

# **Physics-Informed Neural Networks for Gravity Field Modeling**

by

**J. R. Martin**

B.S. Physics & Astronomy, University of North Carolina at Chapel Hill, 2018

B.A. Music, University of North Carolina at Chapel Hill, 2018

M.S. Aerospace Engineering Sciences, University of Colorado Boulder, 2021

A thesis submitted to the

Faculty of the Graduate School of the

University of Colorado in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

Department of Aerospace Engineering Sciences

2023

Committee Members:

Hanspeter Schaub, Chair

Jay McMahon

Daniel Scheeres

Zachary Sunberg

Roberto Furfaro

Martin, J. R. (Ph.D., Aerospace Engineering Sciences)

## Physics-Informed Neural Networks for Gravity Field Modeling

Thesis directed by Prof. Hanspeter Schaub

Gravity is among the most ubiquitous forces within astrodynamics. The motion of every planet, asteroid, and spacecraft is intrinsically influenced by the gravitational forces of objects both near and far. Despite this, no universal model of this force exists. Rather, dynamicists must choose between many different gravity models that each carry their own unique advantages and drawbacks. For example, some models are fast to compute but lack analytic rigor; some achieve high accuracy but come with computational penalties, and others still are built with intrinsic assumptions or have limited operational validity. To combat these challenges, this thesis proposes the Physics-Informed Neural Network gravity model, or PINN-GM, which shifts attention away from analytic approaches and towards data-driven models. Specifically, the PINN-GM leverages recent advances in the field of Scientific Machine Learning to blend the power of neural networks with dynamical systems theory to produce high-fidelity models of complex dynamical systems without sacrificing analytic rigor. Through multiple iterations of development, the PINN-GM now offers high-accuracy, fast execution times, data efficiency, global validity, and exact differentiability. Taken together, these attributes make the PINN-GM well-suited to assist astrodynamicists in a variety of applications including reinforcement learning, periodic orbit discovery, and orbit determination.

## **Dedication**

To my teachers.

## Acknowledgements

They say it takes a village...

No matter what I say, there are simply too many people to thank, and one page will not do it justice. Thanks to Dr. Schaub for giving me the opportunity to pivot into a new field. Thanks to my committee for the many conversations, both professional and personal on my journey through academia. Thanks to my brother who taught me how to climb (have we made it to the top yet?). Thanks to all the AVS folk, new and old, for the memories, advice, and donuts. Thanks to the Burroughs and Martin families for celebrating the highs and supporting the lows. Minor thanks to my cat Callie, whose incessant meows encourage me to go to the lab every morning. Most importantly, thanks to Melissa — for everything.

## Contents

### Chapter

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Significance . . . . .	6
1.2	History of Gravity Models . . . . .	8
1.3	Gravity Missions . . . . .	11
1.4	Goal of Thesis . . . . .	11
<b>2</b>	<b>Review of the Gravity Modeling Problem</b>	<b>14</b>
2.1	Overview . . . . .	14
2.2	Analytic Models . . . . .	14
2.2.1	Spherical Harmonics . . . . .	14
2.2.2	Ellipsoidal Harmonics . . . . .	16
2.2.3	Interior Spherical Harmonics / Bessel Models . . . . .	17
2.2.4	Mascons . . . . .	17
2.2.5	Polyhedral Model . . . . .	17
2.2.6	Heterogeneous Polyhedral . . . . .	19
2.3	Machine Learning Models . . . . .	19
2.3.1	Neural Networks . . . . .	19
2.3.2	Extreme Learning Machines . . . . .	21
2.3.3	Gaussian Processes . . . . .	22

2.3.4	GeodesyNet . . . . .	23
2.3.5	Physics-Informed Neural Networks . . . . .	23
<b>3</b>	<b>Physics-Informed Neural Network Gravity Models</b>	<b>25</b>
3.1	Generation I . . . . .	25
3.1.1	Network Architecture and Hyperparameters . . . . .	28
3.1.2	Spherical Harmonic Performance . . . . .	30
3.1.3	Representational Compactness . . . . .	35
3.1.4	Generalization . . . . .	40
3.1.5	Computational Speed . . . . .	45
3.1.6	Network Performance Applied to the Moon’s Gravity Field . . . . .	46
3.2	Generation II . . . . .	54
3.2.1	Normalization . . . . .	55
3.2.2	Feature Engineering . . . . .	55
3.2.3	Additional PINN Constraints . . . . .	56
3.2.4	Modified Network Architectures . . . . .	57
3.2.5	Performance . . . . .	59
3.3	Generation III . . . . .	72
3.3.1	Preprocessing Efforts . . . . .	72
3.3.2	Modified Loss Function to Account for high altitude Samples . . . . .	75
3.3.3	Improve Numerics by Learning a Proxy to the Potential . . . . .	76
3.3.4	Enforcing Boundary Conditions to Avoid Extrapolation Error . . . . .	80
3.3.5	Leveraging Preexisting Gravity Information into PINN-GM Solution . . . . .	83
3.4	PINN-GM-III Performance: Comparative Study . . . . .	84
3.4.1	Generation I Versus Generation III . . . . .	85
3.4.2	Generation II versus Generation III . . . . .	88
3.5	PINN-GM-III Performance: Heterogeneous Density Asteroid . . . . .	88

<b>4 Application I: Reinforcement Learning</b>	<b>94</b>
4.1 Markov Decision Process Formulation . . . . .	97
4.2 Soft-Actor Critic . . . . .	98
4.3 Environment . . . . .	100
4.4 Experiment . . . . .	102
4.5 Results . . . . .	103
<b>5 Application II: Periodic Orbit Discovery</b>	<b>108</b>
5.1 Background . . . . .	109
5.2 Methodology . . . . .	111
5.2.1 Automatic Differentiation . . . . .	114
5.2.2 Characterization of the PINN-GM . . . . .	115
5.3 PINN-GM Cartesian Shooting Method . . . . .	118
5.4 PINN-GM Orbit Element Shooting Method . . . . .	121
5.5 PINN-GM Constrained Orbital Element Shooting Method . . . . .	123
5.6 Additional Initial Conditions . . . . .	126
<b>6 Application III: Gravity Field Estimation and Filtering</b>	<b>128</b>
6.1 Overview . . . . .	128
6.2 Offline Estimation . . . . .	130
6.3 Online Estimation . . . . .	136
6.3.1 Kalman Filter . . . . .	137
6.3.2 Dynamic Model Compensation . . . . .	139
6.3.3 PINN-GM Kalman Filter . . . . .	139
6.4 Problem Setup . . . . .	142
6.5 Metrics . . . . .	145
6.6 Experiments . . . . .	147
6.6.1 Hyperparameter Search . . . . .	147

6.6.2 Sensitivity to Orbit Geometry . . . . .	153
<b>7 Conclusions</b>	<b>156</b>

<b>Bibliography</b>	<b>160</b>
---------------------	------------

## Appendix

<b>A PINN-GM-III Training Details</b>	<b>172</b>
<b>B Learning Rate Annealing Algorithm</b>	<b>174</b>
<b>C Avoiding Spectral Bias with Fourier Feature Mapping</b>	<b>178</b>
<b>D Comments on Past Machine Learning Performance</b>	<b>181</b>
<b>E Pines Algorithm</b>	<b>183</b>
E.1 Introduction . . . . .	183
E.2 Pines' Formulation . . . . .	186
E.2.1 Gravitational Potential . . . . .	186
E.2.2 Gravitational Acceleration . . . . .	187
<b>F GPU Algorithm</b>	<b>193</b>
F.1 GPGPU Software . . . . .	193
F.2 GPGPU Hardware . . . . .	194
F.3 GPGPU General Optimization Strategies . . . . .	194
F.4 Pines' Formulation Core Routines . . . . .	196
F.4.1 Core Routine 1: Data Reduction . . . . .	196
F.4.2 Core Routine 2: Legendre Matrix . . . . .	197
F.4.3 Memory Bound . . . . .	199

F.4.4 Compute Bound . . . . .	200
F.5 Scalability to Constellations . . . . .	201
F.6 Benchmarks . . . . .	202
F.7 Conclusion . . . . .	205

## Tables

### **Table**

3.1	Shared hyperparameters for the traditional and physics-informed neural networks trained in this work . . . . .	30
3.2	Unique hyperparameters for the traditional and physics-informed neural networks trained in this work . . . . .	31
3.3	Nominal Hyperparameters . . . . .	62
3.4	Machine Learning Gravity Model Statistics – See Appendix D . . . . .	72
5.1	Initial Orbital Element Distribution . . . . .	116
5.2	Initial Conditions and Solutions . . . . .	125
6.1	Initial orbital elements . . . . .	145
6.2	Initial state, covariance, process noise matrix, and measurement noise matrix . . . .	146
6.3	Orbits for Trajectory Metric . . . . .	146
6.4	Table of the metrics for the standard gravity models. . . . .	147
6.5	Hyperparameters . . . . .	148
A.1	Default Hyperparameters for PINN-GM-III . . . . .	173
C.1	Hyperparameter search for Fourier feature mapping. . . . .	179
F.1	The buffers transferred to the GPU across algorithm lifetime . . . . .	200

## Figures

### **Figure**

1.1	Gravitational Perturbations for Earth and Moon . . . . .	3
1.2	Using Scientific Machine Learning models to solve the gravity modeling problem. . . . .	4
1.3	Asteroids often have complex geometries leading to irregular gravity fields. . . . .	7
2.1	Compute time and final error associated with simulating a spacecraft at 600km altitude orbiting for four hours real-time using Pines' formulation as a function of spherical harmonic degree. . . . .	16
3.1	PINN-GM Generation I . . . . .	26
3.2	Map of $\delta a$ at the Earth's Brillouin sphere . . . . .	31
3.3	Map of $\delta a$ of the Earth at a LEO altitude (approximately 420 km) . . . . .	32
3.4	Plot of MRSE( $\mathcal{A}$ ), MRSE( $\mathcal{F}$ ), and MRSE( $\mathcal{C}$ ) as a function of total parameters, $p$ , in used the spherical harmonic gravity model where $p = l(l + 1)$ . . . . .	33
3.5	Plot of MRSE as a function of total model parameters, $p$ . Solid lines represent the spherical harmonic representation. Dashed lines represent traditional neural networks. The lines with circle markers represent the physics-informed neural networks	36

3.6	Subset of the intermediate transformations and resulting basis function of the $N = 40$ traditional neural network. Each row corresponds with a single layer of the network in order of input (top) to output (bottom). The individual plots represent the normalized and dimensionless output of a particular node's activation function when evaluated across the Earth's Brillouin sphere . . . . .	38
3.7	Zoomed in final layer of Figure 3.6 representing the predicted cartesian components of the acceleration vectors plotted at the Brillouin sphere . . . . .	38
3.8	Gravity model of Earth using (a) the neural network representation and (b) the spherical harmonic representation given approximately the same number of free parameters ( $p \approx 3,000$ ) . . . . .	39
3.9	MRSE of $\mathcal{A}_m$ (top) and $\mathcal{F}_m$ (bottom) for the traditional (dashed) and physics-informed (solid) neural networks converted into the equivalent spherical harmonic degree as function of altitude. The blue histogram represents the training data distribution . . . . .	42
3.10	Training data distribution and equivalent spherical harmonic degree at varying altitudes for the $\beta = 3$ and $\beta = 10$ datasets. Solid lines represent the PINNs and dashed lines represent the traditional neural networks . . . . .	43
3.11	Total evaluation time to evaluate 10,000 random data using the various gravity models	45
3.12	Contrasting gravity field and acceleration distributions of the Earth and Moon . . .	47
3.13	Plot of MRSE as a function of total model parameters for the Moon. Dashed lines represent traditional neural networks. The lines with circle markers represent the physics-informed neural networks . . . . .	49
3.14	Gravity model of Moon using (a) the full $l = 1,000$ spherical harmonic model, (b) the PINN representation with $p = 3,040$ and (c) the low fidelity $l = 55 \Leftrightarrow p = 3080$ spherical harmonic representation . . . . .	50

3.15 MRSE of $\mathcal{A}_m$ (top) and $\mathcal{F}_m$ (bottom) for the traditional (dashed) and physics-informed (solid) neural networks converted into the equivalent spherical harmonic degree as function of altitude for the Moon. The blue histogram represents the training data distribution . . . . .	53
3.16 PINN-GM Generation II (1) . . . . .	54
3.17 Modified Network Structure . . . . .	58
3.18 Training Distributions Asteroid 433-Eros . . . . .	60
3.19 Acceleration residuals, $ \mathbf{a} - \hat{\mathbf{a}} / \mathbf{a}  \times 100$ , their moving averages, and training data distribution as a function of radius. Blue scatter plots correspond to error of the PINN trained on perfect measurements, and green corresponds to the PINN trained on noisy measurements. Dashed lines represent the error of the spherical harmonic models fit on the same data. The gray histogram represents the radial distribution of the training data. . . . .	63
3.20 Eros Data Distributions . . . . .	65
3.21 Acceleration residuals as a function of training distribution with $J_{ALC}$ demonstrate how additional physics constraints help desensitize the model to noise in the training data . . . . .	66
3.22 Model error <b>outside</b> the Brillouin sphere (exterior) as a function of amount of training data and noise added to the training data. . . . .	68
3.23 Model error <b>inside</b> the Brillouin sphere (interior) as a function of amount of training data and noise added to the training data. . . . .	69
3.24 Model error <b>at the surface</b> of the asteroid (surface) as a function of amount of training data and noise added to the training data. . . . .	70
3.25 PINN-GM Generation III with new modifications contained in dark gray boxes . . .	73

3.26 Different loss function change network performance at high and low altitudes. Blue points are the individual errors of the test data, the blue histogram is the distribution of training data, and the gray line is the average test error within a sliding window of 100 points. . . . .	77
3.27 The true potential $U$ (left) quickly decays to numerically unfavorable values whereas versus the proxy potential $U_{NN}$ (right) remains numerically well-conditioned between $[-1, 1]$ . . . . .	79
3.28 Effect of learning a potential proxy on the network inference error. . . . .	79
3.29 Top Row: Percent error of PINN-GM inside (0- $3R$ ) and outside ( $3R$ - $10R$ ) of the training domain. Gray vertical line is Brillouin radius. Green vertical line is maximum bounds of training data. Bottom Row: % Error of PINN-GM in XY plane. . . . .	82
3.30 Average acceleration percent error defined in Equation (3.33) of PINN-GM-I (top) and PINN-GM-III (bottom) trained on Earth. . . . .	85
3.31 Average acceleration percent error of PINN-II (top) and PINN-III (bottom) trained on Eros. . . . .	87
3.32 Asteroid with a mass heterogeneity (top) and the corresponding surface gravity field when constant density is assumed versus the true field (bottom left vs. bottom right respectively). . . . .	89
3.33 Constant density polyhedral acceleration error reported as $\mu \pm \sigma$ (max). . . . .	90
3.34 PINN-GM-III acceleration error reported as $\mu \pm \sigma$ (max). . . . .	90
3.35 Surface error of the constant density polyhedral gravity model and PINN-GM-III. . . . .	91
3.36 Propagation speed and error of the constant density polyhedral model and PINN-GM-III. . . . .	92
4.1 Trajectories taken <b>without</b> the Enhanced Safe Mode agent enabled. The darker shades correspond with earlier parts of the trajectory. . . . .	101
4.2 Average return as a function of clock time . . . . .	104

4.3	Trajectories taken with Enhanced Safe Mode agent enabled. . . . .	105
4.4	Success rates of the agents trained in different environments . . . . .	107
5.1	Outline of the former cartesian shooting method (top) and the novel orbital element shooting method (bottom). . . . .	110
5.2	Percent error of the acceleration vector for each gravity model . . . . .	116
5.3	Metrics comparing the orbits generated by the PINN-GM and the polyhedral gravity models. . . . .	117
5.4	Results from the Cartesian Shooting Method. Top row: Solution error in dimension- alized coordinates (tilde) and non-dimensionalized coordinates (no tilde). Middle row: Starting orbit (gray) and the discovered solutions (color) propagated for one orbit. Bottom row: Solutions found propagated for 10 orbits. . . . .	120
5.5	Results from the Orbital Elements Shooting Method. Top row: Solution error after one orbit in dimensionalized coordinates (tilde) and non-dimensionalized coordinates (no tilde). Middle row: Starting orbit (gray) and the discovered solutions (color) propagated for one orbit. Bottom row: Solutions found propagated for 10 orbits. . . . .	124
5.6	Orbit solutions found by cartesian shooting method (blue), unconstrained OE shoot- ing method (red), and constrained OE shooting method (green) compared to the original orbit (gray/black). . . . .	125
5.7	Average Percent Error of Solutions using the Cartesian LPE and OE LPE . . . . .	126
6.1	A random selection of NEAR-Shoemaker science orbits around 433-Eros . . . . .	131
6.2	Error of Regressed PINN-GM Gravity Model of Eros without Ejecta . . . . .	133
6.3	Error of Regressed PINN-GM Gravity Model of Eros with Ejecta Measurements (Magenta) . . . . .	134
6.4	PINN-GM Kalman Filter . . . . .	140
6.5	Heterogeneities of Asteroid . . . . .	143

6.6	Top Row: The true heterogeneous density asteroid gravity field accelerations assessed along cartesian planes from $[-2R, 2R]$ . Bottom Rows: Acceleration percent error for different gravity models capped at 10% error. . . . .	144
6.7	Spacecraft trajectory about the asteroid . . . . .	145
6.8	Hyperparameter search of the PINN-GM initialized with the constant density polyhedral model. . . . .	151
6.10	Hyperparameter search of the PINN-GM initialized with the point mass model. . . .	152
B.1	Model % error as a function of altitude using different combinations of pre-training and adaptive learning rate annealing algorithms. . . . .	176
E.1	GPGPU enhanced gravity algorithms for high-fidelity astrodynamics software enables support for high accuracy orbit propagation with lower runtimes. . . . .	184
E.2	Compute time and final error associated with simulating a spacecraft at 600km altitude orbiting for four hours real-time using Pines' formulation as a function of spherical harmonic degree. . . . .	185
F.1	Data Reduction Technique . . . . .	197
F.3	Video memory consumed as a function of $l_{\max}$ and number of spacecraft, $N_{S/C}$ . .	202
F.4	Data Flow of GPU Kernel Invocation . . . . .	206

## Chapter 1

### Introduction

Astro dynamics is the study of objects' motion through space. Be it for spacecraft or celestial bodies, the goal is to better understand the underlying forces acting on these systems and characterize or leverage the resulting motion. Despite the simplicity of the overarching theme, astrodynamics is full of a diverse array of academic topics, spanning trajectory optimization, formation flying, state estimation, control theory, attitude dynamics, and even remote sensing. Amidst the wide range of problems, one thing ties them all together: the force of gravity.

Consider the fact that nearly all problems in astrodynamics begin with equations of motion of the form:

$$\ddot{\mathbf{r}} = -\nabla U(\mathbf{r}) + \mathbf{a}_d \quad (1.1)$$

where  $U$  is a model of the gravitational potential, and  $\mathbf{a}_d$  are the remaining accelerations. The presentation of this differential equation alone highlights the significance of the gravitational force. In nearly all cases, gravity serves as the dominant force and the remaining perturbations are secondary. Yet despite the critical role that gravity plays in all astrodynamics problems, no universal model of the force exists. Instead, dynamicists are left to choose between a variety of analytic or numerical approximations, referred to as gravity models, each with their own corresponding advantages and drawbacks.

The simplest gravity model assumes that all objects can be represented as perfect point masses whose gravitational potential takes the form

$$U(\mathbf{r}) = -\frac{\mu}{r} \quad (1.2)$$

where  $\mu$  is the gravitational parameter of the body and  $r$  is the distance from the body. While this choice may be sufficient for first-order analyses and proof-of-concept work, it becomes considerably less reliable when transitioning into real mission scenarios. In these settings, higher-fidelity models are required, and dynamicists are forced to choose between a plethora of different models, including but not limited to spherical harmonics (2), ellipsoidal harmonics (3), mascons (4), polyhedral models (5), extreme learning machines (6), GeodesyNets (7), and so on.

Each of these models have tradeoffs, but the one thing they all recognize is that gravity fields are far more complex than a point mass approximation. Every celestial body comes with asymmetries and heterogeneous densities, or have surfaces peppered with craters, mountains, mineral deposits, and other geologic features that each contribute their own unique perturbation to the total gravity field. One need only look at the woefully textured surfaces of the Earth and Moon shown in Figure 1.1 as evidence. To treat these bodies as perfect spheres, or even oblate ellipsoids, is a gross over-simplification of the true system. If dynamicists and spacecraft neglect these higher-order gravitational features, the consequences can range from the steady drift away from reference trajectories, to missing entry corridors, or in worst cases, unexpectedly transitioning orbiters into intercepting probes. To avoid these scenarios, it is imperative that dynamicists have efficient and accurate models to represent these gravitational perturbations to high-accuracy.

There exists a long history and rich body of literature dedicated to the construction of high-fidelity gravity models. The majority of this literature explores ways to represent gravity fields analytically, leveraging clever mathematical tricks and basis functions to guarantee convergence. While these models offer compelling physical interpretations and desireable mathematical properties, many come with operational limitations, assumptions, or computational overhead. While these analytic models have provided meaningful advances to the gravity modeling community, their lingering drawbacks highlight a fundamental question facing the dynamics community: Are there ways to construct universal models of complex dynamical forces free of these shortcomings?

To answer this question, one must first acknowledge that dynamics models are notoriously difficult to construct. Ideally, these models should be accurate, compact, computationally efficient,

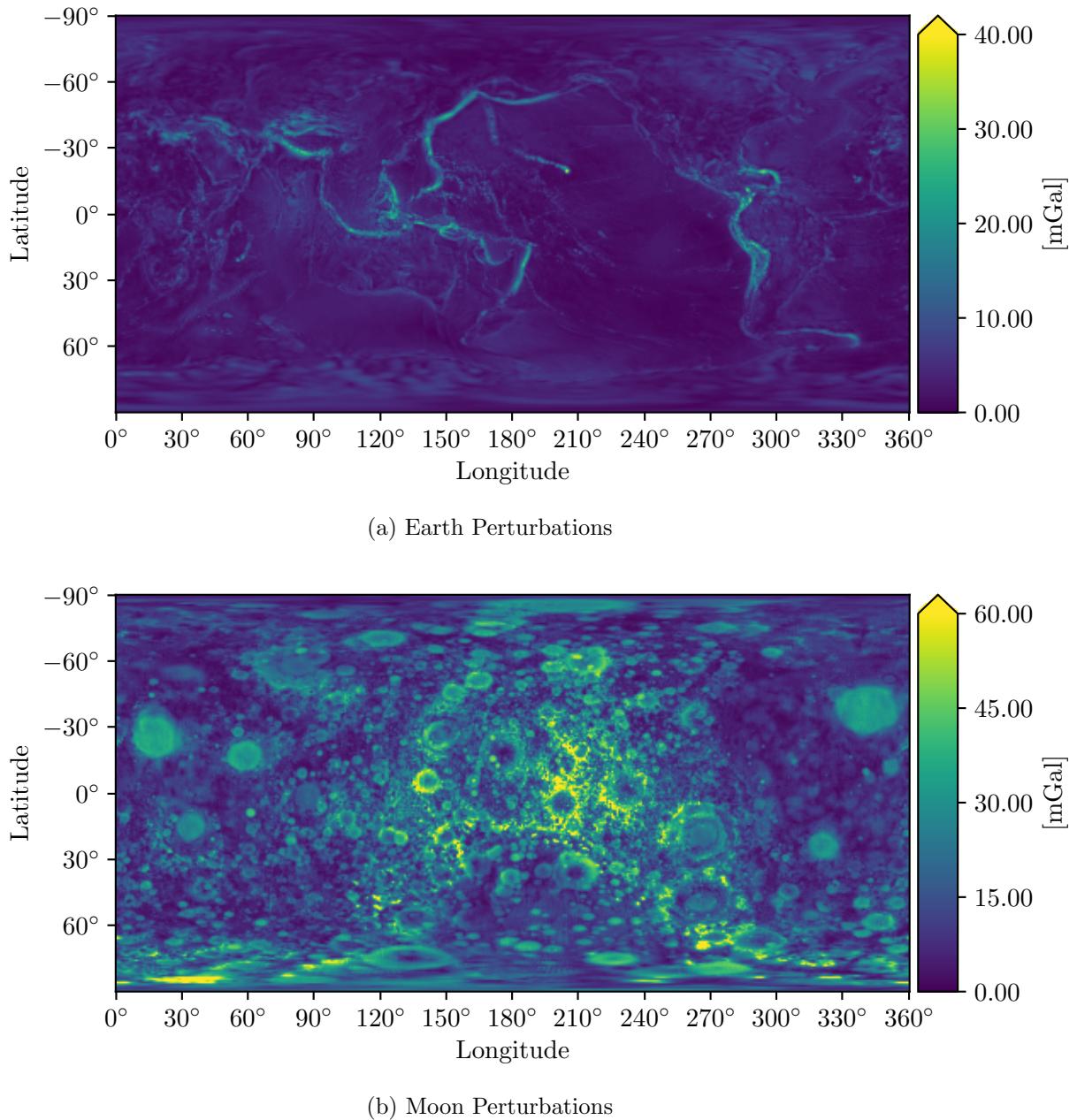


Figure 1.1: Gravitational Perturbations for Earth and Moon

and physically-compliant; however, these objectives are often in natural competition with one another. In many cases, high-accuracy models come with many parameters, large memory footprints, and high computational cost. Compact models come with less overhead, but typically lose accuracy or cut corners in their analytic rigor. It remains an open question if there even exist models for which these compromises do not need to be made. For the case of gravity field modeling, there has yet to be a model that is flexible enough to represent the most dominant perturbations to high-accuracy while maintaining small memory footprints and fast execution times. If such a model did exist, the applications would be far reaching — expanding on-board capabilities for spacecraft guidance, navigation, and control, improving ground-based simulation fidelity for trajectory design and mission planning, as well as unexplored improvements for planetary scientists and geophysicists who use these models for other purposes.

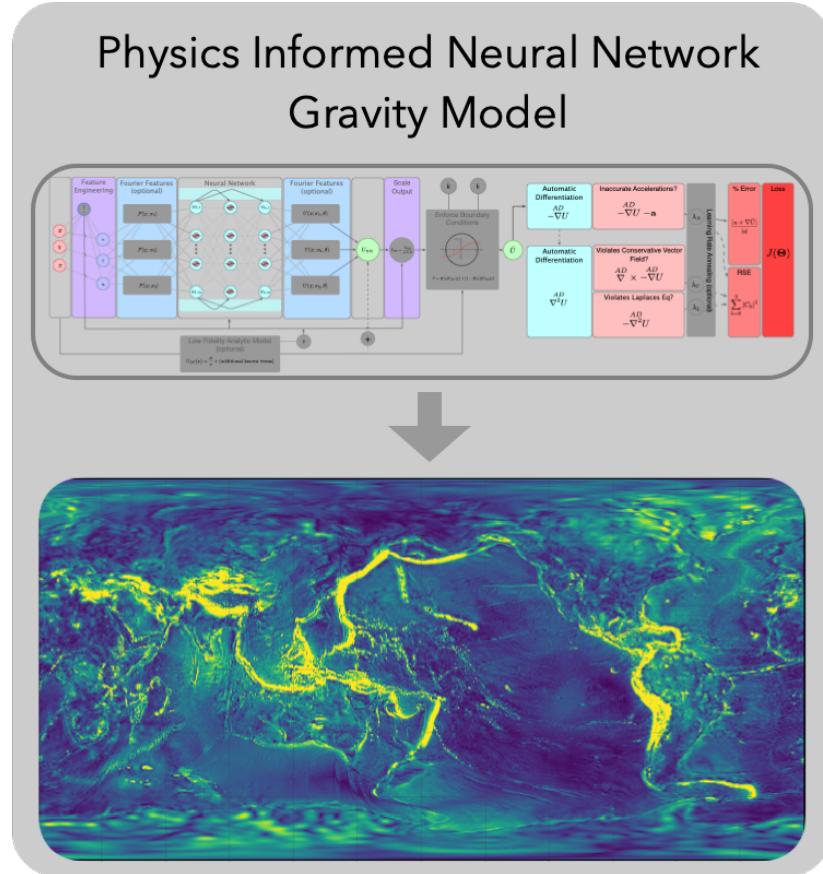


Figure 1.2: Using Scientific Machine Learning models to solve the gravity modeling problem.

This thesis aims to take a step towards the construction of a universal, high-fidelity gravity model. By shifting the focus away from purely analytic approaches, this work instead introduces a class of gravity model that leverages novel techniques within the emerging field of Scientific Machine Learning, or SciML. SciML is subset of artificial intelligence that seeks to construct high-fidelity models of complex dynamical systems in a data-driven manner (8). There exists a wide variety of candidate tools and function approximators in this branch machine learning, but one of the most popular is the Physics-Informed Neural Network (PINN).

PINNs are neural networks specifically designed to learn solutions to differential equations (9). While this can technically be accomplished with traditional neural networks — sampling values of the dynamical system at various positions and points in time and regressing a neural network to that data — PINNs provide a more satisfying and powerful framework. Specifically, PINNs introduce an augmented loss function which simultaneously penalizes mis-modeling of the training data while also including additional terms which penalize the model when it violates differential equations governing the true system. This simple change yields a surprising amount of utility, granting dynamicists a powerful and flexible way to represent complex dynamical phenomena while ensuring that the learned solution is intrinsically complaint with the underlying physics. Not only does this help quell concerns about the learned solution validity, but these changes also produce sizable improvements in the model’s accuracy, generalizability, and data efficiency.

This thesis explores how Physics-Informed Neural Networks can be leveraged to solve the gravity modeling problem. Rather than prescribing analytic basis functions which come with their own unique challenges, the following chapters demonstrate how PINNs can learn more compact representations of the gravitational potential while maintaining the desirable physical properties of their analytic predecessors. After multiple iterations of development, the PINN-GM now produces gravity models that offer high-accuracy, fast execution times, data efficiency, global validity, and exact differentiability. Taken together, these attributes make the PINN-GM well-suited to assist astrodynamacists in a variety of applications including reinforcement learning, periodic orbit discovery, and orbit determination.

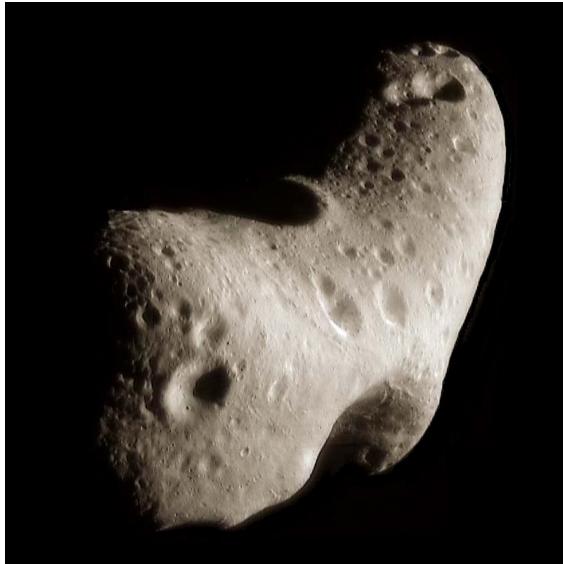
## 1.1 Significance

As discussed, gravity field modeling is among the most fundamental problems in astrodynamics. In nearly all space-based dynamical systems, gravity plays a key role in the guidance, navigation, and control of the object in question. Consider the most common problems in astrodynamics: the two body problem, three body problem, formation flying, rendezvous, proximity operations and docking, trajectory design, etc. For each of these topics, the gravity model can entirely change the nature of the design space and solutions discovered. Despite this, dynamicists often rely on the point mass gravity model or low-fidelity alternatives, as the transition to a higher-fidelity model risks the problem reaching an level of untenable complexity.

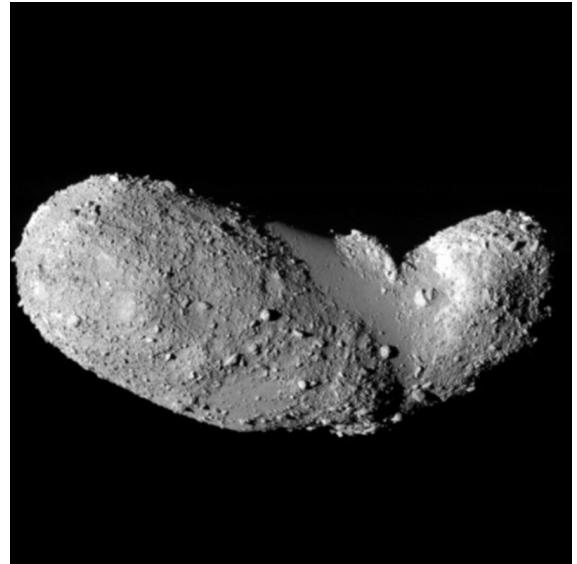
The use of low-fidelity gravity models, however, is becoming unsustainable. As near-Earth, cis-lunar, and deep space missions continue to increase in complexity, there grows a simultaneous need for more powerful dynamics models to assist spacecraft in achieving their goals. The recent decades spent exploring asteroids and comets, or small-bodies, provides a compelling example of this phenomenon. For the earliest missions in the 1980s like Giotto (10) and Galileo (11), the goal was simply to flyby the small-body of study (Halley's comet and the asteroid Gaspra respectively). These missions did not require extremely precise gravity models, only models accurate enough to get the spacecraft in the vicinity of the object to capture images before continuing on into deep space. Later, these missions began requiring greater precision, as designers sought to place spacecraft in orbit around these bodies (NEAR Shoemaker, 1996 (12)) or to conduct multiple flybys (Deep Space 1, 1998 (13)). Once it was demonstrated that spacecraft could enter orbit, priority was then shifted to collecting material (Stardust, 1999 (14)), purposefully impacting the object (Deep Impact, 2005 (15)), even landing on the body's surface (Rosetta, 2004 (16); Hayabusa, 2003 (17)). These increasingly ambitious goals continue to present day today with missions like Hayabusa2 (18) and OSIRIS-REx (19) attempting sample return for the asteroids Ryugu and Bennu respectively.

It is not difficult to extrapolate these trends and recognize that missions will continue to increase in complexity for the foreseeable future, and dynamicists will require increasingly precise

dynamics models to accommodate these ambitions. Of these forces, accurately capturing the effects of the asteroid's gravity field is particularly important. These bodies can have exotic geometries and density profiles (see Figure 1.3), which, when left unaccounted for, can place spacecraft on trajectories that may collide with the body. While avoiding catastrophic failure is the highest priority, high-fidelity gravity models can also be valuable for precision touchdowns. Consider the recent OSIRIS-REx mission whose flagship Touch-And-Go phase had originally been designed to maneuver the spacecraft into 25 meter landing zone. After discovering that the asteroid was covered in boulders, the largest landing site available extended only a mere eight meters (20). In these moments, precise knowledge of the gravity field is essential to ensure mission safety during touchdown.



(a) 433-Eros (21)



(b) 25413-Itokawa (22)

Figure 1.3: Asteroids often have complex geometries leading to irregular gravity fields.

Beyond gravity models' application to small-body missions, there are also other academic communities that leverage high-fidelity representations of gravity fields. Specifically, gravity modeling has played a key role for problems in geophysics, ocean and climate science, and planetary science. Gravity models are commonly used to better understand water resources across the globe (23),

variations in sea level (24), and glacial melt (25). These studies play a critical role in predicting droughts (26) and floods (27). Similarly, gravity field models are used to better understand the interior of the Earth (28) and the motion of the tectonic plates (29). The latter is critical for applications in GNSS, which in-turn help determine elevations, reference frames, and coordinate systems used by other disciplines (30).

This work proposes a novel gravity model which seeks to provide utility to each of these fields; however the lessons learned through the development of this model extend beyond the space science community. Specifically, this work fundamentally explores how machine learning can be used to construct high-fidelity dynamics models; investigating how to systemically discover better basis functions and coordinate descriptions of complex dynamical phenomena. In principle, the lessons gathered here can benefit communities which seek to explore data-driven alternatives to otherwise cumbersome analytic models. The focus on gravity modeling offers a productive test bench for the scientific machine learning community at large, as these systems are often time-invariant and have well-studied pre-existing models. From this, SciML practitioners can use this problem to benchmark new machine learning techniques without introducing additional overhead of more complex dynamical systems.

## 1.2 History of Gravity Models

The search for high-fidelity gravity models predates the age of spacecraft. While the fundamentals of gravity were first characterized by Newton in the late 17th century, it was only in the early 1900s that scientists began constructing higher-fidelity models. Rapp provides a detailed history of these developments for Earth’s gravity models in his review paper (31) which is summarized here for convenience. Spanning from 1901 to 1958, scientists first recognized that the Earth’s gravity field deviates from that of a point mass approximation. To account for this, they first crafted gravity models that were the superposition of gravity variations in latitude (zonal perturbations), and later extended these models to include variations in longitude — the precursor to the spherical harmonic gravity model used today. To estimate the parameters of these models,

geodesists averaged ground-based gravity measurements in bins of fixed degree (30x30, 10x10, 5x5, and 1x1) and regressed the corresponding coefficients.

With the launch of Sputnik and other early satellites, geodesists like Kaula made first use of satellite data to estimate select coefficient in the 1960s. Likewise institutions like the Smithsonian Astrophysical Observatory, the Applied Physics Laboratory, and the Naval Weapons Laboratory computed their own gravity models from satellite data referred to as SE, APL, and NWL models respectively. Through the 1960s and into the 1970s, additional techniques were proposed to blend terrestrial and satellite data achieving higher degree models with greater precision. In the 1970s, Goddard Space Flight Center introduced their own gravity model GEM, which began with GEM-I in 1972 (32) but continued to be updated until GEM-10C which reached degree 180 in 1978. In the 1980s, the introduction of satellite laser tracking provided higher quality and greater quantity of satellite data to update these models. Missions like Lageos and TOPEX/POSEIDON data were used with the later generations of GEM to form models like GEM-L2 (33) and GEM-T1 (34) respectively. Similarly, researchers at the Center for Space Research at the University of Texas at Austin began developing their TEG models which utilized tracking data from multiple satellites. In the 1990s, GSFC and UT Austin's Center for Space Research joined forces and produced the JGM models 1-3 (35; 36). Most recently, the Earth Gravity Models (EGM) from the National Geospatial-Intelligence Agency provide some of the highest-fidelity models. Among the most well-known is EGM96 (37) which reached a spherical harmonic solution spanning degree 360. In 2008, NGA introduce the EGM2008 model which reached degree 2190 and is currently the highest fidelity spherical harmonic model for Earth publicly available (38).

The study of the Moon's gravity field followed a similar trend, albeit at a slower rate due to limited data. Among the first gravity models of the lunar field came as a result of the Apollo program. Studying the motion of the spacecraft which orbited the moon, researchers realized that the spacecraft orbital motion was perturbed over certain region of the lunar surface. These perturbations were later characterized as mass concentrations in the lunar surface, dubbed mascons, which could effectively be modeled as the superposition of multiple point mass elements embedded

on, or just below, the surface of the Moon (39). Analyses of this mascons continued through the late 20th century with missions like the Clementine mission (40), Lunar Reconnaissance Orbiter (41), and the Gravity Recovery and Interior Laboratory (GRAIL) (42) continuing to take measurements of the Moon's topology and gravity field to help construct both higher fidelity mascon models as well as spherical harmonic models.

Beyond the study of large planetary bodies and moons, small-body exploration has also fueled considerable work for gravity field modeling. As discussed, small-body gravity fields can be notoriously difficult to represent, particularly given their irregular geometry and unknown density profiles. Originally researchers continued to model these fields with spherical harmonics; however, as noted by Brillouin in 1933, the spherical harmonic model begins to diverge when the spacecraft enter the sphere which bounds all mass elements (43). This divergence make the spherical harmonic model particularly risky for proximity or landing operations.

In 1997, Werner and Scheeres introduced the polyhedral gravity model which offers a non-diverging solution to represent the gravity fields of these bodies (5). If a polyhedron shape model of the small-body in question exists, then there exist a way to compute the potential of that body analytically under the assumption of constant density. Such model offers a solution that remains stable down to the surface, making it a popular and favorable choice for many small body missions. Notably, this model does come with relatively high computational cost, and the constant density assumption has been shown to be invalid in some cases as demonstrated for the asteroid Bennu from recent OSIRIS-REx data (44).

Recently dynamicists have begun exploring alternative ways to represent the gravity fields of these exotic bodies, relying less on analytic approximations of the system, and instead on more data-driven models. These efforts have included the use of Gaussian processes (45), extreme learning machines (6), and neural networks (46). These models each offer computational advantages over some of their analytic counterparts, but they also lack the analytic niceties of past approaches. To date, none of these models demonstrate that they satisfy important differential properties, and relatively little work has gone into carefully validating these models. In many cases, the learned

solution is only tested in local regions or within the bounds of the training data. These models can also come with large memory footprints using tens or hundreds of thousands of parameters, and many require large quantities of training data to regress which can be difficult to acquire in practice.

### 1.3 Gravity Missions

Multiple missions have been flown with the goal of refining gravity field models for the Earth and Moon. For Earth, the first of these missions was the Challenging Minisatellite Payload (CHAMP), launched in 2000 by GFZ Potsdam. CHAMP was intended to conduct a variety of atmospheric and ionospheric research in addition to better characterize Earth's gravity and magnetic fields over the span of five years (47). Following CHAMP, was the Gravity Recovery and Climate Experiment (GRACE) launched in 2002 which leveraged two twin satellites and their relative distance to produce more precise measurements of the gravity field (48). Using this distance the mission was able to recover detailed measurements of water distribution (49), glacial ice melt (25), and contribute to state of the art knowledge on climate change (23). Following GRACE was the Gravity Field and Steady-State Ocean Circulation Explorer (GOCE) which used a highly sensitive gradiometer to estimate temporal variations the gravity field (50). After GOCE and the end of GRACE, GRACE Follow On (GRACE-FO) was launched which continued the efforts of the original GRACE mission to acquire time varying gravity field estimates (51). Between these missions, the Gravity Recovery and Interior Laboratory (GRAIL) was also launched in 2011 which sought to measure the gravity field of the Moon using a similar mission concept to GRACE (52).

### 1.4 Goal of Thesis

This thesis presents the first comprehensive document that discusses how to design, train, and validate physics-informed neural network gravity models (PINN-GM). Explicitly, this thesis answers questions including: How does the design of the PINN-GM impact its performance? How can input features be constructed maximize the parametric capacity of the model while avoiding

numerical instability? Are there advantages to incorporating preexisting models into the system, and if so, how can this be accomplished? What role does the quality and distribution of training data play in the model performance? These questions, among others, are investigated through multiple generations of development outlined in the coming chapters.

Moreover, this thesis also provides a number of new ways to characterize gravity model performance. Rather than randomly sampling test points and computing the corresponding acceleration error, this thesis introduces additional measures to provide a more complete depiction of model performance. These measures include evaluating total integration error during trajectory propagation, testing model accuracy across a range of altitudes within and beyond the training data, measuring model compactness as a function accuracy versus total parameters, and testing the models with varying data distributions and quality. These metrics become useful in not only evaluating the PINN-GM performance with respect to past models, but also provide a framework for guiding the development of future PINN-GM generations.

Through these investigations, the PINN-GM is shown to be the first data-driven model that learns powerful new basis functions of complex gravitational systems while enforcing differential constraints on the learned solution. These constraints provide the PINN-GM greater modeling accuracy and data-efficiency than their analytic and data-driven predecessors while maintaining small memory footprints and fast runtimes. Taken together, these attributes grant the PINN-GM wide utility for the astrodynamics community at large as shown through three case study applications. The first of these applications demonstrates how the PINN-GM's high-accuracy and fast evaluation speeds can improve the quality of autonomous spacecraft agents trained through reinforcement learning. The second application uses the differentiability of PINN-GM to discover candidate periodic orbits around complex gravitational bodies in arbitrary element sets. The third and final application explores how the PINN-GMs can be directly integrated into orbit determination pipelines and filters to estimate complex gravity fields in both an online and offline fashion in less time than current approaches.

In summary, the physics-informed neural network gravity model is a novel way to construct

high-fidelity models of the gravity fields for both large and small celestial bodies in a manner that is free of many pitfalls of former approaches. This thesis provides an overview of the generational improvements of this model and highlights candidate applications within the broader astrodynamics community.

