

Physics-Informed Generative Design of Invariant Manifolds in the Circular Restricted Three Body Problem

by

Gad Marconi

BSc, Delft University of Technology

Submitted to the School of Aerospace, Transport and Manufacturing
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE IN ASTRONAUTICS AND SPACE ENGINEERING

at

CRANFIELD UNIVERSITY

September 2025

© 2025 Gad Marconi. All rights reserved.

Authored by: Gad Marconi
School of Aerospace, Transport and Manufacturing
August 28, 2025

Certified by: Dr Marta Ceccaroni
Lecturer in Space Engineering, IRP Supervisor

IRP MEMBERS

IRP SUPERVISOR

Dr Marta Ceccaroni

Lecturer in Space Engineering

Centre for Autonomous and Cyberphysical Systems

IRP MEMBERS

Gad Marconi

Author

Student No. 441806

Physics-Informed Generative Design of Invariant Manifolds in the Circular Restricted Three Body Problem

by

Gad Marconi

Submitted to the School of Aerospace, Transport and Manufacturing
on August 28, 2025 in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE IN ASTRONAUTICS AND SPACE ENGINEERING

ABSTRACT

The increasing prevalence of small spacecraft platforms in Cis-Lunar space has underscored the need for computationally efficient and fuel-optimal trajectory design methods. Invariant manifolds in multi-body gravitational systems provide natural, low-energy pathways for space missions, but their computation remains reliant on numerically intensive techniques that require integration of the variational equations and sampling across large families of periodic orbits. These methods are poorly suited for onboard computation or rapid preliminary analysis.

This thesis proposes a physics-informed generative framework for invariant manifold design in the [Circular Restricted Three Body Problem \(CR3BP\)](#), using a [Vector Quantised Variational Autoencoder \(VQ-VAE\)](#) trained on trajectory data associated with Earth–Moon L_1 halo orbits. The model learns a discrete latent representation of the manifold geometry conditioned on physical parameters such as Jacobi constant and manifold branch. A physics-consistent loss is introduced to enforce compliance with the [CR3BP](#) dynamics during training.

The results presented in this work demonstrate the feasibility of using generative deep learning models to approximate invariant manifold structures in the [CR3BP](#). By learning a discrete latent representation of manifold trajectories directly from trajectory data, the proposed model provides a novel, data-driven perspective on a classic problem in astrodynamics. This work represents an initial step toward the use of generative models in astrodynamics. It offers a foundation for further exploration of latent-space representations, and highlights the potential for deep learning to support manifold discovery, low-energy transfer design, and the generation of physically consistent trajectories without reliance on traditional generating orbit methods.

IRP supervisor: Dr Marta Ceccaroni

Title: Lecturer in Space Engineering

Acknowledgments

I would like to express my deepest gratitude to my family-my parents and brother-for their unwavering support and encouragement throughout this journey. Even during my most challenging moments, you believed in me and provided everything you could to help me succeed. Your faith in my abilities has been a constant source of strength.

To my friends, thank you for the countless hours of stimulating discussions, both academic and beyond. Your insights have enriched my understanding of my research. Our friendship helped me stay sane during the tough times and reminded me to take breaks when I needed them. This whole process would have been much harder without you.

I am grateful to my supervisor, Dr. Marta Ceccaroni, for her invaluable guidance and mentorship throughout this research. Your expertise, constructive feedback, and direction have been instrumental in shaping this thesis and my development as a student.

Contents

<i>Acronyms and Abbreviations</i>	VI
<i>List of Figures</i>	VIII
<i>List of Tables</i>	IX
<i>1 Introduction</i>	1
<i>2 Background</i>	3
2.1 The Circular Restricted Three Body Problem	3
2.2 Dynamics and Stability in the Circular Restricted Three Body Problem . . .	5
2.3 Invariant Manifolds and Space Mission Design	9
2.4 Machine Learning Approaches to Dynamical Systems	10
2.5 Applications of Machine Learning in the Circular Restricted Three Body Problem	12
2.5.1 State of the Art	14
2.6 Aims and Objectives	17
<i>3 Methodology</i>	19
3.1 Numerical Computation of Invariant Manifolds	19
3.1.1 Reduction to the Centre Manifold	19
3.1.2 Differential Correction of Seed Orbits	24
3.1.3 Generating Families of Halo Orbits	26
3.1.4 Invariant Manifolds of Halo Orbits	26
3.2 Network Architecture and Training	28
3.2.1 Training Data	28
3.2.2 Network Architecture	29
3.2.3 Physics-Informed Loss	30
3.2.4 Training	31
<i>4 Results</i>	32
4.1 Reconstruction of Manifolds	33
4.2 Generation of Manifolds	37
4.2.1 Benchmarking	41
4.3 Latent Space	41
<i>5 Discussion</i>	47

5.1	Conclusion	47
5.2	Recommendations and Future Work	49
5.2.1	Extended Time Horizons	50
5.2.2	Improved Manifold Coverage	50
5.2.3	Alternative Generative Architectures	51
5.2.4	Physics Constraints and Accuracy	51
5.2.5	Architectural Improvements	52
5.2.6	Continuous Manifold Representations	52
5.2.7	Higher-Fidelity Dynamical Models	53
5.2.8	Real-Time Generation and Computational Efficiency	54
A	<i>Implementation of Algorithms for Centre Manifold Reduction</i>	56
A.1	Polynomial Representation	56
A.1.1	Benchmarks	60
A.2	Implementing the Lie series Transform	60
A.2.1	Polynomial Multiplication	60
A.2.2	Polynomial Differentiation	61
A.2.3	Poisson Bracket	62
A.2.4	The Lie Series Transform	63
A.2.5	Hamiltonian Expansion	63
A.2.6	Coordinate Transformation Expansions	64
	<i>References</i>	67

Acronyms and Abbreviations

AI Artificial Intelligence. [12](#)

ANN Artificial Neural Network. [11](#)

CNN Convolutional Neural Network. [11](#)

CPU Central Processing Unit. [41](#)

CR3BP Circular Restricted Three Body Problem. [II](#), [1](#), [2](#), [3](#), [4](#), [5](#), [9](#), [10](#), [11](#), [12](#), [14](#), [17](#), [18](#), [20](#), [28](#), [30](#), [40](#), [46](#), [47](#), [51](#), [53](#)

DL Deep Learning. [11](#)

ELBO Evidence Lower Bound. [12](#), [13](#), [14](#)

GAN Generative Adversarial Network. [11](#)

GPU Graphics Processing Unit. [10](#), [41](#), [54](#)

GRU Gated Recurrent Unit. [29](#), [30](#)

KL Kullback-Leibler. [14](#)

LIDVAE Linear Identifiable Deep Variational Autoencoders. [51](#)

ML Machine Learning. [10](#)

MLP Multilayer Perceptron. [29](#), [30](#)

MSE Mean Squared Error. [32](#), [33](#)

NHIM Normally Hyperbolic Invariant Manifold. [7](#), [9](#), [26](#)

NODE Neural Ordinary Differential Equation. [51](#), [52](#)

RMSE Root Mean Squared Error. [32](#)

STM State Transition Matrix. [6](#), [10](#), [24](#), [25](#), [27](#)

SVM Support Vector Machines. [11](#)

UMAP Uniform Manifold Approximation Projection. [41](#)

VAE Variational Autoencoder. [11](#), [12](#), [13](#), [14](#), [15](#)

VQ-VAE Vector Quantised Variational Autoencoder. [II](#), [2](#), [14](#), [15](#), [16](#), [17](#), [29](#), [32](#), [41](#), [46](#),
[47](#), [49](#), [51](#), [54](#)

List of Figures

2.1	NASA/Szebehely Primary-Secondary convention in the CR3BP.	4
2.2	Invariant manifolds of an Earth-Moon L_1 halo orbit.	8
3.1	Transformation process from synodic to centre manifold frames.	20
3.2	Poincaré map of the $p_2 = 0$ section of the Earth-Moon L_1 libration point at energy $C = 0.2$	22
3.3	Poincaré map of the $p_2 = 0$ section of the Earth-Moon L_1 libration point at energy $C = 1.0$	23
3.4	Coordinate transformation process from centre manifold frame to synodic frame.	24
3.5	Seeded Earth-Moon L_1 northern halo for $C = 1.0$ before and after correction.	25
3.6	Seeded Earth-Moon L_1 southern halo for $C = 1.0$ before and after correction.	25
4.1	Energy-conditioned reconstruction errors. Positive branch shows consistently higher median error, particularly at high and low C values.	34
4.2	Numerically obtained and reconstructed positive branches of halo manifolds at $C = 3.1168$ and $C = 3.1608$	35
4.3	Numerically obtained and reconstructed negative branches of halo manifolds at $C = 3.1253$ and $C = 3.1742$	36
4.4	Mean state error and maximum trajectory Jacobi error vs Jacobi constant.	37
4.5	Maximum Jacobi error $ \Delta C _{\max}$ vs minimum distance to either primary in canonical units.	38
4.6	Generated negative and positive halo manifolds with Jacobi constant 3.174344.	39
4.7	Generated negative and positive halo manifolds with Jacobi constant 3.135462.	39
4.8	Generated negative and positive halo manifolds with Jacobi constant 3.096582.	40
4.9	Uniform manifold approximation and projection embedding (UMAP) of the model's pre-quantisation latent vectors z_e	42
4.10	Hierarchical clustering dendrogram of learnt codes.	43
4.11	Conditional entropy and information decomposition of learnt codebook structure.	44
4.12	Mutual information plot of different variables.	45
A.1	Packing and unpacking operations for the custom polynomial representation.	58

List of Tables

4.1	Channel-wise reconstruction root mean squared error in scaled space.	32
4.2	Mean squared error in reconstruction in scaled space per branch.	33
4.3	Root distance, and median average and maximum Jacobi error for each branch in canonical units.	38
A.1	Packed and decoded polynomial structure for degree 2.	57
A.2	Scaling of monomials with degree and number of variables.	58
A.3	Custom polynomial representation benchmarks compared to SymPy [49] for different polynomial degrees.	60
A.4	Monomial count, computation time, memory usage, and evaluation time of different centre manifold expansions.	65

Chapter 1

Introduction

Although comets *Oterma* and *Gehrels 3* start in heliocentric orbits beyond Jupiter, they quickly shift to heliocentric orbits within its radius, a phenomenon common among many comets in the Solar System. During this transition, comets are frequently captured by Jupiter for up to several orbits as they pass near the Sun-Jupiter libration points L_1 and L_2 , which are the closest equilibrium points to Jupiter along the Sun-Jupiter line. [1].

While studying the behaviour of *Oterma* and *Gehrels 3*, *Lo* and *Ross* noticed that the orbits of these comets lie within the invariant manifolds of L_1 and L_2 . With their observations, they established the existence of a new heteroclinic connection between periodic orbits around L_1 and L_2 with the same Jacobi constant [2]. Several years later, they applied this mechanism to spacecraft mission design, leading to novel trajectories that used low-energy transfers and gravitational captures for interplanetary missions [3].

Invariant manifolds have played a crucial role in previous mission designs, such as the low-energy trajectory of the Genesis mission and lunar transfers for Artemis. They enable fuel-efficient transitions between orbital regions, making them a powerful tool in trajectory planning. However, current methods struggle with computational efficiency and scalability to high-fidelity models.

The theoretical foundation for manifold-driven transport lies in the existence of stable and unstable invariant manifolds associated with periodic and quasi-periodic orbits near equilibrium points in the [CR3BP](#). These manifolds act as separatrices in phase space, guiding motion between dynamically distinct regions.

Computing invariant manifolds in the [CR3BP](#) is an indispensable step in space mission design exploiting low-energy capture and escape. Typically, invariant manifolds are computed by finding the stable and unstable directions through the eigendecomposition of the monodromy matrix of a periodic orbit around a libration point. The stable and unstable directions are then propagated along the entire orbit using the state transition matrix [4]. This method is too complex for widespread use in practical mission design, as the manifolds are defined only for periodic orbits, whereas these do not occur in high-fidelity models. Additionally, certain

missions require analysis of millions of resonant orbits, which is not practical to implement on modern computing hardware [5]. This is especially pronounced in small spacecraft platforms, such as CubeSats, where computational resources are constrained [6].

This paper investigates the use of generative deep learning to approximate invariant manifolds directly from trajectory data. Training a [VQ-VAE](#) on manifold trajectories associated with Earth-Moon halo orbits yields a compact, discrete representation of the manifold structure. The model is conditioned on physical parameters such as Jacobi constant and manifold branch, and incorporates a physics-informed loss to ensure consistency with the governing equations of motion.

The remainder of this paper is organised as follows: Chapter [2](#) provides theoretical background on the [CR3BP](#) and machine learning approaches to dynamical systems. Chapter [3](#) details the methodology used to generate the training data and network architecture. Chapter [4](#) presents results from the trained model. Finally, Chapter [5](#) discusses limitations, conclusions, and future work.

Chapter 2

Background

2.1 The Circular Restricted Three Body Problem

The [CR3BP](#) describes a dynamical system in which two massive bodies, referred to as the *primary* and *secondary* (and together as the *primaries*) are bound in a Keplerian orbit, while a smaller, infinitesimal particle is subject to their combined gravitational attraction. By convention, the primary sits on the left of the secondary, as seen in [Figure 2.1](#). The equations of motion in the synodic frame (rotating about the barycentre of the primaries, with angular velocity such that they appear immobile), after appropriate normalisation are:

$$\ddot{X} - 2\dot{Y} = \frac{\partial\Omega}{\partial X}, \quad (2.1)$$

$$\ddot{Y} + 2\dot{X} = \frac{\partial\Omega}{\partial Y}, \quad (2.2)$$

$$\ddot{Z} = \frac{\partial\Omega}{\partial Z}. \quad (2.3)$$

Where X , Y , and Z are the synodic coordinates of the particle, and Ω is the effective potential:

$$\Omega = \frac{1}{2}(X^2 + Y^2) + \frac{1-\mu}{R_1} + \frac{\mu}{R_2}. \quad (2.4)$$

It is well known that in the [CR3BP](#), there exist five equilibrium points as solutions to the ordinary differential equations in [Equation 2.1](#) [7]. The dynamics in the rotating frame can be understood through the effective potential Ω . The gradient of this potential governs the motion of the infinitesimal particle. Setting the velocities and accelerations to zero implies that equilibria satisfy:

$$\nabla\Omega = \mathbf{0}. \quad (2.5)$$

This condition yields five isolated equilibrium points, known as the Lagrange points. The first three, known as collinear points, are located along the axis that joins the primaries (shown in blue in Figure 2.1). The remaining two, referred to as the triangular points, form an equilateral triangle with the primaries (shown in red in Figure 2.1). [7]

The collinear (especially L_1 and L_2) points are of special interest for space mission design. Orbits around these points can be used for continuous observation of the primaries, or to provide an unobstructed view of deep space. These unique properties have made L_1 and L_2 the preferred locations for numerous missions: Sun-Earth L_1 hosts solar observatories like *SOHO* [8], Sun-Earth L_2 serves as home to telescopes like the *JWST* [9], while Earth-Moon L_2 is designated for *Artemis*' Lunar Gateway [10].

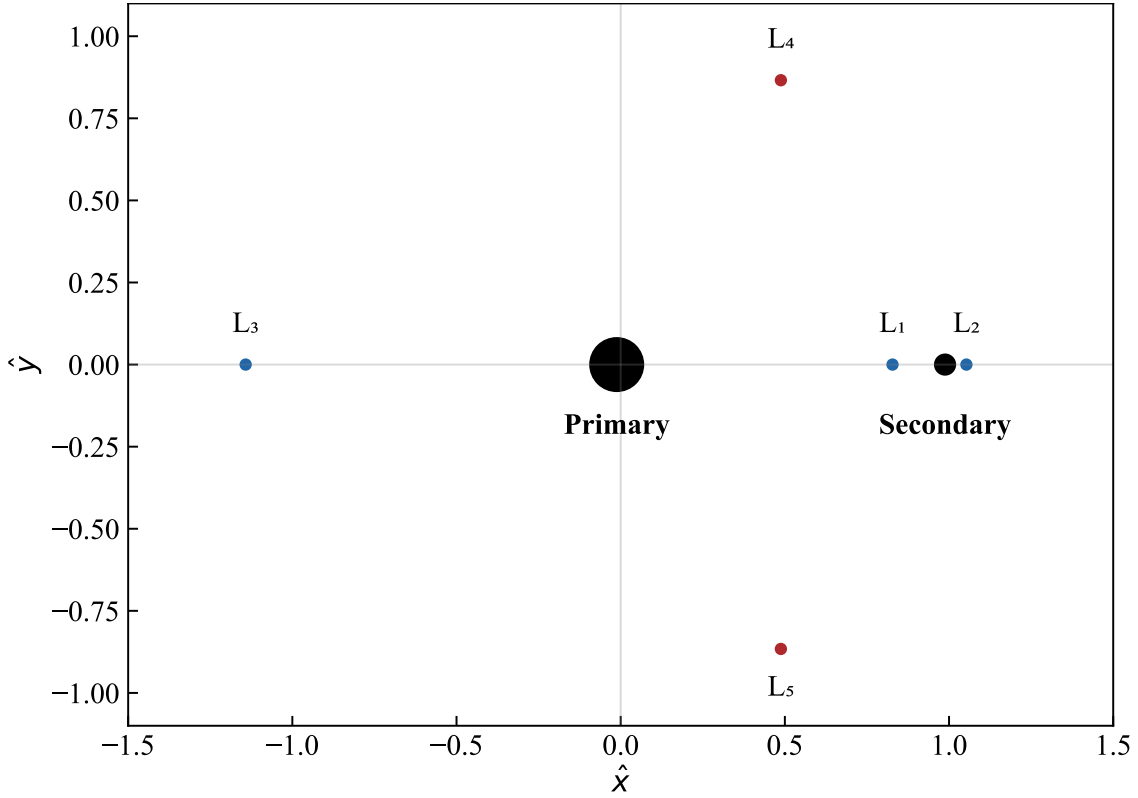


Figure 2.1: NASA/Szebehely Primary-Secondary convention in the CR3BP.

Notably, there exists a conserved quantity in the CR3BP, known as the Jacobi constant, which arises from the time-independence of the equations of motion in the rotating frame. The Jacobi constant is defined as:

$$C = 2\Omega(X, Y, Z) - (\dot{X}^2 + \dot{Y}^2 + \dot{Z}^2), \quad (2.6)$$

It can be interpreted as a combination of the particle's kinetic energy and the effective potential energy in the rotating frame. Since C is conserved along trajectories, it imposes forbidden regions in configuration space: the particle cannot enter regions where

$$2\Omega(X, Y, Z) < C. \quad (2.7)$$

These forbidden regions are visualised as zero-velocity surfaces, which provide a powerful tool for understanding the qualitative behaviour of trajectories in the [CR3BP](#). For example, the Jacobi constant determines whether a particle can transition between the vicinities of the primaries or remain confined to a specific region around a Lagrange point. The value of C at the collinear points is particularly important for mission design, as it defines the energy threshold for trajectories passing near and between L_1 , L_2 , or L_3 . [\[7\]](#)

2.2 Dynamics and Stability in the Circular Restricted Three Body Problem

The linear stability of the equilibrium points of the autonomous system $\dot{\mathbf{x}} = f(\mathbf{x})$ in [Equation 2.1](#) is determined by the eigenvalues of its Jacobian:

$$Df(x) = \begin{pmatrix} \mathbf{0}_{3 \times 3} & \mathbf{I}_{3 \times 3} \\ \boldsymbol{\Omega}_{ij} & \mathbf{K} \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ \Omega_{xx} & \Omega_{xy} & \Omega_{xz} & 0 & 2 & 0 \\ \Omega_{xy} & \Omega_{yy} & \Omega_{yz} & -2 & 0 & 0 \\ \Omega_{xz} & \Omega_{yz} & \Omega_{zz} & 0 & 0 & 0 \end{pmatrix}. \quad (2.8)$$

Where $\boldsymbol{\Omega}_{ij}$ is the matrix of second partial derivatives of the potential Ω , and $\mathbf{K}$ is the matrix containing the Coriolis terms. The eigenvalues λ of this system are the roots of the characteristic polynomial, which has the form:

$$(\lambda^2 + \omega_1^2)(\lambda^2 + \omega_2^2)(\lambda^2 - \nu^2), \quad (2.9)$$

and yields a pair of real eigenvalues, $\pm\nu$, and two pairs of purely imaginary eigenvalues $\pm i\omega_1$ and $\pm i\omega_2$, where ω_1 and ω_2 are the fundamental frequencies of the linear system. The linearised system is of type saddle $\times$ saddle $\times$ centre; the eigenvalues induce a decomposition of the tangent space at the equilibrium point $T_{\mathbf{x}}\mathbb{R}^6$ into three invariant eigenspaces under the linearised flow [\[11\]](#):

$$T_{\mathbf{x}}\mathbb{R}^6 = E^s \oplus E^u \oplus E^c. \quad (2.10)$$

The stable and unstable subspaces (E^s and E^u respectively) are of dimensions 1, and correspond to the spans of $\mathbf{v}_s$ and $\mathbf{v}_u$ respectively (the eigenvectors of the eigenvalues $-\nu$ and

$+\nu$). The presence of the subspaces E^s and E^u makes the collinear points linearly unstable, as any perturbation in these directions leads to exponential divergence from equilibrium.

The centre subspace E^c is of dimension 4, which contains the two pairs of purely imaginary eigenvalues $\pm i\omega_1$ and $\pm i\omega_2$. Centre manifold theorem establishes that the topological structure of the eigenspaces persists in the full non-linear system; there exist non-linear invariant manifolds tangent to the linear eigenspaces at the equilibria [11]. Specifically, there exists a locally invariant centre manifold:

$$W_l^c(x) \subset \mathbb{R}^6, \quad (2.11)$$

tangent to E^c , which contains all small amplitude bounded orbits near the equilibrium. A manifold in this context is a smooth, locally Euclidean surface embedded in phase space; that is, around every point, it resembles a lower-dimensional Euclidean space. As a dynamical structure, the manifold is invariant under the flow; meaning any trajectory that starts on it will remain on it for all forward and backward time.

As the linearised system has a two distinct pairs of purely imaginary eigenvalues ω_1 and ω_2 where $\omega_1/\omega_2 \notin \mathbb{Z}$, by Lyapunov centre theorem [12], there exist two distinct one-parameter families of periodic orbits emanating from each point $\gamma_k(C)$, where the parameter C can be related to the Hamiltonian:

$$H(\gamma_k(C)) - H(\mathbf{x}_0), \quad (2.12)$$

and the periodic orbits $T_k(C) \rightarrow 2\pi/\omega_k$ as $C \rightarrow 0$. The first pair of eigenvalues admits the planar periodic orbits (symmetric about the x -axis), known as Lyapunov orbits, while the latter is responsible for the vertical family (oscillatory in the z direction). At energy levels near the ones of the equilibrium points themselves, the interaction between these modes gives rise to Lissajous families, which densely fill 2-tori in W_l^c when ω_1/ω_2 is irrational. [13]

These primitive families can themselves undergo bifurcations. A bifurcation is a qualitative change in the behaviour of a dynamical system that occurs when varying a parameter μ . For a system $\dot{\mathbf{x}} = f(\mathbf{x}, \mu)$, a bifurcation occurs at $\mu \cdot c$ if the topological structure of the phase portrait is not equivalent for μ in neighbourhoods on either side of $\mu \cdot c$. [11]

A periodic orbit γ can be analysed by defining a local transverse Poincaré section Σ and studying the map $\Pi : \Sigma \rightarrow \Sigma$. The orbit γ corresponds to a fixed point $\mathbf{p} = \gamma \cap \Sigma$ of Π . The stability of γ can be determined by the eigenvalues of the linearised map $D\Pi(\mathbf{p})$ which are related to the eigenvalues of γ 's monodromy matrix (State Transition Matrix (STM) propagated for one period). A bifurcation of the orbit family occurs when, as the parameter C is varied, an eigenvalue of $D\Pi(\mathbf{p})$ crosses the unit circle in the complex plane [14]. For the planar Lyapunov family, a pitchfork bifurcation occurs at a critical energy when a 1 : 1 resonance between the in-plane and vertical frequencies causes an eigenvalue of $D\Pi(\mathbf{p})$ to cross +1. This causes the orbit to become unstable and creates two, new stable fixed points in the map Σ , which correspond to the northern and southern halo families [13].

The full dynamics in a neighbourhood of the equilibria are governed by the interactions of the centre, stable, and unstable manifolds of the equilibrium points:

$$W_l^s(\mathbf{x}) = \{\mathbf{x} \in \mathbb{R}^6 : \phi_t(\mathbf{x}) \rightarrow \mathbf{x}_0\} \quad \text{as } t \rightarrow \infty, \quad (2.13)$$

$$W_l^s(\mathbf{x}) = \{\mathbf{x} \in \mathbb{R}^6 : \phi_t(\mathbf{x}) \rightarrow \mathbf{x}_0\} \quad \text{as } t \rightarrow -\infty, \quad (2.14)$$

where $\phi_t(\mathbf{x})$ denotes the trajectory of the system at $\mathbf{x}$. For the equilibrium points themselves, these manifolds are one-dimensional and tangent to E^s and E^c at $\mathbf{x}_0$. More significantly, every periodic and quasi-periodic orbit Λ within the centre manifold of these equilibria possesses stable and unstable directions inherited from the hyperbolic structure of the equilibrium. Each Λ is itself a compact invariant manifold of the flow ϕ_t . The tangent bundle to the phase space restricted to Λ can be decomposed as

$$T\mathbb{R}^6|_{\Lambda} = T\Lambda \oplus N\Lambda, \quad (2.15)$$

where $T\Lambda$ is the tangent bundle of the orbit and $N\Lambda$ is its normal bundle [15]. The dynamics in $N\Lambda$ are described by the Floquet multipliers of the periodic orbit, which are perturbations of the exponential rates $e^{\pm\nu T}$ inherited from the equilibrium, where T is the orbit period. Since one Floquet multiplier lies outside and one inside the unit circle, Λ is a **Normally Hyperbolic Invariant Manifold (NHIM)**. This is because the transverse directions to the periodic orbit exhibit exponential expansion and contraction that dominate the neutral dynamics along the orbit itself, fulfilling the condition of normal hyperbolicity and ensuring the manifold's persistence under small perturbations [16]. By *Fenichel's* theorems [17], [18], such a **NHIM** possesses locally unique, smooth stable and unstable manifolds $W^s(\Lambda)$ and $W^u(\Lambda)$, which are fibrations over the orbit Λ with dimensions:

$$\dim(W^s(\Lambda)) = \dim(\Lambda) + \dim(E^s) = 1 + 1 = 2, \quad (2.16)$$

$$\dim(W^u(\Lambda)) = \dim(\Lambda) + \dim(E^u) = 1 + 1 = 2. \quad (2.17)$$

In the full six-dimensional phase space, these manifolds form geometric structures that guide asymptotic motion towards and away from periodic orbits in the vicinity of libration points.

