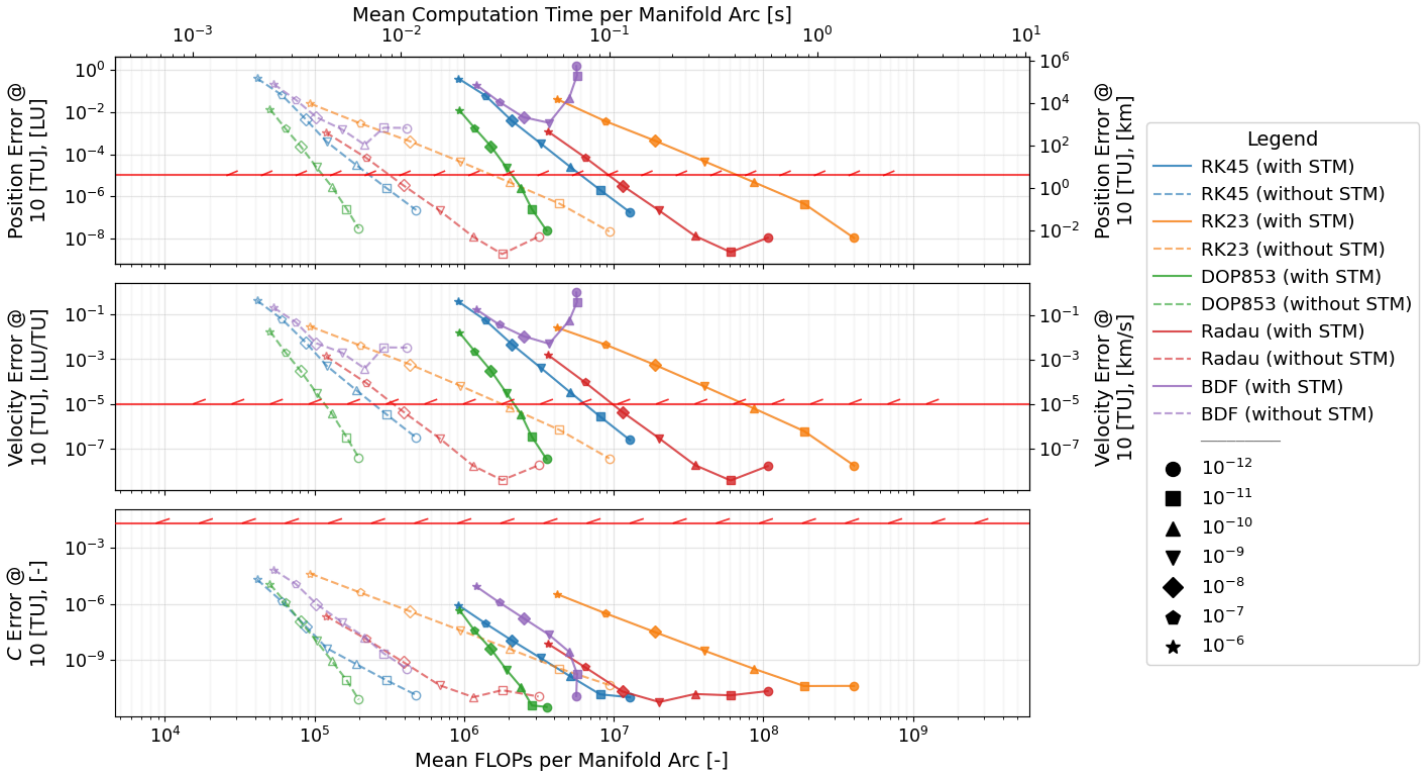
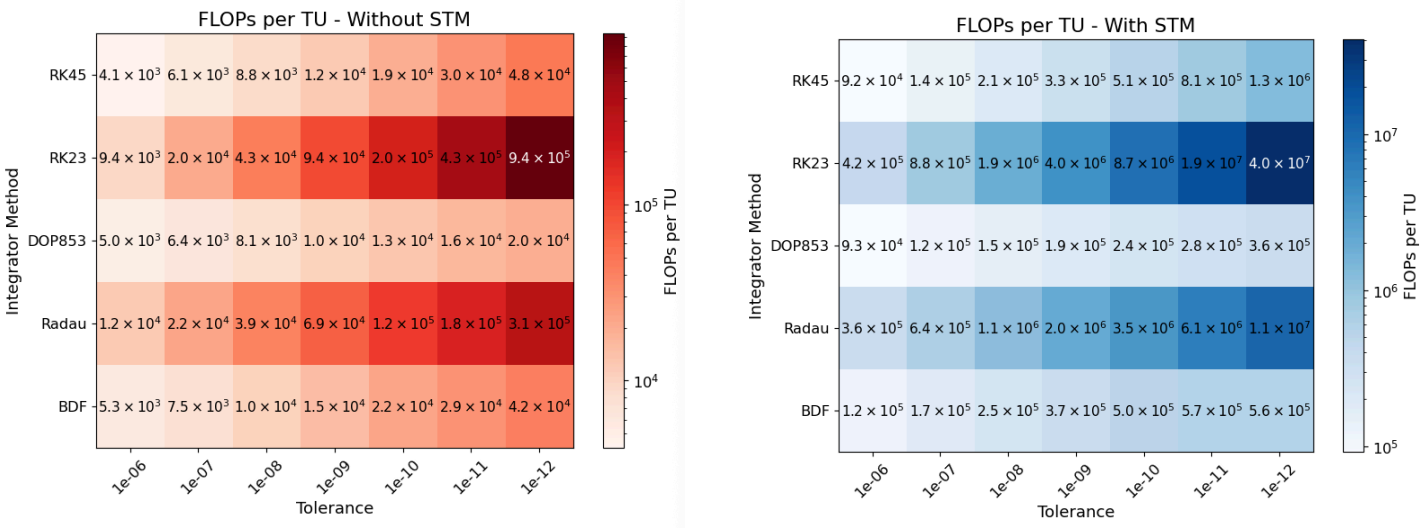


Figure 6.17: Mean position error norm (top), velocity error norm (middle) norm, Jacobi constant error (bottom) at 10 [TU] for various integrators and integration tolerances. Red line denotes the accuracy requirements, $\Delta p \leq 1 \times 10^{-5}$ [LU], $\Delta v \leq 1 \times 10^{-5}$ [LU/TU], $\Delta C \leq 1\%$.

RK45 and DOP853 show high efficiency across tolerances, with DOP853 requiring less FLOPs for a similar accuracy level to RK45. Furthermore, whilst relatively computationally cheap for lower accuracy levels, the computational cost for RK23 grows significantly at tight tolerances. The Radau integrator achieves the highest accuracy levels for position and velocity at the tightest integration tolerances, however at the cost of an order of magnitude more FLOPs when compared to RK45/DOP853. BDF fails to achieve sufficient accuracy levels, namely being restricted to 10^{-3} [LU] for position, and 10^{-3} [LU/TU] for velocity. Position and velocity exhibit similar error and scale patterns as a result of the non-dimensionalization of the system in the synodic reference frame.

Figure 6.18 compares the average number of FLOPs per TU across integration schemes and tolerances, both without (red, Figure 6.18b) and with (blue, Figure 6.18a) STM propagation, with shading indicating the order of magnitude of the computational cost. Including the STM generally does not degrade numerical accuracy across integrators; however, it introduces a substantial increase in computational expense. This overhead is typically on the order of one to two orders of magnitude in FLOPs to achieve comparable accuracy levels and becomes increasingly pronounced across all integration schemes as tighter tolerances are enforced, visible an increase in FLOPs for each rightward shift in the heatmap. Notably, both the RK23 and Radau integrators exhibit a significant computational cost, requiring on the order of 10^5 FLOPs per TU without STM propagation and on the order of 10^7 FLOPs per TU when STM propagation is included in the extreme case (tolerance of 10^{-12}[-]), a difference in computational effort on the order of magnitude of $\approx 10^2$.



(a) FLOPs per TU in numerical integration schemes, without STM.

(b) FLOPs per TU in numerical integration schemes, with STM.

Figure 6.18: Comparison of the average number of FLOPs per TU for integration schemes without and with STM.

The most computationally efficient integrator that still satisfies the required accuracy thresholds is the DOP853 method with a tolerance of 10^{-9} [-], for both the non-STM (Figure 6.18a) and STM (Figure 6.18b) propagation cases. As accuracy will inevitably degrade by a few orders of magnitude during the regression tasks, training data itself will be generated by the highest-accuracy configuration to ensure a robust foundation for all learning models. Prioritizing dataset reliability enables fair and meaningful comparison against alternative, less computationally demanding approaches explored in later stages. Furthermore, the state errors dominate the overall error landscape, as the Jacobi accuracy requirement in Figure A.19 (below dashed red line) is satisfied for all numerical propagation settings. Consequently, it can be concluded that the spatial and dynamic state errors are the primary limiting factors, and the Jacobi error requirement can be discontinued.

6.5.2. Familial Interpolation Benchmarking and Sensitivity Analysis

Adopting SEMPy nomenclature, an abacus is defined as a parameterized list of initial conditions representing a specific dynamical family, from which additional family members exhibiting desired mission characteristics can be interpolated. An abacus is generated by applying either NPV or PAL familial continuation methods, which both yield a continuous mapping of orbit families, as detailed in Section 2.11.1 and Section 2.11.2. To assess the computational and storage efficiency of interpolating and propagating initial conditions from an abacus stored at varying resolutions, relative to in-family latent sampling enabled by the generative models introduced in Chapter 7, this section presents a comparative analysis of the two approaches. This section assumes an abacus produced through familial continuation is available for interpolation within a family. The influence of the interpolator degree (outlined in Section 2.11.3) and the continuity tolerance is examined separately. To facilitate this analysis, the required number of FLOPs and the associated SSDC iterations needed to obtain a converged orbit, in combination with the convergence rate within 10 iterations are evaluated for various interpolator degrees (linear, quadratic, and cubic) and periodicity-continuity requirements, as a function of the abacus resolution, is explored. As previously stated, the behaviour in this section is presented for the DOP853 integrator with a tolerance of 10^{-13}, and no imposed maximum timestep, applying a convergence criterion of $\tau_{SSDC} \leq 10^{-10}$ in the scheme (i.e., $|\mathbf{X}_0 - \mathbf{X}_P| \leq \epsilon = 10^{-12}$ [-]). This configuration is restated here to emphasize that the corresponding results obtained with alternative integrators are retained for analysis in the subsequent subsections (Section 6.5.2.1, Section 6.5.2.2). These other integrators (as identified in Table 6.3) exhibit similar behaviour, with only minor variations in the number of FLOPs and mean iteration counts.

The resolution is defined by the difference in the Jacobi constant (ΔC) between the initial conditions of consecutive stored family members. This metric provides insight into the level of detail at which the abacus should be sampled during the familial continuation process to ensure accurate interpolation,

efficient convergence, and efficient storage. The meaning of this resolution is better understood through Figure 6.19. In this example, the data stored in the abacus is sparse, with a mean $\Delta C \approx 0.05$ (horizontal spacing between red points). Initial estimates obtained using the interpolators (linear: blue cross; quadratic: orange square; cubic: green diamond) are used as starting points for SSDC to converge toward the reference dataset (grey line). While interpolation may succeed for $C \approx 3.085$ and $C \approx 3.130$, owing to the close agreement between the estimated values (markers) and the reference dataset (grey line), convergence fails near $C \approx 3.035$ due to the degraded quality of the initial guess (markers), for which the deviation from the true solution (grey line) is too large.

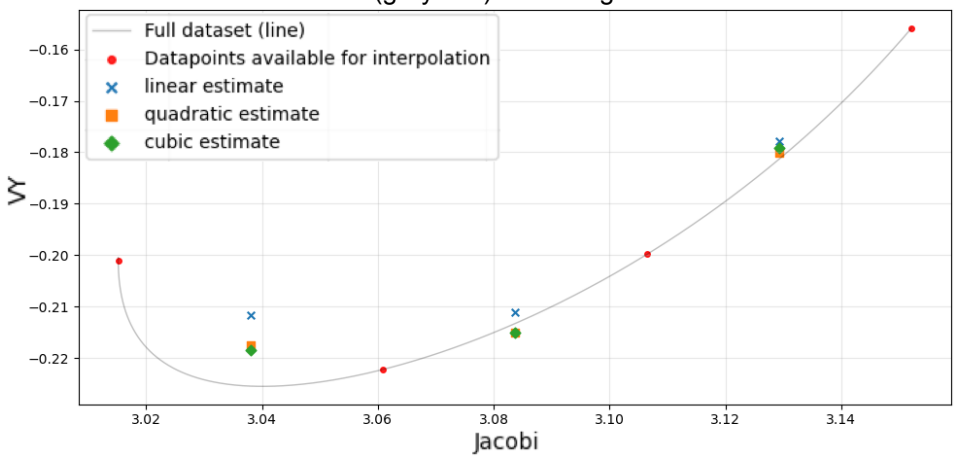


Figure 6.19: $L_{2,N/S}$ Halo family interpolation with $\Delta C \approx 0.05$. Initial guesses for linear, quadratic and cubic interpolators marked (VY).

6.5.2.1. Interpolator Degree

The influence of abacus resolution on interpolator performance is shown in Figure 6.20. The left sub-figure reports the mean number of FLOPs required for convergence across a range of abacus resolutions, shown separately for each interpolator degree (colour-coded), together with the corresponding number of SSDC iterations. These statistics are computed using up to 50 samples per resolution (note that lower resolutions do not permit the full sample count), across nine distinct resolutions, after which the mean value is reported for each resolution. The right sub-figure similarly presents the convergence rate achieved within ten SSDC iterations.

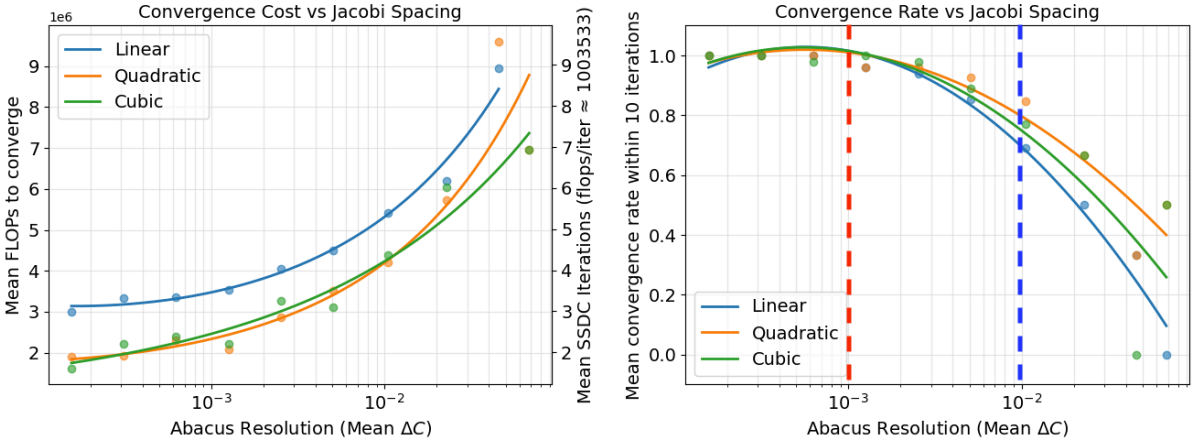


Figure 6.20: Interpolation convergence cost and convergence rate of SSDC scheme for varying abacus resolutions and interpolator degrees.

In both cases, a second-degree polynomial fitted in logarithmic space provides the best representation of the observed trends, yielding R^2 coefficients of > 0.95 for convergence cost (left) and > 0.88 for convergence rate (right). The resolution at which the abacus is stored exhibits a direct correlation with the number of FLOPs and iterations required for convergence. Figure 6.20 indicates that the abacus resolution should not exceed $\Delta C \approx 10^{-3}$, as the SSDC convergence rate begins to degrade beyond this threshold (right of red dashed line). For $\Delta C \gtrsim 10^{-2}$ (right of blue dashed line), a pronounced re-

duction in convergence rate is observed across all interpolation schemes. While the relative decrease in convergence rate is clearly evident for the linear interpolator (blue, solid), the performance difference between quadratic (orange, solid) and cubic (green, solid) interpolators is negligible. It should be noted that the computational cost associated with parsing the abacus is not considered in this analysis, becoming prohibitive for extremely high-resolution datasets [28].

6.5.2.2. Continuity Tolerance Requirement

The associated computational cost and convergence rate are also dependent on the specified continuity tolerance requirements within the SSDC. The sensitivity of the convergence rate to the selected continuity tolerance requirement is therefore evaluated for an initial guess provided by a cubic interpolator, using the same propagation scheme as in the previous experiment. A range of requirement tolerance levels are tested for multiple abacus resolutions, and heatmaps are generated to visualize both the average number of iterations required for convergence and the convergence rate within ten SSDC iterations, with results presented in Figure 6.21. Similarly to Figure 6.20, a second-degree polynomial per tolerance curved (solid, coloured) is fitted in logarithmic space.

As shown in the bottom-right panel, the convergence rate within ten iterations is relatively insensitive to the periodicity tolerance; instead, the dominant increases in computational cost are driven by changes in abacus resolution, visible as left–right shifts across the heatmap columns. Across all tolerance levels, the convergence rate falls below 0.95 for resolutions of $\Delta C \geq 5 \times 10^{-3}$ and drops to zero for resolutions exceeding $\Delta C \geq 5 \times 10^{-2}$. This behavior indicates that the abacus should not be stored at resolutions coarser than $\Delta C = 5 \times 10^{-3}$if reliable data retention and convergence performance are required. In contrast, increasing the periodicity tolerance leads to a noticeable rise in the mean number of iterations required for convergence, as shown in the bottom-left panel by the increased sensitivity to shifts along the periodicity-tolerance axis (top–bottom). In this case, both higher periodicity tolerances and coarser abacus resolutions result in a larger number of SSDC iterations required to achieve convergence.

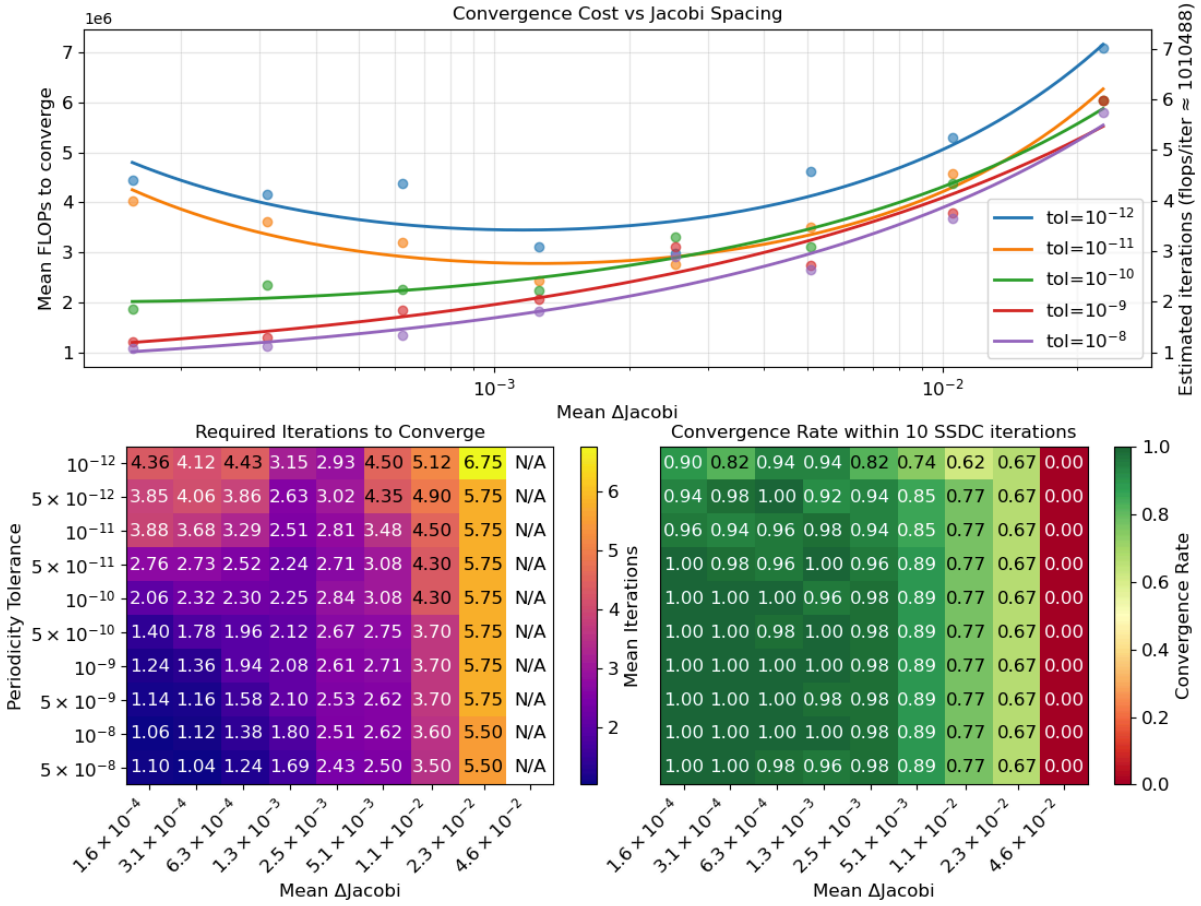


Figure 6.21: Convergence cost and rate as a function of abacus resolution (ΔC) and periodicity tolerance.

As this is not performed on the filtered dataset explored in Section 7.6.1, artifacts of the data verification process are visible. Particularly in the top plot for tolerance levels $\geq 10^{-10}$ at finer Jacobi resolutions ($\Delta C \leq 10^{-3}$). At these levels, a smaller proportion of the original data points satisfy the specified tolerance threshold, resulting in additional computational cost to refine the dataset to the required accuracy. The most efficient abacus resolution for an acceptable convergence rate (> 0.95) across all periodicity tolerance levels, is $\Delta C = 3 \cdot 10^{-3}$. Increasing resolution past this point solves only to reduce the computational burden of converging to a tolerance, demonstrated by the bottom left and top subplot through a reduction in mean iteration amount. Furthermore, a relaxation in the periodicity tolerance constraint will naturally result in a quicker convergence rate.

6.5.3. Manifold Arc Set Population

Combining the information presented in Section 6.5.1 and Section 6.5.2 allows the computational effort for the optimization process described in Section 3.2 to be quantified. This is demonstrated for the DOP853 integrator with tolerance 10^{-13}, with no minimum required time-step, in combination with a cubic interpolator for the abacus, based on the methods described in Section 6.5 and Section 6.5.2. The computational burden of both interpolation of family members and and propagation of manifold arcs in time (without STM) is presented in Figure 6.22, benchmarked on a single core of an *Apple M4 CPU*. Allowable regions as defined by the requirements (T < 10 [TU], $\Delta C < 0.003$) are hatched, where the intersection of these two hatching lines correspond to the nominal scenario considered (Abacus resolution $\Delta C = 0.003$, arc propagation time 10 [TU]).

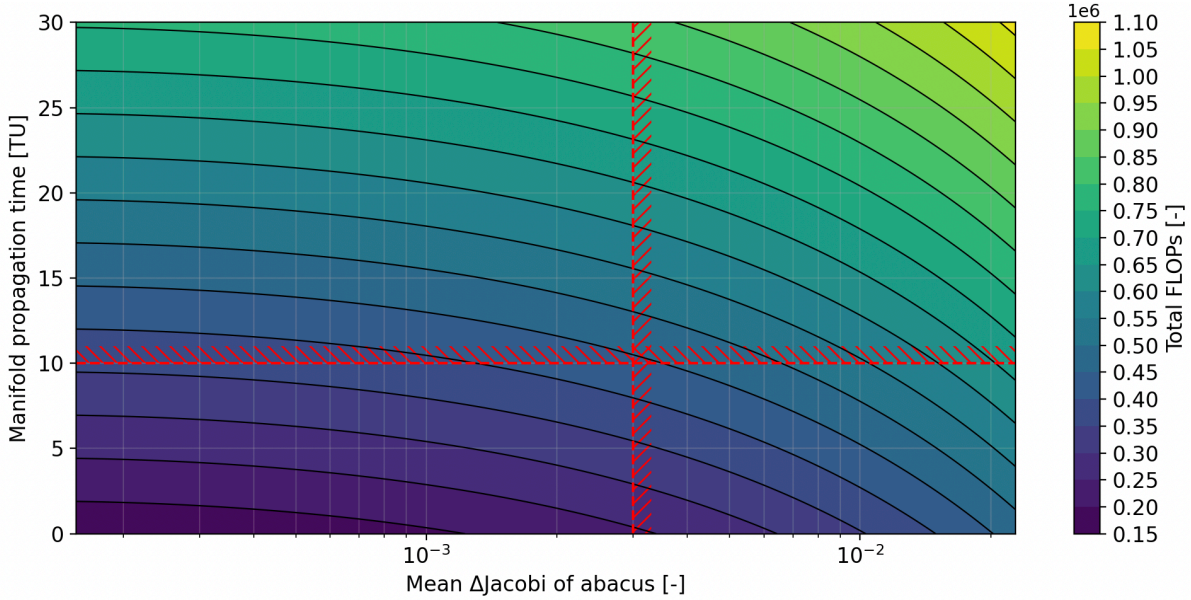


Figure 6.22: Computational cost in FLOPs for interpolation of a parent orbit and propagation of a manifold arc, DOP853 (tol 10^{-13}).

The primary insights from analysis of Figure 6.22, the sensitivity of the computational cost to abacus resolution and arc propagation time, is that the computational cost is primarily derived as a result of manifold arc propagation for higher abacus resolutions, yet for lower abacus resolutions (Higher ΔC) this becomes dominated by the interpolation and SSDC scheme. This is reflected by the contour structure, as for small ΔC, the equidistant contours are nearly parallel to the x-axis, indicating negligible dependence on ΔC. As ΔC decreases and resolution increases, the contours tilt, revealing a stronger sensitivity of the overall cost to ΔC.

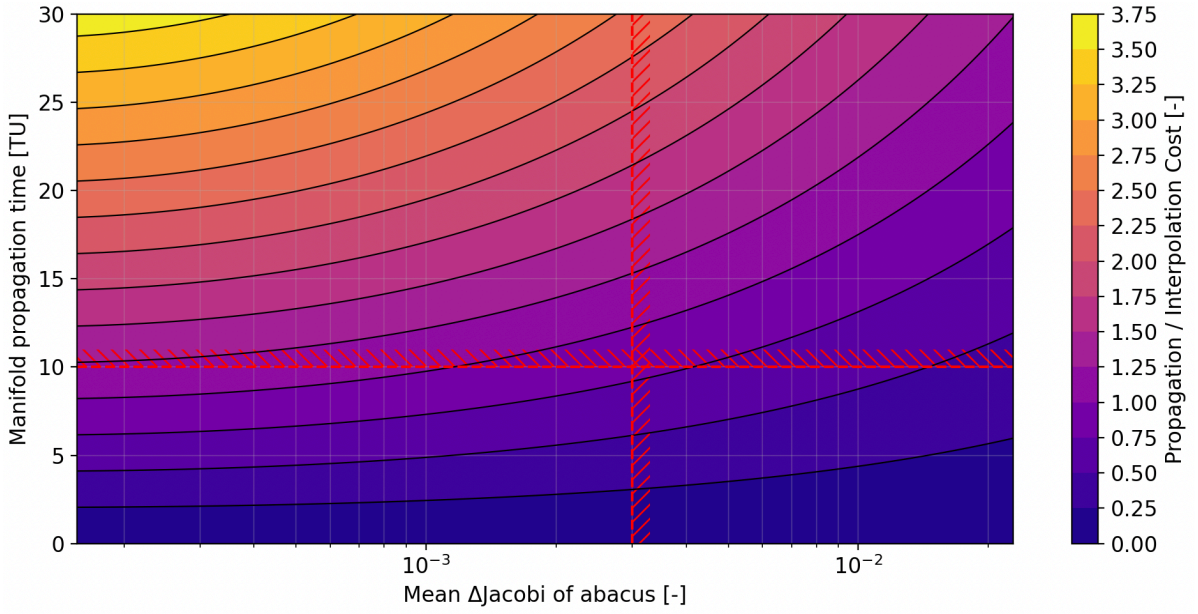


Figure 6.23: Ratio of computational cost between initial SSDC interpolation and Manifold arc propagation, DOP853 (tol 10^{-13}).

Figure 6.23 illustrates the ratio between interpolation and propagation costs. In the nominal case, the total computational expense is approximately evenly divided between these components, as indicated by a ratio of ≈ 1 at the intersection of the hatched lines. Within the viable region, however, most queries are dominated by interpolation, reflected by a propagation-to-interpolation cost ratio < 1 (colour-coded contours). Arc propagation contributes a comparable computational cost only at larger propagation times (i.e., T > 7.5 [TU]), where the ratio approaches 1. Consequently, the total cost of manifold-arc–set population (as described in Section 3.2.2) during the optimization process remains dependent on both the interpolation cost within the family and the cost of propagating the manifold arcs. In order for the neural surrogates explored in Section 7.7 to remain economically viable, they must be able to achieve similar accuracy performance, within this nominal computational budget explored in Figure 6.22.

7

Periodic Orbit Generative Schemes

The discovery of periodic orbits for use in spacecraft mission design in the CR3BP typically relies on grid searches for initial family seeding, and iterative differential correction for familial continuation, see Section 6.1. In this chapter, the use of generative neural models are investigated for the purpose of generating periodic orbits in the CR3BP. As explained in Section 4.3.1 and Section 4.5.3, VAEs are a class of generative models that learn to encode input data into a latent space, from which new samples are drawn by sampling from an assumed prior distribution and decoding back to the original phase domain. When trained on periodic orbit datasets, VAEs are tested for their the potential to capture the underlying structure of families of solutions to enable both generating new members within known families, and extrapolating to new families. Figure 7.1 presents the pipeline used for data preprocessing, training, inference, and, when required, post-processing if additional accuracy refinement is necessary, in accordance with the model requirements defined in Section 5.1.3.

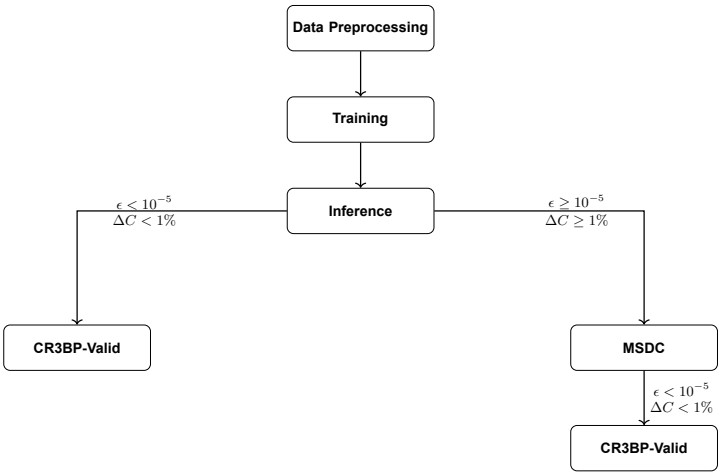


Figure 7.1: Overview of the Data Processing, Training, and Inference Pipeline.

7.1. Data Pre-Processing & Normalization

For all architectures considered in this work, the input data is of shape $[N \times 7]$ where N is the sequence length, selected to be 100 [75]. The dimensionality of seven channels arises from the six distinct features present in the initial state vector $\mathbf{X}_0$, in combination with the time channel required for eventual convergence in multiple shooting schemes. Since the dataset is synthetically generated, N can be adjusted ad-hoc, however each sequence always spans exactly one complete periodic orbit, and all time-points are equidistant over that period. As elaborated upon in Section 6.1, two datasets are explored, namely JPL and JPL-A, in order to determine the impact of class amount and periodic orbit quantity on model performance.