## Chapter 2

### Review of the Gravity Modeling Problem

#### 2.1 Overview

There exist two families of gravity models: analytic and numerical. Analytic models take a variety of forms, but at their core share the fact that they are derived from first principles, offering solutions to fundamental differential equations like Laplace's equation. In contrast, numerical models are constructed in a data-driven manner — regressing functions capable of interpolating between measured data. Each family of model has its own corresponding advantages and drawbacks, and the choice of which model to use is often dictated by the application. The following chapter details the available gravity models and their respective pros and cons.

#### 2.2 Analytic Models

##### 2.2.1 Spherical Harmonics

Early in the 1900s, it was proposed that spherical harmonic basis functions could be superimposed to produce a high-fidelity estimate of the gravitational potential (43).

$$U(r) = \frac{\mu}{r} \sum_{l=0}^l \sum_{m=0}^l \left(\frac{R}{r}\right)^l P_{l,m}(\sin \phi) [C_{l,m} \cos(m\lambda) + S_{l,m} \sin(m\lambda)] \quad (2.1)$$

Equation (2.1) is referred to as the spherical harmonic gravity model where  $r$  is the radius to the field point,  $\mu$  is the gravitational parameter of the body,  $R$  is the circumscribing radius of the body,  $l$  is the degree of the model,  $m$  is the order of the model,  $C_{l,m}$  and  $S_{l,m}$  are the Stokes coefficients,  $\lambda$  is the longitude,  $\phi$  is the latitude, and  $P_{l,m}$  are the associated Legendre polynomials (2).

The spherical harmonic gravity model is the primary model of choice to represent the gravity fields of large planetary bodies like the Earth (53), Moon (54), and Mars (55). One of the most compelling advantages of the spherical harmonic gravity model is how it can compactly capture the large gravitational perturbation of planetary oblateness. Because the Earth and these other large bodies rotate about their axes, a centrifugal acceleration is produced which flattens the body from a sphere into an oblate ellipsoid (56). This flattening or excess of mass near the equator is referred to as planetary oblateness, and its presence has sizable effects on spacecraft trajectories. It is therefore important that this perturbation is accurately captured in a gravity model. The spherical harmonic gravity model makes this simple, needing only one harmonic in the expansion to capture this oblateness ( $C_{2,0}$ , or its alternative form  $J_2 = -C_{2,0}$ ).

While spherical harmonics are effective at representing planetary oblateness and other global scale perturbations, they struggle to model the remaining and more discrete gravitational perturbations like mountain ranges, tectonic plate boundaries, and craters. Discontinuities are notoriously difficult to represent using periodic basis functions, often requiring hundreds of thousands of harmonics being superimposed to overcome the 3D equivalent of Gibbs phenomena (57). Not only must these harmonics be superimposed, they must also be regressed, and the high-frequency signals become increasingly difficult to detect due to the  $(R/r)^l$  term in Equation (2.1). Taken together, these conditions results in memory inefficient models that become challenging to keep on-board spacecraft. In addition, these high-fidelity models come with a high computational cost. High-degree spherical harmonics models require reevaluating all of the associated Legendre polynomials at each field point. This calculation is recursive and therefore no trivial way to parallelize these computations exist. This leads to an unavoidable  $\mathcal{O}(n^2)$  computational complexity which can severely limit both ground-based simulation and algorithms flown on-board (58) — see Appendix E and F for implementation details.

Beyond the computational inefficiencies, spherical harmonics also has operational limitations. The derivation of this model requires that all mass elements exist within a sphere of fixed radius (the Brillouin radius). If dynamicists require a potential or acceleration estimate within this sphere, the

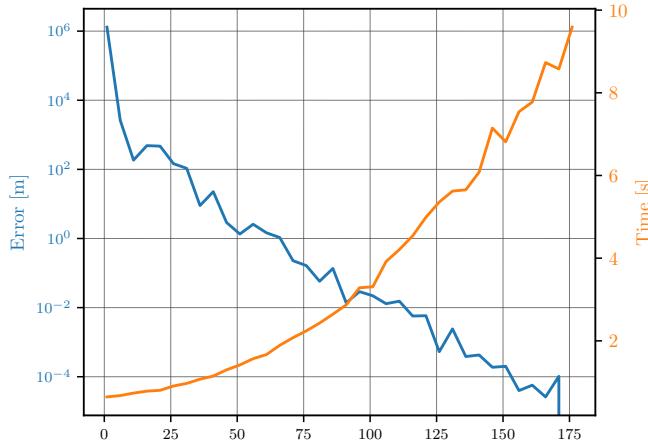


Figure 2.1: Compute time and final error associated with simulating a spacecraft at 600km altitude orbiting for four hours real-time using Pines' formulation as a function of spherical harmonic degree.

model can diverge thanks to the  $(R/r)^l$  term in the expansion. While such effects are negligible for near-spherical planets or moons, they can become problematic in small-body settings. For asteroids or comets that exhibit highly non-spherical geometries like Eros or Itokawa, these effects can lead to large errors in the predicted accelerations and potentially risk the safety of a spacecraft (5).

### 2.2.2 Ellipsoidal Harmonics

The ellipsoidal harmonic model follows a similar approach to that of spherical harmonics but uses ellipsoidal harmonic basis functions instead (3). At a glance the potential can be represented as

$$V = \sum_{n=0}^{\infty} \sum_{p=1}^{2n+1} \alpha_n^p E_n^p(\lambda_1) E_n^p(\lambda_2) E_n^p(\lambda_3) \quad (2.2)$$

where  $\alpha_n^p$  are constants,  $E_n^p$  are Lamé functions of the first kind,  $\lambda_i$  are the ellipsoidal coordinates,  $n$  is the degree, and  $p$  are the eigenvalues. This choice allows for a tighter bounding ellipsoid about the body than spherical harmonics, minimizing the region in which the model could potentially diverge. Despite this, the divergence remains a possibility inside the bounding ellipsoid and the model still suffers from the same challenges in representing discontinuity with periodic basis functions.

### 2.2.3 Interior Spherical Harmonics / Bessel Models

The interior spherical harmonic model inverts the classical spherical harmonic formulation and can model a local region whose boundary intersects only one point on the surface of the body (59). Explicitly, rather than computing the potential by using the law of cosine and factoring out  $1/r$  through

$$U^e(r, \phi, \lambda) = G \int_M \frac{1}{r} \frac{1}{\sqrt{1 - 2(\frac{\rho}{r}) \cos S + \frac{\rho^2}{r^2}}} dm \quad (2.3)$$

the interior spherical harmonic model instead factors out  $\rho$

$$U^i(r, \phi, \lambda) = G \int_M \frac{1}{\rho} \frac{1}{\sqrt{1 - 2(\frac{r}{\rho}) \cos S + \frac{r^2}{\rho^2}}} dm \quad (2.4)$$

This model maintains stability down to that single point making it valuable for precise landing operations; however, the solution becomes invalid on any other point on the surface and outside of the corresponding local sphere. The interior spherical Bessel gravity model expands on this theme, using bessel functions rather than spherical harmonics, and is able to achieve a wider region of validity. However, this comes at the expense of cumbersome analytics and retains some of the challenges of using harmonic basis functions to capture discontinuity (60).

### 2.2.4 Mascons

Mascon gravity models distribute a set of point mass elements within a body whose sum can form a global representation of the gravity field (61). Unfortunately, the accelerations can grow inaccurate at field points near the individual mascons (4). Hybrid mascon models offer a slightly more robust alternative to the pure mascon approach by representing each mascon with a low fidelity spherical harmonic model, but this incorporates additional complexity in regression and remains prone to both challenges of the mascon and spherical harmonic models (62).

### 2.2.5 Polyhedral Model

The polyhedral gravity model provides an alternative to the spherical harmonic model in these settings, offering a solution that maintains validity down to the surface of any body regardless of

shape. This stability makes the polyhedral model the primary gravity model for exploration about small-bodies such as asteroids and comets. If a shape model of the body is available (a collection of triangular facets and vertices which captures the geometry of the object), then the associated gravitational potential of that geometry can be computed under the assumption of constant density through:

$$\nabla U = -G\sigma \sum_{e \in \text{edges}} \mathbf{E}_e \cdot \mathbf{r}_e \cdot L_e + G\sigma \sum_{f \in \text{facets}} \mathbf{F}_f \cdot \mathbf{r}_f \cdot \omega_f \quad (2.5)$$

where  $G$  is the gravitational constant,  $\sigma$  is the density of the body,  $\mathbf{E}_e$  is an edge dyad,  $\mathbf{r}_e$  is the position vector between the center of the edge and the field point,  $L_e$  is an analog to the potential contribution by the edge,  $\mathbf{F}_f$  is the face normal dyad,  $\mathbf{r}_f$  is the distance between the face normal and the field point, and  $\omega_f$  is an analog to the potential contribution by the face (5).

The polyhedral gravity model circumvents the numerical divergence within the Brillouin sphere, making it an extremely popular choice for small-body operations. Multiple works use this model to characterize the dynamical environment around small-bodies, and design corresponding trajectories (63; 64; 65; 66). Moreover, this model remains under active development, with researchers developing models which incorporate uncertainty in the shape (67) and more efficient evaluation strategies (68). However, this model does with some disadvantages. Foremost, this gravity model can be computationally expensive. High-resolution shape models have hundreds of thousands of facets and vertices which must be individually looped over to compute the acceleration at a single field point. When computing many accelerations or propagating trajectories, this computational burden can lead to excessively long runtimes. While this model can take advantage of parallelization more easily than spherical harmonics, such computational capabilities are not available on-board spacecraft — intrinsically limiting this model to ground-based simulation.

Additionally the polyhedral gravity model assumes that researchers know a density profile for the body in question. Most commonly the density is assumed to be constant, but literature shows that such assumption is not necessarily valid (69; 44). In addition, the polyhedral gravity model requires that a shape model of the body already exists. While these models can be acquired

in-situ, the process is non-trivial and time-consuming in practice (70; 71; 72).

### 2.2.6 Heterogeneous Polyhedral

An alternative formulation of the polyhedral model proposes treating each facet of the shape model as a tetrahedron connected to the center of the shape. The tetrahedra can be sliced into multiple parts, and different densities assigned to each slice based on some candidate density distribution (73). This allows for an analytic approximation of a heterogeneous density body, but continues to suffer from assumptions made about the candidate density distribution.

## 2.3 Machine Learning Models

As an alternative to the analytic models, recent efforts explore the use of machine learning to learn representations of complex gravity fields in a data-driven manner. In principle, these machine learning models can learn high-accuracy gravity fields without making assumptions about the body in question, while maintaining low computational cost.

### 2.3.1 Neural Networks

Neural networks a series of learned, non-linear transformations that map data from an input space to a desired output space by minimizing a prescribed loss function such as mean squared error:

$$\mathcal{J}(\boldsymbol{\Theta}) = \frac{1}{N_f} \sum_{i=1}^{N_f} |y_i - \hat{y}(x_i|\boldsymbol{\Theta})_i|^2 \quad (2.6)$$

where  $y_i$  is the true output,  $\hat{y}_i(x_i|\boldsymbol{\Theta})$  is predicted output by the artificial neural network given the vector of trainable parameters  $\boldsymbol{\Theta}$  which includes the weights,  $\mathbf{w}$ , and biases,  $\mathbf{b}$ , of the network, and  $N_f$  is the total number of points used to train the network.

Commonly these networks are constructed as a series of densely connected hidden layers with  $N$  nodes per layer expressed as:

$$h_i^{(k)} = \sigma \left( w_{ij}^{(k-1)} h_j^{(k-1)} + b_i \right) \quad k \in \{1, \dots, k_{\max}\} \quad (2.7)$$

where  $h^{(k)}$  is the  $k$ -th hidden layer,  $i$  is the node in the layer,  $w_{ij}$  are the weights connecting the hidden layers,  $b_i$  are the biases attached to the nodes in the layer, and  $\sigma$  is the non-linear transformation (typically sigmoid, hyperbolic tangent, or rectified linear unit). Note that  $h^{(0)} = x$ , and  $h^{(k_{\max})} = \hat{y}$ .

Neural network are trained by iteratively updating the weights and biases of the model to minimize Eq. (2.6) such that:

$$\mathbf{w}^* = \arg \min_{\mathbf{w} \in \Theta} (\mathcal{J}(\mathbf{w})); \quad \mathbf{b}^* = \arg \min_{\mathbf{b} \in \Theta} (\mathcal{J}(\mathbf{b})) \quad (2.8)$$

which can be solved using a gradient descent algorithm like Adam or SGD (74; 75)

$$\boldsymbol{\Theta}^{m+1} = \boldsymbol{\Theta}^m - \eta \nabla_{\boldsymbol{\Theta}^m} \mathcal{J}^m(\boldsymbol{\Theta}), \quad (2.9)$$

where  $\eta$  is the learning rate and  $m$  is the training iteration, and the gradient is taken using automatic differentiation.

Neural networks demonstrate considerable potential in their ability to regress highly accurate models of complex phenomena across a wide variety of scientific domains. Despite this much criticism exists regarding these models, as their overparameterization can lend itself to overfitting leading to unreliable models outside of the bounds of the original trained data. Moreover, these models are often described as black-box regressors which are difficult to interpret or provide any analytic guarantees. Despite their drawbacks, neural networks have been proposed as a candidate solution to the gravity modeling problem. By training on position and acceleration data, literature has shown that neural networks are capable of representing complex gravitational phenomena (46).

## Automatic Differentiation

Automatic differentiation is method to compute the exact derivative of an algorithm with respect to any input. This is done by constructing a computational graph and using either a forward

or backward form of chain rule such as:

$$\begin{aligned}\frac{\partial y}{\partial x} &= \frac{\partial y}{\partial w_{n-1}} \frac{\partial w_{n-1}}{\partial x} \\ &= \frac{\partial y}{\partial w_{n-1}} \left( \frac{\partial w_{n-1}}{\partial w_{n-2}} \frac{\partial w_{n-2}}{\partial x} \right) \\ &= \dots\end{aligned}$$

where  $x$  is the input to some arbitrary algorithm,  $w_i$  are the sequence of intermediate calculations performed to produce the final output,  $y$ .

Because all algorithms are constructed from elementary functions with known derivatives, the partials of each intermediate expressions can always be computed alongside the original calculation. This property ensures that partial of the output  $y$  with respect to any input  $x$  can be computed automatically. Automatic differentiation is best known for its application within deep learning, where it commonly applied in the stochastic gradient descent algorithms used to train neural networks (76). Specifically, Equation 2.9 is where automatic differentiation is applied. The gradient of the prescribed loss function with respect to the weights and biases of a network can be computed automatically and applied in the descent to move the network's parameters in a direction that will minimize that loss function.

### 2.3.2 Extreme Learning Machines

Seeing as neural networks often receive warranted criticism due to their overparameterization, alternative models have been developed to alleviate some of these concerns. Specifically, an alternative class of machine learning model has been proposed called Extreme Learning Machines, or ELMs. ELMs are single layer neural networks which are fit by randomly initializing the weights from the inputs to the hidden layer, and then solving for the weights to the output layer using a least squares approach to minimize some quadratic cost function such as:

$$L(\theta) = \frac{1}{N} \sum_{i=0}^N |\hat{\mathbf{y}}_i(\mathbf{x}_i|\theta) - \mathbf{y}_i|^2 \quad (2.10)$$

where  $\hat{y}_i(\mathbf{x}_i|\theta)$  is the machine learning model prediction at input  $\mathbf{x}_i$  with trainable model parameters  $\theta$  (77). The ELM is advantageous over other methods because it can be trained in a single iteration using least squares as opposed to neural networks which require many iterations of stochastic gradient descent and backpropagation. ELMs of sufficiently large width have also been proven as universal function approximators (78). ELMs were recently shown to successfully model the gravity field of the asteroid Itokawa in for an landing scenario, and have demonstrated fast evaluation speeds with relatively high accuracy (6). That said, despite the model's simplicity, ELMs do introduce different challenges. Namely, ELMs require taking large matrix inversions to perform the least squares fit, intrinsically limiting the amount of data that can be used to update the model at any given point in time.

### 2.3.3 Gaussian Processes

In addition to ELMs, Gaussian processes have also been proposed as candidate solutions to the gravity modeling problem (45). Gaussian processes are fit by specifying a prior distribution over functions, and updating that prior based on observed data. This requires the user to specify some kernel function which provides a metric of similarity between data, and computing a covariance matrix between all data points using that function. Once the covariance matrix is computed, it can be inverted and used evaluate the mean and covariance of the learned function at a test point.

Using Gaussian processes for gravity modeling is advantageous because it provides a probabilistic estimate of the uncertainty in the model's prediction; however the model does not scale well to large datasets. Specifically, the Gaussian process is characterized by a covariance matrix built from the training data. This covariance matrix scales as  $\mathcal{O}(n^2)$  where  $n$  is the size of the training data. This scaling makes it computationally challenging for Gaussian processes to leverage large quantities of data, as the size of the covariance matrix and complexity of corresponding inversion grows rapidly with small sets of training data. As an example, in Reference (45) a Gaussian process was regressed over a mere 3,600 data points resulting in a model covariance matrix of over 12,960,000 parameters. Because this model size grows rapidly, researchers must carefully

select which training data to leverage, as these models inevitably cannot model the entire testing domain. Aside from this fact, the study does demonstrate that these models, once fit, can achieve fast evaluation times and relatively high-accuracy predictions near the training data.

#### 2.3.4 GeodesyNet

In 2021, Reference (7) proposed the use of neural density fields to learn density maps for small bodies. This research takes inspiration from recent machine learning developments in Neural Radiance Fields (NeRFs) which are used to construct 3D shape models from relatively sparse image data (79). Once trained, the density predictions can be integrated numerically to produce corresponding gravitational potentials and accelerations. This work demonstrated promising results – achieving competitive acceleration accuracies and the ability to account for heterogeneous densities; however, the models are relatively large (in excess of 90,000 parameters), the numerical integration necessary to compute accelerations is computationally expensive, and the models continue to require large training datasets.

#### 2.3.5 Physics-Informed Neural Networks

One of the disadvantages of traditional machine learning models is that they lack an analytic foundation. There are not guarantees that the model intrinsically satisfies some a differential equation or will perform accurately over the entire problem domain. When neural networks, extreme learning machines, or GeodesyNets are trained to represent a gravity field, researchers cannot ensure that these models are intrinsically compliant with the underlying physics like Laplace’s equation or conservative vector field properties.

In 2019, Raissi et. al. recognized this problem and suggested that neural networks models do not need to be agnostic of the physics which govern the function they are attempting to represent (9). Instead, Raissi argues that networks **can** be trained specifically to ensure that learned representations obey underlying differential equation with the introduction of the original Physics-Informed Neural Network (PINN). PINNs inject differential equations and boundary conditions

into the cost function of traditional neural networks, using automatic differentiation to ensure that these solution learned naturally satisfy these constraints.

As an example, consider the following arbitrary differential equation:

$$f''(x) + f'(x) + f(x) = 0, \quad (2.11)$$

Assume there exist measurements of  $x$  as well as the corresponding values of  $f(x)$ . A traditional neural network can use these observations as training data to learn a mapping from  $x \rightarrow \hat{f}(x\|\Theta)$  by minimizing the cost function  $J(x\|\Theta) = \|f(x) - \hat{f}(x\|\Theta)\|^2$ . The risk of training the network with this particular loss function is that the network does not know that the mapping,  $\hat{f}(x\|\Theta)$  must also satisfy Eq. (2.11). PINNs change this paradigm by inserting the original differential equation into the cost function:

$$J(\Theta) = \frac{1}{N_f} \left( \sum_{i=1}^{N_f} \left\| f(x_i) - \hat{f}(x_i\|\Theta) \right\|^2 + \left\| \hat{f}''(x_i\|\Theta) + \hat{f}'(x_i\|\Theta) + \hat{f}(x_i\|\Theta) \right\|^2 \right), \quad (2.12)$$

where the derivatives of the network  $\hat{f}(x_i\|\Theta)$  are taken with automatic differentiation. This cost function, while similar to Eq. (2.6), not only penalizes erroneous values of  $\hat{f}(x\|\Theta)$ , but also penalizes when the learned function violates the differential form of the problem. This extra term serves as a form of regularization in the training process which can lead to improved solutions that conveniently also satisfy important physics properties. Much work has gone into the study and utility of these PINNs. Recent work has shown that these models often offer advantages in data efficiency and modeling accuracy over their traditional counterparts.

## Chapter 3

### Physics-Informed Neural Network Gravity Models

This thesis explores the use of the Physics-Informed Neural Network (PINN) to regress a form of the gravitational potential. The PINN gravity model has undergone multiple generations of development — each stage introducing additional sophistication to improve model accuracy, robustness, and data efficiency. This chapter highlights the chronological developments of each generation and the corresponding performance at each stage.

#### 3.1 Generation I

Past gravity modeling efforts have demonstrated that machine learning can be used to regress models of a gravity field (6; 45). This can be accomplished by collecting estimates of spacecraft position,  $\mathbf{x}$ , and acceleration,  $\mathbf{a}$ , and using this as a training data set to fit a neural network or extreme learning machine by minimizing the following cost function

$$J_{00}(\Theta) = \frac{1}{N_f} \sum_{i=1}^{N_f} |\mathbf{a}_i - f(\mathbf{x}_i|\Theta)|^2 \quad (3.1)$$

where  $f(\mathbf{x}|\Theta)$  is the learned mapping of the acceleration vector parameterized by the weights and biases of the network  $\Theta$  (i.e.  $f(\mathbf{x}|\Theta) = \hat{\mathbf{a}}$ ).

These former efforts, while important contributions to the literature, are not especially satisfying as they are agnostic of the physics of the system they are trying to represent. Unlike the spherical harmonic or polyhedral gravity models, the solutions learned by the neural networks had no knowledge that the gravitational accelerations it is representing are the byproduct of a more

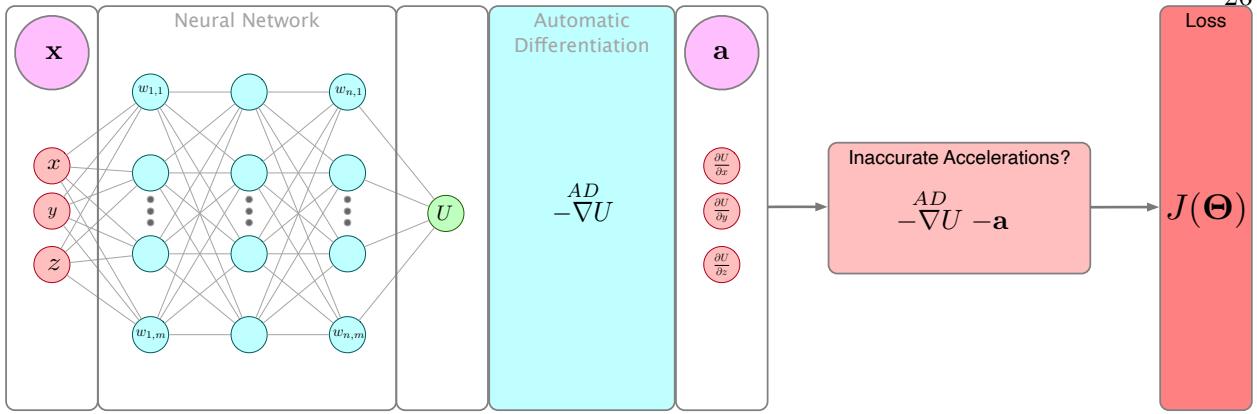


Figure 3.1: PINN-GM Generation I

fundamental scalar potential function. Moreover, these solutions are not aware that the gravitational force is conservative, nor that the behavior at high-altitudes needs to tend towards zero. Until recently, researchers did not have a way to include this knowledge in the construction of these machine learning models. However, in 2019, Raissi et. al. introduced the physics-informed neural network (PINN) which offered the first glance on how this could be accomplished (9).

As discussed in Chapter 2, PINNs are machine learning models that leverage an augmented loss function. These loss functions not only penalize modeling inaccuracies between the predicted and true value, as is the case in traditional machine learning, but they also penalize violations of the underlying differential equations of the system they are attempting to model. In the case of the gravity field modeling problem, physics implies that gravitational accelerations are really byproducts of a more fundamental scalar potential through:

$$\mathbf{a} = -\nabla U \quad (3.2)$$

PINNs can leverage this differential equation directly within the cost function:

$$J_A(\Theta) = \frac{1}{N_f} \sum_{i=1}^{N_f} \left\| \mathbf{a}_i + \overset{\text{AD}}{\nabla} \hat{U}(r_i \|\Theta) \right\|^2, \quad (3.3)$$

where  $\mathbf{a}_i$  is the estimated acceleration at position  $\mathbf{r}_i$ ,  $\hat{U}$  is the learned potential function, and  $\overset{\text{AD}}{\nabla}$  is the gradient operator applied via automatic differentiation.

Equation 3.3 is a physics-informed loss function. Rather than simply minimizing the difference between a predicted acceleration and the true acceleration, Eq. 3.3 enforces that the predicted acceleration must be related to the gravitational potential. In this way, the neural network is not directly representing the acceleration vector, but instead is learning the scalar potential function,  $\hat{U}$ , and enforcing that its negative gradient matches the measured acceleration.

One of the advantages of this approach is the improved data efficiency. When training a traditional neural network learns the acceleration vector directly, the three components of vector are effectively treated as distinct and individual features for the network to learn. In fact, these components are, by construction, orthogonal such that there is no observable relationship between them. As such, a traditional neural network must learn a basis set that can solve three distinct problems simultaneously. The PINN gravity model, in contrast, only needs to learn a scalar potential function; leveraging all three of the acceleration components to regress a single feature. As a consequence, PINNs can make more efficient use of the same amount of training data, both decreasing training times and helping alleviate dynamicists from needing to collect large datasets.

It should be noted that additional physics-informed constraints can be added to the loss function. For example, the cost function could also include a penalty for:

- (1) Mis-modeling the potential function itself,  $U(r) - \hat{U}(r\|\Theta) = 0$
- (2) Violating the boundary condition  $U(\mathbf{r}) = 0$  as  $\|\mathbf{r}\| \rightarrow \infty$
- (3) Not satisfying conservative vector field properties like  $\nabla^2 \hat{U} = 0$  and  $\nabla \times \hat{U} = 0$ .

This first generation PINN-GM, or PINN-GM-I, purposefully omits these additional physics-informed constraints for two reasons. Optional constraint (1) is omitted because the domain of the potential is order of magnitudes larger than the domain of the corresponding accelerations. These differing scales pose numerical challenges for learning, and can introduce excessively large gradients during backpropagation. Additionally, this choice reflects the more common data circumstances where a high-fidelity potential of the body in question does not exist and therefore cannot be used.

Constraints (2) and (3) are omitted because, in the early stages of this work, they tended to drive the PINN solution for  $\hat{U}$  towards zero or simply slow convergence. This is attributed to the multi-objective optimization that gradient descent algorithms struggle to accommodate (80). Explicitly, the constraints of a conservative vector field can be satisfied if the network identifies  $U = 0$  as the optimal solution. This can force the network away from the non-zero solution guided by the constraint  $\mathbf{a} + \nabla U = 0$ . In theory, these competing objectives can be better balanced through dynamically adjusting learning rates for each of the objective costs (81). This strategy is explored in later generations of the PINN-GM.

Finally, it is worth stressing that the presented PINN gravity model formulation **does not require** a preexisting gravity model to produce training data. PINNs can be trained in-situ, relying on estimates of accelerations either via finite differencing of relative velocities or more advanced filtering techniques. This chapter chooses to focus primarily on the gravity modeling problem, not the gravity estimation problem. As such, existing high-fidelity models are used as training data for this chapter, but Chapter 6 will provide a more in-depth discussion of how PINNs can be trained in-situ without a pre-existing gravity model.

### 3.1.1 Network Architecture and Hyperparameters

All networks trained in the upcoming experiments share an architecture of eight densely connected hidden layers with  $N$  nodes per layer. The choice of eight hidden layers offered a desirable balance between network capacity and reasonable training times. As discussed, the networks' training data consists of position and acceleration pairs generated from preexisting high-fidelity gravity models. For all networks, the position data are preprocessed via a min-max transformation fit to each component of  $\mathbf{r}$  into the bounds of  $[-1, 1]$ . This normalization ensures that the inputs exist in the favorable, non-linear part of the network's activation function, as well as assist with numerical stability during training. In addition, the acceleration data output is also normalized via a min-max transformation. Note, that for the traditional neural networks, the min-max transformation can be applied for each individual component of the acceleration vector. For PINNs,

however, all components must be scaled simultaneously to the dimensions are physically compliant for the learned potential function.

For PINN-GM-I, the hyperparameters that have the largest effect on performance are the learning rate and mini-batch size. If the batch size is too small, the gradient descent algorithm can move in directions other than the local minimum resulting in longer training times. Unfortunately small batches are often unavoidable when the available GPU does not have sufficient VRAM to store the entire dataset on the device. In this case, smaller and more cautious learning rates should be used to ensure the optimizer does not quickly lead the network to a suboptimal minimum that later becomes challenging to escape. Conversely, if a large batch size can be used, larger learning rates are encouraged as they yield shorter training durations that are more likely to descend in the direction of the true gradient of the cost function.

Independent of initial learning rate magnitude, it can be advantageous to slowly decrease the learning rate towards the end of training to prevent the weights from oscillating above the cost function's local minimum (82). To achieve this, an exponential decay is applied to the learning rate:

$$\eta_i = \begin{cases} \eta_0 & i < i_0 \\ \eta_0 * \text{pow}(\alpha, -\frac{i-i_0}{\sigma}) & i \geq i_0 \end{cases}, \quad (3.4)$$

where  $\eta_i$  is the learning rate at epoch  $i$ ,  $i_0$  is the reference epoch after which the decay begins,  $\sigma$  is the scale factor,  $\alpha$  is the decay rate, and  $\eta_0$  is the initial learning rate.

An additional critical hyperparameter is the activation function. When training a PINN, it is important that the activation function selected has a sufficiently high order of continuity. If the activation function does not have smooth high-order derivatives, and if gradients of the network are taken using automatic differentiation to enforce the physics constraints, the cost function will no longer be well-behaved for gradient descent. As such, it is recommended to avoid using the popular rectified linear unit (ReLU) or leaky ReLU and instead opt for functions with infinite orders of continuity like hyperbolic tangent or the gaussian exponential linear unit (GELU). The remaining default hyperparameters for the PINN-GM-I are shown in Table 3.1 and any custom

hyperparameters are shown in Table 3.2.

Each network is trained for 100,000 epochs using the Adam optimizer in Tensorflow 2.4<sup>1</sup> on a NVIDIA RTX 2060 graphics card. This training duration ensured the validation loss consistently plateaued, indicating that learning had terminated. All network weights are initialized using the Xavier uniform initialization scheme detailed in Reference (83).

Table 3.1: Shared hyperparameters for the traditional and physics-informed neural networks trained in this work

Hyperparameter	Value	Hyperparameter	Value
Learning Rate, $\eta_0$	0.005	Initializer	Glorot Uniform
Patience Epoch Start, $i_0$	25000	Epochs	100000
LR Scheduler Decay Rate, $\alpha$	0.5	$\mathbf{x}$ transform	MinMax
Scale Factor, $\sigma$	25000	Activation	GELU
Optimizer	Adam	Number of Layers	8

### 3.1.2 Spherical Harmonic Performance

Spherical harmonics are the defacto gravity model for large planetary bodies. The most accurate spherical harmonic gravity model for the Earth is the Earth Gravitational Model 2008 (EGM-2008) which contains spherical harmonics that extend to degree and order of 2,160 — totalling over 4,000,000 parameters (53). While spherical harmonics are especially convenient at representing global scale gravitational features with low degree harmonics, their initial efficiency does not persist at higher degrees.

To demonstrate, consider the prominent gravitational features that remain after removing the point mass, planetary oblateness, and planetary obliquity accelerations (i.e. the accelerations produced by harmonics above degree and order 2). To view these features, a variable  $\delta a$  is introduced:

$$\delta a(\mathbf{r}) = \| -\nabla U_{\text{Truth}}^{\text{SH}}(\mathbf{r}) - (-\nabla U_2^{\text{SH}}(\mathbf{r})) \| , \quad (3.5)$$

---

<sup>1</sup> <https://www.tensorflow.org/>

Table 3.2: Unique hyperparameters for the traditional and physics-informed neural networks trained in this work

Network Type	Nodes Per Layer ( $N$ )	Model Parameters	Batch Size	$\mathbf{a}$ Transform
Traditional	10	843	262144	MinMax
Traditional	20	3083	262144	MinMax
Traditional	40	11763	262144	MinMax
Traditional	80	45923	262144	MinMax
PINN	10	820	262144	Uniform
PINN	20	3040	262144	Uniform
PINN	40	11680	262144	Uniform
PINN	80	45760	131072	Uniform

where  $U_{\text{Truth}}^{\text{SH}}$  is constructed using EGM-2008 expanded to degree  $l = 1,000$ ,  $U_2^{\text{SH}}$  is that same model expanded to only degree and order 2, and the gradient of the potential is taken using Pines' algorithm to avoid singularities at the poles (84). Figure 3.2 shows these perturbations computed at the Brillouin sphere of Earth. This figure verifies that the perturbations are discontinuous features in the crust like mountain ranges such as the Himalayas and Andes, tectonic subduction zones as best seen in the Pacific, and hotspots scattered across the globe. These findings align with intuition as the accelerations are directly proportional to the gradient of the potential ( $\mathbf{a} = -\nabla U$ ) and large displacements in landmass generate large changes in the potential.

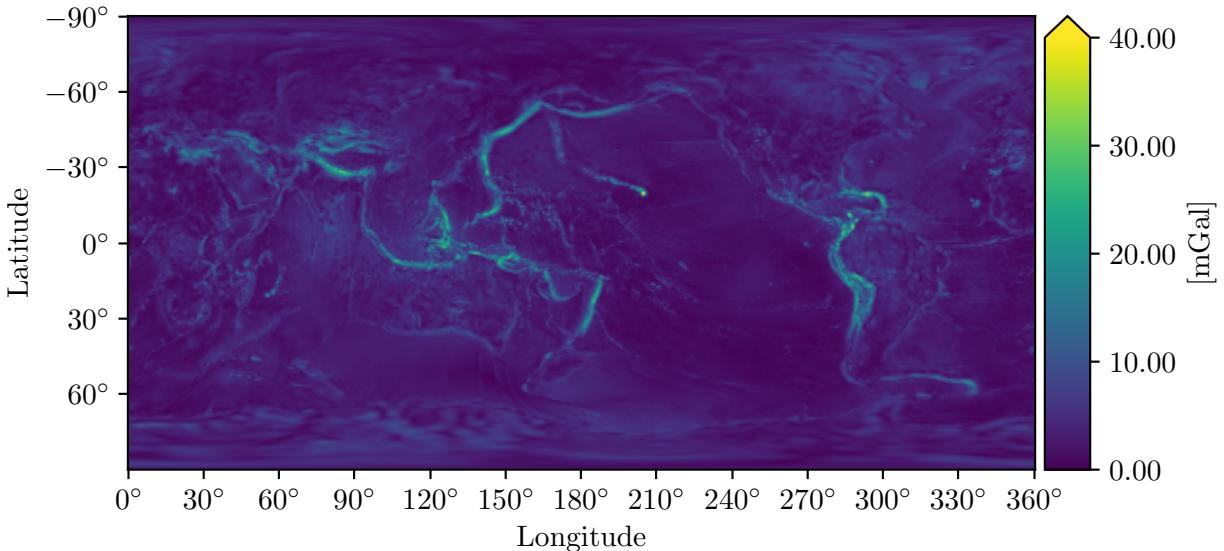


Figure 3.2: Map of  $\delta a$  at the Earth's Brillouin sphere

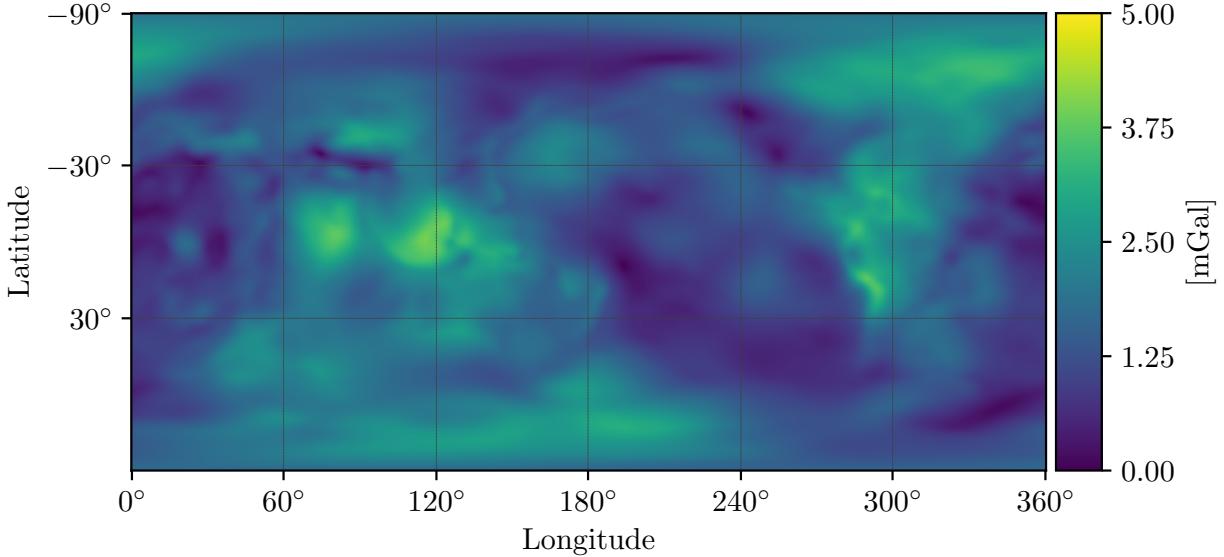


Figure 3.3: Map of  $\delta a$  of the Earth at a LEO altitude (approximately 420 km)

These perturbations are especially important features to capture with a gravity model, second only to planetary oblateness. It is therefore reasonable to question how efficiently do spherical harmonic models represent these perturbations. Specifically, when low-degree spherical harmonic models are used, how much error exists in these important parts of the gravity field? To investigate, a mean root-squared error (MRSE) metric is introduced:

$$\text{MRSE}(\mathcal{A}) = \frac{1}{N} \sum_{i=1}^{N_f} \delta a_p(\mathbf{r}_i) \quad \mathbf{r}_i \in \mathcal{A}, \quad (3.6)$$

where  $\mathcal{A}$  is the set of positions for which the gravity field is evaluated, i.e. field points,  $N_f$  is the total number of field points in set  $\mathcal{A}$ , and  $\delta a_p$  is a generalization of Eq. (3.5) used to measure the corresponding acceleration error of a low-fidelity spherical harmonic model through:

$$\delta a_p(\mathbf{r}_i) = \| -\nabla U_{\text{Truth}}^{\text{SH}}(\mathbf{r}_i) + \nabla U_p^{\text{model}}(\mathbf{r}_i) \|, \quad (3.7)$$

where  $p$  represents the maximum number of parameters / coefficients used in the gravity field model being evaluated.

To characterize the error of low-fidelity spherical harmonic models, the MRSE metric is applied to three datasets:  $\mathcal{A}$ ,  $\mathcal{F}$ , and  $\mathcal{C}$ . The  $\mathcal{A}$  dataset includes  $N_f = 250,000$  field points distributed

in a Fibonacci grid at the Brillouin sphere of Earth. The Fibonacci grid is chosen to ensure a near isotropic distribution of data about the Brillouin sphere, thereby avoiding the clustering of data at the poles were the set to be distributed uniformly in latitude and longitude (85). The second set,  $\mathcal{F}$ , is a subset of  $\mathcal{A}$  that only includes data points within the dominant gravitational perturbations. Specifically,  $\mathcal{F}$  is generated by selecting the field points within  $\mathcal{A}$  whose acceleration exceed 2 standard deviation of the mean acceleration such that  $\mathcal{F} : \{\|\delta\mathbf{a}(\mathbf{r}_i) - \bar{\delta}\mathbf{a}\| > 2\sigma_{\mathbf{a}}(\mathcal{A})\}$ . The third and final set,  $\mathcal{C}$  is the compliment of set  $\mathcal{F}$  ( $\mathcal{C} : \mathcal{A}/\mathcal{F}$ ) representing the background of the gravity field.

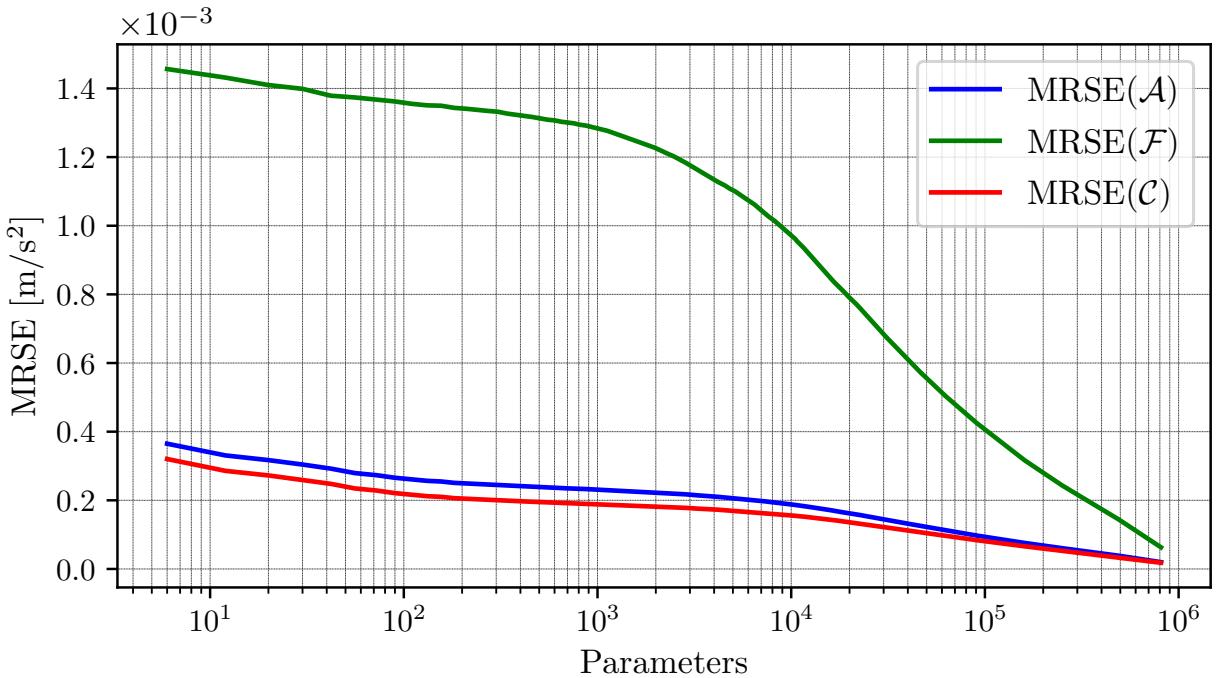


Figure 3.4: Plot of  $\text{MRSE}(\mathcal{A})$ ,  $\text{MRSE}(\mathcal{F})$ , and  $\text{MRSE}(\mathcal{C})$  as a function of total parameters,  $p$ , in used the spherical harmonic gravity model where  $p = l(l + 1)$

The MRSE metric is applied to sets  $\mathcal{A}$ ,  $\mathcal{F}$ , and  $\mathcal{C}$  and presented in Figure 3.4 as a function of total parameters used in the spherical harmonic representation. This figure quantifies the relationship between spherical harmonic model size and model accuracy. The blue MRSE curve for  $\mathcal{A}$  tells a deceptively simple story: the more parameters used in a spherical harmonic model, the better the average accuracy across the Brillouin sphere. However, if the data is separated between the dynamically significant features  $\mathcal{F}$  versus the less significant features  $\mathcal{C}$ , a much more interest-

ing result emerges. Specifically, the error within the dominant gravitational features,  $\mathcal{F}$ , is over 10 times greater than its background counterpart in the low-parameter regime. In fact, it takes nearly 10,000 parameters, or a spherical harmonic model of degree 110 or larger, before that gap in performance begins to decrease at an appreciable rate.