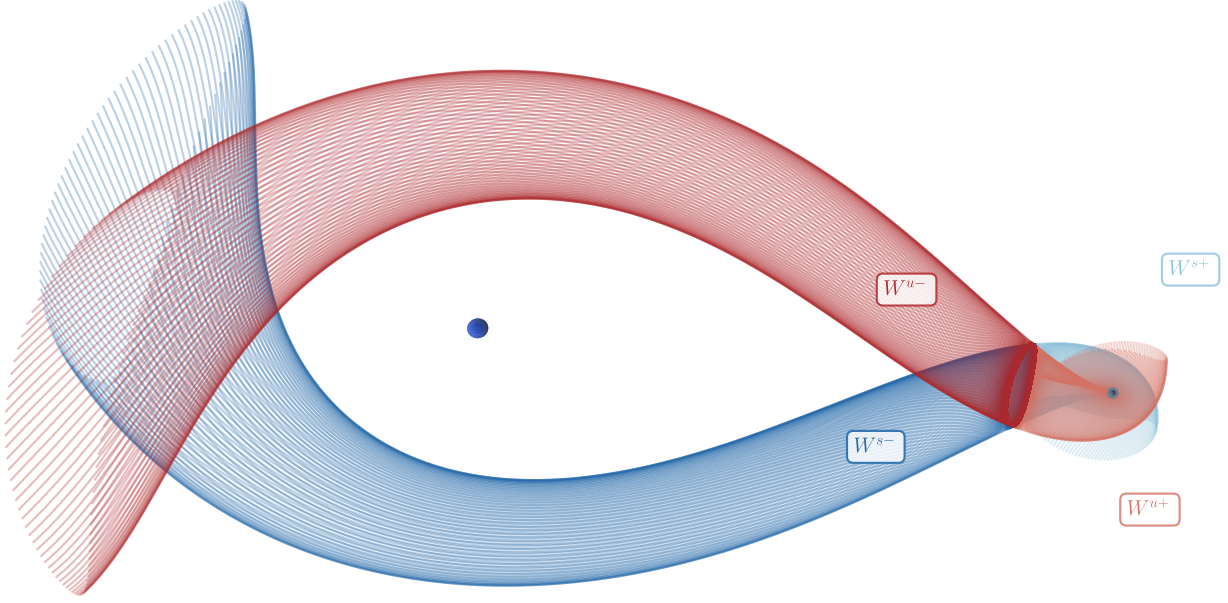


Figure 2.2: Invariant manifolds of an Earth-Moon L_1 halo orbit.

Figure 2.2 shows these structures for an Earth-Moon L_1 halo orbit. The existence of these structures have direct implications for astrodynamics and space-mission design; as they form natural separatrices that connect the solar system. The stable and unstable manifolds $W^s(\Lambda)$ and $W^u(\Lambda)$ extend throughout the phase space. Their intersections (known as heteroclinic connections), when they exist, form the backbone of a natural transport network. A heteroclinic connection is defined as:

$$W^i(\Lambda_1) \cap W^j(\Lambda_2) \neq \emptyset, \quad (i, j) \in \{(s, u), (u, s)\}, \quad (2.18)$$

where Λ_1 and Λ_2 are two distinct periodic orbits [11]. A trajectory within this set, known as a heteroclinic orbit, is one that departs asymptotically from Λ_1 and arrives asymptotically at Λ_2 . A second type of intersection, known as homoclinic connections, also exists. In this case, they are defined as:

$$W^u(\Lambda) \cap W^s(\Lambda) \neq \emptyset. \quad (2.19)$$

A homoclinic orbit is a trajectory that leaves the vicinity of Λ along its unstable manifold, and after a long excursion through phase-space, returns along its stable manifold. The existence of a single homoclinic connection implies the existence of infinitely many, creating a structure known as the *Smale* horseshoe [11]. If two manifolds W_1 and W_2 intersect, transversely, or their tangent spaces at the intersection span the tangent space of the ambient manifold:

$$T_p(W_1) + T_p(W_2) = T_p(\mathbb{R}^6), \quad (2.20)$$

then the intersection is structurally stable [19]. If the system’s vector field is slightly perturbed (for example, from solar radiation pressure), the intersection will not disappear; it will merely be slightly displaced, reflecting the structural stability of transverse intersections of **NHIMs** [19]. Let Γ_1 and Γ_2 be **NHIMs** with stable and unstable manifolds $W^s(\Gamma_i)$ and $W^u(\Gamma_i)$, for $i = 1, 2$. A transverse intersection occurs at a point p such that:

$$T_p W^u(\Gamma_1) + T_p W^s(\Gamma_2) = T_p \mathbb{R}^6. \quad (2.21)$$

By the λ -lemma [20], if a small manifold D intersects $W^s(\Gamma_2)$ transversely, its forward image under the flow $\phi_t(D)$ is C^1 -close to $W^u(\Gamma_2)$ for sufficiently large $t > 0$, effectively guiding trajectories along the heteroclinic connection. This ensures that trajectories do not require infinite precision to follow these pathways, as the flow itself aligns them along the unstable directions of the **NHIMs**. The collection of all such stable and unstable manifolds forms a dynamic network of "tubes" through phase space. For example, the unstable manifold of a Sun-Earth L_2 halo orbit, $W^u(L_2)$, can intersect the stable manifold of a Sun-Mars L_1 orbit, $W^s(L_1)$, and make a low-energy transfer from Earth to Mars possible.

2.3 Invariant Manifolds and Space Mission Design

The first use of invariant manifolds theory for trajectory design can be attributed to *Gómez, Jorba, Masdemont, and Simó* [21], which searched for transfers between an Earth parking orbit and L_1 . Since then, this has been applied extensively to trajectory design in multi-body environments.

The *Genesis* solar-wind sample return spacecraft was placed on a Sun–Earth L_1 halo orbit using invariant-manifold guidance. Its mission design explicitly exploited conjectured heteroclinic connections between L_1 and L_2 periodic orbits, allowing low Δv transfers into and out of the halo orbit. [22]

Two repurposed *THEMIS* satellites (*P1* and *P2*) for the *Artemis* program were sent from Earth orbit into lunar libration point orbits via invariant manifolds. *P1*’s transfer followed an unstable manifold of a Sun–Earth L_2 halo to enter an L_1 Lissajous orbit, while *P2* entered an L_1 Lissajous orbit along its stable manifold. In both cases, the spacecraft required no large insertion burn; they “fell” into the libration point orbit along the invariant manifold tubes. [23]

More exotic studies, such as multiple-moon tours of the Jupiter system have been proposed, such as *DEIMOS* developed patched-tour trajectories visiting Callisto, Ganymede, Europa, and Io using weak stability boundary arcs. [24]

Computing invariant manifolds in the **CR3BP** is an indispensable step in space mission design exploiting low-energy capture and escape. Historically, invariant manifolds are computed by finding the stable and unstable directions through the eigendecomposition of the monodromy matrix of a periodic orbit around a libration point. The stable and unstable directions are

then propagated along the entire orbit using the [STM](#) [4]. This method is too complex for widespread use in practical mission design, as the manifolds are defined only for periodic orbits, whereas these do not occur in high-fidelity models. Additionally, certain missions require analysis of millions of resonant orbits, which is not practical to implement on modern computing hardware [5]. Novel methods, such as the ones developed by *Haro* [25] and *Kumar* [26] instead rely on computing both the quasi-periodic invariant tori and the invariant manifolds of the tori, and then use a recursive parametrisation method to approximate the invariant manifolds. Although the latter approach already offers benefits in terms of efficiency, *Vega et al.* [5] found that the central direction was sufficient to obtain an approximation of the invariant manifold. By eliminating the need to compute the stable and unstable directions, they were able to develop an algorithm that could be parallelised across multiple [Graphics Processing Unit \(GPU\)](#) cores, and outperformed previous algorithms by several orders of magnitude.

Although this algorithm is a major step forward in computing invariant manifolds, it only approximates the true asymptotic manifold. Moreover, the traditional monodromy-based approach is not easily parallelised on [GPUs](#). Although these models work well for cases where the manifolds belong to periodic orbits, they do not translate well to higher-fidelity dynamical models, such as in non-periodic or quasi-periodic cases, or when considering n -body ephemerides and perturbations. While in the [CR3BP](#), invariant manifolds offer a convenient roadmap for low-energy trajectories, in higher-fidelity models, the analogous structure increases in complexity and computational requirements increase exponentially [27]. This means that the current low-energy trajectory design framework is unsuitable for real-time optimisation and fast transfer cost estimation [28], highlighting the need for novel approaches that maintain the benefit of manifold-driven design while handling the complexity of higher-fidelity models and improving efficiency.

Given these computational challenges, researchers have explored alternative approaches to trajectory design. One promising direction is machine learning, which offers a data-driven framework to approximate manifold structures. By learning implicit representations from trajectory data, machine learning methods have the potential to bypass the computational bottlenecks of classical techniques and enable real-time evaluations that were previously infeasible.

2.4 Machine Learning Approaches to Dynamical Systems

In recent years, there has been increased interest in [Machine Learning \(ML\)](#) techniques to address the computational and practical limitations of classical analytical methods. The application of [ML](#) to spacecraft dynamics represents a paradigm shift from constructing explicit mathematical models to learning implicit representations from data. This transition has been driven by several factors: the exponential growth in computational power, the accumulation of high-fidelity simulation data, and the success of [ML](#) in solving complex problems in other domains.

Within the context of orbital mechanics and trajectory design, several [ML](#) approaches have

shown particular promise. [Artificial Neural Network \(ANN\)](#)s have been successfully applied to learn optimal control policies for spacecraft manoeuvres, with recent work demonstrating that deep networks can capture the underlying Hamilton-Jacobi-Bellman equations governing optimal trajectories [29]. [Support Vector Machines \(SVM\)](#)s have proven effective for trajectory classification in the [CR3BP](#), rapidly distinguishing between transit and non-transit orbits, a task that traditionally requires expensive numerical integration [30]. Random Forests and Gaussian Process Regression have been employed to approximate invariant manifold structures, achieving comparable accuracy to traditional methods while offering orders-of-magnitude improvements in evaluation speed [31].

The emergence of [Deep Learning \(DL\)](#) has further expanded the possibilities. *Guidance and Control Networks*, trained through imitation learning on databases of optimal trajectories, can replace complex guidance algorithms with a single neural network evaluation. These networks have demonstrated remarkable generalisation capabilities, suggesting they learn fundamental dynamical principles rather than merely memorising training examples [32]. For problems involving visual navigation or sensor fusion, [Convolutional Neural Network \(CNN\)](#)s combined with recurrent architectures have enabled end-to-end learning from raw sensor data to control commands [33].

Even more novel is the introduction of generative models. Unlike discriminative approaches, which focus on mapping inputs to outputs, generative models aim to capture the underlying probability distribution of the data, enabling the synthesis of new, physically consistent samples. In the context of dynamical systems and trajectory design, this capability opens several promising avenues.

Diffusion models, a recent advancement in the generative modelling landscape, have demonstrated superior performance in learning complex, multimodal distributions [34]. Their application to long-term propagation problems or Monte Carlo simulations enables the generation of diverse yet dynamically consistent trajectories, potentially reducing the need for costly ensemble integrations.

[Variational Autoencoder \(VAE\)](#)s and [Generative Adversarial Network \(GAN\)](#)s can be trained on datasets of optimal or feasible trajectories to learn a latent representation that encodes dynamical properties. Sampling from this latent space allows for the generation of novel trajectories, effectively serving as a low-dimensional generative prior for mission planning.

Among the broad class of generative models, [VAEs](#) have garnered particular interest for their ability to learn structured, continuous latent representations of data [35]. [VAEs](#) are especially attractive in the context of dynamical systems, as they provide a probabilistic framework for encoding high-dimensional trajectories into a lower-dimensional latent space in a manner that preserves the underlying geometry of the system.

2.5 Applications of Machine Learning in the Circular Restricted Three Body Problem

Recently, *Gil et al.* [36] applied VAEs for generative design of orbits in the CR3BP. Demonstrating that VAEs can effectively learn and represent characteristics from a comprehensive dataset of periodic orbits in the Earth-Moon system. By sampling the VAE’s latent space, the model can generate new, approximated periodic trajectories which, after being refined by a multiple-shooting algorithm, successfully converge to physically accurate periodic orbits that were not present in the original training data. This proved the potential of generative Artificial Intelligence (AI) for orbit discovery, by reducing analysis costs, and enabling the creation of new types of orbits.

While classical autoencoders provide deterministic mappings between input and latent spaces, they lack the probabilistic structure necessary for generation of new samples. VAEs address this limitation by introducing a probabilistic framework that enables both reconstruction and generation capabilities. A classical autoencoder consists of an encoder $f_\phi : \mathbb{R}^n \rightarrow \mathbb{R}^d$ that maps input data x to a latent representation z , and a decoder $g_\theta : \mathbb{R}^d \rightarrow \mathbb{R}^n$ that reconstructs the input from this representation [37]. The training objective minimises the reconstruction error:

$$\mathcal{L}_{AE} = \|x - g_\theta(f_\phi(x))\|^2 \quad (2.22)$$

VAEs reformulate the autoencoder framework within a probabilistic generative model. Instead of learning deterministic encodings, a VAE learns to estimate the parameters of a probability distribution over latent variables. The key insight is to model the data generation process as:

$$p_\theta(x) = \int p_\theta(x|z)p(z)dz \quad (2.23)$$

$$p(z) \sim \mathcal{N}(0, I) \quad (2.24)$$

$$p_\theta(x|z) = \text{Decoder}(z; \theta) \quad (2.25)$$

where $p(z)$ is a prior distribution (typically standard normal), and $p_\theta(x|z)$ is the decoder that defines the likelihood of data given latent variables.

The fundamental challenge lies in computing the posterior distribution $p(z|x) = \frac{p_\theta(x|z)p(z)}{p_\theta(x)}$, which is intractable due to the intractable marginal likelihood $p_\theta(x)$. VAEs address this through variational inference by introducing an approximate posterior $q_\phi(z|x)$ parametrised by an encoder network. The VAE training objective is derived by optimising the Evidence Lower Bound (ELBO), which provides a tractable lower bound on the log marginal likelihood. Starting from the log-likelihood:

$$\log p_\theta(x) = \log \int p_\theta(x|z)p(z)dz \quad (2.26)$$

$$= \log \int p_\theta(x|z)p(z) \frac{q_\phi(z|x)}{q_\phi(z|x)} dz \quad (2.27)$$

$$= \log \mathbb{E}_{q_\phi(z|x)} \left[\frac{p_\theta(x|z)p(z)}{q_\phi(z|x)} \right] \quad (2.28)$$

Applying Jensen's inequality (since log is concave):

$$\log p_\theta(x) \geq \mathbb{E}_{q_\phi(z|x)} \left[\log \frac{p_\theta(x|z)p(z)}{q_\phi(z|x)} \right] \quad (2.29)$$

$$= \mathbb{E}_{q_\phi(z|x)} [\log p_\theta(x|z)] - D_{KL}(q_\phi(z|x) || p(z)) \quad (2.30)$$

$$= \mathcal{L}_{ELBO}(\theta, \phi; x) \quad (2.31)$$

The **ELBO** consists of the reconstruction term, $\mathbb{E}_{q_\phi(z|x)} [\log p_\theta(x|z)]$, which encourages the decoder to reconstruct input data accurately from latent samples; and the regularisation term $D_{KL}(q_\phi(z|x) || p(z))$, which constrains the approximate posterior to remain close to the prior, ensuring the latent space remains structured and allows for sampling. [38]

A critical challenge in optimising the **ELBO** is computing gradients with respect to the encoder parameters ϕ through the expectation $\mathbb{E}_{q_\phi(z|x)} [\log p_\theta(x|z)]$. Direct sampling from $q_\phi(z|x)$ would make this expectation non-differentiable with respect to ϕ .

The reparametrisation trick resolves this by expressing samples from $q_\phi(z|x)$ as a deterministic function of the parameters and an independent random variable. For a Gaussian approximate posterior $q_\phi(z|x) = \mathcal{N}(\mu_\phi(x), \sigma_\phi^2(x))$:

$$z \sim q_\phi(z|x) \quad (2.32)$$

$$\Leftrightarrow z = \mu_\phi(x) + \sigma_\phi(x) \odot \epsilon, \quad \epsilon \sim \mathcal{N}(0, I) \quad (2.33)$$

This reparametrisation moves the stochasticity to the independent noise term ϵ , allowing gradients to flow through $\mu_\phi(x)$ and $\sigma_\phi(x)$. The expectation becomes:

$$\mathbb{E}_{q_\phi(z|x)} [\log p_\theta(x|z)] = \mathbb{E}_{\epsilon \sim \mathcal{N}(0, I)} [\log p_\theta(x | \mu_\phi(x) + \sigma_\phi(x) \odot \epsilon)] \quad (2.34)$$

This can be approximated using Monte Carlo sampling and is fully differentiable with respect to both θ and ϕ . A **VAE** consists of two neural networks. The first component is the encoder, which maps input data to a distribution over latent variables:

$$q_\phi(z|x) = \mathcal{N}(\mu_\phi(x), \sigma_\phi^2(x)), \quad (2.35)$$

where μ_ϕ and σ_ϕ are neural networks that output the mean and variance of the approximate posterior. The decoder on the other hand reconstructs the original data from latent samples:

$$p_\theta(x|z), \quad (2.36)$$

which is typically modelled as Gaussian for continuous data or Bernoulli for binary data [39]. The complete VAE loss function combines both terms of the ELBO:

$$\mathcal{L}_{VAE} = -\mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)] + D_{KL}(q_\phi(z|x)||p(z)) \quad (2.37)$$

$$= \|x - \hat{x}\|^2 + D_{KL}(\mathcal{N}(\mu_\phi(x), \sigma_\phi^2(x))||\mathcal{N}(0, I)) \quad (2.38)$$

For the Gaussian case, the Kullback-Leibler (KL) divergence has a closed form:

$$D_{KL}(q_\phi(z|x)||p(z)) = \frac{1}{2} \sum_{i=1}^d (1 + \log(\sigma_{\phi,i}^2(x)) - \mu_{\phi,i}^2(x) - \sigma_{\phi,i}^2(x)) \quad (2.39)$$

2.5.1 State of the Art

Traditional VAEs suffer from known limitations when applied to manifold learning. The assumption of a continuous Gaussian latent space may result in blurry reconstructions and difficulty capturing the fine-grained topological structure of non-linear dynamical manifolds, such as those arising in the CR3BP. Additionally, VAEs trained on data suffering from latent variable non-identifiability suffer from posterior collapse, which renders the model unable to distinguish the learned posterior distribution $q(z|x)$ from the prior $p(z)$ regardless of the input x [40]. Further limitations include the tendency of the Gaussian decoder likelihood to produce overly smooth reconstructions that average over multiple possible outputs, and difficulty capturing temporal dependencies in sequential trajectory data.

To address these challenges, the more recent VQ-VAE variant is employed [41]. The key insight behind VQ-VAEs is replacing the continuous Gaussian latent space with a discrete codebook of learnable embeddings. This discrete approach offers several critical advantages: avoiding posterior collapse by eliminating the problematic KL regularisation term, providing natural clustering of similar dynamical behaviours, enabling sharper reconstructions without continuous distribution smoothing effects, and compatibility with powerful autoregressive priors for structured generation.

A VQ-VAE maintains a discrete codebook and quantises the continuous encoder outputs through a nearest-neighbour lookup. Like a traditional VAE, a VQ-VAE consists of three

main components: an encoder, a quantisation layer, and a decoder. The encoder maps input data $x \in \mathbb{R}^n$ to a continuous latent vector $z_e(x) \in \mathbb{R}^d$. However, instead of using $z_e(x)$ directly, it is first quantised through a vector quantisation layer. The model defines a learnable codebook or embedding space $e \in \mathbb{R}^{K \times D}$, which acts as a learned dictionary. The embedding space e has dimensions $K \times D$, where K is the size of the discrete latent space (a K -way categorical dimension), and D is the dimensionality of each individual latent embedding vector e_i . There are K distinct vectors $e_i \in \mathbb{R}^D$. Each continuous encoded vector $z_e(x)$ is matched to the closest codeword e_k from the codebook using a nearest-neighbour lookup:

$$q(z = k | x) = \begin{cases} 1 & \text{for } k = \arg \min_j \|z_e(x) - e_j\|_2, \\ 0 & \text{otherwise.} \end{cases} \quad (2.40)$$

The corresponding embedding vector e_k is chosen as the quantised latent variable $z_q(x) = e_k$. The $z_q(x)$ serves as the input to the decoder network. The posterior categorical distributions are defined as one-hot, making it a deterministic process. The decoder takes the quantised embedding vector $z_q(x)$ as its input and reconstructs the original input data x , effectively defining a distribution $p(x|z)$ over the input data.

A critical challenge arises from the discrete quantisation operation $z_q(x) = e_{\arg \min_j \|z_e(x) - e_j\|_2}$, which is non-differentiable and prevents standard backpropagation. **VQ-VAEs** address this through the straight-through estimator: during the backward pass, gradients from the decoder are copied directly to the encoder output, bypassing the non-differentiable quantisation step. Mathematically, this approximates $\frac{\partial z_q}{\partial z_e} \approx I$, becoming more accurate as training progresses and $z_e(x)$ and $z_q(x)$ converge. The overall **VQ-VAE** objective function consists of three terms:

$$\mathcal{L}_{recon} = \|x - \hat{x}\|^2, \quad (2.41)$$

$$\mathcal{L}_{vq} = \|\text{sg}[z_e(x)] - e_k\|^2, \quad (2.42)$$

$$\mathcal{L}_{commitment} = \beta \|z_e(x) - \text{sg}[e_k]\|^2, \quad (2.43)$$

where $\text{sg}[\cdot]$ denotes the stop gradient operator, which ensures that gradients are not propagated through certain paths during backpropagation. The reconstruction term $\mathcal{L}_{recon}$ penalises reconstruction error. Unlike traditional **VAEs**, there is no probabilistic interpretation here; the loss directly measures the L2 distance between input and reconstruction. Gradients flow to both decoder parameters θ and, via the straight-through estimator, to encoder parameters ϕ :

$$\frac{\partial \mathcal{L}_{recon}}{\partial \theta} = \frac{\partial \|x - g_\theta(z_q(x))\|^2}{\partial \theta}, \quad (2.44)$$

$$\frac{\partial \mathcal{L}_{recon}}{\partial \phi} = \frac{\partial \|x - g_\theta(z_q(x))\|^2}{\partial z_q} \frac{\partial z_q}{\partial z_e} \frac{\partial z_e}{\partial \phi}, \quad (2.45)$$

where the middle term $\frac{\partial z_q}{\partial z_e} = I$ through the straight-through estimator approximation. The codebook loss $\mathcal{L}_{vq}$ specifically updates the selected codebook entry e_k to move closer to the encoder output $z_e(x)$. The stop-gradient operator is crucial as it ensures that gradients only flow to the codebook embeddings e_k , not back to the encoder:

$$\frac{\partial \mathcal{L}_{vq}}{\partial e_k} = 2(e_k - z_e(x)), \quad (2.46)$$

$$\frac{\partial \mathcal{L}_{vq}}{\partial z_e(x)} = 0. \quad (2.47)$$

This asymmetric gradient flow is essential for stable training; without it, both the encoder and codebook would chase each other, leading to instability. The commitment term $\mathcal{L}_{commitment}$ encourages the encoder output to commit to chosen codebook entries. Here, the stop gradient operates on e_k , so gradients only flow to the encoder:

$$\frac{\partial \mathcal{L}_{commitment}}{\partial z_e(x)} = 2\beta(z_e(x) - e_k), \quad (2.48)$$

$$\frac{\partial \mathcal{L}_{commitment}}{\partial e_k} = 0. \quad (2.49)$$

The hyperparameter β typically ranges from 0.1 to 0.5, balancing reconstruction quality against encoder commitment. Higher values force stronger commitment but may degrade reconstruction quality.