Additionally, as described in Section 6.3.1, data preprocessing is applied to avoid the need for CDTW by enforcing phase alignment among trajectories. This is accomplished by selecting a Poincaré section, specifically the XZ-plane, with a positive v_y crossing as the reference alignment point. To address the issue of multiple XZ intersections in families arising from period-doubling bifurcations, the intersection farthest from the barycenter is selected to ensure a consistent and unambiguous initial phase reference point across all orbits.

The time-series data (shape $M \times N \times F$, where $F = 7 \in \mathbf{X_t} = [t, x, y, z, v_x, v_y, v_z]$) is further normalized using dataset-level feature scaling and a reproducible train/test split. Here, M denotes the number of samples, N the sequence length, and F the number of features. A MinMaxScaler is applied per feature across all samples and timesteps in the training data to map all data values to [0,1]. The per feature minima and maxima of the training data are persisted and reused to scale both the validation data, and to inverse-transform all model outputs back to physical units. Nominally, a 80/20 train/test split is used, however this is parameter is tuned during hyperparameter optimization.

7.2. Neural Architectures and Loss Landscapes

To assess the impact of the architectural choices on the reconstruction and latent representation of periodic orbital time-series data, three VAE-based architectures are evaluated in this work: a fully connected Dense-VAE, a temporally-aware Convolutional-VAE, and the TimeVAE. These models represent increasing levels of structural assumptions about the underlying temporal dynamics, ranging from generic latent (Dense) compression to explicit modelling of temporal and periodic structure (TimeVAE). The Dense-VAE serves as a baseline without explicit temporal assumptions, the Convolutional-VAE exploits local temporal correlations through strided convolutions, and the TimeVAE further decomposes the reconstructed signal into explicitly assumed periodic components, tailored to the periodic time series data. Detailed background and architectural descriptions of each model are presented in Section 4.5.3.

All architectures are evaluated using the nominal VAE loss function (ELBO), extended with a weighted term as in the standard β-VAE formulation, as presented in Equation 4.4 of Section 4.3. In addition to this standard β-VAE loss, a physics-informed (PINN) term (L_C) is introduced to enforce energy conservation through the Jacobi constant:

$$L_{\mathrm{C}} = \gamma \frac{1}{N_t} \sum_{i=1}^{N_t} \left(\hat{C}_i - C_i \right)^2 \tag{7.1}$$

Unlike typical PINN formulations which rely on residuals of the governing differential equations, the CR3BP possesses an integral of motion, the Jacobi constant (C), which enables the use of its conservation residual directly. The contribution of this constraint to the total loss is weighted by the hyperparameter γ. To evaluate the Jacobi residual consistently, the normalization procedure must also be considered. Either the network's prediction must be transformed back to the physical state before computing C, or the true state X must be mapped into the normalized space used by the model through the scaler. Specific to the VAE formulation, this PINN term is also designed to penalize in-sequence discrepancies in variations of C, such that each sequence is physically consistent. The resulting combined loss function is presented in its probabilistic form:

$$L_{\theta,\phi} = \underbrace{-\lambda\, \mathbb{E}_{q_\phi(z|\mathbf{X})} \left[\log p_\theta(\mathbf{X}|z) \right]}_{L_{\mathrm{MSE}}} + \underbrace{\beta\, D_{KL}(q_\phi(z|\mathbf{X}) \,\|\, p_\theta(z))}_{L_{\mathrm{KL}}} + \underbrace{\gamma\, \mathbb{E}_{q_\phi(z|\mathbf{X})} \left[\log p_\theta(C|z) \right]}_{L_{\mathrm{C}}} \tag{7.2}$$

For a more detailed explanation of each variable in this formulation, refer to the description of Equation 4.4 in Section 4.3. The loss is expressed in probabilistic form to emphasize the variational nature of the VAE, in which the latent variables are modelled as Gaussian distributions. In this work, this means assuming that the latent representations of periodic orbits and periodic orbit families can be approximated by learnable Gaussian distributions in this latent space. This assumption does not imply that the physical distributions of periodic orbits or orbit families are Gaussian, but that their latent representations can be approximated as such for learning and inference.

7.3. Hyperparameter Optimization

In order to identify suitable architectures, a neural architecture search (NAS) is performed with randomized parameters. To guide this search more efficiently, Bayesian optimization is employed as the underlying hyperparameter selection strategy. Unlike grid or purely random search, Bayesian optimization constructs a probabilistic surrogate model of the objective function, and iteratively selects new architectures based on an acquisition function that balances exploration of the search space with exploitation of promising regions. This approach is selected for the search space because the relationship between hyper-parameters and model performance is highly non-linear and often noisy. By incorporating information from previous trials, Bayesian optimization reduces the number of required evaluations, converges more rapidly toward high-performing architectures, and enables more efficient use of limited computational resources. Further note that the hidden-layer widths are sorted in descending order to form a VAE-style funnel for the decoder, with the encoder architecture reversing this order. Separate NAS's are performed for each architecture variant (Dense, Convolutional,timeVAE), with and without L_C, and for both JPL and JPL-A datasets, totalling 12 explorations. Nomenclature per hyperparameter optimization is introduced in Table 7.1. Details regarding datasets are provided in Table 6.5 in Section 6.4.2.

Table 7.1: Summary of the 12 NAS configurations.

Option Name	Network Type	$\gamma = 0$	Dataset
Dense–NoLC–JPL	Dense	No	JPL
Dense–LC–JPL	Dense	Yes	JPL
Conv–NoLC–JPL	Convolutional	No	JPL
Conv–LC–JPL	Convolutional	Yes	JPL
TimeVAE–NoLC–JPL	TimeVAE	No	JPL
TimeVAE–LC–JPL	TimeVAE	Yes	JPL
Dense–NoLC–JPL-A	Dense	No	JPL-A
Dense–LC–JPL-A	Dense	Yes	JPL-A
Conv–NoLC–JPL-A	Convolutional	No	JPL-A
Conv–LC–JPL-A	Convolutional	Yes	JPL-A
TimeVAE–NoLC–JPL-A	TimeVAE	No	JPL-A
TimeVAE–LC–JPL-A	TimeVAE	Yes	JPL-A

The hyperparameter search space and fixed optimization constants used in the NAS procedure are summarized in Table A.2. This includes both architectural parameters and training-related hyperparameters, along with their respective sampling strategies under Bayesian optimization. Within the hyperparameter optimization, Hyperband-based early termination is used to prune unpromising hyperparameter configurations, resulting in poorly performing runs stopping early. Hyperband is a scheduling algorithm that allocates training resources progressively across a sequence of evaluation stages known as rungs. Each rung corresponds to a specific training budget (e.g., a fixed number of iterations), and all configurations begin at the lowest rung with a small allocation. After evaluation, only a fraction of configurations advance to the next rung, while the remainder are terminated. This fraction is controlled by the parameter η, which governs the aggressiveness of pruning. $\eta = 3$ is used in this setting, allowing a third to be promoted to the next stage at each stage, the other two-thirds are stopped early. This strategy reduces overall computational cost while still exploring the hyperparameter space effectively. This is performed for $n = 100$ trials per experimental combination, totalling 1200 trials.

The objective function is chosen to maximize the accuracy of the state-space predictions, in order to achieve the 10^{-5} state accuracy requirement necessary for a reliable transition from the CR3BP to the EHF model. It is important to note that this optimization objective places less emphasis on learning the underlying latent distribution as governed by the KL divergence, since predictive accuracy is prioritized. All experiments are performed on a *Nvidia T4 Tensor GPU*. Model performance is evaluated using multiple metrics to capture both overall prediction quality and specific physical accuracy requirements, as summarized in Table 7.7. In particular, the average position and velocity errors are computed over each time series and subsequently aggregated to obtain the mean and standard deviation across the test set.

Table 7.2: VAE evaluation metrics

Metric	Description	Logging Frequency
Train loss	Loss on training set	Epoch (Final)
Test loss	Loss on test set	Epoch (Final, Best)
Reconstruction loss (train/test)	Reconstruction component to loss	Epoch (Final, Best)
KL loss (train/test)	KL-divergence component to loss	Epoch (Final, Best)
Jacobi loss (train/test)	PINN component to loss	Epoch (Final, Best)
Position error (train/test)	Mean absolute error in position across time series	Epoch (Final, Best)
Velocity error (train/test)	Mean absolute error in velocity across time series	Epoch (Final, Best)

7.4. Results

As the requirements for position and velocity are on the same order of magnitude, the euclidean accuracy of the state (X) is considered for evaluation across the validation set. Figure 7.2 presents the results of the hyperparameter optimization for each architecture/dataset combination, for all runs that were not pruned. Each box and whisker bar summarizes the distribution of model performance across completed hyperparameter trials, where the central line denotes the median accuracy (orange), the box indicates the interquartile range (IQR), and the whiskers extend to the most extreme non-outlier values, with outliers shown as individual points. Note that all architectures (individual bars) have a similar range in terms of best achievable model accuracy, demonstrated by the minimum state accuracy all being in the range of $\approx 10^{-2}$ [-]. Sensitivity of the architecture types to the diversity of the dataset is however notable, when comparing JPL (31 families) vs JPL-A (9 families), the maximum achievable accuracy drops by $\approx 1/2 \cdot 10^{-2}[-]$ for the larger and more diverse dataset (JPL). A larger variance in model performance is observed when L_C is included, indicating increased sensitivity to hyperparameter initialization in the presence of the physics-informed term. This effect is most pronounced for the Dense–JPL-A implementations. Furthermore, the Dense implementation exhibits more consistent model performance across a wider range of hyperparameter initializations, as indicated by a smaller IQR (box) around the median accuracy (orange). However, it also shows the most extreme cases, evidenced by a larger number of outliers beyond the IQR.

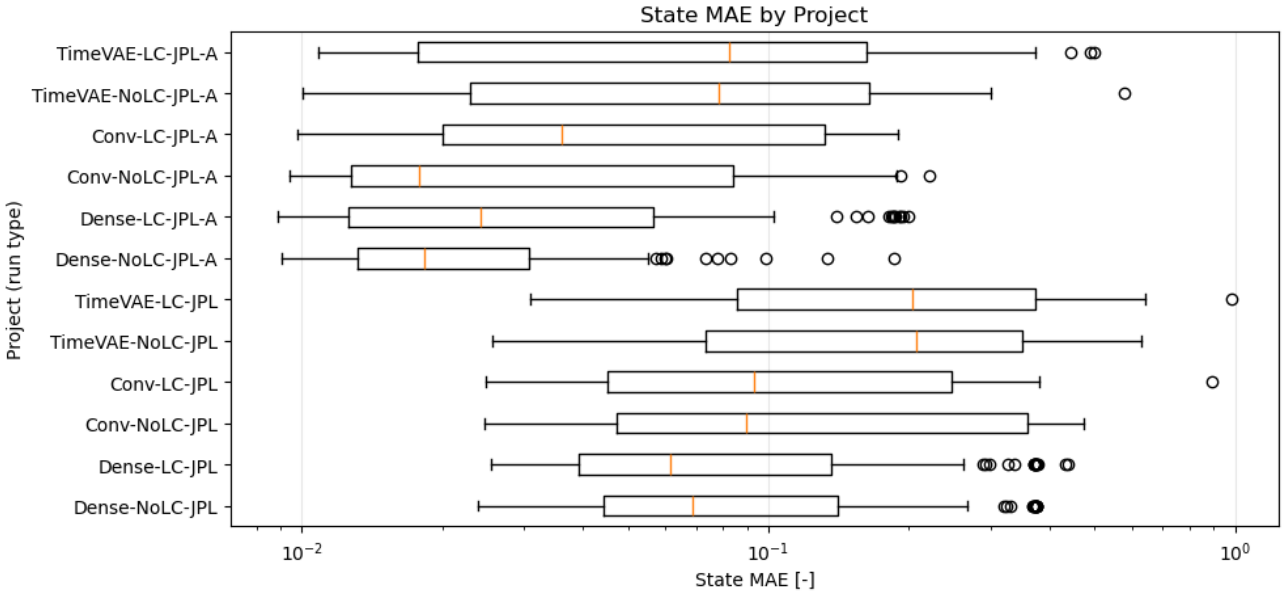


Figure 7.2: VAE Dataset State Reconstruction Error

The sensitivity to computational effort is considered in Figure 7.3 for JPL and JPL-A datasets, through the comparison of the state MAE (vertical axis) to parameter amount (horizontal axis), which is used as

a surrogate metric for computational effort. Parameter count is used as a surrogate for computational effort as it provides a simple architecture independent proxy that correlates with training and inference cost. Nonetheless, it should be acknowledged that this surrogate does not capture all implementation specific factors, such as differences in operation types or sequence-length dependence. This is however advantageous, as it enables comparisons that are relatively independent of sequence length, which is particularly suitable for architectures employing shared convolutional weights. Using parameter count as a surrogate thus supports extrapolation of results to future implementations in which reduced sequence lengths are explored, as discussed in Section 7.6.1.

The Pareto fronts (lines) of each individual architecture type (coloured), in combination with the overall Pareto front across all architectures (dotted) for efficiency and accuracy are presented in Figure 7.3, for the JPL (left) and JPL-A (right) datasets, coloured by the specific architecture configuration. The Dense implementations (green, red) dominate the other architectures in terms of computational efficiency, with the exception of the best-performing Conv–LC configuration (blue, $\approx 8 \cdot 10^5$ parameters) on the JPL dataset. In contrast, the Convolutional and TimeVAE architectures generally require approximately half an order of magnitude more parameters to achieve comparable accuracy levels at higher accuracy scales. This trend is especially pronounced for the JPL-A dataset (right).

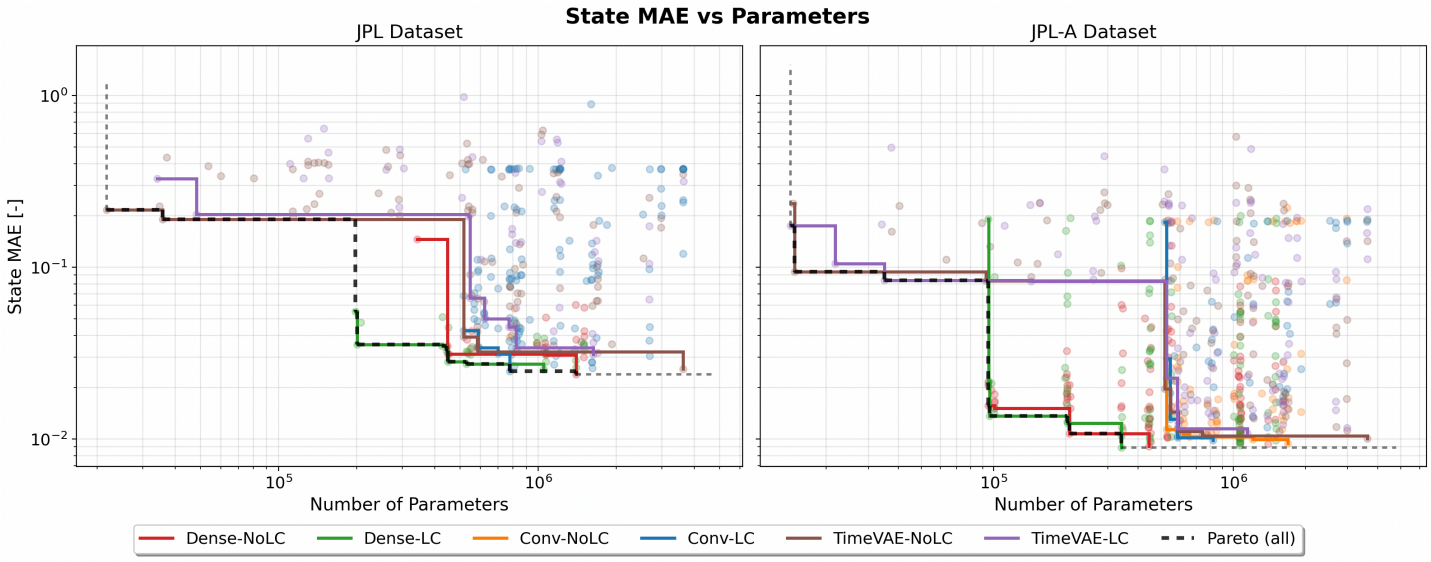


Figure 7.3: Pareto fronts of generated sample state accuracy and computational effort (using parameter count as a surrogate) across different architectures (coloured)

The sensitivity of all architectures to dataset diversity is evident in Figure 7.4, where an outward (increasing) shift of Pareto fronts is observed when comparing results for the JPL (blue) and the less diverse JPL-A (red) datasets. As in Figure 7.3, Figure 7.4 compares state accuracy against model parameter count across architectures, with the outward displacement of the Pareto front indicating the need for a larger receptive field to accurately model more diverse data. In particular, JPL configurations require approximately $1 - 2 \cdot 10^5$ additional parameters to achieve comparable state accuracy. This effect may be further amplified by normalization differences, as the measured minima and maxima in the JPL dataset span a wider range, yet the shift nonetheless reflects performance complexity trade-offs as dataset diversity increases.

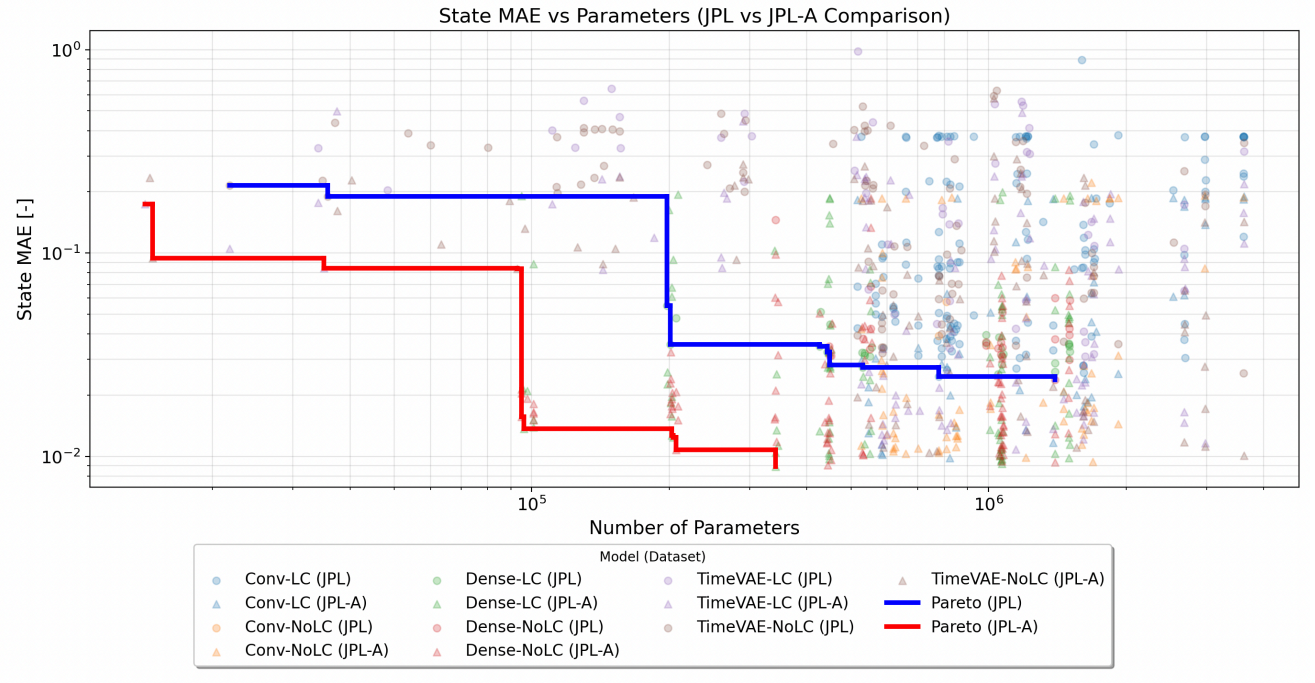


Figure 7.4: Pareto front per dataset (JPL, JPL-A) of state accuracy and computational effort

The best performing architecture of each type for each dataset remain on comparable orders of magnitude in terms of accuracy, as shown in Table 7.3. Across all best performing architectures, the inclusion of the Jacobi-based loss term (L_C) was able to decrease the amount of required parameters significantly, up to approximately one third of parameters for the convolutional and timeVAE implementations, whilst the overall best performing architectures remain the Dense implementations. The implications of the achieved accuracy levels in the context of cislunar mission design is further explored in Section 7.4.1, Section 7.4.4, and Section 7.6.1.

Table 7.3: Comparison of JPL-A and JPL accuracy (MAE) metrics per architecture. Lowest values per dataset are bolded.

Architecture	**Pos. MAE** [LU]	**Vel. MAE** [LU/TU]	**State MAE** [-]	**Parameter Count** [-]
		JPL-A Dataset		
Dense-NoLC-JPL-A	0.008459	0.009571	0.009051	445,020
Dense-LC-JPL-A	**0.008383**	**0.009451**	**0.008917**	**341,532**
Conv-NoLC-JPL-A	0.008884	0.009932	0.009408	1,691,651
Conv-LC-JPL-A	0.009588	0.009909	0.009822	582,955
TimeVAE-NoLC-JPL-A	0.009652	0.010517	0.010084	3,622,702
TimeVAE-LC-JPL-A	0.010191	0.011474	0.010833	1,145,542
		JPL Dataset		
Dense-NoLC-JPL	**0.023868**	**0.023800**	**0.023834**	1,397,660
Dense-LC-JPL	0.025494	0.025420	0.025457	1,050,060
Conv-NoLC-JPL	0.024517	0.024821	0.024669	2,103,482
Conv-LC-JPL	0.024625	0.024911	0.024768	**778,275**
TimeVAE-NoLC-JPL	0.025577	0.025667	0.025622	3,618,544
TimeVAE-LC-JPL	0.031356	0.030608	0.030982	1,627,218

7.4.1. Reconstruction of Periodic Orbits (Posterior Decoding)

Posterior decoding on the validation set is used to measure reconstruction errors with respect to the dataset, in order to verify that state accuracy requirements are satisfied, where posterior decoding refers to reconstructing samples using the mean of the learned latent posterior distribution. Figure 7.5 presents the reconstructed state (dotted orange) of an arbitrary sample (blue) over one full period for

each state parameter. This sample is obtained using the best-performing Convolutional VAE architecture for the JPL dataset, which is selected as a representative architecture given that all best performing models obtained comparable levels of accuracy. Figure 7.6 presents the reconstruction error for each state over one full period, demonstrating that the VAE reconstructs the sequence within an accuracy tolerance of $\approx 10^{-2}$ for both position and velocity. Observe that the prediction error is most defined at extremes of the orbit per state variable, suggesting sensitivity to the data normalization process described in Section 7.1.

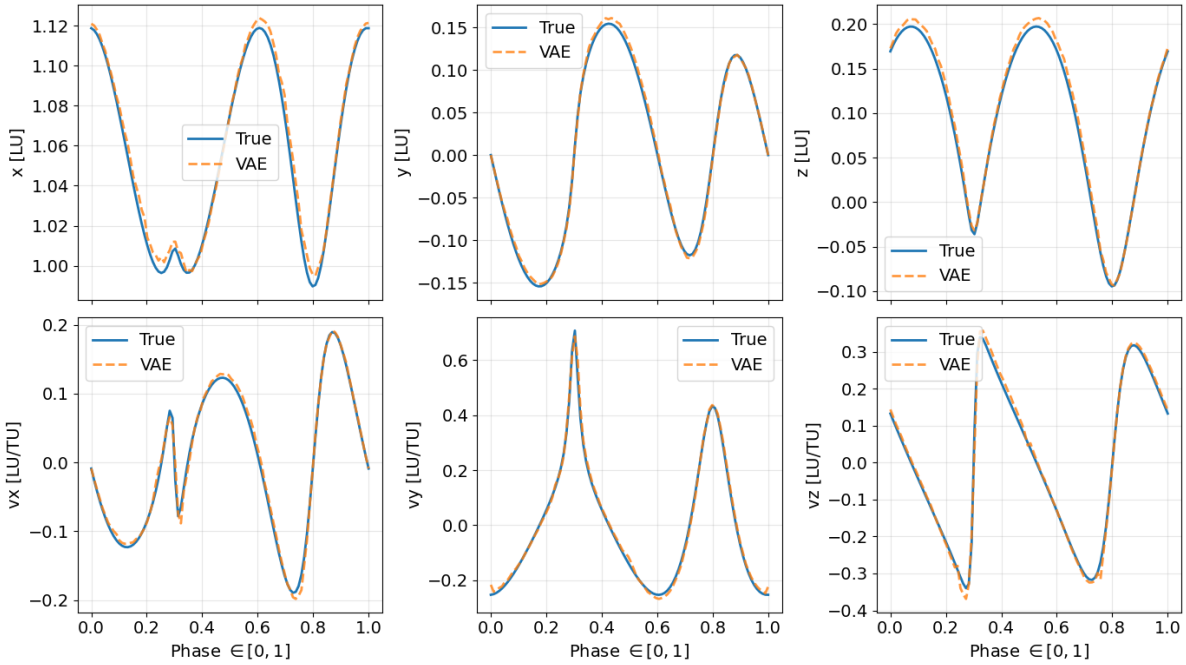


Figure 7.5: State variable reconstructions across full period phase for sampled VAE architecture and sampled orbit

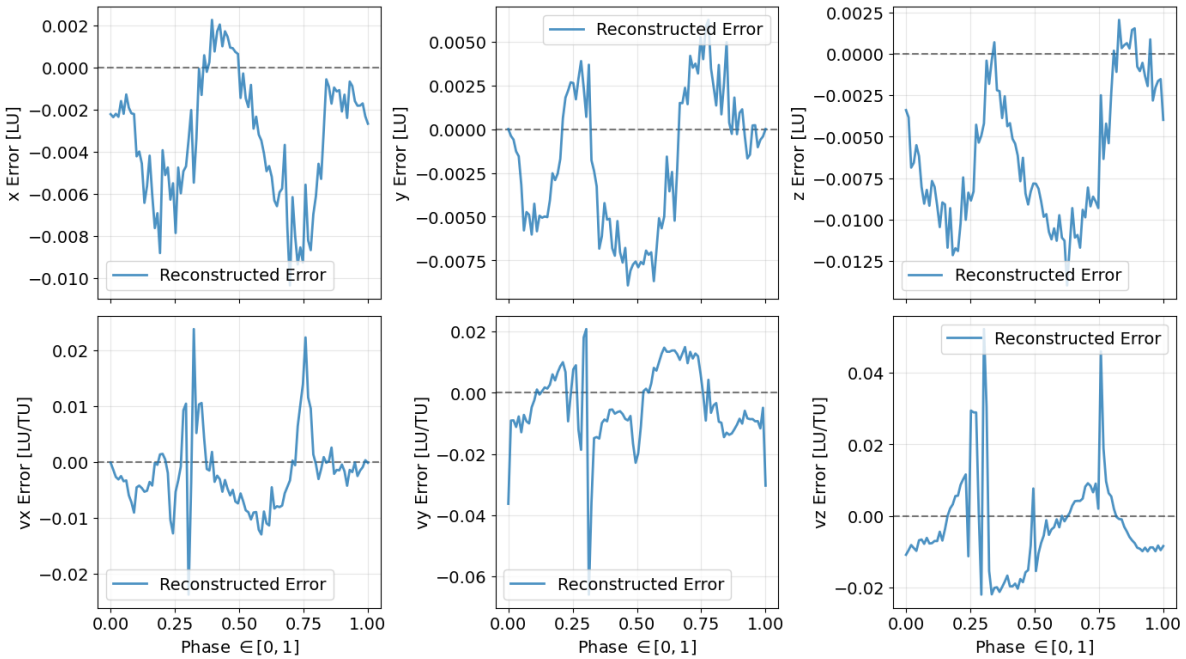
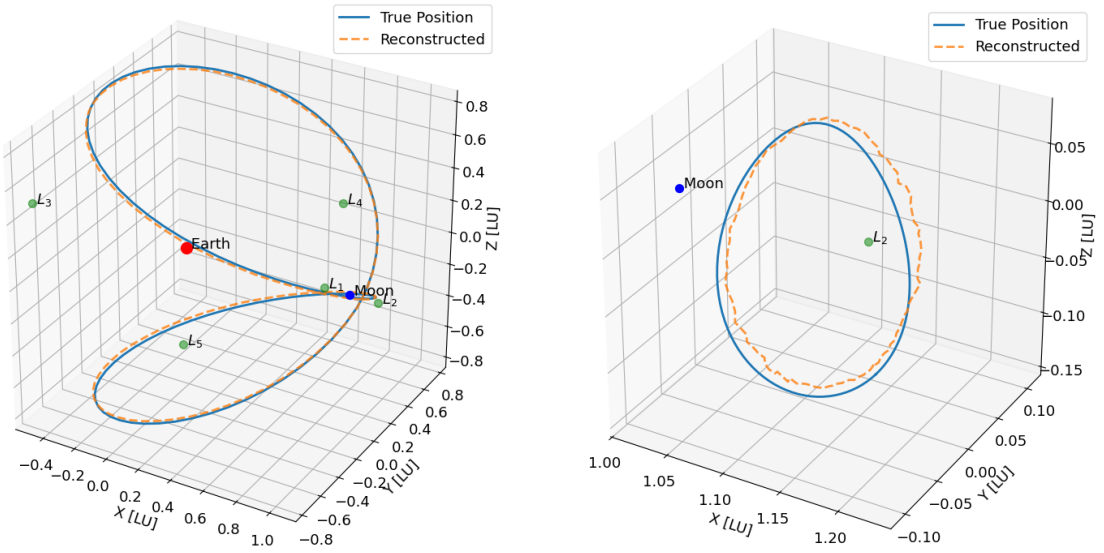


Figure 7.6: Reconstruction error per state variable for each VAE architecture for a sampled orbit

The impact of this obtained accuracy level is be spatially visualized in Figure 7.7 for the reconstruction

of two sample orbits, L_2 Axial (Figure 7.7a), and $L_{2,N}$ Halo (Figure 7.7b). This demonstrates that whilst the spatial reconstruction accuracy is only to a tolerance of 10^{-2}, the periodic orbits still closely resemble a member of their family. Further note the difference in error between the reconstructed samples, with Figure 7.7a possessing less a less noisy reconstruction error than Figure 7.7b, despite the L_2 Axial family being more complex in phase space. This again indicates that reconstruction fidelity is influenced not only by phase-space complexity, but also by factors such as the effective normalization scale of the data.



(a) Spatial representation of true (blue) and reconstructed (orange, dashed) L_2 Axial orbit

(b) Spatial representation of true (blue) and reconstructed (orange, dashed) L_2 Halo orbit

Figure 7.7: Reconstructed orbital trajectories for representative samples from two periodic orbit families.

7.4.2. Latent Structure Verification

As the latent space of the VAE's in question are inherently higher-dimensional representations, it becomes difficult to interpret as it is only possible to visualize 2 to 3 latent dimensions at a time. In order to better understand the prior distribution of the VAE across all latent dimensions, Uniform Manifold Approximation and Projection (UMAP) dimensionality reduction is applied. UMAP is a nonlinear dimensionality-reduction technique that models high-dimensional data as a low-dimensional manifold and constructs a graph-based representation to preserve both local neighbourhood structure and global geometry. Compared to tSNE, which excels at capturing local clusters but often distorts global relationships and scales poorly with dataset size, UMAP is selected for its faster speed and nonlinear structure presentation.