The discrepancy between the modeling error of  $\mathcal{A}$  and  $\mathcal{F}$  showcases how the spherical harmonic representation struggles to capture perturbations in order of dynamical significance. While the initial efficiency of representing Earth's oblateness is undeniable, the convenience does not extend into the next most important perturbations. This is because spherical harmonics prioritize fitting prescribed geometries onto a system in which those geometries are not naturally present. As a consequence, the spherical harmonic model must superimpose many high-order frequencies / harmonics before capturing these perturbations.

To a dynamicist, this implies that the next most important perturbations beyond  $J_2$  require, at the minimum, a spherical harmonic model that exceeds degree  $l = 110$  if they aspire to incorporate the dynamic effects of the Earth's high-order perturbations into their application. In some circumstances this may not be a problem. When a sufficiently high-fidelity model exists and the researcher is not computationally limited, spherical harmonics will eventually converge even over the discontinuous features. On-board spacecraft, however, computational resources may be limited or a high-fidelity spherical harmonic model may not exist for the body in question. In these conditions, operations over short time scales and near large surface features could be negatively affected by spherical harmonics inability to efficiently represent these perturbations.

Moreover, the results shown in Figure 3.4 motivate why this research turns to learned neural network gravity representations as an alternative to spherical harmonics. The spherical harmonics basis is inherently limited in resolving discontinuous perturbations while retaining a compact model size. The perturbations present on the Earth require small-wavelength harmonics which are only present in high-degree expansions. Neural networks, in contrast, do not have prescribed basis functions and corresponding characteristic wavelengths. There is no inherent minimum of 10,000 parameters needed to represent a specific mountain range or other discontinuity. Rather, the

neural networks learn a convenient basis that represents the most important perturbations of the field independent of their geometry or scale. In principle, neural network gravity models can therefore yield more compact representations that achieve comparable, if not greater, accuracy than traditional spherical harmonics.

### 3.1.3 Representational Compactness

The following experiment aims to determine if the PINN-GM-I can produce equally accurate gravity models using fewer parameters than its spherical harmonic counterpart. For this experiment, model parameters,  $p$ , refer to the total number of coefficients used in a spherical harmonic model or the number of trainable weights and biases of a network model.

The experiment begins by training both traditional and physics-informed neural networks on 5,000,000 position / acceleration vector pairs which are drawn randomly from a uniform distribution in altitude (0-420 km), latitude, and longitude. A total of eight networks are trained, each with varying model capacity as presented in Table 3.2. Once trained, the MRSE metric (Eq. (3.6)) is used to evaluate the performance of each network using the same Fibonacci grid data as was used to generate Figure 3.4 —  $\mathbf{r}_i \in \{\mathcal{A}, \mathcal{F}, \mathcal{C}\}$ . The Fibonacci test samples provide a entirely decoupled dataset from the training data to ensure a fair evaluation of the network performance. The MRSE for the networks is juxtaposed with the MRSE of the spherical harmonic model and presented in Figure 3.5.

Figure 3.5 demonstrates that there exists a wide range in which the neural networks generate more accurate models of the gravity field using fewer parameters compared to their spherical harmonic counterparts. This range spans between 1,000 to 50,000 parameters, or between spherical harmonic degree  $l = 30$  and  $l = 225$ . This is best seen in the  $\mathcal{F}$  dataset, where the PINNs can produce models of equal accuracy using an order of magnitude fewer parameters than spherical harmonics. Traditional neural networks, also produce more compact representations, albeit less than the PINN model of similar size.

That said, there are conditions where the PINNs' compactness advantage becomes less appar-

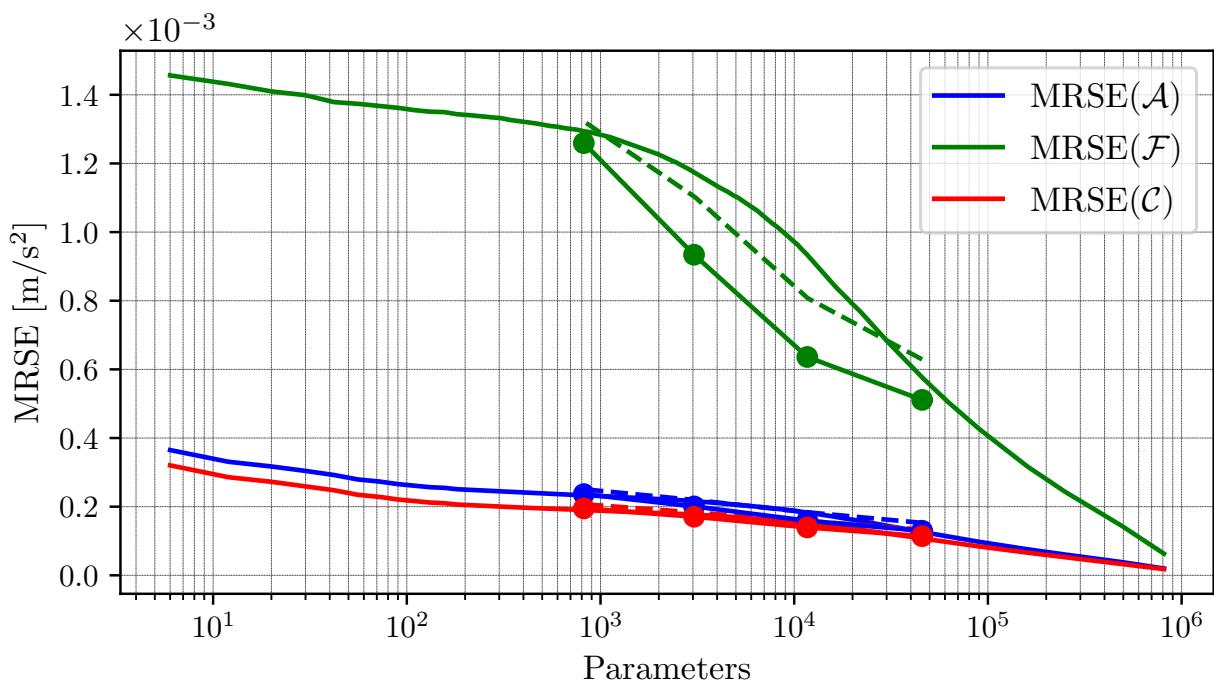


Figure 3.5: Plot of MRSE as a function of total model parameters,  $p$ . Solid lines represent the spherical harmonic representation. Dashed lines represent traditional neural networks. The lines with circle markers represent the physics-informed neural networks

ent. Specifically, note the neural networks with the smallest and largest capacities ( $N = \{10, 80\}$ ). These particular networks have a less pronounced compactness advantage which is assumed to be the result of two factors. In the case of  $N = 10$ , the networks do not have a sufficiently high modeling capacity to represent the non-linear perturbations of the gravity field. With a mere  $p \approx 1,000$  trainable parameters, these small networks do not have the parametric flexibility necessary to capture the discontinuous high-order perturbations. The  $N = 80$  case, in contrast, has such a large network capacity that it manages to model these features quite well — so well in-fact that the network begins to overfit to the training data and suffer when tested on new data. This can be remedied with additional training data, or more clever model design as will be shown in later sections.

The compactness advantage presented here suggest that the machine learning models are able to learn a set of basis functions that are substantially more efficient at representing the high-order perturbations of Earth's gravity field. In the case of the traditional neural networks, these basis functions and the intermediate non-linear transformations used to generate them can be directly plotted as shown in Figure 3.6.

Figure 3.6 can be interpreted as follows: The top row represents the first layer in the network — i.e the normalized cartesian position vectors inputs. Because these images are generated at the Brillouin sphere, the  $x$  and  $y$  components of the position vectors grow small at the poles and large at the equator, whereas the  $z$  component grows linearly from the south to north pole. The second row represents weighted, linear combinations of the first three inputs that are then passed through the gaussian exponential linear unit (GELU) activation function. This row corresponds to the outputs of the first hidden layer of the network. This non-linear transformation is repeated for each intermediate hidden layer until the 9th row of the figure (the 8th hidden layer) which represents the learned basis functions. These penultimate functions are then combined linearly without the GELU transformation to produce the three predicted acceleration components at the Brillouin sphere of Earth (Figure 3.7).

Notably, as the inputs propagate deeper into the network, the corresponding outputs of the

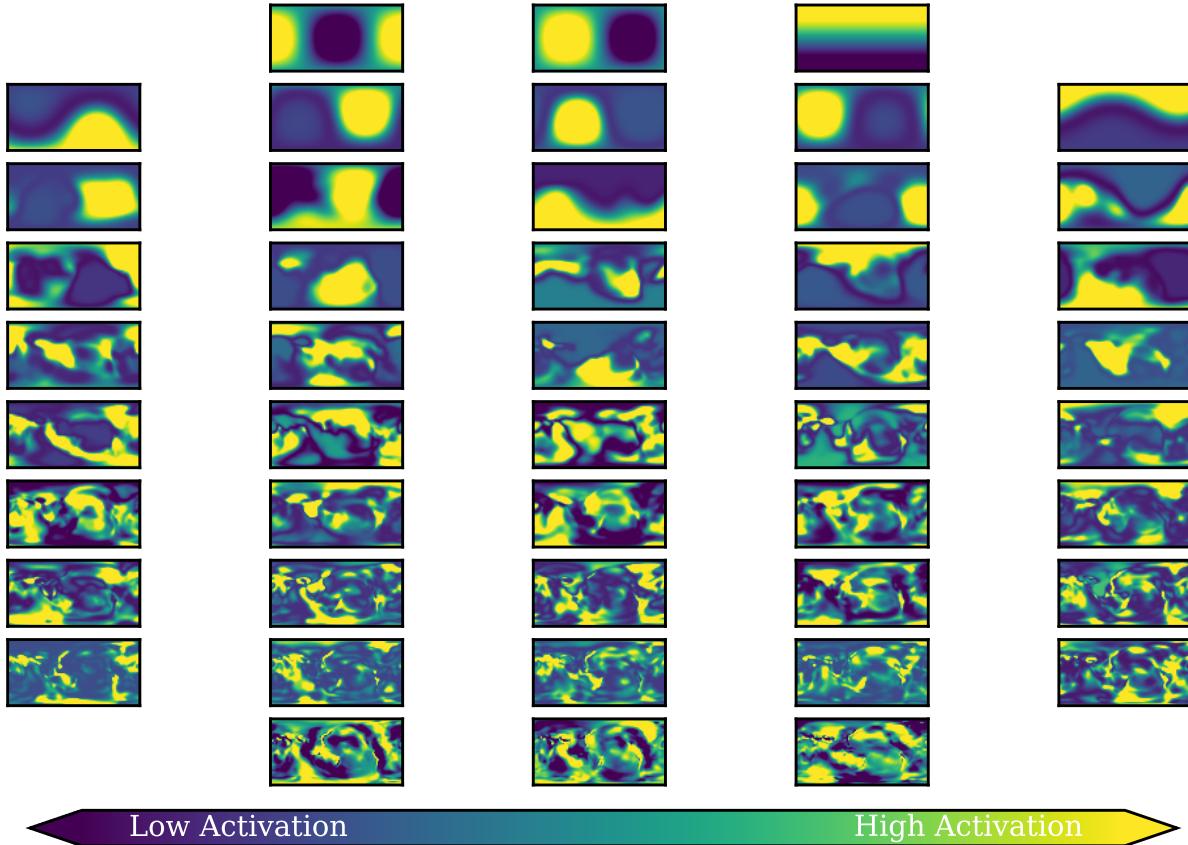


Figure 3.6: Subset of the intermediate transformations and resulting basis function of the  $N = 40$  traditional neural network. Each row corresponds with a single layer of the network in order of input (top) to output (bottom). The individual plots represent the normalized and dimensionless output of a particular node's activation function when evaluated across the Earth's Brillouin sphere

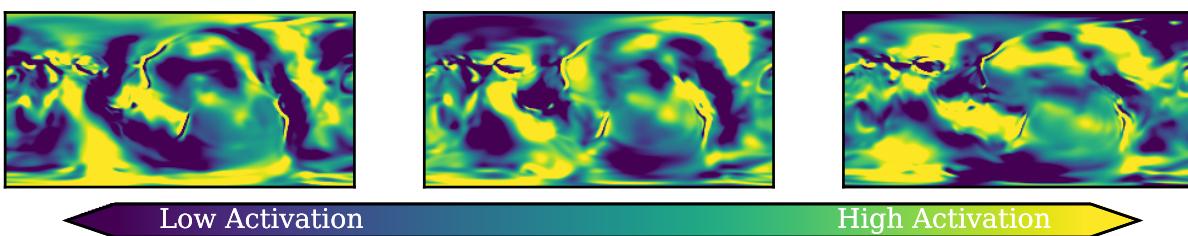


Figure 3.7: Zoomed in final layer of Figure 3.6 representing the predicted cartesian components of the acceleration vectors plotted at the Brillouin sphere

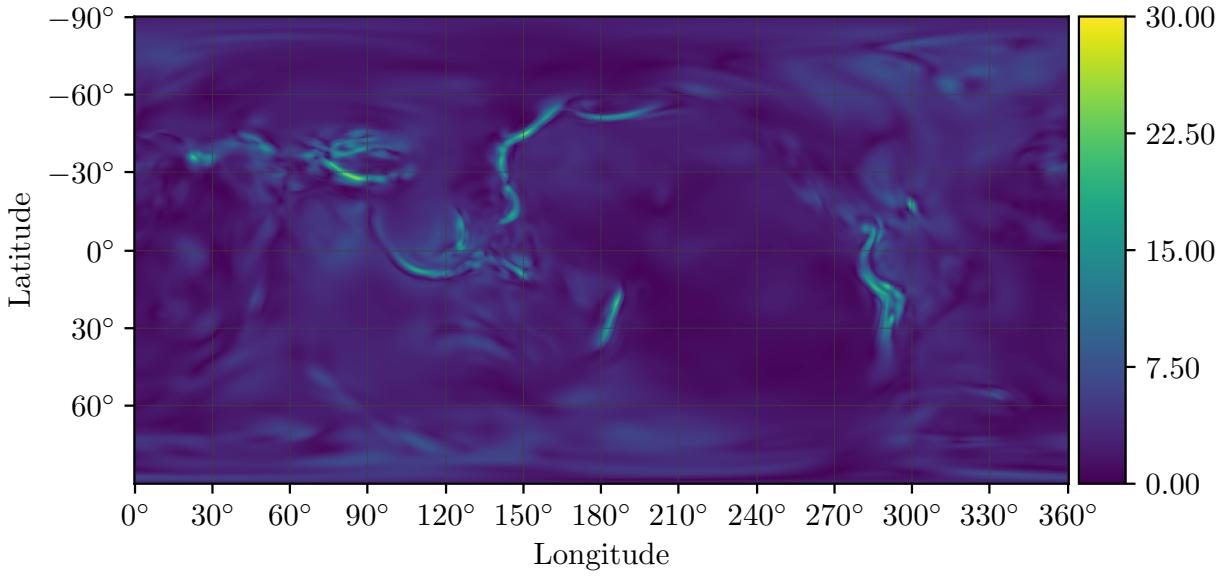
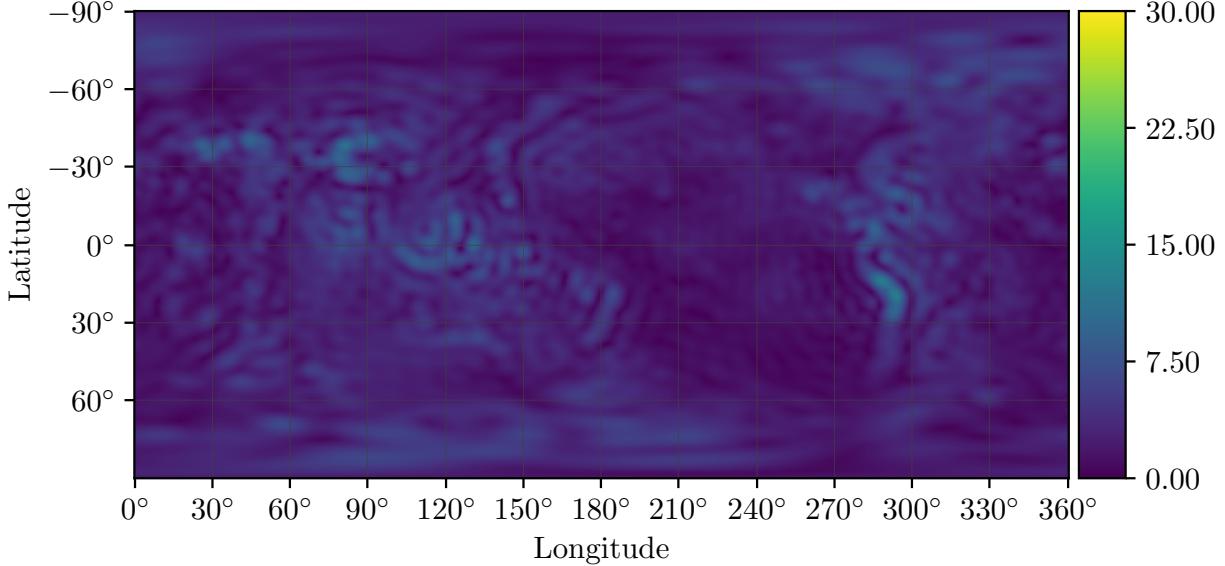
(a) PINN  $N = 20 \Leftrightarrow p = 3,040$ (b) Spherical harmonics  $l = 55 \Leftrightarrow p = 3,080$ 

Figure 3.8: Gravity model of Earth using (a) the neural network representation and (b) the spherical harmonic representation given approximately the same number of free parameters ( $p \approx 3,000$ )

hidden layers grow increasingly complex. The earlier layers activate over broad regions across the entire Brillouin sphere, whereas the deeper layers activate over more localized features. This highlights how the shallower layers in networks tend to resolve high-level, abstracted feature spaces while the deeper layers begin conforming to the specific perturbations of the body in question.

Figure 3.8 visually demonstrates the difference between modeling Earth’s discontinuous perturbations using low parameter spherical harmonic model and the more flexible neural network representation. Figure 3.8a shows how the network representation is able to generate a sensible and accurate basis set capable of representing the most prominent perturbations. Conversely, spherical harmonics prescribe oscillatory basis functions which are not amenable to modeling discontinuous mountain ranges and subduction zones as shown in Figure 3.8b. In fact, the spherical harmonic basis can leave unintended wave patterns that obfuscate the important perturbations. Figure 3.8 thereby highlights one of the important takeaways of this research: Astrodynamics do not need to apply a one-size-fits-all basis to every gravity field (i.e. spherical harmonics); instead neural network provide astrodynamists with the choice of generating unique basis functions that are maximally efficient for their specific gravity modeling problem.

### 3.1.4 Generalization

The prior experiment focused on the PINN-GM-I’s accuracy at the Brillouin sphere where the gravitational perturbations are most prominent. While the surface of the body offers the most complex dynamics due to the unattenuated perturbations, most spacecraft operate at higher altitudes where dynamics tend to simplify. As shown in Figure 3.3, the perturbations tend to decrease in magnitude and span larger spacial scales as altitude increases. This is a function of the  $(R/r)^l$  term within Eq. 2.1 which rapidly reduces the contribution of high-degree / small wavelength harmonics at larger radii. A second experiment is proposed which investigates how well the PINN-GM-I generalizes to these higher altitudes, as well as studies the role of the distribution of training data on network performance.

## Uniform Distribution

The first part of this experiment trains the traditional and PINN gravity models with 5,000,000 position and acceleration pairs distributed uniformly between 0-420 kilometers. This represents the most complete and optimistic set of training data to benchmark an upper-bound on performance. The model performance is measured using the same latitude and longitude of the 250,000 data Fibonacci grid detailed in Sect. 3.1.3, however the grid is evaluated at altitudes varied between 0 to 500 km in 10 km increments. These grids maintain the same  $\mathcal{A}$  and  $\mathcal{F}$  sets, simply evaluated at different altitudes which will henceforth be referred to as  $\mathcal{A}_m$  and  $\mathcal{F}_m$ , where  $m \in [0, 500]$  km. The MRSE metric is then applied to these altitude specific sets, converted into an equivalent spherical harmonic model degree, and plotted in Figure 3.9.

Figure 3.9 demonstrates that the networks' performance decays as a function of altitude. On average, the networks outperform their corresponding spherical harmonic equivalent for the first 100 km, but beyond 100 km, spherical harmonics become the more compact representation. Across all tested altitudes, the networks remain more accurate than degree and order 25 spherical harmonic model suggesting the networks are well-behaved within the domain of the training data and remain a viable option for gravity field modeling up to a LEO altitude.

## Non-Uniform Distributions

The second part of the experiment investigates the PINN-GM-I performance when the training data is not distributed uniformly in altitude. In more realistic mission circumstances, spacecraft will not be able to fly close to the surface easily and samples must be collected primarily from orbit. To reflect these circumstances, two new datasets of 1,000,000 position / acceleration pairs are drawn from the following exponential distribution:

$$\mathbb{E}(x, x_0, \beta) = \exp \left\{ -\frac{\|x - x_0\|}{\beta} \right\}, \quad (3.8)$$

where  $x_0$  is the reference altitude of 420 km, and  $\beta$  is the scaling parameter. The first dataset sets  $\beta = 10$  km and simulates a data distribution collected by a spacecraft that begins in a high

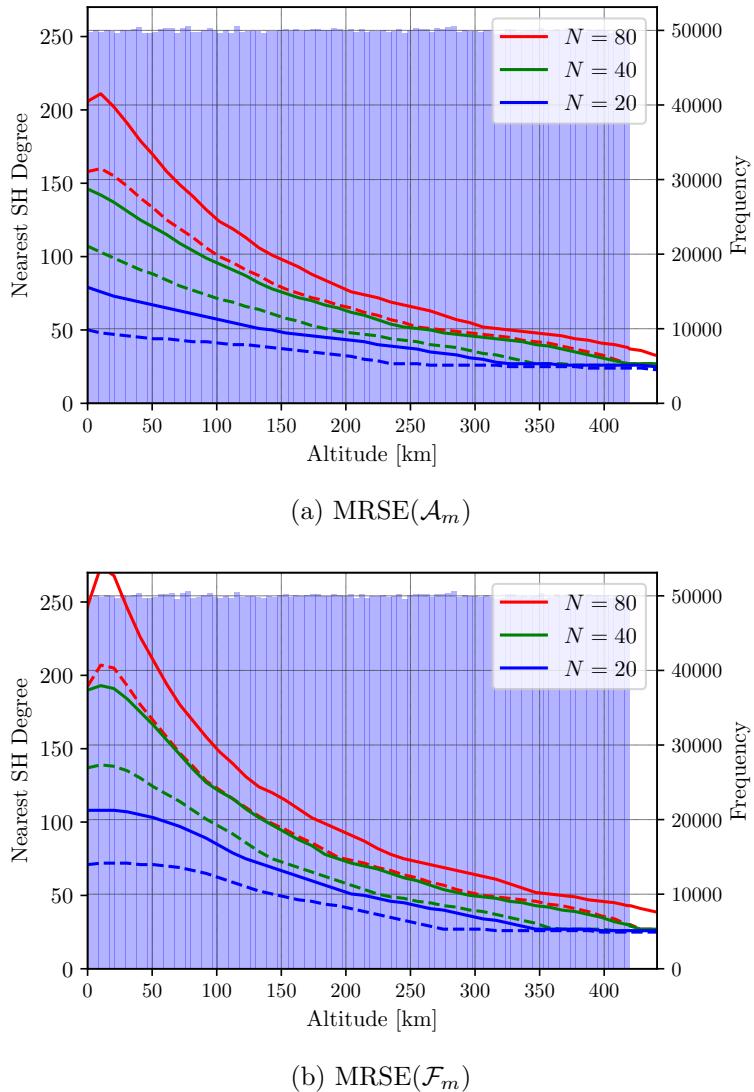


Figure 3.9: MRSE of  $\mathcal{A}_m$  (top) and  $\mathcal{F}_m$  (bottom) for the traditional (dashed) and physic-informed (solid) neural networks converted into the equivalent spherical harmonic degree as function of altitude. The blue histogram represents the training data distribution

altitude orbit before gradually deorbiting. As such, the majority of the data would come from an operational orbit regime with sparser measurements closer to the surface of the body. The second distribution sets  $\beta = 3$  km and represents a satellite in an eccentric orbit that remains at — and collects data from — an operational orbit altitude. There may exist infrequent measurements near the surface but virtually all of the data are collected at altitudes greater than 200 km. Note that to prevent sampling from inside the Brillouin sphere, the distribution for both datasets is restricted to  $x \in [0, 420]$  km. The results of the  $N = \{20, 40, 80\}$  traditional networks and PINNs are shown in Figure 3.10.

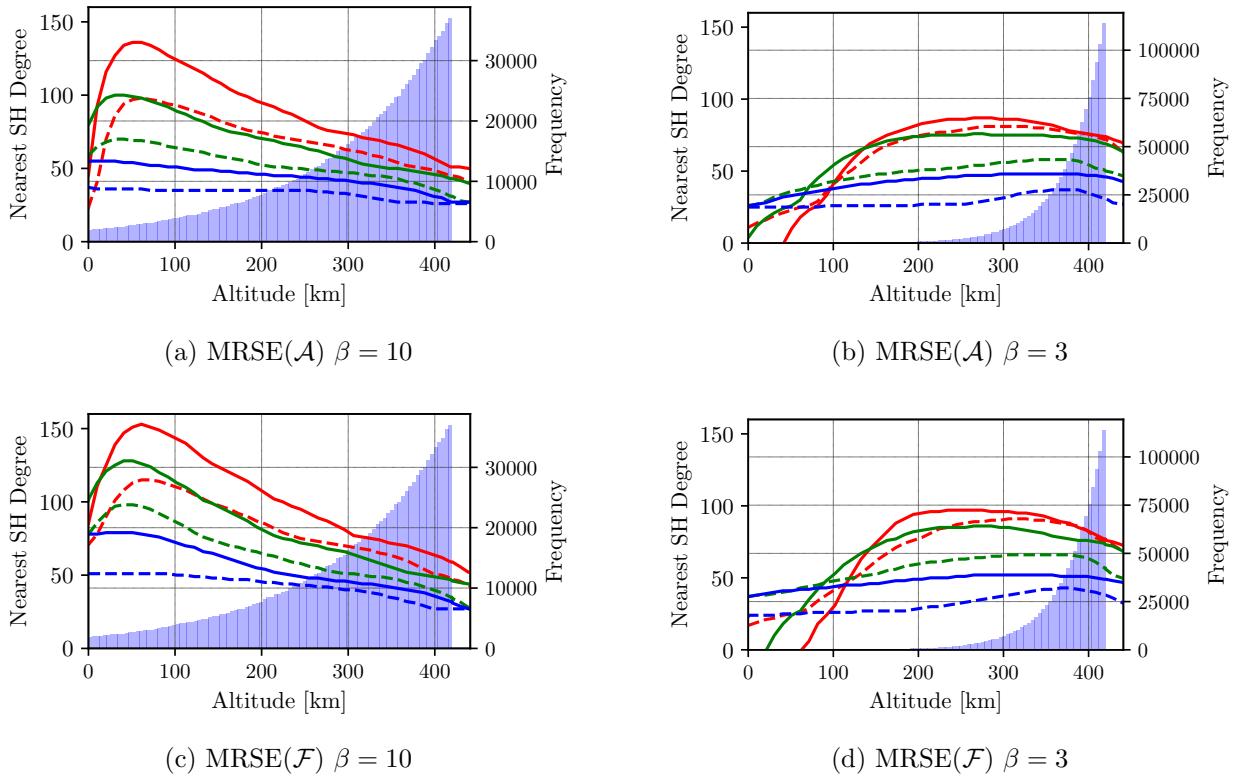


Figure 3.10: Training data distribution and equivalent spherical harmonic degree at varying altitudes for the  $\beta = 3$  and  $\beta = 10$  datasets. Solid lines represent the PINNs and dashed lines represent the traditional neural networks

## Discussion

Figures 3.9 and 3.10 demonstrate that both the traditional neural network and PINN gravity models are most productive when exposed to low-altitude samples, even if infrequently. Figures 3.10b and 3.10d show that the networks struggle to predict accurate accelerations at the Brillouin sphere given no data, but with a mere 2,000 data collected near the surface, the networks achieve substantially better performance at lower altitudes as shown in Figures 3.10a and 3.10c.

This sensitivity to low-altitude sample highlights an intrinsic bias of the early generations of the PINN-GM. Specifically, these neural network models are trained to minimize a cost function that compares the squared difference between the true and predicted accelerations. This inadvertently biases the network to accurately model low-altitude samples first, because low-altitudes are where the accelerations vectors are the largest. At these low-altitudes, even small relative modeling errors will contribute more to the cost function than large relative errors at high altitudes. By consequence, the early generation PINN-GMs will always prioritize these samples, even if they are relatively small fraction of the total training dataset.

Despite this bias, it is worth noting that the physics-informed networks consistently outperform their traditional networks counterparts again highlighting their greater data efficiency. There are minor exceptions, however, in the low-altitude regime of the  $\beta = 3$  distribution. In these regimes, the networks are fully extrapolating beyond the bounds of their original training data, so it is generally not advisable to use either the traditional or PINN gravity models.

These generalization results suggest that the PINN-GM-I is most advantageous when used near the surface of a body. For Earth-based spacecraft operations, this is not a common operating regime as, at these altitudes, atmospheric drag alone would produce greater dynamic uncertainty than the high-order gravity perturbations. For bodies with very thin or non-existent atmospheres however (e.g. the Moon or small-bodies), lower orbit altitudes are feasible and present a viable use-case for the network representations. Later PINN-GM generations explore ways to remedy this altitude dependence as will be shown in Chapter 3.3.

### 3.1.5 Computational Speed

The last experiment for the PINN-GM-I investigates how quickly the trained network models can be executed as compared to other popular gravity representations. The neural network gravity models are written in Tensorflow 2.4 and executed on an NVIDIA RTX 2060 GPU and on the Ryzen 3400G for the GPU and CPU cases respectively. Figure 3.11 shows the execution time required to evaluate the accelerations of 10,000 randomly distributed position data for gravity models with increasing numbers of parameters. The PINN performance is compared first to the spherical harmonic representation, and then to the polyhedral gravity model. The spherical harmonics representations are each generated from the EGM-2008 model with different truncation degrees. The polyhedral models tested use increasingly degraded shape models of 433-Eros generated using Blender<sup>2</sup>. The two analytic representations are written in Python, just-in-time compiled using Numba<sup>3</sup>, and executed on a Ryzen 3400G CPU.

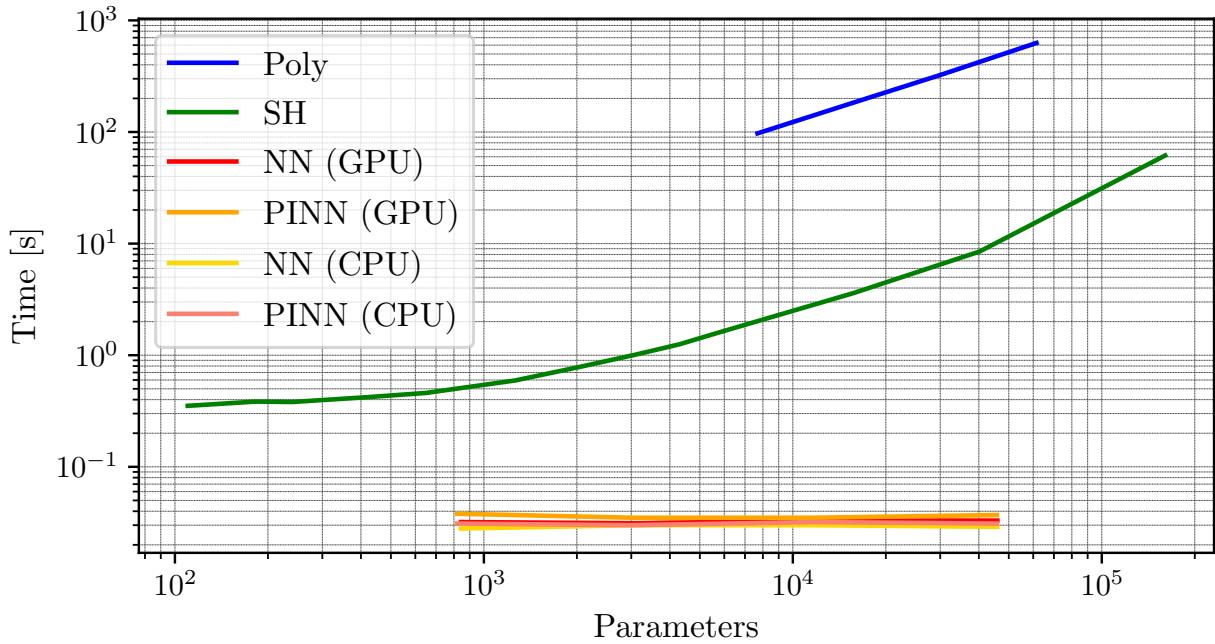


Figure 3.11: Total evaluation time to evaluate 10,000 random data using the various gravity models

<sup>2</sup> <https://www.blender.org/>

<sup>3</sup> <https://numba.pydata.org/>

Figure 3.11 demonstrates that the spherical harmonics representation is relatively efficient to evaluate at low degree, but it also has the steepest gradient as more parameters are included in the model. This verifies spherical harmonics  $\mathcal{O}(n^2)$  complexity. The polyhedral representation is by far the most time consuming to evaluate — taking nearly two orders-of-magnitude longer than that of a spherical harmonic model equipped with the same number of parameters. The neural network representation run either on the GPU or CPU is considerably more efficient than both of these representations — with performance that is nearly an order of magnitude more efficient than the lowest spherical harmonic model tested ( $l = 10$ ), independent to the number of parameters by these networks. These results are encouraging both for use within simulation and on-board spacecraft, producing high-fidelity estimates of the gravity field that can be used for trajectory design or on-board control purposes.

### 3.1.6 Network Performance Applied to the Moon’s Gravity Field

The experiments presented thus far are focused specifically on the Earth, but the conclusions on model compactness and generalization are not necessarily universal to any celestial body whose gravity field is traditionally represented by spherical harmonics. To demonstrate this, the prior experiments are repeated for a body with characteristically different perturbations: the Moon.

The Moon offers an interesting point of comparison to the Earth, as the gravitational perturbations of the Moon are substantially more frequent and of larger magnitude. The Earth’s perturbations are typically generated by large, infrequent, and localized geologic structures (mountains, tectonic plate boundaries, etc). The Moon’s perturbations, in contrast, are generated by craters and associated mascons which cover most of its surface. As will be shown in upcoming experiments, the complexity of the Moon’s gravity field makes efficient modeling of such perturbations more challenging for the PINN-GM-I.

Figure 3.12 aims to contrast the differences between the two bodies and their gravity fields. Qualitatively, note how much simpler the Earth’s gravity field is compared to that of the Moon. The craters on the Moon not only form a near random-surface topology, but also they intermingle

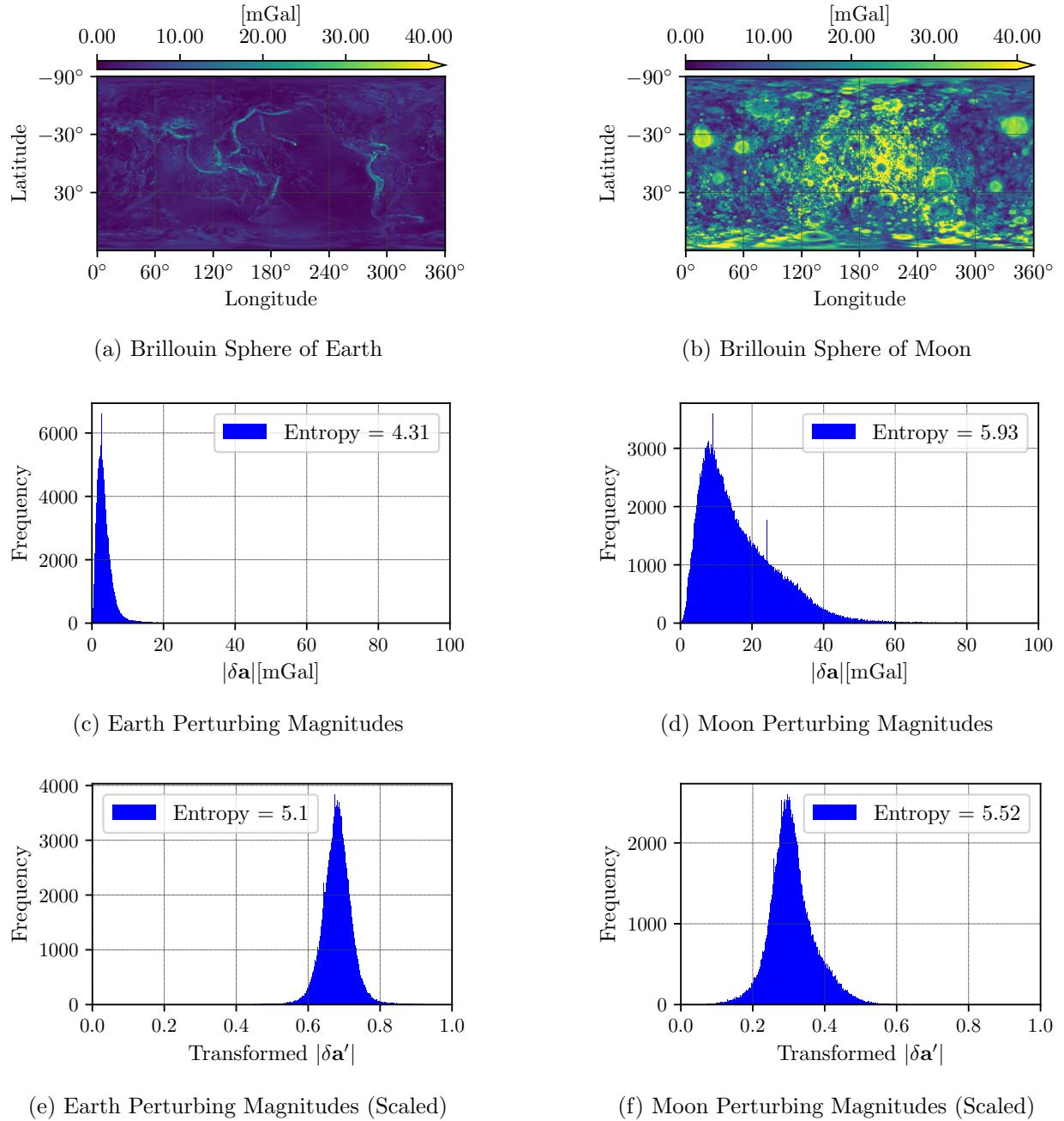


Figure 3.12: Contrasting gravity field and acceleration distributions of the Earth and Moon

some of the strongest perturbations with the weakest. The Earth’s perturbations, in comparison, are much more well-behaved. The complexity of the two fields is quantified by estimating the Shannon entropy of the acceleration distributions shown in Figures 3.12c and 3.12d. For the Earth, the field entropy is 4.31 nats (unit of information expressed in base  $e$ ) whereas the entropy for the Moon is 5.93 nats. This suggests that the Moon’s gravity field contains approximately five times more information on average than that of Earth’s gravity field — a significantly more challenging modeling task for the neural networks. These distributions are preprocessed before using them as training data for networks, so Figures 3.12e and 3.12f show how the distributions change once having applied the uniform min-max transformation to the acceleration vectors as detailed in Sect. 3.1.1. After the preprocessing, the Moon’s gravity field still contains more information compared to the Earth, albeit by only 50%. This suggests in both the raw and preprocessed form, the Moon’s gravity field is the more challenging modeling task.

## Compactness

The PINN-GM-I model accuracy is evaluated as a function of parameters and compared to the spherical harmonic gravity model in Figure 3.13. The figure replicates the same experiment of Sect. 3.1.3, evaluating error on just the dominate gravitational perturbations,  $\mathcal{F}$ , as well as the average and background data ( $\mathcal{A}$  and  $\mathcal{C}$ ). Instead of training the networks from field points drawn uniformly between 0-420 km altitude, the training data spans between a 0-50 km altitude. This narrower training domain attempts to reflect the fact that spacecraft can orbit at lower altitudes around bodies like the Moon given its very thin atmosphere. In addition, the true potential,  $U_{\text{Truth}}^{\text{SH}}$  is generated using the lunar GRGM1200A gravity field model (54).

The spherical harmonic gravity model is considerably more efficient at representing the Moon’s gravity field than that of the Earth as shown in Figure 3.14. For the Earth, nearly 10,000 spherical harmonic coefficients are required to begin converging on the feature set,  $\mathcal{F}$ , whereas the Moon requires mere 300 spherical harmonic coefficients. This suggests that long wavelength harmonics play a greater role in representing the Moon’s gravitational features, which is surpris-

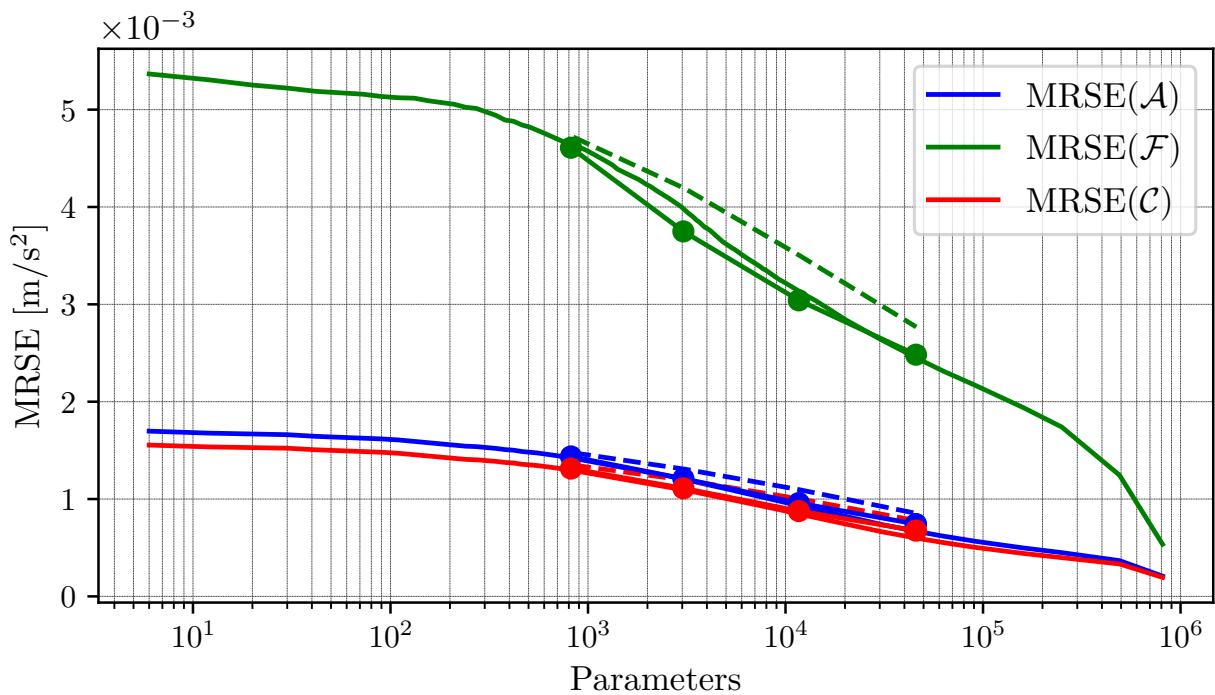


Figure 3.13: Plot of MRSE as a function of total model parameters for the Moon. Dashed lines represent traditional neural networks. The lines with circle markers represent the physics-informed neural networks

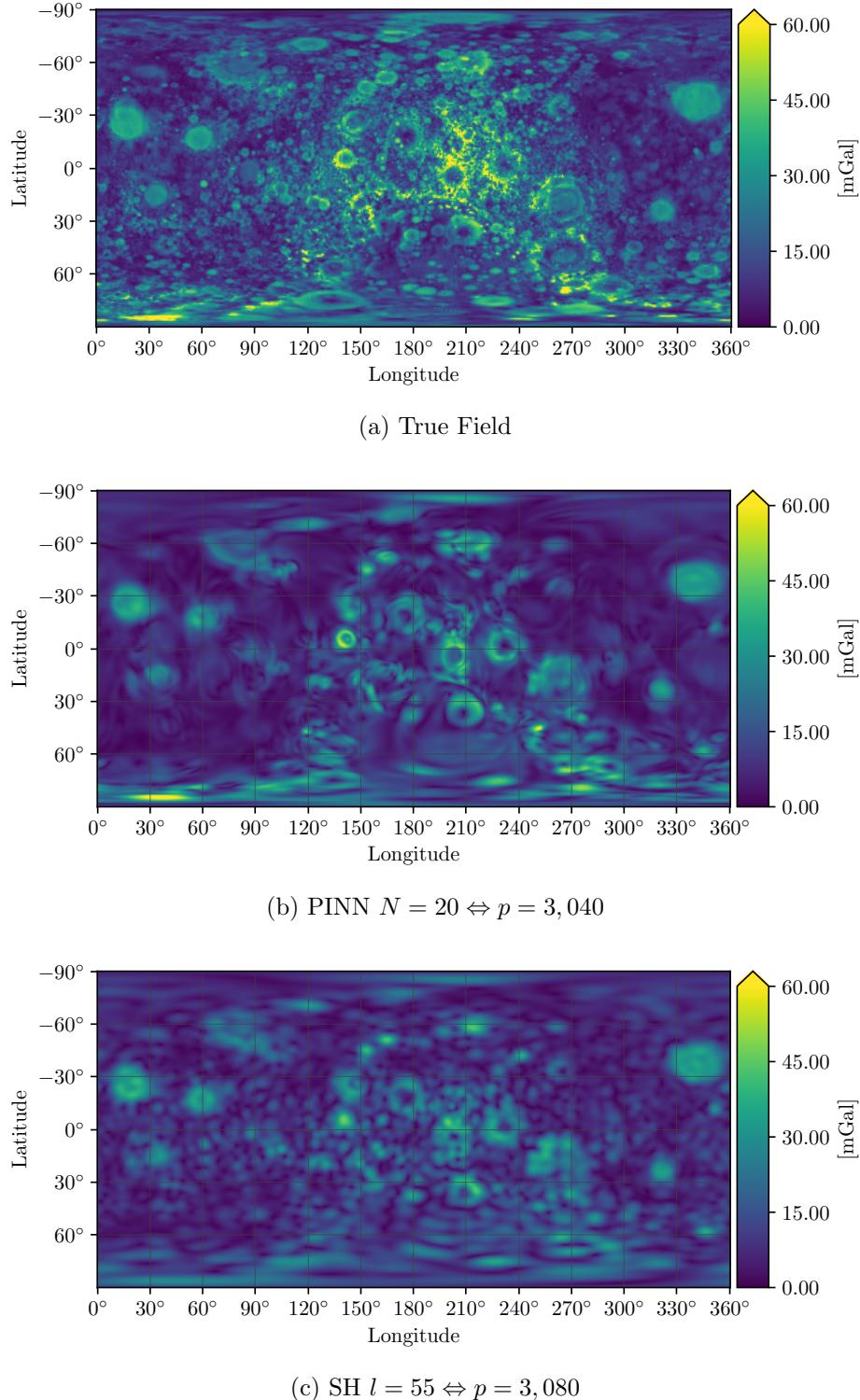


Figure 3.14: Gravity model of Moon using (a) the full  $l = 1,000$  spherical harmonic model, (b) the PINN representation with  $p = 3,040$  and (c) the low fidelity  $l = 55 \Leftrightarrow p = 3,080$  spherical harmonic representation

ing given the sharp discontinuous features present in the Moon’s field. The efficiency of these low-degree harmonics is attributed to the near isotropic nature of the perturbations. Even if the low-degree harmonics cannot efficiently reduce the error of a single discontinuous feature like a crater, they do contribute a small amount to all of the craters covering the entire surface. This collective contribution generates a measurable reduction in error seen within the feature set,  $\mathcal{F}$ .