A critical aspect of [VQ-VAE](#) training is ensuring effective utilisation of the codebook. Several phenomena can occur during training, the most common are codebook and index collapse [\[40\]](#). In the first case, codebook entries may never be selected, becoming "dead" entries that don't contribute to learning. This reduces the effective capacity of the model and can be monitored by tracking codebook utilisation statistics. In the latter, the model may learn to use only a small subset of codebook entries, effectively reducing the discrete latent space dimensionality. To address this, restart mechanisms that reinitialise unused codebook entries with encoder outputs from randomly sampled data points as well as alternative codebook update schemes are employed. One of these update schemes is the exponential moving averages instead of gradient-based updates:

$$e_k^{(t+1)} = \gamma e_k^{(t)} + (1 - \gamma) z_e(x) \quad \text{if } k = \arg \min_j \|z_e(x) - e_j\|_2, \quad (2.50)$$

where γ is a momentum parameter (typically 0.99) [\[42\]](#). This approach can provide more stable codebook learning but requires careful tuning. Crucially, the [VQ-VAE](#) introduces a discretised latent space that lends itself to semantic clustering: trajectories with similar

dynamical characteristics are likely to be mapped to the same or nearby codebook entries. Because the quantization is performed at multiple spatial or temporal positions, the effective capacity of the model is exponentially greater than the size of the codebook itself. Specifically, for a latent sequence of length L and codebook of size K , the number of unique latent representations is K^L . Thus, the model retains a high capacity for expressing complex, multi-modal distributions while maintaining the interpretability of a discrete latent space.

In the context of [CR3BP](#) trajectory learning, [VQ-VAEs](#) can serve as powerful representation learners by discretising the latent space into a compact set of learned prototypes. Each codeword may correspond to a distinct pattern or segment of trajectory data, potentially reflecting localised structures in the underlying phase space. These discrete representations can be decoded to reconstruct approximate trajectories, though their physical feasibility depends on decoder architecture and training constraints, and can be sampled in a structured manner using autoregressive or diffusion-based priors.

2.6 Aims and Objectives

The computational cost and methodological rigidity of traditional techniques for calculating invariant manifolds remain a major constraint in contemporary astrodynamics. In particular, methods based on variational equations and eigendecomposition scale poorly when applied to large families of orbits or when extended to higher-fidelity models that lack periodic solutions. This poses a challenge for both preliminary trajectory analysis and the development of new mission concepts that rely on low-energy transfers. Motivated by these limitations, this research explores a data-driven alternative for approximating manifold structures in the [CR3BP](#), using recent advances in generative deep learning.

The principal aim of this work is to develop and assess a generative modelling framework capable of learning and reproducing the geometric and dynamical structure of invariant manifolds in the [CR3BP](#). Specifically, the use of a [VQ-VAE](#) to encode trajectory data into a discrete latent space, conditioned on physical parameters such as the Jacobi constant and manifold branch, is investigated. The proposed model is trained on a dataset of high-fidelity stable and unstable manifold trajectories associated with Earth–Moon halo orbits, which are computed through a centre manifold reduction and differential correction process. To ensure dynamical consistency, the network is augmented with a physics-informed loss that penalises deviations from the [CR3BP](#) equations of motion during reconstruction. The objectives are as follows:

- Develop a generative modelling framework to learn and reproduce the geometric and dynamical structure of invariant manifolds in the [CR3BP](#).
- Investigate the use of a [VQ-VAE](#) to encode trajectory data into a discrete latent space, conditioned on physical parameters such as the Jacobi constant and manifold branch.
- Generate a high-fidelity dataset of stable and unstable manifold trajectories associated with Earth–Moon halo orbits, computed via centre manifold reduction and differential

correction.

- Augment the generative model with a physics-informed loss function to ensure dynamical consistency by penalising deviations from the [CR3BP](#) equations of motion.
- Evaluate the model’s ability to reconstruct manifold trajectories with high fidelity and generate new, dynamically consistent samples from the latent space.
- Compare generated trajectories with the original dynamical model to assess both qualitative and quantitative accuracy.
- Explore the potential of generative architectures to support data-driven trajectory design and orbit discovery in multi-body dynamical systems.
- Lay the groundwork for future extensions to perturbed models, non-periodic structures, and higher-dimensional systems beyond the classical [CR3BP](#).

Chapter 3

Methodology

Given the lack of readily available datasets of invariant manifolds, a custom Python package [43] was developed to aid in the generation of high fidelity seed orbits and their associated manifolds. Section 3.1 discusses the procedure of finding and generating invariant manifolds, while Section 3.2 discusses the network architecture and training hyperparameters.

3.1 Numerical Computation of Invariant Manifolds

As invariant manifolds are dynamical structures of periodic orbits, identifying suitable periodic orbits is a prerequisite for manifold generation. The present paper restricts attention to halo orbits, which are the most commonly employed orbits for low-energy transfers [23], [44]. Halo orbits are of special interest in mission design for several reasons. Halo orbits offer unobstructed views and continuous communication. For example, spacecraft in certain halo orbits in Sun-Earth L_2 never passes in the shadow of the earth or the moon, providing it with continuous solar power, thermal stability, and uninterrupted line-of-sight to scientific and communication targets [9], [44]. Additionally, while halo orbits are fundamentally unstable, it takes remarkably little fuel to stay on one, in some cases, requiring less than 1 m s^{-1} in a 4 year period [45].

3.1.1 Reduction to the Centre Manifold

To find a suitable generating orbit, a lie series expansion is performed to reduce the hamiltonian to the centre manifold, following the methods of *Jorba* and *Masdemont* [13]. An overview of the transformation process is shown in Figure 3.1.

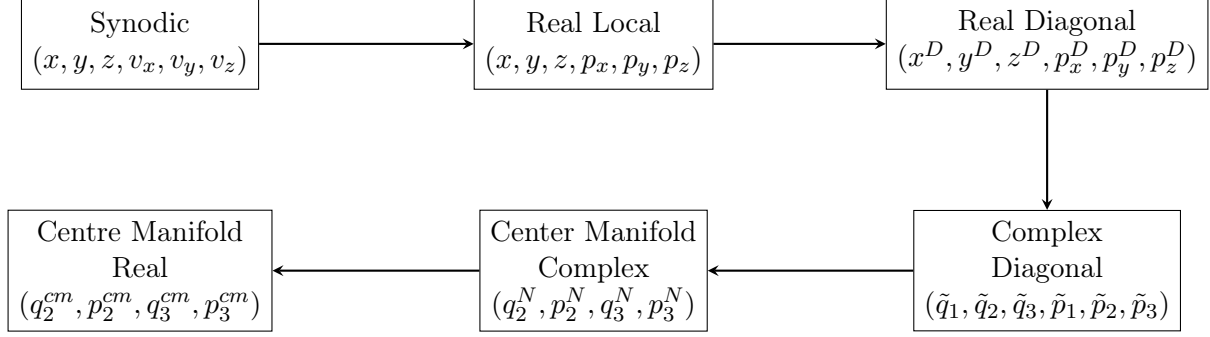


Figure 3.1: Transformation process from synodic to centre manifold frames.

The process begins with the Hamiltonian of the [CR3BP](#), translated to the collinear points. The first step is to analyse the linear dynamics by examining the quadratic part of the Hamiltonian, $H^{(2)}$. The characteristic equation of the linearized system reveals a centre $\times$ centre $\times$ saddle structure. This means the motion consists of two stable, oscillatory modes and one unstable, hyperbolic mode.

A symplectic transformation, derived from the eigenvectors of the linearised dynamics about each point is applied to diagonalise the quadratic Hamiltonian. This puts $H^{(2)}$ into the form:

$$H^{(2)} = \lambda_1 x p_x + \frac{\omega_1}{2}(y^2 + p_y^2) + \frac{\omega_2}{2}(z^2 + p_z^2). \quad (3.1)$$

Here, λ_1 is the real eigenvalue associated with the hyperbolic motion, while ω_1 and ω_2 are the frequencies of the planar and vertical oscillatory motions, respectively. x , y , and z are the generalised coordinates; and p_x , p_y and p_z are the generalised momenta of the hamiltonian system. This form clearly separates the unstable and stable components. To further simplify the algebraic manipulations required for higher-order terms, a canonical complexification transformation is applied. This converts the real normal form into a simpler, complex diagonal form:

$$H^{(2)} = \lambda_1 q_1 p_1 + i\omega_1 q_2 p_2 + i\omega_2 q_3 p_3. \quad (3.2)$$

In this representation, the coordinate pair (q_1, p_1) corresponds to the hyperbolic motion, while (q_2, p_2) and (q_3, p_3) correspond to the two oscillatory motions. Applying the same transformations to the full Hamiltonian results in higher-order terms $(H^{(3)}, H^{(4)}, \dots)$ that couple the hyperbolic and elliptic motions. The goal is to find a canonical transformation that removes these coupling terms. This is achieved using a partial lie series transform. A generating function, G , is constructed degree by degree $(G = G^{(3)} + G^{(4)} + \dots)$ to eliminate these terms. For each degree j , the corresponding generator $G^{(j)}$ is found by solving the homological equation:

$$\{H^{(2)}, G^{(j)}\} = -H_{\text{elim}}^{(j)}, \quad (3.3)$$

where $H_{\text{elim}}^{(j)}$ contains the terms in the Hamiltonian of degree j that couple the hyperbolic and elliptic modes (monomials where the exponents of q_1 and p_1 are not equal). Applying the transformation generated by G results in a new Hamiltonian where the hyperbolic variables only appear as the product $q_1 p_1$. After normalization, the Hamiltonian takes the form $\bar{H}(q_1 p_1, q_2, p_2, q_3, p_3)$. The dynamics on the centre manifold are obtained by eliminating the hyperbolic terms. This is accomplished by setting the hyperbolic action to zero:

$$q_1 p_1 = 0. \quad (3.4)$$

This effectively reduces the system from 3 to 2 degrees of freedom. The resulting Hamiltonian describes the system's long-term, bounded dynamics in the vicinity of the libration point. An efficient numerical algorithm was developed to aid in the computation of the centre manifold hamiltonian and corresponding coordinate transformation process. Details can be found in [Appendix A](#). To maintain a balance between accuracy and computation time, expansions are done to degree 12. After the Hamiltonian in the centre manifold is computed, the equations of motion can be derived:

$$\dot{q}_2 = \frac{\partial H}{\partial p_2}, \quad \dot{q}_3 = \frac{\partial H}{\partial p_3}, \quad (3.5)$$

$$\dot{p}_2 = -\frac{\partial H}{\partial q_2}, \quad \dot{p}_3 = -\frac{\partial H}{\partial q_3}. \quad (3.6)$$

A common tool to understand the dynamics on the 4-dimensional centre manifold, W^c , is the use of a Poincaré map. Since the centre manifold is, by definition, invariant under the flow ϕ_t , any trajectory starting on it remains on it. The system also admits the conserved Jacobi integral, $H(\mathbf{x}) = C$. This quantity is conserved for all trajectories, including those confined to W^c .

Therefore, the flow restricted to the centre manifold, $\phi_t|_{W^c}$, is studied at a constant energy level. Under this restriction, bounded orbits with energy C lie on a three-dimensional manifold, namely the level set of the Hamiltonian function restricted to W^c .

To analyse the continuous three-dimensional flow, a two-dimensional Poincaré section, Σ , is introduced, chosen to be transverse to the flow within the three-dimensional energy surface. The section is defined in momentum space by recording intersections with the hyperplane $p_2 = 0$:

$$\Sigma = \{\mathbf{x} = (\mathbf{q}, \mathbf{p}) \in W^c \mid p_2 = 0, H(\mathbf{x}) = C, \dot{p}_2 > 0\}, \quad (3.7)$$

where the transversality condition is given by the sign of $\dot{p}_2 = -\partial H / \partial q_2$. The resulting Poincaré map, $\Pi : \Sigma \rightarrow \Sigma$, is then defined for a point $\mathbf{p}_k \in \Sigma$ as the next intersection point of its trajectory:

$$\Pi(\mathbf{p}_k) = \mathbf{p}_{k+1} = \phi_\tau(\mathbf{p}_k), \quad (3.8)$$

where τ is the time of first return to Σ . This procedure reduces the study of the original 6D continuous flow to the analysis of a 2D discrete map. This allows us to distinguish between four main types of behaviour. A fixed point of the map, where $\Pi(\mathbf{p}) = \mathbf{p}$, corresponds to a periodic orbit in the original system that intersects Σ exactly once per period. A k -periodic point, where $\Pi^k(\mathbf{p}) = \mathbf{p}$, corresponds to a periodic orbit that makes k crossings of Σ before closing. An invariant curve under the map corresponds to a quasi-periodic orbit (a torus) in the full flow. Finally, regions of chaos can also be visually identified. Halo orbits can be located by starting at an energy level C just above that of the libration-point equilibrium, where the map is typically dominated by a single stable fixed point corresponding to the planar Lyapunov family. Computations employ a fourth-order Runge-Kutta integration scheme. [Figure 3.2](#) shows the Poincaré map of Earth–Moon L_1 at an energy $C = 0.2$ above the libration point.

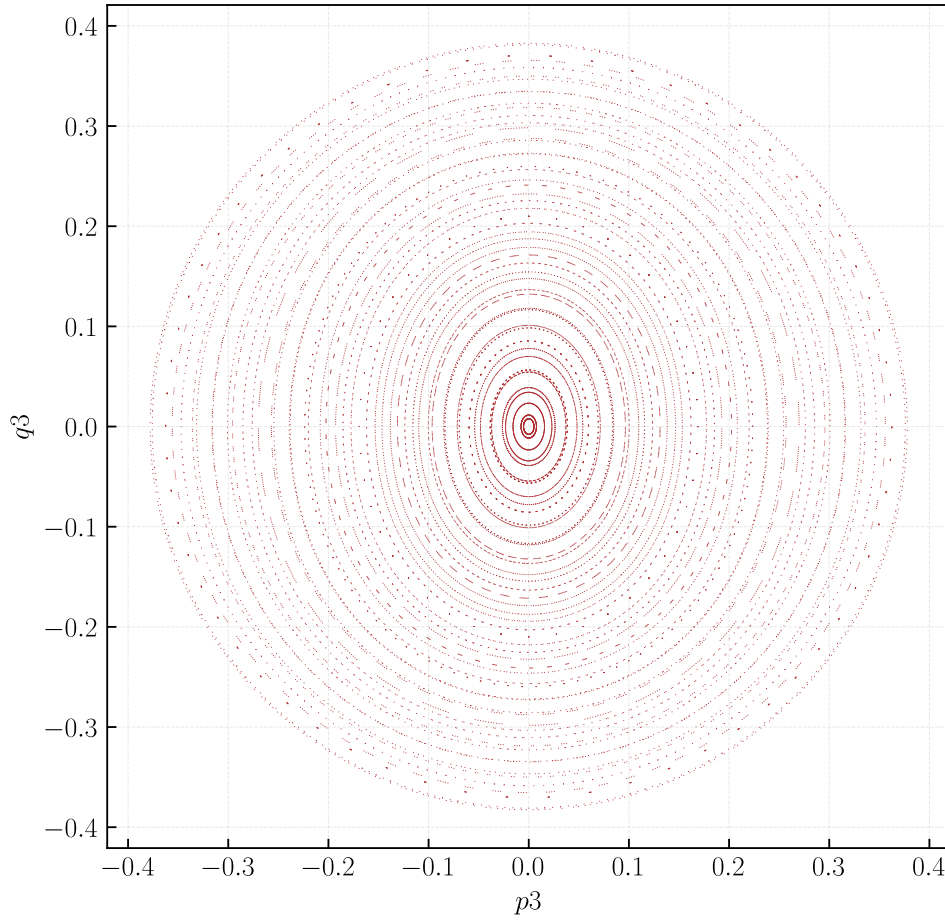


Figure 3.2: Poincaré map of the $p_2 = 0$ section of the Earth–Moon L_1 libration point at energy $C = 0.2$.

The search is for a critical energy at which the fixed point undergoes a pitchfork bifurcation, losing stability and generating two new, symmetrically-located stable fixed points. These new fixed points are the seeds of the northern and southern halo orbit families. [Figure 3.3](#) shows the same map at an energy level of $C = 1.0$ above the libration point. The northern and southern halo families can be seen as fixed points at $q_3 = \pm \sim 0.5$.

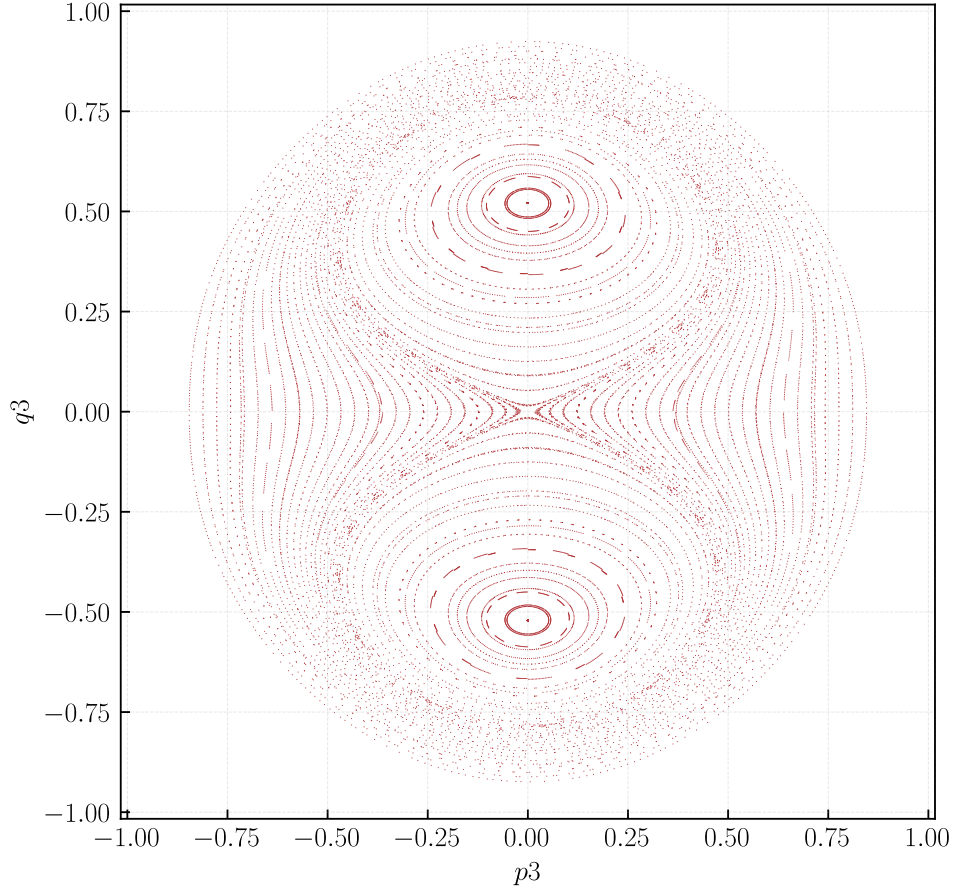


Figure 3.3: Poincaré map of the $p_2 = 0$ section of the Earth-Moon L_1 libration point at energy $C = 1.0$.

Casting these points back to the synodic frame gives us initial conditions for a halo orbit. As the transform from synodic to centre manifold coordinates is symplectic, the transform that takes coordinates from the centre manifold to the synodic frame is the inverse of the forward transform, and is shown in [Figure 3.4](#). The one caveat being that the scaling and translation from the real local frame to the synodic frame is not symplectic, and must therefore be inverted separately.

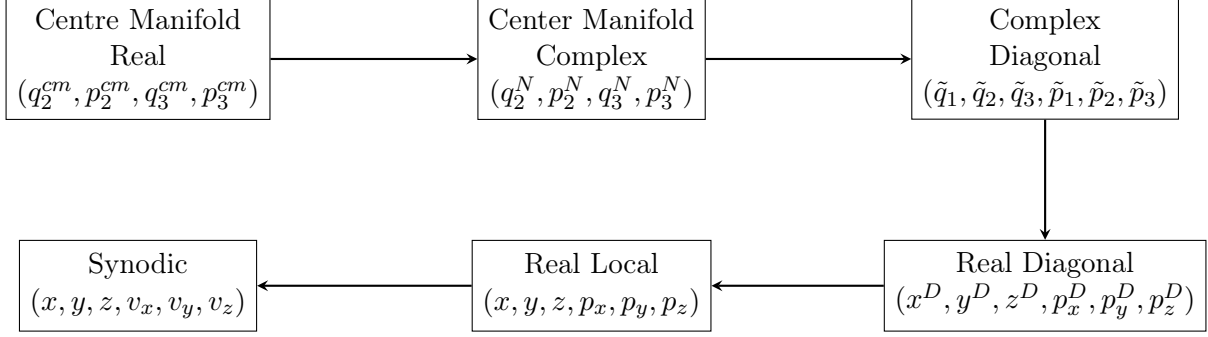


Figure 3.4: Coordinate transformation process from centre manifold frame to synodic frame.

3.1.2 Differential Correction of Seed Orbits

The resulting initial condition is close to, but not exactly on, a periodic orbit. A differential correction scheme is required to converge to a true periodic solution. First, the linearised dynamics are introduced. The evolution of a small deviation, $\delta\mathbf{x}$, from a reference trajectory $\mathbf{x}(t)$ is described by the linear variational equations:

$$\delta\dot{\mathbf{x}} = Df(\mathbf{x}(t))\delta\mathbf{x}, \quad (3.9)$$

where $Df(\mathbf{x}(t))$ is the Jacobian of the vector field evaluated along the reference trajectory. The solution to this is given by the [STM](#), $\Phi(t, t_0)$, which maps an initial deviation $\delta\mathbf{x}(t_0)$ to a deviation at a later time t :

$$\delta\mathbf{x}(t) = \Phi(t, t_0)\delta\mathbf{x}(t_0). \quad (3.10)$$

The [STM](#) itself is found by integrating:

$$\dot{\Phi}(t, t_0) = Df(\mathbf{x}(t))\Phi(t, t_0), \quad (3.11)$$

with the initial condition $\Phi(t_0, t_0) = \mathbf{I}$. A single shooting corrector is applied to the resulting orbit, looking for a constraint:

$$|\mathbf{x}(T) - \mathbf{x}(0)| < d, \quad (3.12)$$

where $\mathbf{x}(t)$ is the state of the orbit at time t , T is the period, and d is the tolerance. The symmetry of halo orbits about the xz -plane, which they intersect perpendicularly, is utilised. The corrector's goal is to find an initial state $\mathbf{x}(0) = (x_0, 0, z_0, 0, \dot{y}_0, 0)$ that results in a state at the half-period plane crossing ($t = T/2$) where the perpendicularity conditions $\dot{x}(T/2) = 0$ and $\dot{z}(T/2) = 0$ are met.

The differential corrector iteratively adjusts the controllable initial variables $(x_0, z_0, \dot{y}_0)$ to drive the residual target variables $(\dot{x}(T/2), \dot{z}(T/2))$ to zero. The relationship between a change in the initial state, $\delta\mathbf{x}(0)$, and the resulting change in the final state, $\delta\mathbf{x}(T/2)$, is given precisely by the STM. To simplify calculations, the initial z_0 is fixed, and only x_0 and $\dot{y}_0$ are varied to correct the two target variables. This process is repeated until Equation 3.12 is satisfied for some d . Figure 3.5 shows the seeded northern halo ($q_3 = +0.5$ in Figure 3.3) of the Earth-Moon L_1 point for $C = 1.0$ before and after differential correction.

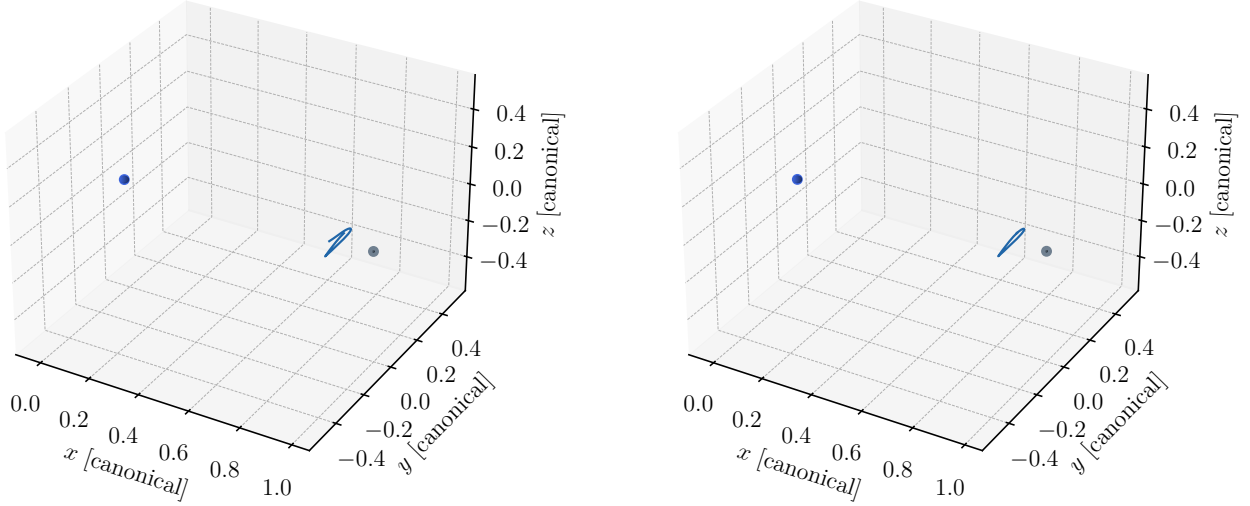


Figure 3.5: Seeded Earth-Moon L_1 northern halo for $C = 1.0$ before and after correction.

While Figure 3.6 shows the seeded southern halo ($q_3 = -0.5$) of the Earth-Moon L_1 point for $C = 1.0$ before and after differential correction.