To accomplish this dimensionality reduction, the bifurcating orbit list for JPL-A, described in Figure 6.4 and Table 6.2 of Section 6.1.1.1, is excluded from both the training and validation sets. The latent representations of these orbits are then passed through the encoder and compared with the latent representations of the validation set. This procedure assesses whether the learned latent space preserves structural features consistent with the underlying bifurcation relationships, and is presented with (left) and without (right) the bifurcation node list in Figure 7.8. Figure 7.8 demonstrates that the latent representation of the VAE exhibits structural similarities to the original bifurcation diagram in Figure 6.4.

In particular, the tangent bifurcation at node XI (red, dashed, right sub-figure) provides a clear example, as the XI members of each family appear close to one another in latent space. Although the tangent bifurcations connecting the $L_{2,N/S}$ Halo and L_2 Lyapunov families (cyan, dashed, left sub-figure) to the L_2 Axial and L_2 Vertical families (orange, dashed, left sub-figure) are not captured, the tangent bifurcation relationship between the $L_{2,N/S}$ Halo and L_2 Lyapunov families is preserved. This discrepancy is likely due to the L_2 Axial and Vertical families exhibiting much larger deviations in the phase space z-axis compared to the other families, making them harder for the model to represent in a comparable way. It is possible this effect is amplified by the fact that the amount of samples seen in the

validation set effectively increases the sample spacing ΔC, in turn meaning the orbits in the vicinity of the bifurcation are not seen in this validation set. An alternative explanation is that the original ΔC resolution in the data is not high enough in the vicinity of the bifurcation such that this translates to a latent representation.

Furthermore, orbital families that bifurcate from the same parent family, and through the same type of bifurcation, tend to cluster together in latent space. This is evident in the proximity of the $L_{2,N}$ Dragonfly and $L_{2,N}$ Butterfly families, and the $L_{2,S}$ Dragonfly and $L_{2,S}$ Butterfly families (blue, dashed, left sub-figure). Interestingly, the model preserves the close proximity of the north/south symmetric branches only in the case of the $L_{2,N/S}$ Halo family, this symmetry is not maintained for the $L_{2,N/S}$ Dragonfly or $L_{2,N/S}$ Butterfly families. The sensitivity of UMAP dimensionality reduction to initialization parameters is demonstrated by the tight clustering of the $L_{2,N}$ Dragonfly and $L_{2,N}$ Butterfly families in the iteration without bifurcation nodes (left sub-figure), but not in the second iteration with nodes included (right sub-figure). This suggests that although these families are relatively close in latent space, meaningful differences remain between their latent representations.

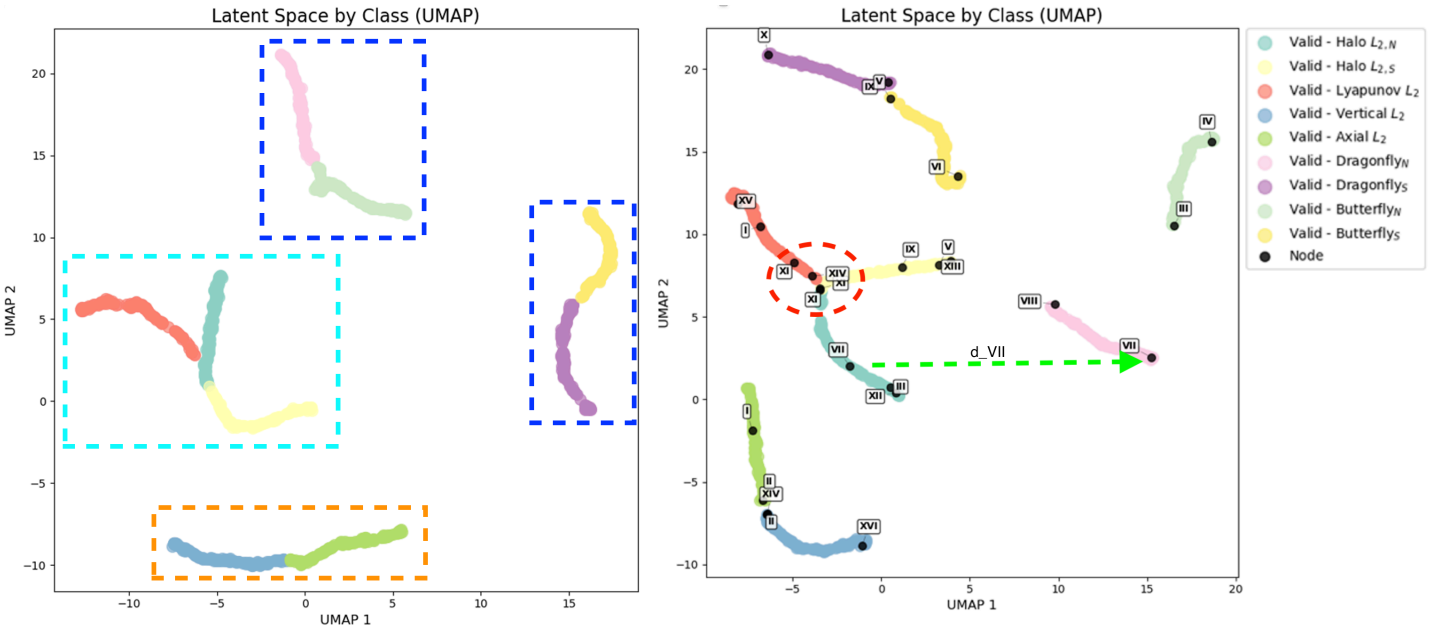


Figure 7.8: UMAP reduction of the latent distribution of the validation set, highlighting bifurcating nodes.

Note that the underlying data distributions lie on a thin manifold in the latent space. This arises as a result of the fact of strict physical sensitivity to the periodicity threshold, suggesting that assuming a Gaussian prior could be problematic. This is as the model maps a highly curved, low-volume data manifold in the latent space into a distribution that spreads distribution uniformly in all directions. This mismatch is potentially distorting the learned representations, possibly hindering reconstruction accuracy, and leading the encoder to over-regularize in order to satisfy the KL penalty. Relaxing the prior, increasing the latent dimensionality, or using a more suitable or learned prior may produce a better latent representation of the geometry of the data.

7.4.3. Latent Operations

With context to this investigation, the ability to use the latent representation for manipulation and sampling of periodic orbits is verified by comparing the latent representations within families, and the latent difference between nodes corresponding to bifurcations. This is performed for vector arithmetic methods described in Section 4.3.2. Cosine similarity is a measure of how similar two vectors are based on the angle between them, and is independent of their length. In the context of learned latent representations, cosine similarity is used to compare how closely two encoded states or features align in the latent space. Whilst cosine similarity focuses on directional alignment, the vector norm (magnitude) ratio is used to compare the magnitude of latent representations. Together, these metrics help quantify both

the directional and magnitude-based differences between latent vectors.

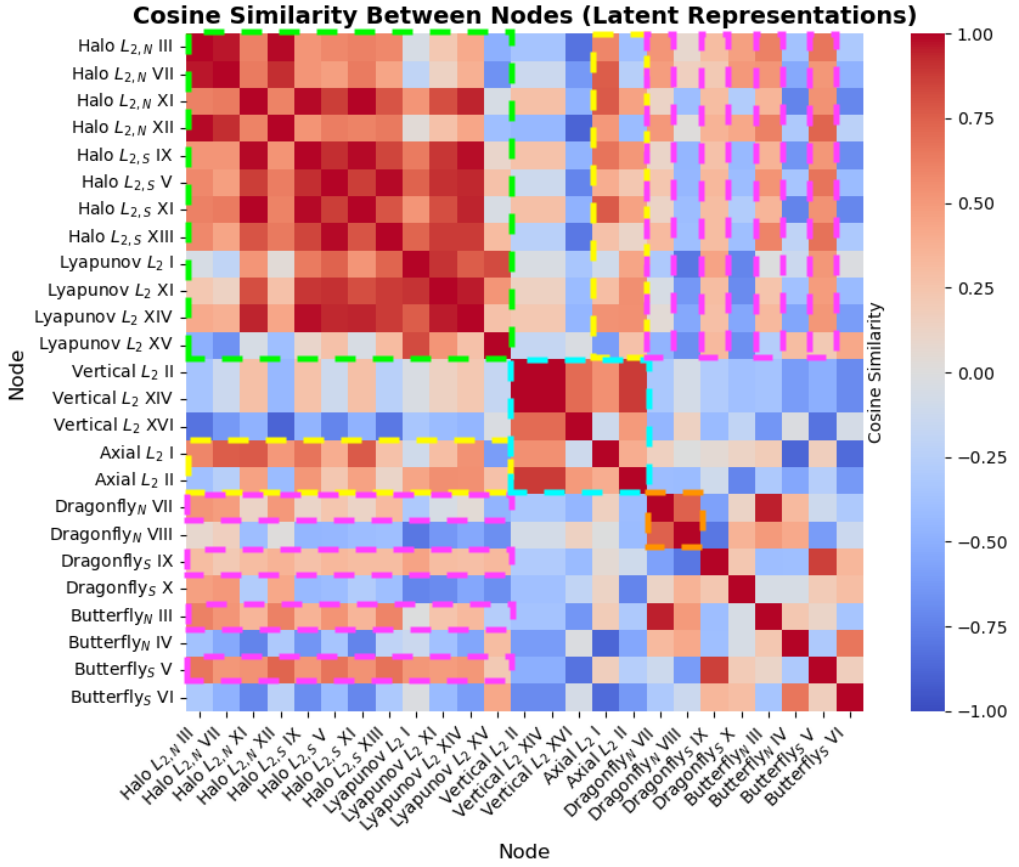


Figure 7.9: Cosine similarity between latent representation of node members

Figure 7.9 compares the cosine similarity of all node pairs, both within and across families. Pairwise similarities are colour-coded by magnitude, with higher similarity shown in red, orthogonality in white, and opposing vector directions in blue. High cosine similarity is observed among nodes within the same family and between tangentially bifurcating families. This behaviour is evident for the $L_{2,N/S}$ Halo and L_2 Lyapunov families (green, dashed), as well as for the L_2 Axial and L_2 Vertical families (blue, dashed), corroborating the UMAP-based similarity identified in Figure 7.8. Reduced similarity is observed at the extremities of these families (e.g. L_2 Lyapunov XV and L_2 Vertical XVI). The high mutual similarity within individual families is most clear when the family is isolated, as in the case of the $L_{2,N}$ Dragonfly family (orange, dashed). Vector similarity between the L_2 Axial family and the $L_{2,N/S}$ Halo and L_2 Lyapunov representations is indicated by elevated cosine similarity between select members of these families (yellow, dashed), confirming that the tangential bifurcation associated with node I is encoded in the latent representation. However, this similarity is generally reduced, as the bifurcation at node I introduces a rapidly developing vertical component in the L_2 Axial family relative to the L_2 Lyapunov family. This nonetheless suggests that interpolation within families is feasible, with implications for mission design applications and the potential use of unsupervised clustering algorithms on the latent representation, both explored in Section 8.1.

Although relative latent cosine similarity is observed for nodes arising from period-doubling bifurcations, such as between the $L_{2,N/S}$ Halo families and the $L_{2,N/S}$ Butterfly and Dragonfly families (pink, dashed), the magnitude of the vector differences between corresponding nodes in each family is relatively large. Figure 7.10 is used to analyze this behaviour by presenting the cosine similarity (left) and norm ratio (right) of distance vectors computed per node. An demonstration for how these difference vectors are identified for node VII (green, dashed in Figure 7.8) is given by:

$$\vec{d}_{\text{VII}} = \vec{z}_{L_{2,N}\text{ Halo-VII}} - \vec{z}_{L_{2,N}\text{ Dragonfly-VII}} \tag{7.3}$$

This concept is extended to compare the vector difference between each parent–child orbit pair arising from period-doubling bifurcations, is order to analyze the $\vec{d}_{\text{attr}} = \vec{z}_{\text{attr}} - \vec{z}_{\text{ref}}$ associated with a period-doubling bifurcation ($\vec{d}_{\text{2P}}$, as described in Section 4.3.2.

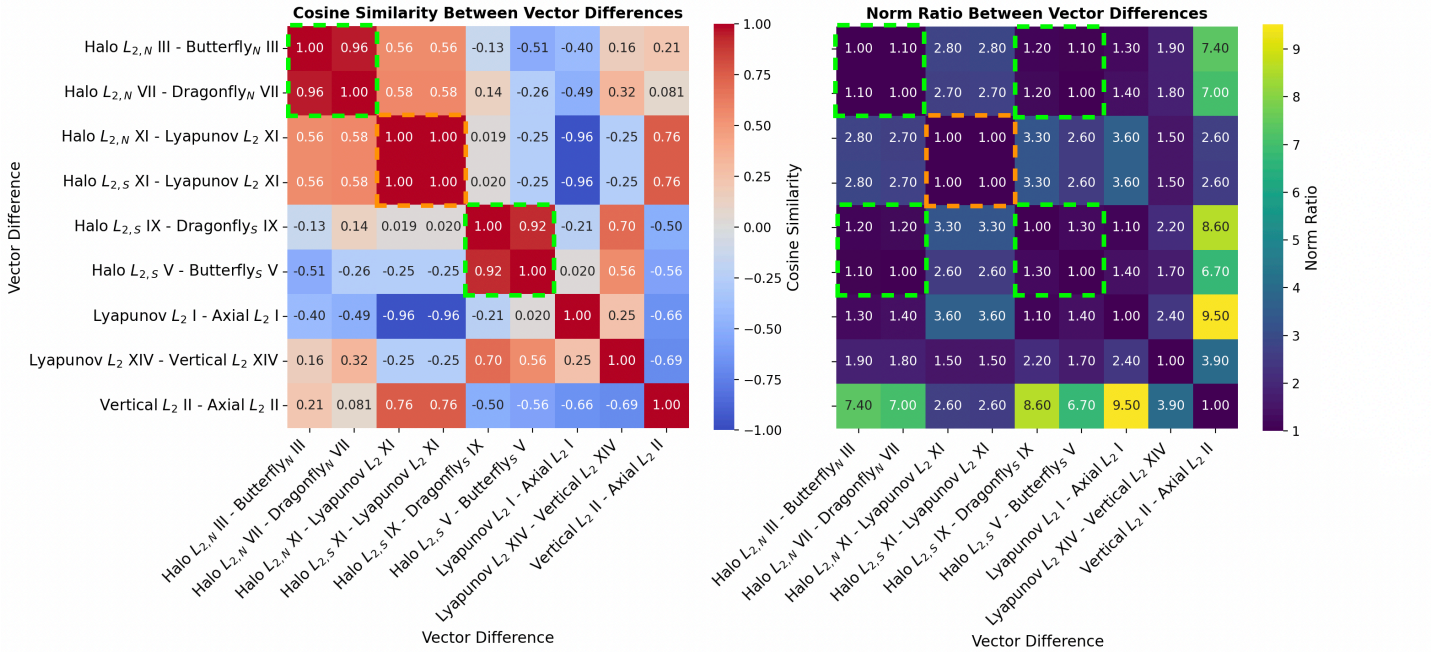


Figure 7.10: Cosine similarity between vector differences of latent representations per node

All period-doubling bifurcations (green, dashed) exhibit not only high mutual cosine similarity (left sub-figure, restricted to North/South implementations), but also strong similarity in the magnitude of the difference vectors, as evidenced by the norms (right sub-figure) of these vectors being ≈ 1 (green, dashed). Further analysis for a more diverse dataset is however required to statistically confirm these results, due to the limited number (4) of period doubling bifurcations present within the JPL-A dataset. Nonetheless, the vector-based similarity metrics, specifically a maximum cosine deviation between vector differences of 0.92 and a maximum magnitude ratio discrepancy of 1.3, along with maximum deviations from the mean of 0.96 (cosine) and 1.12 (norm ratio), indicate that these constitute the most closely aligned vector-difference relationships among the nodes examined within the dataset. Furthermore, this behaviour is despite hyperparameter optimization for state reconstruction error, not KL divergence. These results indicate that a latent vector arithmetic attribute associated with period-multiplying bifurcations can thus be identified in the latent space.

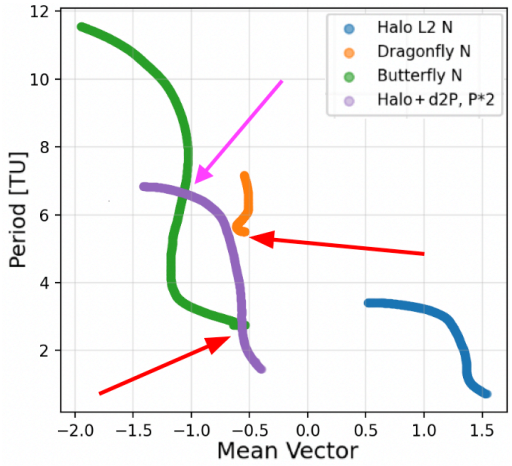


Figure 7.11: Transformation of latent representations of L_2 Halo along direction of period-doubling $\vec{d}_{\text{2P}}$

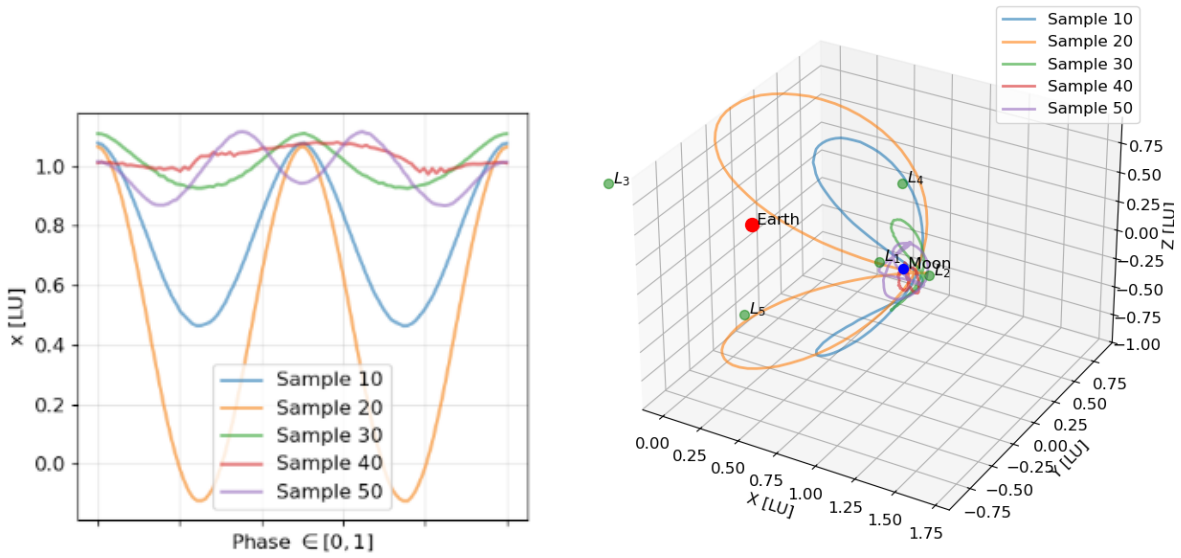
Due to the limited number of period-doubling bifurcations present in the dataset, Figure 7.11 applies the proposed mean-vector arithmetic to an entire family to illustrate the underlying concept. Specifically, the figure shows the projection of latent representations onto the mean period-doubling direction $\vec{d}_{\text{2P}}$, with the horizontal axis representing the component of each latent vector along this direction and the vertical axis indicating the corresponding orbital period. To demonstrate latent-space arithmetic, the parent $L_{2,N}$ Halo family (blue) is translated along $\vec{d}_{\text{2P}}$ by a magnitude determined in Figure 7.10, while simultaneously applying a period-doubling transformation to the original family. This yields the transformed family $L_{2,N}$ Halo + $\vec{d}_{\text{2P}}$, $P \cdot 2$ (purple). When compared with the latent representations of the $L_{2,N}$ Butterfly (green) and Dragonfly (orange) families, the approximate locations of the period-doubling bifurcations in latent space can be identified (red arrows). This property thus enables both the application of unsupervised clustering techniques to the latent representation and the use of vector arithmetic for orbit period multiplication in mission design. Both of these concepts are further explored in Section 8.1. Note the additional intersection (pink arrow) between the $L_{2,N}$ Butterfly (green) and transformed $L_{2,N}$ Halo curve (purple) arises as a result of the projection about this mean vector for illustration purposes, and does not hinder the implementation of clustering algorithms.

In conventional bifurcation exploration, as explained in Section 2.13.1, eigenvalue analysis is used to identify the location along an orbit at which a period-multiplying bifurcation occurs. However, when only information about the bifurcated child family is available, this approach is insufficient for recovering the corresponding parent family. Although eigenvalue analysis can determine the bifurcation point, in the case of a period-doubling bifurcation it does not provide a physically meaningful direction in phase space along which a perturbation may be applied to obtain the parent orbit, as the associated eigenvectors are fully complex. In order to overcome this, sampling in the vicinity of the proposed bifurcation point is typically employed. In contrast, analysis and sampling in the opposite direction along $-\vec{d}_{\text{attr}}$ in the latent space could enable identifying parent families of period-multiplying bifurcations families and thus overcome this limitation. Conceptually, this corresponds to applying the inverse of the transformation shown for the $L_{2,N}$ Halo family in Figure 7.11 to the $L_{2,N}$ Butterfly and$L_{2,N}$ Dragonfly families, in order to infer their originating families.

These results indicate that latent representations can be effectively leveraged for unsupervised clustering and classification tasks, owing to the proximity of related phase-space representations in the latent space. This is particularly relevant for databases in which periodic orbits are identified through grid-search methods rather than through bifurcation-based familial continuation, such as the Franz–Russell database [72]. In this context, latent-space structure has the potential to enhance clustering approaches that operate directly on phase-space similarity, as opposed to initial-condition–based methods. Moreover, when such databases are constructed at sufficiently high resolution, bifurcation relationships between samples and families may be inferred using the methods described above, enabling the construction of bifurcation diagrams for grid-search derived orbits without requiring computationally expensive continuation techniques, with further discussion explored in Section 8.1.

7.4.4. Latent Prior Sampling

To assess whether the VAE is capable of generating orbits from classes not seen during training, samples are decoded from the latent prior distribution. In a VAE, the latent variables are modelled as independent Gaussian random variables with zero mean and unit variance. Accordingly, for each latent dimension $z_i \in d$, z_i is sampled from the standard normal distribution ($z_i \sim \mathcal{N}(0, 1)$), to reflect the prior the VAE is trained to match through the KL divergence term. This approach ensures that generated samples are consistent with the latent space distribution imposed during training and allows us to evaluate the model's ability to generalize beyond the observed data.



(a) X channel over a full period of samples drawn from VAE latent space.

(b) Spatial representation of orbits generated from latent space sampling.

Figure 7.12: Latent sampling, sequence and spatial representation.

Figure 7.12a presents five samples generated through prior decoding, showing the evolution of the state variable x over a full period. The figure reveals an increased noise level for spatially smaller samples (red) and a reduced noise level for spatially larger samples (orange), suggesting a similar sensitivity to the normalization procedure, as described in Section 7.4.1. This sensitivity manifests as increased decoded noise for samples with smaller phase-space magnitude. Figure 7.12b presents the same five samples in phase space, further highlighting the increased noise observed for the spatially smaller periodic orbits. The noise is of similar absolute magnitude across all samples; however, it constitutes a larger relative proportion for the smaller samples.

7.4.4.1. MSDC Post-Processing

The generated samples exhibit a reconstruction state accuracy of approximately 10^{-2} [–], which does not satisfy the imposed accuracy constraints and therefore motivates the use of a post-processing scheme. MSDC post-processing is applied using $n = 9$ initial nodes, to enforce a convergence tolerance of $\tau_{\text{MSDC}} < 10^{-8}$ [–] in accordance with the model requirements. This procedure is illustrated for a sample generated from the prior in Figure 7.13, where selected nodes are sampled from the prior and differentially corrected to obtain a physically valid orbit. Figure 7.13 presents each state variable, with all 100 original time steps scaled to the [0,1] orbital phase. The nine sub-sampled initial nodes (yellow squares) are propagated over their respective segment lengths (red dashed) to their terminal points (yellow diamonds). MSDC is then applied iteratively until the propagated initial points (green triangles) and segments (blue) converge, reducing discontinuities at the segment endpoints (green inverted triangles) below the threshold $\tau_{\text{MSDC}} < 10^{-8}$ [–]. The sample presented converges to a physically similar orbit, as the differences between the initial and converged segments remain small relative to the overall orbital dynamics, and therefore satisfy the DTW criterion described in Section 6.3.1. Observe the increased relative noise in the initial segments (red, dashed) compared to the converged orbit (blue) in z and v_z. This noise persists during sampling even when the orbit is planar, as is the case here for a planar L_2 Lyapunov orbit.

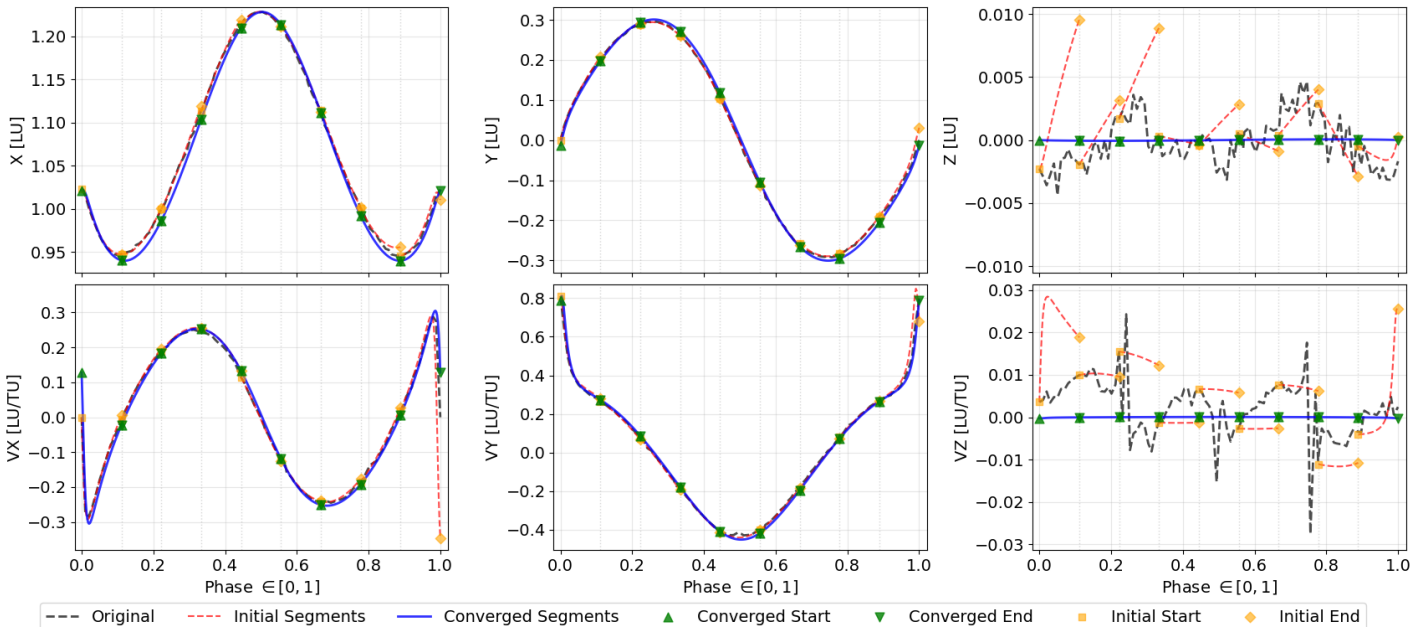


Figure 7.13: MSDC scheme demonstration for a L_2 Lyapunov orbit sampled from the prior.

Out of 1,000 decoded orbits tested, the conversion success rate was 54.4% when enforcing the DTW criterion, increasing to 62.7% when this criterion was disabled. The mean number of MSDC iterations required for convergence was 5.87.

7.4.4.2. Prior-Sample Clustering

To assess whether the model samples genuinely new families or merely performs in-family interpolation, clustering is applied to the converged MSDC prior samples. Dynamic Time Warping (DTW) is used to compute distances between prior samples and the original JPL-A trajectories, after which DBSCAN is performed on the resulting DTW distance matrix for clustering purposes. All trajectories are scaled to the range [0, 1] to eliminate DTW's sensitivity to amplitude differences. Additionally, the original sequences are sub-sampled to match the number of nodes in the prior samples, in an effort to mitigate the $O(n^2)$ time complexity of DTW with respect to sequence length.

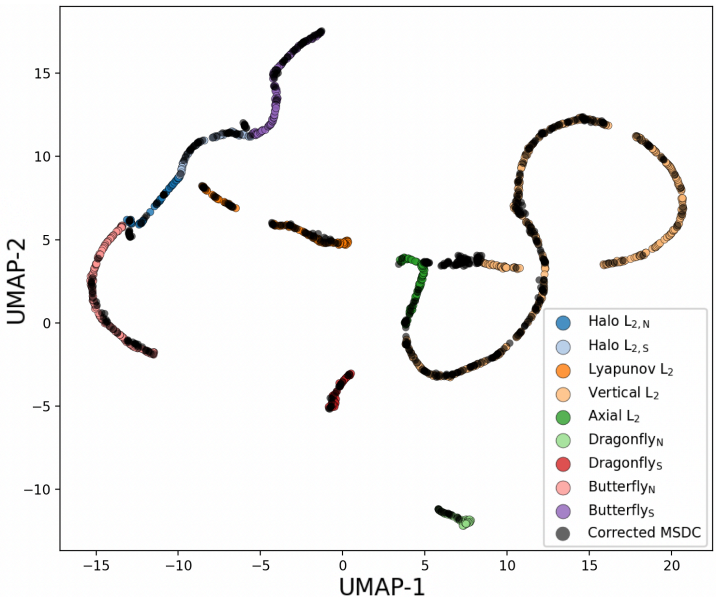


Figure 7.14: Clustering of decoded and post-processed solutions (DBSCAN, DTW). UMAP dimensionality reduction is applied for visualization.

Figure 7.14 presents the DTW distance results, with dimensionality reduction performed using UMAP and the DTW distance employed as a precomputed metric. Each black point corresponds to a decoded sample, while the coloured points represent the original training and validation set samples. The decoded samples originate from architectures trained exclusively on the JPL-A subset (nine classes) but are evaluated against the full JPL dataset (31 classes) during clustering. The close correspondence between decoded samples and the families observed during inference, evidenced by the proximity of black and coloured points in Figure 7.14, together with the absence of additional clusters identified by DBSCAN, indicates that the models primarily perform in-family interpolation rather than generating genuinely novel orbit families. This behaviour is likely attributable to the large number of similar orbits per family present during training, or more generally, to the limited diversity of the dataset with respect to family count.

If the MSDC DTW filter is removed, two additional classes not present in the dataset are identified: the inner and outer circular families, corresponding to circular orbits about the Earth and the Earth–Moon system, respectively, with approximately Keplerian dynamics. This observation preliminarily suggests that these families act as convergence sinks for instabilities arising during the MSDC post-processing procedure, further explored in Section 7.6.1.

7.5. Other Architectures Considered: TransFusion

In addition to the VAE-based architectures explored in this work, we also conduct a preliminary investigation of a transformer-based diffusion model, TransFusion. This exploration is limited due to training budget constraints, as training the diffusion model requires approximately 50 times more compute time (20 minutes vs 18 hours) than the VAE architectures. As a result, no hyperparameter optimization is performed for this model, nor is L_C tested.

TransFusion is applied directly to the raw time series data without the VAE latent representation, serving as a baseline comparison to evaluate the benefits and trade-offs of the latent-conditioned approach. This allows TransFusion to learn the full data distribution without information loss from compression, providing a reference point for assessing whether the VAE's structured latent space provides meaningful advantages in terms of generation quality, training efficiency, or sample diversity.