That said, the PINN-GM-I still offers a slight advantage over spherical harmonics in terms of model compactness, producing comparable gravitational estimates using approximately 30% fewer parameters than spherical harmonics at best. The relatively small performance gain is an interesting result attributed to two factors: First is the aforementioned efficiency of the long-wavelengths in the spherical harmonic representation. The analytic model is simply more effective for the Moon than it was for the Earth, so the networks are competing with a more productive representation.

The second contribution to the performance discrepancy is the greater complexity / higher entropy of the Moon’s gravity field. The networks are being tasked to generate a distribution that contains more information than was present in the Earth’s gravity field — i.e. the networks have to work harder. It remains possible that greater network performance exists, but to witness that performance the networks would likely require additional training data, additional feature engineering, and longer training times to compensate for this more complex problem.

Future work is required to determine if there is a more fundamental relationship between the entropy of a gravity field and the capacity for a successful neural network gravity model. Because environments with high gravitational entropy will generate accelerations that appear unstructured or random, it is possible that all basis sets (learned or analytic) will struggle equally to model these types of fields.

## Generalization

Following the same procedure as presented in Sect. 3.1.4, the PINN-GM-I performance is tested at varying altitudes to determine how these networks generalize to different orbits. The altitudes tested vary between 0-55 km and the MRSE for each dataset of  $\mathcal{A}_m$ ,  $\mathcal{F}_m$  and  $\mathcal{C}_m$  is shown

in Figure 3.15.

The general performance of the PINN-GM-I for the Moon closely mimics that of the Earth networks. Both the traditional and physics-informed neural networks tend to have greatest accuracy at low-altitudes, again highlighting the intrinsic bias of these models to low-altitude samples. That said, there is a less pronounced performance peak for the PINN-GMs representing the Moon than those of the Earth. This difference attributed to the Moon’s high-entropy gravity field. The field’s greater complexity ultimately acts as a type of regularization that helps prevent the networks from overfitting to the training data. The perturbations are so diverse and ever-present that the networks do not develop an overconfidence in their learned solution. Analogous effects are seen when using small mini-batch sizes to train neural networks. These small batches can cause a noisier gradient descent which takes longer to converge but assists the optimizer in exploring the cost landscape and ultimately can produce more robust solutions (86). The difference here is instead of decreasing the batch size, the complexity of the cost landscape is increased to achieve a similar effect.

In summary, the PINN-GM-I demonstrates some advantages over spherical harmonics when representing the gravity fields of large celestial bodies like the Earth and Moon. Among the most compelling of these advantages is the rapid executability of these models over their analytic counterparts, as well as their smaller model sizes and memory footprints.

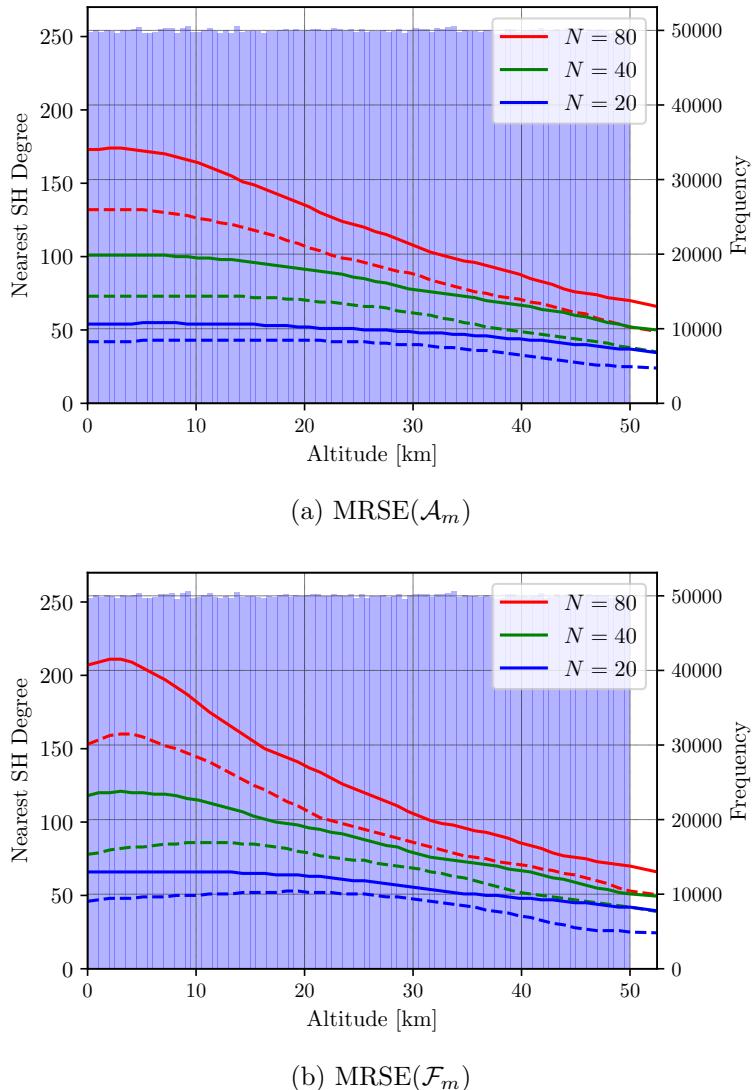


Figure 3.15: MRSE of  $\mathcal{A}_m$  (top) and  $\mathcal{F}_m$  (bottom) for the traditional (dashed) and physic-informed (solid) neural networks converted into the equivalent spherical harmonic degree as function of altitude for the Moon. The blue histogram represents the training data distribution

### 3.2 Generation II

While the results of the PINN-GM-I provide an encouraging start for the use of PINNs to represent gravity fields of large planetary bodies, there remain open questions regarding the capabilities and limits of the PINN gravity model particularly in the context of small-body exploration. For example, are these network representations equally robust when modeling the gravity field of non-spherical celestial bodies? Do the networks continue to generate accurate potentials and accelerations once inside of the Brillouin sphere? What training data conditions are necessary to achieve a robust model, and how sensitive are these networks to noise in the training data?

In addition, little work has been done discussing how these networks can be designed and tuned for increased performance. In Reference (87), only a single physics constraint is used in the cost function ( $\mathbf{a} = -\nabla U$ ), however other constraints can be introduced. Do such constraints assist in generalization or does the multi-objective nature of the additional constraints impede the learning process? Does the densely connected architecture of the network preclude the ability of the networks to learn abstract symmetries and conservation laws as suggested in Reference (81)?

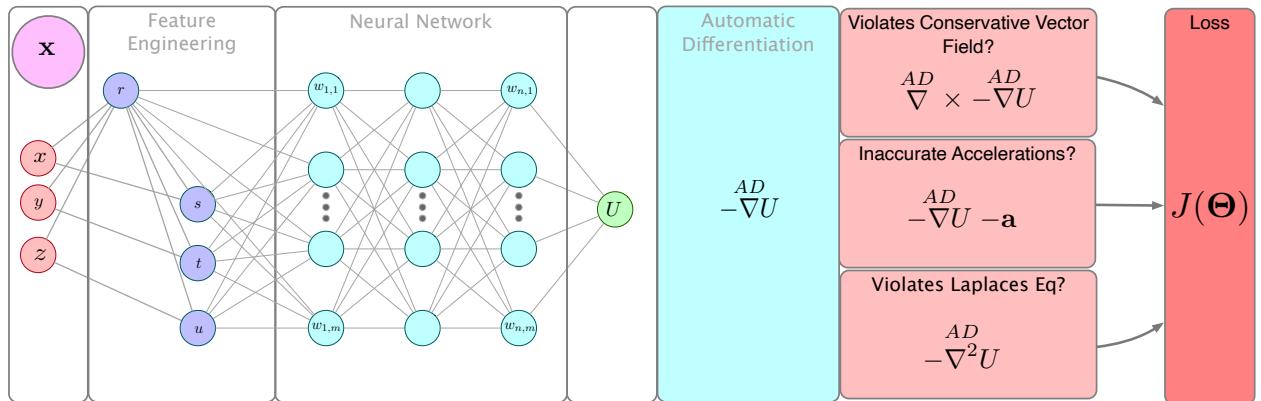


Figure 3.16: PINN-GM Generation II (1)

This section introduces the PINN-GM generation II, or PINN-GM-II, which proposes a collection of new design modifications to the original PINN-GM that aim to address these questions. Specifically, this section focuses on small-body gravity field modeling, exploring how the PINN-GM can be designed to produce high-fidelity gravity field models for the asteroid 433-Eros. The model

design changes of the PINN-GM-II and their corresponding effect on performance are detailed below.

### 3.2.1 Normalization

In many machine learning applications, the data used to train neural networks are normalized to ensure that the weights and biases learned by the network are not of exceedingly large magnitude, with the goal of avoiding numerical issues (88). In the first generation PINN-GM, the training data is normalized via a min-max transform on each acceleration component such that all training data existed between -1 and 1. While effective, this approach does not actually guarantee numerical stability for the network output. Normalizing the position and acceleration to [-1,1] provides no guarantees regarding the range of the potential, the actual output of the neural network. To ensure that the numerical stability of the network output is prioritized, PINN-GM-II introduces a different non-dimensionalization through:

$$\mathbf{x}^* = \frac{\mathbf{x} - x_0}{x_s} \quad (3.9)$$

$$U^* = \frac{U - U_0}{U_s} \quad (3.10)$$

$$\mathbf{a}^* = \frac{x_s}{U_s} \frac{dU}{d\mathbf{x}} \quad (3.11)$$

$$(3.12)$$

where  $x_0$  corresponds to the minimum value of all position components and  $x_s$  corresponds to the range of the position vector.  $U_0$  and  $U_s$  are the minimum and range of the potential.

### 3.2.2 Feature Engineering

In PINN-GM-I, the networks' inputs are position data represented in cartesian coordinates. While this approach is not inherently problematic, the domain of each unscaled feature  $(x, y, z)$  extends from  $(-\infty, \infty)$ , forcing the network to accommodate a substantially wider feature space than if represented in spherical coordinates. PINN-GM-II changes this construct by instead transforming the PINN inputs into a 4D spherical coordinate set of  $r, s, t$ , and  $u$  before passing them

through the network. These variables are defined as:

$$\begin{aligned} r &= \sqrt{x^2 + y^2 + z^2} \\ s &= \frac{x}{r} \\ t &= \frac{y}{r} \\ u &= \frac{z}{r} \end{aligned}$$

where  $s$ ,  $t$ , and  $u$  represent the tangents of angles between the field points and each of the cartesian unit vectors  $\hat{x}$ ,  $\hat{y}$ , and  $\hat{z}$  (84).

This 4D spherical coordinate set is chosen over the more traditional 3D set ( $r$ ,  $\theta$ , and  $\phi$ ) for multiple reasons. First, the values of  $s$ ,  $t$ , and  $u$  naturally exist between the desirable bounds of  $[-1, 1]$  whereas the  $\theta$  and  $\phi$  coordinates of the 3D set have discontinuity due to angle wrapping. Second, the 4D coordinate set avoids a singularity in the gradient of the potential at the poles. This can be seen when evaluating the 3D spherical gradient at  $\phi = -90^\circ$  and  $90^\circ$  as seen in:

$$\nabla U = \frac{\partial U}{\partial r} + \frac{1}{r} \frac{\partial U}{\partial \phi} + \frac{1}{r \cos \phi} \frac{\partial U}{\partial \theta} \quad (3.13)$$

This feature engineering that converts the cartesian position into the 4D spherical coordinates is performed within the official Tensorflow graph as a prepended layer to the neural network model. By integrating this transformation into the model, this means the potential can still be automatically differentiated with respect to the cartesian inputs. This construct provides the best of both coordinate sets: a neural network that trains on a significantly reduced feature space (increasing network convergence and accuracy) while maintaining the derivatives in a convenient cartesian basis.

### 3.2.3 Additional PINN Constraints

In addition to modifying the input features, PINN-GM-II introduces a modified loss function which include additional physics-informed constraints to improve modeling accuracy and robust-

ness. Explicitly, the loss function proposed for the PINN-GM-I only penalized errors in the predicted acceleration (as denoted by the subscript A in Eq. 3.3). While effective, this loss function fails to account for other dynamical properties that must be observed by the gravity model. For example, the force of gravity is a conservative force — therefore the scalar potential learned by the network must also obey these additional physics properties:

$$\nabla^2 U = 0 \quad \text{Laplacian (L)} \quad (3.14)$$

$$\nabla \times \mathbf{a} = 0 \quad \text{Curl (C)} \quad (3.15)$$

These conservative vector field properties can be embedded into the original PINN loss function through:

$$J_{\text{ALC}}(\Theta) = \frac{1}{N_f} \sum_{i=1}^{N_f} \left| \mathbf{a}_i + \overset{\text{AD}}{\nabla} f(\mathbf{x}_i | \Theta) \right|^2 + \left| \overset{\text{AD}}{\nabla}^2 f(\mathbf{x}_i | \Theta) \right|^2 + \left| \overset{\text{AD}}{\nabla} \times \overset{\text{AD}}{\nabla} f(\mathbf{x}_i | \Theta) \right|^2 \quad (3.16)$$

In addition, if a model of the gravitational potential exists, the loss function can be extended to penalize those modeling errors (subscript P):

$$J_{\text{APLC}}(\Theta) = J_{\text{ALC}}(\Theta) + \frac{1}{N_f} \sum_{i=1}^{N_f} |U_i - f(\mathbf{x}_i | \Theta)|^2 \quad (3.17)$$

These additions to the cost function should act as a form of regularization during training which can help improve robustness, particularly in the presence of noise.

### 3.2.4 Modified Network Architectures

Beyond incorporating more physics constraints into the network loss function, the PINN-GM-II also introduces an alternative network architecture. In Reference (81), it is observed that the success of many modern machine learning models rely on embedding symmetry groups or invariances into the network architectures (be it translational invariances in convolutional networks or temporal invariances in recurrent neural networks). Embedding these properties into the network architectures for physical processes can be challenging as it assumes preexisting knowledge of these symmetry groups which may not be present. For this reason, it is common to rely only on fully connected networks in modeling physical processes as opposed to more advanced architectures.

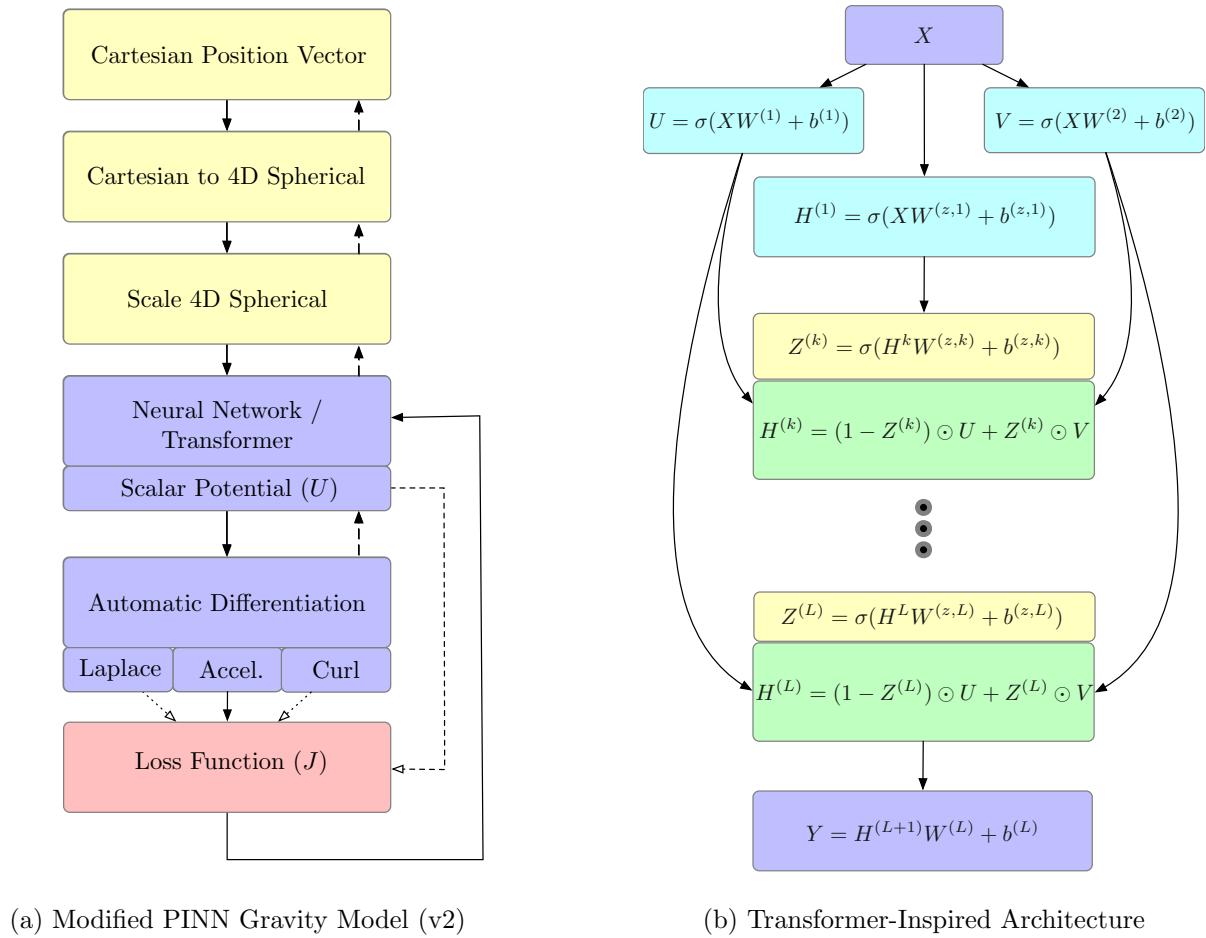


Figure 3.17: Modified Network Structure

Reference (81) implies that this choice may not be sufficient, as there does not yet exist a proof of the convergence of fully connected Physics-Informed Neural Networks. By consequence, researchers can only hope that fully connected networks provide sufficient modeling capacity and flexibility to represent the solution to the corresponding differential equation. Reference (81) supplement this claim by introducing how a relatively simple change in network architecture can significantly improve the accuracy of the learned model in a collection of archetypal PINN problems. This work adopts the proposed architecture, by extending the fully connected network with two encoders, which project the inputs into a high-dimensional space. In addition, the hidden states are enhanced with residual connections from prior layers through multiplicative, element-wise interactions as shown in Figure 3.17b. These modifications introduce relatively minor changes to the memory and computational footprint of the fully connected network but offer appreciable gains in model accuracy. This architecture change will be referred to as the Physics-Informed Transformer (PIT) for the remainder of this section.

### 3.2.5 Performance

To investigate the effect of these design changes on performance, a collection of experiments are introduced. These experiments are performed by sequentially adding the various design changes introduced above and characterizing the model performance at each intermediate stage.

To begin, baseline metrics are established to assess how well the PINN gravity model is able to learn the gravity field of asteroid 433-Eros from position and acceleration estimates. These experiments are designed assuming that a pre-existing high-fidelity gravity model exists (be it spherical harmonics, polyhedral, mascon, or other) from which perfect or noisy measurements can be drawn and used as training data to generate the PINN representation. For these experiments, the polyhedral representation is assumed to be ground truth and makes use of the Glaskell<sup>4</sup> shape model. Three unique datasets are proposed to characterize the effect of data quantity and distribution on model performance.

---

<sup>4</sup> <https://arcnav.psi.edu/urn:nasa:pds:gaskell.ast-eros.shape-model>

### Uniform Distribution $0 - 3R$ ( $r$ )

The first training distribution samples data uniformly from the surface of the asteroid to a radius equal to three times the maximum radius of the asteroid,  $R$  (Figure 3.19a). This sampling represents the best case scenario — where the network is trained on data that spans the entire problem domain (within, at, and above the Brillouin sphere).

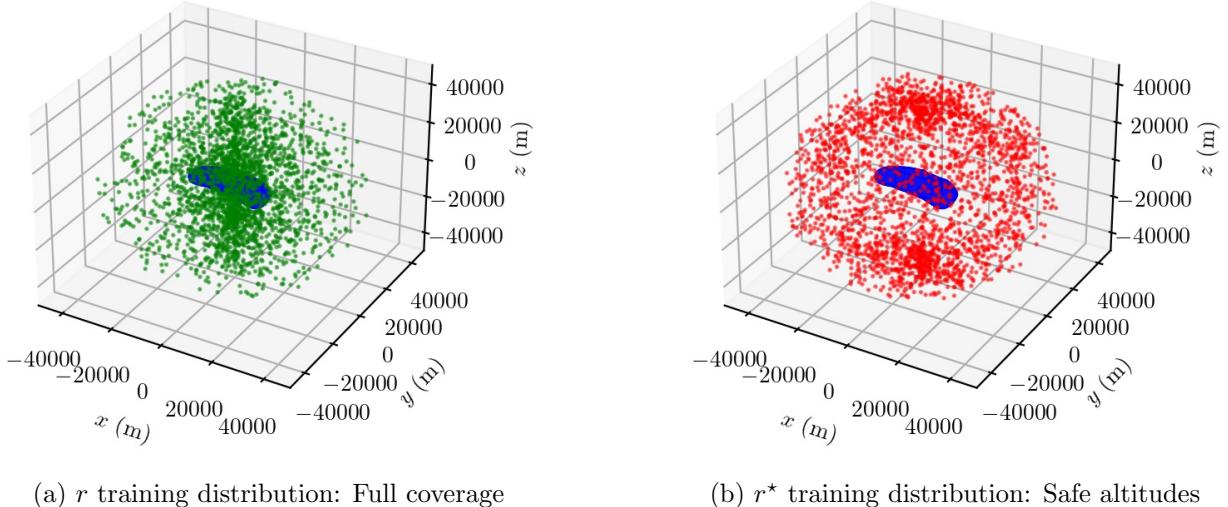


Figure 3.18: Training Distributions Asteroid 433-Eros

### Uniform Distribution $2R - 3R$ ( $r^*$ )

The second training distribution reflects more realistic circumstances — where samples are collected uniformly, but only from a range of safe altitudes above the asteroid (2-3 times the maximum radius). Such distribution is shown in Figure 3.19b. These altitudes are generally favorable for estimating low-degree spherical harmonic models or collect images that can be used to resolve a shape model for use in a polyhedral gravity model.

### Hybrid Distribution $2R - 3R$ plus Sparse Low-Altitude Samples ( $\bar{r}$ )

The final training distribution forms a hybrid set from the prior two distributions. The majority of the samples are collected from the  $2R - 3R$  ( $r^*$ ) distribution, but 10% are drawn between the surface and  $2R$ . These sparser samples could represent two potential phenomena: active asteroid ejecta or advance mission concepts like gravity poppers. As recently found in the OSIRIS-REx radio science experiment, some asteroids eject particles which can be tracked using optical measurements and their trajectories estimated to be used in gravity field recovery (89; 90). These particles greatly enhanced the resolution of the original radio science returns of OSIRIS-REx and has consequently motivated a new mission technology called gravity poppers (91). Gravity poppers are a technology demonstration which deploy artificial, uncontrolled probes from a mothercraft onto the surface of a small-body. Those probes then hop off of the surface of the asteroid, and the resulting dynamics are used to assist in estimating the gravity field of the body.

## Experimental Setup

These experiments begin by training PINN-GM-IIs with only the new normalization and feature engineering modifications applied. 5,000 training data are sampled from each of the aforementioned distributions and used to train a corresponding PINN gravity model for 7,500 epochs. This process is then repeated with artificial error added to the acceleration measurements to generate a more representative, noisy dataset as might be produced by an orbit determination process using dynamic model compensation. These noisy estimates are generated by adding an error vector with a random orientation and magnitude equal to 20% of the true acceleration vector magnitude. The error is therefore non-gaussian where the larger the true acceleration, the larger the corresponding error. The remaining hyperparameters used to train these networks are listed in Table 3.3.

After training each model, 20,000 test samples are selected randomly from the surface of the body to an altitude corresponding to a  $3R$  radius. The residuals between the true and predicted accelerations at the test locations are plotted alongside their moving average. In addition, two

Table 3.3: Nominal Hyperparameters

Parameter	Value	Parameter	Value
Activation	GELU	Hidden Layers	8
Batch Size	5,000	Nodes Per Layer	20
Optimizer	Adam	Weight Initialization	Glorot Normal
Epochs	7,500	Learning Rate	0.002

spherical harmonic models (one of degree and order 4, the other of degree and order 8) are fit to the same 5,000 training data using least squares regression and are tested on the same 20,000 test samples. The spherical harmonic model residuals and average error are also plotted to serve as a comparison point. These curves are presented over the corresponding training data distribution histograms in Figure 3.19.

### Baseline Results

In the best case scenario ( $r$  without noise), the PINN models produce accelerations with less than 3% average error all the way down to the surface — an encouraging feat given that the network was exposed to only 5,000 data points. When the 20% acceleration error is introduced into the estimates, the PINN’s error also increases, but remains below an average of 10%. The regressed spherical harmonics models, in contrast, never achieve better than 5% average error — independent of amount of error in the estimates — and can be seen diverging as they approach lower altitudes.

In the case of training PINNs on  $r^*$ , the learned representations achieve  $< 0.1\%$  average error within the training domain with perfect measurements and  $< 3\%$  when the measurements are noisy. However, the PINNs do suffer from extrapolation error beyond these bounds. While this behavior is undesirable, it is not unique to the PINN models. The regressed spherical harmonic models also suffer from greater extrapolation error. Moreover, even within the training domain, spherical harmonics models generate higher error than the PINNs. For both PINNs and spherical harmonics,

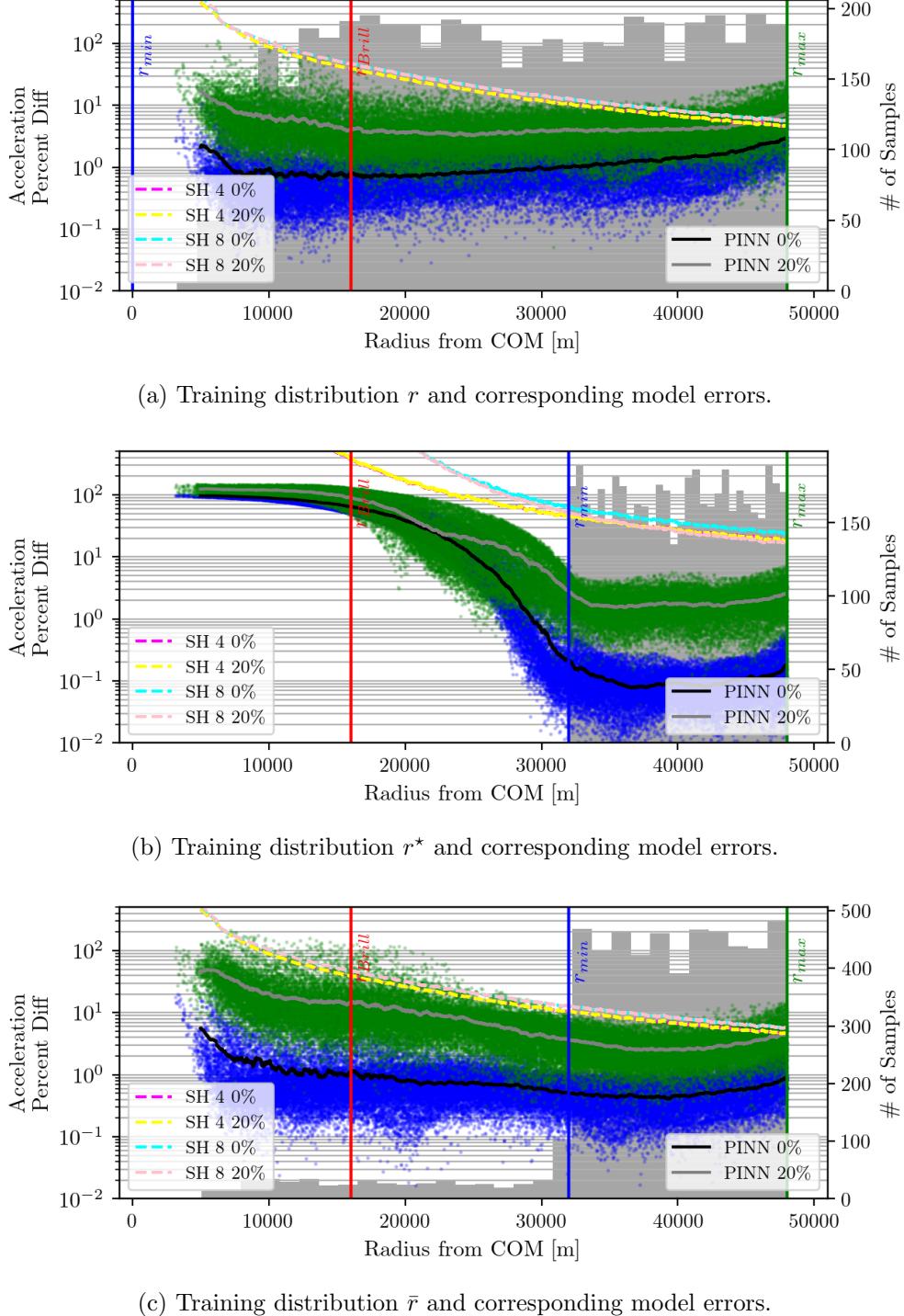


Figure 3.19: Acceleration residuals,  $|\mathbf{a} - \hat{\mathbf{a}}|/|\mathbf{a}| \times 100$ , their moving averages, and training data distribution as a function of radius. Blue scatter plots correspond to error of the PINN trained on perfect measurements, and green corresponds to the PINN trained on noisy measurements. Dashed lines represent the error of the spherical harmonic models fit on the same data. The gray histogram represents the radial distribution of the training data.

these results suggest that generalizable models require training samples from lower altitudes if they are to be used in close proximity to the body.

To investigate how many low altitude samples are necessary, the  $\bar{r}$  distribution supplements the  $r^*$  distribution with 500 additional samples collected between the asteroid surface and  $2R$  radius. By including this relatively small set of measurements, the PINN modeling accuracy stabilizes significantly. When no error is included within the training data, the average error of the PINN remains  $< 1\%$  for the majority of altitudes. Only at the surface of the body does the PINN model achieve an average modeling error of 5%. When error is added to the acceleration training data, the modeling accuracy decreases, but the total error remains below 10% for the majority of altitudes and only reaches a maximum average modeling error of 40% when evaluated at the surface — both sizable improvements in accuracy compared to the networks trained only on  $\bar{r}$ . This experiment is particularly encouraging as it implies that PINNs are quite efficient at extracting information about the gravity field when given access to sparse low-altitude samples. Such findings align with the findings of PINN-GM-I and provide strong evidence in favor of concepts like gravity poppers. Given how little data is needed to achieve generalizable models, there may not be a need for many gravity poppers or even poppers that can produce diverse trajectories.

### **Architecture and Loss Experiment**

To evaluate how additional physics constraints and modified network architecture improve model accuracy, another experiment is performed which trains the PINNs and PITs with different cost functions on increasingly sparse datasets ( $N = \{2500, 1250, 625\}$ ) comprised of position / acceleration pairs that are sampled randomly from  $0 - 3R$ . Once trained, the modeling error of these PINN and PIT representations are then evaluated on separate test datasets comprised of 20,000 randomly distributed position / acceleration pairs. The first test set contains 20,000 positions / acceleration pairs sampled outside of the Brillouin sphere and within  $3R$  of the body (exterior), the second test set is sampled between the surface of the asteroid and the Brillouin sphere (interior), and finally the third set samples positions / accelerations from 20,000 randomly

selected facets of the polyhedral shape model (surface). These three test dataset regions are shown in Figure 3.20. Each network model is trained using only one of the cost functions ( $J_A, J_{AP}, J_{ALC}$ ,  $J_{APLC}$ ) and a non-physics-informed cost function, denoted  $J_{00}$  or

$$J_{00} = \frac{1}{N} \sum_{i=0}^N |\mathbf{a}_i - \hat{\mathbf{a}}_i| \quad (3.18)$$

In addition, each of the models is trained on datasets with increasing levels of error added to the acceleration data. The erroneous acceleration estimates are generated in the same manner as presented in Sect. 3.2.5. The percent errors tested are 0%, 10%, and 20% of the true acceleration magnitude. Three spherical harmonic gravity models (corresponding to degree  $l = \{4, 8, 16\}$ ) are also regressed on each training dataset using least squares and evaluated using the same test datasets.

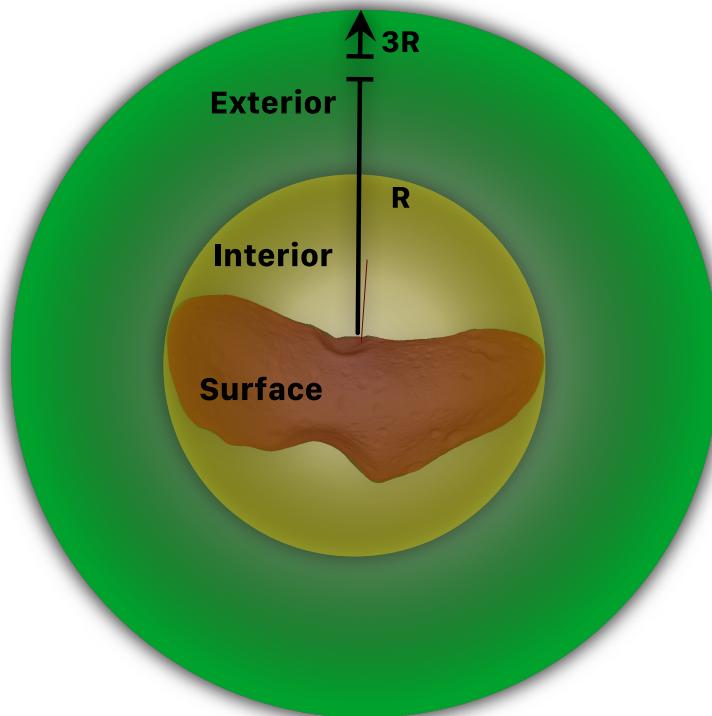


Figure 3.20: Eros Data Distributions

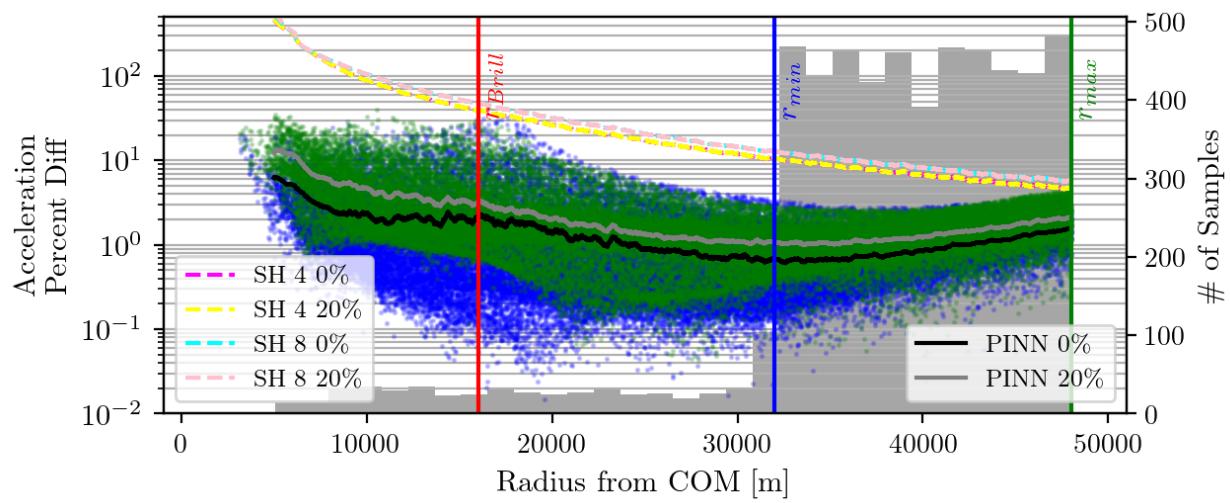


Figure 3.21: Acceleration residuals as a function of training distribution with  $J_{ALC}$  demonstrate how additional physics constraints help desensitize the model to noise in the training data

The performance of the PINNs, PITs, and spherical harmonics gravity models are plotted as a function of amount of training data,  $N$ , and error in the training data (0%, 10%, and 20%). The corresponding results for each test dataset (exterior, interior, and surface) are plotted in Figure 3.22, Figure 3.23, and Figure 3.24 respectively. Note that the model errors are quantified as acceleration percent error distributions, and they are represented as box-and-whisker plots. The box represents the 1st and 3rd quartiles (25% to 75%) of the percent error distribution, i.e. 50% of data lies within the box. The line within the box corresponds to the median or 2nd quartile. The lines extending beyond the box correspond to  $1.5 \times IQR$  where  $IQR$  is the interquartile range, and the points beyond those lines correspond with outliers. Also, Figure 3.22, Figure 3.23, and Figure 3.24 cluster the PINN, spherical harmonics (SH), and PIT models into different color groups. As such, the rows within each cluster of the legend correspond to the models in the figure plotted from left to right.

Figure 3.22, 3.23, and 3.24 demonstrate a number of interesting behaviors of the PINN and PIT gravity models. First, the additional constraints of the cost function act as a form of regularization to the regression. When there exists sufficient data ( $N = 2500$ ) with little-to-no noise, the multiple-constraint PINNs (ALC, APLC) tend to perform worse than their simpler counterparts (00, A, AP). However, when the noise in the dataset increases, or the number of samples decrease, the multiple-constraint networks are more robust. These networks verify that the additional constraints minimize the risk of overfitting to noise or sparse data and are a more trustworthy formulation in uncertain or novel environments. This can also be seen by comparing Figures 3.19 and 3.21 and noting how the PINN ALC network trained on the noisy  $\bar{r}$  distribution is functionally desensitized to the noise.

The second interesting finding from Figures 3.22, 3.23, and 3.24 is the consistent performance improvement of the PITs over the PINNs. In both the low- and high-sample regimes, the transformer-inspired architecture can offer as high as an order of magnitude reduction in error despite being trained with the same data as best illustrated in the low-to-no noise cases of Figure 3.22 and 3.23. It is worth noting, however, that the transformers are also more prone to overfitting than the PINNs due to their greater modeling capacity — as seen when using the simpler physics con-

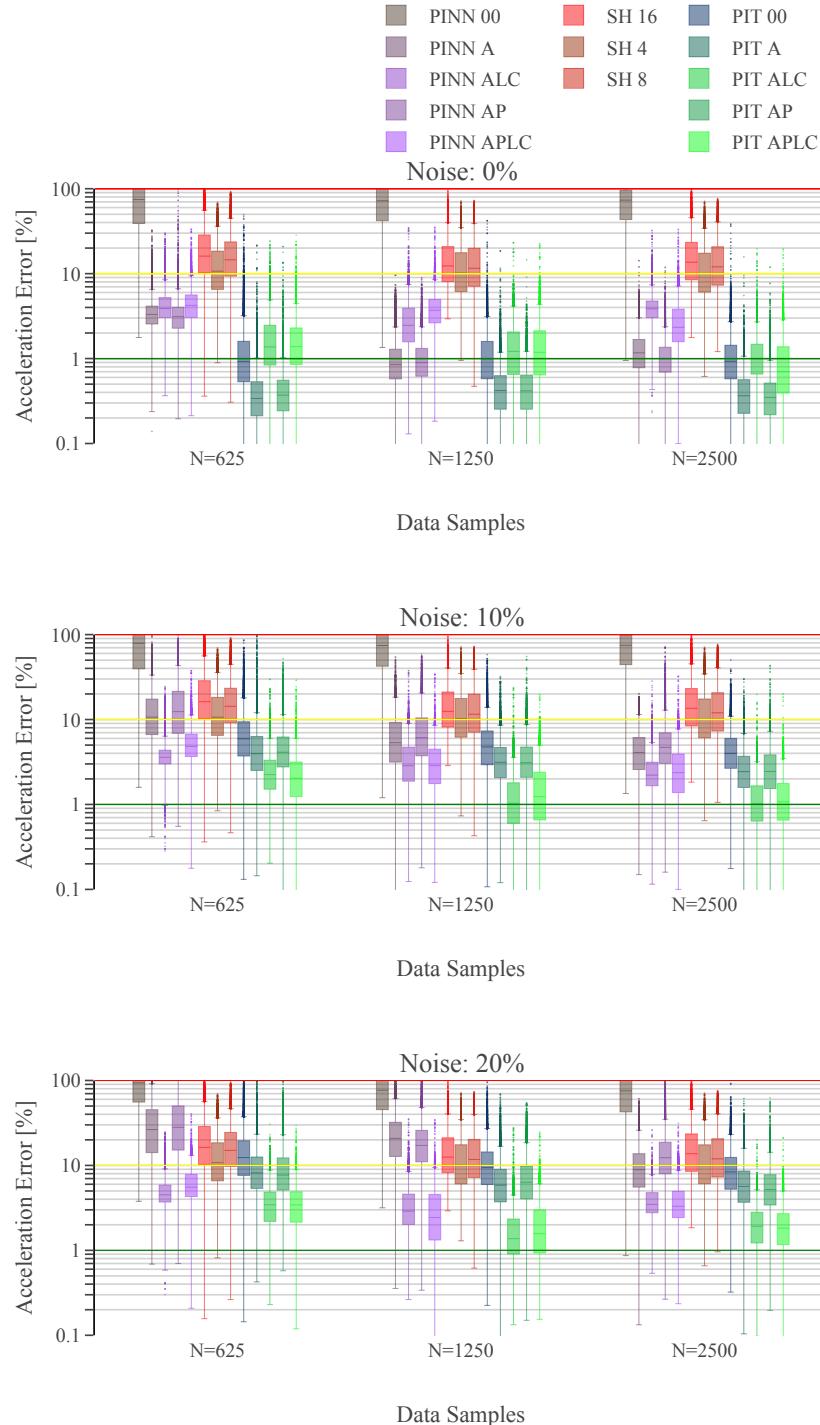


Figure 3.22: Model error **outside** the Brillouin sphere (exterior) as a function of amount of training data and noise added to the training data.