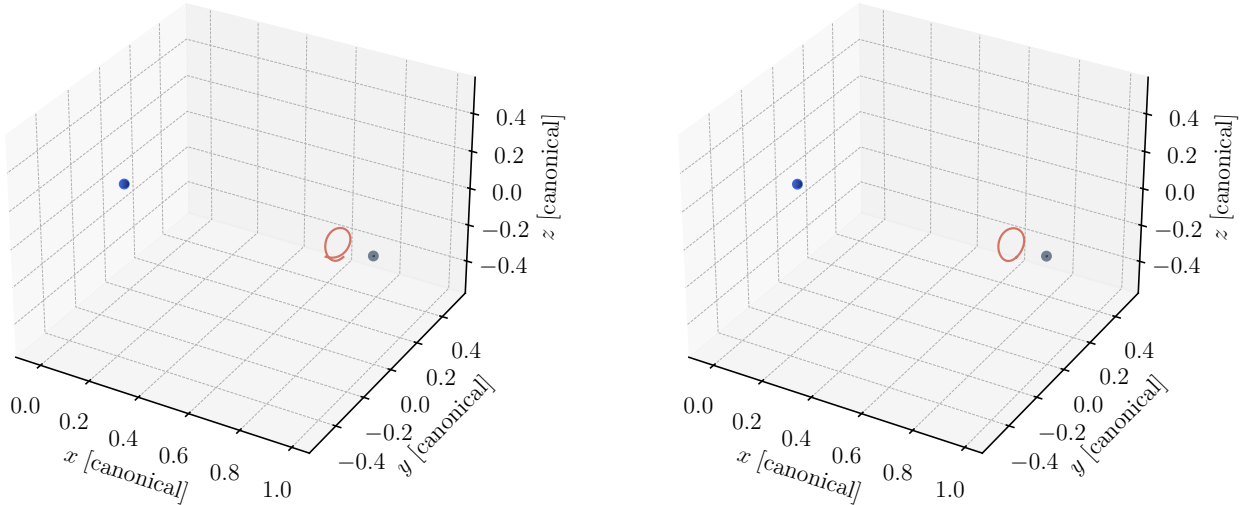


Figure 3.6: Seeded Earth-Moon L_1 southern halo for $C = 1.0$ before and after correction.

3.1.3 Generating Families of Halo Orbits

Once a seed halo orbit has been successfully computed and corrected, it serves as the starting point for generating the entire family of orbits to which it belongs. This is achieved through a natural parameter continuation. The fundamental principle is to incrementally vary a system parameter, in this case, the Jacobi constant C , and track how the initial conditions of the periodic orbit evolve.

The algorithm is an iterative predictor-corrector scheme. Given a converged solution $(\mathbf{x}_n, C_n)$, consisting of the initial state vector and its corresponding Jacobi constant, the goal is to find the next solution $(\mathbf{x}_{n+1}, C_{n+1})$ in the family. First, a small step is taken in the continuation parameter:

$$C_{n+1} = C_n + \Delta C, \quad (3.13)$$

where ΔC is a prescribed step size. A prediction for the new initial state, $\mathbf{x}_{n+1}^{(p)}$, is then made. The simplest approach, known as a zero-order or trivial predictor, is to use the initial state of the previous orbit:

$$\mathbf{x}_{n+1}^{(p)} = \mathbf{x}_n. \quad (3.14)$$

This initial guess assumes that the orbit's initial conditions do not change significantly for a small change in energy. While more sophisticated predictors exist, this method is simple and effective for sufficiently small step sizes ΔC . The predicted state $\mathbf{x}_{n+1}^{(p)}$ is not a true periodic orbit for the new energy level C_{n+1} . Integrating this state will result in a trajectory that fails to meet the periodicity conditions outlined previously.

To refine this guess, the same differential correction algorithm described in the previous section is employed. The corrector iteratively adjusts the control variables to drive the residual variables at the half-period plane crossing to zero, but this time for the system with energy C_{n+1} . The process is repeated until the periodicity constraints are met to within the specified tolerance d , yielding the converged state $\mathbf{x}_{n+1}$. The corrected state $\mathbf{x}_{n+1}$ and its corresponding energy C_{n+1} now form a new, validated member of the halo orbit family. This new solution then serves as the input for the next iteration of the predictor-corrector loop. By repeating this process, the entire family of orbits is traced out, step by step, revealing how the orbit's geometry and stability properties evolve as a function of the Jacobi constant.

Applying this method to the previously identified northern and southern halo orbit seeds yields their respective families. The dataset comprises 626 orbits with Jacobi constants ranging from 3.03826 to 3.17434.

3.1.4 Invariant Manifolds of Halo Orbits

The final step is to generate the associated manifolds for the computed families of halo orbits. Each periodic orbit, Λ , in the family acts as a [NHIM](#). The dynamics in the neighbourhood of

this orbit, and therefore the structure of its manifolds, are determined by the linearisation of the flow around the orbit.

The stability properties are encapsulated in the Monodromy matrix, $M = \Phi(T, 0)$, which is the [STM](#) integrated over one full period, T . The eigenvalues of M dictate the linear stability. For the unstable halo orbits of interest, the eigensystem of M has a characteristic structure. Specifically, there exists a real eigenvalue $\lambda_u > 1$, which corresponds to an unstable, expanding direction, and a real eigenvalue $\lambda_s = 1/\lambda_u < 1$, which corresponds to a stable, contracting direction. In addition to these, there are four other eigenvalues with unit modulus, corresponding to the centre directions tangent to the orbit and its energy surface.

The eigenvectors $\mathbf{v}_u$ and $\mathbf{v}_s$ associated with the real eigenvalues λ_u and λ_s are of particular importance. At any point $\mathbf{x}$ on the periodic orbit Λ , these vectors span the one-dimensional unstable and stable eigenspaces of the linearised map, denoted $E^u(\mathbf{x})$ and $E^s(\mathbf{x})$ respectively. According to the theorems of Fenichel, these linear eigenspaces are tangent to the true, smooth, non-linear invariant manifolds of the orbit, $W^u(\Lambda)$ and $W^s(\Lambda)$.

Formally, these manifolds are the sets of all points in phase space that are asymptotic to the orbit Λ in backward or forward time:

$$W^u(\Lambda) = \{\mathbf{y} \in \mathbb{R}^6 : \lim_{t \rightarrow -\infty} \text{dist}(\phi_t(\mathbf{y}), \Lambda) = 0\}, \quad (3.15)$$

$$W^s(\Lambda) = \{\mathbf{y} \in \mathbb{R}^6 : \lim_{t \rightarrow +\infty} \text{dist}(\phi_t(\mathbf{y}), \Lambda) = 0\}, \quad (3.16)$$

where $\phi_t(\mathbf{y})$ denotes the trajectory starting at $\mathbf{y}$. To obtain a discrete approximation of the manifold surfaces, a set of N points $\{\mathbf{x}_i\}_{i=1}^N$ is sampled along the halo orbit Λ , taken from one of the generated families. At each point $\mathbf{x}_i$, the local stable and unstable eigenvectors, $\mathbf{v}_{s,i}$ and $\mathbf{v}_{u,i}$, of the Monodromy matrix are computed.

Using these points, initial conditions for integration are obtained by applying a small displacement ϵ along the corresponding eigenvector directions. Crucially, for each eigenvector, there are two opposing directions for this perturbation, corresponding to $+\epsilon$ and $-\epsilon$. These two choices generate the two distinct sheets or branches of each manifold. For an orbit around L_1 or L_2 , these branches correspond to physically distinct pathways, and are referred to as positive and negative branches:

$$\mathbf{y}_{u,i}^\pm = \mathbf{x}_i \pm \epsilon \mathbf{v}_{u,i} \quad (\text{for the two unstable branches}), \quad (3.17)$$

$$\mathbf{y}_{s,i}^\pm = \mathbf{x}_i \pm \epsilon \mathbf{v}_{s,i} \quad (\text{for the two stable branches}). \quad (3.18)$$

The parameter ϵ must be chosen to be small enough to remain within the linear regime but large enough to allow the trajectory to diverge from the orbit within a reasonable computation time.

To trace the manifold fibres, these perturbed initial conditions are then integrated. The trajectories originating from each $\mathbf{y}_{u,i}^\pm$ pair are integrated forward in time ($t > 0$) to generate the two branches of the unstable manifold $W^u(\Lambda)$, while those from each $\mathbf{y}_{s,i}^\pm$ pair are integrated backward in time ($t < 0$) to generate the two branches of the stable manifold $W^s(\Lambda)$. The union of all these propagated trajectories forms a discrete representation of the complete, two-sheeted manifold surfaces.

Applying this procedure to each of the 626 orbits in the generated families yields a comprehensive dataset of halo-orbit invariant manifolds, including both positive and negative branches. Each manifold in the dataset is generated with a step of 0.02 (i.e. 50 trajectories per manifold) and integrated for 5 dimensionless time units with a step size of 0.01 dimensionless time units.

3.2 Network Architecture and Training

This section presents the neural network architecture and training methodology developed for learning stable manifold trajectories in the [CR3BP](#). The approach combines vector quantisation with physics-informed learning to create a generative model capable of producing dynamically consistent trajectories. The methodology discusses the training data format, network architecture design, loss formulations, and training hyperparameters.

3.2.1 Training Data

The training data consists of a four-dimensional tensor $\mathbf{x} \in \mathbb{R}^{B \times 6 \times N \times T}$, where B represents the number of manifolds, N denotes the number of trajectories per manifold, and T indicates the integration steps per trajectory. For this initial study, the focus is exclusively on the positive and negative branches of stable manifolds associated with Earth–Moon L_1 northern halo orbits.

Halos with Jacobi constants ranging from 3.03826 to 3.17434 are selected, resulting in a final tensor size of $1252 \times 6 \times 50 \times 503$. All states and the Jacobi constant are normalised using standard min–max scaling to the interval $[-1, 1]$. While the Jacobi constant does not form part of the training tensor itself, it serves as a conditioning parameter for the learned prior and is masked during the initial training phases. The symbol x is used for unscaled states (canonical units), x^{sc} for scaled states in $[-1, 1]$, and $\hat{x}^{\text{sc}}$ for reconstructed scaled states. Unscaling is performed with the inversion:

$$x_c = s_c(x_c^{\text{sc}} + 1) + m_c, \quad s_c = \frac{r_c}{2}, \quad (3.19)$$

where m_c is the channel minimum and r_c the channel range; $s = \{s_c\}_c$ is the vector of half-ranges. This configuration provides several advantages. First, successfully learning both branches of the stable manifold directly implies that the model can also learn unstable manifolds due to the inherent symmetry of the system. Second, L_1 halos offer a rich dynamical

environment where the gravitational effects of both primaries are pronounced, providing the model with diverse training examples.

3.2.2 Network Architecture

The model implements a [VQ-VAE](#) using an exponential moving average vector quantizer. The architecture draws inspiration from the methods developed by *Razavi, van den Oord*, and *Vinyals* [46], adapted specifically for this application. The complete system comprises four main components: an encoder network, a vector quantisation bottleneck, a decoder network, and a prior over code indices conditioned on Jacobi constant and branch.

Encoder Design

The encoder transforms input trajectories into a discrete latent representation through a two-stage process. The first stage employs a two-layer [Gated Recurrent Unit \(GRU\)](#) [47] with hidden layer dimensions of 256 and dropout regularisation set to 0.2. This recurrent component captures the temporal dependencies inherent in the trajectories.

The second stage consists of a three-layer [Multilayer Perceptron \(MLP\)](#) that progressively reduces dimensionality from 256 to 128, then to 64, and finally to the target latent size of 8. ReLU activation functions provide nonlinearity between layers. To prevent immediate codebook collapse in low-variance regions, small Gaussian noise $\varepsilon \sim \mathcal{N}(0, 10^{-3}I)$ is injected into the pre-quantisation latent representations during training.

Vector Quantisation Component

The vector quantiser maintains a codebook of 64 embedding vectors, updated using exponential moving averages with a decay parameter of 0.95. To ensure robust codebook utilisation throughout training, a dead code detection mechanism is implemented. Codes with usage below 10^{-6} relative to the average are identified as inactive and reinitialised with random batch vectors every 20 epochs. Perplexity:

$$\text{Perp} = \exp\left(-\sum_k p_k \log p_k\right), \quad (3.20)$$

where p_k is the empirical code probability, is reported as an effective code count computed over the evaluation set.

Conditioned Prior Network

A compact [MLP](#) maps the 2-D conditioning $[C, \text{branch}]$ to logits $\ell \in \mathbb{R}^K$. The prior network follows a simple architecture with layers of dimensions 2, increasing to 32, and finally to 64, producing logits defined as:

$$\ell = \text{MLP}_{\text{prior}}([C, \text{branch}]), \quad p(k | C, \text{branch}) = \text{softmax}(\ell)_k. \quad (3.21)$$

During training, the cross-entropy between these prior logits and the indices selected by the encoder is computed, adding this term to the total loss with a weight of 0.05. This design enables controllable trajectory generation by sampling from the prior conditioned on desired physical parameters. Importantly, the energy and branch information is not directly provided to the decoder; all control is exercised through this learned probabilistic prior.

Decoder Architecture

The decoder reconstructs trajectories from quantised latent vectors using a combination of learned temporal embeddings and recurrent processing. Each quantised latent vector is concatenated with a learned per-time-step embedding of dimension 16, creating a 24-dimensional input vector at each temporal step.

This concatenated representation passes through an [MLP](#) with layer dimensions 24, increased to 64, and then 256, generating appropriate inputs for the subsequent recurrent layers. The core of the decoder consists of a two-layer [GRU](#) with hidden dimensions of 256 and dropout regularisation of 0.1. A final linear projection maps the [GRU](#) outputs back to the original six-dimensional state space, with a tanh activation function ensuring outputs remain within the normalised range $[-1, 1]$.

3.2.3 Physics-Informed Loss

Deviation from [CR3BP](#) dynamics is penalised using a midpoint residual. All dynamics are evaluated in canonical units. Let $x_t \in \mathbb{R}^6$ denote the state (x, y, z, v_x, v_y, v_z) at time index t , with time step $\Delta t = 0.01000507$ and Earth–Moon mass parameter $\mu = 0.012150585609624$. Define midpoint states and the discrete derivative:

$$x_{t+\frac{1}{2}} = \frac{1}{2}(x_{t+1} + x_t), \quad D_t x_t = \frac{x_{t+1} - x_t}{\Delta t}, \quad (3.22)$$

the residual is

$$r_t = D_t x_t - f\left(x_{t+\frac{1}{2}}\right), \quad (3.23)$$

where f are the equations of motion in [Equation 2.1](#). To reduce sensitivity near primaries and at high speeds adaptive weights are applied:

$$w_t = \sigma(100(r_1(x_t) - d_0)) \sigma(100(r_2(x_t) - d_0)) \frac{1}{1 + 0.1 \|v_t\|_2}, \quad d_0 = 0.01, \quad (3.24)$$

with $\sigma(u) = \frac{1}{1+e^{-u}}$, $r_i(x_t)$ the distance to primary $i \in \{1, 2\}$ from the position components of x_t , and v_t the velocity vector. Velocity residuals are down-weighted by 0.1 via the diagonal matrix:

$$W_{\text{comp}} = \text{diag}(1, 1, 1, 0.1, 0.1, 0.1). \quad (3.25)$$

Before aggregation, residual components are normalised by the channel half-ranges s_c . For T time samples, the loss is:

$$\mathcal{L}_{\text{phys}} = \frac{1}{T} \sum_{t=1}^{T-1} \min\left(\|W_{\text{comp}} \text{diag}(s)^{-1} r_t\|_2^2 w_t, 100\right). \quad (3.26)$$

Gradients through the normalisation and weights are stopped to stabilise optimisation.

3.2.4 Training

Given scaled inputs $x^{\text{sc}} \in [-1, 1]^{C \times N \times T}$ and reconstructions $\hat{x}^{\text{sc}}$ (same shape), the reconstruction loss is

$$\mathcal{L}_{\text{rec}} = \frac{1}{BCNT} \sum_{b,c,n,t} (\hat{x}^{\text{sc}} - x^{\text{sc}})_{b,c,n,t}^2, \quad C = 6. \quad (3.27)$$

Where B , C , N , and T are the dimensions of the dataset as described previously. For interpretability, channel-wise root mean squared error in physical units is obtained by unscaling with each channel's half-range s_c . The total loss is

$$\mathcal{L} = \mathcal{L}_{\text{rec}} + \beta(e) \mathcal{L}_{\text{vq}} + \lambda_{\text{prior}} \mathcal{L}_{\text{prior}} + \lambda_{\text{phys}}(e) \mathcal{L}_{\text{phys}} \quad (3.28)$$

With $\lambda_{\text{prior}} = 0.05$ and epoch index e . Schedules are:

$$\beta(e) = \begin{cases} 0.1 \frac{e}{20}, & e \leq 20, \\ 0.1, & e > 20, \end{cases} \quad \lambda_{\text{phys}}(e) = \begin{cases} 0, & e \leq 200, \\ 0.05 \frac{e-200}{200}, & 200 < e \leq 400, \\ 0.05, & e > 400. \end{cases} \quad (3.29)$$

Training begins without this component for the first 200 epochs, allowing the model to establish basic reconstruction capabilities. The physics loss is then gradually introduced with its weight increasing linearly from 0 to 0.05 over epochs 201 – 400, preventing training instability while progressively enforcing dynamical consistency. Energy masking is applied for $e \leq 100$, setting the conditioning C to zero with probability $p(e) = 1 - \frac{e}{100}$ independently per sample.

The model is trained for 600 epochs with AdamW (weight decay 0.01), batch size 32, and cosine-annealed learning rate from 10^{-3} to 10^{-5} .

Chapter 4

Results

This chapter presents an evaluation of the trained [VQ-VAE](#) model across four key dimensions: reconstruction fidelity of existing manifolds, generation quality of new manifolds, computational efficiency compared to traditional methods, and interpretability of the learned latent representations. The evaluation covers all 1250 manifolds spanning both positive and negative branches across the full energy range of the dataset.

Reconstruction errors are computed as [Mean Squared Error \(MSE\)](#) in the scaled space (see [Section 3.2](#)); states are unscaled only for visualisation. Evaluation covers all 1250 manifolds across both branches, and per-manifold error, aggregated by branch and overall, are recorded. Statistical measures are reported as follows: means with 95% confidence intervals, medians, and 95th percentiles where appropriate. Confidence intervals are computed using standard error ($SE = \sigma/\sqrt{n}$) with $\pm 1.96 \times SE$ bounds.

After 600 training epochs, the total loss converges to 0.107 and is dominated by the prior term. At convergence, the prior loss is 0.1065, while the physics residual, vector-quantisation commitment, and reconstruction terms are substantially smaller at 3.40×10^{-4} , 1.33×10^{-4} , and 4.87×10^{-5} , respectively. The magnitude of the prior term reflects residual uncertainty in predicting code usage from energy and branch alone, consistent with the complexity of the initial-condition to manifold mapping.

Reconstruction accuracy in canonical units is summarised by channel-wise [Root Mean Squared Error \(RMSE\)](#) in [Table 4.1](#), corresponding to position errors of approximately [1169, 665, 561] km for (x, y, z) in the Earth-Moon system.

$x/10^{-3}$	$y/10^{-3}$	$z/10^{-3}$	$v_x/10^{-2}$	$v_y/10^{-2}$	$v_z/10^{-2}$
3.04	1.73	1.46	3.91	2.57	2.52

Table 4.1: Channel-wise reconstruction root mean squared error in scaled space.

Two additional metrics are used to assess generated manifolds: root proximity and Jacobi

error. Let $\mathbf{r}(h, t) \in \mathbb{R}^3$ denote the unscaled generated position for trajectory index h and time t , and let $\mathbf{r}_{\text{halo}}$ be the position of the nearest-in-energy reference halo. Define:

$$d_{\text{root}}^{(t=\tau)} = \min_h \|\mathbf{r}(h, \tau) - \mathbf{r}_{\text{halo}}\|_2, \quad (4.1)$$

which measures the proximity of the of the generated manifold to the corresponding energy halo location at time τ . The second metric is the Jacobi error. For unscaled states $\mathbf{x}(h, t)$ and target Jacobi C^* (for the corresponding halo), the Jacobi constant:

$$C(h, t) = 2\Omega(x, y, z) - \|\mathbf{v}\|_2^2, \quad (4.2)$$

is evaluated in canonical units. Mean and median deviations of the maximum and mean Jacobi errors:

$$\Delta C_{\text{max}} = \max_{h,t} |C(h, t) - C^*|, \quad (4.3)$$

$$\Delta C_{\text{mean}} = \frac{1}{NT} \sum_{h,t} |C(h, t) - C^*|, \quad (4.4)$$

$$(4.5)$$

are reported, as well as the relative errors $\Delta C/|C^*|$.

4.1 Reconstruction of Manifolds

To evaluate reconstruction fidelity, manifold tensors are rescaled to $[-1, 1]$, energy and branch metadata is attached, and the tensors are passed through the trained model to obtain reconstructions in the same scaled space. For visualisation only, original and reconstructed manifolds are unscaled to canonical units and plotted side-by-side. Across all 1250 reconstructions, the mean scaled MSE is 4.87×10^{-5} . Branch-wise, the positive branch exhibits larger errors than the negative branch. Table 4.2 reports mean, median, and 95th percentile.

Branch	MSE Mean/ 10^{-5}	MSE Median/ 10^{-5}	MSE p95/ 10^{-4}
Positive	7.23	6.71	1.35
Negative	2.51	2.23	0.565
Total	4.87	4.47	0.959

Table 4.2: Mean squared error in reconstruction in scaled space per branch.

The negative branch exhibits consistently lower reconstruction error than the positive across mean, median, and upper-tail metrics. The mean error on the negative branch is approximately

65% lower, with the 95% confidence intervals of the mean ranging from 6.97×10^{-5} to 7.49×10^{-5} for the positive branch and from 2.37×10^{-5} to 2.65×10^{-5} for the negative branch. This behaviour extends across the distribution: the median gap mirrors the mean gap, and the upper tail (p95) is substantially lighter, indicating fewer large deviations; full point estimates and 95% confidence intervals are reported in [Table 4.2](#).

Error magnitudes are smallest at mid-range C and increase toward the extremes. The positive branch shows a higher baseline and heavier tails, consistent with closer passages to the secondary manifold and stronger curvature, whereas the negative branch remains uniformly low with noticeably smaller dispersion.

The left panel of [Figure 4.1](#) shows the relationship between reconstruction error and Jacobi constant; the smallest errors are around mid-range C and larger errors toward the extremes. The positive branch exhibits a higher plateau and wider interquartile band, consistent with close passages to the secondary and stronger curvature; the negative branch remains uniformly low with a narrower band.

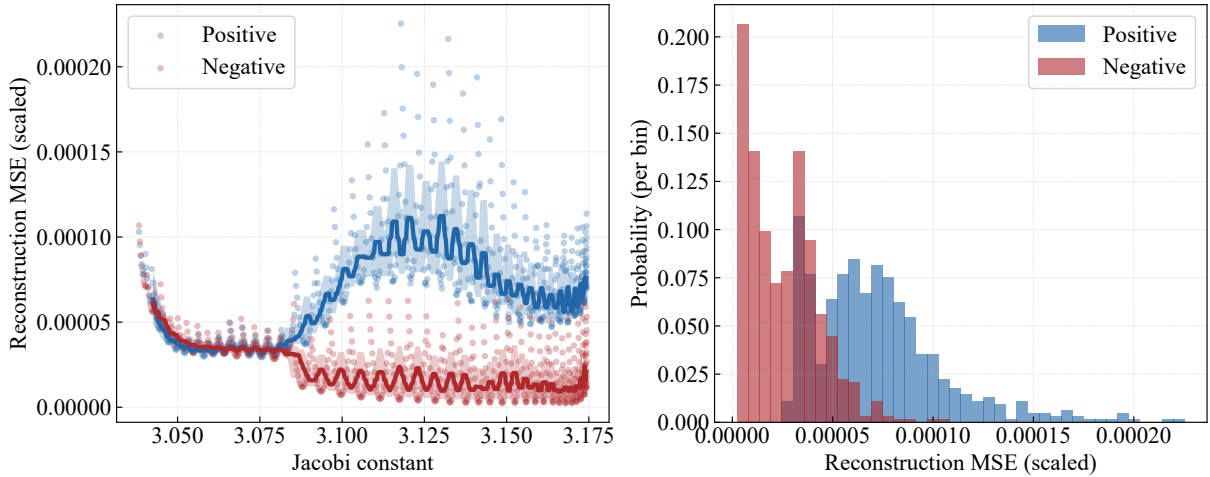
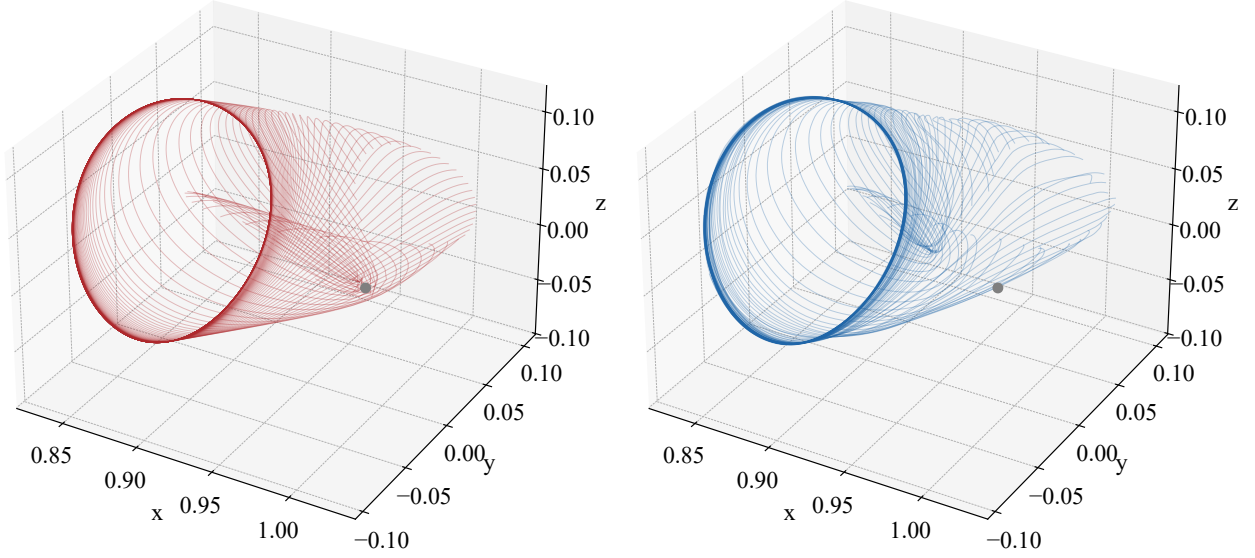


Figure 4.1: Energy-conditioned reconstruction errors. Positive branch shows consistently higher median error, particularly at high and low C values.