Similar MSDC post-processing and decoded-result clustering to that described in Section 7.4.4 is applied to the nominal TransFusion architecture, yielding a convergence rate of 49.7% and a mean MSDC iteration count of 7.34. This indicates poorer performance than the VAE, despite requiring roughly fifty times more computational effort. Future work should thus investigate the use of a frozen latent representation in combination with additional hyperparameter tuning to determine whether further improvements are achievable. It is also worth considering that such complex architectures may be unnecessarily powerful for reconstructing smooth sequences, as commonly encountered in periodic orbits of the CR3BP.

Future work should investigate an alternative framework in which the latent representation learned by the VAE is explicitly utilized, allowing the encoder to learn a structured latent space prior to employing a transformer–diffusion based decoder to reconstruct sequences at higher fidelity. In this approach, a frozen latent representation of the dataset of the best performing VAE model (as measured by either L_{MSE}, L_{KL} or L_C) is used to train a latent-conditioned diffusion model. By contrast, this explored approach does not produce a learned, meaningful latent distribution that can be utilized for downstream tasks such as interpolation, latent space exploration, or transfer learning applications. By comparing these two scenarios, it is possible to determine whether the (computational) benefits of working in latent space outweigh any potential quality degradation from the two-stage encoding-decoding process.

7.6. Model Sensitivity

Minor architectures sensitivities in the explored hyperparameter range, summarized in Table A.2 of Appendix A, are identified during the hyperparameter optimization process. These sensitivities are highly nonlinear and depend on the combined configuration of hyperparameters within a given architecture. For this reason, model variable sensitivities are examined primarily in the context of their relevance to MSDC post-processing in Section 7.6.1 and with respect to the PINN loss components L_C in Section 7.6.2.

7.6.1. MSDC Node Sampling/Noise Impact on Convergence Rates

As no trained models satisfy the accuracy and in sequence energy preservation requirements, a sensitivity analysis of the post processing mechanism (MSDC) is performed. To assess the impact of a noisy CR3BP trajectory on its convergence rates to the CR3BP and to the EHF model, a Monte-Carlo sensitivity analysis is conducted to evaluate the convergence behaviour of orbits exhibiting varying levels of discontinuity noise when corrected to both a clean CR3BP orbit and an EHF representation. This is done across a range of node amounts ($2 < N < 100$) for a single period of 300 periodic orbits, spread equally across all classes seen in the dataset. Each MSDC segment is deemed converged with a discontinuity tolerance of $\tau < 10^8$ in accordance with the requirements (Section 5.1.3). Each sampled shooting node is perturbed by adding an isotropic Gaussian vector whose expected Euclidean norm equals a specified noise level. For a node with state X_i and noise magnitude parameter $\eta > 0$, the noise δ_i is injected such that:

$$X_i^{\text{noisy}} = X_i + \delta_i, \qquad \delta_i \sim \mathcal{N}\left(0,\ \left(\frac{\eta}{\sqrt{6}}\right)^2\right), \quad i \in 6 \tag{7.4}$$

Note that an increase in node amount results also in increase of frequency of noise perturbations, resulting in MSDC needing to correct more local defects. An increase in MSDC node count however is expected to increase convergence rate to combat this issue, whilst also decreasing the required amount of iterations to converge. Figure 7.15 demonstrates this, where convergence rates for noise levels above 10^{-3} [-] are measured to reduce more quickly, as convergence rates are only $\approx 70\%$ for noise levels of $5 \cdot 10^{-3}$ [-], compared to $\approx 93\%$ for noise levels of $5 \cdot 10^{-4}$ [-] (top left) under optimal initial node phase (left column). The top row of Figure 7.15 shows the convergence success rate as a function of noise level (coloured) and node count (horizontal axis), while the bottom row reports the mean number of iterations required for MSDC convergence with these noise levels (coloured), computed over successfully converged runs. The right and left columns present the difference in convergence rates for initial node phase locations outside (left) and inside (right) the special set $\{0, 0.5, 1.0\}$, the location of intersection with the xz plane.

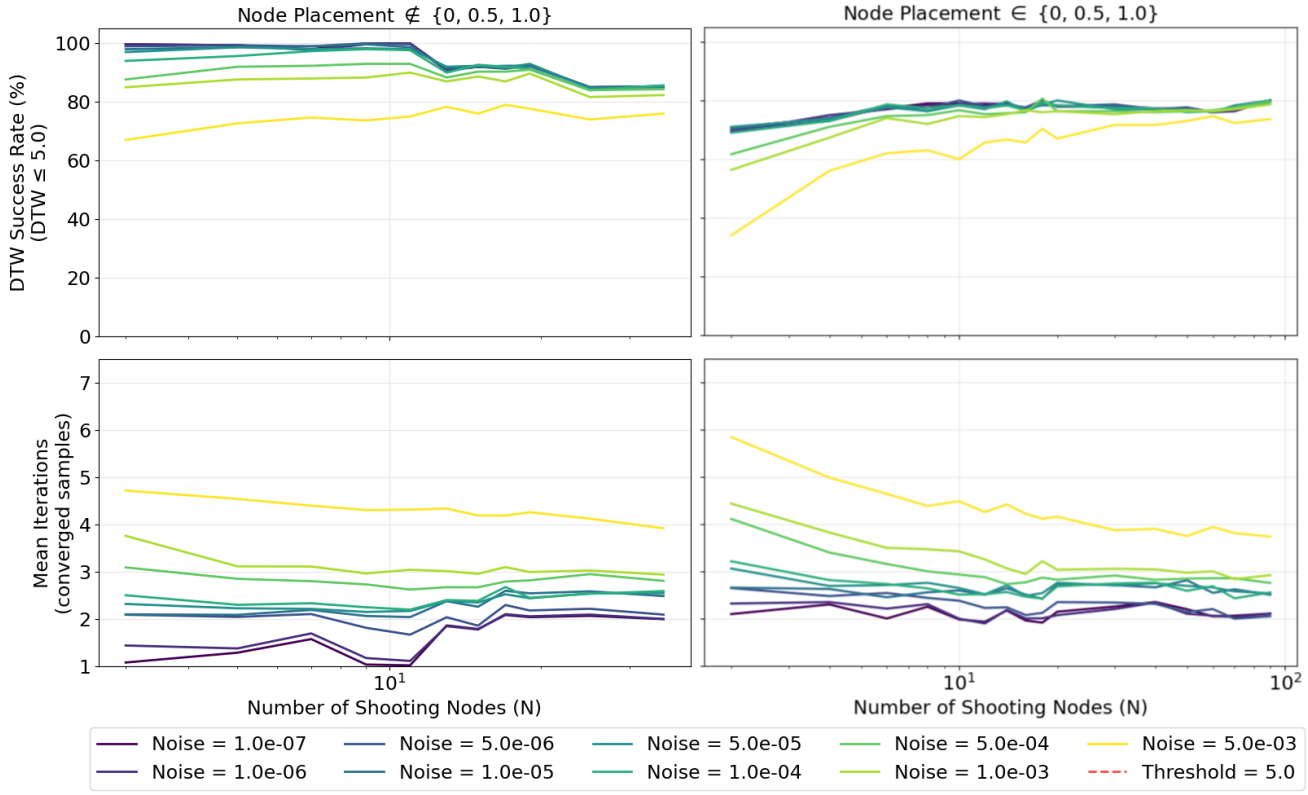


Figure 7.15: Comparison of initial phase shift behaviour for node placements outside (left) and inside (right) the special set $\{0, 0.5, 1.0\}$.

The drop off in convergence rates in Figure 7.15 when comparing the initial phase selection outside (left) and inside (right) $\{0, 0.5, 1.0\}$ is better understood by means of Figure 7.16. Figure 7.16 compares the initial node-phase selection for the sub-sampling scheme to the average success rate across all noise levels. Selection of nodes at the xz plane, (phases $\{0, 0.5, 1.0\}$) present reduction in convergence rates of $\approx 10\%$ when compared to other initial node values values. The need for a initial node phase selection not in $\{0,\ 0.5,\ 1.0\}$ is thus thus demonstrated by the large sensitivity of the MSDC convergence rates to this node positioning.

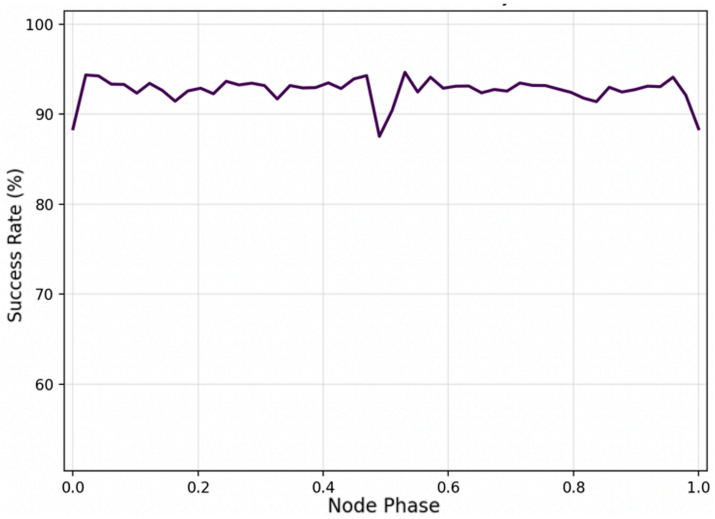


Figure 7.16: Impact of initial node–phase selection on MSDC convergence rates.

Extreme performance reduction in Figure 7.15 as a result of this initial node placement is further measured to be on the order of magnitude of $\approx 25 - 35\%$ for low node counts ($N < 10$) and high noise levels (yellow), when comparing performance between initial node placement not in $\{0, 0.5, 1.0\}$ (top, left) and initial node placement in $\{0, 0.5, 1.0\}$ (top, right). This impact is lessened by selecting a higher amount of nodes ($N \geq 10$), however the reduction is still on the order of magnitude of $\approx 10 - 20\%$ when comparing the curves not in the $\{0, 0.5, 1.0\}$ set (left) and in the $\{0, 0.5, 1.0\}$ set (right).

Further note the increase in mean amount of iterations (and resulting computational effort), when comparing initial node placement not in $\{0, 0.5, 1.0\}$ (bottom, left) and initial node placement in $\{0, 0.5, 1.0\}$ (bottom, right), corresponding to an mean increase in iteration amount of $\approx 20\%$ across all noise levels and node amounts. This behaviour is likely rooted in the fact that intersections with the XZ plane typically correspond to periapsis locations in the CR3BP, where the dynamics are most sensitive. At these points, the motion is dominated by kinetic energy, such that noise in the sampled states produces comparatively larger perturbations in the system energy and, consequently, a greater impact on trajectory evolution. An example of this node positioning is presented in Figure A.15 of Appendix A.

Figure 7.15 shows that increasing the number of nodes generally reduces the mean number of iterations required for MSDC convergence across all noise levels. However, this trend breaks down at low noise amplitudes ($\delta \leq 5 \cdot 10^{-4}$) when the node count becomes large ($N \geq 10$), where a decline in convergence success rate is observed. In the context of the generation pipeline, this indicates that at the noise levels encountered during decoding ($\approx 3 \cdot 10^{-3}$), convergence of sampled solutions is limited not only by the learned distribution but also by the MSDC correction scheme itself. Under ideal MSDC convergence, overall success rates of generated solutions could therefore increase by approximately 30–40

Additionally, the high convergence rates observed for low node counts ($N < 10$, Figure 7.15, top left) indicate that a reduction in the generation time-step length is feasible, enabling a denser model representation. In particular, reducing the generated sequence length by approximately an order of magnitude ($N \approx 10$) appears near-optimal for convergence success, albeit at the cost of increased iteration counts. This increase in iteration count is most pronounced for the noise level $5 \cdot 10^{-3}$ under node initialization within the set $\{0, 0.5, 1.0\}$ (Figure 7.15, bottom right), where the number of iterations required for $N < 10$ is notably higher than for $N \geq 10$.

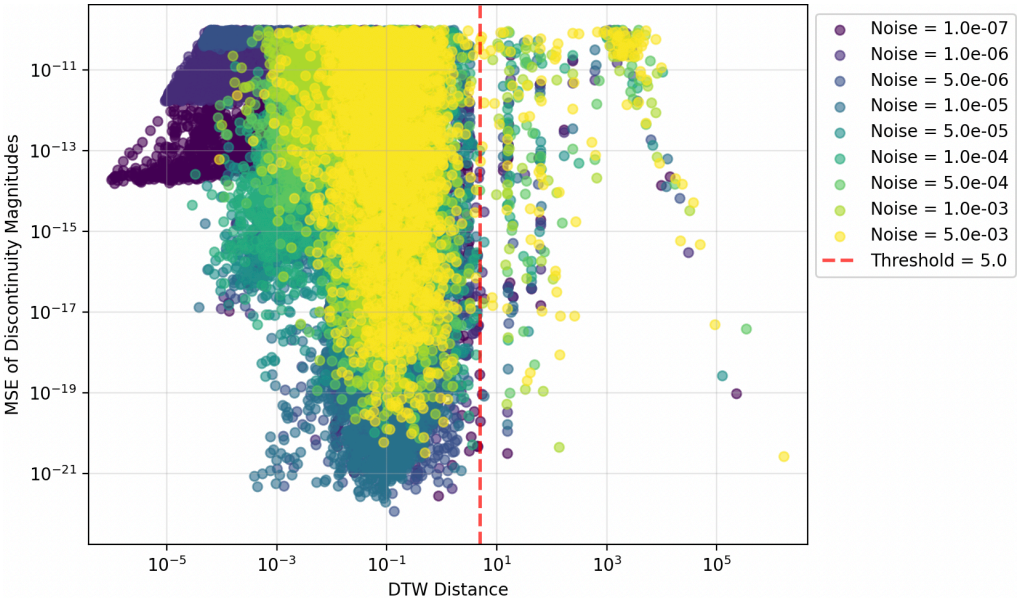


Figure 7.17: DTW similarity threshold used to determine trajectory correspondence.

The DTW filtering step included in the analyses of Figure 7.15 and Figure 7.16, as described by Section 6.3, is performed in order to filter for successful convergences to unintentional orbits. This filtering step consists of applying a DTW filter, where the MSDC converged orbit is compared to its original noisy counterpart, without any normalization step. If the DTW distance between these two orbits is greater then a value of 5, this is considered an unsuccessful convergence. This threshold (red, dashed) is presented in Figure 7.17 and is chosen conservatively to lie on the lower side of the gap observed in the range $5 < DTW < 10$, which corresponds to a region largely devoid of converged samples. Points to the left of this threshold indicate convergence to an orbit that remains similar to the original sample, whereas points to the right correspond to convergence to an orbit with significantly different dynamics than the sample, typically an inner or outer circular family. Averaged across all noise levels, tolerance settings, and node counts, convergence to such undesirable orbits occurs in approximately 3% of cases, with a higher incidence observed at elevated noise levels. This higher incident at elevated noise levels is demonstrated by the fact that the majority of samples above the boundary are observed to have noise levels $\approx 10^{-3}[-]$. Shifting this boundary to 10 [-] results in a minor reduction in undesirable convergence rates of $\approx 0.1\%$, suggesting only minor sensitivity to the DTW threshold within the $5 < DTW < 10$ region.

7.6.2. PINN Gradient Sensitivity

In order to effectively implement the PINN, the sensitivity of how gradients develops to different training regimes is analyzed. The scenario in which only L_{MSE} and L_C are considered is used, as in early training regimes in VAE's that focus on reconstruction accuracy, L_{MSE} tend to dominate gradient updates, compared to later training regimes being dominated by L_{KL}. Ideally, in order to prevent L_C from overwhelming the training by having the gradient update being on different orders of magnitude, γ is set such that:

$$\gamma \left\| \nabla_{\hat{\mathbf{X}}} L_{\text{Jac}} \right\| \approx \left\| \nabla_{\hat{\mathbf{X}}} L_{\text{MSE}} \right\|, \quad \therefore \gamma = \frac{\left\| \nabla_{\hat{\mathbf{X}}} L_{\text{MSE}} \right\|}{\left\| \nabla_{\hat{\mathbf{X}}} L_{\text{Jac}} \right\|} \tag{7.5}$$

Instead of considering the batch average, the per sample scenario is derived. As both L_{MSE} and L_C average over the batch, this is factored out. $\nabla_{\hat{\mathbf{X}}} L_{MSE}$ is derived as follows, using the deterministic (non-probabilistic) representation of L_{MSE}:

$$\mathcal{L}_{\text{MSE}} = \frac{1}{N}\sum_{i=1}^{N}\left(\hat{\mathbf{y}}_i - \mathbf{y}_i\right)^2 \tag{7.6}$$

Using the multivariate chain rule:

$$\nabla_{\hat{\mathbf{X}}} L_{\text{MSE}} = \frac{\partial L_{\text{MSE}}}{\partial \hat{X}} \begin{bmatrix} \frac{\partial L_{\text{MSE}}}{\partial \hat{x}} \\ \frac{\partial L_{\text{MSE}}}{\partial \hat{y}} \\ \frac{\partial L_{\text{MSE}}}{\partial \hat{z}} \\ \frac{\partial L_{\text{MSE}}}{\partial \hat{v}_x} \\ \frac{\partial L_{\text{MSE}}}{\partial \hat{v}_y} \\ \frac{\partial L_{\text{MSE}}}{\partial \hat{v}_z} \end{bmatrix} = \begin{bmatrix} \frac{1}{3}(\hat{x} - x) \\ \frac{1}{3}(\hat{y} - y) \\ \frac{1}{3}(\hat{z} - z) \\ \frac{1}{3}(\hat{v}_x - v_x) \\ \frac{1}{3}(\hat{v}_y - v_y) \\ \frac{1}{3}(\hat{v}_z - v_z) \end{bmatrix} \tag{7.7}$$

Similarly, $\nabla_{\hat{\mathbf{X}}} L_C$ can be derived as follows:

$$\mathcal{L}_{\text{C}} = \gamma \frac{1}{N_t}\sum_{i=1}^{N_t}\left(\hat{C}_i - C_i\right)^2 \tag{7.8}$$

$\frac{\partial L_{\text{C}}}{\partial \hat{X}}$ is similarly obtained by applying multivariate chain rule:

$$\frac{\partial L_{\text{Jac}}}{\partial \hat{X}} = \frac{\partial L_{\text{Jac}}}{\partial \hat{C}}\,\frac{\partial \hat{C}}{\partial \hat{X}}, \qquad \therefore \frac{\partial L_{\text{Jac}}}{\partial \hat{X}} = 2(\hat{C} - C)\frac{\partial \hat{C}}{\partial \hat{X}} \tag{7.9}$$

In vector form:

$$\nabla_{\hat{\mathbf{X}}} L_C = 2(\hat{C} - C)\cdot \nabla_{\hat{\mathbf{X}}}\hat{C}, \qquad \nabla_{\hat{\mathbf{X}}}\hat{C} = \begin{bmatrix} \frac{\partial \hat{C}}{\partial \hat{x}} & \frac{\partial \hat{C}}{\partial \hat{y}} & \frac{\partial \hat{C}}{\partial \hat{z}} & \frac{\partial \hat{C}}{\partial \hat{v}_x} & \frac{\partial \hat{C}}{\partial \hat{v}_y} & \frac{\partial \hat{C}}{\partial \hat{v}_z} \end{bmatrix}^\top \tag{7.10}$$

Substituting the partial derivatives derived in Section 2.3 into the vector form yields the following:

$$\nabla_{\hat{\mathbf{X}}} L_C = 2(\hat{C} - C) \begin{bmatrix} 2\hat{x} - \frac{2(1-\mu)(\hat{x}+\mu)}{((\hat{x}+\mu)^2 + \hat{y}^2 + \hat{z}^2)^{3/2}} - \frac{2\mu(\hat{x} - (1-\mu))}{((\hat{x} - (1-\mu))^2 + \hat{y}^2 + \hat{z}^2)^{3/2}} \\ 2\hat{y} - \frac{2(1-\mu)\hat{y}}{((\hat{x}+\mu)^2 + \hat{y}^2 + \hat{z}^2)^{3/2}} - \frac{2\mu\,\hat{y}}{((\hat{x} - (1-\mu))^2 + \hat{y}^2 + \hat{z}^2)^{3/2}} \\ -\frac{2(1-\mu)\hat{z}}{((\hat{x}+\mu)^2 + \hat{y}^2 + \hat{z}^2)^{3/2}} - \frac{2\mu\,\hat{z}}{((\hat{x} - (1-\mu))^2 + \hat{y}^2 + \hat{z}^2)^{3/2}} \\ -2\hat{v}_x \\ -2\hat{v}_y \\ -2\hat{v}_z \end{bmatrix}. \tag{7.11}$$

In order to assess how frequently the weighting factor γ must be updated across different training regimes, the partial derivatives of both L_{MSE} and L_C are evaluated across 1000 randomly selected states X in the dataset with an induced Gaussian error with predetermined magnitude, similarly to Section 7.6.1. This is an optimistic surrogate for the local gradient behaviour under a range of prediction errors $\epsilon = \hat{X} - X$. The magnitude of ϵ corresponds to different stages of training: large errors represent early epoch behaviour, whereas small errors represent later epoch convergence. Noise within each each individual state variable is examined, as well as the combined noise on all states prediction error. Importantly, the comparison is performed on the gradients, specifically magnitudes and directions, rather than on the loss values themselves.

Figure 7.18 shows the magnitude of the gradient updates of L_{MSE} and L_C. The top panel presents these magnitudes per state component (coloured, dashed, left) as well as for the full state (red, right), while the bottom panel shows the ratios between the corresponding gradient magnitudes. As L_{MSE} is equal per state component, only a single component is presented in the top left figure (blue, solid), compared to the full state presented in the top right figure (blue, solid). Similarly, the rightmost plots present the magnitude and ratios of loss for noise in all state components (red). The bottom right figure in particular demonstrates the need to adapt γ in different training regimes, as the order of magnitude of the ratio of the gradients of the losses (green) differs vastly across ranges of prediction errors. In order to accurately scale gradient update steps to be equal in magnitude across different regimes, γ would in an ideal scenario be scaled with the inverse of this ratio in order to maximize training stability, such that their ratio is equal to unity (red, dashed). Note that sensitives for the x component in state produce the majority of noise in this ratio. The lack of a leading term in $\frac{\partial C}{\partial z}$ reduces the impact of noise in z in training regimes with large errors, demonstrated by the lack of impact of noise in the ratio between the loss functions. In training regimes with small errors, gradient updates are most sensitive to prediction errors in the x component of the state, followed by z, v_y, v_x and v_z in descending order.

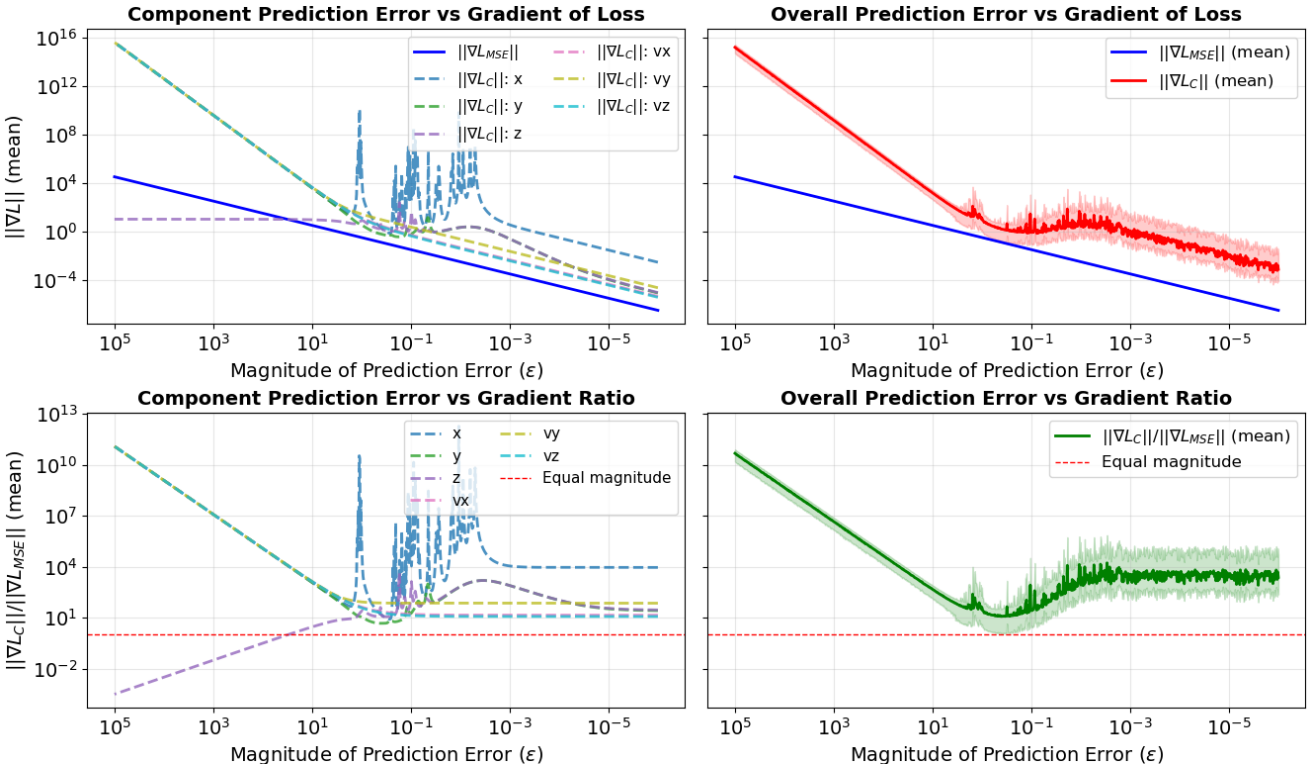


Figure 7.18: Impact of prediction errors on magnitude and ratio of L_{MSE} and L_C gradients

The cosine similarity in different training regimes between L_{MSE} and L_C is presented in Figure 7.19 per component (left), and and cross all state components (right). Cosine similarity, explained in Section 7.4.3, in this scenario measures the angle between the gradient updates, and is magnitude invariant, approximates a perfectly balanced γ. In earlier training regimes (large prediction errors), notice orthogonality (cosine similarity ≈ 1 between z and all other components, signalling that errors in the z state of predictions will result in a different gradient update direction than all other components, which start with gradient updates in the same direction as L_{MSE}. This effect is however negated by the low (relative) magnitude of the z update vector in these training regimes, visible in bottom left sub figure of Figure 7.18 (dashed, purple). High variations in the cosine similarity indicate the similarity in the angle is highly dependent on which exact state is being evaluated, a result of $\hat{X}$ components in $\nabla_{\hat{\mathbf{X}}} L_C$. A lack of negative cosine similarity values (below dashed, red) indicate that there are no measured scenarios in which gradient conflict between loss functions will occur, signalling the potential for training stability and compatibility of loss functions.

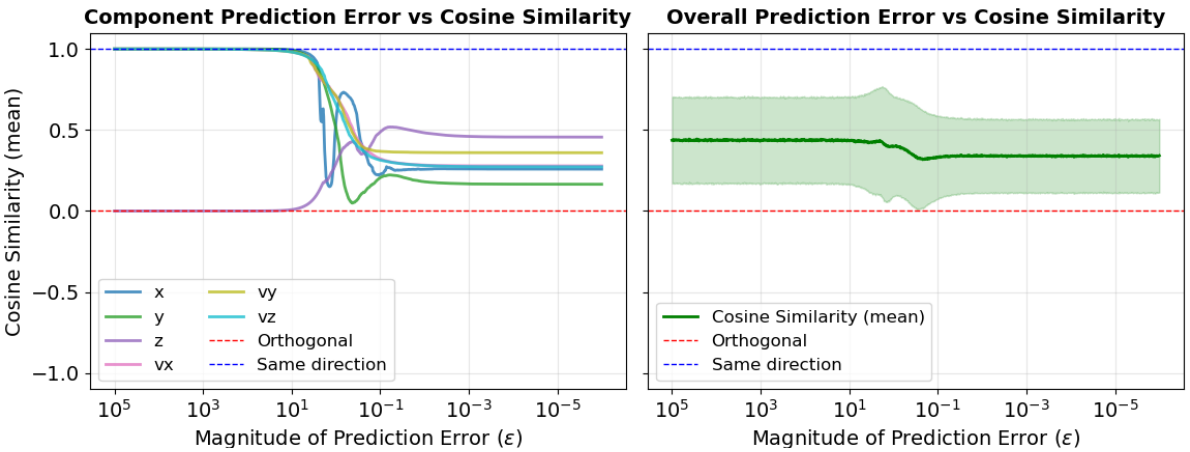


Figure 7.19: Cosine similarity between L_{MSE} and L_C gradient updates per component (left) and overall (right) across a range of prediction errors ($||\epsilon||$).

These sensitivity patterns highlight the importance of adopting an adaptive strategy for selecting the weighting factor γ in future training regimes. Although the present results expose substantial variability in how each state component influences the gradient magnitude, most notably the disproportionate effect of noise in the x component, they demonstrate that including L_C offers meaningful additional structure to the learning process. The observed behaviour indicates that a fixed γ is unlikely to remain optimal across all stages of training, particularly as prediction errors shrink and the relative influence of each state variable shifts. The heightened sensitivity of the loss landscape to perturbations in the x direction underscores the necessity for high-quality, low-noise training data in this dimension to ensure stable gradient updates and consistent convergence, or for the tailoring of models to specific ranges in the phase space (e.g. seperate models for the vicinity of L_2 or L_2 or for planar/spatial orbits). Furthermore, the utility of incorporating L_C is demonstrated by the fact that no gradient conflicts occur in all training regimes, despite the need for dynamically balancing its contribution throughout the learning trajectory.

7.7. Neural Surrogates of Invariant Manifolds

As stated in Section 2.9, optimizing low-energy transfers in the CR3BP often involves frequently computing invariant manifolds. During the seeding process of optimizers, hetero/homoclinic connections between manifolds are use as initial guesses to iterate upon during the optimization process. From a selecto-recombinative perspective, seeding with these connections acts as a mechanism for introducing high-quality guesses into the optimization process, rather than relying on purely random or uninformed initial guesses [28]. These guesses are then perturbed during the optimization process to find local minima near the invariant manifolds. In order to identify heteroclinic/homoclinic connections between periodic orbit families, it requires computing new periodic orbits with desired characteristics through family continuation or interpolation, propagating these orbits with knowledge about the arc's starting points, and then continuing to propagating the arcs to trace out the manifold, as presented in Section 3.2.2.