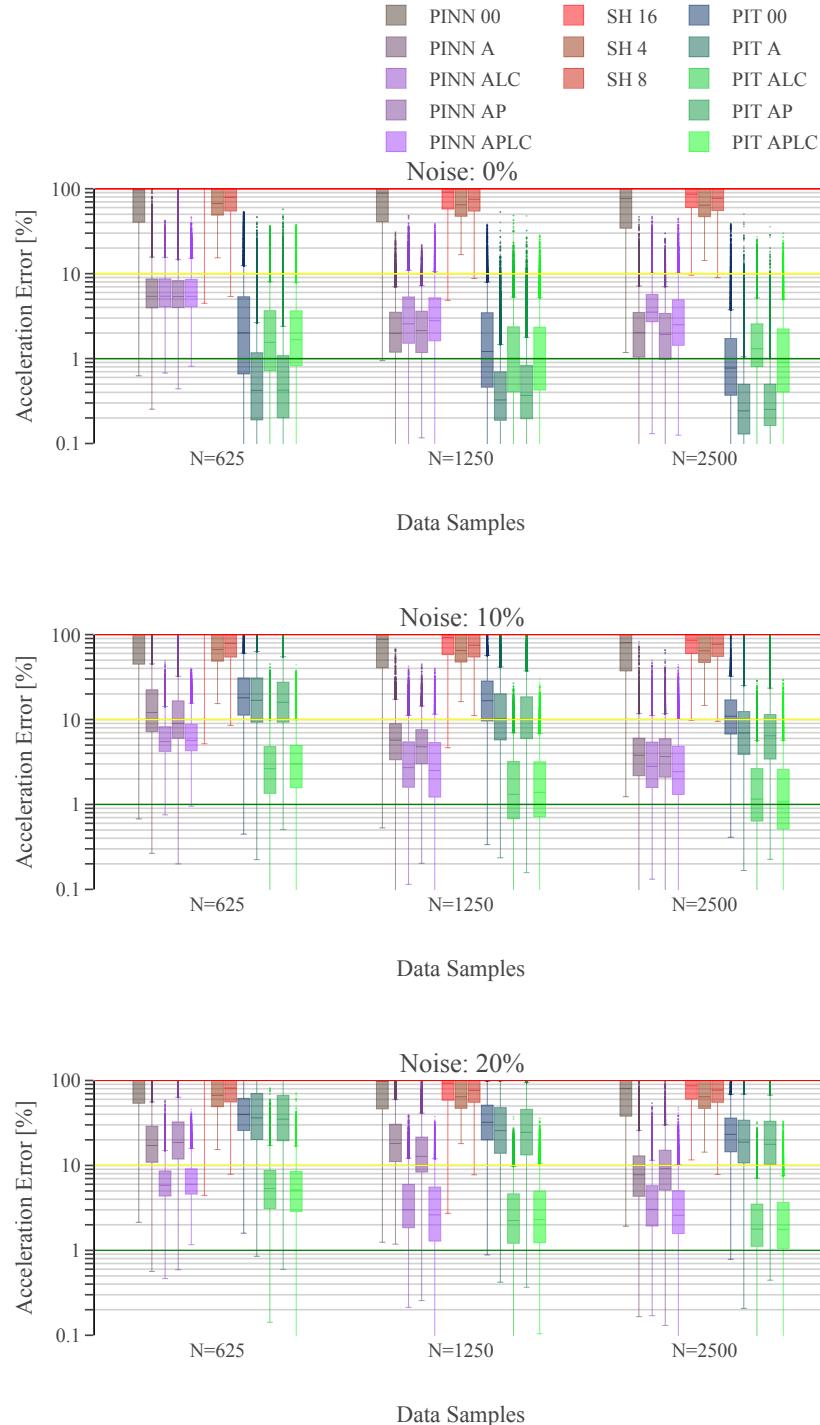


Figure 3.23: Model error **inside** the Brillouin sphere (interior) as a function of amount of training data and noise added to the training data.

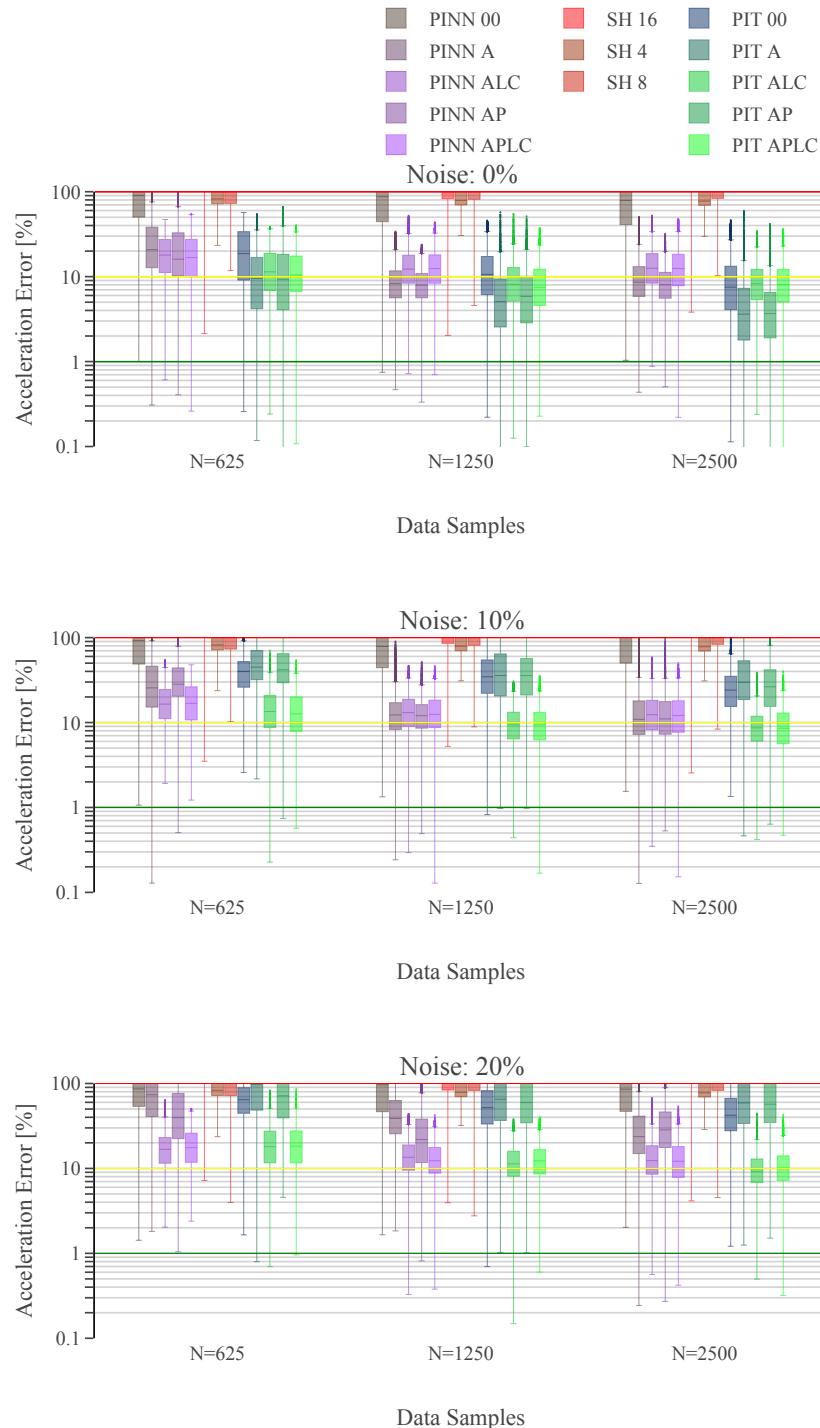


Figure 3.24: Model error **at the surface** of the asteroid (surface) as a function of amount of training data and noise added to the training data.

straints (00, A, AP) on the 10% and 20% noise cases in Figures 3.23 and 3.24. In the 0% noise case, this modeling flexibility could be considered a strength as the fewer physics constraints allow the model to converge more quickly than their more regularized variants. However, in more practical applications like when the field is not known a priori, it is recommended to use the additional constraints to minimize the risk of overfitting.

Finally, both the PINNs and PITs offer sizable performance gains over their spherical harmonic predecessor not only in the interior and surface distributions for which spherical harmonics will diverge, but also the exterior distribution where spherical harmonics has often had the analytical high-ground. This consistently worse performance of the spherical harmonic representation is a function of the small sample size, the random distribution of these samples, and the low altitude data — each capable of posing numerical challenges in the least squares estimate of the spherical harmonic coefficients.

Together the PINN-GM-II demonstrates favorable performance when applied to small-body environments. With improved feature engineering, additional loss terms, and improved network architecture, the PINN-GM-II achieves considerably better sample efficiency than its predecessor. Moreover, the PINN-GM-II demonstrates robustness to noisy data, and is successful at regressing low-altitude regimes with relatively few samples.

### 3.3 Generation III

The third and most recent generation of PINN gravity model (PINN-GM-III) introduces a collection of additional design modifications to address various pain points left unaddressed by past generations, while also increasing model accuracy and robustness. A visual depiction of these changes are shown in dark gray in Figure 3.25 and can be juxtaposed with the previous PINN-GM-II in Figure 3.16. All modifications are detailed in the sections below.

Model	Parameters	Training Data	Avg. Error [%]	Valid Globally
GP (45)	12,960,000	3,600	1.5%	✗
NNs (92)	1,575,936	800,000	0.35%	✗
ELMs (6)	100,000	768,000	1-10%	✗
GeodesyNet (7)	91,125	500,000	0.36%	✗
PINN-GM-III	3,048	4,096	0.20%	✓

Table 3.4: Machine Learning Gravity Model Statistics – See Appendix D

#### 3.3.1 Preprocessing Efforts

PINN-GM-III continues on the journey to find the most productive normalization of the PINN-GM inputs. As discussed previously, one of the most common practices is to normalize the data such that they exist within the bounds  $[-1, 1]$  as many activation functions exhibit their greatest non-linearity in this regime. Non-linearities are what provides neural networks their powerful approximation capabilities, and if the inputs saturate the activation functions then the network's modeling capacity is reduced. In addition, normalization also improves numerical stability during training, and decreases the risk of ill-conditioned matrix operations if the training values are too large or small.

Traditionally, the inputs and outputs of neural networks are normalized in a manner agnostic of one another, e.g. if the inputs are  $x$  and the outputs are  $y$  then normalization yields  $[x_{\min}, x_{\max}] \rightarrow [-1, 1]$  and  $[y_{\min}, y_{\max}] \rightarrow [-1, 1]$ . In PINNs, however, the inputs and outputs may share units, and this decoupled normalization can produce non-compliant physics. For the gravity field modeling

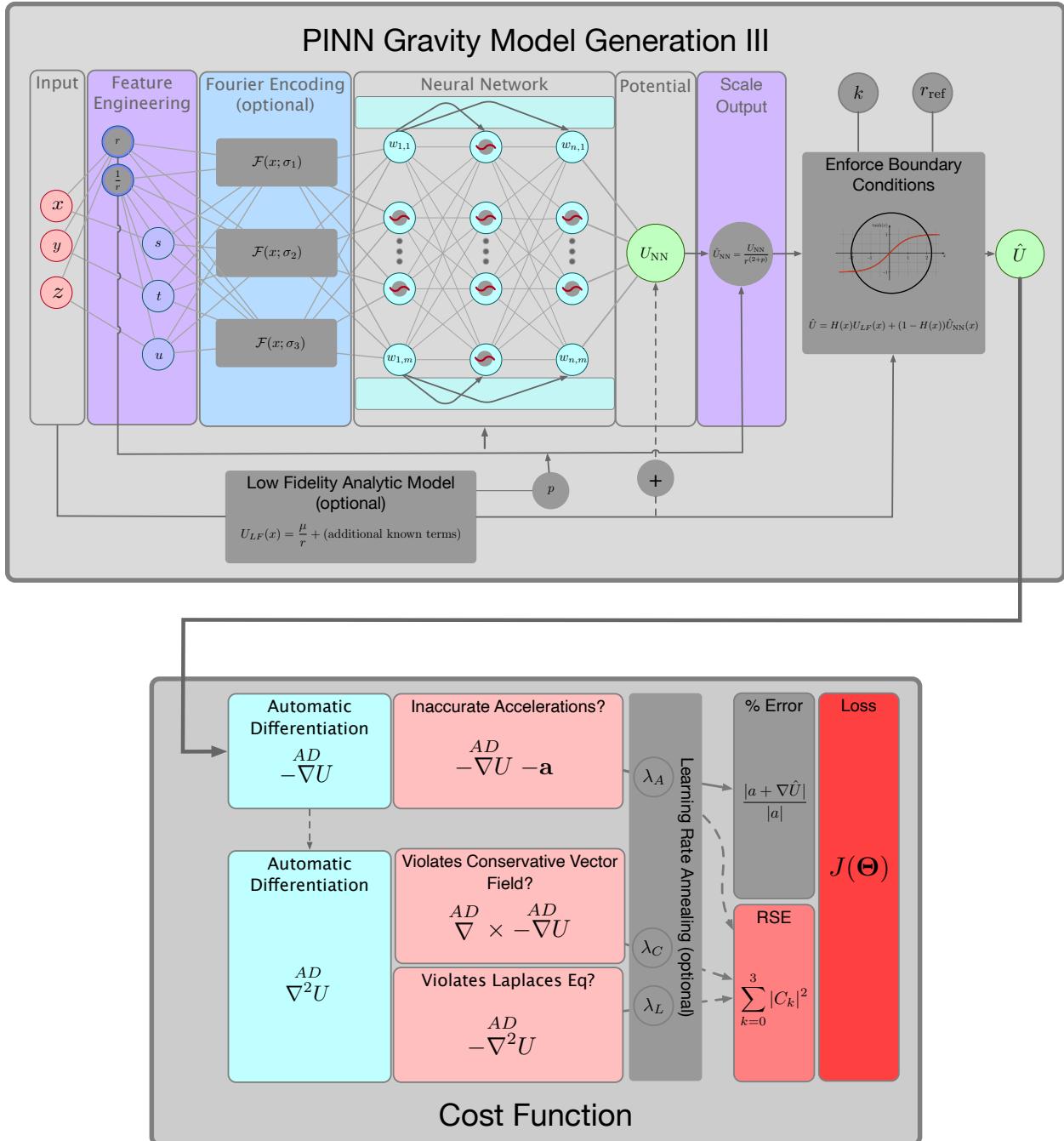


Figure 3.25: PINN-GM Generation III with new modifications contained in dark gray boxes

problem, it is therefore important to normalize the inputs and outputs in a dimensionally-informed manner. In PINN-GM-III this is accomplished by normalizing the position and potential by the characteristic length  $x^*$  equal to the planet radius  $R$  and maximum potential value in the training data  $U^*$  respectively. Using these characteristic scalars, a time constant can be computed and used in conjunction with  $x^*$  to non-dimensionalize the accelerations. Explicitly this manifests through:

$$x = \frac{\bar{x}}{x^*}, \quad U = \frac{\bar{U}}{U^*}, \quad a = \frac{\bar{a}}{a^*} \quad (3.19)$$

where  $x^*$ ,  $U^*$ , and  $a^*$  are the non-dimensionalization constants defined as:

$$x^* = R \quad (3.20)$$

$$U^* = \max_i (\bar{U}_i - \bar{U}_{\text{LF},i}) \quad (3.21)$$

$$t^* = \sqrt{\frac{x^{*2}}{U^*}} \quad (3.22)$$

$$a^* = \frac{x^*}{t^{*2}} \quad (3.23)$$

where  $R$  is the maximum radius of the celestial body,  $\bar{U}_i$  is the true gravitational potential at the training datum at  $\mathbf{x}_i$ , and  $U_{\text{LF}}$  is any low-fidelity potential contributions already accounted for within the PINN-GM (discussed in Section 3.3.5).

With the network inputs and outputs non-dimensionalized, one additional preprocessing step is introduced to ensure that all inputs remain bounded. Like with PINN-GM-II, the non-dimensionalized Cartesian position coordinates are converted into their 4D spherical coordinate description of  $(r, s, t, u)$  where  $r$  is the radius, and  $s$ ,  $t$ , and  $u$  are the sine of the angle between the field point and each Cartesian axes defined as  $x/r$ ,  $y/r$ , and  $z/r$  (84). This conversion ensures  $s$ ,  $t$ , and  $u$  remain in the numerically desirable range of  $[-1, 1]$ . The radial coordinate, however, is not guaranteed these same conveniences. For field points that exist at infinite radii from the body, the  $r$  coordinate can scale from  $[0, \infty)$  which can pose numerical challenges to the network. To circumvent these difficulties, two proxy values of the radius are introduced,  $r_{\text{clip}}$  and  $(1/r)_{\text{clip}}$ .  $r_{\text{clip}}$  is the radial component that is clipped for values greater than one, and  $(1/r)_{\text{clip}}$  is the reciprocal of  $r$  that is also clipped when its value exceeds one. This ensures that information about the radial

coordinate are always captured, but ensure that they never exceed the bounds of  $[0, 1]$  and remain numerically stable for learning.

### 3.3.2 Modified Loss Function to Account for high altitude Samples

The loss function for the original PINN-GM is a root mean squared (RMS) error metric:

$$\mathcal{L}_{\text{RMS}}(\theta) = \sqrt{\frac{1}{N} \sum_{i=0}^N \left| -\nabla \hat{U}(\mathbf{x}_i | \theta) - \mathbf{a}_i \right|^2} \quad (3.24)$$

This loss function is especially common in supervised machine learning regression problems, and it encourages models to minimize the most flagrant residuals between the true and predicted values. For the PINN-GM, this means minimizing the difference between the differentiated network potential,  $\nabla \hat{U}(\mathbf{x}_i | \theta)$ , and the true acceleration,  $\mathbf{a}_i$ , to satisfy the differential equation  $-\nabla U = \mathbf{a}$ . Despite its popularity, this loss function comes with unexpected disadvantages for the gravity modeling problem as shown in earlier sections.

Specifically, gravitational accelerations produced closer to a celestial body have considerably larger magnitudes than the accelerations produced at high altitudes. As a consequence, any small relative errors in low-altitude predictions will appear disproportionately large compared to any high altitude errors. Therefore gravity models trained with this cost function always prioritize low-altitude samples over high altitude samples, even if the high altitude samples are more erroneous in a relative sense. This was demonstrated by both the PINN-GM-I and the PINN-GM-II, for which both models prioritized the accurate modeling of the low-altitude samples, even when the majority of data existed at high altitudes.

To combat this design flaw, PINN-GM-III adopts a new mean percent error loss function:

$$\mathcal{L}_{\%}(\theta) = \frac{1}{N} \sum_{i=0}^N \frac{| -\nabla \hat{U}(\mathbf{x}_i | \theta) - \mathbf{a}_i |}{|\mathbf{a}_i|} \quad (3.25)$$

By using a percent error loss instead, the PINN-GM-III is no longer biased by the absolute magnitudes of the acceleration vectors, but instead seeks to minimize relative errors. This choice ensures that all samples, regardless of altitude, are contributing equally to the loss function and

are accurately modeled by the network. If a user has applications which are especially sensitive to low-altitude accelerations — such as a landing or touch-and-go operation — Equation (3.25) can optionally be augmented with Equation (3.24).

To illustrate how the network loss function affects model performance, a test is proposed which trains two PINN-GMs on 5,000 position / acceleration training data pairs distributed from 0-15 Earth radii above the Earth’s surface. One PINN-GM is trained using the original RMS loss function and the other is trained on with the percent error loss function. Once trained, each network is evaluated on a set of 10,000 randomly distributed test points within the 0-15R domain, and their resulting RMS and percent error values<sup>5</sup> are reported as a function of altitude in Figure 3.26.

Figure 3.26a confirms that the networks trained with the RMS loss function disproportionately favor low-altitude field points (0-2R) at the expense high altitude field points (5-15R) despite being trained across the entire 0-15R domain. In contrast, the PINN-GM model trained with the percent error loss function (Figure 3.26c) prioritizes accurate modeling at all altitudes and conveniently produces lower RMS values for high altitude samples as shown in Figure 3.26d.

### 3.3.3 Improve Numerics by Learning a Proxy to the Potential

All gravitational potentials decay at high altitudes according to a power law of the form  $1/r^p$ . As discussed, this decay poses challenges when using a RMS loss function, but it also introduces other numerical difficulties for neural networks. Consider that the largest gravitational potential represented by the network is non-dimensionalized to magnitude 1 from Equations (3.19) and (3.21). For field points at a sufficiently high altitude, the neural network will need to compute potentials which decay to values to less than or equal to machine precision (i.e.  $U(r \rightarrow r_{\text{crit}}) \leq \epsilon_{\text{machine}}$ ). Representing these vastly different numerical scales with the same matrix operations (neural network) is undesirable and can lead numerical instability during training and inference. As a consequence, this instability unnecessarily caps the maximum altitude for which the model is

---

<sup>5</sup> Note that the errors reported are only of acceleration contributions above a degree and order 2 spherical harmonic model—i.e. how accurately the network is captures all remaining gravitational perturbations beyond  $C_{2,0}$  and  $C_{2,2}$ . This avoids these contributions obfuscating the modeling error of the more discontinuous features and follows the procedure found in Section 3.1.2.

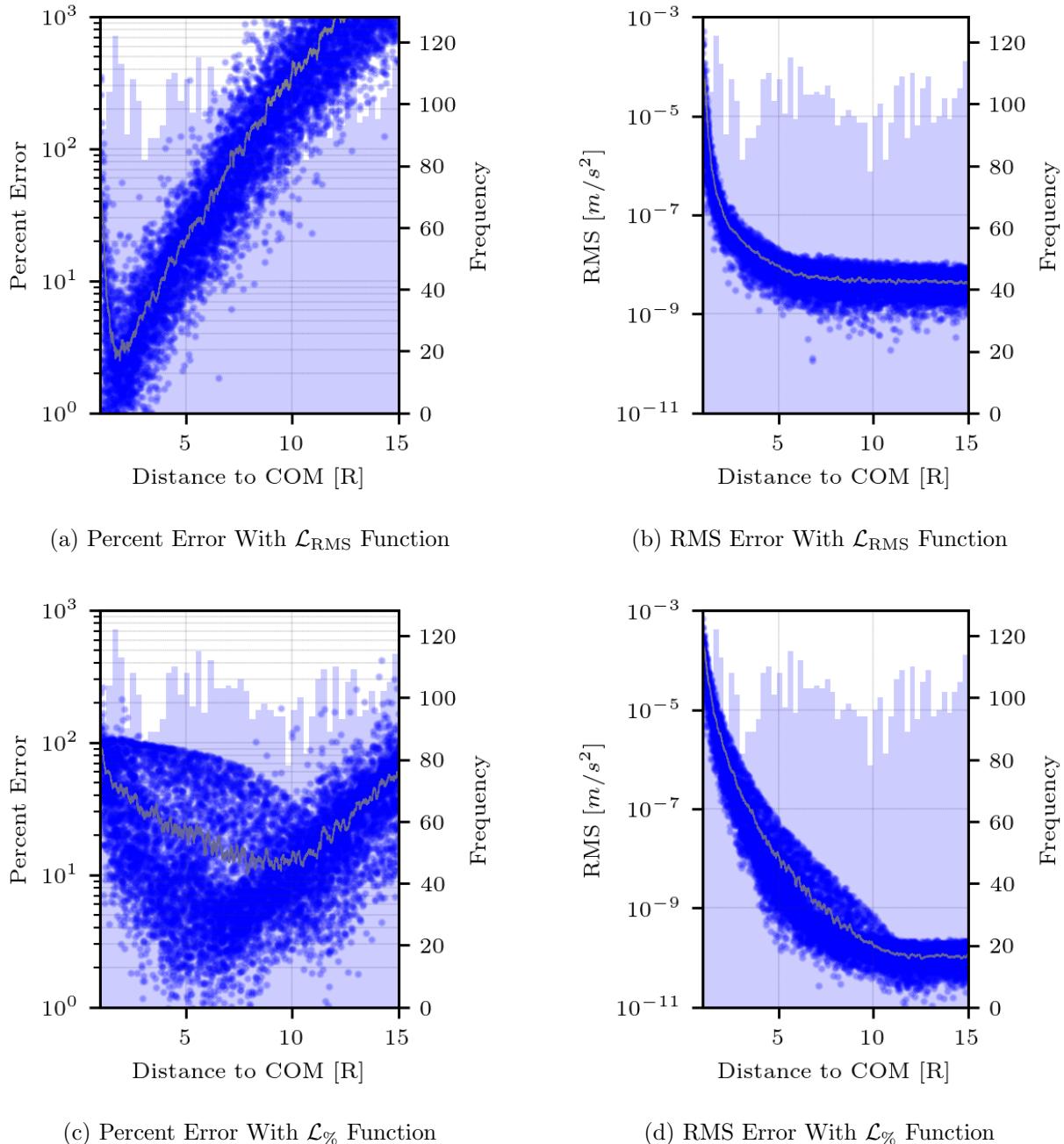


Figure 3.26: Different loss function change network performance at high and low altitudes. Blue points are the individual errors of the test data, the blue histogram is the distribution of training data, and the gray line is the average test error within a sliding window of 100 points.

viable.

PINN-GM-III addresses this phenomenon by learning a more numerically favorable representation of the potential that can be later transformed into the proper order of magnitude. Explicitly, PINN-GM-III proposes learning an altitude-invariant proxy to the potential,  $U_{\text{NN}}$ , defined as

$$U_{\text{NN}} = Ur^p \quad (3.26)$$

where  $U$  is the true non-dimensionalized potential function,  $r$  is the non-dimensionalized radius of the field point, and  $p$  is defined through

$$p = \begin{cases} l_r + 1 & l_r = 0 \vee 1 \\ l_r + 2 & l_r \geq 2 \end{cases} \quad (3.27)$$

where  $l_r$  is the maximum spherical harmonic degree included within the optional analytic part of the model,  $U_{\text{LF}}$  (Section 3.3.5).

$U_{\text{NN}}$  is designed to consistently span the domain of [-1,1] regardless of altitude. Unlike the true potential,  $U_{\text{NN}}$  has considerably lower risk of introducing numerical error and prematurely capping accuracy of the model at high altitudes. After the network produces its prediction for the proxy potential, it is then transformed into the true potential distribution later in the model through

$$\hat{U}_{\text{NN}} = \frac{U_{\text{NN}}}{r^p} \quad (3.28)$$

This ensures that the numerics of the network inference remain well-conditioned, while still allowing the model to produce the correct value / order of magnitude of the potential.

Figure 3.27 visualizes the difference between the true potential and the proxy potential as a function of altitude and Figure 3.28 shows the corresponding effect on model error. Specifically Figure 3.27a shows 10,000 values of the Earth potential<sup>6</sup> distributed between 0-5R (radii) from the Earth's surface. The perturbations are seen decaying at a rate of  $1/r^4$  consistent with Equa-

---

<sup>6</sup> Note that these values exclude contributions of a spherical harmonic model of degree and order 2 (i.e.  $\delta U = U - U_{\text{LF}}$  where  $U_{\text{LF}}$  is a degree  $l = 2$  spherical harmonic model).

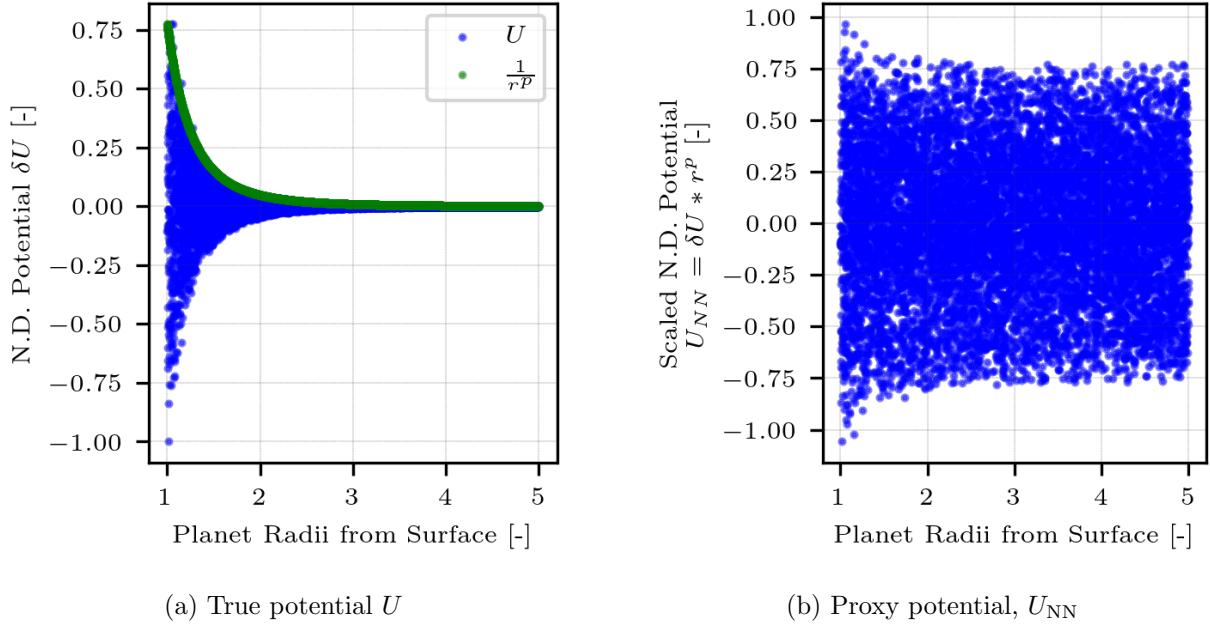


Figure 3.27: The true potential  $U$  (left) quickly decays to numerically unfavorable values whereas versus the proxy potential  $U_{NN}$  (right) remains numerically well-conditioned between  $[-1, 1]$ .

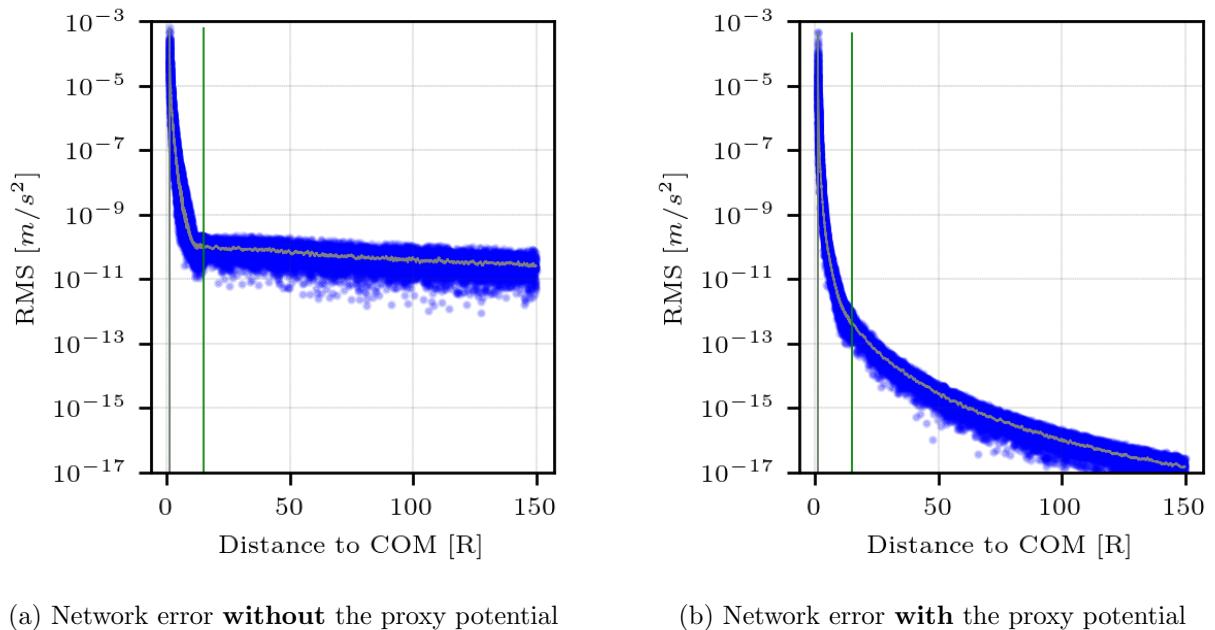


Figure 3.28: Effect of learning a potential proxy on the network inference error.

tion (3.27). In contrast, the proxy potential distribution is shown in Figure 3.27b. Note how the

values consistently exists in the more numerically favorable domain of  $[-1, 1]$  regardless of altitude.

The utility of this scaling within the PINN-GM-III is shown with an experiment where two PINN-GM are trained. Both models are trained with 100 samples randomly distributed from 0-15R from the Earth's surface. One network is trained to learn the potential directly, without the scaling proposed in Equation (3.26) (Figure 3.28a). The other PINN-GM is trained to learn the potential proxy which is later divided by  $r^p$  to transform it into the correct order of magnitude (Figure 3.28b). The RMS error of each model then is evaluated on 10,000 test points that span from 0-150R.

Figure 3.28a clearly shows that PINN-GMs trained to learn the potential directly hits a numerical barrier at approximately  $r = 12R$ . Beyond this critical altitude, the model numerics become poorly conditioned and are unable to represent these very small values — prematurely limiting the range of altitudes in which this model can be utilized. In contrast, when the PINN-GM is trained to learn a proxy potential which is later scaled to the correct order of magnitude through Equation (3.28) there is no longer a critical altitude or numerical floor that cannot be surpassed.

### 3.3.4 Enforcing Boundary Conditions to Avoid Extrapolation Error

While PINNs are most commonly designed to satisfy physics through their cost function, there are also other ways to enforce compliance through the design of the machine learning model itself. Reference (93) notes how machine learning models can be designed to seamlessly transition into known boundary conditions through the use of Heaviside-like functions.

The PINN-GM-III proposes the following design modification to enforce relevant boundary constraints:

$$\hat{U}(r) = (1 - H(r))\hat{U}_{\text{NN}}(r) + H(r)U_{\text{BC}}(r) \quad (3.29)$$

where  $\hat{U}_{\text{NN}}$  is the predicted gravitational potential,  $U_{\text{BC}}$  is the potential at the boundary condition,

and  $H(r)$  is a Heaviside-inspired function defined as

$$H(r) = \frac{1 + \tanh(k(r - r_{\text{ref}}))}{2} \quad (3.30)$$

where  $r$  is the radius of the field point,  $r_{\text{ref}}$  is a reference radius, and  $k$  is a smoothing parameter to control for a more continuous or discrete transition. Note that both  $r_{\text{ref}}$  and  $k$  can be a user-prescribed or learned parameter.

Equation (3.30) enforces that the PINN-GM must smoothly transition into a known boundary condition past the reference altitude  $r_{\text{ref}}$ . In this way, the machine learning model can leverage the modeling flexibility of the neural network to represent complex regions of the gravity field near the body for which the boundary condition is irrelevant, but then smoothly decrease the network's responsibility in the limit as the model approaches the boundary.

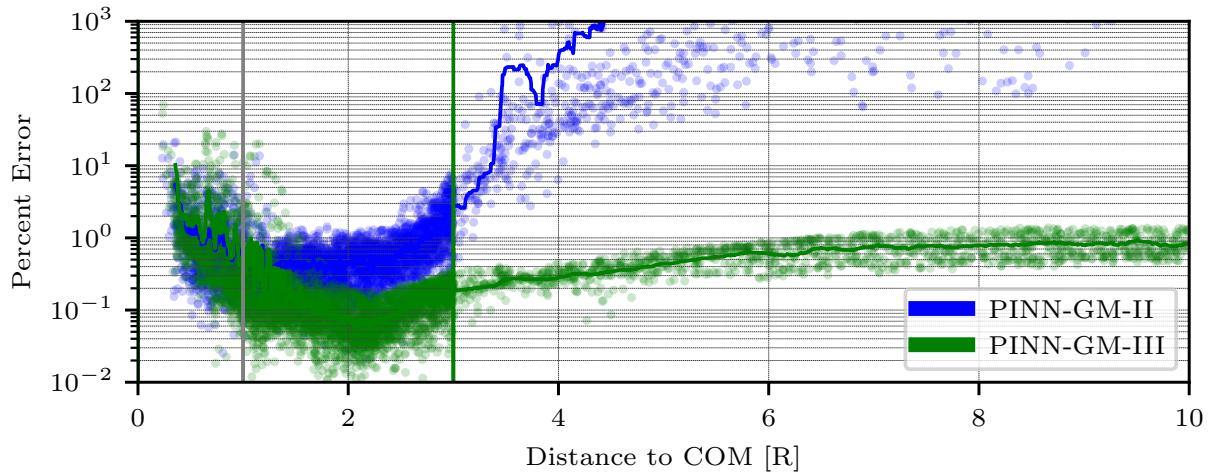
There are multiple ways in which the boundary condition can be enforced. In the limit of  $r \rightarrow \infty$ , the potential decays to zero as discussed in Section 3.3.3. However, setting  $U_{\text{BC}} = 0$  and  $r_{\text{ref}} = \infty$  in Equation (3.29) is not practical as it demands the neural network must learn a model of the potential for the entire domain  $r \in [0, \infty)$ . A more useful choice is to use a lower fidelity model in the limit, where higher accuracy is not necessary. PINN-GM-III accomplishes this by taking insights from the spherical harmonic gravity model and recognizing that high frequency components of the gravitational potential decay to zero more quickly than the point mass contribution at high altitudes. I.e.

$$U_{\text{BC}}(r) = U_{LF} = \frac{\mu}{r} + \sum_{l=0}^n \sum_{m=0}^l \frac{\mu}{r} \left(\frac{R}{r}\right)^l (\dots) \xrightarrow{0} 0 \quad (3.31)$$

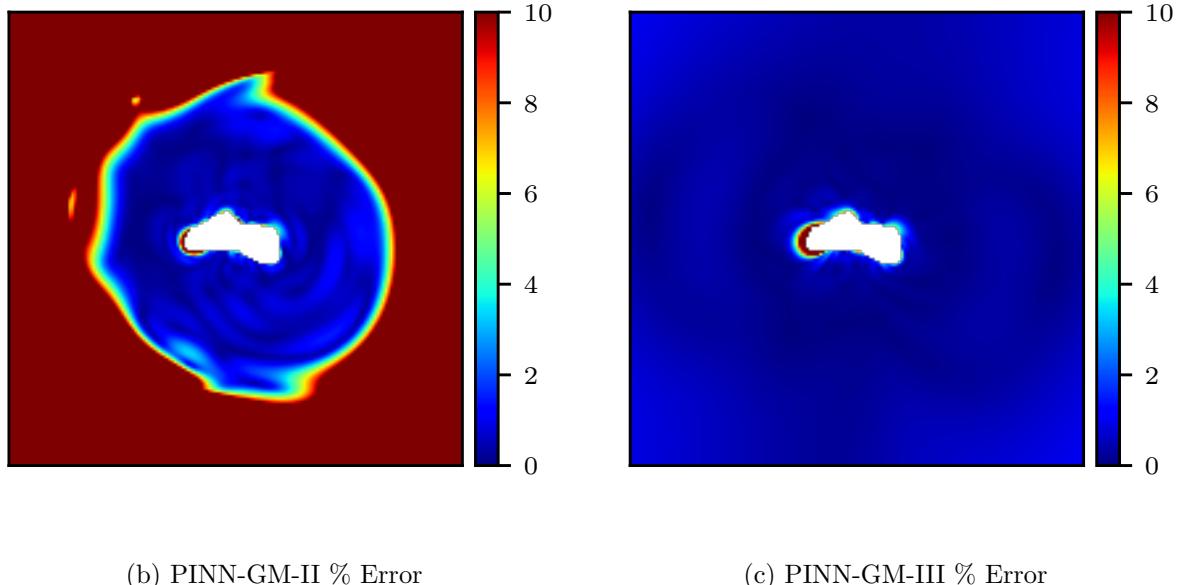
as  $r \rightarrow \infty$ .

This observation implies that  $U_{\text{BC}}(r)$  can be set to  $\frac{\mu}{r}$  assuming  $r \gg R$ . As such, the PINN-GM-III sets  $U_{\text{BC}} = \frac{\mu}{r} + f(r)$  in Equation (3.29), where  $f(r)$  are any higher order terms in the spherical harmonic gravity model that the user knows a priori and wishes to leave as part of the boundary condition.

To illustrate the effect of enforcing these boundary conditions, an experiment is performed where a PINN-GM-II and a PINN-GM-III are trained. PINN-GM-II does not contain any infor-



(a) PINN-GM-II and III performance inside ( $0R-3R$ ) and outside ( $3R-10R$ ) the training domain.



(b) PINN-GM-II % Error

(c) PINN-GM-III % Error

Figure 3.29: Top Row: Percent error of PINN-GM inside ( $0-3R$ ) and outside ( $3R-10R$ ) of the training domain. Gray vertical line is Brillouin radius. Green vertical line is maximum bounds of training data. Bottom Row: % Error of PINN-GM in XY plane.

mation about the boundary condition in its model design, whereas PINN-GM-III incorporates the proposed transition captured by Equation (3.29). Each network contains eight hidden layers with 10 nodes each, and both models are trained on the same 1,000 data points spanning 0-3R about the asteroid Eros. The average errors of both models are reported as a function of altitude from 0-10R in Figure 3.29a, and the XY Cartesian plane cross sections are show in Figures 3.29b and 3.29c.

Figure 3.29 shows that when Equation (3.29) is not included in the model design, the PINN-GM diverges when tested outside of the bounds of its training data. In contrast, when Equation (3.29) is included, PINN-GM-III is able to maintain low errors even at high altitudes for which there is no training data. This stark contrast in model behavior demonstrates the advantage of enforcing boundary conditions in the model design. PINN-GM-III is able to use the neural network to learn a rich gravity model representation within the bounds of the training data, but as soon as the model exits those bounds it is guaranteed to perform no worse than a point mass approximation or a low-fidelity spherical harmonic model. This is a powerful development for these machine learning gravity models as it showcases that these models can be reliably used beyond the bounds of their training data and will always perform as well or better than their low-fidelity counterpart.

### 3.3.5 Leveraging Preexisting Gravity Information into PINN-GM Solution

In addition to enforcing the boundary condition through the model design, another design choice enabled by the PINN-GM-III is the ability to fuse an a priori gravity model with the neural network solution. For example, most large celestial bodies exhibit planetary oblateness which is succinctly captured with the  $C_{2,0}$  spherical harmonic coefficient as discussed in Chapter 2. Rather than requiring the network to relearn this prominent and easily observable perturbation, this information can be directly embedded into the network model. In this way, the PINN-GM-III can predict accelerations by leveraging a low-fidelity, first-order analytic model with a network responsible for capturing high-order perturbations through:

$$\hat{U}(r) = (1 - H(r))(U_{\text{LF}}(r) + U_{\text{NN}}(r)) + H(r)(U_{\text{LF}}(r)) \quad (3.32)$$

where  $U_{\text{LF}}$  refers to the known, low-fidelity analytic model such as  $U_{\text{LF}} = \frac{\mu}{r} + U_{J_2}$ .