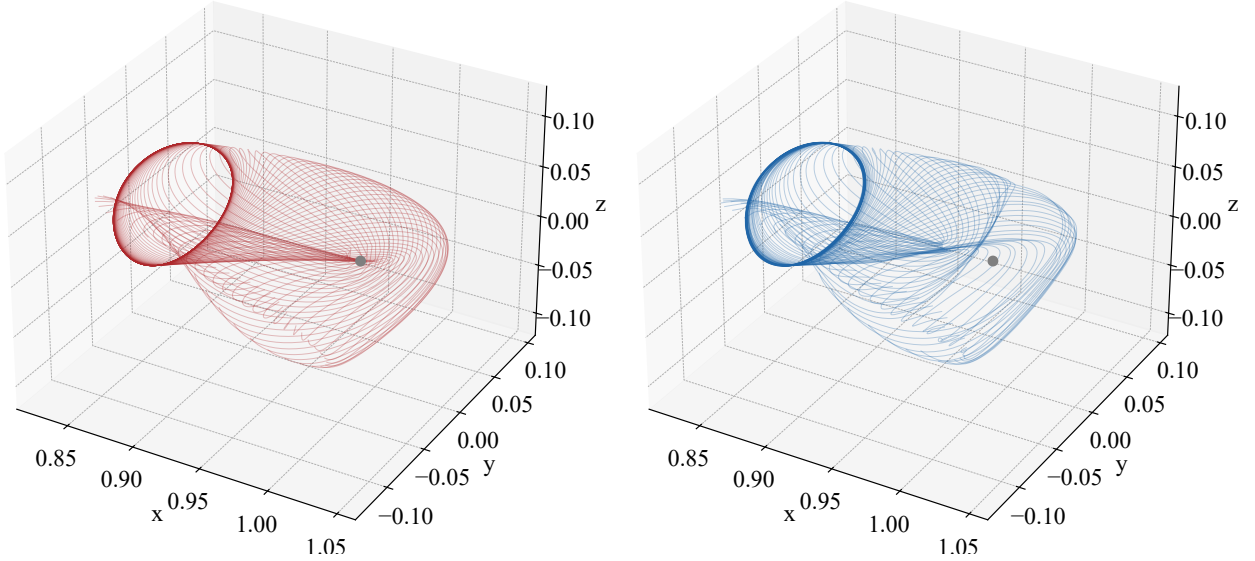
Figures [4.2a](#) and [4.2b](#) compare the numerically integrated (left) and reconstructed (right) positive branch of a manifold with Jacobi constant 3.1168. The reconstructed manifold closely follows the reference, preserving both the global shape and local curvature of the orbit. Errors are small near the initial segment but gradually accumulate with time, becoming most visible where the trajectory curvature changes more sharply. This drift is consistent with the compounding effect of small local reconstruction errors. Overall, the topology, phase, and scale of the manifold are faithfully reproduced, indicating that the learnt latent representation retains the key dynamical features in this regime.

[Figure 4.2c](#) and [Figure 4.2d](#) compare manifolds at $C = 3.1608$. As in the previous case, the reconstructed manifold preserves the overall geometry and orientation of the manifold, maintaining consistency in both scale and topology. However, large deviations occur in the

vicinity of the secondary, where the reference manifold exhibits strong curvature and steep velocity gradients. In this high-sensitivity region, small reconstruction inaccuracies amplify, producing a noticeable widening of the reconstructed path. Additionally, the reconstructed manifold appears sparser than the numerically integrated one, particularly around the secondary.



(a) Numerically obtained manifold with Jacobi constant 3.1168. (b) Reconstructed manifold with Jacobi constant 3.1168.

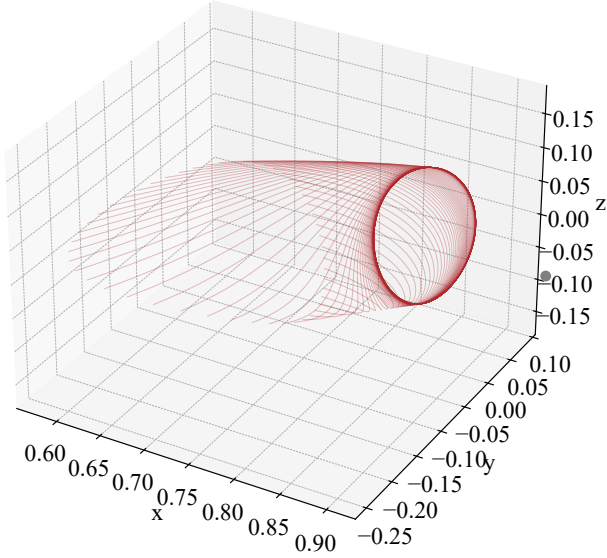


(c) Numerically obtained manifold with Jacobi constant 3.1608. (d) Reconstructed manifold with Jacobi constant 3.1608.

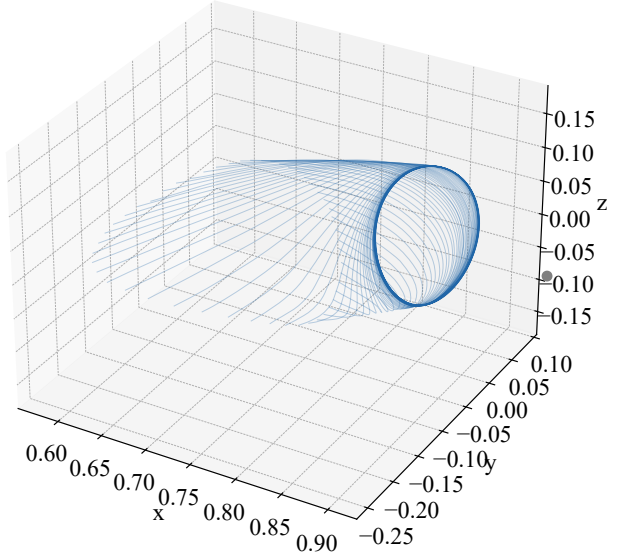
Figure 4.2: Numerically obtained and reconstructed positive branches of halo manifolds at $C = 3.1168$ and $C = 3.1608$.

Figure 4.3a and Figure 4.3b compare manifolds at a Jacobi constant of 3.1253. The reconstruction closely matches the reference trajectory, with minimal deviations throughout the

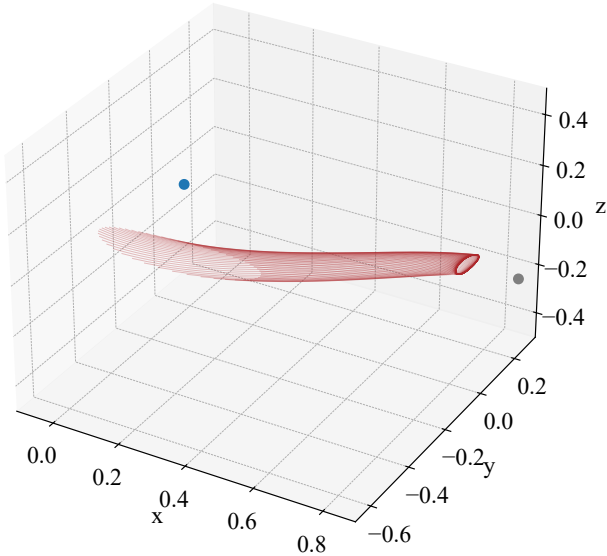
integration period. The global geometry and loop structure are preserved, and local curvature changes are reproduced with high fidelity. The reduced reconstruction error in this case is due to the absence of close approaches to the secondary, which eliminates singularities and reduces local velocity gradients. As a result, small reconstruction inaccuracies do not compound as strongly, leading to a smoother and more accurate reproduction.



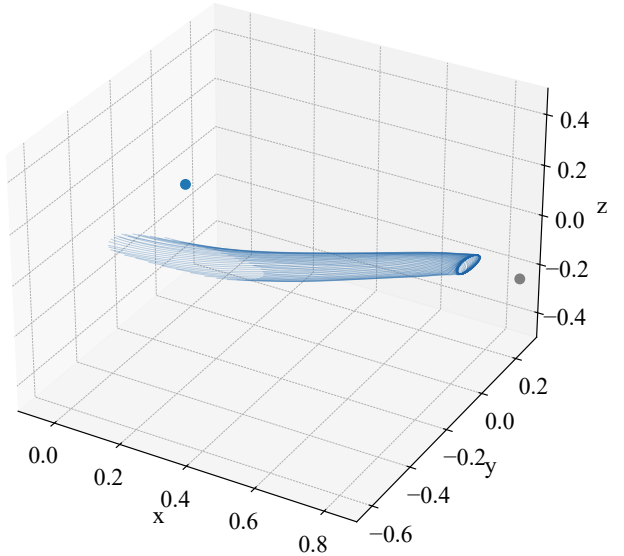
(a) Numerically obtained manifold with Jacobi constant 3.1253.



(b) Reconstructed manifold with Jacobi constant 3.1253.



(c) Numerically obtained manifold with Jacobi constant 3.1742.



(d) Reconstructed manifold with Jacobi constant 3.1742.

Figure 4.3: Numerically obtained and reconstructed negative branches of halo manifolds at $C = 3.1253$ and $C = 3.1742$.

Figure 4.3c and Figure 4.3d show the manifolds at $C = 3.1742$. Even in the presence of the primary, the reconstruction error remains low, and the overall manifold topology is

faithfully preserved. Across the tested Jacobi constants, the model consistently maintains global structure in the negative branch. However, performance degrades at the extrema of C : at high values, manifolds become more elongated and pass closer to the secondary, increasing reconstruction difficulty; at low values, the long integration time in the training data causes manifolds to resemble closed orbits, reducing branch-specific geometric differences and making accurate reproduction more challenging.

4.2 Generation of Manifolds

Manifolds are generated by sweeping evenly spaced energies across the dataset range for both positive and negative branches. Generation uses deterministic prior-argmax selection, ensuring a unique manifold for each branch–energy pair (Section 3.2).

The Jacobi error exhibits a clear dependence on the value of the constant itself, as shown in the right panel of Figure 4.4. At high Jacobi constant values of around 3.17, the positive manifold branch shows larger errors with peaks reaching approximately 1.7. In contrast, the negative branch performs better at these energy levels, with peaks near 1.0. As the Jacobi constant decreases, both branches show improvement. The negative branch achieves maximum Jacobi constant errors below 0.05 by $C \approx 3.15$, while the positive branch reaches this level of accuracy by $C \approx 3.07$. Table 4.3 summarises generation quality across the energy sweep.

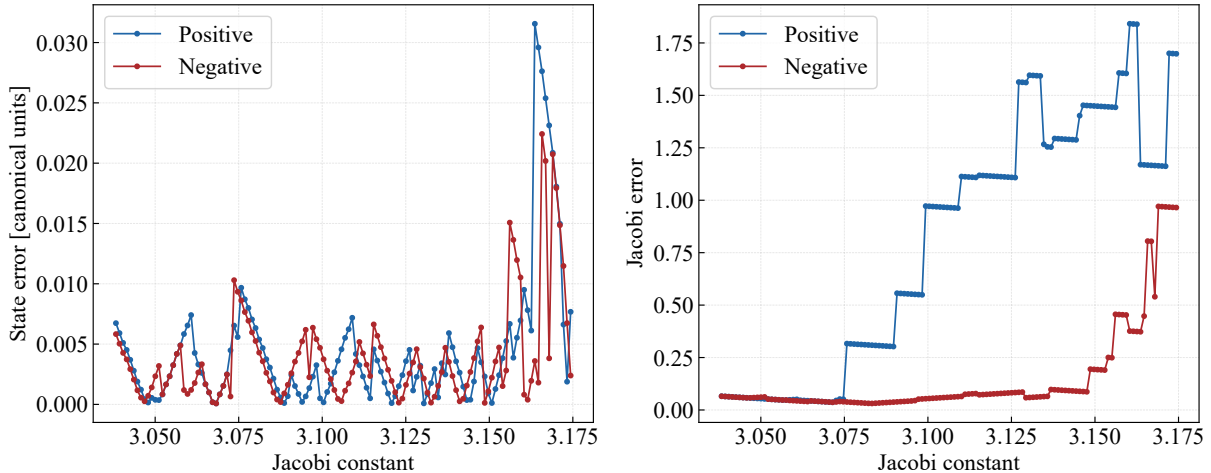


Figure 4.4: Mean state error and maximum trajectory Jacobi error vs Jacobi constant.

Across energies, the initial root distance $d_{\text{root}}^{(0)}$ remains small, with tighter initial anchoring on the negative branch; see Table 4.3 for medians and 95th percentiles. The median $d_{\text{root}}^{(0)}$ corresponds to approximately 1100 km (positive) and 1015 km (negative), indicating consistent anchoring to the halo location.

Mean Jacobi errors are small on average (median $\Delta C_{\text{mean}} \sim 10^{-2}$ or below), with 95% confidence intervals of $[8.97 \times 10^{-3}, 1.02 \times 10^{-2}]$ for the positive branch and $[4.81 \times 10^{-3}, 5.82 \times$

$10^{-3}]$ for the negative branch. However, $\Delta C_{\max}$ is asymmetric: the negative branch attains low means and maxima, whereas the positive branch shows large outliers at high C , as shown both in [Figure 4.4](#) and [Table 4.3](#). Relative errors corroborate this trend: the negative branch attains a median $|\Delta C|_{\max}/|C^*| = 2.0\%$ (95th 25.4%), whereas the positive branch shows 31.3% (95th 50.9%).

Branch	$d_{\text{root}}^{(0)}$	Median/ 10^{-2}	$d_{\text{root}}^{(0)}$	p95/ 10^{-1}	ΔC_{mean}	Median/ 10^{-3}	ΔC_{mean}	p95/ 10^{-2}	ΔC_{max}	Median/ 10^{-2}	ΔC_{max}	p95/1
Positive	4.74		8.39		10.3		1.47		9.69		1.61	
Negative	7.62		50.7		4.73		1.08		6.19		0.805	
Total	4.96		41.7		6.67		1.40		8.73		1.57	

Table 4.3: Root distance, and median average and maximum Jacobi error for each branch in canonical units.

The largest ΔC violations occur where trajectories pass closest to the secondary. This relationship is explicit in [Figure 4.5](#), which plots $|\Delta C|_{\max}$ versus minimum distance to either primary.

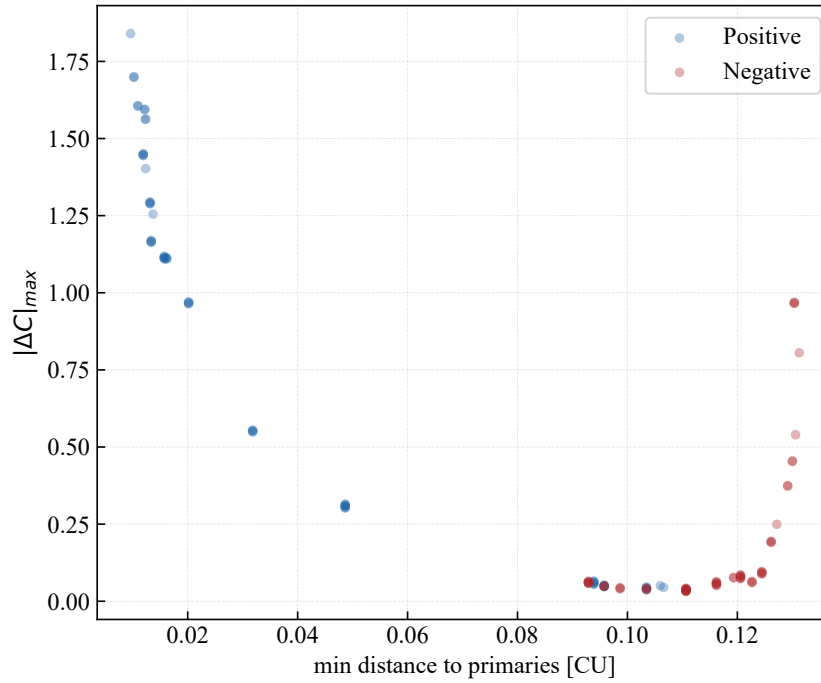
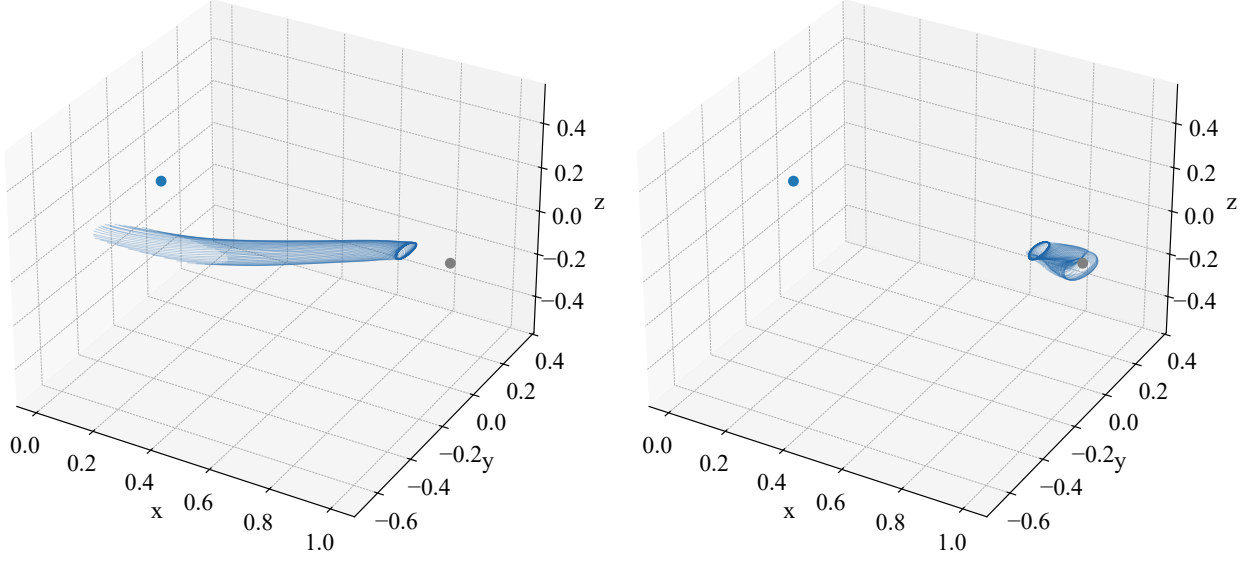


Figure 4.5: Maximum Jacobi error $|\Delta C|_{\max}$ vs minimum distance to either primary in canonical units.

[Figure 4.6a](#) and [Figure 4.6b](#) show generated negative and positive branches at $C = 3.174344$. Both branches anchor to the same halo location, with initial root distances of 2.4×10^{-3} and 7.7×10^{-3} canonical units for the negative and positive branches, respectively. The negative branch forms an elongated, gently curved sheet that departs outward from the halo, whereas the positive branch remains compact and wraps tightly around the secondary. The

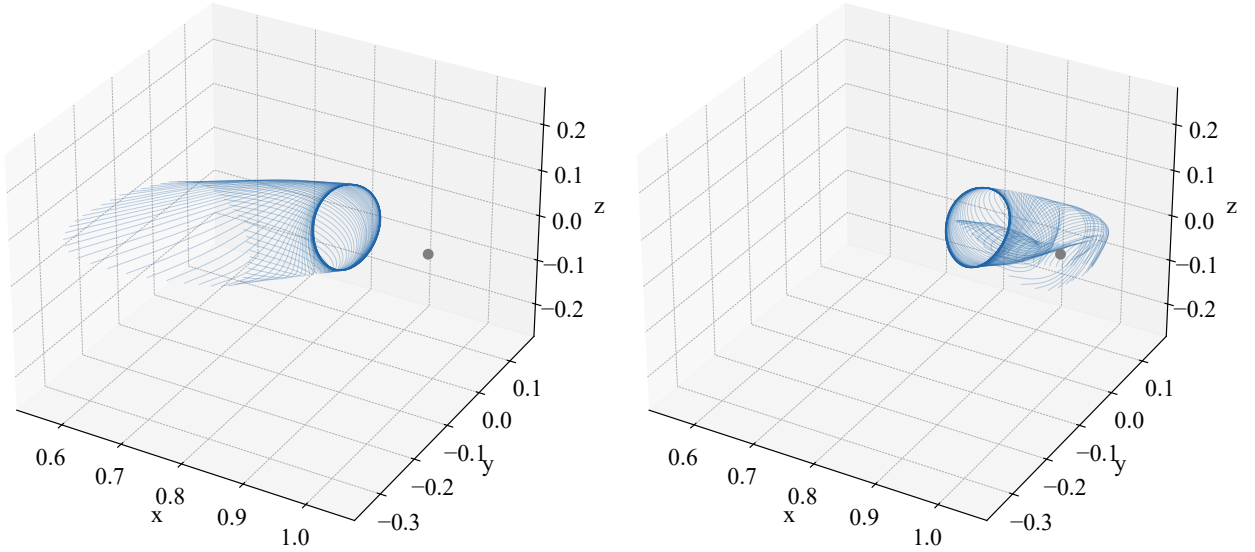
pronounced asymmetry at this high C is characteristic of closer approaches to the secondary and stronger local curvature, consistent with the larger energy-surface deviations observed for the positive branch in Figure 4.4.



(a) Generated negative branch of a halo manifold with Jacobi constant 3.174344. (b) Generated positive branch of a halo manifold with Jacobi constant 3.174344.

Figure 4.6: Generated negative and positive halo manifolds with Jacobi constant 3.174344.

Figure 4.7a and Figure 4.7b show generated negative and positive branches at $C = 3.135462$. Both branches anchor to the same halo, with initial root distances on the order of 10^{-3} canonical units, indicating consistent origin alignment.

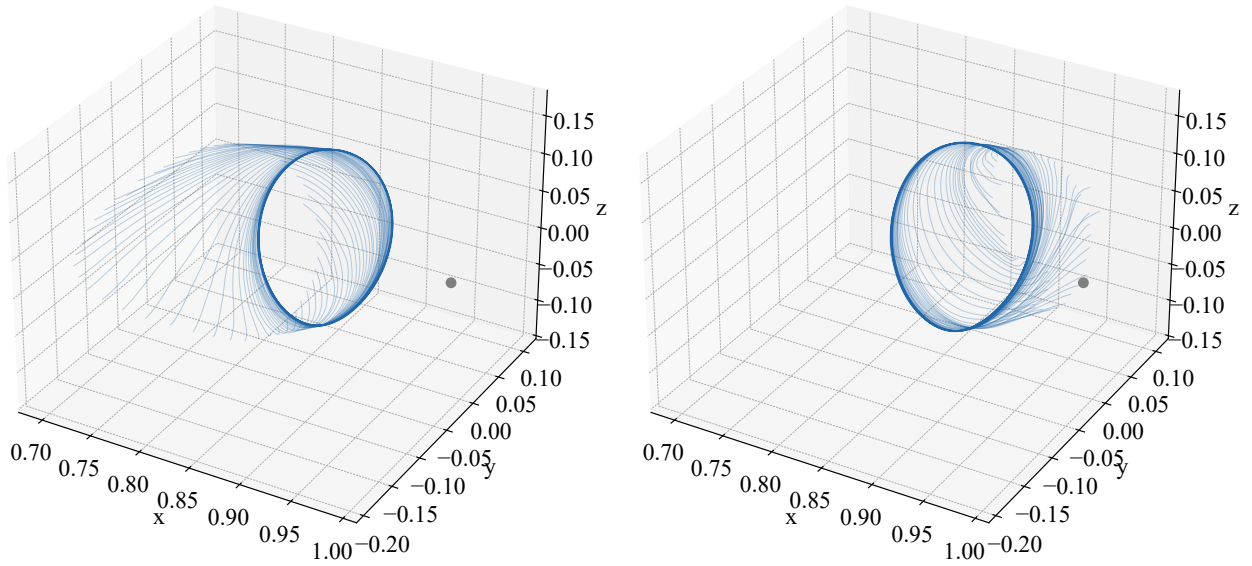


(a) Generated negative branch of a halo manifold with Jacobi constant 3.135462. (b) Generated positive branch of a halo manifold with Jacobi constant 3.135462.

Figure 4.7: Generated negative and positive halo manifolds with Jacobi constant 3.135462.

The geometry at this mid-range C forms tube-like sheets with nearly circular cross-sections; the negative branch extends outward while the positive branch remains more compact, reflecting the expected reduction in branch asymmetry relative to the higher- C case. Quantitatively, nearby points in the energy sweep yield $\max_t |C(t) - C^*|$ of around 2.5×10^{-3} for the negative branch, whereas the positive branch exhibits a larger deviation of approximately 1.27. This contrast is consistent with stronger curvature and closer secondary approaches on the positive branch at these energies.

Figure 4.8a and Figure 4.8b show the positive and negative branches at a Jacobi constant of 3.096582. Notably, there is reduced branch asymmetry compared to higher Jacobi constant cases. Similar to the reconstruction case, the trajectories near the secondary have larger errors.



(a) Generated negative branch of a halo manifold with Jacobi constant 3.096582. (b) Generated positive branch of a halo manifold with Jacobi constant 3.096582.

Figure 4.8: Generated negative and positive halo manifolds with Jacobi constant 3.096582.

Across the tested Jacobi constants, the generated manifolds exhibit the expected dynamical trends observed in the [CR3BP](#). At higher C values, trajectories become more elongated and approach the secondary more closely, displaying sharper curvature and increased asymmetry between branches. As C decreases, manifolds contract and adopt more circular geometries, consistent with the reduced velocity gradients in these regimes. This progression suggests that the model not only captures manifold geometry at discrete energies, but also reproduces its smooth variation across the energy spectrum.