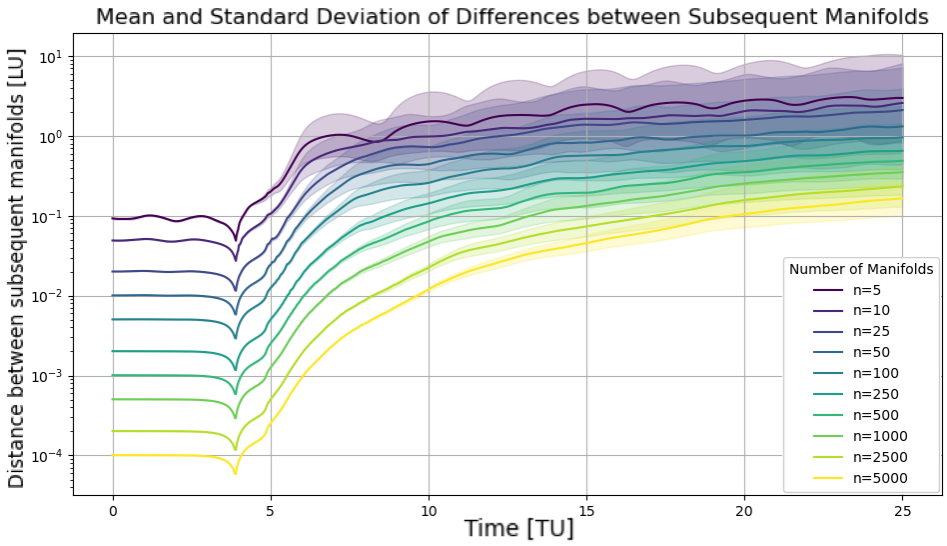


Figure 7.20: Mean distance ($\pm 1\sigma$, shaded) between subsequent manifold arcs for an arbitrary $L_{2,S}$ Halo orbit

Generating and storing detailed manifold datasets is computationally impractical and exceeds the limits of local storage or memory [28]. This is demonstrated by Figure 7.20, where, similarly to Figure 6.16, the distance between subsequent manifolds is plotted against propagation time for varying amounts of manifold arcs (coloured). Should state resolution of 10^{-2} [LU] at the end of the propagation be desired, they must be propagated with a initial arc separation density of 10^{-4} [LU], requiring 5000 arcs to be propagated. As also observed in Figure 6.16, the manifolds begin to converge around t = 4 [TU], corresponding to their close approach to the Moon. Beyond this point, positional differences between successive manifold arcs increase with propagation time, reaching approximately 10% of the system characteristic length (LU) by t = 10 [TU], and remaining of the same order of magnitude thereafter, around t = 20 [TU]. The resulting spatiotemporal divergence of these manifolds is further illustrated in Figure 7.21, which presents manifold arcs (red) propagated for 3, 5, and 10 [TU]. Notice the increasing distance between the arcs in at 10 [TU] (right), compared to 3 [TU] (left).

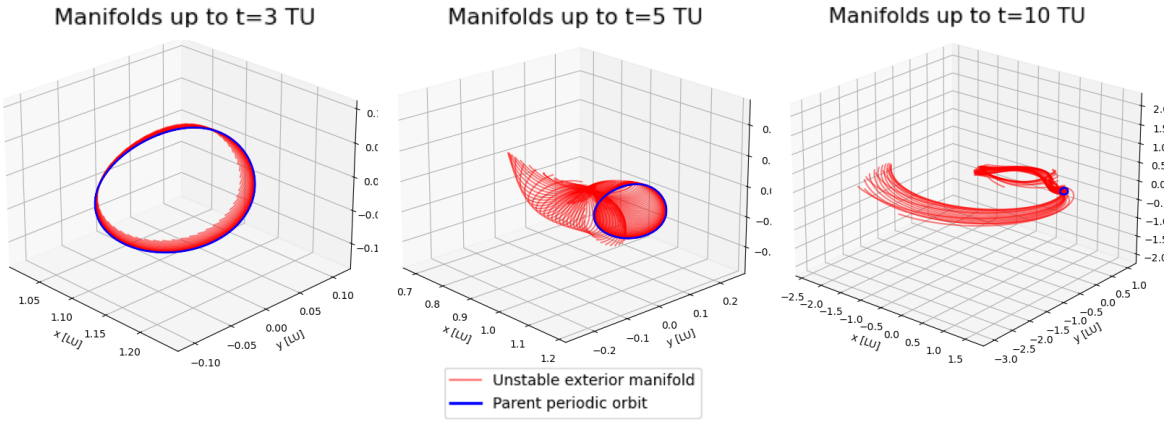


Figure 7.21: Propagation of manifold arcs for $t = 3$ (left), 5 (middle), 10 (right) [TU] for an arbitrary $L_{2,S}$ Halo orbit

7.7.1. Manifold Set Population Experimental Parameters

In order to mitigate the computational cost demonstrated in Section 6.5, the potential of neural surrogates to approximate the set of manifold arcs outlined in Section 3.2.2 is explored. This architecture is designed as a neural surrogate for the population of the set of the manifold arcs for use during the environment querying stage of optimization algorithms. For experimental purposes, the dataset is limited to a single family of periodic orbits $L_{2,S}$ Halo across a range of orbits that posses invariant manifolds (unstable orbits). The $L_{2,S}$ Halo family is selected for experimental purposes due to both its frequent

use in missions (list missions here), in combination with the majority of its members being relatively unstable ($|\nu|$ >10) and as a result possessing rapidly diverging invariant manifolds. Furthermore, note that the total amount of FLOPs for the surrogate model is limited to the number required for numerical integration at inference time.

Table 7.4: Neural Surrogate Formulation: Input/Output Parameters

Parameter	Description	Values
Network Constants		
κ	Parent orbit Family	[0, N_{classes}]
χ	Manifold direction (stable/unstable, interior/exterior)	[0, 4]
Inputs		
ξ	Relative position along parent orbit	[0, 1]
C	Jacobi of parent orbit	[C_{min}, C_{max}]
t_f	Time of flight on manifold arc	[0, T_{max}]
Output		
$\mathbf{X_f}$	State at time t_f	$\mathbb{R}^6$

Furthermore, a subset of these orbits is selected such that the control parameter, the Jacobi constant C, listed as an input in Table 7.4, has an injective representation. This restriction is required because a single value of C within the $L_{2,S}$ Halo family can correspond to two distinct orbits, as discussed in Section 3.2.1 and visually presented in Figure 3.2a. Although alternative parameters (e.g., perigee or apogee radius relative to one body, or the orbital period P) could enforce injectivity, the Jacobi constant is chosen due to its immediate computability from a single state and its role in defining the abacus resolution. This is confirmed by means of Figure 7.22, where the Jacobi value of the orbit is compared to its stability and period within the $L_{2,S}$ Halo family. Notice that for each Jacobi value, two solutions exist; a short ($P < 2.375$[TU]) and long period orbit ($P > 2.375$ [TU]). This distinction between short and long-period orbits serves as a basis for segmentation, with a dividing threshold set at 2.375 [TU].

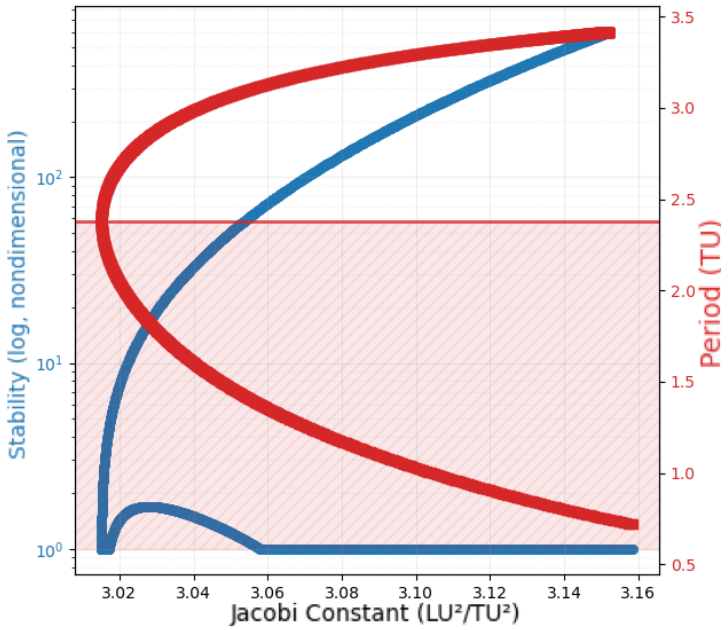


Figure 7.22: Segmentation division of the $L_{2,S}$ Halo family. The line and hatched region correspond to 2.375 TU and to orbits with periods shorter than 2.375 TU, respectively.

The orbit family is therefore segmented so that C uniquely identifies a single orbit within each segment, eliminating this ambiguity. This segmentation is illustrated in Figure 7.22, which shows the separation between short-period (red, hatched) and long-period (unhatched) orbits. Within the selected region, the Jacobi constant admits an injective mapping, yielding a unique orbit for each value of C. The

final segmented structure is visualized in state space, with the Jacobi constant treated as a control parameter, coloured along the initial conditions of the periodic orbits in Figure 7.23.

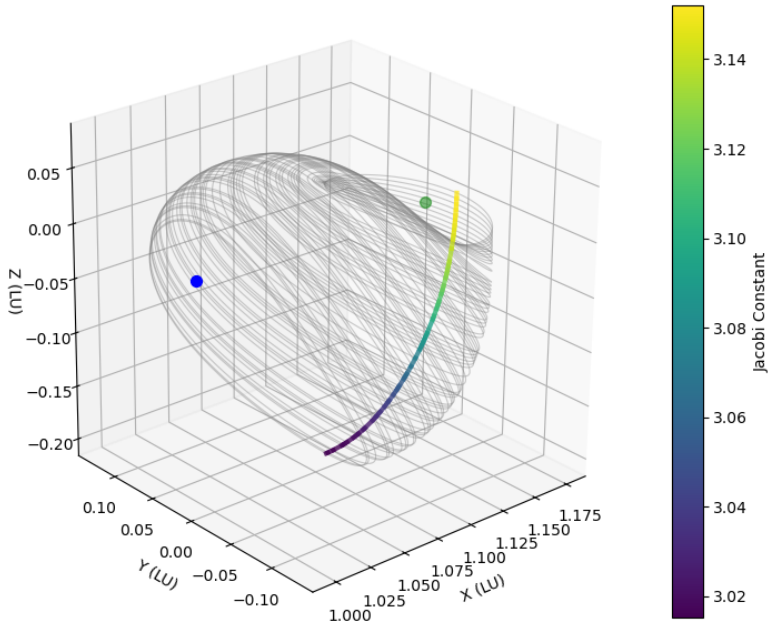


Figure 7.23: Injective representation of the $L_{2,S}$ Halo family using the Jacobi constant. The green point marks $L_{2,S}$, and the blue point marks the Moon (not to scale).

Notably, this segmentation also coincides with the isolation of the near-rectilinear halo orbit (NRHO) sub-family. NRHO orbits exhibit inherent or near-neutral stability ($\nu \approx 1$ or $1 \lessapprox \nu \lessapprox 5$), which appears in Figure 7.22 as the region of the $L_{2,S}$ Halo family characterized by $\nu \approx 1$. This high level of stability restricts the generation of rapidly diverging invariant manifolds, as such manifolds require an intrinsically unstable parent orbit.

The primary drawback of using C as a control parameter is its lack of direct physical interpretability, making it less intuitive for mission design. As a result, it does not offer immediate insight into the shape, size, or period of an orbit, in turn requiring prior knowledge of the family structure to make meaningful use of it. However for experimental purpose, it is deemed as adequate to prevent network bloating during the test phase. In a more generalized network, these parameters could be replaced with latent representations (explored in Section 7.4.4) for orbital information, and one hot encoding for manifold direction, however this is left for future implementations. Nonetheless, this segmentation possesses a range of stability values, allowing the performance of surrogate models across this range to be explored. This behaviour is illustrated in Figure 6.16, which shows that the divergence of manifold arcs associated with different parent orbits occurs at different propagation times depending on the Jacobi constant (colour-coded) and the inherent stability of the orbit (where the relationship between Jacobi constant and orbital stability for the $L_{2,S}$ Halo family is characterized in Figure 7.22). As seen in Figure 6.16, divergence occurs at earlier times for orbits with higher values of the stability metric, since larger stability values correspond to more unstable dynamics, as discussed in Section 2.7. This observation is consistent with the manifold-generation process proposed in Section 3.2.2 and illustrated in Figure 3.3.

Consequently, uniform temporal sampling of manifold arcs may bias the dataset toward chaotic regimes when the Jacobi constant of the parent orbit corresponds to an unstable orbit, which could turn lead to reduced predictive accuracy across all parent samples, not only the unstable ones.

7.7.2. Dataset Generation

As outlined in Section 6.5, the type of integrator (in terms of integration scheme, order, and selected tolerance) has a large impact on the accuracy of data generated, in combination with a computational effort. In order to ensure accurate training/test data, the DOP853 integrator is selected with a tolerance

of 10^{-13}, and a maximum allowable time-step of $\Delta t = 10^{-3}$, in order to keep the order of magnitude of the accuracy of the dataset to 10^{-9}, in accordance with Figure A.17. In order to generate a representative dataset for the surrogate model, the inputs and outputs have been sampled according to the distributions in Table 7.5. This assessment is compared to the the test case of a third-degree interpolator with an abacus resolution of 10^{-3} [ΔC] and 10 [TU] propagation time of the manifolds, as described by the numerical setup in Section 6.5.1.

Table 7.5: Neural Surrogate Control Parameters and Sampling Rates

Parameter	Description	Value/Range	Sampling
Network Constants			
κ	Class of parent orbit	$L_{2,S}$ Halo	N/A
χ	Manifold direction	Unstable Exterior	N/A
Inputs			
ξ	Relative position along parent orbit	[0, 1]	Uniform
C	Jacobi of parent orbit	[3.01, 3.15]	Uniform
t_f	Time of flight on manifold arc [TU]	[0, 10]	Uniform

7.7.3. Regression Models

An overview of the suitable regression models identified in Section 4.2, along with their corresponding search parameters, is outlined in the subsections below. These models were selected based on their demonstrated ability to capture the underlying structure of orbital trajectories while maintaining computational efficiency during training. For each model, the hyperparameter search space is detailed to illustrate the breadth of configurations evaluated during optimization.

In order to identify suitable architectures, a similar hyperparameter optimization process is performed to Section 7.3. 2 architectures are identified, and their suitability in combination with Table 7.6. The output space of these architectures is intentionally designed to provide an initial exploration of final states, enabling SSDC corrections to be applied subsequently during the iterative refinement process.

Table 7.6: Summary of the four NAS configurations.

Option Name	Network Type	L_C
DNN–NoLC	DNN	No
DNN–LC	DNN	Yes
KAN–NoLC	KAN	No

The hyperparameter search space, in combination with their sampling distribution rates are presented in Table A.3 of . Architectures sizes are preliminarily limited to those explored in literature, as identified in Section 4.2. During each trial, model performance is evaluated across multiple metrics to capture both overall prediction quality and specific physical accuracy requirements. The primary optimization objective during training is ,similarly to Section 7.3, state accuracy. These metrics were evaluated at both the final epoch, and the best performing test epoch (as an alternative to early stopping), determined by the minimum test loss. All experiments are again performed on an *Nvidia T4 Tensor GPU*.

Table 7.7: Evaluation metrics summary

Metric	Description	Logging Frequency
Train loss	Loss on training set	Epoch (Final)
Test loss	Loss on test set	Epoch (Final, Best)
Position error (train/test)	Mean absolute error in position	Epoch (Final, Best)
Velocity error (train/test)	Mean absolute error in velocity	Epoch (Final, Best)

7.7.3.1. DNN Implementation

In all cases, each hidden layer in the DNN network is immediately followed by a nonlinear activation function and a dropout layer, demonstrated by each individual block (line, dashed) in Figure 7.24. The only exception is the final output layer, which does not include an activation function or dropout layer, in order to accurately representing the full range of possible system states.

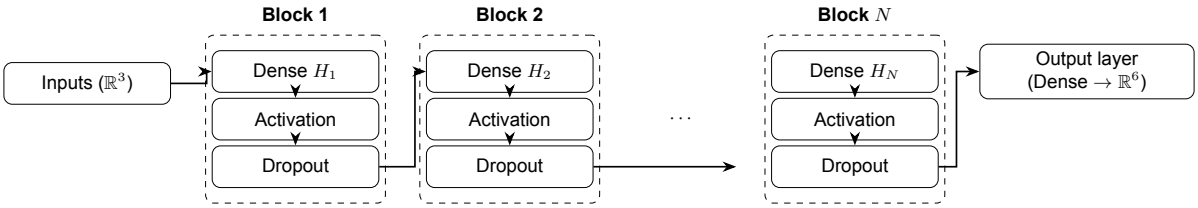


Figure 7.24: Architecture pattern used in DNN NAS. Each hidden layer applies Dense $\rightarrow$ Activation $\rightarrow$ Dropout. The final output layer omits activation and dropout to allow unrestricted (including negative) outputs.

7.7.3.2. KAN Implementation

Similarly to the DNN implementation, a NAS is again performed with for optimal hyperparameters. The search parameters, in combination with their sampling distribution is outlined below in Table A.4. Based on these hyperparameters, the KAN then constructed. It should be noted, that due to the slower training/convergence rate of KAN's with the LBFGS optimizer, the training epoch count has been increased with respect to the DNN implementation. This is in combination with reduced parameter sizes, as KAN training proves to be very memory intensive compared to DNN implementations, restricting model size during training.

7.7.3.3. Loss Landscape

The primary loss function used for evaluation is the MSE loss, computed as the average of the squared differences between the predicted state vector ($\hat{\mathbf{y}}$) and the true state vector ($\mathbf{y}$) over all samples (N) in the evaluation set:

$$\mathcal{L}_{\text{MSE}} = \frac{1}{N}\sum_{i=1}^{N}\left(\hat{\mathbf{X}}_i - \mathbf{X}_i\right)^2 \tag{7.12}$$

Similarly to Section 7.3, a loss term for L_C is implemented such that deviations of the predicted Jacobi ($\hat{C}_{\text{arc}}$) from the reference/parent Jacobi (C_{par}) is penalized for each trajectory sample, and then similarly to the MSE, an average is computed over all samples:

$$\mathcal{L}_{\text{PINN}} = \frac{1}{N_t}\sum_{i=1}^{N_t}\left(\hat{\mathbf{X}}_i - \mathbf{X}_i\right)^2 + \gamma\,\frac{1}{N_t}\sum_{i=1}^{N_t}\left(\hat{C}_{\text{arc}} - C_{\text{par}}\right)^2 \tag{7.13}$$

7.7.4. Results

Figure 7.25 present the results of the hyperparameter optimization for all explored architectures, each represented as a different colour line, similarly to Figure 6.17. Note that training times for the DNN architecture were significantly less compared to KANs, however this is not reflected in the figure and is as a result of the KAN architectures requiring CPU evaluations during training (not required at inference time, and as such not represent in inference FLOP count). None of the evaluated setups achieved accuracy comparable to the numerical integration schemes. It is also noteworthy that no regression scheme met the target accuracy of $\Delta p = 10^{-5}$ [LU], indicated by the red dashed line in the figure. Further note that at these noise levels, the expected convergence rate of differential correction schemes is $\approx 27.5\%$ per the sensitivity analysis described in Section 7.6.1.

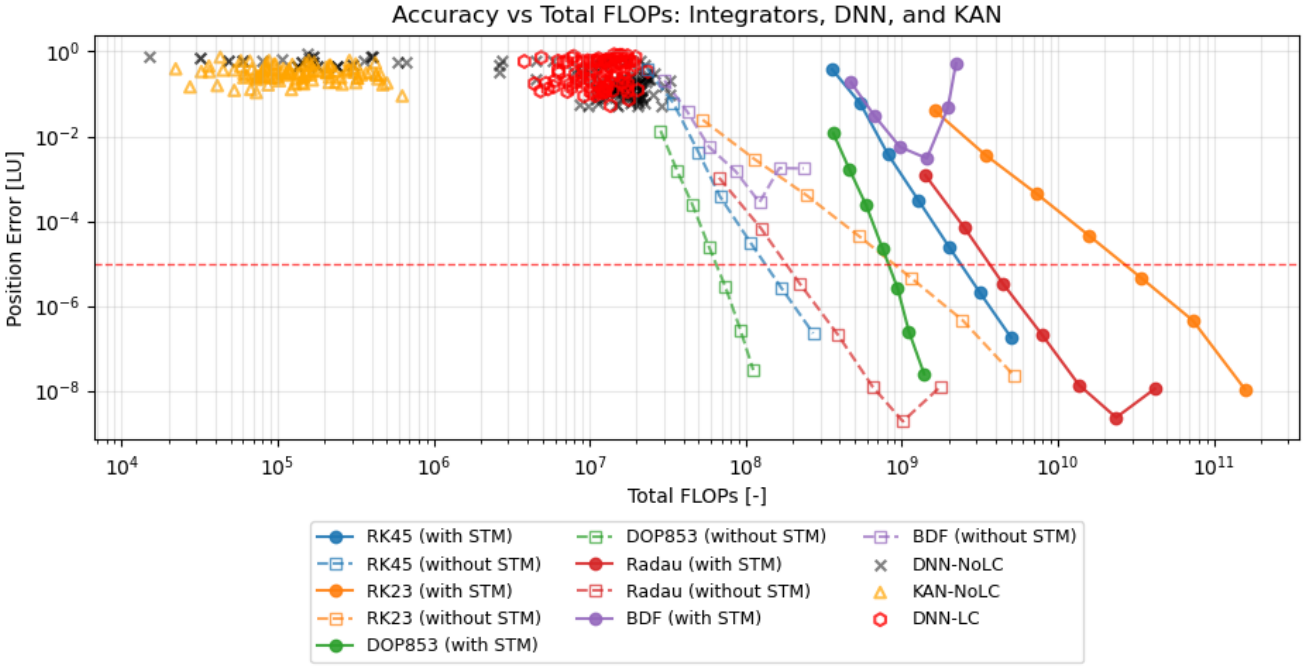


Figure 7.25: Accuracy vs. function evaluations for classical integrators (with/without STM), DNN, KAN. Red dashed line is accuracy requirement. Familial interpolation is cubic at abacus resolution 10^{-3}, 10 [TU] propagation for a single manifold arc.

Figure 7.26 presents the expected FLOP count required for each architecture to reach the desired accuracy threshold is estimated by means of a log–log linear extrapolation across the Pareto fronts per architecture type, based on neural scaling law for accuracy versus computational effort, consistent with trends observed in large neural systems [76]. This in turn shows that increasing the number of parameters and neurons generally improves surrogate accuracy across the Pareto fronts, whilst the resulting projections for these Pareto-dominant models reveal that even the most efficient explored architecture (DNN-LC, red, dashed) would require on the order of 10^9-10^{10} [FLOPs] to achieve the target accuracy. In contrast, the most efficient numerical integrator that satisfies the state-accuracy requirements, DOP853 (green, dashed in Figure 7.25), requires only $\sim 10^8$ [FLOPs]. Whilst the KAN (yellow, dashed) is more efficient compared to the baseline DNN-NoLC (blue, dashed) by approximately one order of magnitude, the DNN-LC (red, dashed) is able to achieve higher levels of accuracy at much lower computational costs. This saving corresponds to approximately $\sim 10^4$ - $\sim 10^8$ orders of magnitudes of FLOPs when comparing the DNN-LC to other neural architectures (KAN, DNN-NoLC). Furthermore, due to uncertainties in the estimated scaling laws ($R^2 < 0.9$ across all architectures), the extrapolated computational effort spans a wide range. This variability is represented by the shaded $\pm 1\sigma$ confidence bands, and suggests that the explored Pareto fronts are not fully populated.

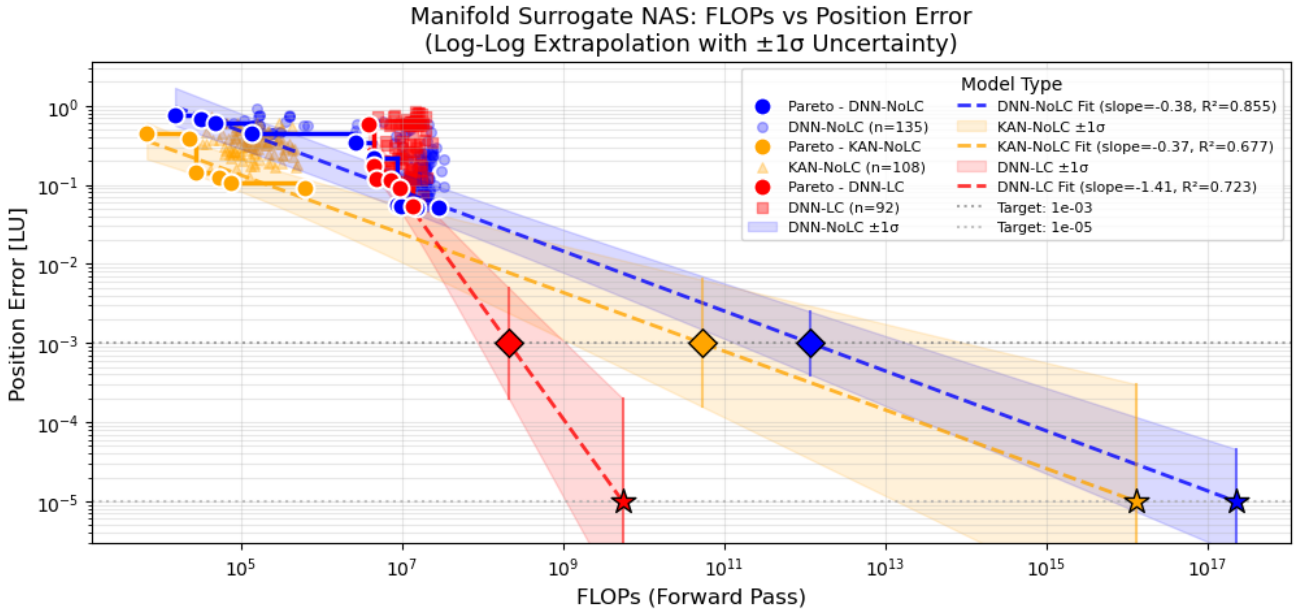


Figure 7.26: Log-Log linear neural scaling for invariant manifold surrogate models within explored architectures.

The poor scaling behaviour of the neural surrogates suggests that achieving the required accuracy targets with reasonable computational cost is currently infeasible unless those accuracy requirements are relaxed. This effect is illustrated by reducing the target accuracy to 10^{-3} [LU], for which the most efficient neural surrogate architectures become comparable to the least efficient numerical integration schemes, requiring an estimated 10^{8} [FLOPs] to achieve similar performance. Note that this is on par with the worst performing integrator for the non-STM scenario identified in Section 6.4. It is also important to consider the sensitivity of numerical schemes to both the abacus resolution and the time of flight (TOF) of the propagated manifold arc, as elaborated upon in Section 6.5.3, and demonstrated by the curvature of the contour lines in Figure 6.22. This sensitivity implies that for sufficiently long propagation times or sufficiently sparse abacus discretizations, neural surrogates could become computationally favourable. However, in precisely these regimes, the success rate of numerical integration schemes in meeting the desired accuracy criteria drops to 0, negating their reliability and making such comparisons less meaningful.

8

Neural Mission Design Pipeline, Future Research Directions

This chapter presents the proposed neural mission design pipeline using the orbit-generation results from Chapter 7. This is done by considering two datasets: a structured dataset (Dataset A), organized according to a bifurcation scheme (e.g., JPL-A) with known bifurcation nodes (Bi-Node A; see, for example, Table 6.2), and an unstructured dataset (Dataset B; e.g., the Franz–Russell database [72]). Finally, future research directions and additional applications of this work are discussed in Section 8.2.

8.1. Proposed Mission Design Pipeline

First, a VAE is trained on the combined datasets (Data = $A \cup B$). The labelled subset is then encoded to obtain latent representations of the bifurcation nodes, yielding a latent vector for each node in A. For each bifurcation type (e.g., period doubling, tripling, etc.), the mean bifurcation direction d_{attr} in latent space is computed as the average vector difference between the corresponding node pairs. These bifurcation directions enable two downstream applications: one concerned with identifying bifurcation relationships within unseen families, and another focused on supporting mission design by predicting suitable child orbits without full numerical continuation.

As a first downstream application, the latent representations are used to infer bifurcation relationships between families in Dataset B, without requiring full continuation of each family to explicitly locate the bifurcating orbits. This is achieved by comparing inter-family distances and directions in latent space with the bifurcation direction d_{attr} associated with a particular bifurcation type, as described in Section 7.4.3. The resulting edge-node clustering is illustrated in Figure 8.1, where the approach is applied to the JPL-A dataset for verification against the true bifurcation diagram shown in Figure 6.4. Relationships involving the Lagrange point (red marker in Figure 6.4) are not captured by the latent-based classification. This relationship would correspond to a connection between the L_2 Vertical and L_2 Lyapunov families, as indicated by the red dashed line.

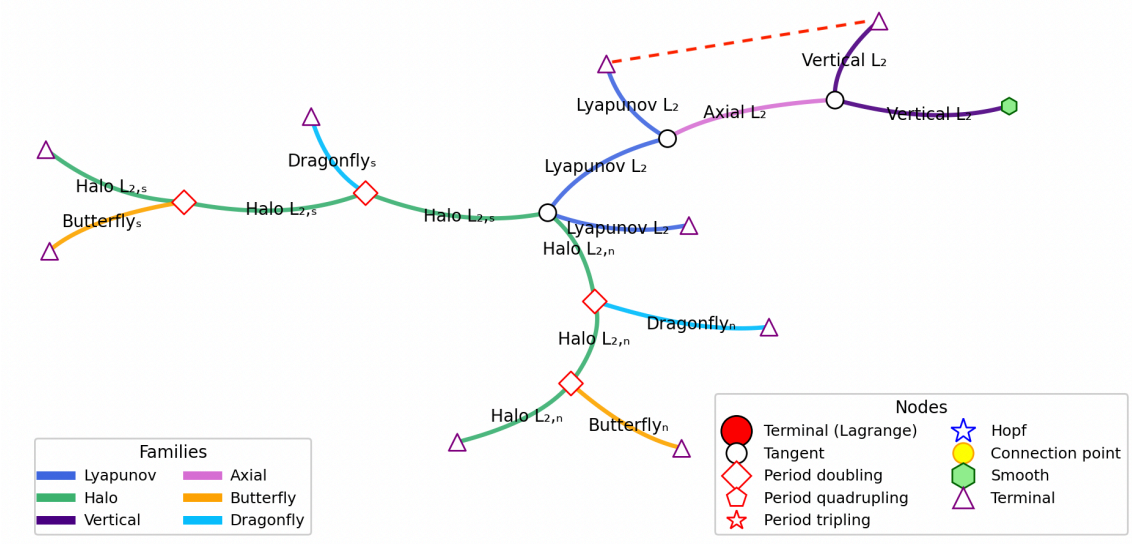


Figure 8.1: Node–edge graph of the JPL-A subset derived from VAE-based clustering, used for verification against Figure 6.4.

As a second downstream mission design application, suppose that an orbit A with favourable dynamical characteristics is identified, possessing period P and Jacobi constant C. A desirable mission design outcome may involve locating a related orbit with similar characteristics but with a period approximately $2P$. Rather than numerically continuing the entire family to identify the relevant bifurcation point, the orbit A is encoded into its latent representation z_{parent} (blue), as illustrated for the two-dimensional latent space in Figure 8.2. The bifurcation direction associated with a period-doubling transition, d_{2P} (green), is then applied to obtain a candidate child latent point z_{child} (red).