---

**Algorithm 1:** PINN-GM-III algorithm

---

- 1 Collect training data ( $\mathbf{x}, \mathbf{a}$ ) from:
    - (a) a pre-existing model
    - (b) online state estimates (94) Non-dimensionalize the training data // Sec. 3.3.1
  - 2 Convert to spherical coordinates ( $r_{\text{clip}}, 1/r, s, t, u$ ) // Sec. 3.3.1
  - 3 (Optional) Encode inputs into Fourier space // App. C
  - 4 Propagate through the trained neural network Output proxy potential // Sec. 3.3.3
  - 5 Scale proxy potential into true potential // Sec. 3.3.3
  - 6 Enforce boundary conditions on the network potential // Sec. 3.3.4  
// Autodifferentiate (AD) potential to produce acceleration  $\hat{\mathbf{a}}$
  - 7 if Training then
    - 8 Add acceleration percent error to the loss // Sec. 3.3.2
    - 9 (Optional) Compute jacobian for  $\hat{\mathbf{a}}$  via AD (Optional) Perform learning rate annealing // App. B
    - 10 (Optional) Add annealed  $\nabla^2 \hat{U}$  term to the loss function Compute gradients of loss function Update network parameters
- 

### 3.4 PINN-GM-III Performance: Comparative Study

In this section, the performance of the PINN-GM-III is compared to that of its two model predecessors, PINN-GM-I and PINN-GM-II. The models are trained in the environments of their original papers, PINN-GM-I trained to represent the gravity field of the Earth, and PINN-GM-II trained to represent the asteroid 433-Eros. In both cases, the models' performance are evaluated across various key hyperparameter configurations including the number of training data, the number of training epochs, and the total model capacity. For the sake of legibility, PINN-GM-I, PINN-GM-II, and PINN-GM-III will be referred to as PINN-I, PINN-II, and PINN-III respectively.

Three model capacities are compared in each experiment. The model capacity is defined by the number of nodes per hidden layer,  $N_{\text{nodes}} = \{20, 40, 80\}$ , for which there are a total of 8 hidden layers. Each model is trained 25 times, each with a different number of training data and training epochs drawn from  $N_{\text{data}} = \{2^{10}, 2^{11}, 2^{12}, 2^{13}, 2^{14}\}$  and  $N_{\text{epochs}} = \{2^{12}, 2^{13}, 2^{14}, 2^{15}, 2^{16}\}$  respectively. Remaining details regarding the PINN-III's training are found in Appendix A.

### 3.4.1 Generation I Versus Generation III

The first experiment investigates the performance of the PINN-III as compared to the original PINN-I. Each network is trained on data generated from Earth's high-fidelity static gravity field model: Earth Gravity Model 2008 or EGM 2008 (95). This model extends to degree and order 2,160 but this experiment is limited to degree and order 1,000 ( $\sim 1$  million parameters) to ensure the fidelity is sufficiently high to capture the high-frequency perturbations.

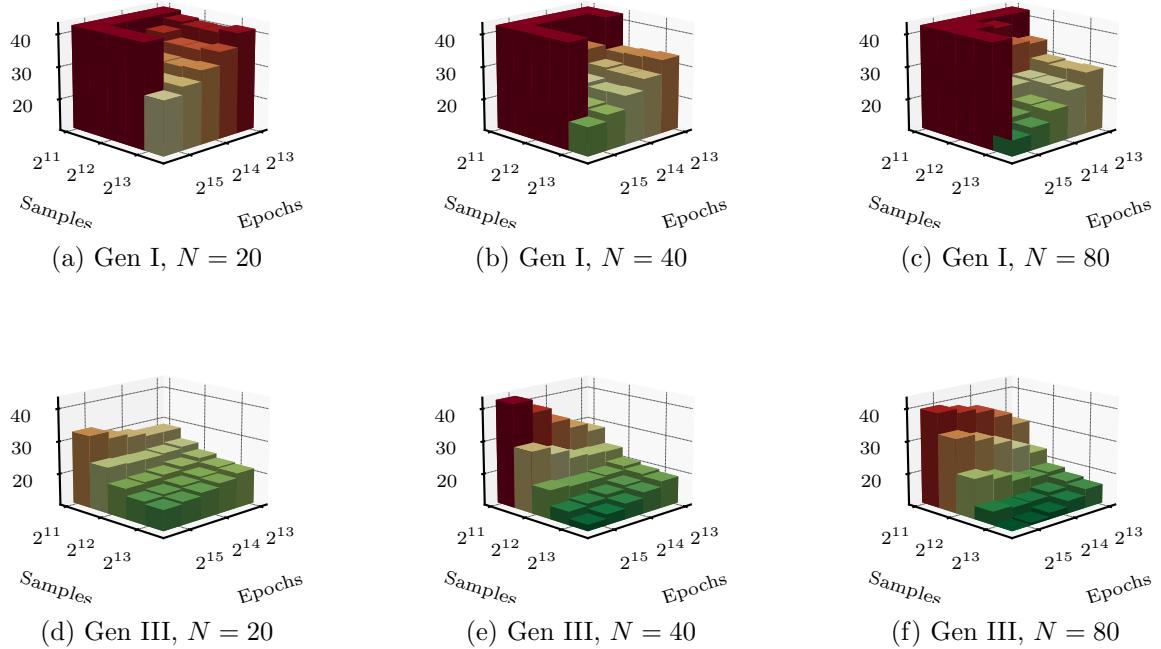


Figure 3.30: Average acceleration percent error defined in Equation (3.33) of PINN-GM-I (top) and PINN-GM-III (bottom) trained on Earth.

The training data for each network are distributed randomly from the Earth's radius to an approximately LEO altitude of 420 km. Once the field points are selected, their corresponding accelerations are computed using EGM2008 and together form the training data. The remaining hyperparameters for the PINN-III network are included in Table A.1 and the default hyperparameters for PINN-I are found in Table 3.1.

The performance of each network is evaluated using a mean percent error metric computed on

a separate test set of 50,000 randomly distributed field points within the same bounds as the training data. Note that the percent error metric purposefully excludes the acceleration contributions from the Earth's point mass and planetary oblateness through:

$$\mathcal{P} = \frac{1}{N} \sum_{i=0}^N \frac{|(\hat{\mathbf{a}}_i - \mathbf{a}_{i,2}) - (\mathbf{a}_i - \mathbf{a}_{i,2})|}{|\mathbf{a}_i - \mathbf{a}_{i,2}|} * 100 \quad (3.33)$$

where  $\hat{\mathbf{a}}$  are the network accelerations produced via  $-\nabla U$  and  $\mathbf{a}_{i,2}$  are the accelerations produced by the point mass and planetary oblateness terms of the spherical harmonic gravity model. The omission of  $\mathbf{a}_2$  from the percent error makes it easier to assess how well the PINN-GM is capturing the remaining high-order perturbations like mountain ranges and tectonic plates.

Figure 3.30 shows that there is a considerable increase in performance between PINN-I and PINN-III. For the low capacity cases, where the number of nodes per layer is 20 (model size of  $\approx 3,000$  parameters), PINN-I consistently produces low-accuracy predictions ( $\geq 35\%$  error). In the low-data cases, some PINN-I even diverge. In contrast, PINN-III consistently converges to a solution with errors less than 30% even in the low-data cases. Similar findings manifest for the higher capacity cases of  $N = \{40, 80\}$ , with PINN-III consistently outperforming its predecessor across all data sizes and training durations.

Notably, these results not only imply that the PINN-III is able to achieve considerably lower error than PINN-I, but also that it can be achieved with less data, less training time, and smaller models. Consider the fact that the 20 node PINN-III model trained with  $N_{\text{data}} = 1024$  and  $N_{\text{epochs}} = 4096$  (Figure 3.30d smallest sample, fewest epochs) has a near identical error to the 80 node PINN-I model with  $N_{\text{data}} = 32,768$  and  $N_{\text{epochs}} = 8192$  (Figure 3.30a most samples, second fewest epochs). **This suggests that a PINN-III that is 94% smaller, trained with 97% less training data, and for 50% fewer epochs is able to achieve the same accuracy as PINN-I.**

One additional noteworthy result from Figure 3.30 is that high capacity PINN-III models can experience overfitting. When there is insufficient data ( $N_{\text{data}} = 1024$ ), the error begins to increase for the longer training durations as best exhibited in Figures 3.30e and 3.30f. While the

overfitting of PINN-III remains less problematic than what is seen with the PINN-I, this can be trivially remedied by adding a small amount of dropout to each layer, incorporating the Laplacian and curl physics constraints into the cost function, or simply by adding more training data. The dropout and additional constraints are purposefully not included in this experiment as they act as regularizing terms which can obfuscate the maximum performance of these networks in the low-data conditions.

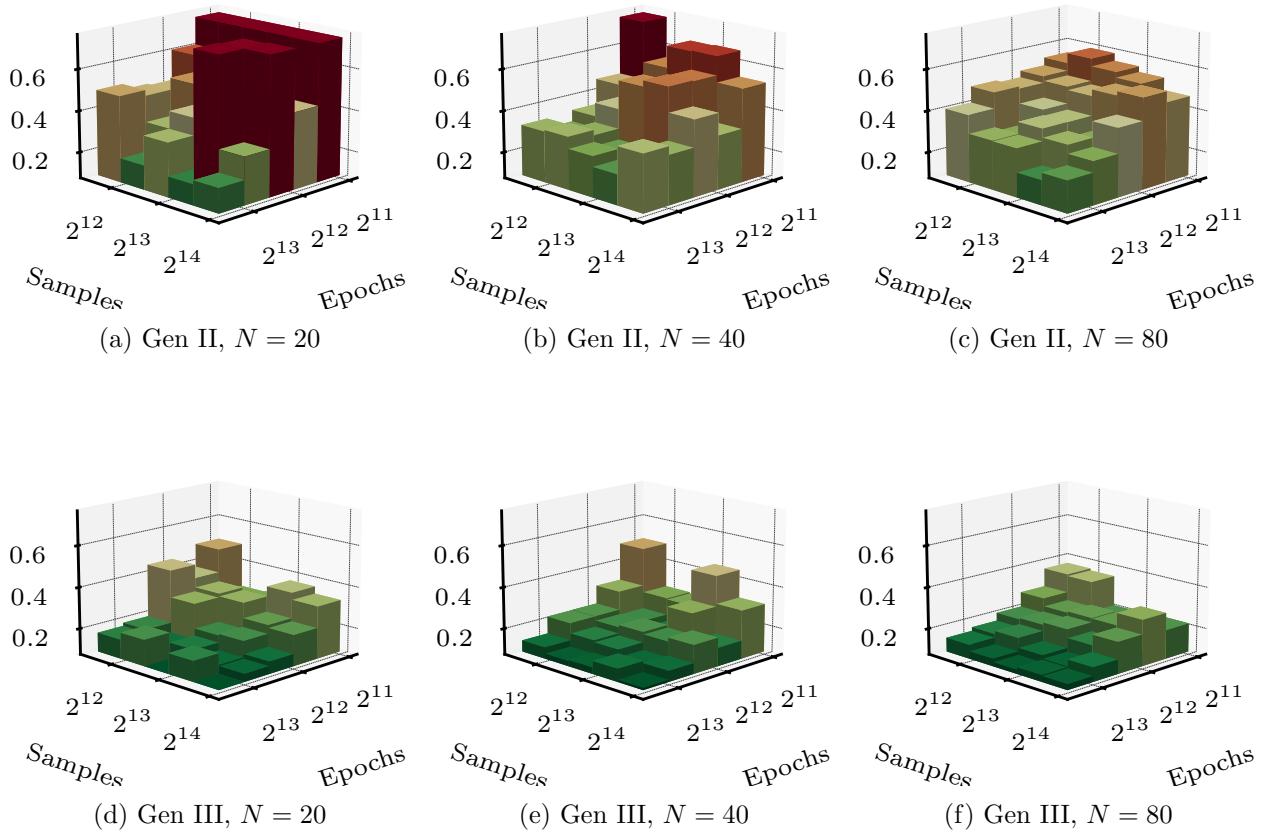


Figure 3.31: Average acceleration percent error of PINN-II (top) and PINN-III (bottom) trained on Eros.

### 3.4.2 Generation II versus Generation III

The second experiment investigates the improvements of the PINN-III over its more recent predecessor PINN-II. In this experiment, the testing environment transitions to the asteroid 433-Eros for which the PINN-II was originally developed. Rather than using a spherical harmonic gravity model, the truth accelerations are computed using a polyhedral model based on a low-resolution shape model of Eros comprised of approximately 11,000 total facets and vertices. The training data are distributed uniformly between the surface of the asteroid and a maximum radius equal to three radii from the center of mass.

Figure 3.31 illustrates the acceleration error of PINN-II and PINN-III. Note that the percent error reported is of the full acceleration vector ( $|\hat{\mathbf{a}} - \mathbf{a}|/|\mathbf{a}|$ ) rather than with the point mass and oblateness terms removed. While PINN-II is more robust than PINN-I, converging to a reasonable solution in all cases tested, PINN-III continues to outperform it. While both PINN-II and III maintain an error of  $< 1.5\%$ , PINN-III consistently remains below  $< 0.5\%$  improving performance by a factor of 3. PINN-III is also able to achieve better performance with less data, training time, and smaller model sizes. The smallest models of PINN-III produced errors comparable to the majority of hyperparameters tested with the 80 node PINN-IIs.

## 3.5 PINN-GM-III Performance: Heterogeneous Density Asteroid

One challenge modeling gravity fields of small-bodies is the uncertainty surrounding the asteroid's or comet's internal density distribution. Recent work has shown that the constant density assumption often employed by the polyhedral gravity model is not always valid (44). Some asteroids may contain over and under dense regions within their interior, or may have been formed by two asteroids merging together, each with different densities. Unfortunately the internal structure of these small bodies is extremely difficult to measure, as it requires directly probing the interior of the object. As a less intrusive alternative, dynamicists currently make informed assumptions about these distributions based on the asteroid's gravity field and shape and compare the measured

spherical harmonic coefficients against those that would be generated by their assumed profile (73). At best, this process leaves researchers with heuristic assessments of the body's density profile which can be used in heterogeneous forms of the polyhedral model (73). However, it remains common practice to simply proceed with a constant density assumption. This section explores the ramifications of this choice and highlights how the PINN-GM can bypass many of the consequences of this practice.

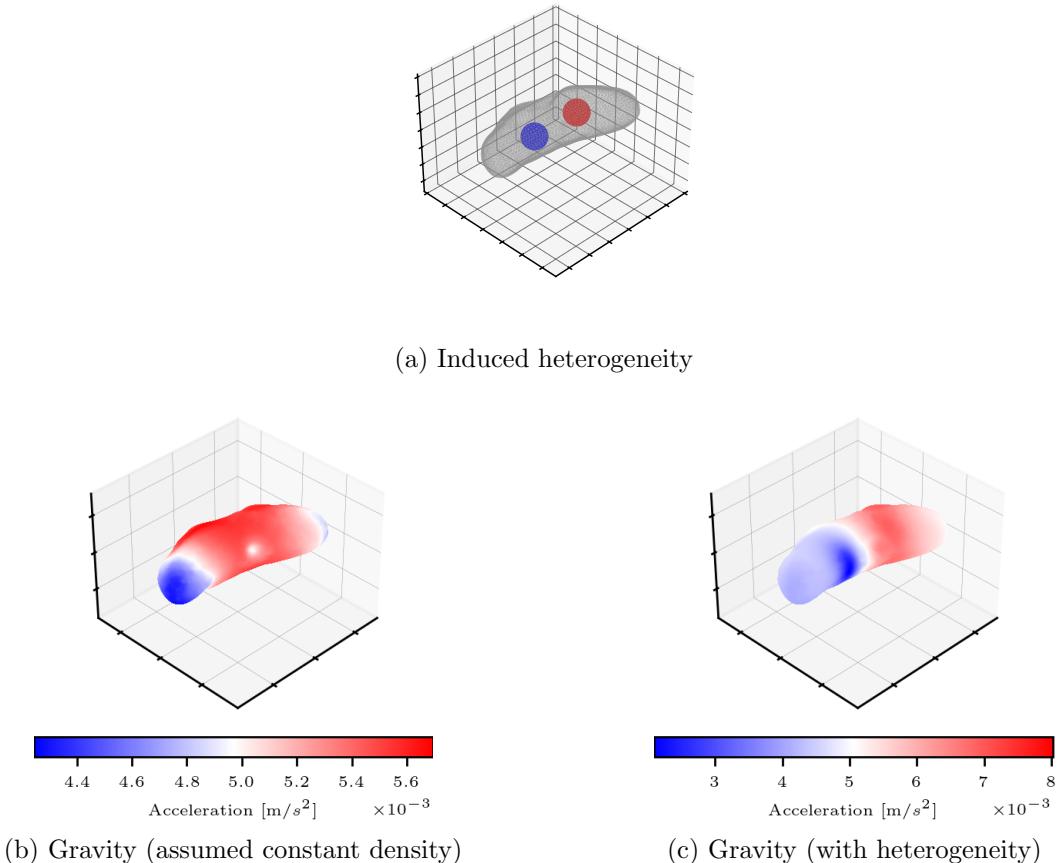


Figure 3.32: Asteroid with a mass heterogeneity (top) and the corresponding surface gravity field when constant density is assumed versus the true field (bottom left vs. bottom right respectively).

For this experiment, a small mass heterogeneity is introduced to the asteroid 433-Eros. In one hemisphere, a mass element is added to the interior of the body, and in the other hemisphere, a mass element is removed. Each mass element contains 10% of the total mass of the asteroid,

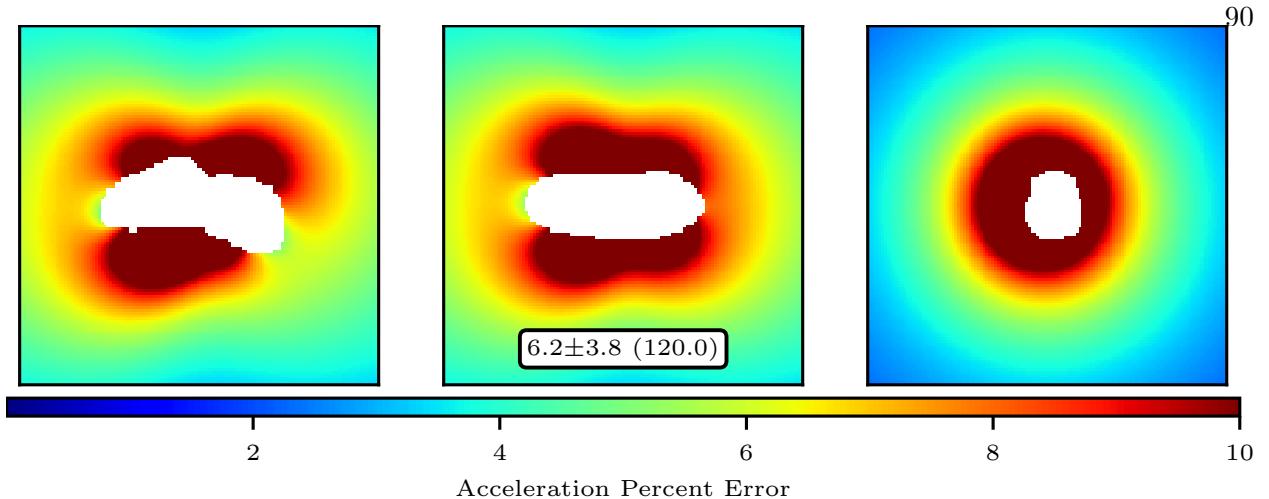


Figure 3.33: Constant density polyhedral acceleration error reported as  $\mu \pm \sigma$  (max).

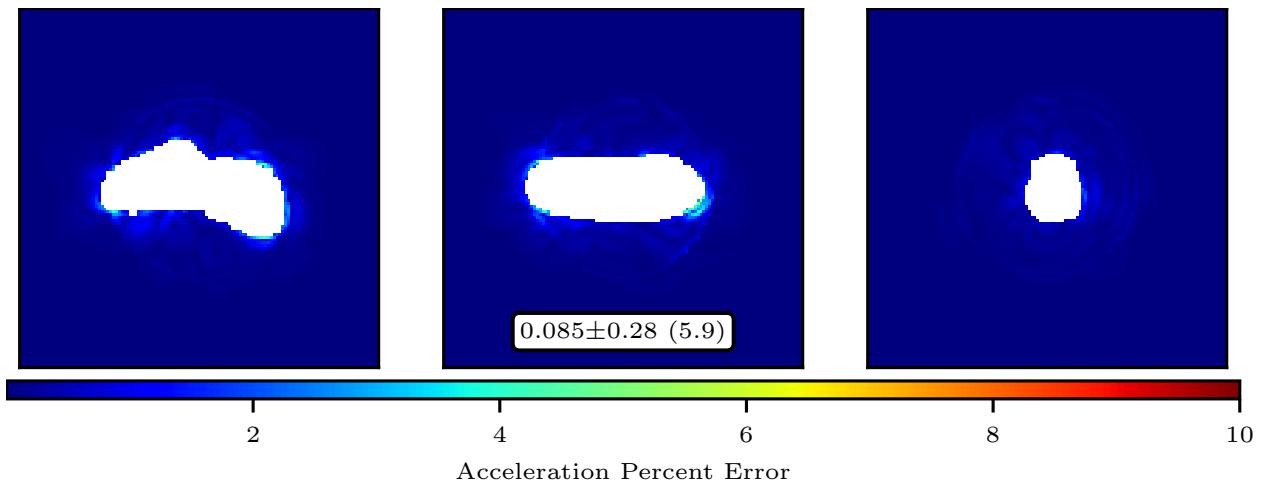


Figure 3.34: PINN-GM-III acceleration error reported as  $\mu \pm \sigma$  (max).

and they are symmetrically displaced along the x-axis by 0.33R (see Figure 3.32a). This choice is meant to emulate a scenario where two asteroids with different densities may have merged into one. The choice to make each mass element 10% of the total mass is motivated based on literature with similar candidate density distributions (96).

This small heterogeneity can have large implications for the asteroid's gravity field. Figure 3.32b depicts the field under the assumption of constant density, whereas Figure 3.32c shows the true, heterogeneous density gravity field. From these figures, it is apparent that the constant density assumption can lead to vastly different gravitational signatures and, by association, space-

craft trajectories.

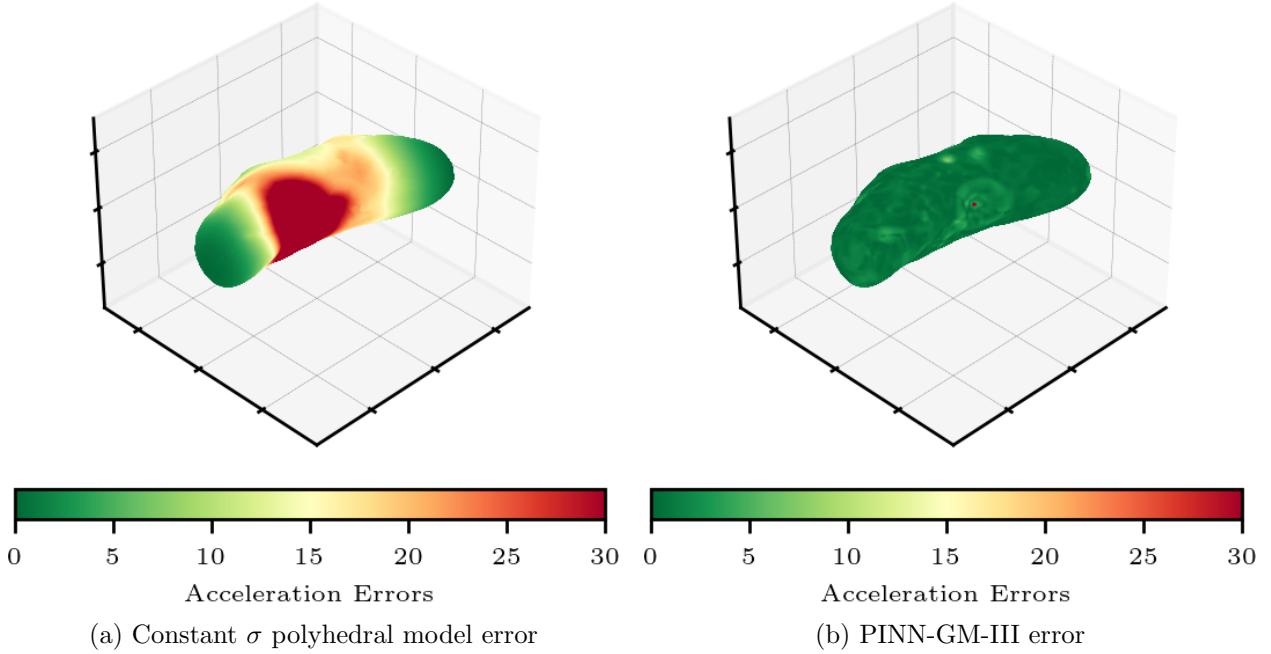


Figure 3.35: Surface error of the constant density polyhedral gravity model and PINN-GM-III.

Figure 3.33 illustrates the acceleration error of a constant density polyhedral model on the XY, XZ, and YZ planes out to  $3R$  and reports the average, standard deviation, and maximum acceleration error. The constant density assumption becomes increasingly invalid near the surface, producing maximum errors in excess of 100%. Errors of these magnitudes introduced considerable risk for missions that aspire to land or operate in close proximity to the asteroid.

As an alternative, a PINN-GM-III is trained to represent gravity field of the heterogeneous density asteroid. Explicitly, the training data for the PINN-GM is the superset of two datasets: the first dataset contains data exclusively sampled on the perimeter of the shape model where each data point's position corresponds with one facet of the polyhedral shape model (200,700 facets). The second dataset is one million data points spanning from the surface of the asteroid out to 10 radii. While experiments in Section 3.4.2 studied data-sparse performance of the PINN-GM-III, this experiment purposefully studies performance in a data rich environment, when ample training

samples exists.

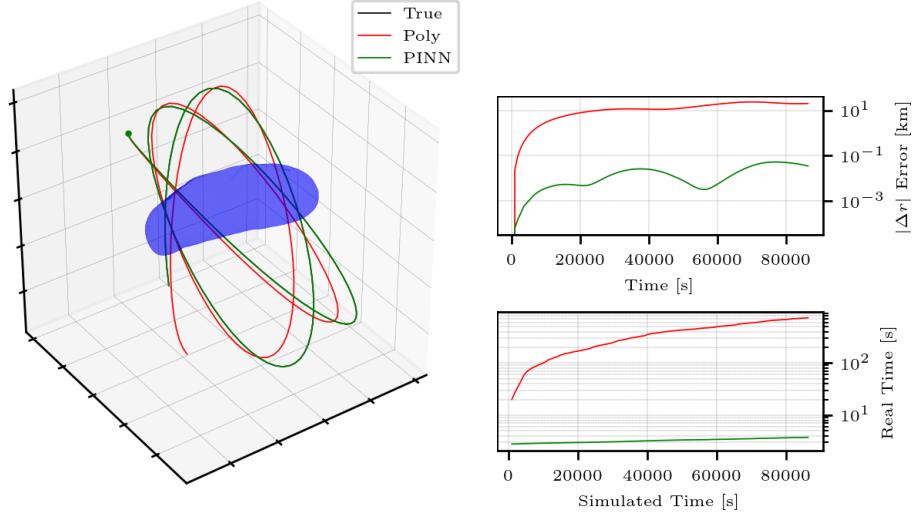


Figure 3.36: Propagation speed and error of the constant density polyhedral model and PINN-GM-III.

The performance of the trained PINN-GM is visualized in Figures 3.34 and 3.35. The average and maximum error of the PINN-GM is nearly two orders of magnitude lower than the constant density polyhedral model at both high altitudes and at the surface. Note that the PINN-GM-III only contains approximately 3,000 parameters in contrast to the polyhedral model, which includes over 200,000 facets and 100,000 vertices. These results demonstrate that the PINN-GM-III is able to achieve an order of magnitude reduction in error using two orders of magnitude fewer parameters in the model.

Not only is the PINN-GM-III more accurate and compact, it is also more computationally efficient. This is demonstrated in Figure 3.36 in which a set of initial conditions is propagated once using the constant density polyhedral model, and once with the trained PINN-GM-III. The figure shows that propagating the spacecraft for one simulated day requires approximately 30 minutes in real time, whereas the PINN-GM-III requires a mere three seconds — a 600x speedup. Moreover, Figure 3.36 also highlights the risk of making the constant density assumption when orbiting exotically shaped small bodies. After one day of propagating the state under a constant density polyhedral gravity model, the trajectory evolves to have position errors norms in excess of

20 km from the true trajectory. The PINN-GM, in contrast, makes no assumptions and produces errors norms of  $< 0.1\text{km}$ .

## Chapter 4

### Application I: Reinforcement Learning

Reinforcement learning is framework used to train decision making agents to perform some complex behavior. This can manifest as teaching cars how to drive themselves (97), building recommender systems that suggest which netflix show to watch next (98), or even training chess bots how to outperform international grandmasters (99). At it's core, reinforcement learning is performed by having an agent repeatedly take actions within some environment and receive a reward signal indicating if the action was good or bad. For example, if a chess bot performs a move which puts their opponent in check, that would receive a positive reward. In contrast, if the bot performs a move that jeopardizes their own king's safety, that would receive negative reward. By continuously trying different moves and estimating the associated long-term reward, the reinforcement learning agent builds an history of experiences from which it can learn more sophisticated and desirable behavior.

Reinforcement learning is extremely general, and there exist a wide range of possible applications which has garnered the attention of many scientific and engineering communities (100; 101). In many cases, these communities aspire to train agents to learn more complex behaviors or solutions that outperform current state-of-the-art methods. The field of astrodynamics is no exception. In the past few years, multiple researchers have begun to explore how reinforcement learning can be used to produce more flexible and robust autonomous spacecraft for deep space exploration. These efforts include investigating how spacecraft can design more robust guidance algorithms capable of handling environmental uncertainty (102; 103), identifying candidate transfers trajectory-

ries (104; 105), or even building operational schedules (106; 107).

One of the key challenges of reinforcement learning is how much time it can take to train an agent. These agents need to learn through repeated trial and error, requiring many interactions with the environment before they can begin to learn desirable behavior. Oftentimes it is too time-consuming or too risky to conduct these trials in the real world. In the case of self-driving vehicles, untrained agents cannot be simply be deployed on the street, as they would risk the safety of other vehicles and pedestrians. To bypass these challenges, agents are typically trained within simulated, digital environments. These simulations often allow the agents to be trained in faster-than-real time, though this can come at the cost of real world fidelity. This tradeoff manifests as something referred to as the sim-to-real gap: the inevitable mis-modeling of reality through simulation, where the environmental conditions of real life cannot be perfectly captured (108).

While the sim-to-real gap exists for all simulated environments, it is often most apparent when the environment is too expensive to model and therefore must be approximated. Consequently, reinforcement learning practitioners are forced to trade fidelity for speed. This begs the question: are low-fidelity approximations of the system sufficient for the agent to learn proper behavior, or do these approximations risk the agent learning erroneous behavior that can jeopardize its safety?

Safety and reliability are major concerns for many problems, though this is especially true for autonomous spacecraft. Many of these missions cost hundreds-of-millions of dollars, and if the spacecraft behaves in an unsafe manner, there are not simple opportunities for repair and recovery. Consequently, there is justifiable caution around using reinforcement learning agents on-board spacecraft. To mitigate potential scrutiny, it becomes the responsibility of researchers to ensure that their autonomous spacecraft agents are trained in simulated environments that are as close to ground truth as possible. Here-in-lies the value of the PINN-GM for reinforcement learning.

Spacecraft dynamics are heavily influenced by the force of gravity and simulating high-fidelity gravity fields is notoriously expensive. Propagating orbits with high-resolution polyhedral gravity models can take days of compute. For reinforcement learning agents that need hundreds-of-thousands of interactions with their environment, these runtimes are prohibitive and demand that

researchers use lower-fidelity options. The PINN-GM offers a potential remedy to this problem. Capable of achieving both high-accuracy and fast-runtimes, the PINN-GM presents a powerful way to enhance the fidelity of simulated environment in which reinforcement learning agents can be trained.

To demonstrate the utility of the PINN-GM for spacecraft reinforcement learning, this section introduces a problem scenario centered around spacecraft safe mode. Imagine, a spacecraft is in orbit about a small-body and has just begun a complex operation such as executing a touch-and-go (T.A.G.) maneuver. In the middle of this operation, the spacecraft experiences an unexpected loss of communication with the ground. This anomaly triggers the spacecraft to enter Safe Mode — a power-positive, stationary mode that allows engineers to investigate the problem, propose a fix, and uplink the solution (109). For spacecraft orbiting large-celestial bodies, enabling Safe Mode is considered a robust and risk-free action. Because the dynamics experienced by spacecraft in orbit around near-spherical bodies is primarily keplerian, it is extremely unlikely the vegetative spacecraft will deviate from its reference trajectory and collide with the body over a short time span. In small-body exploration, such guarantee is far less apparent.

The gravity fields produced by irregularly shaped asteroids do not provide the same nominally stable orbits that are present around large celestial bodies. Instead, spacecraft experience complex gravitational accelerations coupled with the effects of solar radiation pressure which can produce chaotic trajectories around the small-body without proper station-keeping. Consequently, entering a traditional Safe Mode during a mission critical operation like T.A.G. dramatically increases the odds that the spacecraft will collide with the body.

To minimize this risk of collision, an Enhanced Safe Mode agent is proposed. When the spacecraft enters Enhanced Safe Mode, it will trigger an agent that must prioritize three safety objectives. First, the spacecraft must not collide with the body. Second, the spacecraft must conserve fuel. Third and finally, the spacecraft must remain close to the body such that the gravity field remains the dominant perturbation. These safety objectives ensure that, regardless of mission phase, the spacecraft will not intersect the body and will maneuver onto orbits that require relatively

little station-keeping. In principle, once this Enhanced Safe Mode is executed and a quasi-stable orbit is found, traditional Safe Mode operations can ensue and operators can return to diagnosing the original problem.

The challenge with designing an Enhanced Safe Mode agent is that there does not exist a control solution known a priori which will satisfy these high-level safety constraints. As such, reinforcement learning is proposed to learn a policy capable of converting these high-level safety constraints into an rapidly executable control solution for the spacecraft.

#### 4.1 Markov Decision Process Formulation

To solve the reinforcement learning problem, the following the Markov Decision Process (MDP) of  $(S, A, T, R, \gamma)$  is proposed where  $S \in \mathcal{S}$  is the state,  $A \in \mathcal{A}$  is the set of actions,  $T(s'|s, a)$  is the transition function,  $R(s, a)$  is the reward at state  $s$  when action  $a$  is taken, and  $\gamma$  is the discount factor. Reinforcement learning seeks to learn a policy  $\pi : \mathcal{S} \rightarrow \mathcal{A}$  which maximizes the expected return

$$R = \mathbb{E} \left[ \sum_{t=0}^T \gamma^t r_t \right] \quad (4.1)$$

For Enhanced Safe Mode, the state space is defined  $\mathcal{S} : \mathbb{R}^4 \times \mathbb{R}^3 \times \mathbb{R}$  where an instance of the state is

$$\mathbf{s} = (\bar{\mathbf{r}}, \bar{\mathbf{v}}, \bar{m}_f) \quad (4.2)$$

where  $\bar{\mathbf{r}}$  is the normalized position vector expressed as  $(\bar{r}, s, t, u)$  where  $\bar{r}$  is the radial distance of the spacecraft with respect to the asteroid center of mass, normalized by the maximum radius of the asteroid  $R$ . The remaining position coordinates,  $s, t, u \in [-1, 1]$ , are the sines of the angles between the cartesian basis vectors  $\hat{\mathbf{x}}, \hat{\mathbf{y}}, \hat{\mathbf{z}}$  respectively. Likewise  $\bar{\mathbf{v}}$  represents the velocity vector of the spacecraft  $\bar{v}_x, \bar{v}_y, \bar{v}_z$  normalized by the escape velocity as defined at the Brillouin sphere of the asteroid:

$$v_{\text{norm}} = \sqrt{2 \frac{\mu}{R}} \quad (4.3)$$

Finally  $\bar{m}_f$  is the remaining fuel in the spacecraft normalized by the fuel tank capacity. The

action space for the MDP,  $\mathcal{A} : \mathbb{R}^3$ , is continuous and represents impulsive  $\Delta V$ s that can be applied instantaneously at the beginning of every simulation step. The magnitude of  $\Delta V$  is constrained to  $\pm 10$  centimeters per second in each cartesian direction. The reward function is defined such that failure is heavily penalized, and the spacecraft is incentivized to remain near the asteroid through

$$R(s, a) = \begin{cases} -100 & \text{if } \text{failure}(s, a) \\ 1 - \bar{r} & \text{otherwise} \end{cases} \quad (4.4)$$

where failure is defined as:

$$\text{failure}(s, a) = (\bar{r} \in \text{asteroid}) \vee (\text{fuel} < 0) \vee (\bar{r} < 3R) \quad (4.5)$$

The transition function  $T(s'|s, a)$  is defined using the gravitational dynamics of the system. The acceleration is provided by one of the aforementioned gravity models, the change in fuel is governed by the rocket-equation, and the  $\Delta V$  generated by the action is applied instantaneously. The discount factor  $\gamma$  is 0.99.

## 4.2 Soft-Actor Critic

A Soft Actor-Critic (SAC) algorithm is used to solve this MDP (110). Actor-critic algorithms use two cooperative function approximators to generate a policy which maximizes the value function of the MDP. The value function  $V(s)$  is the expected return of the MDP (Equation 4.1). Given that the value function is not originally known at runtime, a function approximator — often a neural network — is used instead and is referred to as the critic or  $V_\psi$ . A second function approximator forms the actor, or policy  $\pi_\phi$ , which takes actions that maximize the expected return estimated by the critic.

The difference between traditional actor-critic algorithms and soft actor-critic algorithms is that SAC augments the value function to include a maximum entropy term in addition to the expected return through

$$R^\star(\pi) = \mathbb{E}_\phi \left[ \sum_{t=1}^T r(s_t, a_t) - \alpha \log(\pi_\phi(a_t|s_t)) \right] \quad (4.6)$$

such that the actor not only seeks reward, but it also learns a policy that promotes diverse behavior. This helps to balance exploration and exploitation during training of the agent and also improves sample efficiency and decreases sensitivity to initial hyperparameters. SAC also have the ability to be trained in an off-policy manner (110).

Explicitly, Soft Actor-Critic methods make use of three function approximators:

(1)  $V_\psi(s_t)$  — the state value function (the critic)

(2)  $Q_\theta(s_t, a_t)$  — the soft Q-function or the state-action value function

(3)  $\pi_\phi(a_t | s_t)$  — the tractable policy (the actor)

Typically the state value function and soft Q-function are represented as neural networks such that the parameters  $\psi$  and  $\theta$  are the trainable weights and biases of the network.  $\pi_\phi$  is a Gaussian distribution with mean and covariance determined by neural networks.

The goal of the SAC algorithm is to optimize the following cost functions:

$$J_V(\psi) = \mathbb{E}_{\mathbf{s}_t \sim \mathcal{D}} \left[ \frac{1}{2} (V_\psi(\mathbf{s}_t) - \mathbb{E}_{\mathbf{a}_t \sim \pi_\phi} [Q_\theta(\mathbf{s}_t, \mathbf{a}_t) - \log \pi_\phi(\mathbf{a}_t | \mathbf{s}_t)])^2 \right] \quad (4.7)$$

$$J_Q(\theta) = \mathbb{E}_{(\mathbf{s}_t, \mathbf{a}_t) \sim \mathcal{D}} \left[ \frac{1}{2} (Q_\theta(\mathbf{s}_t, \mathbf{a}_t) - \hat{Q}(\mathbf{s}_t, \mathbf{a}_t))^2 \right] \quad (4.8)$$

$$J_\pi(\phi) = \mathbb{E}_{\mathbf{s}_t \sim \mathcal{D}} \left[ D_{KL} \left( \pi_\phi(\cdot | \mathbf{s}_t) \| \frac{\exp(Q_\theta(\mathbf{s}_t, \cdot))}{Z_\theta(\mathbf{s}_t)} \right) \right] \quad (4.9)$$

where  $\mathcal{D}$  is the distribution of state-action tuples taken thus far and stored in a replay buffer. Each of these cost functions can be updated using stochastic gradient descent. Specifically, the value and soft Q networks can be updated using:

$$\hat{\nabla}_\psi J_V(\psi) = \nabla_\psi V_\psi(\mathbf{s}_t) (V_\psi(\mathbf{s}_t) - Q_\theta(\mathbf{s}_t, \mathbf{a}_t) + \log \pi_\phi(\mathbf{a}_t | \mathbf{s}_t)) \quad (4.10)$$

$$\hat{\nabla}_\theta J_Q(\theta) = \nabla_\theta Q_\theta(\mathbf{a}_t, \mathbf{s}_t) (Q_\theta(\mathbf{s}_t, \mathbf{a}_t) - r(\mathbf{s}_t, \mathbf{a}_t) - \gamma V_{\bar{\psi}}(\mathbf{s}_{t+1})) \quad (4.11)$$

$$(4.12)$$

The policy distribution could be updated via likelihood ratio gradient estimator (avoids backpropagating gradients), but because the target density is the Q-function and can be differentiated, the

reparameterization trick is used instead to produce a lower variance estimator. This can be done by reparameterizing the policy using a neural network transformation

$$\mathbf{a}_t = f_\phi(\epsilon_t; \mathbf{s}_t) \quad (4.13)$$

where  $\epsilon_t$  is a noise vector. This allows the cost function to be rewritten as

$$J_\pi(\phi) = \mathbb{E}_{\mathbf{s}_t \sim \mathcal{D}, \epsilon_t \sim \mathcal{N}} [\log \pi_\phi(f_\phi(\epsilon_t; \mathbf{s}_t) | \mathbf{s}_t) - Q_\theta(\mathbf{s}_t, f_\phi(\epsilon_t; \mathbf{s}_t))] \quad (4.14)$$

and solved via

$$\hat{\nabla}_\phi J_\pi(\phi) = \nabla_\phi \log \pi_\phi(\mathbf{a}_t | \mathbf{s}_t) + (\nabla_{\mathbf{a}_t} \log \pi_\phi(\mathbf{a}_t | \mathbf{s}_t) - \nabla_{\mathbf{a}_t} Q(\mathbf{s}_t, \mathbf{a}_t)) \nabla_\phi f_\phi(\epsilon_t; \mathbf{s}_t) \quad (4.15)$$

Note that the update process uses a target value network  $V_{\bar{\psi}}$  where the parameters are an exponentially moving average of the value network weights, and

$$\hat{Q}(\mathbf{s}_t, \mathbf{a}_t) = r(\mathbf{s}_t, \mathbf{a}_t) + \gamma \mathbb{E}_{\mathbf{s}_{t+1} \sim p} [V_{\bar{\psi}}(\mathbf{s}_{t+1})] \quad (4.16)$$

Additional implementation details about soft actor-critic can be found in the original paper (110).

### 4.3 Environment

The environment used to train the Enhanced Safe Mode agent is configured as follows: For every episode, the spacecraft is initialized to a randomized position which exists between  $[0.6R, 3R]$ , where  $R$  is the maximum radius of the asteroid. The initial velocity is also randomized, but bound in magnitude to less than  $\sqrt{3}$  meters per second. These initial conditions are designed to replicate states found during T.A.G. mission phases, where if no actions are taken, the spacecraft will likely collide with the asteroid. A set of these initial conditions propagated without an Enhanced Safe Mode are shown colliding with the body in Figure 4.1.