The generated results also reveal certain limitations. At high C values, trajectories that approach the secondary exhibit larger errors due to the steep velocity gradients and strong curvature in this region. Overall, positive branches tend to incur higher errors than their negative counterparts. At low C , the manifolds for both branches become increasingly similar in geometry, resembling near-closed orbits, which can make branch differentiation more challenging for the model.

4.2.1 Benchmarking

To evaluate the computational efficiency of the trained model, manifold generation speed is benchmarked against state-of-the-art numerical methods. *Vega et al.* [5] recently demonstrated a massively parallel GPU-based method for computing invariant manifolds that achieves significant speed-ups over traditional monodromy matrix approaches. Their method generates manifolds by perturbing discrete orbit states with arbitrary unit vectors and forward/backward integrating, avoiding eigendecomposition of the monodromy matrix entirely. In their benchmark, 10000 points along a Lyapunov orbit were perturbed and forward-simulated to generate manifold trajectories discretised by a total of 20000 points (integration time of 5.971226 dimensionless time units) on a 12th generation Intel Core i9 12900K Central Processing Unit (CPU) with an NVIDIA GeForce RTX 3090 GPU, taking 5.050 ms compared to 39.050 s for the conventional CPU-based monodromy method, a speed-up of over three orders of magnitude.

The trained VQ-VAE model demonstrates competitive performance on a 13th generation Intel Core i7 13650HX CPU with an NVIDIA GeForce RTX 4080M GPU, batch size 32; generating complete manifolds (approximately 25,000 trajectory points total) in an average of 0.931 ms, achieving a rate of 1074 manifolds per second (an improvement of around 5 times). The generative approach offers the additional advantage of bypassing orbit integration entirely, generating physically consistent manifolds directly from energy specifications without requiring pre-computed periodic orbits or perturbation strategies.

4.3 Latent Space

This section characterises the geometry and semantics of the model’s latent space. The encoder’s pre-quantisation activations z_e are examined to assess how the Jacobi constant C and manifold branch organises the representation. A Uniform Manifold Approximation Projection (UMAP) embedding is used to test whether energy provides a smooth ordering and whether branch induces a separation in z_e . The continuous structure is then related to the discrete codebook learned by the vector-quantised bottleneck by clustering the code embeddings, revealing macro- and meso-scale groupings. Information-theoretic measures (entropy, conditional entropy, and mutual information) are employed to quantify the degree to which conditioning controls code selection, enabling assessment of codebook utilisation and code predictability given (C, Branch) . Collectively, the analyses indicate that energy primarily orders the latent space, branch cleanly separates it, and the codebook remains well utilised while preserving controlled diversity within each physical condition.

Figure 4.9a shows a UMAP embedding of the model’s pre-quantisation latent vectors z_e . The embedding uses 15 nearest neighbours, projects to 2 dimensions for visualization, and employs Euclidean distance with a minimum distance of 0.1 to maintain clear separation between distinct clusters. The embedding forms smooth, arc-like curves along which the (normalised) energy varies monotonically, indicating that the encoder organises samples by energy. Colouring by branch reveals two disjoint submanifolds corresponding almost perfectly to the positive and negative branches, demonstrating that branch is disentangled from energy

in z_e . The apparent “split” near the mid-levels of energy is therefore branch-driven rather than a true energy-driven bifurcation. The presence of the third trajectory with mixed branches is attributed to high-energy halos, which, have manifolds that diverge from the orbit extremely slowly, and thus are similar for both positive and negative branches for the integration periods on which the model is trained on.

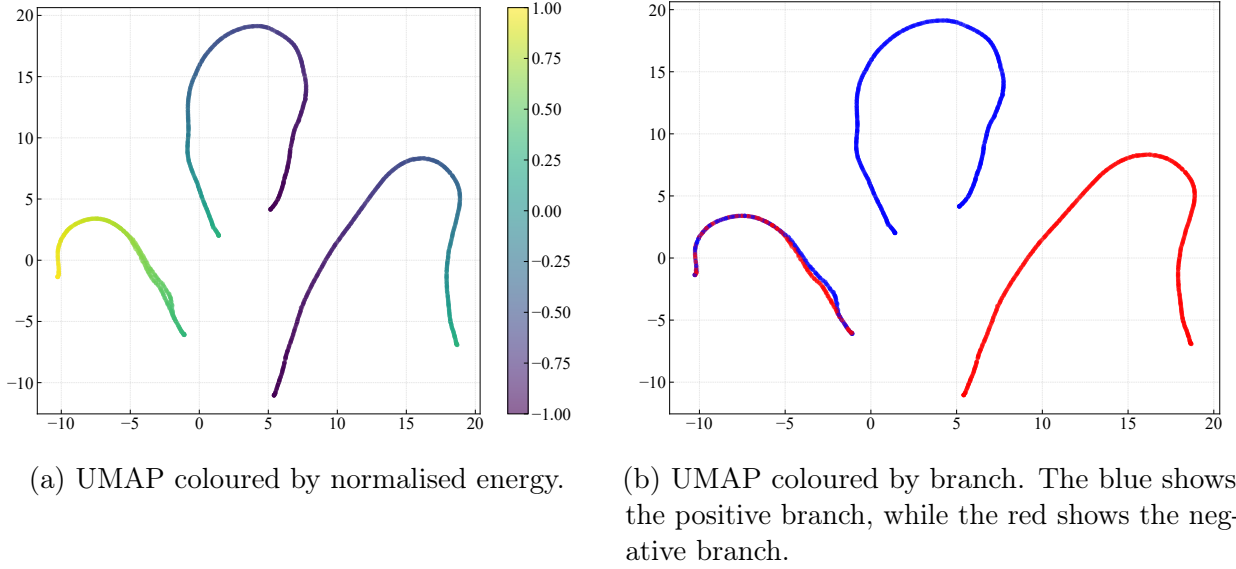


Figure 4.9: Uniform manifold approximation and projection embedding (UMAP) of the model’s pre-quantisation latent vectors z_e .

Figure 4.9b shows the same plot coloured by branch, where the blue branch is the positive branch, and red is the negative branch. The disentanglement can be clearly seen, as the model has learnt two distinct embeddings for the branches. Having established that energy orders the latent space and branch separates it, attention now turns to how the discrete bottleneck organises this structure. To relate the continuous arcs to the codebook, the learnt code embeddings are clustered.

Figure 4.10 shows a dendrogram of learnt codes created using a ward clustering algorithm. The algorithm’s goal is to find pairs of clusters to merge such that the increase in the total within-cluster variance is minimised. The y -axis shows dissimilarity between indices, while the x -axis shows the code index. The entire codebook is split into two distinct super-clusters, indicating that the model has learnt two primary types of latent features, and these two types are very dissimilar from each other. Within each of these two main groups, smaller sub-clusters arise which are less dissimilar, implying they are modelling shared or similar properties. The representation is faithful to the codebook geometry (cophenetic correlation ≈ 0.712), supporting interpretation of the merge structure. The tree exhibits a clear macro-/meso- organisation that reflects physical conditioning.

At a coarse cut (4 clusters), two clusters are branch-pure and concentrated at low energies, while two mixed clusters occupy mid/high energies. Concretely, one low-energy cluster is entirely positive-branch (positive ratio=1.000, mean energy ≈ -0.82 , narrow spread), and

a companion low-energy cluster is entirely negative-branch (positive ratio=0.000, mean ≈ -0.79). In contrast, the remaining clusters are mixed across branches with broader energy spans (means ≈ -0.10 and $\approx +0.66$, positive ratio ≈ 0.53 and 0.50).

Refining to 8 clusters reveals five branch-pure groups in distinct low-energy bands (e.g., means $\approx -0.98, -0.91, -0.75, -0.59, -0.34$ with small standard deviations), and three mixed clusters emerging at higher energies (means $\approx +0.87, +0.52, -0.02$ with positive ratio $\approx 0.51, 0.50, 0.38$). The low-energy, branch-pure groups are tight (energy spreads ~ 0.02 – 0.12), whereas mixed groups are broader (~ 0.08 – 0.20), consistent with increased sharing of codes across conditions.

At 16 clusters, the low-energy regime further decomposes into tight, branch-pure micro-clusters (small within-cluster spreads), while the mixed, higher-energy groups persist. For example, multiple clusters remain strictly positive- or negative-branch in narrow low-energy bands, whereas a high-energy cluster (mean $\approx +0.87$) retains an approximately balanced branch ratio (≈ 0.51), indicating genuine mixing in that regime.

This structural picture aligns with the conditioning analysis: in regions where clusters are branch-pure and energy-tight, conditioning effectively selects a single code (low ambiguity), whereas in regimes where the dendrogram yields mixed clusters, multiple codes are used for the same (C, Branch) (higher ambiguity). Complementarily, k -means on the continuous latent space tends to split primarily by energy with near-balanced branch ratios in each group (for example, two clusters with mean energies ≈ -0.78 and $\approx +0.29$, positive ratios ≈ 0.51 and 0.49), suggesting that strong branch separation is a property expressed by the discrete codebook rather than the continuous latent alone.

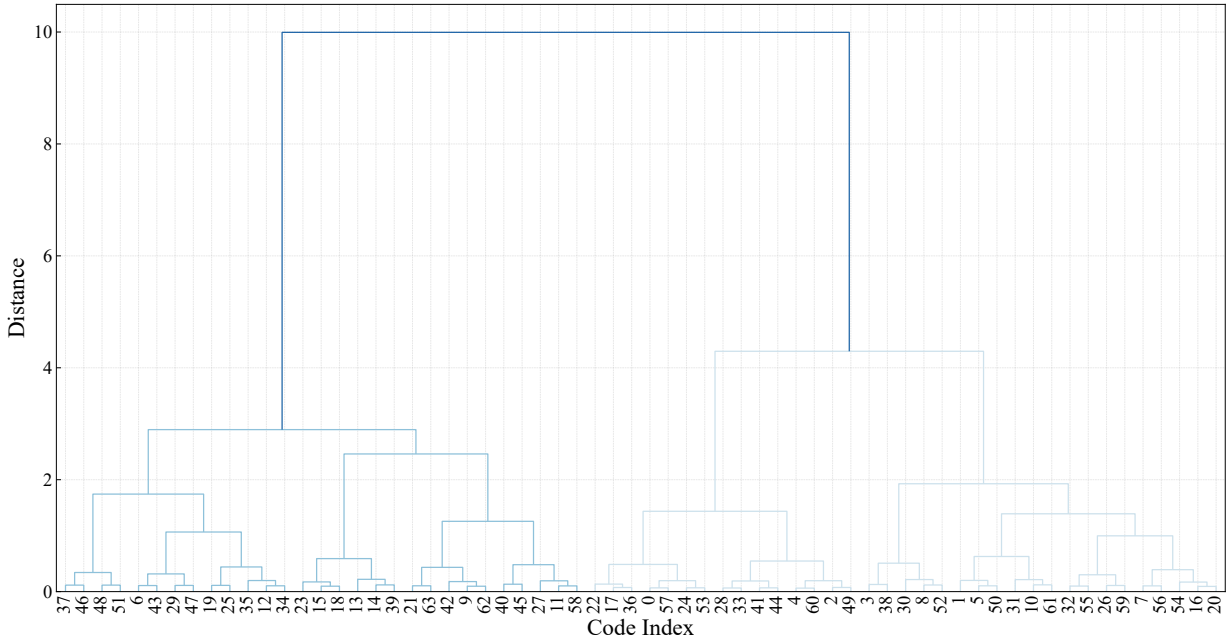
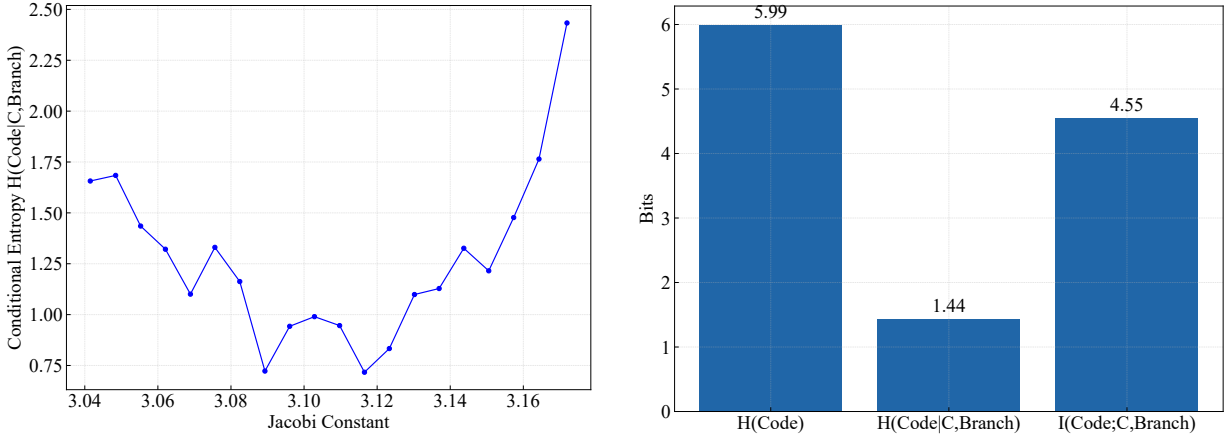


Figure 4.10: Hierarchical clustering dendrogram of learnt codes.

While the dendrogram provides structural evidence for macro- and meso-scale organisation, quantitative assessment is required to determine how strongly physical conditioning governs code selection. This dependence is evaluated using information-theoretic measures. Figure 4.11a quantifies how strongly the discrete codebook is controlled by the physical conditioning. The left panel plots the conditional entropy $H(\text{Code} \mid C, \text{Branch})$ (bits) across the Jacobi constant. Entropy is lowest in the mid-range ($\approx 3.11\text{--}3.15$), indicating that conditioning almost pins down a single code, and increases toward the edges of the range (≈ 3.04 and ≈ 3.17), where multiple codes are used for the same (C, Branch) . This pattern suggests sparser coverage near the extremes while maintaining tight control in the bulk of the data.



(a) Conditional entropy $H(\text{Code} \mid C, \text{Branch})$ (bits) across the Jacobi constant. (b) Information decomposition of learnt codebook structure.

Figure 4.11: Conditional entropy and information decomposition of learnt codebook structure.

Figure 4.11b presents an information-theoretic decomposition of the learnt codebook structure. The total code entropy reaches $H(\text{Code}) \approx 5.99$ bits, approaching the theoretical maximum of 6 bits for a 64-code vocabulary. This near-maximum entropy confirms healthy codebook utilisation with no evidence of code collapse, indicating that the model effectively uses the full range of available discrete representations.

When conditioning on physical parameters, the code uncertainty reduces substantially to $H(\text{Code} \mid C, \text{Branch}) \approx 1.44$ bits. This reduction means that approximately $2^{1.44} \approx 2.7$ codes remain plausible for any given combination of Jacobi constant and manifold branch, preserving controlled diversity while constraining generation. The explained information amounts to:

$$I(\text{Code}; C, \text{Branch}) = H(\text{Code}) - H(\text{Code} \mid C, \text{Branch}), \quad (4.6)$$

which is equal to approximately 4.55 bits, indicating that the physical conditioning parameters determine approximately 76% of the code uncertainty. This demonstrates that the learnt prior provides strong and interpretable control over the discrete representation while maintaining sufficient flexibility within each physical condition.

The information decomposition reveals how predictable the codes are from the conditioning parameters. To complement this analysis, the mutual information between codes and various physical properties is examined to understand what information the codes themselves encode.

Figure 4.12 displays mutual information values between discrete code indices and several trajectory descriptors, where continuous variables are uniformly discretised prior to computation. The Jacobi constant carries the most information about code selection, reaching approximately 3.8 bits and confirming that the codebook organisation is strongly structured by orbital energy. Geometric and kinematic trajectory descriptors demonstrate nearly equivalent informativeness, with maximum distance to primaries, mean velocity, and total arc length each contributing approximately 3.5 – 3.6 bits. This indicates that the learnt codes capture trajectory morphology and dynamical characteristics beyond simple energy classification.

Trajectory compactness shows moderate informativeness at approximately 3.1 bits, while the manifold branch alone contributes relatively little information at approximately 0.75 bits. This low branch-specific information is consistent with substantial code sharing across positive and negative manifold branches, suggesting that the model learns representations based primarily on trajectory characteristics rather than branch identity.

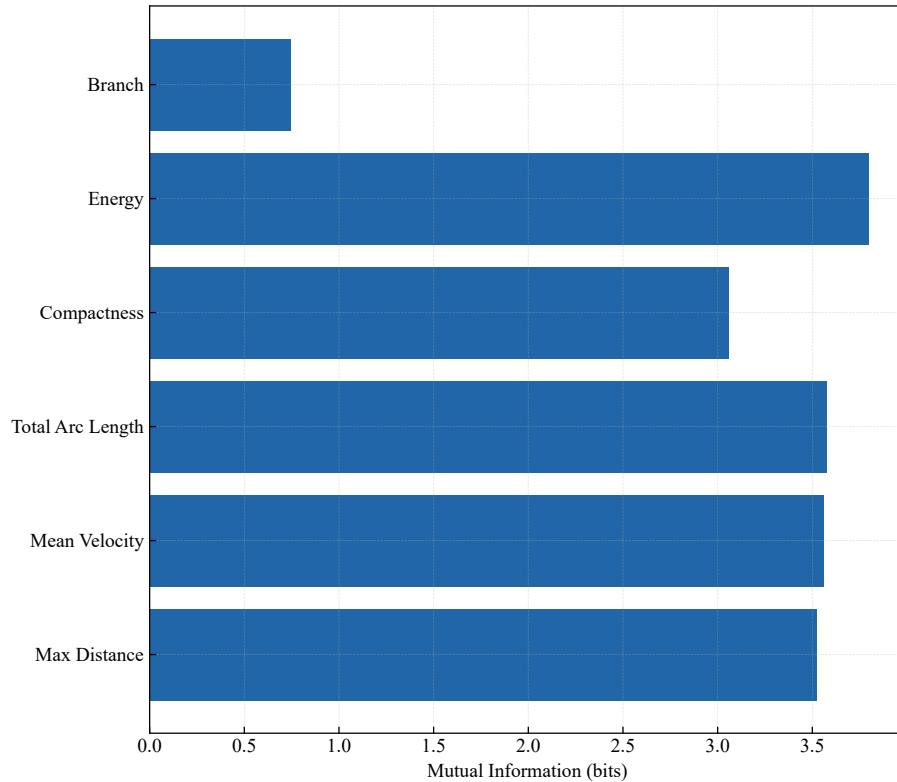


Figure 4.12: Mutual information plot of different variables.

When combined with the conditional entropy results, where $I(\text{Code}; C, \text{Branch}) \approx 4.55$ bits, these findings reveal the relative importance of different physical parameters in determining

code selection. The Jacobi constant emerges as the dominant factor, accounting for approximately 63% of the total code uncertainty on its own. The manifold branch provides only limited additional predictive power beyond energy classification.

The remaining code capacity reflects geometric and kinematic variability that cannot be fully captured by the energy and branch conditioning alone. This residual information content suggests that the learnt representations encode trajectory morphology and dynamical characteristics that extend beyond the explicit conditioning parameters.

These results demonstrate that the physics-informed [VQ-VAE](#) successfully learns a discrete latent representation of invariant manifolds while respecting the underlying [CR3BP](#) dynamics. The low reconstruction error combined with small physics residuals validates the approach of incorporating dynamical constraints directly into the learning process. The model’s ability to maintain both high fidelity reconstruction and physical consistency makes it suitable for rapid generation of invariant manifolds in the [CR3BP](#) without needing a generating orbit.

Chapter 5

Discussion

5.1 Conclusion

This thesis has investigated the application of physics-informed generative deep learning models to the problem of invariant manifold approximation in the [CR3BP](#). The work represents a preliminary exploration into the feasibility of using [VQ-VAEs](#) to learn discrete latent representations of manifold trajectories, demonstrating that such models can capture essential dynamical structures while maintaining physical consistency with the underlying equations of motion.

The theoretical foundation established in Chapter [2](#) highlighted the computational challenges inherent in traditional manifold computation methods. Classical approaches rely on eigendecomposition of monodromy matrices and propagation of stable and unstable eigenvectors using state transition matrices, techniques that scale poorly with the number of periodic orbits and become computationally prohibitive for real-time applications or onboard spacecraft systems. The prevalence of small spacecraft platforms in Cis-Lunar space, particularly CubeSats with constrained computational resources, underscores the need for more efficient alternatives. Furthermore, traditional methods are fundamentally limited to periodic orbits, making them unsuitable for higher-fidelity models where such periodic solutions may not exist or are difficult to compute.

The proposed [VQ-VAE](#) framework addresses several of these limitations through its data-driven approach. By learning directly from trajectory data rather than requiring explicit periodic orbit computation, the model offers a potential pathway toward manifold approximation in scenarios where traditional methods fail. The discrete latent representation provides interpretable structure while maintaining the generative capabilities necessary for trajectory generation. The integration of physics-informed loss functions helps generated trajectories respect the fundamental dynamics of the [CR3BP](#), a critical requirement for any practical application in astrodynamics.

The numerical methodology developed in Chapter [3](#) successfully generated a comprehensive

dataset of 1252 manifold trajectories associated with Earth-Moon L_1 northern halo orbits across Jacobi constants ranging from 3.03826 to 3.17434. The centre manifold reduction technique, based on the methods of Jorba and Masdemont [13], implemented through a custom Python library, enabled accurate computation of seed orbits and their associated manifolds. This dataset, encompassing both positive and negative branches with integration periods of 5 dimensionless time units, provides a solid foundation for training and validation of the generative model.

The results presented in Chapter 4 demonstrate the feasibility of the proposed approach while revealing important limitations. Reconstruction performance achieved a mean scaled error of 4.87×10^{-5} across all 1250 test cases, with the final physics residual loss of 3.40×10^{-4} , showing that the manifolds are dynamically feasible. The model successfully learned to distinguish between positive and negative manifold branches, with the codebook achieving healthy utilisation and demonstrating clear energy-dependent organisation in the latent space.

However, significant performance disparities emerged between manifold branches. The negative branch consistently achieved superior reconstruction accuracy (mean 2.51×10^{-5}) compared to the positive branch (mean 7.23×10^{-5}), with this difference attributed to the positive branch’s closer approaches to the secondary body, resulting in steeper velocity gradients and stronger curvature that amplify reconstruction errors. Energy-dependent performance patterns revealed optimal reconstruction in mid-range Jacobi constants ($C = 3.11 - 3.15$), with degraded performance at both extremes where manifolds exhibit either excessive elongation or reduced geometric distinctiveness.

A mutual information analysis provided crucial insights into the model’s learned representations. Energy emerged as the dominant organisational principle (≈ 3.8 bits of mutual information), while branch information contributed relatively little (≈ 0.75 bits), suggesting that the model primarily organises trajectories by their energy characteristics rather than their topological branch properties. This finding has important implications for the model’s ability to generalise across different manifold types and suggests potential limitations in scenarios where branch discrimination is critical.

From a broader astrodynamics perspective, this work establishes a foundation for data-driven approaches to manifold-based trajectory design. The ability to generate physically consistent manifold approximations without explicit periodic orbit computation could enable new classes of mission design tools, particularly for preliminary analysis and rapid trajectory cost estimation. The discrete latent representation offers potential for interpretable trajectory clustering and could facilitate the discovery of novel manifold structures through latent space exploration.

Nevertheless, it is crucial to emphasise that this work represents preliminary feasibility demonstration rather than a production-ready tool. The current model’s limitations are substantial and multifaceted. The integration time of 5 dimensionless time units (approximately 3 weeks in physical time) captures only the initial segments of manifold trajectories, missing the long-term asymptotic behaviour that is often critical for mission applications. The restriction to Earth-Moon L_1 northern halo stable manifolds represents a narrow slice of the rich manifold

ecosystem in multi-body dynamics, excluding unstable manifolds, other Lagrange points, and different orbit families that are essential for comprehensive trajectory design.

The physics-informed loss, while effective at maintaining local dynamical consistency, does not guarantee global trajectory accuracy over extended integration periods. The observed accumulation of reconstruction errors, particularly in regions of strong curvature near the secondary body, suggests that the current approach may not achieve the precision required for high-fidelity mission design. Furthermore, the model’s performance degradation at extreme Jacobi constants indicates limited robustness across the full operational envelope of typical space missions.

The discrete nature of the [VQ-VAE](#) representation, while offering interpretability advantages, introduces quantisation artifacts that may not be suitable for applications requiring smooth trajectory variations.

Additionally, the computational requirements for training, while manageable for research purposes, may not scale favourably for operational systems requiring real-time manifold generation across multiple celestial body systems. The need for extensive pre-computed training datasets limits the model’s adaptability to new dynamical environments or perturbed systems without retraining.

Despite these limitations, the work demonstrates that generative deep learning approaches can successfully capture essential manifold characteristics while respecting underlying physical constraints. The energy-dependent organisation of the latent space suggests that the model learns meaningful dynamical principles rather than merely memorising training examples. The successful integration of physics-informed loss functions provides a template for incorporating physical constraints into generative models for other dynamical systems.

This preliminary investigation establishes the viability of machine learning approaches for invariant manifold approximation and highlights both the promise and challenges of such methods. While the current implementation falls short of practical deployment requirements, it provides crucial insights into the fundamental feasibility of data-driven manifold learning and identifies clear pathways for future development. The work contributes to the growing intersection of machine learning and astrodynamics, offering a novel perspective on classical problems in celestial mechanics and laying groundwork for more sophisticated approaches to manifold-based trajectory design.