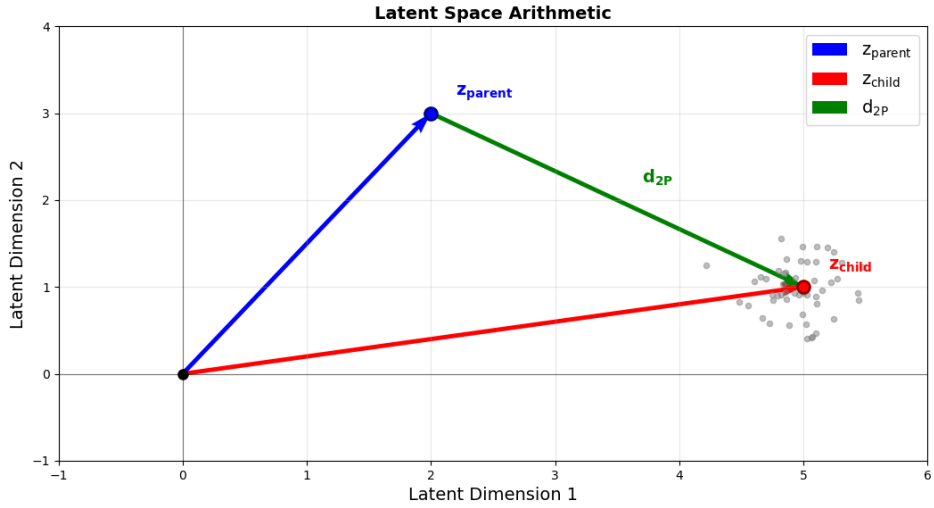
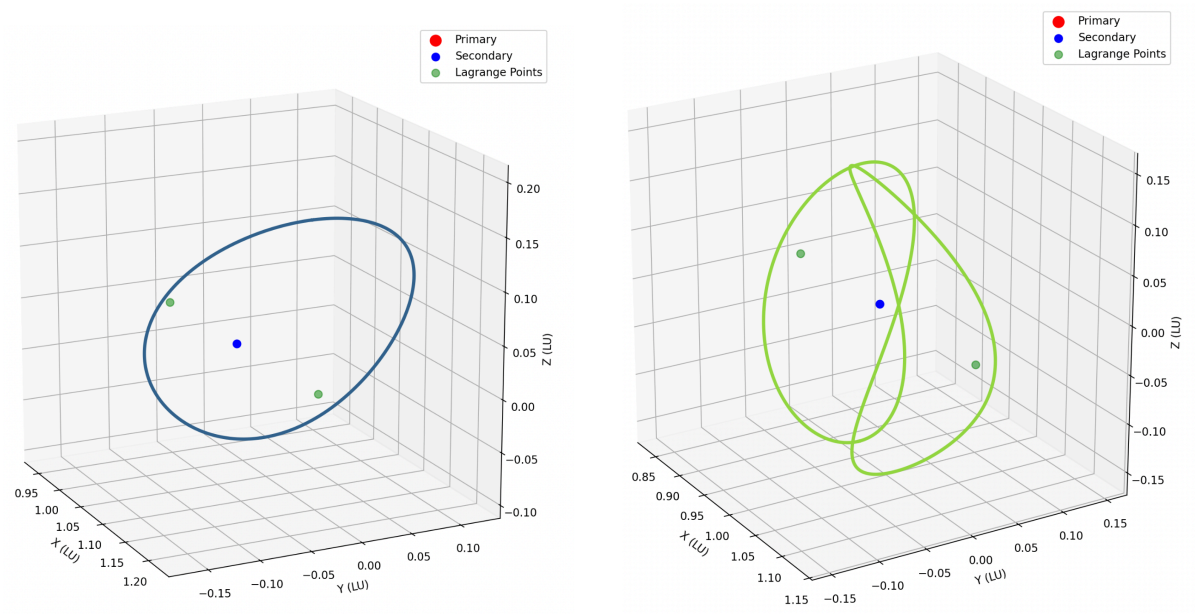


Figure 8.2: Demonstration of Latent Arithmetic for Spacecrat Mission Design

Latent vectors $\mathbf{z}$ are sampled within the shifted latent region shown in grey, decoded using the trained decoder, and subsequently refined numerically using MSDC to yield physically consistent child orbits. The inputs and outputs of this process is demonstrated in Figure 8.3, where the spatial representations of the parent ($L_{2,N}$ Halo, z_{parent}, left) and a sample decoded child orbit ($L_{2,N}$ Butterfly, z_{child}, right) are presented. Through this procedure, a Halo orbit characterized by (C, P) is transformed into a Butterfly orbit with approximately $(C, 2P)$. While the Jacobi constant C is not uniquely determinative of orbital geometry, it provides a useful indicator of the Δv required for transfer between the two orbits. In particular, transfers between orbits with similar C values may admit true heteroclinic connections, should their invariant manifolds intersect (as described by Section 2.9), enabling low-Δv transfer trajectories.



(a) $L_{2,N}$ Halo (Orbit A, z_{parent}) **(b)** $L_{2,N}$ Butterfly (Sample Orbit B, z_{child})

Figure 8.3: Latent arithmetic applied to Halo Butterfly.

Now consider a spacecraft initially operating in orbit A, with the objective of identifying a transfer to a candidate orbit B selected from the set of decoded samples. A discrete set N of manifold arcs associated with orbit A and its local neighbourhood, as well as with orbit B and its nearby samples, is numerically propagated. Because the surrogate model described in Section 7.7 is unable to capture the long-term dynamics of these manifolds, iterative optimization algorithms must instead rely on direct numerical propagation to evaluate transfer feasibility and to identify preliminary minimum $\Delta v, TOF$ trajectories between the neighbourhoods of orbits A and B.

8.2. Future Research Topics

The methods and findings in this work present several directions for future research and iteration. These opportunities span architectural enhancements, expansion or restructuring of datasets, and broader application of the underlying techniques to dynamical systems beyond the CR3BP.

Future work should investigate more accurate (and computationally intensive) decoder architectures (such as the suggested latent-based TransFusion architecture), in an effort to increase convergence rate and noise in decoded samples. Furthermore alternative prior distributions for the latent representation should be considered, as current results suggest that a Gaussian prior may not adequately capture the intrinsic structure of the data. This highlights the need for distribution matching in variational inference, particularly through the use of priors that can better reflect the thin manifold like geometry observed in the latent space [77]. Examples of this have been demonstrated in literature using Riemannian VAEs, where latent variables are learned and approximated along manifolds similar to the ones seen in this paper, however such models have not been applied in a dynamical systems context [78, 79].

Figure 8.4 presents an example of such an assumed alternative Riemannian prior (right) in comparison to the nominal isotropic Gaussian (left) for the same underlying data distribution (blue points). In the isotropic case (left), the VAE imposes equal variance in all latent directions, producing uniform, equidistant variance contours (black). By contrast, the Riemannian prior(right) introduces a trainable, geometry-aware latent variance, yielding non-equidistant contours that adapt to the structure of the data. This added flexibility allows the VAE to better capture the intrinsic, non-Gaussian geometry of the underlying distribution.

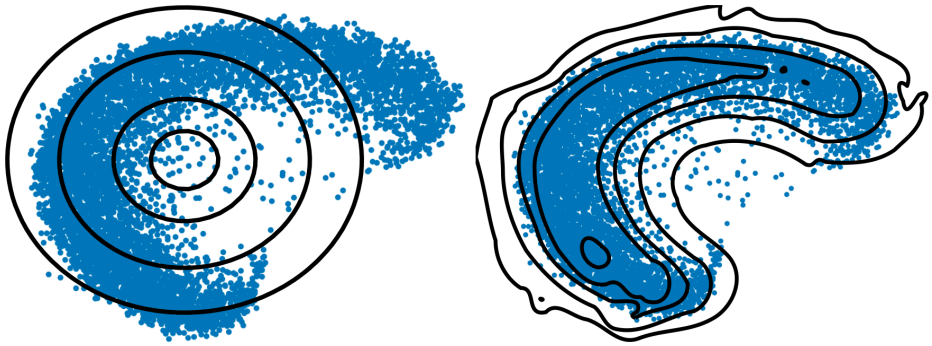


Figure 8.4: Comparison of an assumed isotropic Gaussian prior (left) and trainable Riemannian latent prior (right) for the same data distribution (blue dots) [79].

Another promising direction for future research, particularly at the architectural level, concerns mitigating normalization issues arising from large disparities in spatial and dynamical scales across different orbits and orbital families. Such discrepancies can degrade learning performance and introduce noise when global normalization is applied across heterogeneous datasets. Potential approaches include the use of adaptive or family-aware normalization schemes that adjust scaling dynamically based on orbital characteristics. An alternative strategy is to segment the decoder into multiple heads according to the characteristic scale of the input trajectories. For example, relatively small-amplitude $L_{2,N/S}$ orbits could be grouped and decoded separately, reducing noise introduced by global dataset normalization. This concept is preliminarily demonstrated in Figure 8.5, which shows spatial distributions of decoded samples (individually coloured) from a VAE trained exclusively on local L_2 periodic orbits. Compared to the generalized model in Figure 7.7, these samples exhibit a noticeable reduction in noise, visible as a lack of spatial jittering. In addition to this, implementations with adaptive weighting during the training schedule should demonstrate improvements in training stability for Jacobi loss (L_C) implementations.

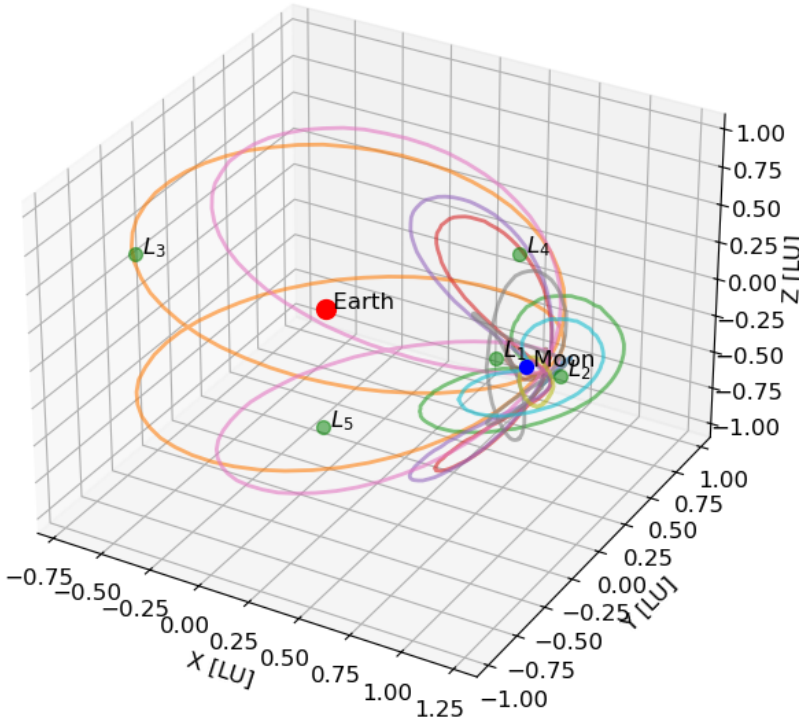


Figure 8.5: Preliminary demonstration for normalization limited to the $L_{2,N/S}$ range, decoded samples

Evaluating the framework on larger and less structured datasets would help to further validate the applicability of latent arithmetic in an astrodynamic context. Resonant orbit datasets constitute a particularly promising test case, as they would enable an investigation into whether the latent space can learn interpretable attributes associated with specific resonance types. The emergence of such structure would support controllable latent decompositions and provide additional evidence for the generality of the learned representations. Alternatively, known bifurcation patterns (e.g. Bi-Node A configuration), could be exploited as a guiding key structure to identify corresponding bifurcations within unknown datasets (e.g. Dataset B), as outlined in Section 8.1.

The primary distinction between the CR3BP and other dynamical systems lies in the governing equations rather than in the nature of the trajectories themselves. Many physical systems across a wide range of scales exhibit structured dynamical behaviour governed by well-defined laws, including oscillatory systems (e.g. vehicle or structural dynamics), electromagnetic control systems, and molecular kinetics. One example of such an alternative dynamical system is found in electromagnetic control schemes used to accelerate electrons in plasma fields [80]. Figure 8.6 illustrates such a system, where the trajectory of an electron (solid line) is shown in the (Z,R) plane alongside contours (dashed) of constant magnetic flux. The particle follows a periodic orbit induced by the magnetic field configuration, enabling controlled acceleration while remaining confined within a bounded spatial region. Consequently, the methods developed in this work could be extended to these domains with minimal architectural modification, given sufficient data. More broadly, applying these techniques could facilitate automated exploration, classification, bifurcation analysis, and controlled manipulation of periodic orbits across a wide range of dynamical environments.

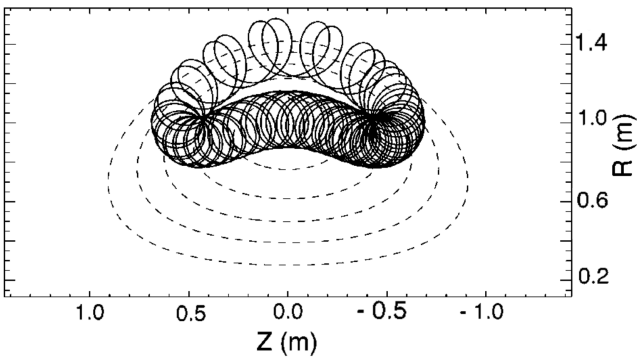


Figure 8.6: Periodic trajectory of an electron in an electromagnetic control system [80][Modified]

9

Conclusion

This work investigated the capability of AI generative and regression models to support and accelerate exploration of periodic orbits and invariant manifolds within the CR3BP. Motivated by the high computational cost and sensitivity of numerical integration for both familial continuation and invariant manifolds, emerging AI techniques are explored as potential surrogates or accelerators within mission design pipelines.

Generative architectures (dense, convolutional, and TimeVAE's) demonstrated capacity to reconstruct and sample periodic orbits with strong geometric similarity to reference data. Raw generated trajectories did not satisfy periodicity or dynamical constraints, requiring MSDC refinement to ensure physical validity. Embedding the Jacobi constant within the loss function (L_C) improved dynamical consistency and energy preservation, facilitating the use of smaller (parameter wise) models, but could not fully replace post-processing. All architectures achieved comparable performance, as measured by phase-state reconstruction accuracy of $\approx 8 - 9 \cdot 10^{-3}$ [–], whilst the baseline dense architecture provided the best performance to model size ratio. Larger and more varied datasets (9 vs 31 families) required more parameters to be able to obtain minimal accuracy levels. Latent-space structure analysis revealed coherent organization of orbital families and bifurcation relationships, indicating that the learned latent representations captured meaningful dynamical information. Further exploration however is required for statistical validation for the implementation of latent arithmetic due to the small amount of period doubling bifurcations explored in the dataset. The regression models proposed, DNN's (with and without L_C), and KAN's demonstrated the ability to act as a neural surrogates for manifold arc propagation, however not to an accuracy level that is acceptable (10^{-5}) in preliminary CR3Bp mission design prior to transfer to the EHF. Whilst the surrogate models demonstrated the ability for convergence using MSDC in short term regimes (period orbits), they do not remain a viable candidate for long term approximations of CR3BP dynamics for use in optimization schemes with SSDC in such a way that would make them computationally advantageous. State accuracy requirements in position and velocity requirements ($< 10^{-5}$[-]) dominated energy requirements measured by change in Jacobi, resulting in the energy requirement being discontinued for this implementation. The following section evaluates how the outcomes of this investigation resolve the research questions posed in Chapter 5:

RQ1. **To what extent can different VAE types, both with and without energy conservation mechanisms, be adapted for periodic orbit generation in the CR3BP?**

RQ1.1. To what extent can different generative model architectures (dense, convolutional, timeVAE's, and TransFusion) reduce the level of post-processing refinement required to generate dynamically valid periodic orbits?

All architectures explored were able to achieve state reconstruction errors on the order of magnitude of $\approx 8 - 9 \cdot 10^{-3}[-]$ for a less varied dataset with 9 different families (JPL-A), and $\approx 1 - 2 \cdot 10^{-2}$ [-] for a more varied dataset with 31 families (JPL). Convolutional and timeVAE implementations required more parameters to obtain these accuracies then the dense baseline, as demonstrated by Figure 7.2 and Table 7.3. Computationally expensive

TransFusion implementations achieved similar accuracy levels to hyperparameter-tuned VAE models without hyperparameter tuning, at the cost of significantly ($\approx 50\times$) more computational effort during training. This indicates noise and accuracy yields are to be obtained here, at the cost of significantly more computational effort, as as such should be limited to decoder-only implementations reliant on latent representations trained by VAEs to mitigate this computational cost.

RQ1.2. What is the comparative efficiency of generative models vs classical continuation methods for in family interpolation in the CR3BP?

It remains more computationally efficient to interpolate within families using SSDC implementations, should initial familial continuation schemes resolve the orbital abacus of a family to levels higher than $\Delta C > 3 \cdot 10^{-3}[-]$, as demonstrated by Figure 6.21. Generative schemes however facilitate mass parallelization for preliminary analysis, in combination with the possibility of interpolation between multiple families without the use of multiple abacuses. MSDC post-processing however is not mass parrelelizable, and as such the amount of noise in decoded samples limits full mass-parallelization.

RQ1.3. Can latent sampling facilitate extrapolation to unseen (bifurcated) families?

When assessing models only trained on JPL-A, a 9 family subset of JPL, no decoded prior samples were found to be in any sample not seen in the training dataset, including families in the full JPL set. Only after MSDC refinement did some sampled trajectories converge to orbits outside the initially decoded family, but this behaviour is more reflective of noise sensitivity in MSDC convergence than of genuine generative generalization. Further exploration of alternative latent priors, together with training on more diverse datasets in which a larger number of classes enables the models to better resolve inter-family relationships, constitutes a promising research direction for assessing broader generalizability.

RQ1.4. To what extent does the inclusion of physical (energy) constraints within generative models improve orbit feasibility and computational efficiency?

The inclusion of a loss term penalizing discrepancies in sequence Jacobi did not facilitate a measurable improvement in the accuracy of the model, with all architectures measured to have similar maximum achievable performance with and without the inclusion of the L_C term in the hyperparameter optimization regions tested. It did however facilitate a measurable impact on the ratio of performance to model size, with models with similar performance where L_C is included having typically 30-300% less parameters for all tested architectures. Increased parameter density suggests quicker scaling of model accuracy with model size through the inclusion of this constraint.

RQ1.5. Can learned latent representations approximate physical bifurcation relationships between orbital families?

Family-based clustering in the latent space is demonstrated in Figure 7.8 and Figure 7.9. Continuous orbits within the same family exhibit correspondingly continuous latent representations. Tangentially bifurcated families show high latent similarity, quantified via cosine similarity and vector-norm ratios, particularly when the bifurcation does not produce families with a significant out-of-plane (z) component. Additionally, latent representations of child families arising from period-doubling bifurcation types from the same parent family display strong similarity. Further study using more diverse bifurcation behaviours is needed to assess the generalizability of these trends.

RQ1.6. Can vector arithmetic be applied to latent space representations for both mission design and classification purposes?

Latent similarity is shown to be useful for identifying and classifying closely related orbits within the same family. In a mission-design context, sampling the latent region surrounding an encoded orbit enables efficient exploration of nearby, dynamically similar trajectories. The latent space also captures tangential bifurcation relationships between families, demonstrated for three cases, both the $L_{2,N/S}$ Halo and L_2 Lyapunov families, and the L_2 Vertical and L_2 Axial families.

Preliminary analysis of assigning specific vector arithmetic to represent particular operations suggests that period-doubling bifurcations correspond to identifiable vectors in the latent space. These difference vectors exhibit high cosine similarity and closely matched magnitude ratios, with maximum deviations of 0.96 (cosine) and 1.12 (norm ratio) from the mean vector associated with period-doubling bifurcations. In the context of mission design, this enables rapid identification of orbits with similar spatial characteristics, dynamical characteristics, and energy levels (C), across orbits with differing period multiples without requiring full familial and bifurcation continuation. This remains to be verified due to small sample size seen in the dataset, with only 4 period-doubling bifurcations present. Classification of datasets lacking known bifurcation relationships between orbit families remains to be demonstrated, though a method for addressing this challenge is proposed in Section 8.2.

RQ2. **To what extent can DNN's and KAN's, with and without energy conservation mechanisms, be used for computationally efficient regression of manifold arcs?**

RQ2.1. What is the associated accuracy of DNN's with and without energy preservation mechanisms (PINN-losses) and KAN's, in terms of spatial, dynamical and energy errors for the regression of invariant manifold transfer trajectories?

Within the computational budget considered, defined by the FLOP count of the cheapest numerical integration scheme capable of achieving the required state-accuracy threshold of 10^{-5}[-], the neural surrogates did not attain sufficient accuracy to be used for invariant-manifold–based transfer trajectory optimization. The best-performing model, the Dense DNN with the L_C term, achieved accuracy only on the order of $5 \cdot 10^{-2}$[-], far above the mission-design requirement. KANs demonstrated roughly an order of magnitude greater efficiency compared to the baseline DNN architecture for similar accuracy levels. Neural-scaling law extrapolation indicates that achieving the necessary accuracy would require architectures with FLOP costs orders of magnitude larger than those of the numerical integrators themselves, eliminating any computational advantage during the exploration phase, in combination with the extra data needed to compare them. Although these surrogate models can be parallelized efficiently, their accuracy limitations prevent practical deployment in mission design. Relaxing accuracy requirements could make them competitive only with the poorest-performing numerical schemes. Consequently, SSDC-based optimization using neural surrogates remains unsuitable for long-term CR3BP trajectory analysis, especially when compared to MSDC convergence performance demonstrated for short-term trajectories in the VAE-based pipeline.

RQ2.2. How does the inclusion of energy preservation mechanisms facilitate the accuracy and computational efficiency of DNN regression techniques, in the context of invariant manifold approximation?

Similar to the short-term propagation results observed in the VAE implementation, the Pareto front for the DNN with the L_C term indicates that a substantial reduction in required model size and inference cost measured in FLOPs, can be achieved through the inclusion of the energy-preservation term. This effect becomes more pronounced when extrapolating to larger model sizes; however, within the tested range, the improvement remained negligible because no high-accuracy architectures were obtained in that region.

In summary, this work demonstrates both the promise and limitations of AI-driven approaches for trajectory generation and manifold prediction in the CR3BP in a spacecraft mission design context. Generative models offer clear value for rapid exploration of periodic-orbit families, latent-space organization, in the context of preliminary mission-design analysis, particularly when aided by MSDC refinement and energy-aware loss formulations. Regression-based surrogate models, while computationally efficient and highly parallelizable compared to their numerical counterparts, remain insufficiently accurate for manifold-based optimization and thus cannot yet replace classical numerical integrators in mission-design settings, even in combination with SSDC. The future research directions outlined in Section 8.2 aim to close this gap through improved decoder architectures, alternative latent priors, expanded datasets, adaptive normalization schemes, and broader applications to other dynamical systems.

References

[1] Xiaoming Li and Shijun Liao. “More than six hundreds new families of Newtonian periodic planar collisionless three-body orbits”. In: (Apr. 2017). doi: 10.1007/s11433-017-9078-5.

[2] Shijun Liao, Xiaoming Li, and Yu Yang. “Three-body problem – from Newton to supercomputer plus machine learning”. In: (May 2021). doi: 10.1016/j.newast.2022.101850.

[3] Tjarda Boekholt and Simon Portegies Zwart. “On the Reliability of N-body Simulations”. In: (Nov. 2014).

[4] Tjarda C. N. Boekholt, Arend Moerman, and Simon F. Portegies Zwart. “The relativistic Pythagorean three-body problem”. In: (Sept. 2021). doi: 10.1103/PhysRevD.104.083020.

[5] G. De Cesare and R. Capuzzo-Dolcetta. “On the stability of planetary orbits in binary star systems I. The S-type orbits”. In: *Astrophysics and Space Science* 366 (6 June 2021). issn: 1572946X. doi: 10.1007/s10509-021-03959-x.

[6] Mario Juri´c Juri´c and Scott Tremaine. *DYNAMICAL ORIGIN OF EXTRASOLAR PLANET ECCENTRICITY DISTRIBUTION*. Tech. rep. 2008.

[7] Yanqin Wu and Yoram Lithwick. “Secular chaos and the production of hot Jupiters”. In: *Astrophysical Journal* 735 (2 July 2011). issn: 15384357. doi: 10.1088/0004-637X/735/2/109.

[8] Alessandro Alberto Trani et al. “Isles of regularity in a sea of chaos amid the gravitational three-body problem”. In: *Astronomy and Astrophysics* 689 (Sept. 2024). issn: 14320746. doi: 10.1051/0004-6361/202449862.

[9] Juan Senent. *POINCARE: A MULTI-BODY, MULTI-SYSTEM TRAJECTORY DESIGN TOOL*. Tech. rep.

[10] European Space Agency, Advanced Concepts Team. *Machine Learning for Trajectory Design in Multi-Body Dynamics*. https://www.esa.int/gsp/ACT/projects/ml_for_cr3bp/. Mar. 2023.

[11] Shobhit Jain and George Haller. “How to compute invariant manifolds and their reduced dynamics in high-dimensional finite element models”. In: *Nonlinear Dynamics* 107 (2 Jan. 2022), pp. 1417–1450. issn: 1573269X. doi: 10.1007/s11071-021-06957-4.

[12] Shane Ross. “The Interplanetary Transport Network”. In: *American Scientist* 94 (3 2006), p. 230. issn: 0003-0996. doi: 10.1511/2006.59.230.

[13] Flavio Tagliaferri et al. “Global Optimization for Trajectory Design via Invariant Manifolds in the Earth-Moon Circular Restricted Three-Body Problem”. In: (May 2024).

[14] Philip G. Breen et al. “Newton vs the machine: solving the chaotic three-body problem using deep neural networks”. In: (Oct. 2019). doi: 10.1093/mnras/staa713.

[15] Adrien Kuntz, Francesco Serra, and Enrico Trincherini. “Effective two-body approach to the hierarchical three-body problem”. In: (Apr. 2021). doi: 10.1103/PhysRevD.104.024016.

[16] Kedron Silsbee and Scott Tremaine. “Lidov-Kozai Cycles with Gravitational Radiation: Merging Black Holes in Isolated Triple Systems”. In: (Aug. 2016). doi: 10.3847/1538-4357/aa5729.

[17] Jeremy Goodman. “On the Exponential Instability of N-Body Systems”. In: (1993).

[18] Simon Portegies Zwart and Tjarda Boekholt. “Numerical verification of the microscopic time reversibility of Newton’s equations of motion: Fighting exponential divergence”. In: (Feb. 2018). doi: 10.1016/j.cnsns.2018.02.002.

[19] William Gilpin. “Chaos as an interpretable benchmark for forecasting and data-driven modelling”. In: (Oct. 2021).

[20] Alvaro Francisco Gil et al. “Generative Design of Periodic Orbits in the Restricted Three-Body Problem”. In: (Aug. 2024). doi: 10.5281/zenodo.13885649.

[21] Pierfrancesco Di Cintio and Alessandro Alberto Trani. “Partial suppression of chaos in relativistic three-body problems”. In: (Oct. 2024). doi: 10.1051/0004-6361/202452678.

[22] Gordon I. Ogilvie. “Tidal dissipation in stars and giant planets”. In: (June 2014). doi: 10.1146/annurev-astro-081913-035941.

[23] F. Marzari and G. Gallina. “Stability of multiplanet systems in binaries”. In: *Astronomy and Astrophysics* 594 (Oct. 2016). issn: 14320746. doi: 10.1051/0004-6361/201628342.

[24] Eric Zhang, Smadar Naoz, and Clifford M. Will. “A Stability Timescale for Non-Hierarchical Three-Body Systems”. In: (Jan. 2023). doi: 10.3847/1538-4357/acd782.

[25] Francesco Marzari. “Planet-planet scattering in systems of multiple planets of unequal mass”. In: (Nov. 2024).

[26] NASA. *Gateway: Supporting Artemis Missions to the Moon*. https://www.nasa.gov/mission/gateway/. Accessed: 2025-04-17. Apr. 2025.

[27] Jet Propulsion Laboratory. *Three-Body Periodic Orbits Tool*. Accessed: 2025-04-17. 2025. url: https://ssd.jpl.nasa.gov/tools/periodic_orbits.html.

[28] Vishwa Shah and Ryne Beeson. “RAPID APPROXIMATION OF INVARIANT MANIFOLDS USING MACHINE LEARNING METHODS”. en. In: ().

[29] Stijn De Smet and Daniel J. Scheeres. “Identifying heteroclinic connections using artificial neural networks”. In: *Acta Astronautica* 161 (Aug. 2019), pp. 192–199. issn: 00945765. doi: 10.1016/j.actaastro.2019.05.012.

[30] Davide Guzzetti. *Reinforcement Learning and Topology of Orbit Manifolds for Stationkeeping of Unstable Symmetric Periodic Orbits AAS 19-680 REINFORCEMENT LEARNING AND TOPOLOGY OF ORBIT MANIFOLDS FOR STATIONKEEPING OF UNSTABLE SYMMETRIC PERIODIC ORBITS*. Tech. rep.

[31] Stijn De Smet and Daniel J. Scheeres. “Identifying heteroclinic connections using artificial neural networks”. en. In: *Acta Astronautica* 161 (Aug. 2019), pp. 192–199. issn: 00945765. doi: 10.1016/j.actaastro.2019.05.012. url: https://linkinghub.elsevier.com/retrieve/pii/S0094576518320745 (visited on 08/14/2025).

[32] Jeffrey S. Parker and Rodney L. Anderson. *Low□Energy Lunar Trajectory Design*. en. 1st ed. Wiley, June 2014. isbn: 978-1-118-85387-0 978-1-118-85506-5. doi: 10.1002/9781118855065. url: https://onlinelibrary.wiley.com/doi/book/10.1002/9781118855065 (visited on 08/14/2025).

[33] E. J. Doedel et al. “ELEMENTAL PERIODIC ORBITS ASSOCIATED WITH THE LIBRATION POINTS IN THE CIRCULAR RESTRICTED 3-BODY PROBLEM”. en. In: *International Journal of Bifurcation and Chaos* 17.08 (Aug. 2007), pp. 2625–2677. issn: 0218-1274, 1793-6551. doi: 10.1142/S0218127407018671. url: https://www.worldscientific.com/doi/abs/10.1142/S0218127407018671 (visited on 08/14/2025).

[34] Emily M Zimovan Spreen. “TRAJECTORY DESIGN AND TARGETING FOR APPLICATIONS TO THE EXPLORATION PROGRAM IN CISLUNAR SPACE”. en. In: ().

[35] Emmanuel Blazquez et al. *SEMpy: a Python Open-Source Toolbox for Non-Keplerian Astrodynamics*. Tech. rep. 2021. url: https://www.researchgate.net/publication/369658080.

[36] Matthew J Bolliger et al. *CISLUNAR MISSION DESIGN: TRANSFERS LINKING NEAR RECTILINEAR HALO ORBITS AND THE BUTTERFLY FAMILY STATEMENT OF THESIS APPROVAL*. Tech. rep. 2019.

[37] Wang Sang Koon et al. *Heteroclinic Connections between Periodic Orbits and Resonance Transitions in Celestial Mechanics*. Tech. rep. 1997.

[38] Jeannette Heiligers. *Homo-and Heteroclinic Connections in the Spatial Solar-Sail Earth-Moon Three-Body Problem*. Tech. rep. 2019, pp. 1593–1612.

[39] Carolin Frueh et al. “Cislunar Space Situational Awareness”. In: *Proceedings of the AAS/AIAA Astrodynamics Specialist Conference*. AAS 21-290. 2021.

[40] Marisa Exnicious, Cody Short, and Teresa Nicole Brooks. “Exploring an AI/ML Approach to Automating the Transition from the Circular Restricted Three-Body Problem to Higher-Fidelity Solutions”. In: *Proceedings of the AAS/AIAA Astrodynamics Specialist Conference*. AAS 25-267. 2025.

[41] J. Bas Fernandez. *Optimal Low-Thrust Transfers Between Halo Orbits in the Circular Restricted Three-Body Problem*. 2024.

[42] Alexander J. Dittmann. “Modified Hermite Integrators of Arbitrary Order”. In: (June 2020). doi: 10.1093/mnras/staa1631.

[43] Shijun Liao. “On the reliability of computed chaotic solutions of nonlinear differential equations”. In: (Jan. 2009). doi: 10.1111/j.1600-0870.2009.00402.x.

[44] Leone Costi et al. “The Drosophila Connectome as a Computational Reservoir for Time-Series Prediction”. In: *Biomimetics* 10.5 (2025). issn: 2313-7673. doi: 10.3390/biomimetics10050341. url: https://www.mdpi.com/2313-7673/10/5/341.