At the beginning of training, a replay buffer is initialized with 10,000  $(s, a, r, s')$  tuples to warm-start training. These tuples are generated by evolving initial conditions with keplerian dynamics and a policy that applies random  $\delta V$  impulses. While not representative of the true system

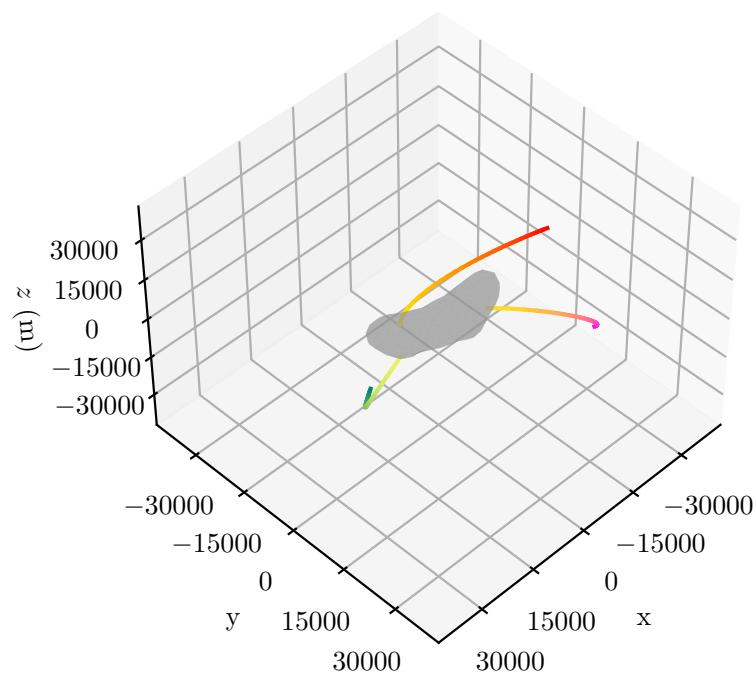


Figure 4.1: Trajectories taken **without** the Enhanced Safe Mode agent enabled. The darker shades correspond with earlier parts of the trajectory.

dynamics, the state-action pairs in the initial buffer help stabilize learning and speed the convergence of the value network — building a coarse knowledge of the system dynamics and identifying general domains in the state space with high and low value. Once the simulation begins to execute, the replay buffer is populated with tuples generated by the true simulation environment and the agent in training. The buffer continues to fill until 100,000 tuples are stored, after which tuples will be loaded into and out of the replay buffer using the first-in, first-out paradigm.

The actor and critic networks are trained for one epoch using a 1024 mini-batch size after a new tuple is loaded into the replay buffer / an additional step is taken in the environment. The length of a step corresponds to 10 minutes in simulation time and the maximum duration of an entire episode is 10 simulation hours. After every step, the impulsive  $\Delta V$  action produced by the actor is applied to the state. After every 1,000 steps in the environment, the performance of the policy is evaluated on 10 randomly generated episodes and the mean return and standard error of those episodes are saved.

#### 4.4 Experiment

An experiment is proposed which seeks to characterize the effect different gravity models have on the Enhanced Safe Mode agent performance. As discussed earlier, it is important to train agents in simulations that are representative of the true environment. The minimization of the sim-to-real gap helps ensure that deployed agents behave in a manner consistent with their training. Realistically, however, it is often difficult or expensive to use high-fidelity dynamics models during training, so practitioners may opt for lower-fidelity dynamics models to decrease training times. This experiment investigates the pros and cons of this choice.

For this experiment, three Enhanced Safe Mode agents are trained. All agents are trained with the same initial conditions described above; however, each agent's is trained in an environment populated with a different gravity model. The first agent is trained in an environment which uses a polyhedral gravity model. The second agent is trained in an environment with a point mass gravity model, and the third environment uses a PINN-GM. In each case, the gravity model is queried for

accelerations that are used by the transition function to evolve the state forward in time.

Each environment offers different advantages and drawbacks. The polyhedral environment provides the most accurate dynamics; however, it also comes with the greatest computational cost. The 200,000 facet shape model used for Eros is notoriously slow, and will ultimately limit the total number of interactions the agent will experience with the environment from which the agent can learn. In contrast, the point mass environment will provide cruder approximation of the true dynamics, but at a much faster rate. Consequently, the agent trained in this environment will have many more experiences from which to learn, but it is unclear if those additional experiences will be sufficient to overcome the introduced sim-to-real gap. Finally, the agent trained in the PINN gravity model environment should circumvent the challenges associated with the prior two environments. Because the PINN-GM is both fast to execute, and provides a high-fidelity approximation of the true system dynamics, the agent trained in this environment should have access many realistic interactions with the environment from which it can learn.

Each agent is trained for a total three wall-clock hours, and the average return for each scenario's agent is plotted as a function of wall-clock times and training steps in Figure 4.2.

## 4.5 Results

Figure 4.2 demonstrates the respective advantages and disadvantages of using different gravity models for the Enhanced Safe Mode agent. As discussed, the agent trained in the polyhedral environment has access to the most accurate dynamics, but the polyhedral model's computational intensity only allowed for 2,000 steps in the environment over the three hour period. These 2,000 interactions are not enough for the agent to learn safe behavior which is reflected by the consistently low average return.

The agent trained in the simple gravity model environment is also unable to learn safe behavior, but not for lack of total environment interactions. Having run over 300,000 steps through the environment in the three hour training window, the agent trained in the simple environment had over two orders-of-magnitude more data. Despite this, the simplified dynamics of the environment

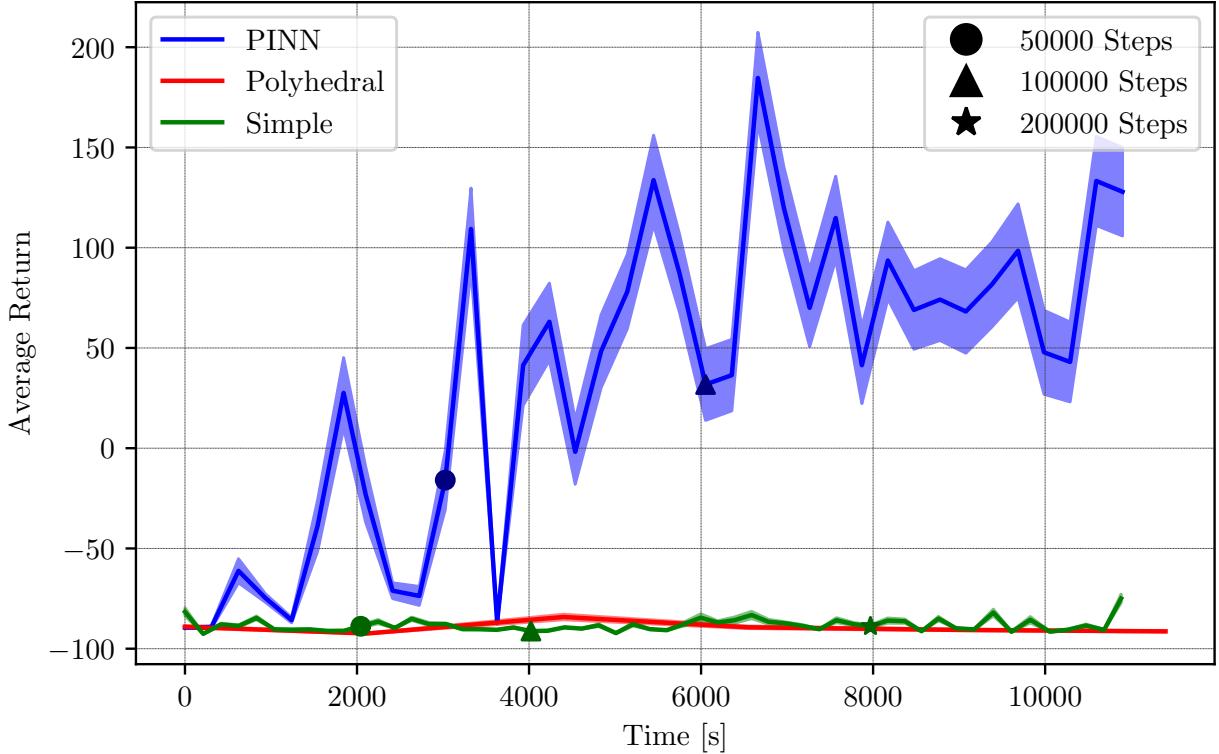


Figure 4.2: Average return as a function of clock time

led to this agent’s demise. Because the environment did not provide the agent with sufficiently realistic episodes, the agent learned behaviors remained unsafe by failing to account for the more complex dynamics of the system.

The agent trained in the PINN-GM environment avoids both of these challenges. The agent experienced over 150,000 interactions with an environment that contained representative dynamics of the true system. This combination allowed the agent to identify safe behavior that avoided collision with the asteroid as represented by the increasing average return in Figure 4.2.

Figure 4.3 shows an example of the different trajectories generated by the three Enhanced Safe Mode agents. The trajectories demonstrate that the agents trained in the simple and polyhedral environments are virtually no better than an untrained agent — colliding very quickly with the asteroid. The agent trained in the PINN environment, in contrast, is able to successfully leverage the complex dynamics of the gravity field to find a close, but safe, orbit around the equator of

the asteroid. Early findings have shown that the trajectory found by the PINN agent is safe for significantly longer than the original 10 hour episode length implying that spacecraft operators could have multiple days to diagnose the underlying anomaly. A formal stability analysis of this learned trajectory is left for future work.

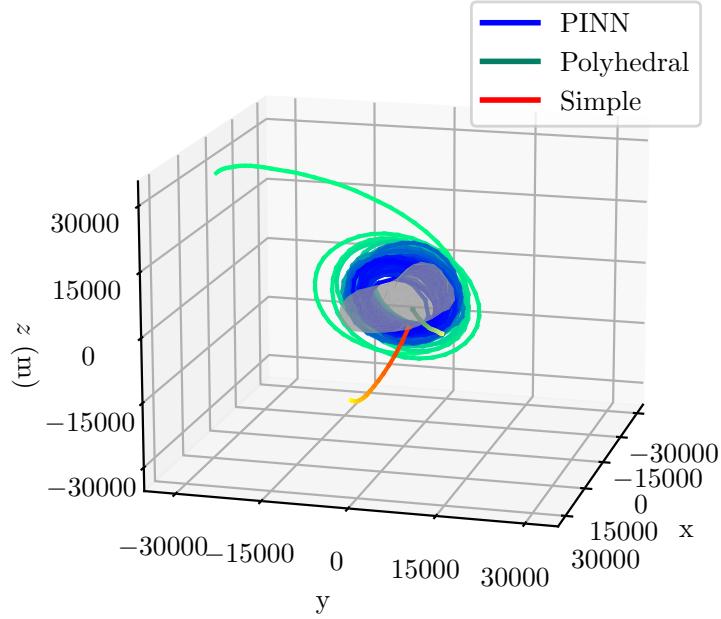


Figure 4.3: Trajectories taken **with** Enhanced Safe Mode agent enabled.

To quantify the trained agents’ robustness, 100 episodes are run without training. Each episode starts with a random initial condition, and the simulation runs for the entirety of the 10 hour episode or until the spacecraft violates one of the safety conditions. The corresponding successes and failures of each agent is shown in Figure 4.4.

Figure 4.4 further emphasizes that neither policy trained in the simple or polyhedral environments are robust to initial conditions. There are rare occasions where those agent “succeed”, but this is actually because the initial conditions are at sufficiently high altitude that the 10 hour episode terminates before the spacecraft ever reaches an altitude low enough that it could collide with the asteroid.

The policy trained in the PINN-GM environment is better, though not universally robust. Only in 45% of the episodes did the spacecraft remain safe. While this is a sizable improvement over the agents trained in the other environments, further work is needed to make the policy robust. The performance of each agent is likely to improve with additional training time. It is not uncommon for agents to be trained over the span of days rather than hours. The choice to limit this experiment to three hours of training time is simply to highlight the efficiency of the PINN model, and how it enables the training of higher-quality agents in less time.

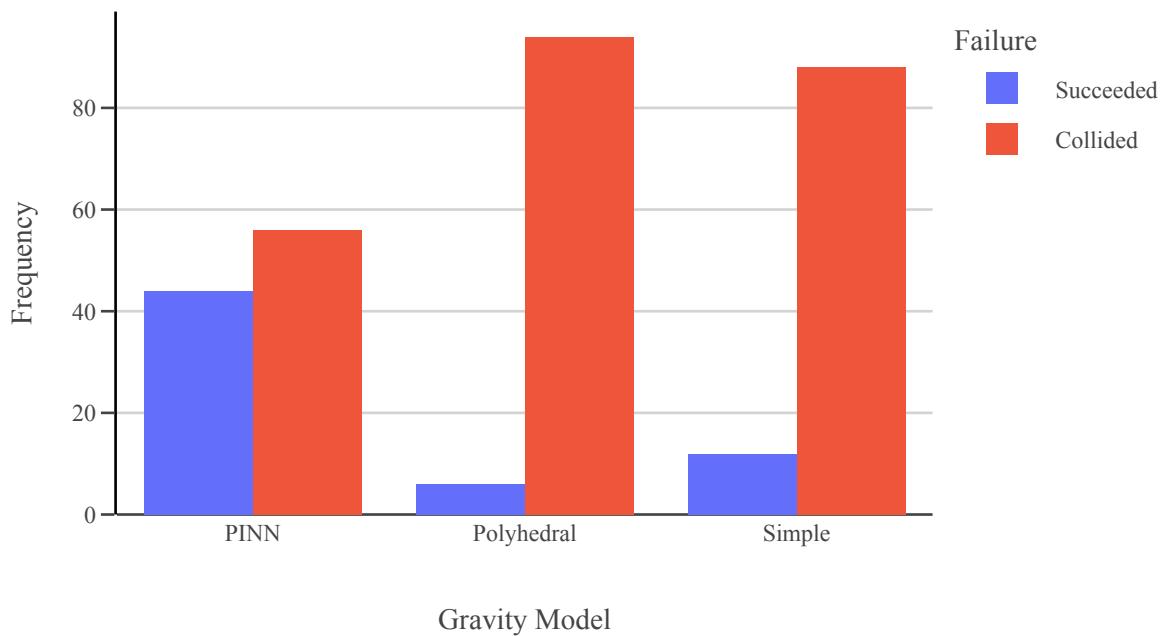


Figure 4.4: Success rates of the agents trained in different environments

## Chapter 5

### Application II: Periodic Orbit Discovery

Periodic orbits are desirable because they allow spacecraft to remain on a fixed, predictable trajectory without the need to expend fuel. Perhaps the most famous of these orbits can be found near the Earth-Sun Lagrange points where a neighborhood exists such that spacecraft motion remains bounded and periodic (111). Notable spacecraft to leverage these periodic orbits in the three body problem include STEREO-A, STEREO-B, and the James Webb Space Telescope, among others. Despite the desirability of periodic orbits, their discovery is a challenging endeavour — particularly for highly non-keplerian environments.

The reason for this difficulty is two-fold. First, analytic methods of computing periodic motion require a closed form expression of the disturbing gravitational potential to compute solutions. For simple models of the potential, i.e. a point mass approximation or a low-fidelity spherical harmonic model, analytic periodic solutions can be found. In fact, this has been accomplished with the three-body problem, which ultimately yields the aforementioned Lagrange points. In reality, however, the point mass assumption is not valid. Instead more complex gravity models are required and identifying exact solutions analytically quickly become intractable. Dynamicists must therefore turn to numerical shooting methods, framing the search for periodic orbits as a boundary value problem. Unfortunately, even the numerical shooting methods are not free of their own difficulties. In many cases, the numerical solvers will not converge if the initial guess deviates too far from the true solution. This disadvantage can be mitigated by testing many initial conditions until a solution is reached (112); however this can be a major bottleneck if the gravity model used

is computationally expensive. The computational expense can be further exacerbated by shooting methods which must use numerically-computed jacobians which add further complexity.

Small-body environments are among the settings for which these challenges are most apparent. Asteroids and comets often exhibit irregular geometries and corresponding gravity fields which produce complex, non-periodic motion. In these environments, the gravitational potential cannot be adequately represented using a point mass gravity model, and instead, expensive polyhedral gravity models must be used to capture these unusual dynamics (5). While the polyhedral gravity model offers a compelling analytic solution to the gravity modeling problem, this model can be computationally expensive to evaluate depending on the fidelity of the corresponding shape model used. This expense prevents the polyhedral model's practical use in the numerical methods used to solve the periodic orbit problem.

This chapter investigates how the Physics-Informed Neural Network gravity model can be leveraged to assist in the efficient discovery of periodic orbits in small-body settings. As shown earlier in this thesis, the PINN-GM is capable of accurately representing the gravity fields of small-bodies by learning efficient basis functions rather than prescribing them (1). This approach bypasses the computational inefficiencies of the polyhedral gravity model and retains differentiability for use in numerical methods thanks to automatic differentiation (113). The efficient and differentiable form of the PINN-GM make it a compelling tool to be leveraged in periodic orbit discovery. Not only can traditional shooting methods be applied in small-body settings without risk of computational inefficiency, but there are also new opportunities to search for orbits in different element spaces. This work investigates how the PINN-GM-III can assist in the discovery of such orbits and how they potentially expand the attracting basins around them such that fewer initial guesses are required to find a solution.

## 5.1 Background

Initial efforts to search for periodic orbits in small-body settings date back to the mid-90s. In 1995, Scheeres provided the first detailed characterization and analysis of the orbital dynam-

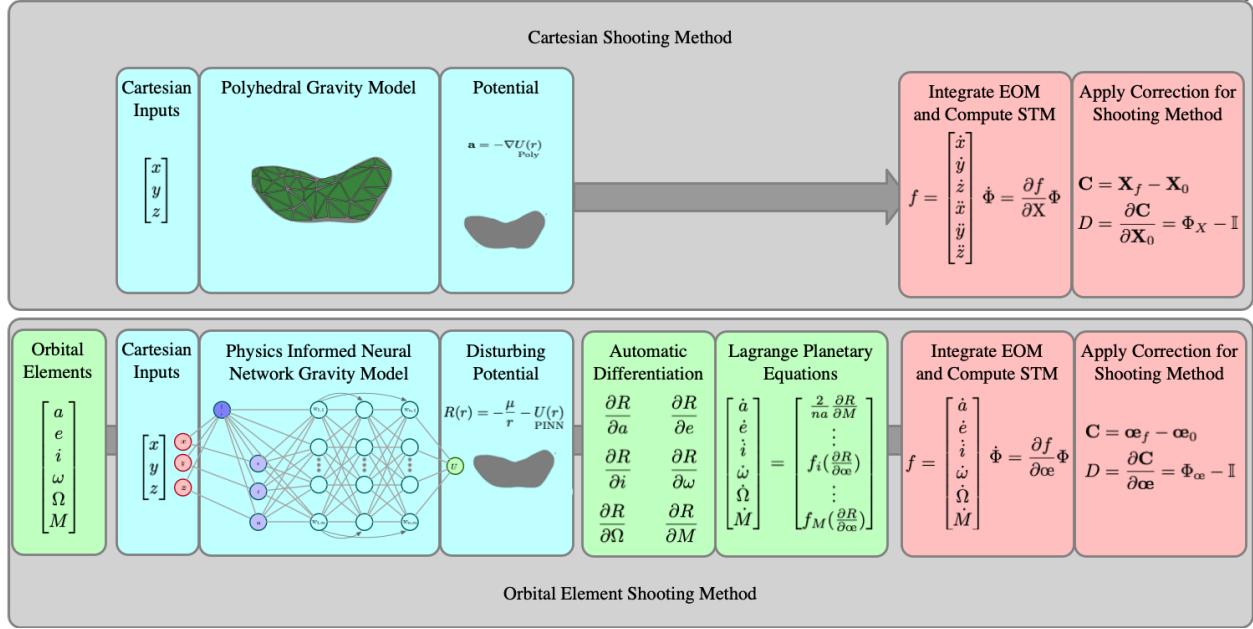


Figure 5.1: Outline of the former cartesian shooting method (top) and the novel orbital element shooting method (bottom).

ics about the asteroid 433-Eros (114). These findings highlight how rotating asteroids' stationary points can be used as the initial search space for periodic orbits, and the paper ultimately reveals three periodic orbit families about Eros: direct near-equatorial orbits, retrograde near-equatorial orbits, and non-planar orbits with resonant nodal periods. In 1998, Scheeres also provided more algebraically involved derivations of bounded, frozen orbits about the asteroid 4179-Toutatis (115). This strategy relies on period averaging the Lagrange planetary equations using a simplified low-degree spherical harmonic expansions of the potential. Additional analysis is conducted discussing how these discovered orbits in the simplified system can be iteratively corrected to produce increasingly stable and periodic motion in a full-fidelity gravitational potential.

The algorithms responsible for the numerical search for periodic orbits date back even further. A particularly influential paper by Howell in 1984 introduces a numerical shooting method to identify periodic orbits in the Earth-Moon system (116). This algorithm assisted in the discovery of the well known Halo orbit family. In 2003, Doedel et. al. provided a comprehensive description of the constraints which can be used in conjunction with these shooting methods to find periodic

orbits in conservative systems (117). In 2009, Abad et. al. provided a detailed analytical approach to solving for periodic orbits for the asteroid 216-Kleopatra by using a Lie Transformation and Delaunay orbital elements with a zonal spherical harmonic gravitational potential assumption (118). In 2012, Yu et. al. used a full polyhedral gravity model and a hierarchical grid search to identify 29 periodic orbits in approximately 12 days of compute time (112). In 2003, Lan et. al. proposed a variational, cost-function minimization method to identify periodic solutions to high-dimensional problems rather than Poincare sections or traditional Newton-Raphson (119). Additional useful references regarding the study of frozen and periodic orbits also include Refs. (111), (120), and (121).

To the best of the authors' knowledge, there have been no attempts to apply the shooting method in orbital element space using the Lagrange planetary equations with a full-fidelity gravity model. Such effort is the primary contribution of this work.

## 5.2 Methodology

The search for periodic orbits is often constructed as a boundary value problem which seeks to minimize the difference between the initial state,  $\mathbf{X}(t_0)$  or  $\mathbf{X}_0$ , and the state after some period  $T$ ,  $\mathbf{X}(t_f)$  or  $\mathbf{X}_f$ :

$$\mathbf{X}_f - \mathbf{X}_0 = \mathbf{0} \quad (5.1)$$

where  $\mathbf{X}$  corresponds to the cartesian state vector  $[\mathbf{r}, \mathbf{v}, T]$  which evolves according to

$$\dot{\mathbf{X}} = \begin{bmatrix} \mathbf{v} \\ \mathbf{a} \\ 0 \end{bmatrix} \quad (5.2)$$

For keplerian motion, this boundary value problem is naturally satisfied; however when non-keplerian forces are introduced like 3rd-body perturbations or non-symmetric gravity perturbations this is no longer the case. This work will focus exclusively on the non-symmetric gravity perturbations. To solve this problem, a traditional shooting method is introduced. The shooting

method is a differential corrector algorithm which begins by taking the Taylor series expansion of the constraint in Equation 5.1 about the true solution and setting the result to zero.

$$C(\mathbf{X}_0^*) = C(\mathbf{X}_0 + \delta\mathbf{X}_0) \quad (5.3)$$

$$0 = C(\mathbf{X}_0) + \frac{\partial C}{\partial \mathbf{X}_0} \delta\mathbf{X}_0 + \text{H.O.T.} \quad (5.4)$$

This expansion can then be solved for a value of  $\delta\mathbf{X}_0$  through a minimum norm solution:

$$\delta\mathbf{X}_0 = - \left( \left( \frac{\partial C}{\partial \mathbf{X}_0} \right) \left( \frac{\partial C}{\partial \mathbf{X}_0} \right)^T \right)^{-1} C(\mathbf{X}) \quad (5.5)$$

where

$$\frac{\partial C}{\partial \mathbf{X}_0} = \frac{\partial}{\partial \mathbf{X}_0} (\mathbf{X}_f - \mathbf{X}_0) \quad (5.6)$$

$$= \begin{bmatrix} \frac{\partial \mathbf{X}_f}{\partial \mathbf{X}_0} - \frac{\partial \mathbf{X}_0}{\partial \mathbf{X}_0}, & \frac{\partial \mathbf{X}_f}{\partial T} \end{bmatrix} \quad (5.7)$$

$$= \begin{bmatrix} \Phi(t_f, t_0) - \mathbb{I}, & \frac{\partial \mathbf{X}_f}{\partial T} \end{bmatrix} \quad (5.8)$$

and  $\Phi$  is the state transition matrix.

Due to the linearization of the original system, this process must be repeated until some termination criteria is reached such as  $|\delta\mathbf{X}_0| < \epsilon$ . This numerical procedure will converge for initial guesses of  $\mathbf{X}_0$  that are sufficiently close to the true periodic solution,  $\mathbf{X}_0^*$ , assuming that a reliable gravity model exists which can provide accurate values of the acceleration vector,  $\mathbf{a}$ , in Equation 5.2.

Historically, the search for periodic orbits is conducted with a shooting method formulated in cartesian space, as most gravity models are designed such that the equations of motion are a function of the cartesian position,  $\mathbf{r}$ . One of the disadvantages of conducting the search in a cartesian space is that the coordinates span the entirety of  $\mathbb{R}^6$  which makes the search space extremely large. This makes the corresponding probability of selecting an initial condition that is sufficiently close to a true periodic orbit quite small. Non-cartesian coordinates sets, however, can have smaller domains. This work hypothesizes that these reduced domains may be easier to search, requiring fewer initial guesses before converging to a solution. For example, consider the

traditional orbital elements set, for which only a single coordinate (the semi-major axis) can extend from  $(0, \infty)$  whereas the remaining coordinates  $e, i, \omega, \Omega$  and  $M$  each remain bounded between  $[0, 1], [-\pi/2, \pi/2], [0, 2\pi], [0, 2\pi], [0, 2\pi]$  respectively. Intuitively, these coordinates would be much more efficient to search through.

Unfortunately, traditional orbital elements cannot be easily used in numerical shooting methods. This is because the time-derivatives of orbital elements,  $\dot{\mathbf{e}}$ , and the jacobian of these derivatives,  $\frac{\partial \dot{\mathbf{e}}}{\partial \mathbf{e}}$ , are required, but notoriously difficult to compute. The time derivatives of the orbital elements can technically be computed through the Lagrange Planetary Equations (LPE):

$$\frac{\partial a}{\partial t} = \frac{2}{na} \frac{\partial R}{\partial M} \quad (5.9)$$

$$\frac{\partial e}{\partial t} = \frac{1 - e^2}{na^2 e} \frac{\partial R}{\partial M} - \frac{\sqrt{1 - e^2}}{na^2 e} \frac{\partial R}{\partial \omega} \quad (5.10)$$

$$\frac{\partial i}{\partial t} = \frac{\cos i}{na^2 \sqrt{1 - e^2} \sin i} \frac{\partial R}{\partial \omega} - \frac{1}{na^2 \sqrt{1 - e^2} \sin i} \frac{\partial R}{\partial \Omega} \quad (5.11)$$

$$\frac{\partial \omega}{\partial t} = -\frac{\cos i}{na^2 \sqrt{1 - e^2} \sin i} \frac{\partial R}{\partial i} + \frac{\sqrt{1 - e^2}}{na^2 e} \frac{\partial R}{\partial e} \quad (5.12)$$

$$\frac{\partial \Omega}{\partial t} = \frac{1}{na^2 \sqrt{1 - e^2} \sin i} \frac{\partial R}{\partial i} \quad (5.13)$$

$$\frac{\partial M}{\partial t} = n - \frac{1 - e^2}{na^2 e} \frac{\partial R}{\partial e} - \frac{2}{na} \frac{\partial R}{\partial a} \quad (5.14)$$

though these equations rely on an differentiable form of the disturbing potential function,  $R$ , which is derived from the expression

$$R(\mathbf{r}) = -\frac{\mu}{r} - U(\mathbf{r}) \quad (5.15)$$

where  $U$  is the total gravitational potential and  $\mu$  is the gravitational parameter for the body in question.

In some simplified cases, the Lagrange planetary equations can be evaluated analytically as is shown by Reference (115) and Reference (118). By expressing the disturbing potential only as a function of low-degree spherical harmonic models, some first-order properties can be derived and candidate periodic or frozen orbits can be found. Unfortunately, this practice becomes intractable as the analytic representation of the disturbing potential function  $R$  increases in fidelity, preventing

the use of high-fidelity spherical harmonic or polyhedral models. In principle, the search can be conducted numerically, but the jacobian  $\partial\dot{\phi}/\partial\phi$  would need to be approximated through finite differencing which adds considerably more compute cycles to an already expensive propagation.

Herein lies the value of the PINN-GM. The PINN-GM not only offers a computationally efficient and high-fidelity representation of the potential, but it can also be differentiated exactly with respect to arbitrary coordinate sets. The PINN-GM therefore bypasses the limitations of past gravity models, allowing users to evaluate the LPE and corresponding jacobians exactly for any coordinate set. This property is the byproduct of automatic differentiation, a generalization of chain rule which allows for the exact differentiation of any numerical algorithm with respect to its input.

### 5.2.1 Automatic Differentiation

Automatic differentiation is covered in Section 2.3.1 but is repeated here for convenience. Automatic differentiation is method to compute the exact derivative of an algorithm with respect to any input. This is done by constructing a computational graph and using either a forward or backward form of chain rule such as:

$$\begin{aligned}\frac{\partial y}{\partial x} &= \frac{\partial y}{\partial w_{n-1}} \frac{\partial w_{n-1}}{\partial x} \\ &= \frac{\partial y}{\partial w_{n-1}} \left( \frac{\partial w_{n-1}}{\partial w_{n-2}} \frac{\partial w_{n-2}}{\partial x} \right) \\ &= \dots\end{aligned}$$

where  $x$  is the input to some arbitrary algorithm,  $w_i$  are the sequence of intermediate calculations performed to produce the final output,  $y$ .

Because all algorithms are constructed from elementary functions with known derivatives, the partials of each intermediate expressions can always be computed alongside the original calculation. This property ensures that partial of the output  $y$  with respect to any input  $x$  can be computed automatically. Automatic differentiation is best known for its application within deep learning,

where it commonly applied in the stochastic gradient descent algorithms used to train neural networks (76).

This work changes the application of automatic differentiation beyond training neural networks, and instead uses it to simply take high-order derivatives of a pre-trained network. For periodic orbit discovery, this means applying automatic differentiation to the PINN-GM to resolve derivatives of the disturbing potential function,  $\frac{\partial \hat{R}}{\partial \dot{\alpha}}$ , to compute the LPE, and using automatic differentiation again to compute the necessary jacobian for the STM propagation. This framework is general, and be used not only for cartesian coordinate sets but any arbitrary set of input coordinates.

### 5.2.2 Characterization of the PINN-GM

Before demonstrating the utility of a PINN-GM for periodic orbit discovery, an experiment is proposed which seeks to characterize the speed and accuracy of the PINN-GM compared to that of a polyhedral gravity model. This will ensure the PINN-GM can be exchanged with its more expensive polyhedral model counterpart without loss of dynamical fidelity. The ground truth gravity model for this experiment is a 200k vertex polyhedral gravity model of the asteroid Eros. This is an extremely expensive gravity model, so a lower-fidelity 8k polyhedral alternative is constructed and tested as well.

First the accuracy of both the 8k polyhedral model and the PINN-GM are characterized on 30,000 test points distributed isometrically along the cartesian planes. The PINN-GM is trained on 950,000 training data distributed between a 0-10R altitude are generated from the 200k polyhedral model and the network is trained for 10,000 epochs. The average acceleration error of the 8k polyhedral model is 0.38% error. In contrast, the PINN-GM produces an average acceleration error of 0.003%, or two orders-of-magnitude smaller than the 8k polyhedral model error.

Second, an experiment is conducted which evaluates the integration error of both gravity models. This begins by uniformly sampling three initial conditions about the asteroid 433-Eros according to the orbital element distributions listed in Table 5.1. These initial conditions are

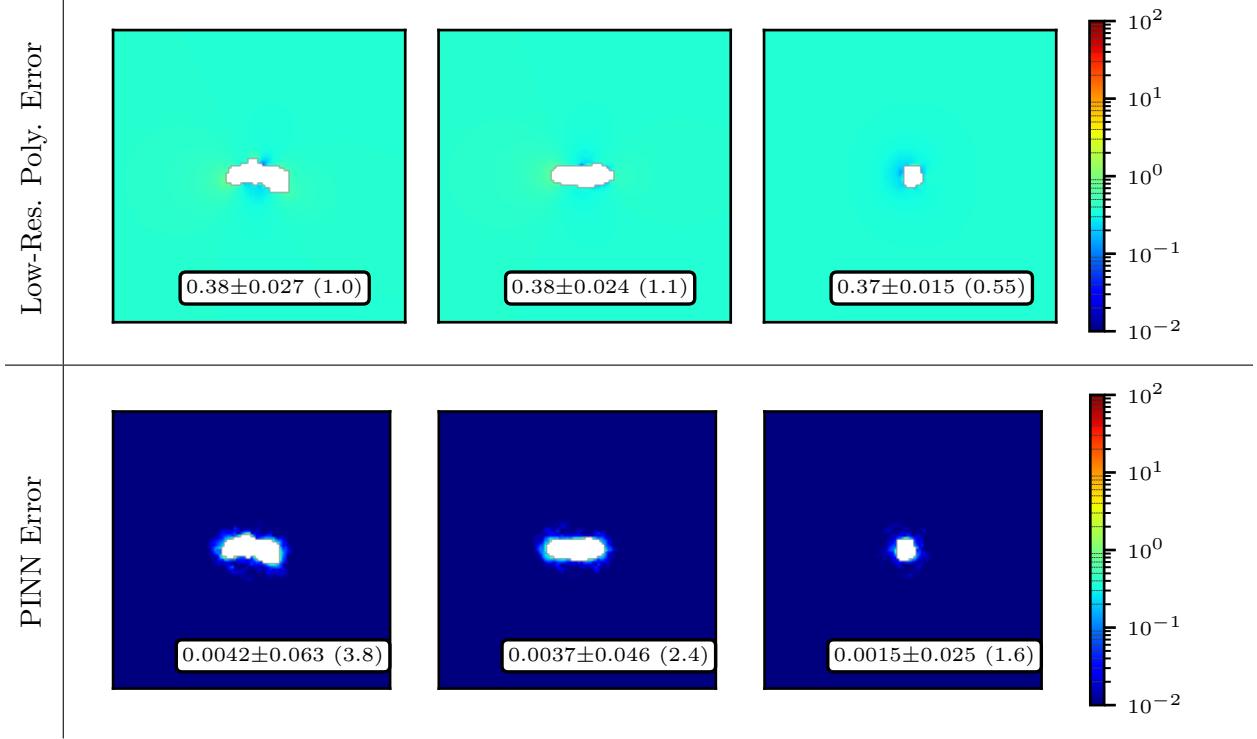


Figure 5.2: Percent error of the acceleration vector for each gravity model

Table 5.1: Initial Orbital Element Distribution

Parameter	Value	Parameter	Value
Semi-major axis	$\mathcal{U}[3R_{\text{Eros}}, 7R_{\text{Eros}}]$	Argument of periapsis	$\mathcal{U}[0, 2\pi]$
Eccentricity	$\mathcal{U}[0.1, 0.3]$	Longitude of the Ascending Node	$\mathcal{U}[0, 2\pi]$
Inclination	$\mathcal{U}[-\frac{\pi}{2}, \frac{\pi}{2}]$	Mean Anomaly	$\mathcal{U}[0, 2\pi]$

converted to cartesian coordinates and then propagated for a keplerian orbit period,  $T$ , defined as:

$$T = 2\pi\sqrt{\frac{a^3}{\mu}} \quad (5.16)$$

These initial conditions are propagated using an adaptive Runge-Kutta integrator with the three aforementioned gravity models. Once propagated, the position, velocity, and acceleration errors of the 8k polyhedral and the PINN-GM trajectories are reported alongside the total computation time in Figure 5.3.

Figure 5.3a shows that for each orbit, the PINN-GM provides considerably more accurate dynamics than the 8k polyhedral gravity model. In all cases, the PINN-GM deviates less than

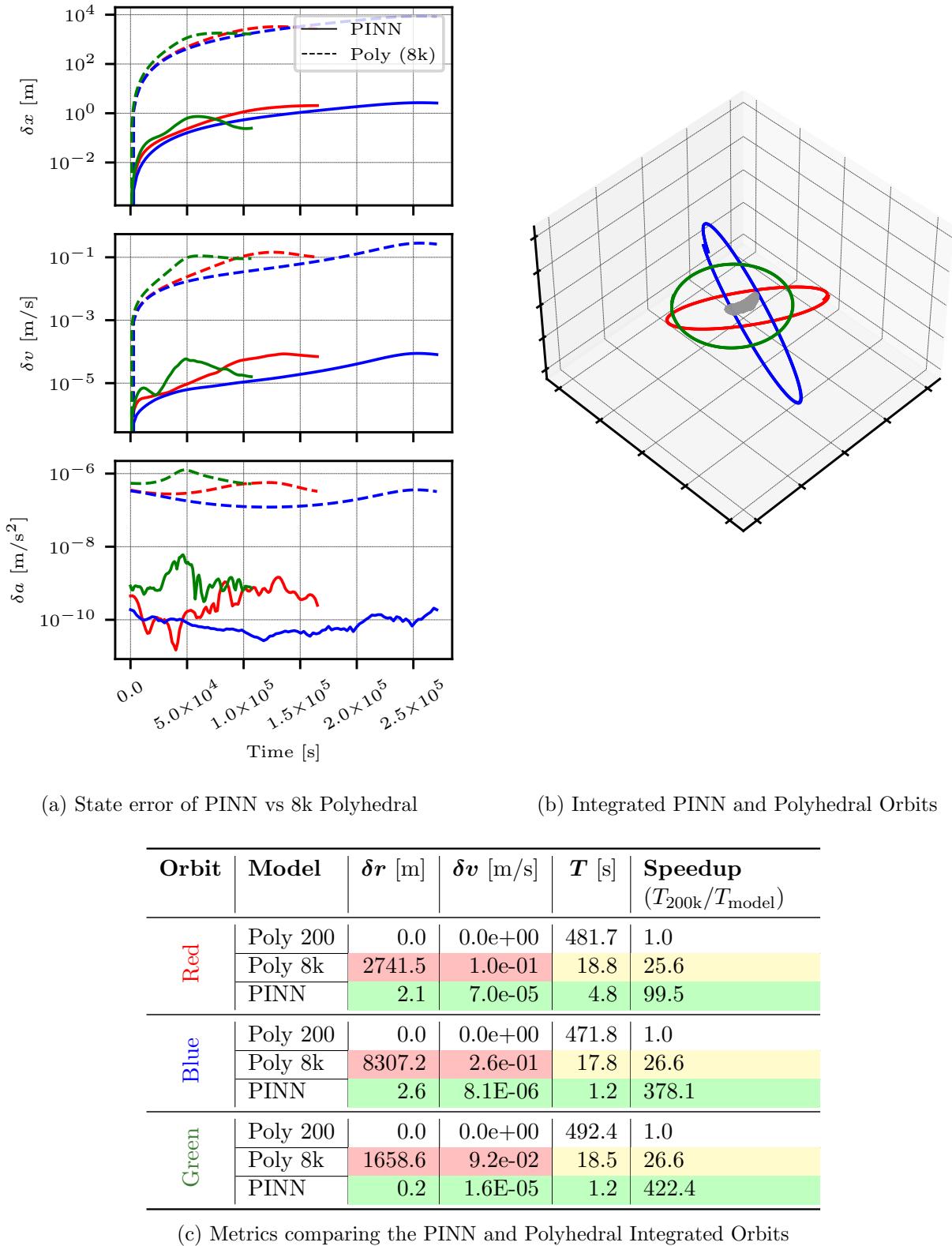


Figure 5.3: Metrics comparing the orbits generated by the PINN-GM and the polyhedral gravity models.

3m from the true trajectory of the 200k polyhedral gravity model. The low-fidelity 8k polyhedral gravity model deviates as much as 8km. Similar metrics are observed for the difference in velocity, with the PINN-GM producing errors up to five orders-of-magnitude smaller than the 8k model. These metrics demonstrate that the PINN-GM is a viable choice to replace the 200k polyhedral model of the asteroid.

The computational speed metrics of the PINN-GM are also provided in Figure 5.3c. The PINN-GM integrates these orbits between approximately 4x and 15x faster than the 8k polyhedral model, and between 99x and 420x faster than the 200k polyhedral model. This speed benefit alone makes the PINN-GM a compelling tool in periodic orbit discovery as these trajectories need to be integrated many times given the iterative nature of the shooting method. In Reference (112), the search for 29 periodic orbit families about the asteroid Kleopatra took 12 days using a high-fidelity polyhedral model. These findings suggest that the same search conducted using a PINN-GM could be conducted on the order of 40 minutes.

### 5.3 PINN-GM Cartesian Shooting Method

Accepting the PINN-GM as a representative model of the true gravitational potential, it can now be used to search for periodic orbits using the shooting method as discussed earlier. In this second experiment, the same three initial conditions will be used to begin the search for periodic orbits using a cartesian shooting method. To compute the state transition matrix used in Equation 5.5, automatic differentiation is again leveraged to compute the jacobian

$$A = \frac{\partial \dot{\mathbf{X}}}{\partial \mathbf{X}} \quad (5.17)$$

to then propagate the STM through

$$\dot{\Phi}(t) = A\Phi(t) \quad (5.18)$$

where  $\Phi(0) = \mathcal{I}$ .

The shooting algorithm's termination is triggered if any of the following criteria are met:

- (1) The norm of the correction vector,  $|\delta \mathbf{X}_0|$ , is less than  $\epsilon_x(\epsilon_x + |\mathbf{X}_0|)$  where  $\epsilon_x = 10^{-8}$ .

- (2) The normalized change in the cost function,  $\delta F/F$ , is less than  $\epsilon_f$  where  $\epsilon_f = 10^{-8}$ .
- (3) The number of iterations exceed 50.

The experiment proceeds by running the shooting methods in two modes: coarse and fine. In the coarse shooting method, higher Runge-Kutta integration error is tolerated than during that of the fine shooting method. This allows for rapid integration of the state and state transition matrix, albeit at lower accuracy. Once the coarse shooting method is completed and the initial conditions are shifted closer to a solution, the fine shooting method is executed starting from the output found with the coarse method. The lower error tolerance of the Runge-Kutta method ultimately assists the algorithm in converging accurately on the local minimum.

### Non-dimensionalization

An important consideration for the shooting method is how the state and equations of motion are non-dimensionalized. For the cartesian shooting method, two intuitive options exist: First, normalize distance by  $l^* = |\mathbf{r}_0|$  and normalize time by the initial period  $t^* = 2\pi\sqrt{a_0^3/\mu}$ . This choice is useful for achieving numerical stability; however, it can bias the results of the optimization process towards the velocity coordinates. Specifically, in certain circumstances the velocity coordinates non-dimensionalize to values that are much larger than that of the position coordinates. Given that the shooting method is tasked with minimizing the difference of the non-dimensionalized state, this scaling prioritizes solutions which provide closer matches between the initial and final velocity at the expense matching the position vector. This behavior can lead to premature termination of the method, and yield solutions which are not periodic. The second non-dimensionalization strategy continue to scaling distance by scale  $l^* = |\mathbf{r}_0|$ , but instead scale the velocity such that  $t^* = l^*/|\mathbf{v}|$ . This ensures the position and velocity are of similar magnitudes in the constraint vector and therefore contribute equally to the cost function used to find periodic solutions. The latter of these two methods is used for the proceeding experiments.