5.2 Recommendations and Future Work

The preliminary results of this investigation reveal substantial opportunities for extending and improving the proposed approach. The following recommendations outline critical directions for future research, organised by their potential impact and feasibility. Each direction addresses specific limitations identified in the current work while building toward more comprehensive and practical manifold learning systems.

5.2.1 Extended Time Horizons

The current integration period of 5 dimensionless time units represents one of the most significant limitations of the present work, capturing only the initial trajectory segments rather than the asymptotic behaviour that defines invariant manifolds. Future research should systematically extend integration periods to capture the full manifold structure, ideally encompassing multiple orbital periods and reveal the long-term geometric characteristics essential for mission design.

Extending integration times introduces several technical challenges that require careful consideration. Longer trajectories increase memory requirements quadratically and computational complexity, necessitating more efficient data representation schemes. The accumulation of numerical integration errors over extended periods may compromise trajectory fidelity, requiring higher-order integration schemes or adaptive time-stepping methods. Additionally, longer trajectories may exhibit greater geometric complexity and variability, challenging the model’s capacity to learn coherent representations.

To address these challenges, future work should explore hierarchical representation schemes that capture manifold structure at multiple temporal scales. Coarse-scale features could encode overall trajectory topology and asymptotic directions, while fine-scale components capture local geometric details and curvature variations. Such multi-resolution approaches could enable efficient storage and processing of long trajectories while maintaining the detail necessary for accurate reconstruction.

The integration time extension should be coupled with comprehensive validation studies that assess trajectory accuracy using independent integration schemes and compare generated manifolds with high-precision reference solutions. Particular attention should be paid to the asymptotic behaviour of generated trajectories, ensuring they converge to the correct equilibrium points and maintain physically realistic departure rates from periodic orbits.

5.2.2 Improved Manifold Coverage

The current work’s focus on stable manifolds of Earth-Moon L_1 northern halos represents a narrow subset of the manifold structures relevant to astrodynamics applications. A comprehensive manifold learning system must encompass unstable manifolds, southern halo orbits, and manifolds associated with other orbit families including Lyapunov, axial, and vertical periodic orbits. Each manifold type exhibits distinct geometric characteristics and dynamical properties that require systematic investigation.

Future work should systematically expand the training dataset to include representative samples from all major orbit families, ensuring balanced coverage across different energy levels and dynamical regimes. The dataset expansion should be coupled with careful analysis of manifold similarities and differences to inform architectural choices and identify opportunities for transfer learning between related manifold types.

The restriction to L_1 manifolds limits the practical applicability of the current approach, as real missions often exploit connections between multiple Lagrange points or transition between

different dynamical regions. Earth-Moon L_2 manifolds, for instance, provide pathways to lunar distant retrograde orbits and serve as gateways to interplanetary space. Sun-Earth Lagrange points offer access to heliocentric orbits and deep space observatories. A comprehensive system must address manifolds associated with all the collinear points.

The extension to multiple Lagrange points should investigate unified representation schemes that capture common dynamical principles while accommodating point-specific characteristics. Conditional generation frameworks could enable single models to generate manifolds for different Lagrange points based on positional or parametric conditioning. Alternatively, ensemble approaches might employ specialised models for each point while sharing learned features through transfer learning or multi-task training schemes.

5.2.3 Alternative Generative Architectures

The [VQ-VAE](#) architecture, while effective for discrete representation learning, introduces quantisation artifacts and may not optimally capture the continuous nature of manifold structures. Future research should systematically explore alternative generative architectures, each offering distinct advantages for manifold learning applications.

Diffusion models have demonstrated remarkable success in high-dimensional data generation and offer several potential advantages for trajectory synthesis. Their iterative de-noising process could naturally incorporate physical constraints at each step, potentially achieving superior dynamical consistency compared to single-step generation approaches.

Variational autoencoders with learned priors (such as β -VAE or Factor-VAE) could provide more flexible latent representations while maintaining the probabilistic framework essential for uncertainty quantification in [CR3BP](#) manifold generation. However, experiments with standard VAE variants consistently encountered posterior collapse, where the approximate posterior $q(z|x)$ degenerates to the prior $p(z)$, effectively ignoring the input data and severely limiting the model’s capacity to learn meaningful manifold representations.

To address these limitations while preserving continuous latent spaces, architectures such as [Linear Identifiable Deep Variational Autoencoders \(LIDVAE\)s](#) [40]. [LIDVAEs](#) fundamentally avoid posterior collapse through their constrained linear decoder architecture, which ensures identifiability of the latent representation. Unlike [VQ-VAEs](#), which discretise the latent space and may inadequately capture the continuous parameter variations inherent in [CR3BP](#) manifold structures, [LIDVAEs](#) maintain continuous latent representations while providing theoretical guarantees on representation learning.

5.2.4 Physics Constraints and Accuracy

While the current physics-informed loss function successfully maintains local dynamical consistency, more sophisticated approaches to physics integration could significantly improve trajectory accuracy and physical realism. [Neural Ordinary Differential Equation \(NODE\)s](#) represent a particularly promising direction, as they explicitly model the continuous-time dynamics underlying trajectory evolution.

NODE based approaches could replace the current discrete trajectory representation with continuous dynamical models that inherently satisfy differential equation constraints. Such models would naturally handle variable integration times and could adapt their temporal resolution based on trajectory complexity. The continuous formulation might better capture the smooth manifold structure while avoiding the discretisation artifacts present in current approaches.

Constraint satisfaction approaches, such as projection-based methods or Lagrange multiplier techniques, could ensure hard constraint satisfaction rather than the soft constraint enforcement provided by loss function penalties. Such approaches might be essential for mission-critical applications where trajectory accuracy requirements are stringent.

Future work should systematically compare different physics integration approaches, evaluating their effectiveness across various trajectory complexity levels and integration time horizons. The computational overhead associated with more sophisticated physics constraints must be balanced against the achieved accuracy improvements to identify optimal approaches for different application scenarios.

5.2.5 Architectural Improvements

The current single-scale architecture treats all temporal and spatial scales uniformly, potentially missing important multi-scale structure inherent in manifold dynamics. Hierarchical architectures that explicitly model different scales of manifold structure could achieve superior performance while providing more interpretable representations.

Attention mechanisms offer promising approaches for capturing long-range dependencies in trajectory data. Self-attention layers could identify critical trajectory segments that influence long-term behaviour while down-weighting regions with minimal dynamical significance. Cross-attention between different trajectory scales could enable the model to focus on relevant features at appropriate temporal and spatial resolutions.

The exploration of dual-bottleneck architectures presents another intriguing direction. Separate latent representations for global manifold topology and local trajectory details could provide more flexible and interpretable models. Global representations could capture overall manifold shape and asymptotic directions, while local representations handle fine-scale curvature and velocity variations. Such separation might enable more targeted optimisation of different aspects of trajectory fidelity.

5.2.6 Continuous Manifold Representations

The current trajectory-based discretisation of manifold structures, while computationally tractable, may miss important geometric properties of the underlying continuous manifolds. Future research should explore continuous representation schemes that directly model manifold geometry rather than sampling discrete trajectories. These methods represent manifolds as continuous functions that can be queried at arbitrary points in phase space, potentially providing infinite resolution and eliminating discretisation artifacts. Such approaches could

naturally handle variable manifold topology and provide smooth interpolation between different regions.

Parametrised manifold learning techniques, such as those based on differential geometry or Riemannian manifolds, could provide more mathematically principled approaches to manifold representation. These methods explicitly model the geometric structure of manifolds while respecting their intrinsic dimensionality and curvature properties. The integration of differential geometric principles could ensure that learned representations preserve essential topological and geometric properties.

Level set methods offer another avenue for continuous manifold representation, particularly for capturing manifold boundaries and intersection regions. These approaches represent manifolds as zero-level sets of higher-dimensional functions, enabling natural handling of topological changes and manifold evolution. The ability to represent manifold intersections explicitly could be particularly valuable for identifying heteroclinic connections and transfer opportunities.

Neural fields approaches, which represent spatial functions as neural networks, could provide flexible frameworks for continuous manifold modelling. Such methods have demonstrated success in representing complex geometric structures and could be adapted to handle the dynamical constraints inherent in invariant manifolds. The continuous nature of neural fields could eliminate many of the discretisation issues present in current approaches. [48]

5.2.7 Higher-Fidelity Dynamical Models

The restriction to the [CR3BP](#), while providing a clean theoretical framework, limits the practical applicability of learned manifold representations. Real space missions operate in perturbed environments where additional forces significantly modify manifold structures. Future work must address the extension to higher-fidelity models that include perturbations from solar radiation pressure, gravitational harmonics, and additional celestial bodies.

The elliptic restricted three-body problem, which accounts for eccentric primary orbits, represents the most natural first extension. The time-varying geometry introduces periodic modulation of manifold structures that requires more sophisticated representation schemes. Seasonal variations in manifold geometry could be captured through time-dependent conditioning or recurrent architectural components.

Solar radiation pressure perturbations, particularly relevant for spacecraft with large area-to-mass ratios, introduce additional complexity through their dependence on spacecraft orientation and solar distance. The incorporation of such perturbations requires extending the state space to include attitude dynamics or developing simplified models that capture essential perturbation effects without excessive complexity.

Multi-body perturbations from the Sun, other planets, or natural satellites create rich dynamical environments where traditional manifold concepts become more complex. The extension to such systems might require fundamental changes to the manifold learning

approach, potentially moving from periodic orbit-based manifolds to more general invariant structures.

The challenge of extending to higher-fidelity models extends beyond merely adding perturbation terms to the equations of motion. The absence of exact periodic solutions in perturbed systems eliminates the theoretical foundation upon which traditional manifold computation relies. Machine learning approaches might provide unique advantages in such environments by learning approximate invariant structures directly from trajectory data without requiring perfect periodic solutions.

Future research should systematically investigate the robustness of learned manifold representations to different perturbation levels and types. The ability to capture perturbation effects through conditioning or architectural modifications could enable single models to handle multiple dynamical environments. Alternatively, transfer learning approaches might enable rapid adaptation of models trained on simplified dynamics to more complex perturbed systems.

5.2.8 Real-Time Generation and Computational Efficiency

The ultimate goal of manifold learning approaches is to enable real-time trajectory generation for operational mission design applications. Current model evaluation times, while faster than traditional manifold computation methods, may not meet the stringent requirements for onboard spacecraft systems or interactive design tools. Future research must systematically address computational efficiency and real-time generation capabilities.

Model compression techniques, including pruning, quantisation, and knowledge distillation, offer pathways to reduce model complexity while maintaining accuracy. Sparse neural networks might achieve comparable performance with significantly reduced computational requirements, enabling deployment on resource-constrained spacecraft systems. The discrete nature of [VQ-VAE](#) representations provides natural compression opportunities through codebook optimisation and redundancy elimination.

Hardware acceleration approaches, particularly [GPU](#) and specialised neural processing units, could enable real-time generation through parallel processing. The embarrassingly parallel nature of manifold generation across different energy levels and initial conditions makes such problems well-suited to accelerated computing platforms. Custom hardware implementations might achieve orders-of-magnitude performance improvements for specific applications.

Progressive generation schemes could enable interactive applications by providing coarse trajectory approximations rapidly while refining details through iterative processing. Such approaches might generate overall manifold topology in milliseconds while requiring longer processing times for high-fidelity details. The multi-scale nature of manifold structure provides natural breakpoints for progressive generation approaches.

Edge computing deployments could enable autonomous spacecraft to generate manifold approximations without ground communication, crucial for deep space missions or scenarios

with communication constraints. The development of lightweight models suitable for space-qualified computing platforms presents unique challenges related to radiation tolerance, power consumption, and thermal management.

The integration of learned manifold representations with trajectory optimisation algorithms presents exciting opportunities for hybrid approaches that combine the efficiency of learned approximations with the precision of numerical optimisation. Initial trajectory guesses generated from learned manifolds could significantly accelerate convergence of traditional optimisation methods while maintaining high accuracy requirements.

Interactive visualisation tools that enable real-time exploration of learned manifold spaces could provide unprecedented insights into multi-body dynamics and facilitate novel mission concepts. Such tools might enable mission designers to interactively explore trade-offs between different trajectory options while maintaining physical realism through learned constraints.

This comprehensive roadmap for future research demonstrates the substantial potential for advancing manifold learning approaches while acknowledging the significant challenges that remain. Each direction offers unique opportunities for contributing to the intersection of machine learning and astrodynamics while building toward more practical and capable tools for space mission design. The preliminary nature of current work should not obscure the transformative potential of these approaches, but rather motivate the sustained research effort required to realise their full capabilities.

Appendix A

Implementation of Algorithms for Centre Manifold Reduction

Given the large number of monomials required to form the expansions for both the Hamiltonian and coordinate transforms, and the lack of available algebraic software or libraries, a custom *Python* library [43] was developed that enables the computation of centre manifold and normal forms for any three-body system with reasonable efficiency.

A.1 Polynomial Representation

The polynomial representation uses a compressed monomial ordering scheme that stores multivariate polynomials as homogeneous components:

$$P(q_1, q_2, q_3, p_1, p_2, p_3) = p^{(0)} + p^{(1)} + p^{(2)} + \cdots + p^{(n)} \quad (\text{A.1})$$

where each component $p^{(i)}$ is represented by a dense coefficient array containing terms of degree i . Polynomials are initialised for a given degree and are stored in lists containing the homogeneous terms up to the desired degree. For degree i , the coefficient array has a size determined by the number of ways to choose from i items with replacement from a set of 6 variables:

$$\text{size} = {}^{i+6-1}C_i = \binom{i+5}{5} = \frac{(i+5)!}{5! \cdot i!} \quad (\text{A.2})$$

For example, a degree 2 polynomial has $\binom{2+5}{5} = 21$ terms, as shown in [Table A.1](#):

Index	Packed Multi-index	Decoded ($k_0 \dots k_5$)	Term
0	0x00000000	(2,0,0,0,0,0)	q_1^2
1	0x00000001	(1,1,0,0,0,0)	$q_1 q_2$
2	0x00000040	(1,0,1,0,0,0)	$q_1 q_3$
3	0x00001000	(1,0,0,1,0,0)	$q_1 p_1$
4	0x00040000	(1,0,0,0,1,0)	$q_1 p_2$
5	0x01000000	(1,0,0,0,0,1)	$q_1 p_3$
6	0x00000002	(0,2,0,0,0,0)	q_2^2
7	0x00000041	(0,1,1,0,0,0)	$q_2 q_3$
8	0x00001001	(0,1,0,1,0,0)	$q_2 p_1$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
19	0x01040000	(0,0,0,0,1,1)	$p_2 p_3$
20	0x02000000	(0,0,0,0,0,2)	p_3^2

Table A.1: Packed and decoded polynomial structure for degree 2.

The representation uses a bit-packing scheme where each multi-index, representing the exponents of a monomial, is encoded into a compact 32-bit integer. This encoding allocates six bits for each of five variables (q_2, q_3, p_1, p_2, p_3), with the sixth variable's exponent (q_1) determined implicitly by the total degree constraint. This approach provides a memory-efficient representation while maintaining fast access patterns for polynomial operations.

bits	0-5	6-11	12-17	18-23	24-29	Implicit
Field	$k_1(q_2)$	$k_2(q_3)$	$k_3(p_1)$	$k_4(p_2)$	$k_5(p_3)$	$k_0(q_1)$

The architecture of the polynomial system is built around three core components: the index table generation, the encoding and decoding mechanisms, and the algebraic operation kernels. The encoding system employs a dictionary-based lookup mechanism that provides constant-time access to coefficient positions, while the decoding system uses bitwise operations to extract individual exponents from the packed representation. An overview of the algorithm is shown in [Figure A.1](#).

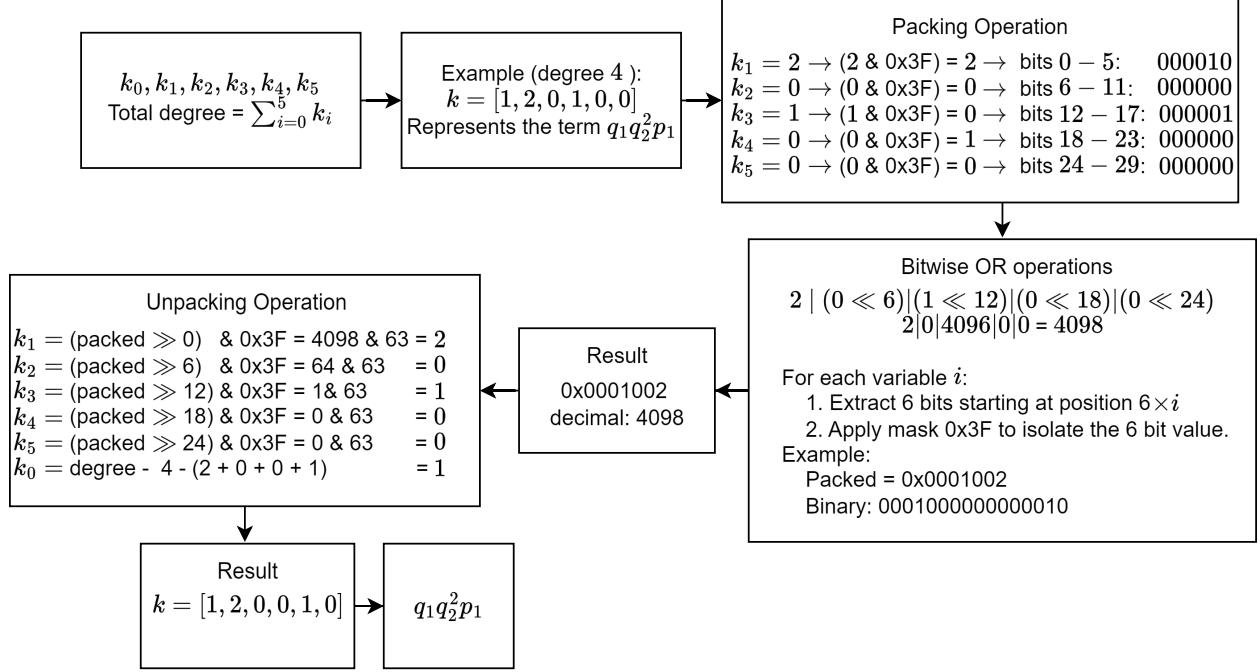


Figure A.1: Packing and unpacking operations for the custom polynomial representation.

The index tables are precomputed combinatorial structures that map between the packed multi-index representation and the coefficient array positions. These tables consist of a combination count matrix that tracks the number of monomials for each degree and variable count, and a packed multi-index list that stores all valid exponent combinations for each degree. The tables are precomputed at initialisation.

The combinatorial (ψ) table is a 2D array containing the monomial count per degree:

$$\psi[i, d] = {}^{i+d-1}C_{i-1} = \binom{i+d-1}{i-1} \quad (\text{A.3})$$

Where d is the degree of the monomial and i is the number of variables. Table A.2 shows the structure of this table for varying number of degrees and variables. It is easy to notice how the number of monomials increases significantly for higher degrees for the case $i = 6$.

i \ d	0	1	2	3	4	5	6	7	8
1	1	1	1	1	1	1	1	1	1
2	1	2	3	4	5	6	7	8	9
3	1	3	6	10	15	21	28	36	45
4	1	4	10	20	35	56	84	120	165
5	1	5	15	35	70	126	210	330	495
6	1	6	21	56	126	252	462	792	1287

Table A.2: Scaling of monomials with degree and number of variables.

The indices (*clmo*) table contains the packed multi-indices, where $\text{clmo}[D]$ is the array of packed multi-indices for degree D :

$$\text{clmo}[0] = [0x00000000] \quad (\text{A.4})$$

$$\text{clmo}[1] = [0x00000000, 0x00000001, 0x00000040, \quad (\text{A.5})$$

$$0x00001000, 0x00040000, 0x01000000] \quad (\text{A.6})$$

$$\text{clmo}[2] = [0x00000000, 0x00000001, 0x00000040, \quad (\text{A.7})$$

$$0x00001000, \dots, 0x0104000, 0x02000000] \quad (\text{A.8})$$

while an encode dictionary maps the packed indices to their array position:

$$0x00000000 : 0, \quad \text{decoded: } (2, 0, 0, 0, 0, 0) \rightarrow \text{Position } 0 \quad (\text{A.9})$$

$$0x00000001 : 1, \quad \text{decoded: } (1, 1, 0, 0, 0, 0) \rightarrow \text{Position } 1 \quad (\text{A.10})$$

$$0x00000040 : 2, \quad \text{decoded: } (0, 2, 0, 0, 0, 0) \rightarrow \text{Position } 2 \quad (\text{A.11})$$

$$\vdots \quad (\text{A.12})$$

$$0x02000000 : 20, \quad \text{decoded: } (0, 0, 0, 0, 0, 2) \rightarrow \text{Position } 20 \quad (\text{A.13})$$

The algebraic operations are implemented as *Numba*-accelerated kernels that operate directly on the coefficient arrays. The multiplication algorithm employs a parallel convolution-style approach where each thread accumulates partial results in private scratch arrays before a final reduction step combines the results. This design eliminates race conditions while maximizing parallel efficiency. The differentiation and integration operations leverage the packed multi-index representation to efficiently compute coefficient updates, while the Poisson bracket computation combines differentiation and multiplication operations in a specialized kernel that maintains the canonical structure of Hamiltonian systems.

The system incorporates several key optimisations that significantly improve performance. The use of *Numba*'s just-in-time compilation with fastmath optimisations enables near-native performance for the computational kernels. The parallel implementation of polynomial multiplication utilizes thread-local accumulation to avoid synchronisation overhead, while the bit-packing scheme reduces memory bandwidth requirements and improves cache locality. The precomputed lookup tables eliminate the need for runtime combinatorial calculations, and the homogeneous component structure allows for efficient degree-bounded operations.

The polynomial representation offers several significant advantages for dynamical systems applications. However, the representation also has notable limitations. The bit-packing scheme limits individual exponents to 63, which may be insufficient for very high-degree polynomials in certain applications. The dense representation becomes memory-intensive for sparse polynomials, where many coefficients are zero. The precomputed lookup tables require initialization time and memory overhead, though this is amortized over multiple operations.

The system is specifically designed for six-variable polynomials, limiting its generality for other applications.

These limitations are acceptable trade-offs given the specific requirements of the Lie series expansions, where the polynomials are typically dense and of moderate degree, and the computational bottleneck lies in a few, highly optimised kernels.

A.1.1 Benchmarks

To benchmark the implementation,

Polynomial Degree	Addition (ms)		Multiplication (ms)		Evaluation (ms)	
	HITEN	Sympy	HITEN	Sympy	HITEN	Sympy
2	0.05	0.00	2.81	233.08	0.09	12.31
4	0.14	0.00	2.35	6975.82	0.09	79.28
6	0.18	0.01	24.39	94716.27	0.17	319.23
8	0.16	0.00	173.54	806500.24	0.20	861.24

Table A.3: Custom polynomial representation benchmarks compared to SymPy [49] for different polynomial degrees.

A.2 Implementing the Lie series Transform

With the polynomial representation infrastructure established, implementation of the Lie series transformations described in Section 3.1 now proceeds. The key challenge lies in efficiently computing the nested Poisson brackets, where each bracket operation involves polynomial differentiation and multiplication of increasingly complex expressions.

The implementation of Lie series transformations fundamentally reduces to three core polynomial operations: multiplication, differentiation, and Poisson bracket computation. These operations form the computational backbone of the method because the Lie series expansion consists of nested Poisson brackets of increasing complexity, where each bracket $\{f, g\}$ requires computing $\sum_{i=1}^n \left(\frac{\partial f}{\partial q_i} \frac{\partial g}{\partial p_i} - \frac{\partial f}{\partial p_i} \frac{\partial g}{\partial q_i} \right)$. This requires $2n$ polynomial differentiations and $2n$ polynomial multiplications per bracket evaluation, with the results combined through polynomial addition. As the series progresses, each subsequent term involves brackets of brackets, creating a cascade of these operations where the computational cost grows exponentially with the transformation order. For instance, computing the fifth-order term $\frac{1}{5!} \{ \{ \{ \{ \{ H, G \}, G \}, G \}, G \}, G \}$ requires 80 differentiations and 80 multiplications just for the innermost operations, with intermediate results that must be stored and reused. Therefore, the efficiency of the entire transformation hinges critically on optimising these three operations.

A.2.1 Polynomial Multiplication

The multiplication of two polynomials, P and Q , is performed by leveraging their homogeneous structure. Since each polynomial is a sum of its homogeneous parts, $P = \sum_{d_1} P^{(d_1)}$ and

$Q = \sum_{d_2} Q^{(d_2)}$, their product is the sum of all cross-products of their respective components:

$$P \cdot Q = \sum_{d_1} \sum_{d_2} P^{(d_1)} \cdot Q^{(d_2)} \quad (\text{A.14})$$

The product of a term of degree d_1 and a term of degree d_2 results in a new term of degree $d_1 + d_2$.

The high-level function, `_polynomial_multiply` iterates through all pairs of homogeneous coefficient arrays from the input polynomials. For each pair, it invokes the low-level kernel, `_poly_mul`, which performs the core computation. This kernel implements a parallelized, convolution-style multiplication on the dense coefficient arrays.

The `_poly_mul` kernel is parallelized using `numba.prange`. To prevent race conditions where multiple threads might attempt to write to the same memory location simultaneously, each thread is allocated a private "scratch" array, initialized to zeros. The algorithm proceeds as follows:

1. For each pair of coefficients from the input arrays, the kernel decodes their packed multi-indices to retrieve their exponent vectors, k_i and k_j .
2. It computes the exponent vector of the resulting product monomial by performing element-wise addition: $k_s = k_i + k_j$.
3. This new exponent vector, k_s , is re-encoded into its unique packed integer representation, which is then used to find its corresponding index in the output coefficient array via the encode dictionary.
4. The product of the two input coefficients is then added to the thread's private scratch array at the calculated index.
5. After the parallel loop completes, a final reduction step is performed, where the scratch arrays from all threads are summed together to produce the final, complete coefficient array for the resulting homogeneous polynomial.