[45] Hannah Lu and Daniel M. Tartakovsky. “Extended dynamic mode decomposition for inhomogeneous problems”. In: *Journal of Computational Physics* 444 (Nov. 2021). issn: 10902716. doi: 10.1016/j.jcp.2021.110550.

[46] George Nehma, Madhur Tiwari, and Manasvi Lingam. “Deep Learning Based Dynamics Identification and Linearization of Orbital Problems using Koopman Theory”. In: (Mar. 2024).

[47] Sam Greydanus, Misko Dzamba, and Jason Yosinski. “Hamiltonian Neural Networks”. In: (June 2019).

[48] Veronica Saz Ulibarrena et al. “A hybrid approach for solving the gravitational N-body problem with Artificial Neural Networks”. In: (Oct. 2023). doi: 10.1016/j.jcp.2023.112596.

[49] Ippocratis D. Saltas and Georgios Lukes-Gerakopoulos. “A deep classifier of chaos and order in Hamiltonian systems of two degrees of freedom”. In: (Feb. 2024).

[50] Miles Cranmer et al. “Lagrangian Neural Networks”. In: (Mar. 2020).

[51] Bin Yang et al. “Fast Solver for J2-Perturbed Lambert Problem Using Deep Neural Network”. In: *Journal of Guidance, Control, and Dynamics* 45 (5 May 2022), pp. 875–884. issn: 15333884. doi: 10.2514/1.G006091.

[52] Jacob Varey et al. “Physics-Informed Neural Networks for Satellite State Estimation”. In: (Mar. 2024). doi: 10.1109/AERO58975.2024.10521414.

[53] Caspar Meijer and Lydia Y Chen. *The Rise of Diffusion Models in Time-Series Forecasting*. Tech. rep. 2023.

[54] Weibo Mao et al. *Leapfrog Diffusion Model for Stochastic Trajectory Prediction*. Tech. rep.

[55] Tyler Presser et al. “Diffusion Models for Generating Ballistic Spacecraft Trajectories”. In: (May 2024).

[56] Abhyuday Desai et al. *TimeVAE: A Variational Auto-Encoder for Multivariate Time Series Generation*. en. arXiv:2111.08095 [cs]. Dec. 2021. doi: 10.48550/arXiv.2111.08095. url: http://arxiv.org/abs/2111.08095 (visited on 08/14/2025).

[57] Jan-Hendrik Bastek, WaiChing Sun, and Dennis M. Kochmann. “Physics-Informed Diffusion Models”. In: (Mar. 2024).

[58] D. S. Tropashko. *Word Embedding Demo: Tutorial*. https://www.cs.cmu.edu/~dst/WordEmbeddingDemo/tutorial.html.

[59] Tiago F.L.L. Pinheiro, Rafael Sfair, and Giovana Ramon. “Machine-learning approach for mapping stable orbits around planets”. In: *Astronomy and Astrophysics* 693 (Jan. 2025). issn: 14320746. doi: 10.1051/0004-6361/202451831.

[60] Florian Lalande and Alessandro Alberto Trani. “Predicting the Stability of Hierarchical Triple Systems with Convolutional Neural Networks”. In: (June 2022). doi: 10.3847/1538-4357/ac8eab.

[61] Nicolas Payot et al. “Active learning meets fractal decision boundaries: a cautionary tale from the Sitnikov three-body problem”. In: (Nov. 2023).

[62] Thiago N.C. Cardoso et al. “Ranked batch-mode active learning”. In: *Information Sciences* 379 (Feb. 2017), pp. 313–337. issn: 00200255. doi: 10.1016/j.ins.2016.10.037.

[63] Ziming Liu et al. *KAN: Kolmogorov-Arnold Networks*. en. arXiv:2404.19756 [cs]. Feb. 2025. doi: 10.48550/arXiv.2404.19756. url: http://arxiv.org/abs/2404.19756 (visited on 08/14/2025).

[64] Ashish Pal and Satish Nagarajaiah. *KAN/MultKAN with Physics-Informed Spline fitting (KAN-PISF) for ordinary/partial differential equation discovery of nonlinear dynamic systems*. en. arXiv:2411.11801 [physics]. Nov. 2024. doi: 10.48550/arXiv.2411.11801. url: http://arxiv.org/abs/2411.11801 (visited on 08/14/2025).

[65] David Valle et al. “AI-Driven Control of Chaos: A Transformer-Based Approach for Dynamical Systems”. In: (Dec. 2024).

[66] Wahyudin P. Syam et al. “Transformer deep learning for accurate orbit corrections in real-time”. In: *Proceedings of the 36th International Technical Meeting of the Satellite Division of the Institute of Navigation, ION GNSS+ 2023*. Institute of Navigation, 2023, pp. 159–174. isbn: 9780936406350. doi: 10.33012/2023.19294.

[67] Tommaso Guffanti and Simone D’Amico. “Transformers for Trajectory Optimization with Application to Spacecraft Rendezvous”. In: (Apr. 2024).

[68] Julia Briden et al. “Diffusion Policies for Generative Modeling of Spacecraft Trajectories”. In: (Jan. 2025).

[69] Alessandro Corbetta and Thomas Geert de Jong. “How neural networks learn to classify chaotic time series”. In: *Chaos* 33 (12 Dec. 2023). issn: 10897682. doi: 10.1063/5.0160813.

[70] Kenza Boudad, Kathleen C. Howell, and Diane C. Davis. “Analogs for Earth-Moon Halo Orbits and their Evolving Characteristics in Higher-Fidelity Force Models”. en. In: *AIAA SCITECH 2022 Forum*. San Diego, CA & Virtual: American Institute of Aeronautics and Astronautics, Jan. 2022. isbn: 978-1-62410-631-6. doi: 10.2514/6.2022-1276. url: https://arc.aiaa.org/doi/10.2514/6.2022-1276 (visited on 08/14/2025).

[71] Marisa Exnicious, Cody Short, and Teresa Nicole Brooks. “EXPLORING AN AI/ML APPROACH TO AUTOMATING THE TRANSITION FROM THE CIRCULAR RESTRICTED THREE-BODY PROBLEM TO HIGHER-FIDELITY SOLUTIONS, PART 2”. en. In: ().

[72] Carter J. Franz and Ryan P. Russell. “Database of Planar and Three-Dimensional Periodic Orbits and Families Near the Moon”. en. In: *The Journal of the Astronautical Sciences* 69.6 (Dec. 2022), pp. 1573–1612. issn: 2195-0571. doi: 10.1007/s40295-022-00361-9. url: https://link.springer.com/10.1007/s40295-022-00361-9 (visited on 08/14/2025).

[73] Ioannis Sakiotis et al. *PAGANI: A Parallel Adaptive GPU Algorithm for Numerical*. 2021. arXiv: 2104.06494 [cs.DC]. url: https://arxiv.org/abs/2104.06494.

[74] J. Zuntz et al. “CosmoSIS: Modular cosmological parameter estimation”. In: *Astronomy and Computing* 12 (Sept. 2015), pp. 45–59. issn: 2213-1337. doi: 10.1016/j.ascom.2015.05.005. url: http://dx.doi.org/10.1016/j.ascom.2015.05.005.

[75] Alvaro Francisco Gil et al. “Generative Design of Periodic Orbits in the Restricted Three-Body Problem”. en. In: (Oct. 2024). arXiv:2408.03691 [cs]. doi: 10.5281/zenodo.13885649. url: http://arxiv.org/abs/2408.03691 (visited on 08/14/2025).

[76] Jared Kaplan et al. “Scaling Laws for Neural Language Models”. In: *CoRR* abs/2001.08361 (2020). arXiv: 2001.08361. url: https://arxiv.org/abs/2001.08361.

[77] Mihaela Rosca, Balaji Lakshminarayanan, and Shakir Mohamed. *Distribution Matching in Variational Inference*. 2019. arXiv: 1802.06847 [stat.ML]. url: https://arxiv.org/abs/1802.06847.

[78] Nina Miolane and Susan Holmes. *Learning Weighted Submanifolds with Variational Autoencoders and Riemannian Variational Autoencoders*. 2019. arXiv: 1911.08147 [stat.ML]. url: https://arxiv.org/abs/1911.08147.

[79] Dimitris Kalatzis et al. *Variational Autoencoders with Riemannian Brownian Motion Priors*. 2020. arXiv: 2002.05227 [cs.LG]. url: https://arxiv.org/abs/2002.05227.

[80] Elena Belova, N. Gorelenkov, and Chio Cheng. “Self-consistent equilibrium model of low aspect-ratio toroidal plasma with energetic beam ions”. In: *Physics of Plasmas* 10 (Aug. 2003), pp. 3240–3251. doi: 10.1063/1.1592155.

A

Supplemental Figures and Tables

A.1. Families of Periodic Orbits in JPL-A

The following sections present the three-dimensional projections of 15 orbits per family, and Broucke stability diagrams for each orbit family, coloured by Jacobi constant. All bifurcation nodes identified in the JPL-A subset using the Broucke stability analysis are also shown. Where necessary, a zoomed-in version of the Broucke stability diagram is additionally provided.

A.1.1. $L_{2,N}$ Halo

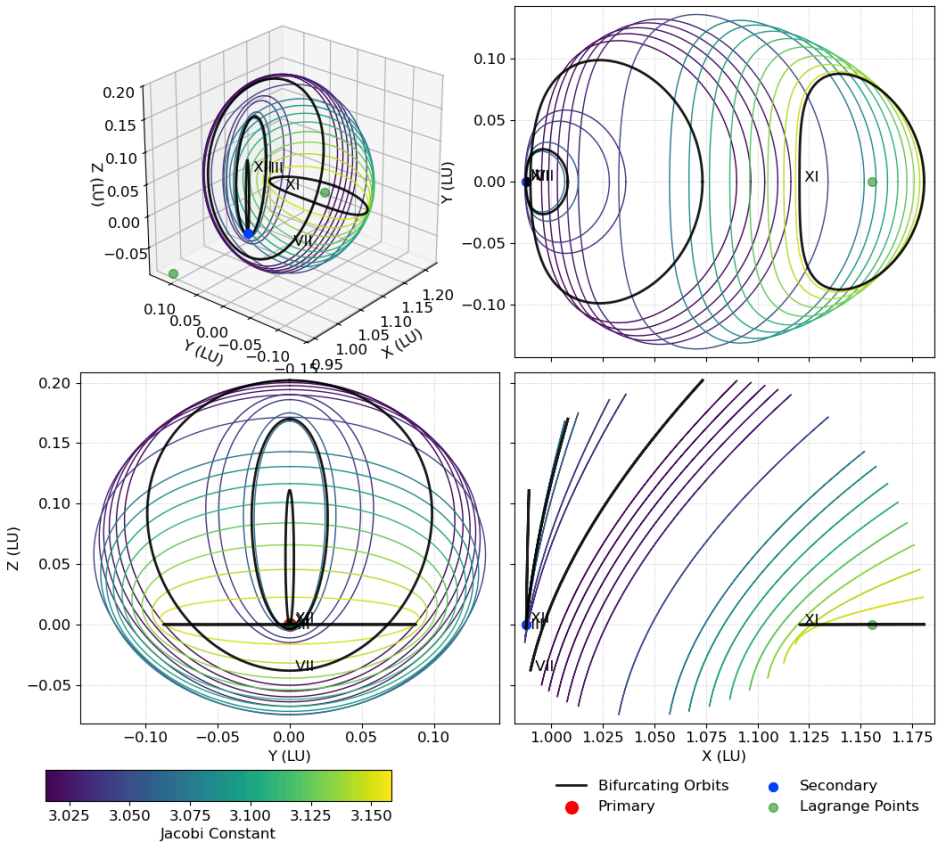


Figure A.1: Projection of the $L_{2,N}$ Halo family.

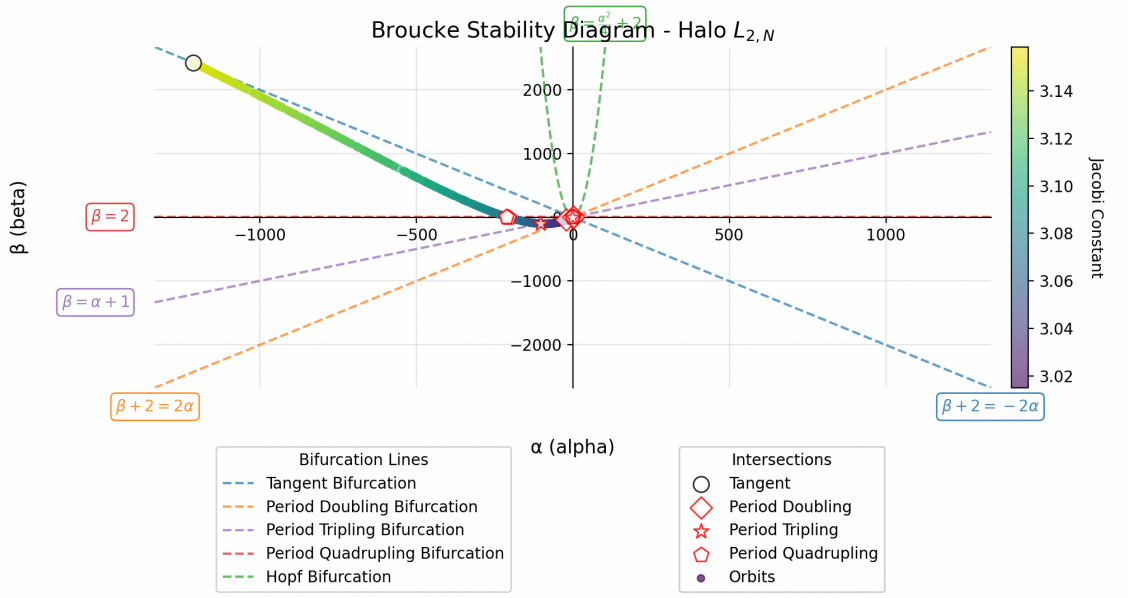


Figure A.2: Broucke stability diagram for the $L_{2,N}$ Halo family.

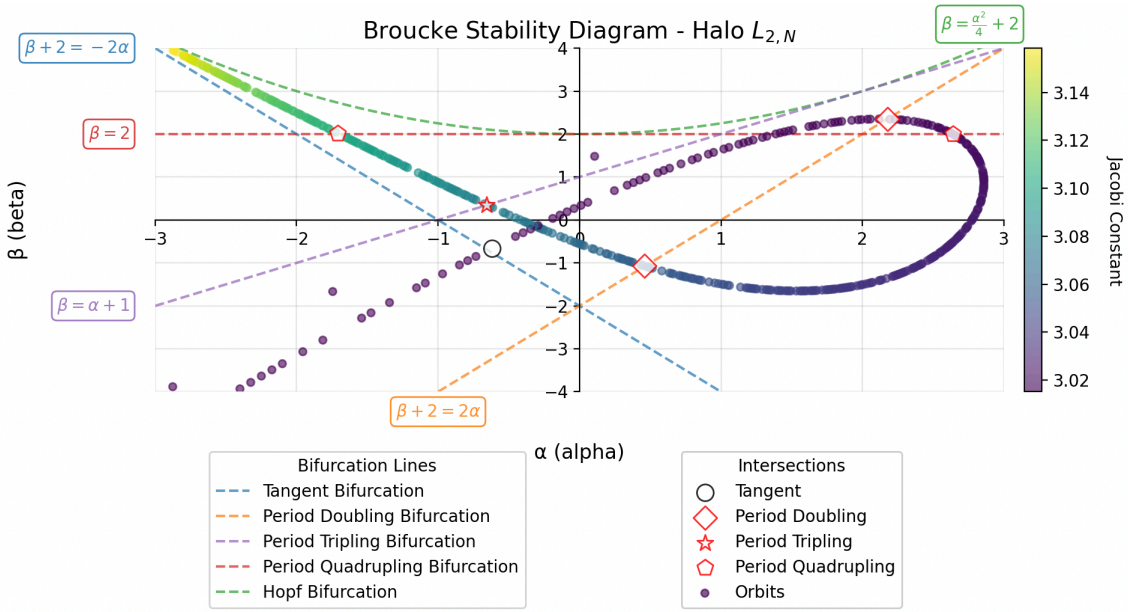


Figure A.3: Zoomed Broucke stability diagram for the $L_{2,N}$ Halo family.

A.1.2. L_2 Lyapunov

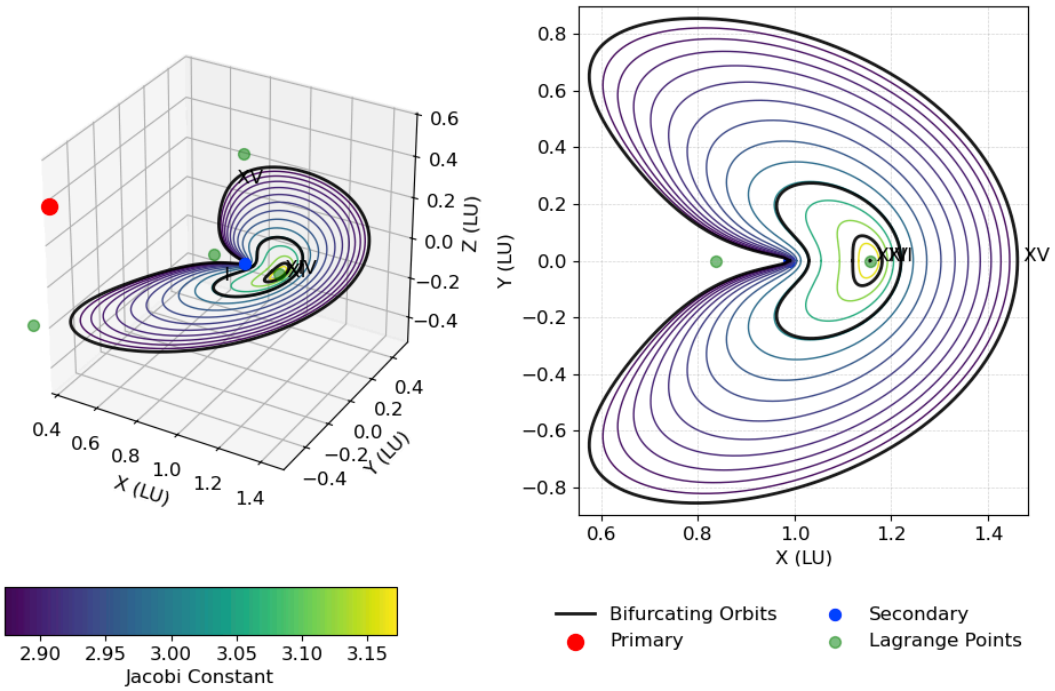


Figure A.4: Projection of the L_2 Lyapunov family.

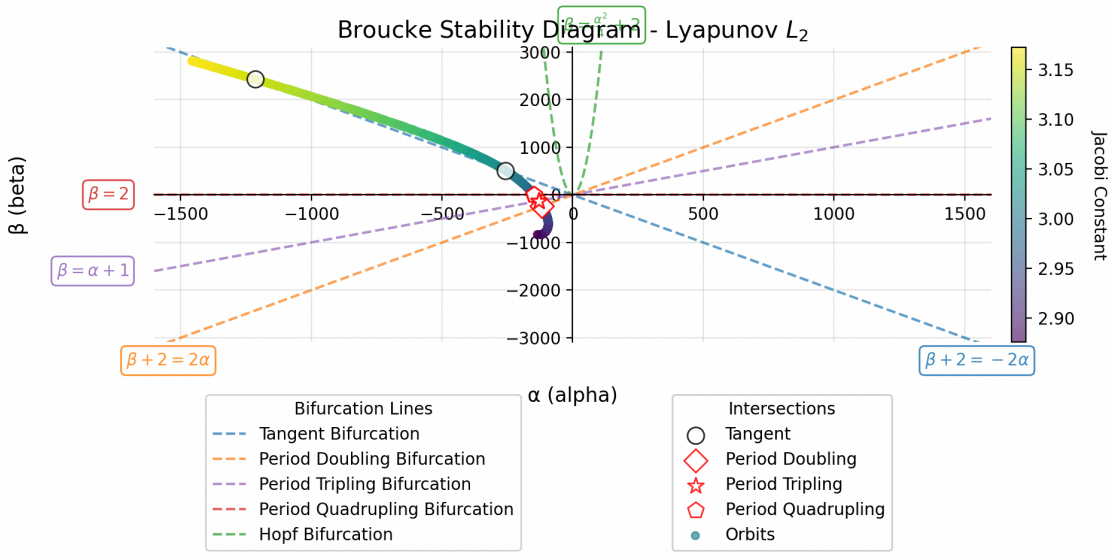


Figure A.5: Broucke stability diagram for the L_2 Lyapunov family.

A.1.3. L_2 Vertical

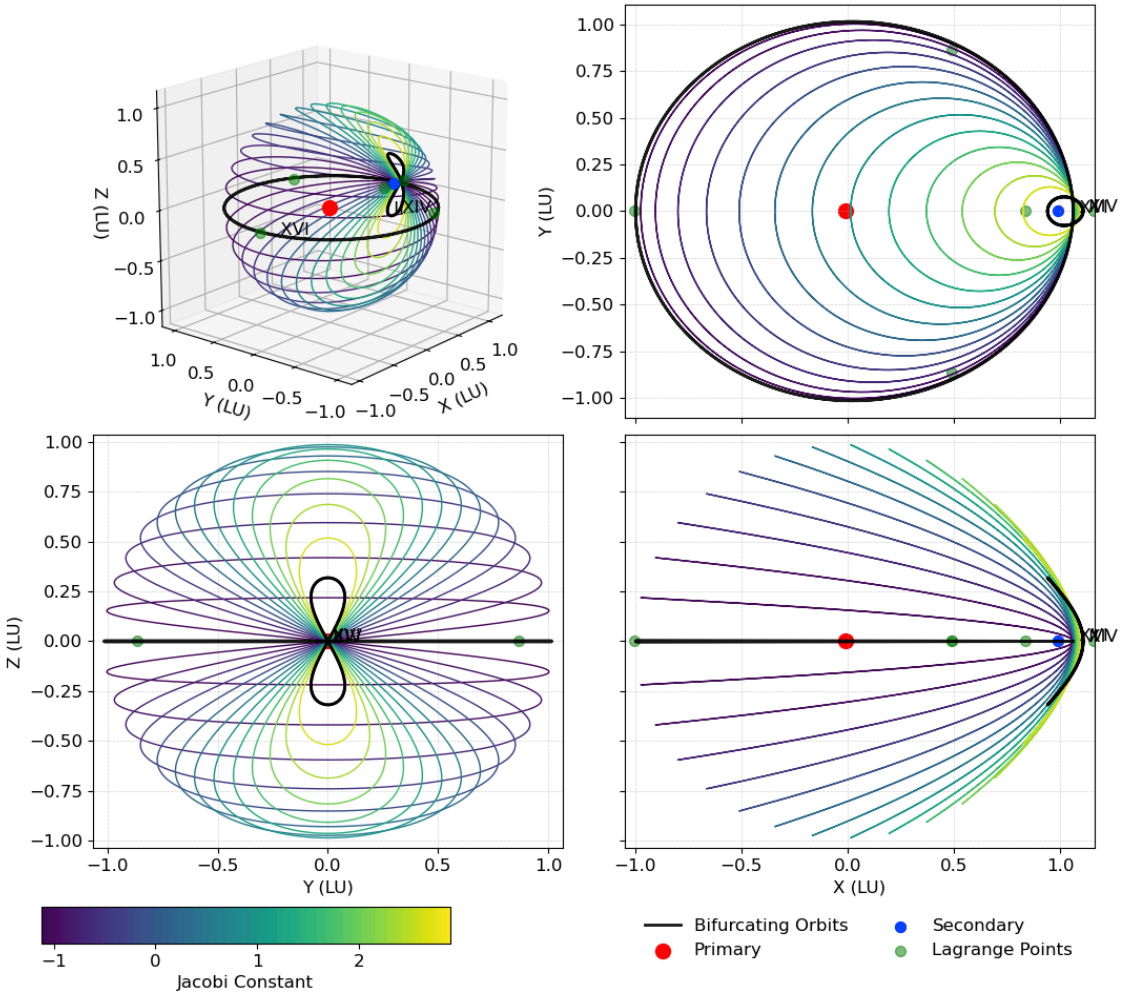


Figure A.6: Projection of the L_2 Vertical family.

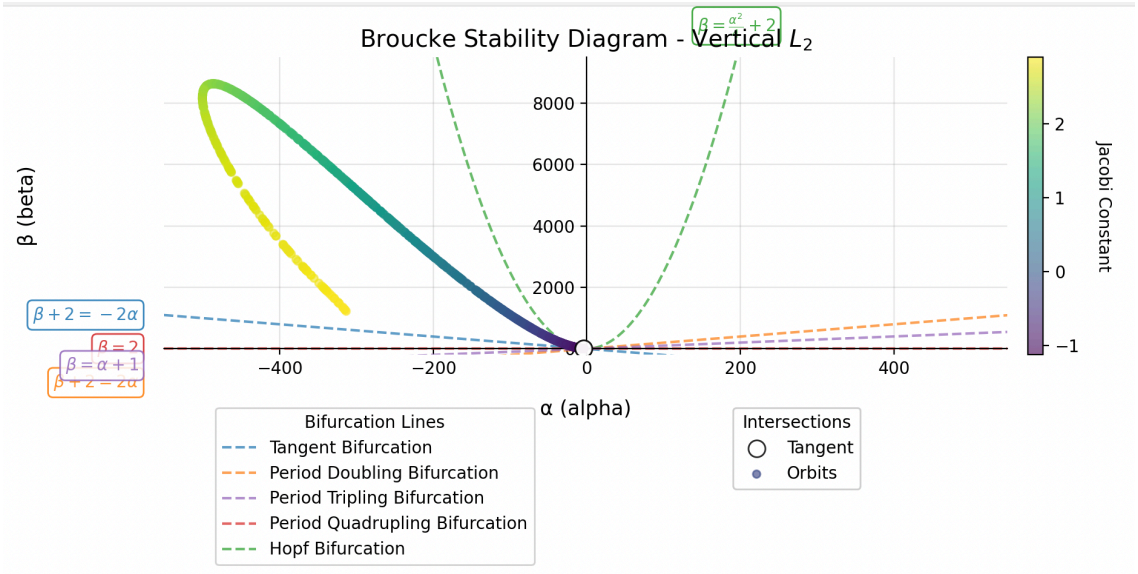


Figure A.7: Broucke stability diagram for the L_2 Vertical family.

A.1.4. L_2 Axial

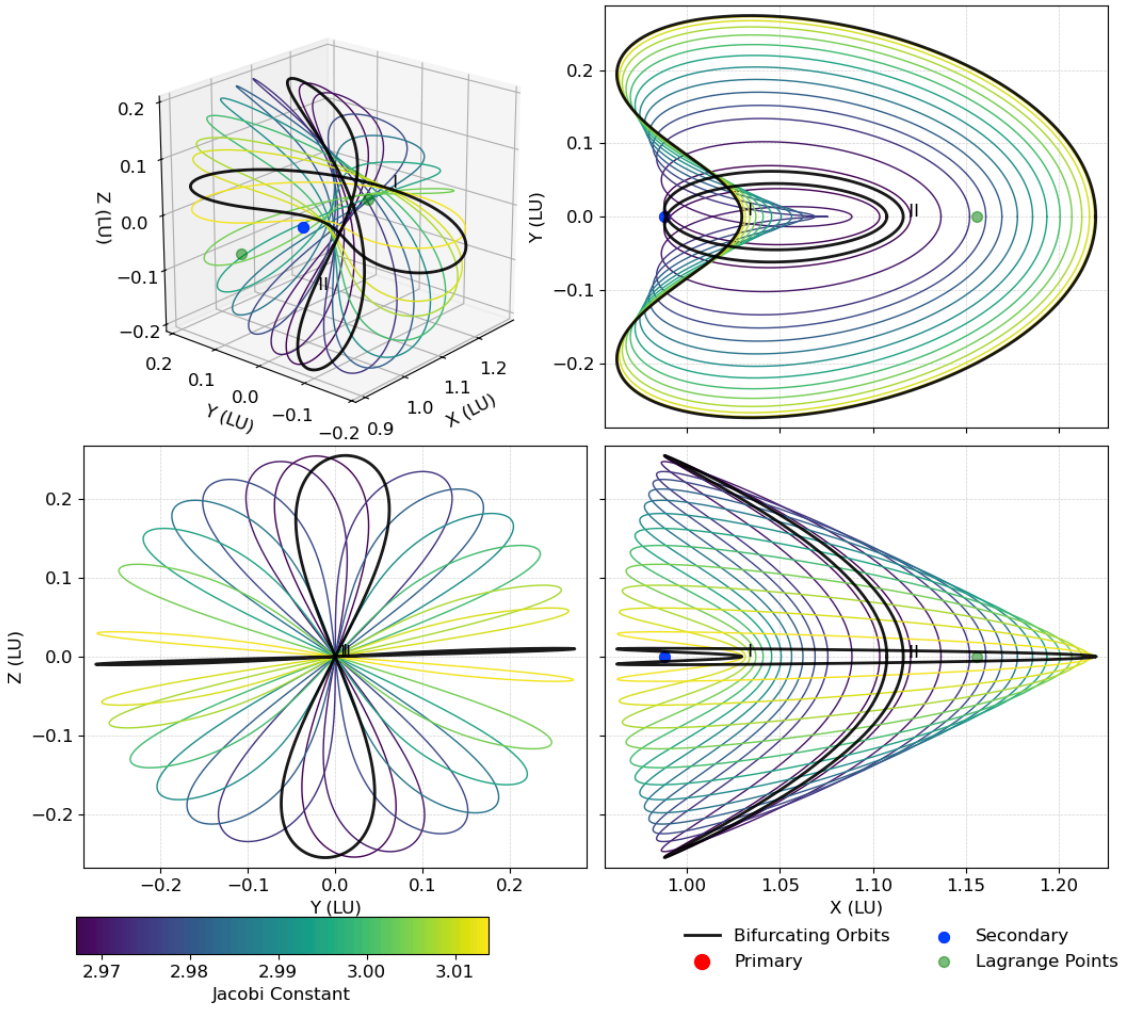


Figure A.8: Projection of the L_2 Axial family.

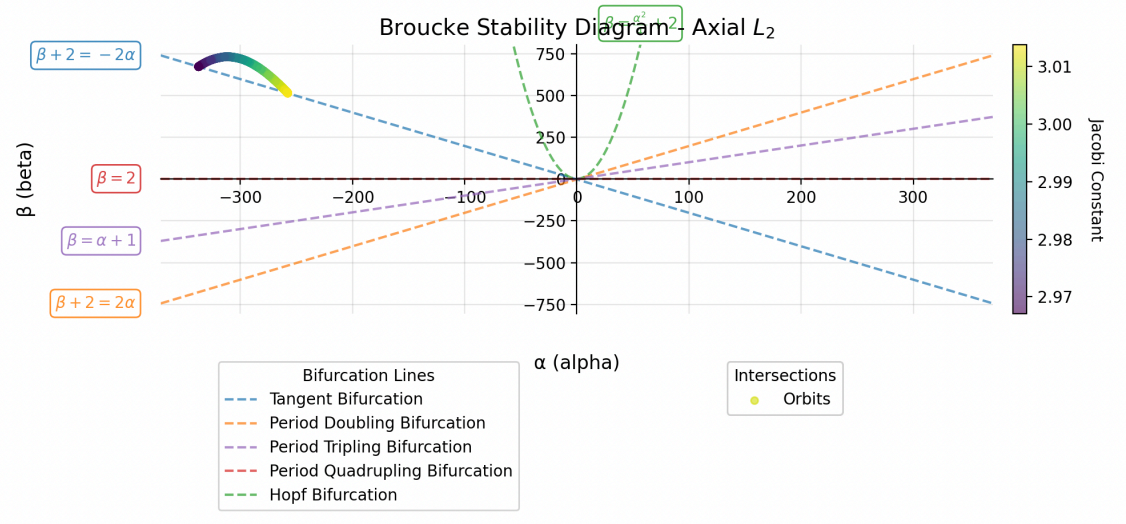


Figure A.9: Broucke stability diagram for the L_2 Axial family.