This thread-local storage approach is highly effective for dense polynomial multiplication, as it avoids the overhead associated with memory locking and allows each thread to operate independently.

A.2.2 Polynomial Differentiation

The partial differentiation kernel, `_polynomial_differentiate`, operates on a single polynomial with respect to a specified variable. The high-level function iterates through each homogeneous component of the input polynomial and applies the low-level differentiation kernel, `_poly_diff`, to its coefficient array. The derivative of a degree- d component is a polynomial of degree- $(d - 1)$, and the results are stored accordingly.

The `_poly_diff` kernel implements the power rule for polynomial differentiation. The process, also parallelized with `numba.prange`, is as follows:

1. For each coefficient in the input array, the kernel decodes its multi-index to obtain the exponent vector, k .
2. It inspects the exponent k_v that corresponds to the differentiation variable. If k_v is zero, the term's derivative is zero, and the algorithm proceeds to the next coefficient.
3. If k_v is non-zero, the new coefficient for the resulting term is calculated as the product of the original coefficient and the exponent: $c_{\text{new}} = c_{\text{old}} \cdot k_v$.
4. The exponent vector for the new term, k_{new} , is formed by decrementing k_v by one.
5. The kernel encodes k_{new} to find its position in the destination array (of degree $d - 1$) and adds c_{new} to the appropriate thread-local scratch buffer.
6. A final reduction step sums the scratch buffers into the complete output coefficient array.

A.2.3 Poisson Bracket

The Poisson bracket kernel, `_polynomial_poisson_bracket`, is a composite operation that combines the functionality of the differentiation and multiplication kernels to implement its mathematical definition:

$$\{P, Q\} = \sum_{i=1}^3 \left(\frac{\partial P}{\partial q_i} \frac{\partial Q}{\partial p_i} - \frac{\partial P}{\partial p_i} \frac{\partial Q}{\partial q_i} \right) \quad (\text{A.15})$$

The high-level function orchestrates the computation by summing the brackets of the constituent homogeneous parts, $\{P^{(d_1)}, Q^{(d_2)}\}$, which yields a term of degree $(d_1 + d_2 - 2)$.

The low-level kernel, `_poly_poisson`, executes a loop over the three canonical coordinate pairs (q_1, p_1) , (q_2, p_2) , and (q_3, p_3) . In each iteration, it performs the following steps:

1. It computes the four required partial derivatives by invoking the `_poly_diff` kernel: $\frac{\partial P}{\partial q_i}$, $\frac{\partial Q}{\partial p_i}$, $\frac{\partial P}{\partial p_i}$, and $\frac{\partial Q}{\partial q_i}$.
2. It then computes the two product terms required for the bracket by calling the `_poly_mul` kernel on the results of the differentiation step.
3. Finally, it subtracts the second product from the first and accumulates the resulting coefficient array into the final output array.

The performance of the Poisson bracket computation is directly dependent on the efficiency of the underlying differentiation and multiplication kernels. By building upon these highly-optimised functions, the system is able to compute the repeated brackets required for the Lie series transform with sufficient speed for high-degree normal form calculations.

A.2.4 The Lie Series Transform

The theoretical framework for the Lie series transform, described in Section 3.1, is implemented as a computational algorithm that operates directly on the custom polynomial representation. The implementation is divided into two primary stages: a forward transformation that normalizes the Hamiltonian by iteratively constructing a generating function, and an inverse transformation that uses this generator to compute the explicit polynomial series for the coordinate change.

A.2.5 Hamiltonian Expansion

The forward pass, which simplifies the Hamiltonian, is performed by the `_lie_transform` function. This function translates the mathematical procedure into a degree-by-degree algorithm. It iterates from degree $n = 3$ up to the maximum specified degree, performing the following sequence of operations at each step:

1. **Term Selection.** The process begins by identifying the non-resonant terms within the current homogeneous component of the Hamiltonian, $H^{(n)}$. The `_select_terms_for_elimination` function inspects the multi-index of each monomial; if the exponents of the hyperbolic variables q_1 and p_1 are not equal (i.e., $k_0 \neq k_3$), the term is flagged for elimination and collected into a new polynomial, p_{elim} .
2. **Solving the Homological Equation.** Next, the `_solve_homological_equation` function computes the generating function $G^{(n)}$ required to cancel the unwanted terms. It implements the solution to the homological equation, $\{H^{(2)}, G^{(n)}\} = -p_{\text{elim}}$, by dividing each coefficient in p_{elim} by a unique combination of the system's linear eigenvalues and frequencies, η . This step effectively computes the precise generator needed to nullify the non-resonant dynamics at degree n .
3. **Applying the Transformation.** With the generator $G^{(n)}$ for the current degree determined, the `_apply_poly_transform` function updates the full Hamiltonian. It computes the Lie series expansion $H_{\text{new}} = \exp(L_{G^{(n)}})H_{\text{old}}$ by iteratively calculating the nested Poisson brackets, $\{\dots \{ \{H_{\text{old}}, G^{(n)}\}, G^{(n)} \}, \dots \}$, using the `_polynomial_poisson_bracket` kernel. Each successive term in the series is scaled by the appropriate factorial ($1/k!$) and added to the Hamiltonian. The series is naturally truncated when the degree of the bracket terms exceeds the maximum degree of interest.
4. **Accumulating the Generator.** The generating function $G^{(n)}$ is added to a master list that accumulates the total generating function, $G = \sum_n G^{(n)}$.

After iterating through all degrees, this procedure yields the fully normalized Hamiltonian and the complete generating function, G , which encapsulates the entire coordinate transformation. Crucially, the computation of the nested Poisson brackets is efficient due to the sequential reuse of intermediate results. The calculation of a term such as $\{\{X, G^{(n)}\}, G^{(n)}\}$ is not performed from scratch; instead, the result of the inner bracket, $\{X, G^{(n)}\}$, which was computed in the previous step of the series, is reused as the input for the next bracket operation. This avoids redundant computations and is critical for performance. Additionally, the generating

functions $G^{(3)} \dots G^{(n)}$ are cached for reuse in the coordinate transforms.

A.2.6 Coordinate Transformation Expansions

Once the total generating function G is known from the Hamiltonian normalization process, it can be used to derive the explicit polynomial series for the coordinate transformation between the original (synodic) and the simplified (normalized) frames. The library implements both the forward and inverse mappings, which are invoked depending on the direction of the desired transformation.

Center Manifold to Synodic Transformation

To map a point from the coordinates of the centre manifold back to synodic coordinates, the transformation is defined by applying the Lie series operator $\exp(L_G)$ to the basis coordinates themselves. This is handled by the `_lie_expansion` function.

The procedure begins by initializing six "identity" polynomials, where the i -th polynomial simply represents the i -th canonical variable (e.g., $q_1, q_2, \dots, p_3$). The algorithm then iterates through the homogeneous components of the total generating function, $G = \sum_n G^{(n)}$, in ascending order of degree ($n = 3, \dots, N_{\max}$).

In each iteration, the `_apply_coord_transform` function is invoked for each of the six coordinate polynomials. This function computes the effect of the Lie series expansion for the current generator, $G^{(n)}$.

After iterating through all the generating functions, the procedure yields six polynomial expansions. Each expansion gives one of the original physical coordinates as a function of the four centre manifold coordinates.

Synodic to Center Manifold Transformation

The inverse transformation, which maps a point from the physical synodic frame onto the four-dimensional centre manifold, is implemented by applying the operator $\exp(-L_G)$. This is also handled by the `_lie_expansion` function, passing an additional flag which changes the order and sign of the operations.

The process is analogous to the forward mapping, but with two key differences. First, the function iterates through the homogeneous components of the total generating function in descending order of degree ($n = N_{\max}, \dots, 3$). Second, at each step, the sign of the generator is flipped, which ensures that the inverse operator $\exp(-L_G)$ is computed.

After the full series transformation has been applied to the input synodic coordinates (which are first mapped into the complex diagonal frame), the resulting six-dimensional coordinate in the normalized frame is explicitly projected onto the centre manifold. This is achieved by setting the components corresponding to the hyperbolic directions, q_1 and p_1 , to zero. This final projection is what reduces the dimensionality of the system and yields a point represented by the four centre manifold coordinates (q_2, p_2, q_3, p_3) .

Both forward and inverse expansions are cached to enable fast evaluation of subsequent coordinates. [Table A.4](#) shows the evaluation time and memory usage for the entire procedure for varying degrees.

Degree	Centre Manifold Reduction			Synodic to Centre Manifold			Centre Manifold to Synodic		
	Monomial Count	Compute (ms)	Memory (kb)	Monomial Count	Compute (ms)	Evaluation (ms)	Monomial Count	Compute (ms)	Evaluation (ms)
4	120	249	3.94	608	258	0.055	608	177	0.059
8	1589	3618	140.77	9132	1486	0.560	9132	1441	0.541
12	9576	29052	870.19	56228	8754	3.547	56228	10295	3.484
16	38049	190420	3497.48	225272	59813	13.328	225272	74600	13.188
20	116688	1203625	10792.03	693784	280501	38.650	693784	321558	37.419

Table A.4: Monomial count, computation time, memory usage, and evaluation time of different centre manifold expansions.



15 May 2025

Dear Mr Marconi ,

Reference: CURES/25153/2025

Project ID: 27885

Title: Physics-Informed Deep Operator Network for Learning Manifold Structures of Periodic Orbits in the Circular Restricted Three Body Problem

We are pleased to inform you that you have successfully declared that your research project is a **Literature Review – based solely on openly available literature which is in the public domain, and you are undertaking desk-based research not involving any other form of data or information.**

You have also confirmed that your project **does not meet** any of the literature review specified exceptions, listed both within the relevant section of the CURES form and below:

1) Your supervisor has requested that you apply for approval through CURES because:

- The journals or data are in a sensitive area (please discuss this with your supervisor)
- The project will be embargoed **i.e. will not be publicly available via the Cranfield library immediately or longer term**

2) Approval is/will be specifically required by another external body e.g. journal publishers

Therefore, **you do not require ethical approval** and your CURES application will be automatically closed.

Please keep a copy of this letter safe, if this exception is in relation to your thesis project, you will need to include a copy with your final thesis submission.

If you have any queries, please contact CURES Support.

We wish you every success with your project.

Regards,

CURES Team

References

- [1] W. S. Koon, M. W. Lo, J. E. Marsden, and S. D. Ross, “Heteroclinic connections between periodic orbits and resonance transitions in celestial mechanics,” Caltech and JPL, Technical Report, Oct. 1997 (cit. on p. 1).
- [2] M. W. Lo and S. D. Ross, “Surfing the solar system: Invariant manifolds and the dynamics of the solar system,” JPL, Technical Report IOM 312/97, 1997 (cit. on p. 1).
- [3] M. W. Lo and S. D. Ross, “Low-energy interplanetary transfers using lagrangian points,” Jet Propulsion Laboratory, California Institute of Technology, Technical Report NPO-20377, Nov. 1999, NASA Tech Brief, Vol. 23, No. 11 (cit. on p. 1).
- [4] B. T. Barden, “Using stable manifolds to generate transfers in the circular restricted problem of three bodies,” Master’s Thesis, Purdue University, Dec. 1994 (cit. on pp. 1, 10).
- [5] F. Vega, M. Lo, J. Sims, and Z. Manchester, “A massively parallel method for fast computation of invariant manifolds,” Jet Propulsion Laboratory, California Institute of Technology and Carnegie Mellon University, Technical Report, 2024 (cit. on pp. 2, 10, 41).
- [6] A. J. J. Cespedes, B. H. B. Pangestu, A. Hanazawa, and M. Cho, “Performance Evaluation of Machine Learning Methods for Anomaly Detection in CubeSat Solar Panels,” *Applied Sciences*, vol. 12, no. 17, p. 8634, Jan. 2022, ISSN: 2076-3417. DOI: [10.3390/app12178634](https://doi.org/10.3390/app12178634). (Accessed: 20/8/2025) (cit. on p. 2).
- [7] V. Szebehely, *Theory of Orbits*. Yale University, 1967, ISBN: 978-0-12-395732-0 (cit. on pp. 3–5).
- [8] *SOHO - NASA Science*, Dec. 2017. (Accessed: 27/8/2025) (cit. on p. 4).
- [9] “Orbit - NASA Science.” (Jun. 2022), (Accessed: 3/8/2025) (cit. on pp. 4, 19).
- [10] “Gateway.” (), URL: https://www.esa.int/Science%5C_Exploration/Human%5C_and%5C_Robotic%5C_Exploration/Exploration/Gateway (Accessed: 27/8/2025) (cit. on p. 4).
- [11] S. Wiggins, *Introduction to Applied Nonlinear Dynamical Systems and Chaos* (Texts in Applied Mathematics). New York: Springer-Verlag, 2003, vol. 2, ISBN: 978-0-387-00177-7. DOI: [10.1007/b97481](https://doi.org/10.1007/b97481). (Accessed: 27/8/2025) (cit. on pp. 5, 6, 8).

- [12] J. Henrard, “Lyapunov’s center theorem for resonant equilibrium,” *Journal of Differential Equations*, vol. 14, no. 3, pp. 431–441, Nov. 1973, ISSN: 0022-0396. DOI: [10.1016/0022-0396\(73\)90059-4](https://doi.org/10.1016/0022-0396(73)90059-4). (Accessed: 27/8/2025) (cit. on p. 6).
- [13] À. Jorba and J. Masdemont, “Dynamics in the center manifold of the collinear points of the restricted three body problem,” *Physica D: Nonlinear Phenomena*, vol. 132, no. 1-2, pp. 189–213, Jul. 1999, ISSN: 0167-2789. DOI: [10.1016/s0167-2789\(99\)00042-1](https://doi.org/10.1016/s0167-2789(99)00042-1). (Accessed: 19/7/2025) (cit. on pp. 6, 19, 48).
- [14] W. S. Koon, M. W. Lo, J. E. Marsden, and S. D. Ross, *Dynamical Systems, the Three-Body Problem, and Space Mission Design*. Jun. 2022, vol. 4 (cit. on p. 6).
- [15] J. M. Lee, *Manifolds and Differential Geometry* (Graduate Studies in Mathematics). Providence: American Mathematical Society, 2009, vol. 107, ISBN: 978-0-8218-4815-9 (cit. on p. 7).
- [16] N. Fenichel, “Persistence and smoothness of invariant manifolds for flows,” *Indiana Univ. Math. J.*, vol. 21, pp. 193–226, 3 1972, ISSN: 0022-2518 (cit. on p. 7).
- [17] N. Fenichel, “Asymptotic stability with rate conditions,” *Indiana Univ. Math. J.*, vol. 23, pp. 1109–1137, 12 1974, ISSN: 0022-2518 (cit. on p. 7).
- [18] N. Fenichel, “Asymptotic stability with rate conditions, ii,” *Indiana Univ. Math. J.*, vol. 26, pp. 81–93, 1 1977, ISSN: 0022-2518 (cit. on p. 7).
- [19] J. Guckenheimer, “Bifurcation,” *Scholarpedia*, vol. 2, no. 6, p. 1517, Jun. 2007, ISSN: 1941-6016. DOI: [10.4249/scholarpedia.1517](https://doi.org/10.4249/scholarpedia.1517). (Accessed: 27/8/2025) (cit. on p. 9).
- [20] J. Cresson and S. Wiggins, *A lambda-lemma for normally hyperbolic invariant manifolds*, Oct. 2005. DOI: [10.48550/arXiv.math/0510645](https://doi.org/10.48550/arXiv.math/0510645). arXiv: [math/0510645](https://arxiv.org/abs/math/0510645). (Accessed: 27/8/2025) (cit. on p. 9).
- [21] G. Gómez, À. Jorba, J. Masdemont, and C. Simó, “A Dynamical Systems Approach for the Analysis of the SOHO Mission,” in *Proceedings of the Third International Symposium on Spacecraft Flight Dynamics*, Darmstadt, Germany, 1991, pp. 449–454 (cit. on p. 9).
- [22] W. S. Koon, M. W. Lo, J. E. Marsden, and S. D. Ross, “THE GENESIS TRAJECTORY AND HETEROCLINIC CONNECTIONS,” *American Astronautical Society*, 2000 (cit. on p. 9).
- [23] S. B. Broschart, M.-K. J. Chung, S. J. Hatch, J. H. Ma, T. H. Sweetser, S. S. Weinstein-Weiss, and V. Angelopoulos, “PRELIMINARY TRAJECTORY DESIGN FOR THE ARTEMIS LUNAR MISSION,” *American Astronautical Society*, 2009 (cit. on pp. 9, 19).
- [24] J. E. Marsden and S. D. Ross, “(PDF) New methods in celestial mechanics and mission design,” *Bulletin of the American Mathematical Society*, Jan. 2006. DOI: [10.1090/S0273-0979-05-01085-2](https://doi.org/10.1090/S0273-0979-05-01085-2) (cit. on p. 9).
- [25] A. Haro and R. de la Llave, “A parameterization method for the computation of invariant tori and their whiskers in quasi-periodic maps: Numerical algorithms,” *Discrete and Continuous Dynamical Systems - Series B*, vol. 6, no. 6, p. 1261, 2006 (cit. on p. 10).

- [26] B. Kumar, R. L. Anderson, and R. de la Llave, “Rapid and accurate methods for computing whiskered tori and their manifolds in periodically perturbed planar circular restricted 3-body problems,” *Celestial Mechanics and Dynamical Astronomy*, vol. 134, no. 1, p. 3, 2022 (cit. on p. 10).
- [27] F. Tagliaferri, E. Blazquez, G. Acciarini, and D. Izzo, “Global optimization for trajectory design via invariant manifolds in the earth-moon circular restricted three-body problem,” *arXiv preprint arXiv:2405.18916*, 2024, Preprint (cit. on p. 10).
- [28] European Space Agency, *Machine learning for trajectory design in multi-body dynamics*, https://www.esa.int/gsp/ACT/projects/ml_for_cr3bp/, Accessed: February 18, 2025 (cit. on p. 10).
- [29] D. Izzo, M. Märten, and B. Pan, “A survey on artificial intelligence trends in spacecraft guidance, dynamics, and control,” *arXiv preprint arXiv:1812.02948*, 2018, Preprint (cit. on p. 11).
- [30] A. Cassioli, D. Di Lorenzo, M. Locatelli, F. Schoen, and M. Sciandrone, “Machine learning for global optimization,” *Computational Optimization and Applications*, vol. 51, no. 1, pp. 279–303, Jan. 2012, ISSN: 1573-2894. DOI: [10.1007/s10589-010-9330-x](https://doi.org/10.1007/s10589-010-9330-x). (Accessed: 27/8/2025) (cit. on p. 11).
- [31] V. Shah and R. Beeson, “Rapid approximation of invariant manifolds using machine learning methods,” in *Adv Astronaut Sci*, vol. 162, Univelt Inc., 2018, pp. 2583–2599, ISBN: 978-0-87703-645-6 (cit. on p. 11).
- [32] D. Izzo, D. Tailor, and T. Vasileiou, *On the stability analysis of deep neural network representations of an optimal state-feedback*, Apr. 2019. DOI: [10.48550/arXiv.1812.02532](https://doi.org/10.48550/arXiv.1812.02532). arXiv: [1812.02532](https://arxiv.org/abs/1812.02532) [cs]. (Accessed: 27/8/2025) (cit. on p. 11).
- [33] R. Fufaro and R. Linares. “Deep Learning for Autonomous Lunar Landing.” (), URL: https://www.researchgate.net/publication/328027107%5C_Deep%5C_Learning%5C_for%5C_Autonomous%5C_Lunar%5C_Landing (Accessed: 27/8/2025) (cit. on p. 11).
- [34] L. Yang, Z. Zhang, Y. Song, S. Hong, R. Xu, Y. Zhao, W. Zhang, B. Cui, and M.-H. Yang, *Diffusion Models: A Comprehensive Survey of Methods and Applications*, Dec. 2024. DOI: [10.48550/arXiv.2209.00796](https://doi.org/10.48550/arXiv.2209.00796). eprint: [2209.00796](https://arxiv.org/abs/2209.00796) (cs). (Accessed: 27/8/2025) (cit. on p. 11).
- [35] D. P. Kingma and M. Welling, “An Introduction to Variational Autoencoders,” *Foundations and Trends® in Machine Learning*, vol. 12, no. 4, pp. 307–392, 2019, ISSN: 1935-8237, 1935-8245. DOI: [10.1561/22000000056](https://doi.org/10.1561/22000000056). eprint: [1906.02691](https://arxiv.org/abs/1906.02691) (cs). (Accessed: 27/8/2025) (cit. on p. 11).
- [36] A. F. Gil, W. Litteri, V. Rodriguez-Fernandez, D. Camacho, and M. Vasile, “Generative design of periodic orbits in the restricted three-body problem,” *arXiv preprint arXiv:2408.03691*, 2024, Preprint (cit. on p. 12).

- [37] D. Bank, N. Koenigstein, and R. Giryes, “Autoencoders,” in *Machine Learning for Data Science Handbook: Data Mining and Knowledge Discovery Handbook*, L. Rokach, O. Maimon, and E. Shmueli, Eds., Cham: Springer International Publishing, 2023, pp. 353–374, ISBN: 978-3-031-24628-9. DOI: [10.1007/978-3-031-24628-9_16](https://doi.org/10.1007/978-3-031-24628-9_16). (Accessed: 27/8/2025) (cit. on p. 12).
- [38] M. N. Bernstein, *Variational autoencoders*, <https://mbernste.github.io/posts/vae/>, Mar. 2023. (Accessed: 27/8/2025) (cit. on p. 13).
- [39] R. M. Neal and G. E. Hinton, “A View of the Em Algorithm that Justifies Incremental, Sparse, and other Variants,” in *Learning in Graphical Models*, M. I. Jordan, Ed., Dordrecht: Springer Netherlands, 1998, pp. 355–368, ISBN: 978-94-011-5014-9. DOI: [10.1007/978-94-011-5014-9_12](https://doi.org/10.1007/978-94-011-5014-9_12). (Accessed: 27/8/2025) (cit. on p. 14).
- [40] Y. Wang, D. Blei, and J. P. Cunningham, “Posterior Collapse and Latent Variable Non-identifiability,” in *Advances in Neural Information Processing Systems*, vol. 34, Curran Associates, Inc., 2021, pp. 5443–5455. (Accessed: 2/8/2025) (cit. on pp. 14, 16, 51).
- [41] A. van den Oord, O. Vinyals, and K. Kavukcuoglu, *Neural Discrete Representation Learning*, May 2018. DOI: [10.48550/arXiv.1711.00937](https://doi.org/10.48550/arXiv.1711.00937). arXiv: [1711.00937](https://arxiv.org/abs/1711.00937) [cs]. (Accessed: 27/8/2025) (cit. on p. 14).
- [42] A. Łańcucki, J. Chorowski, G. Sanchez, R. Marxer, N. Chen, H. J. G. A. Dolfing, S. Khurana, T. Alumäe, and A. Laurent, “Robust Training of Vector Quantized Bottleneck Models,” in *2020 International Joint Conference on Neural Networks (IJCNN)*, Jul. 2020, pp. 1–7. DOI: [10.1109/IJCNN48605.2020.9207145](https://doi.org/10.1109/IJCNN48605.2020.9207145). arXiv: [2005.08520](https://arxiv.org/abs/2005.08520) [cs]. (Accessed: 27/8/2025) (cit. on p. 16).
- [43] G. Marconi. “Hiten - computational toolkit for the circular restricted three-body problem.” (Aug. 2025), URL: <https://www.github.com/iangadmarconi/hiten> (Accessed: 5/8/2025) (cit. on pp. 19, 56).
- [44] “Halo-Orbit Insertion of Aditya-L1 Successfully Accomplished.” (May 2024), (Accessed: 3/8/2025) (cit. on p. 19).
- [45] C. Simó, G. Gómez, J. Llibre, R. Martínez, and J. Rodríguez, “On the optimal station keeping control of halo orbits,” *Acta Astronautica*, Congress of the International Astronautical Federation, vol. 15, no. 6, pp. 391–397, Jun. 1987, ISSN: 0094-5765. DOI: [10.1016/0094-5765\(87\)90175-5](https://doi.org/10.1016/0094-5765(87)90175-5). (Accessed: 3/8/2025) (cit. on p. 19).
- [46] A. Razavi, A. van den Oord, and O. Vinyals, *Generating Diverse High-Fidelity Images with VQ-VAE-2*, Jun. 2019. DOI: [10.48550/arXiv.1906.00446](https://doi.org/10.48550/arXiv.1906.00446). arXiv: [1906.00446](https://arxiv.org/abs/1906.00446) [cs]. (Accessed: 5/8/2025) (cit. on p. 29).
- [47] K. Cho, B. van Merriënboer, D. Bahdanau, and Y. Bengio, *On the Properties of Neural Machine Translation: Encoder-Decoder Approaches*, Oct. 2014. DOI: [10.48550/arXiv.1409.1259](https://doi.org/10.48550/arXiv.1409.1259). arXiv: [1409.1259](https://arxiv.org/abs/1409.1259) [cs]. (Accessed: 5/8/2025) (cit. on p. 29).

- [48] Y. Xie, T. Takikawa, S. Saito, O. Litany, S. Yan, N. Khan, F. Tombari, J. Tompkin, V. Sitzmann, and S. Sridhar, *Neural Fields in Visual Computing and Beyond*, Apr. 2022. DOI: [10.48550/arXiv.2111.11426](https://doi.org/10.48550/arXiv.2111.11426). arXiv: [2111.11426](https://arxiv.org/abs/2111.11426) [cs]. (Accessed: 14/8/2025) (cit. on p. [53](#)).
- [49] *SymPy*, <https://www.sympy.org/en/index.html>, Feb. 2025. (Accessed: 27/8/2025) (cit. on p. [60](#)).