A.1.5. L_2 Dragonfly

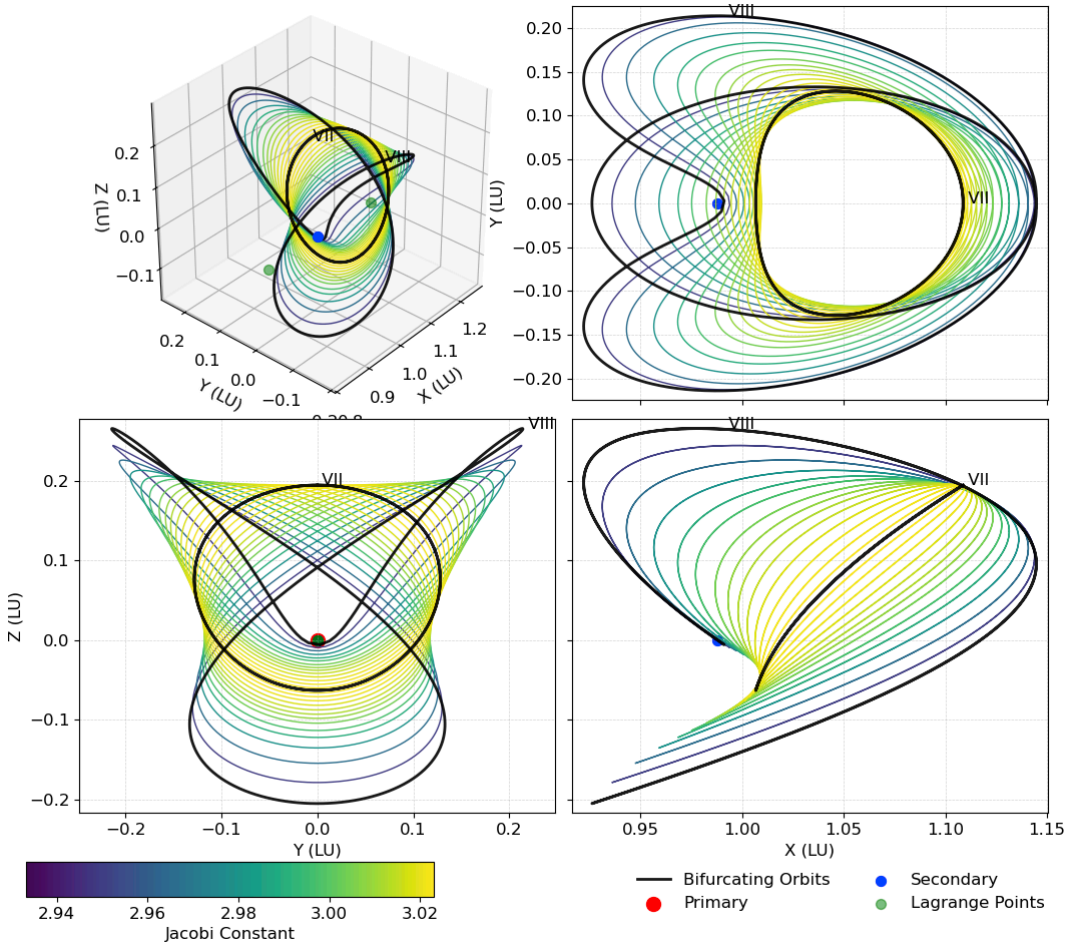


Figure A.10: Projection of the L_2 Dragonfly family.

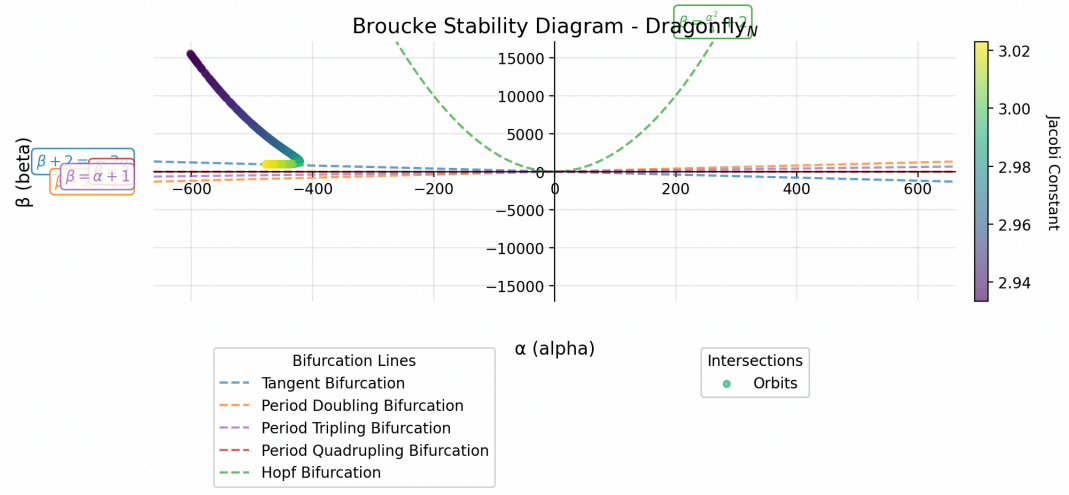


Figure A.11: Broucke stability diagram for the L_2 Dragonfly family.

A.1.6. L_2 Butterfly

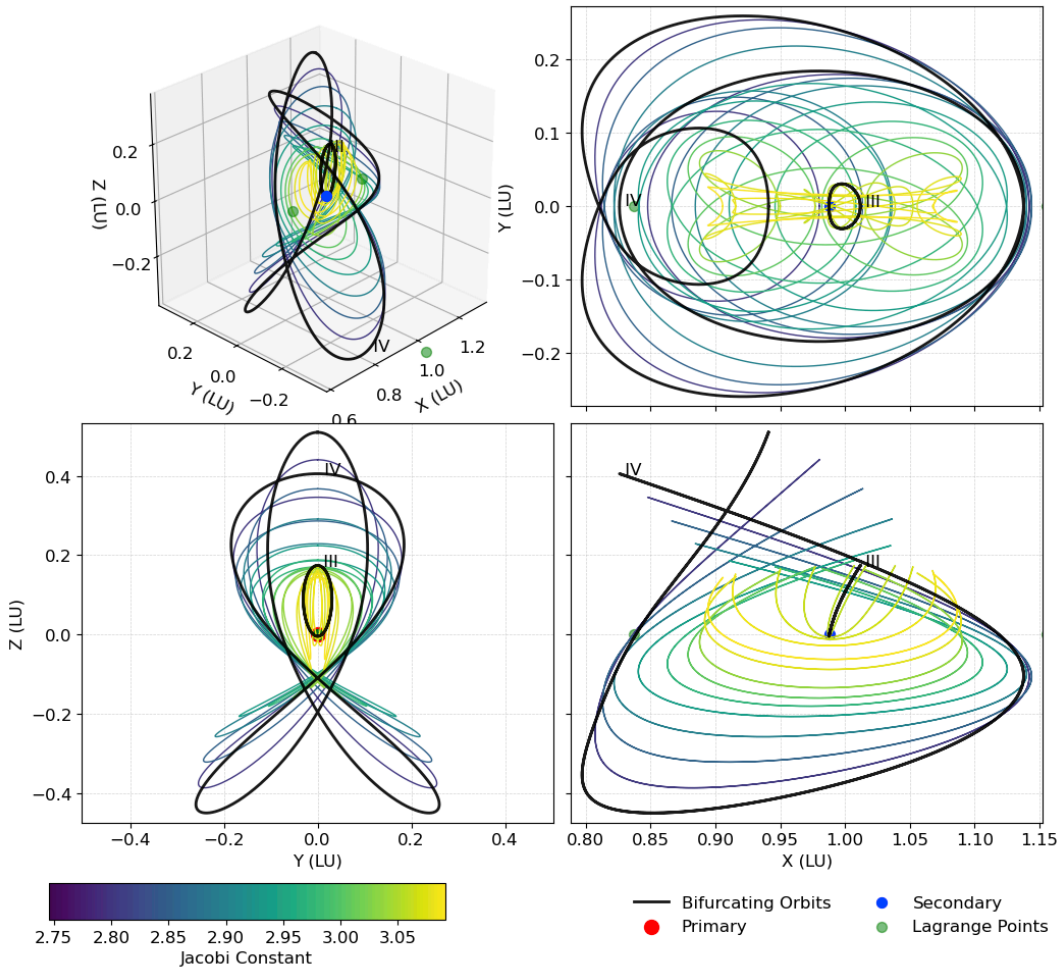


Figure A.12: Projection of the L_2 Butterfly family.

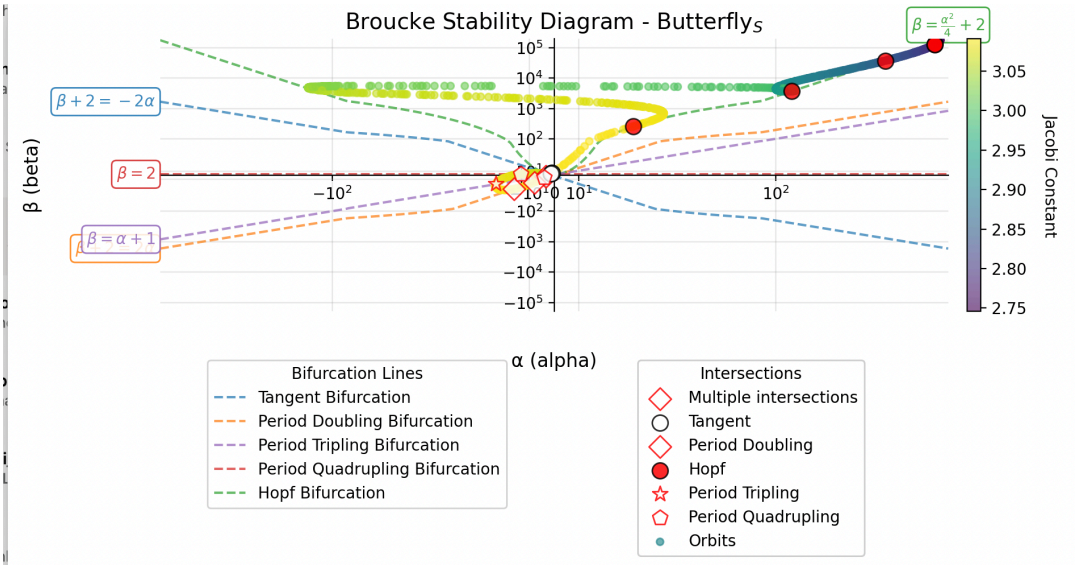


Figure A.13: Broucke stability diagram for the L_2 Butterfly family.

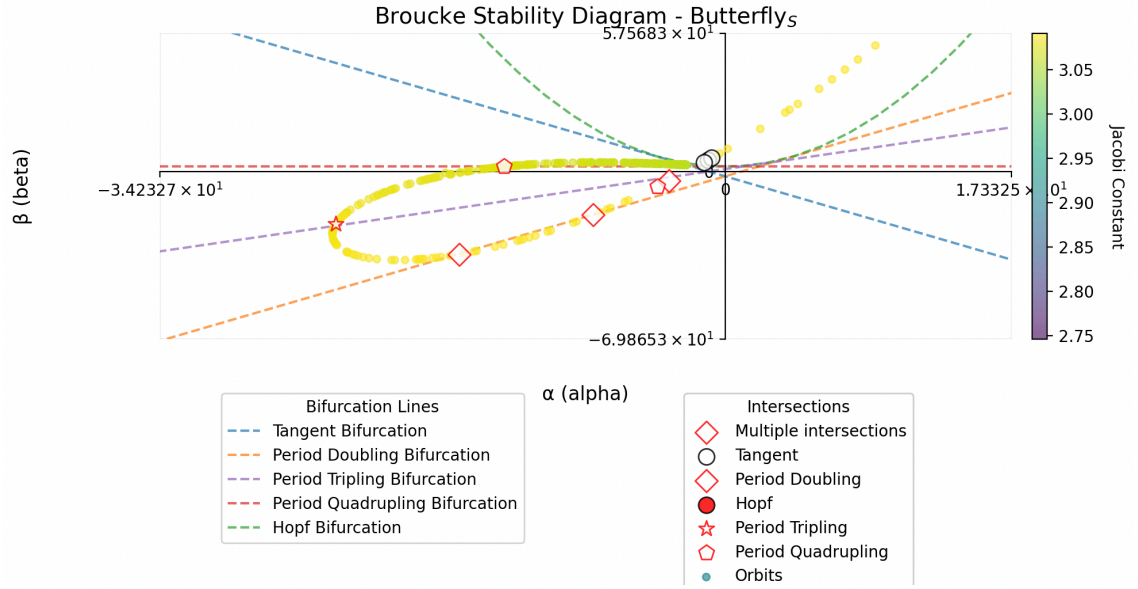


Figure A.14: Zoomed Broucke stability diagram for the L_2 Butterfly family.

A.2. Node Placement

An demonstration for sub-optimal node placement at intersections with the XZ plane is demonstrated below in Figure A.15 and A.16. These figures present the state time sequence evolution for node count N=4 and N=5, for which the MSDC convergence rates with $1 \cdot 10^{-4}$ noise levels experience a increase based off slight node repositioning and increase. Selection of time points for MSDC just slightly off these nodes results in convergence rates increasing dramatically, from 72% (N=4, Figure A.15 to 92%(N=5, Figure A.16).

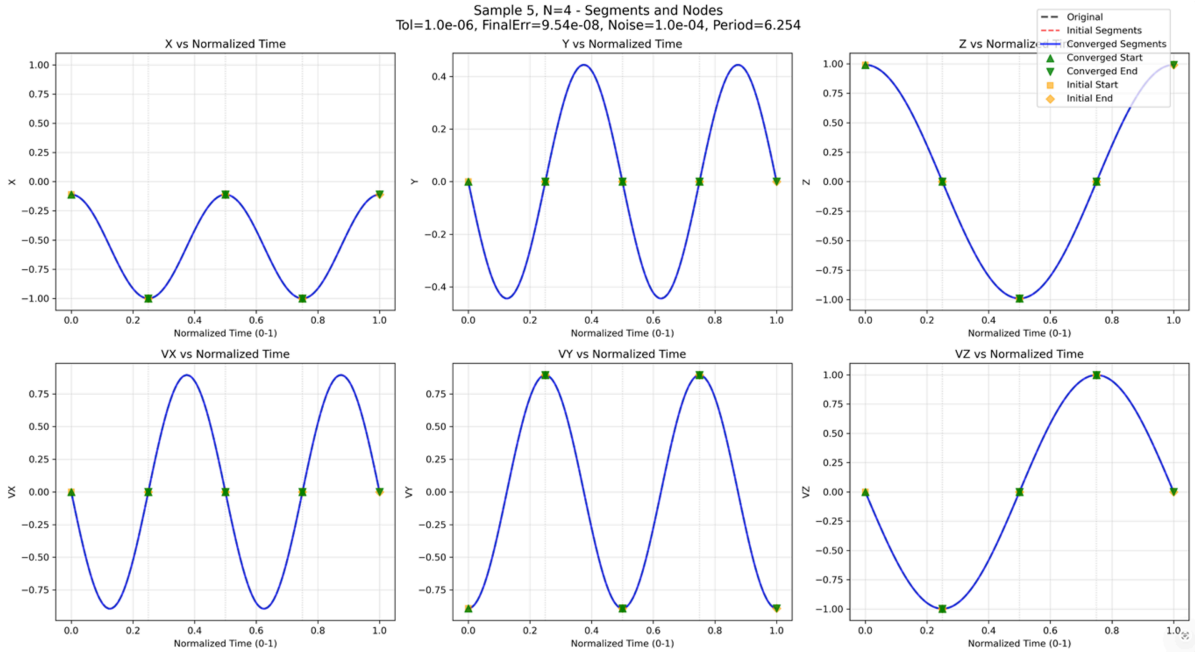


Figure A.15: Subpar node placement, N = 4 nodes/segments

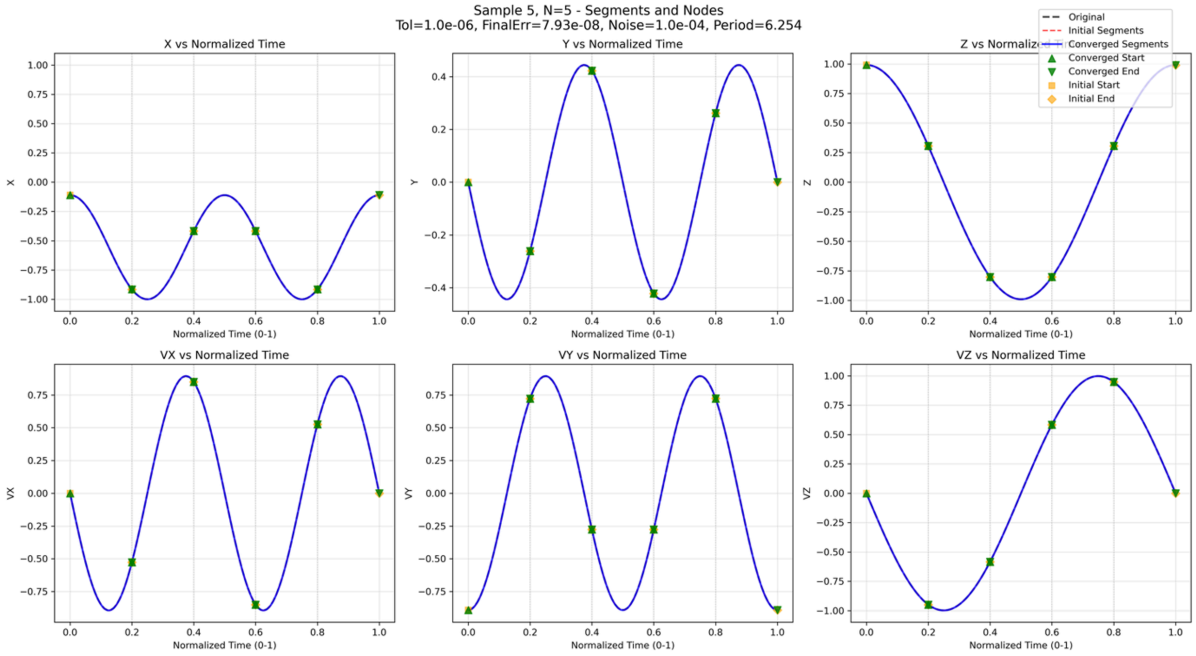


Figure A.16: Non-subpar node placement, N = 5 nodes/segments

A.3. Efficiency and Accuracy of Numerical Integration Schemes

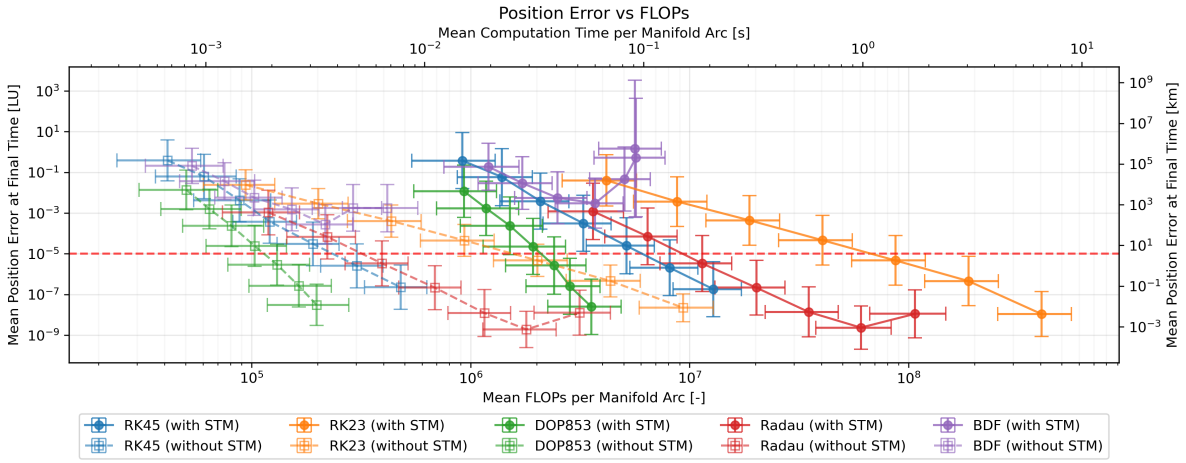


Figure A.17: Position error norm at 10TU for various integrators and integration tolerances. Red line denotes positional accuracy requirement, $\Delta p < 1 \times 10^{-5}[\mathrm{LU}]$

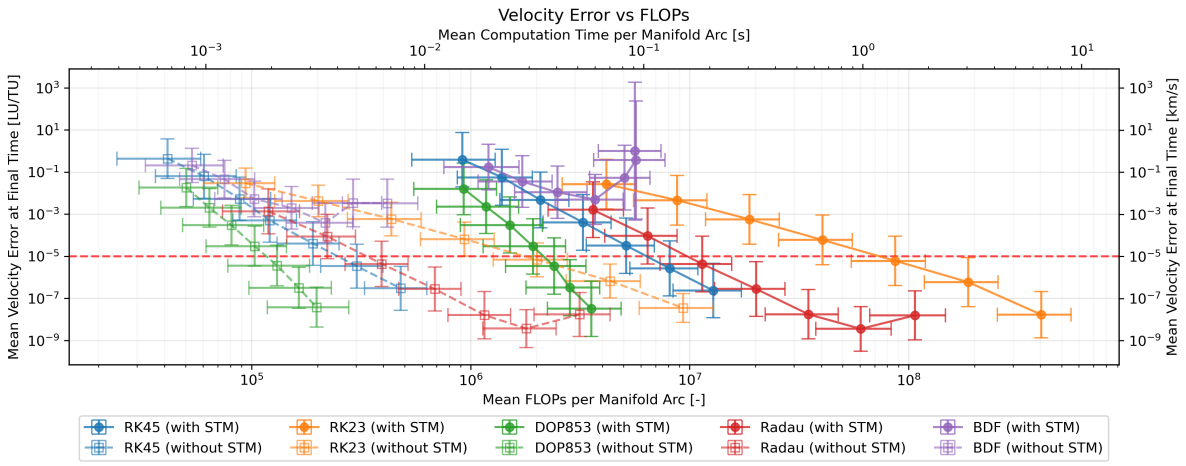


Figure A.18: Velocity error norm at 10TU for various integrators and integration tolerances. Red line denotes velocity accuracy requirement, $\Delta v < 1 \times 10^{-5}$ [LU/TU]

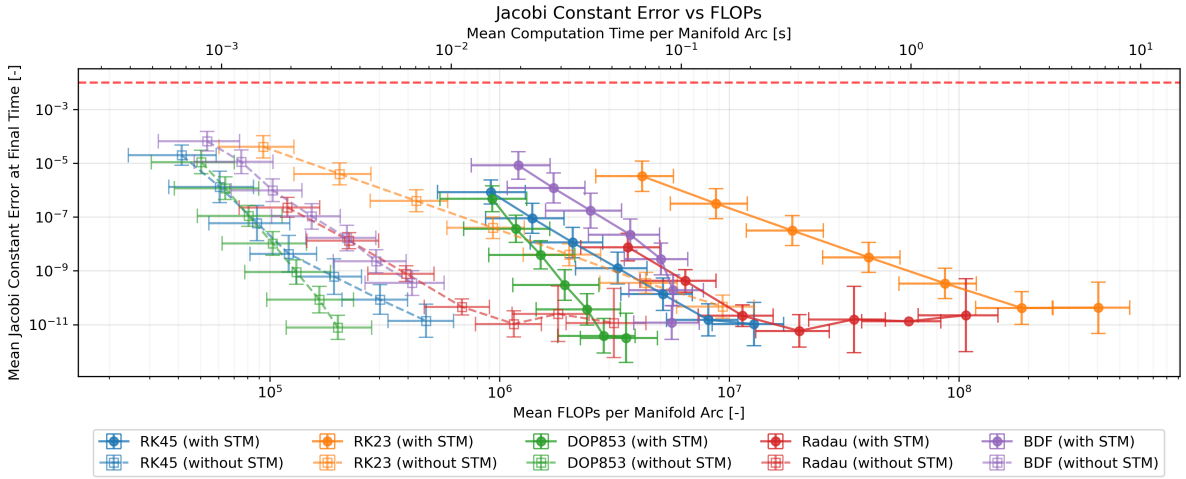


Figure A.19: Jacobi constant error at 10TU for various integrators and integration tolerances. Red line denotes energy. Red line denotes energy preservation requirement, $\Delta C < 1\%$

A.4. Franz-Russel Database

Preliminary investigation also included the Franz-Russel database. In the Franz-Russel Database, periodic orbits are identified through a combination of grid search and differential correction. Afterwards, PAL (see Section 2.13) is used within the AUTO software suite for familial continuation. Initial conditions are provided depending on the type of motion symmetry, dependent on which intersecting plane is the axis of symmetry. This has resulted in a large amount of periodic orbits with trajectories that have multiple revolutions, a parameter which is provided by N, the number of crossings of the plane of symmetry. This database is provided only for the Earth-Moon system, and has class labels defined by an unsupervised clustering algorithm (DB-SCAN) of orbital parameters, resulting in 33,980 families and sub-family clusters, across 13 million planar and three-dimensional periodic orbits [72]. Whilst this database is much more comprehensive then JPL, it is less interpretable due to the large amount of clusters.

$$\text{Axi-Symmetric:} \quad \mathbf{X}_0 = [x_0,\ 0,\ 0,\ 0,\ \dot{y}_0,\ \dot{z}_0], \qquad \mathbf{P} = P, \qquad \mathbf{N} = N \tag{A.1}$$

$$\text{X-Z Symmetric:} \quad \mathbf{X}_0 = [x_0,\ 0,\ z_0,\ 0,\ \dot{y}_0,\ 0], \qquad \mathbf{P} = P, \qquad \mathbf{N} = N \tag{A.2}$$

$$\text{Doubly Symmetric:} \quad \mathbf{X}_0 = [x_0,\ 0,\ 0,\ 0,\ \dot{y}_0,\ \dot{z}_0], \qquad \mathbf{P} = P, \qquad \mathbf{N} = N \tag{A.3}$$

$$\text{Planar (x–y plane):} \quad \mathbf{X}_0 = [x_0,\ 0,\ 0,\ 0,\ \dot{y}_0,\ 0], \qquad \mathbf{P} = P, \qquad \mathbf{N} = N \tag{A.4}$$

During implementation, it was found that the Franz–Russell database is incompatible with the JPL database due to discrepancies in the assumed values of μ, LU, and TU. The parameter sets for both databases, along with their differences, are provided in Table A.1. These inconsistencies imply that transforming one dataset into the other's normalization framework would require applying SSDC-based corrections to every orbit. Preliminary tests showed that this process required an average of eight SSDC iterations for the first 10,000 orbits, with only a 21.8% conversion rate, making a full correction of the entire dataset (≈ 13 million orbits) computationally infeasible. Consequently, the Franz–Russell database was excluded from the current work and deferred to future implementations. This in term prohibited the implementation/testing of latent space based classification methods based off the latent attributes identified in the JPL repository, as described in Section 8.2.

Table A.1: Comparison of CR3BP Normalization Parameters: Franz-Russell vs JPL Databases

Parameter	Franz-Russell	JPL	Δ (%)
Length Unit, l^* [km]	381,218.69	389,703.26	+2.23
Time Unit, t^* [s]	370,370	382,981.29	+3.40
Mass Ratio, μ	0.012153	0.012151	-0.02

A.5. VAE Optimization Constants and Hyperparameter Search Space

Table A.2: VAE optimization constants and hyperparameter search space, in combination with their sampling schemes (Bayesian opt).

Parameter	Description	Value / Range	Sampling
NAS Constants			
Input dim.	Number of input features	$7 \in \{t, \mathbf{X}\}$	N/A
Sequence length	Time steps per sample	100	N/A
Epochs	Training epochs per trial	200	N/A
Trend poly	Trend polynomial degree (TimeVAE)	0	N/A
Optimizer	Optimization algorithm	Adam	N/A
NAS Hyperparameters			
VAE type	Model variant	Dense, Conv, Time	Categorical
Latent dim.	Dimension of latent space	$\{4, 8, 16, 32, 64\}$	Categorical
Hidden layers (count)	Number of hidden layers	$\mathbb{Z} \in [2,\ 8]$	Uniform
Hidden layer width	Units per hidden layer	$\mathbb{Z} \in [32,\ 512]$	Log-uniform
MSE weight (λ)	Weight on reconstruction loss	$[10^{-2},\ 10^{1}]$	Log-uniform
Jacobi weight (γ)	Weight on Jacobi loss	$[10^{-4},\ 10^{1}]$	Log-uniform
Batch size	Mini-batch size	$[64, 2048]$	Log-uniform
Learning rate	Optimizer step size	$[10^{-5}, 10^{-2}]$	Log-uniform
Valid. fraction	Validation split ratio	$[0,\ 0.3]$	Uniform
Seasonality	TimeVAE seasonality configuration	{None, $(2, 50)$, $(4, 25)$, $(10, 10)$, $(20, 5)$, Combinations}	Categorical

A.6. DNN & KAN Optimization Constants and Hyperparameter Search Space

Table A.3: DNN NAS constants and searched hyperparameters with their sampling schemes.

Parameter	Description	Value / Range	Sampling
NAS Search Constants			
Input dim.	Number of input features	3	N/A
Output dim.	Number of output features	6	N/A
Epochs	Training epochs per trial	250	N/A
Optimizer	Optimization algorithm	Adam	N/A
Search Parameters			
Learning rate	Step size for optimization	5×10^{-4}	N/A
Batch size	Batch size per epoch	$\{512, 1024, 2048\}$	Categorical
Valid. split	Validation data ratio	$\mathbb{R} \in [0.0, 0.3]$	Uniform
Hidden layers	Number of hidden layers	$\mathbb{Z} \in [2, 12]$	Uniform
Neurons/layer	Hidden units per layer	$\mathbb{Z} \in [16, 1028]$	Logarithmic
Dropout rate	Fraction of dropped neurons	$\mathbb{R} \in [0.0, 0.3]$	Uniform
Activation fn.	Activation functions explored	{relu, tanh, selu, leaky_relu, elu, gelu}	Uniform
Jacobi weight	Jacobi loss weight (DNN-LC only)	$\mathbb{R} \in [10^{-5}, 10.0]$	Log-uniform

Table A.4: KAN NAS constants and parameters with their sampling schemes.

Parameter	Description	Value / Range	Sampling
NAS Search Constants			
Input dim.	Number of input features	3	N/A
Output dim.	Number of output features	6	N/A
Epochs	Training epochs per trial	300	N/A
Optimizer	Optimization algorithm	LBFGS	N/A
Search Parameters			
Hidden layers	Number of hidden layers in width	$\mathbb{Z} \in [2, 10]$	Uniform
Valid. split	Validation data ratio	$\mathbb{R} \in [0.0, 0.3]$	Uniform
Batch size	Batch size per epoch	$\{512, 1024, 2048\}$	Categorical
Neurons/layer	Units per hidden layer	$\mathbb{Z} \in [5, 50]$	Uniform
Grid	KAN grid resolution	$\mathbb{Z} \in [2, 10]$	Uniform
k	KAN basis order	$\mathbb{Z} \in [1, 5]$	Uniform