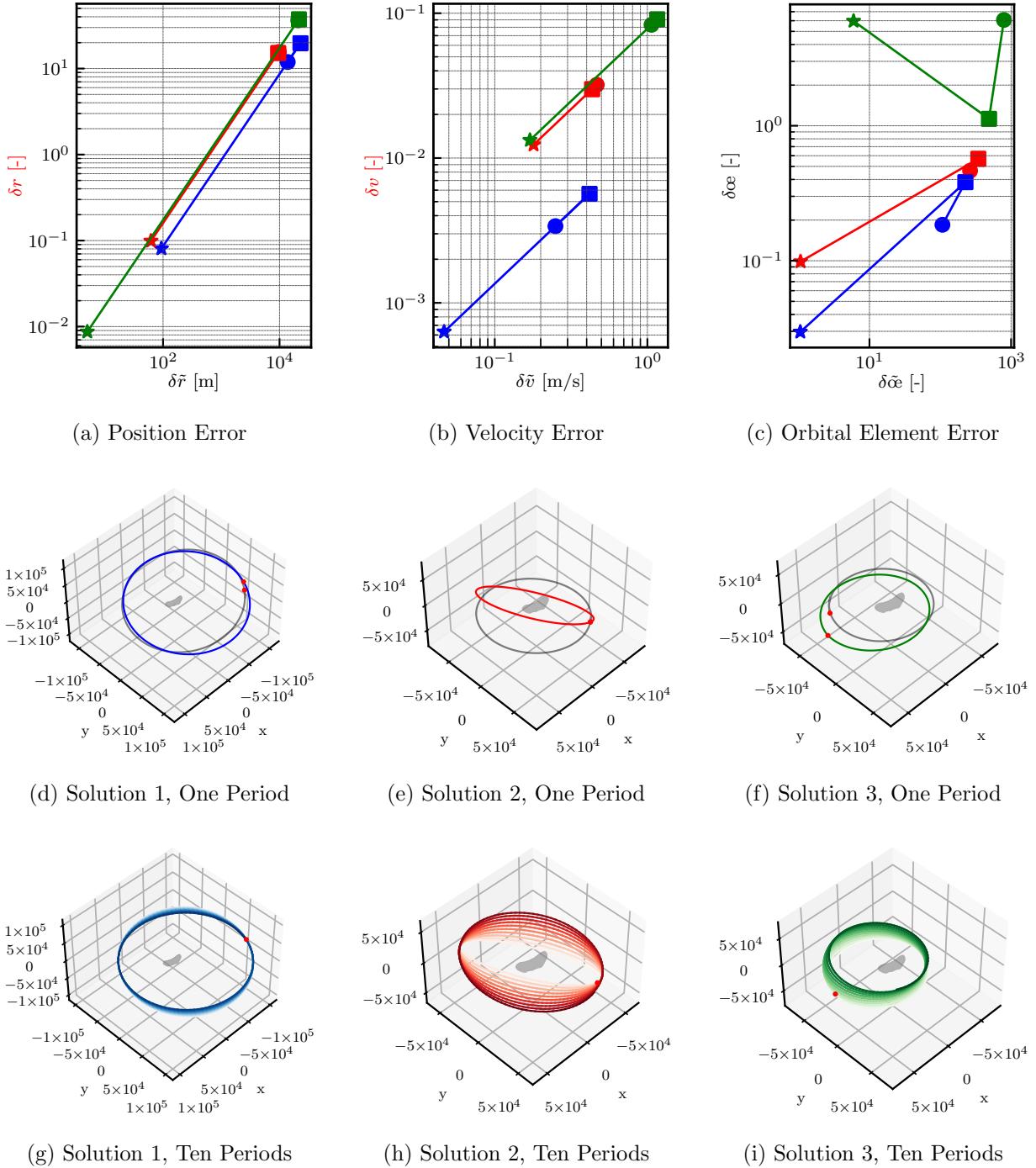


Figure 5.4: Results from the Cartesian Shooting Method. Top row: Solution error in dimensionalized coordinates (tilde) and non-dimensionalized coordinates (no tilde). Middle row: Starting orbit (gray) and the discovered solutions (color) propagated for one orbit. Bottom row: Solutions found propagated for 10 orbits.

## Results

Figure 5.4 shows a collection of results for the cartesian coarse and fine shooting methods run on the three original initial conditions. The first row shows the magnitude of the state error between  $\mathbf{X}_f$  and  $\mathbf{X}_0$  in both dimensionalized  $(\tilde{\mathbf{r}}, \tilde{\mathbf{v}}, \tilde{\omega})$  and non-dimensionalized  $(\mathbf{r}, \mathbf{v}, \omega)$  coordinates. The coarse shooting method error is shown with the circle → box markers, and the fine shooting method is shown with the box → star markers. The shift from high values of  $\delta r$  and  $\delta v$  on the y-axis to low values reflects that the shooting method is in fact moving the initial conditions towards periodic motion. However, this desirable shifting in cartesian space is not mimicked in orbital element space as shown by the green curve in Figure 5.4c.

This inconsistency between the cartesian state error and the orbital element state error is surprising. One would think that the minimization of the cartesian state error would inevitably ensure that the initial and final state would also share very similar orbital elements. This finding, however, suggests that as cartesian shooting methods converge on near-periodic solutions, the orbital elements of the initial and final state are not necessarily getting minimized. The resulting long-term behavior of the converged solution demonstrates this more clearly. If the solutions found via the cartesian shooting method are propagated for a single orbit, the difference between their initial and final coordinates are very small as seen in the second row of Figure 5.4. However, over longer periods of time, the orbit begins to evolve into entirely different orbits with different inclinations and longitudes of the ascending node as seen in the third row of Figure 5.4. This steady drift away from periodicity in orbital element space provides motivation for why it may be advantageous to solve for periodic orbits in an orbital element description instead to ensure that the initial and final orbits are similar, regardless if their cartesian state is.

### 5.4 PINN-GM Orbit Element Shooting Method

A second experiment is proposed which investigates this claim. The search for periodic orbits using the PINN-GM is repeated, however, rather than representing the problem in cartesian

coordinates, the problem is reformulated in orbital element space (i.e.  $(r_x, r_y, r_z, v_x, v_y, v_z) \rightarrow (a, e, i, \omega, \Omega, M)$ ). By framing the periodic orbit problem in element space, three advantages are acquired. First, the search space for periodic solutions is reduced. Four of the six elements have a naturally finite domain due to angle wrapping ( $i, \omega, \Omega, M$ ) and the eccentricity must remain less than 1 for the orbit to remain bounded. Only the semi-major axis is allowed to scale to infinity. In contrast, solutions in cartesian coordinates can span anywhere in  $(-\infty, \infty)$  for each coordinate. The second advantage of using orbital elements is that it gives mission designers greater control over the types of solutions reached by the shooting method. A shooting method that relies on cartesian coordinates may reach a minimum norm solution for which the state difference is small, but the difference in orbital elements is large. By solving the periodic orbit problem in element space, the shooting method guarantees that the spacecraft will remain in a very similar orbit geometry regardless of the cartesian state error. The third advantage of this approach is that certain orbital elements can be held fixed during the minimization process. For example, if a particular semi-major axis value and inclination are required for a particular mission phase, they can be removed from the set of decision variables but kept as part of the constraint vector. All solutions must thereby maintain the prescribed values of  $a$  and  $i$  and seek periodicity using the other unconstrained elements. Enforcing constraints of this nature is not be possible with a cartesian shooting method.

### Non-dimensionalization

Similar to the cartesian shooting method, careful attention must be paid to the normalization of the orbital element state vector, jacobian, and constraint vector. Given the varying domains of the different coordinates within the orbital element vector, the non-dimensionalization must be performed as follows: Time is non-dimensionalized using the keplerian period. Distance is non-dimensionalized using the value of the initial semi-major axis. Angles  $i, \omega, \Omega$  and  $M$  are all scaled by  $2\pi$ . In addition, the  $\delta i, \delta \omega, \delta \Omega$ , and  $\delta M$  coordinates in the constraint vector must be the minimum signed angle between the initial and final coordinate as to not disproportionately bias the update

(i.e.  $\delta M = 1.99\pi \rightarrow \delta M = -0.01\pi$ ).

## Results

For the experiment, the same three initial conditions are used but are instead propagated and corrected in orbital element space. All elements remain unconstrained in the minimization, and the corresponding improvement in orbital element state error (both dimensionalized and non-dimensionalized) are provided in Figure 5.5. The experiment also makes use of the coarse and fine shooting methods, where the square corresponds to the final iteration of the coarse method and the star corresponds to the final iteration of the fine method.

In this experiment, all values of  $\delta r$ ,  $\delta v$ , and  $\delta\omega$  decrease, rather than only the cartesian state error as seen in the top row of Figure 5.5. The second and third row of Figure 5.5 demonstrate that not only are the solutions found periodic over a single period, but they also remain periodic after 10 orbits. These findings suggest that the solutions found using the orbital element shooting method are more likely to maintain their desired element set, even if not perfectly periodic. This is best exhibited by the red orbit whose dimensionalized position difference between  $r_0$  and  $r_f$  is in excess of 1 km, yet its orbit elements remain nearly identical over the full 10 periods.

### 5.5 PINN-GM Constrained Orbital Element Shooting Method

As discussed, one of the advantages of the orbital element shooting method comes from the ability to constrain particular elements of the solution. For example, in some mission settings, a specific semi-major axis or range of inclinations may be required. There are no obvious ways to embed these mission requirements into a cartesian shooting method; however, with the orbital element shooting method, imposing these mission constraints is trivial as users only need to remove the relevant decision variables from the optimization set ( $\delta\omega' = \delta\omega_0 / \{\delta a_0, \delta i_0\}$ ). In this way, periodicity must be sought by changing the elements which remain in the decision variable vector.

To demonstrate this behavior, an experiment is proposed which samples a single random initial condition and solves for a periodic orbit using three methods: the cartesian shooting method,

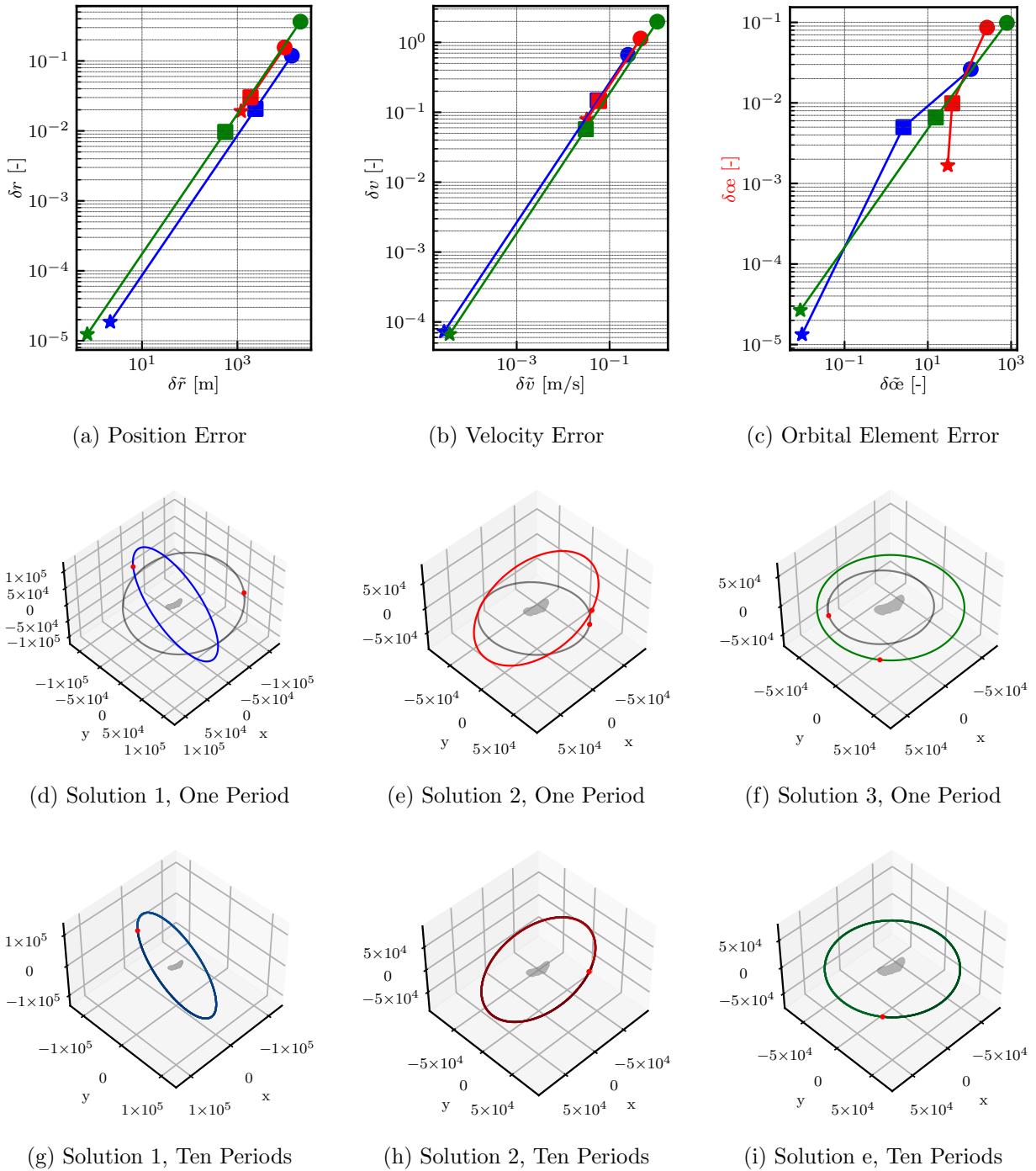


Figure 5.5: Results from the Orbital Elements Shooting Method. Top row: Solution error after one orbit in dimensionalized coordinates (tilde) and non-dimensionalized coordinates (no tilde). Middle row: Starting orbit (gray) and the discovered solutions (color) propagated for one orbit. Bottom row: Solutions found propagated for 10 orbits.

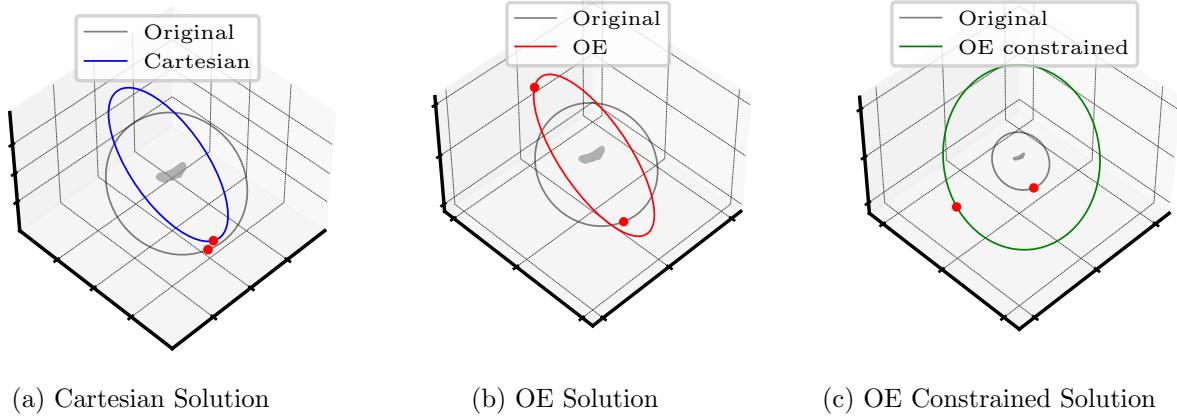


Figure 5.6: Orbit solutions found by cartesian shooting method (blue), unconstrained OE shooting method (red), and constrained OE shooting method (green) compared to the original orbit (gray/black).

Table 5.2: Initial Conditions and Solutions

Scenario	$\mathbf{a}[m]$	$\mathbf{e}$	$\mathbf{i}$ [rad]	$\omega$ [rad]	$\boldsymbol{\Omega}$ [rad]	$\mathbf{M}$ [rad]	$\mathbf{T}$ [s]	$ d\mathbf{X} [m]$
Initial Condition	75162	0.14	0.96	2.27	1.73	3.33	193802.90	4177.45
Cartesian	90646	0.12	1.52	-0.55	1.45	-0.06	255551.65	427.50
Orbital Elements	110911	0.01	1.57	-0.17	1.41	3.10	343826.63	0.03
OE Constrained	240729	0.02	0.96	-1.17	1.42	5.62	1083058.59	933.66

the orbital element shooting method, and the orbital element shooting method with constrained elements. Here a solution is desired for which initial inclination of  $i = 0.96$  radians is maintained. Despite this request, not all methods will converge towards solutions where this condition will be satisfied. To demonstrate, the three corresponding solutions are shown alongside the original initial conditions in Table 5.2 and plotted in Figure 5.6.

Note how the constrained orbital element shooting method is the only algorithm that converges towards an orbit that maintains the specified inclination. The cartesian shooting method and unconstrained orbital element shooting method technically find orbits that have greater periodicity, or smaller values of  $|d\mathbf{X}|$ , but both solutions have significantly larger inclinations than what is requested. These results suggest that there are trade-offs being made by constraining particular elements during the minimization. In principle, a more periodic solution may exist if these constraints

are relaxed. Fortunately, such relaxation is possible through trust region minimization constraints like those proposed in Reference (122) and available through the popular `scipy.optimize` Python package. Trust region minimization allows for softer constraints to be applied to the orbital element shooting method where solutions are to be bounded by some prescribed limits. This optimization bounding can be used in conjunction with the orbital element shooting method to assist in finding periodic solution within a wider range of mission parameters rather than enforcing a single value.

## 5.6 Additional Initial Conditions

While the findings presented thus far are encouraging, there are insufficient samples to yield statistically significant conclusions contrasting the two shooting methods. In an attempt to close this gap, a small Monte Carlo analysis is performed. An additional 300 initial conditions are tested by both the cartesian and the orbital element shooting methods. Once a solution is found or the solver terminates, the solutions are propagated for ten orbits and the resulting position error for each solution is saved and plotted in Figure 5.7.

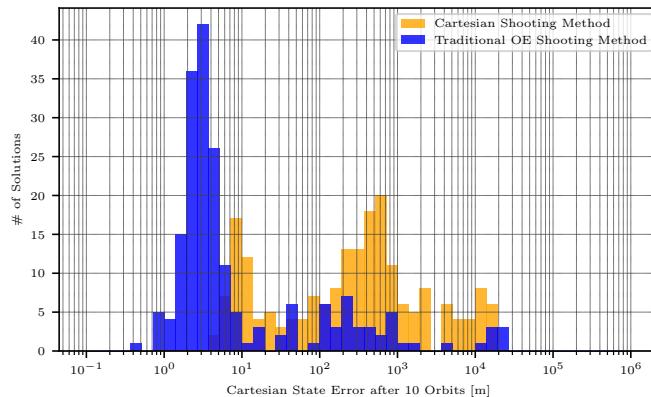


Figure 5.7: Average Percent Error of Solutions using the Cartesian LPE and OE LPE

Figure 5.7 provides further evidence of the advantages posed by the orbital element shooting method over that of the traditional cartesian approach. Of the 300 initial guesses, the orbital element shooting method consistently discovers orbits that are closer to periodic than its cartesian counterpart with an average position error of approximately 3 meters versus the hundreds of meters

found via the cartesian method. This result further supports the idea that by searching for periodic orbits in a orbital element space can offer more accurate solutions using fewer initial guesses.

## Chapter 6

### Application III: Gravity Field Estimation and Filtering

#### 6.1 Overview

In the past two decades, small-body exploration has blossomed into a major research focus for interplanetary exploration. Missions like Hayabusa2, Psyche, DAWN, OSIRIS-REx, Janus, and others have demonstrated this priority and the corresponding need for enhanced tools to enable the rapid and safe exploration of these interesting targets (123; 124; 125; 126; 127). For each mission, it has been, and will continue to be, imperative that scientists and engineers prioritize the determination of a reliable gravity field model. These gravity models form the basis from which planetary scientists can build an intuition about the body's density distribution and surface properties, and by which dynamicists and mission designers can build trajectories that leverage the natural dynamics of the system (72; 128; 69).

Traditionally these gravity models are constructed by first assuming the body in question can be modeled as a point-mass, i.e. the body in question is infinitely small, perfectly spherical, and homogeneous in density. These assumptions are often adequate for placing a spacecraft in an initial, high-altitude orbit around the body, but they quickly become problematic as more nuanced mission operations begin. Specifically, as spacecraft enter lower-altitude orbits, each of the assumptions begin to break down. For one, asteroid shapes are often not spherical. Consider the asteroids Eros or Itokawa which both exhibit particularly irregular geometries. Moreover, asteroid densities are not necessarily homogeneous as shown from recent OSIRIS-REx data (44). Taken together, these irregular shapes and inhomogeneous densities produce non-uniform gravity fields which, in turn,

yield highly non-keplerian motion.

It is important to capture these gravitational perturbations, particularly before the spacecraft enters lower orbits or attempts a landing. To accomplish this, dynamicists must turn to other, alternative gravity models. For ground-based simulation, dynamicists often use the polyhedral gravity model (5) which leverages a polyhedral shape model of the asteroid to compute the gravitational potential and acceleration assuming the body has constant density. This approach provides a considerably more accurate gravity model than the prior point mass model, but it comes with two caveats. First, not all asteroids have constant density. Such findings were recently demonstrated from the gravity science team on OSIRIS-REx which revealed the asteroid Bennu has heterogeneous mass distributions (44). While the polyhedral gravity model can accommodate inhomogeneous density profiles, these profiles are challenging to estimate uniquely (73). The second, and arguably larger, disadvantage of the polyhedral gravity model is its computational requirement (87). High-fidelity shape models of asteroids can contain hundreds of thousands of vertices and facets which must be looped over at each propagation timestep. This can make it quite challenging to compute accelerations both in ground-based simulations and on-board spacecraft.

The alternatives to the polyhedral gravity model are the popular spherical harmonics gravity model (2) or its close cousin, the ellipsoidal harmonics gravity model (3). These gravity models provide slightly more forgiving assumptions about the body in question and represent the gravity field as the superposition of harmonic basis functions—the three-dimensional analogs to a Fourier series. These harmonics can provide a more representative estimate of the true gravity field than a point mass model, and they are most commonly expanded to relatively low degree and order to maintain computational tractability. These truncated low-order models make spherical harmonics easier to include within an orbit determination pipeline where the harmonic coefficients can be directly estimated over the mission lifetime.

Despite this, these harmonic gravity models are not without their own disadvantages. For one, these harmonics models rely on the assumption that the spacecraft will remain in orbit outside of the bounding sphere or ellipsoid. For missions that seek to land on the surface, or merely

attempt closer flybys of the object, this assumption can be operationally limiting. Moreover, these harmonic models are extremely inefficient at capturing discontinuity. Large gravitationally perturbing features like craters, boulders, mountain ranges, etc. can require hundreds-of-thousands of harmonics superimposed together before they are represented accurately (87). The harmonic coefficients can also be difficult to regress, requiring dense sampling requirements at low-altitude for observability.

As discussed throughout, the Physics-Informed Neural Network gravity model (PINN-GM) offers a compelling alternative to representing the gravity field of small-bodies. By learning, rather than prescribing, basis functions, the PINN-GM is able to produce high-fidelity gravity models without making any assumptions or imposing operational limits on the mission. While the PINN-GM offers a compelling solution to the gravity modeling problem given frequent training data, little research has been conducted to investigate how the PINN-GM performs when trained on realistic flight paths or online within an orbit determination pipeline. This chapter aims to fill these holes, investigating how well the PINN-GM can estimate the gravity field of the asteroid 433-Eros in-situ.

## 6.2 Offline Estimation

The PINN-GMs evaluated thus far assume that a prior gravity model exists from which training data can be generated. This assumption makes it easier to explore the model's sensitivity to data distribution, quantity, and measurement error, but fails to provide insight into how these models would perform when trained with data collected from following realistic flight paths. Consider a scenario where a spacecraft enters orbit around an asteroid for the first time. In this setting, no high-fidelity gravity model exists from which training data can be generated, so dynamicists must rely on a orbit determination pipeline to estimate positions and accelerations in-situ. This can be accomplished in one of two ways. First, the PINN-GM can be trained offline using estimates of the position and accelerations taken from an entirely decoupled filter. Alternatively, the PINN-GM can be directly embedded into the filter itself, assisting in the estimation of the position and acceleration directly, and using those estimates as training data online. This chapter investigates

both options, beginning with the offline case. Specifically, this section investigates the accuracy with which a PINN-GM-II can regress the gravity field of the asteroid 433-Eros following the flight path of the spacecraft NEAR-Shoemaker.

The NEAR-Shoemaker spacecraft entered orbit around 433-Eros on February 14, 2000 beginning a year long campaign comprised of 24 increasingly close orbits used to study the asteroid (72). A sample of these trajectories are plotted in Figure 6.1.

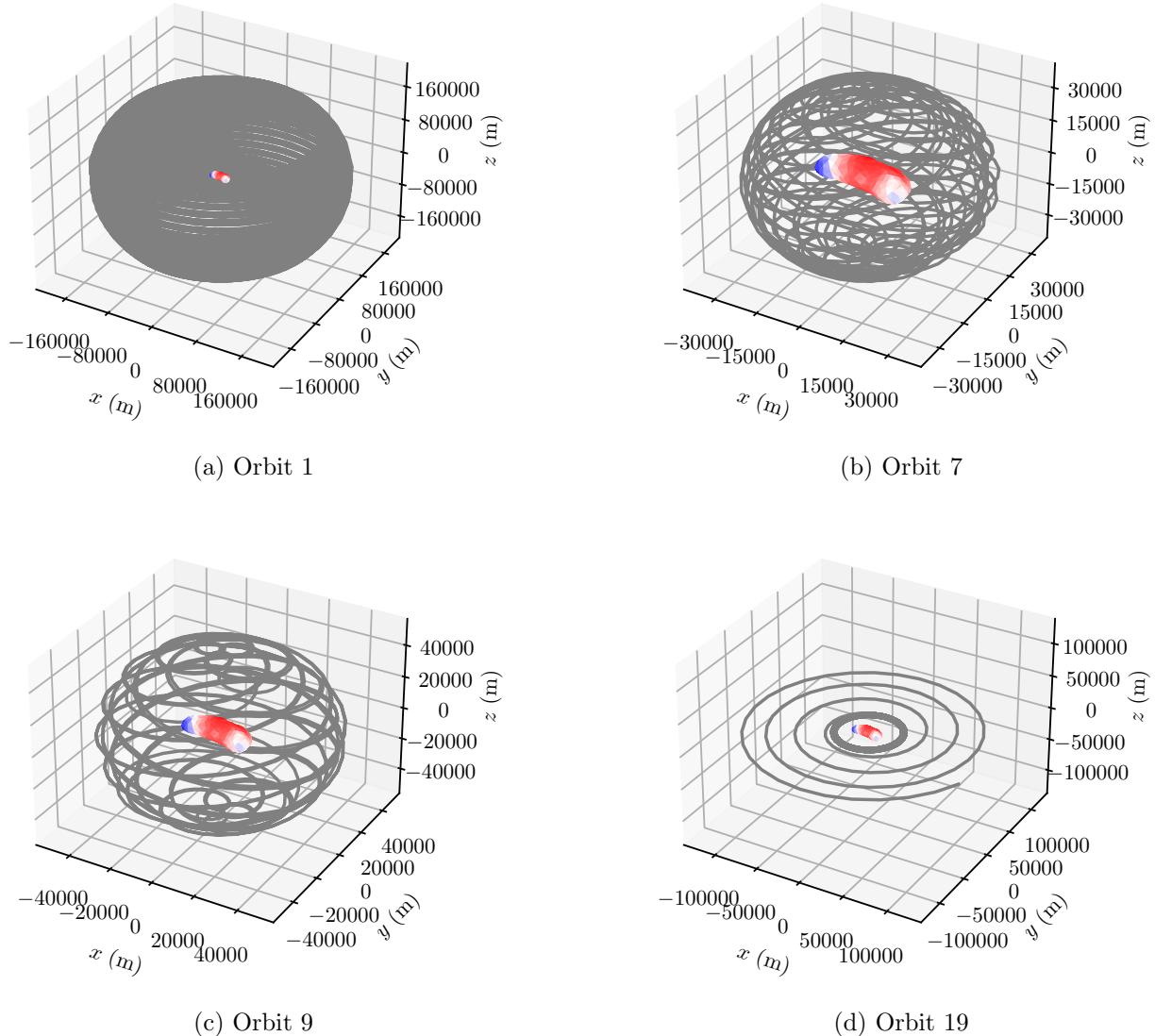


Figure 6.1: A random selection of NEAR-Shoemaker science orbits around 433-Eros

This experiment begins by assuming that only the gravitational parameter has been estimated, and an erroneous acceleration estimate is collected and stored once every 10 minutes during the mission lifetime. These assumptions are supported from two articles, References (129) and (91). The former article demonstrates that high-order gravitational accelerations can be directly estimated in a on-board filter as part of the state using high-order forms of dynamic model compensation. The latter article provides justification for the 10 minute cadence between state estimates in asteroids settings. These noisy acceleration estimates are generated by first calculating the true acceleration at the current spacecraft position, and then adding an error vector with a magnitude equal to 10% of the true acceleration magnitude in a random direction. It is worth emphasizing that this choice results in higher acceleration errors when the spacecraft is closer to the asteroid than when it is far away. While this choice is not necessarily reflective of true filter estimates and evolving covariances, it provides a challenging dataset that can be used to highlight the strengths and weaknesses of different gravity field recovery strategies.

The experiment progresses by generating the corresponding position and noisy acceleration data along the ephemeris of NEAR-Shoemaker. Each time the spacecraft transitions to a new orbital configuration, of which there are 24, all previously stored acceleration estimates are then used in a batched least squares algorithm to estimate three different spherical harmonic models of degree and order 4, 8, and 16 respectively. Likewise, a PINN-GM-II with the ALC cost function is also trained on this data and is given 7,500 epochs to converge.

There exists a rich body of literature discussing more advanced algorithms to estimate high-degree spherical harmonic models (130; 131; 132; 133). The choice to use a traditional least squares approach rather than these alternative algorithms is purposeful. Many of these more sophisticated algorithms rely on embedding heuristic insights about the structure of the spherical harmonic coefficients into the regression (e.g. using Kaula's rule for ridge regression), or they rely on the assumption that the expected measurement error has a mean of zero. Because the zero mean error requirement is not satisfied in this problem, and because the PINN-GMs are not given any additional user heuristics, traditional linear least squares regression is chosen to yield a more balanced

comparison between these two model types.

Figure 6.2 plots the acceleration percent error produced by each intermediate model as the spacecraft progresses through the mission. The model error is calculated by evaluating each model on 20,000 randomly distributed test data within each of the three major distributions shown in Figure 3.20 (exterior, interior, and surface). The lines corresponding to each test dataset distribution are plotted as solid, dashed, and dotted linestyles in the figure respectively. In addition, the altitude distribution of the spacecraft in each of the 24 orbits is shown as black violin plots to demonstrate at what point in time the spacecraft is close to the body and when it is far away. Note that some of the resulting curves/linestyles exceed the y-bounds of the figure due to their high error.

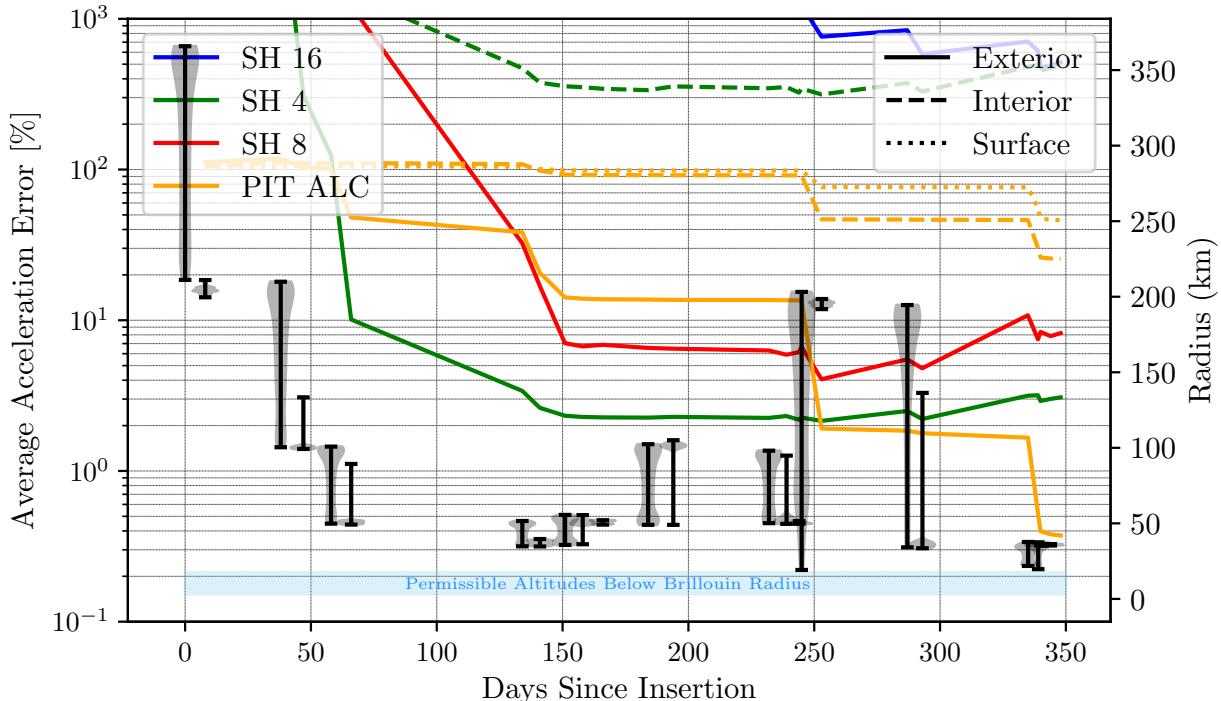


Figure 6.2: Error of Regressed PINN-GM Gravity Model of Eros without Ejecta

Figure 6.2 presents both advantages and drawbacks to using the PINN-GM-II for gravity field estimation. Early in the mission lifetime, when the spacecraft orbits at high-altitude, the low-degree spherical harmonic model performs best. This is not surprising, as low-degree spherical

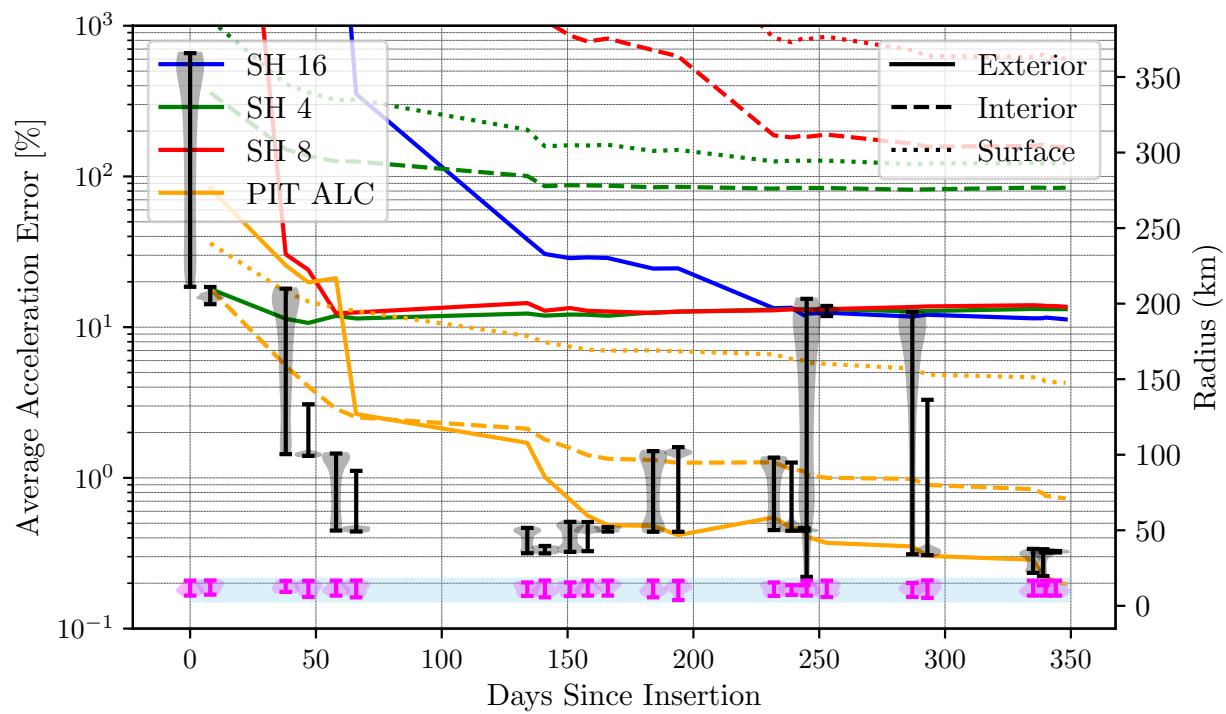


Figure 6.3: Error of Regressed PINN-GM Gravity Model of Eros with Ejecta Measurements (Magenta)

harmonic models are quite well-suited to represent this orbital regime where small wavelength features are heavily attenuated. In contrast, the PINN-GM-II originally struggles to converge in these high-altitudes as this generation model does not yet include the design changes to assist with numerical conditioning at high altitudes.

After approximately 60 days in orbit, the PINN-GM-II does begin to produce more robust models of the interior and surface gravity field than the spherical harmonic models. After 250 days in orbit, the PINN-GM also produces a more accurate model of the exterior gravity field than spherical harmonics. These results are attributed to the spacecraft's lower-altitude flybys of the asteroid. Because the PINN-GM-II cost function is sensitized to low-altitude data, these relatively brief low-altitude passes can make sizable improvements to the PINN-GM accuracy. This is best demonstrated in the near constant modeling error from day 170 to 240 for which the orbits maintained approximately the same minimum altitude. As soon as the spacecraft had a closer approach starting on day 240, the modeling error for the exterior distribution drops by nearly an order of magnitude from the new low-altitude pass. This effect is again seen on day 330.

Another interesting feature of Figure 6.2 is how the low degree spherical harmonic models outperform the high degree spherical harmonic models. Multiple factors contribute to this result. Foremost, the least squares algorithm does not have a way to filter out the error included within the acceleration estimates. A weighted least squares algorithm could be used instead, but such choice assumes that the error in the measurement is of zero mean for the solution to be optimal. This assumption is not true in this experiment. Second, because the high-degree models have greater modeling capacity than their low-degree counterparts, they are more susceptible to incorporating these errors in their solution, thereby producing a more erroneous fit when run on the test set. Encouragingly, the PINN-GM gravity model is not sensitive to these acceleration errors. By including the additional physics constraints in the cost function, the model remains desensitized to the error, similar to what is shown in Chapter 3.2.5.

Given the PINN-GM-II's sensitivity to the low altitude training data, a second experiment is proposed which seeks to exploit this phenomenon. The second experiment supplements the original

training dataset with infrequent acceleration estimates produced by particles tracked between the surface of the asteroid and the Brillouin radius (via ejecta events or gravity poppers). A total of 504 additional acceleration estimates are sampled from this range and are evenly distributed across the entire mission duration. This choice attempts to produce similar conditions to those found during the OSIRIS-REx mission in which a minimum of 600 particle observations were used to update the high-degree spherical harmonic coefficients (89).

By including these additional 504 measurements over a year-long mission, the PINN-GM=II achieves remarkably better performance than the spherical harmonics — converging to < 10% error within 10 days, and < 1% error after only 100 days (see Figure 6.3) — feats never accomplished by spherical harmonics over the entire mission lifetime. This rapid convergence suggests that if a mission could actively search for additional ejecta events near the surface of the body or artificially generate them via gravity poppers, the time necessary to characterize an asteroid’s gravity field could be reduced by an order of magnitude, saving valuable mission time and resources to be repurposed for closer approaches and additional science opportunities. Together, these results demonstrate how the PINN-GM is able to learn a model of the true field that is substantially more accurate and faster to converge than both its low- and high-degree spherical harmonic counterparts.

### 6.3 Online Estimation

The former experiment assumed that an independent orbit determination pipeline provided estimates of the spacecraft position and acceleration to then be used by a PINN-GM for training. While this approach is not intrinsically problematic, it remains an open question how the PINN-GM could be included within the orbit determination pipeline itself. Specifically, there currently exists no literature that shows how PINNs can be incorporated into filters and used to update the spacecraft state, covariance, and PINN-GM in-situ. This section seeks to address this gap, showing how a PINN-GM can be leveraged and trained within a Kalman filter. The performance of learned PINN-GM is then compared to traditional models in a small-body scenario around a heterogeneous density asteroid.

### 6.3.1 Kalman Filter

Traditionally, orbit determination pipelines leverage tools like Kalman filters. Kalman filters are online algorithms used to estimate relevant spacecraft state and environmental parameters given uncertain measurements (134). A comprehensive outline and derivation of the Kalman filter can be found in Reference (135), but the general framework is outlined here for convenience:

- (1) Initialize the filter with an initial reference state  $\mathbf{x}_0$ , the initial state deviation / error  $\Delta\mathbf{x}_0$ , the initial state covariance  $P_0$ , process noise covariance matrix  $Q$ , measurement noise covariance matrix  $R$ .
- (2) Obtain a measurement  $\mathbf{y}_i$
- (3) Propagate the reference state and state transition matrix (STM),  $\Phi(t_i, t_{i-1})$  to the time of the measurement through the differential equations

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{x}, \dot{y}, \dot{z}, \ddot{x}, \ddot{y}, \ddot{z} \end{bmatrix}^T, \quad \dot{\Phi} = \frac{\partial \dot{\mathbf{x}}}{\partial \mathbf{x}} \Phi \quad (6.1)$$

where  $\ddot{x}, \ddot{y}, \ddot{z}$  correspond to the perturbing accelerations like gravity, solar radiation pressure, and/or thrust vectors.

- (4) Use the STM to evolve the state error and the covariance matrix in time using:

$$\Delta x_i^- = \Phi \Delta x_{i-1}^+ \quad (6.2)$$

$$P_i^- = \Phi P_{i-1}^+ \Phi^T + Q_i \quad (6.3)$$

- (5) Update the state error and covariance matrix using information gathered from the mea-

surement through:

$$r_i = y_i - \hat{y}_i \quad (6.4)$$

$$H_i = \frac{\partial h}{\partial x} \Big|_{x_i} \quad (6.5)$$

$$K_i = P_i^- H_i^T (H_i P_i^- H_i^T + R_i)^{-1} \quad (6.6)$$

$$\Delta x_i^+ = \Delta x_i^- + K_i(r_i - H_i \Delta x_i^-) \quad (6.7)$$

$$P_i^+ = (\mathbb{I} - K_i H_i) P_i^- (\mathbb{I} - K_i H_i)^T \quad (6.8)$$

where  $h$  is the measurement function.

(6) Repeat for each incoming measurement

A few brief comments on the Kalman Filter. First, the Kalman filter algorithm typically leverages analytic forms of the equations of motion (EOM) such that the partial  $\frac{\partial \dot{x}}{\partial x}$  can be computed and used for the propagation of the STM. This is one of the many reasons that low-degree spherical harmonics gravity model are popular in this framework. The partials of the equation of motion can be derived analytically, and once programmed, can be efficiently evaluated. While an analytic form of the gravity field is convenient for this reason, it is not explicitly required, as the jacobian of the EOM can also be computed numerically through finite differencing. This simply comes with the cost of additional compute cycles and numerically truncated accuracy.

Second, the Kalman filter incorporates dynamical uncertainty into the state estimate through the process noise matrix  $Q_i$ .  $Q_i$  inflates the covariance matrix to avoid the filter from growing overconfident in its state estimate. The cost of this choice, however, is that any unmodelled dynamical signals that may exist beneath the noise floor will begin to lose observability. To avoid this possible loss of information, a different characterization of the process noise can be used referred to as dynamic model compensation.

### 6.3.2 Dynamic Model Compensation

Dynamical model compensation (DMC) assumes that there is some dynamical structure to the process noise that can be estimated over time and potentially exploited to reconstruct unmodelled dynamics. This is accomplished by assuming that there exists an underlying time-correlation between samples of the noise that can be modelled with as a first-order Gauss-Markov process:

$$\dot{\mathbf{w}} = -\frac{1}{\tau} \mathbf{w} \quad (6.9)$$

where  $\mathbf{w}$  is the noise vector, and  $\tau$  is a characteristic time scale of the process. Using this assumed dynamic model, the noise vector can be added to the original state and estimated over time. If the researcher prefers not to impose dynamical structure to the noise, the system can also be modeled with a zero-order hold approximation  $\dot{\mathbf{w}} = 0$  such that the filter is directly estimating the residual between the observed dynamics and the current equations of motion. More information regarding DMC can be found in Reference (129).

While DMC is most often employed to resolve unmodelled dynamics in spacecraft components like thrusters firing, it can also be used to estimate other accelerations like high-order gravitational perturbations. This section proposes using a zero-order DMC method ( $\dot{\mathbf{w}} = 0$ ) to estimate the unmodelled gravitational accelerations alongside the spacecraft position. Together, these estimates can form intermediate sets of training data from which a PINN-GM can be trained.

### 6.3.3 PINN-GM Kalman Filter

This section introduces the PINN-GM Kalman Filter (PINN-GM-KF), which provides the first demonstration of how to fully integrate PINN-GMs into classical orbit determination frameworks. A architectural glance of this framework is shown in Figure 6.4. The PINN-GM Kalman filter works by using an untrained PINN-GM-III within the EOM of the filter, and uses zero-order DMC to accumulate “observations” of the unmodelled gravitational accelerations at the estimated spacecraft position. When a sufficient number of position and acceleration data are accumulated,

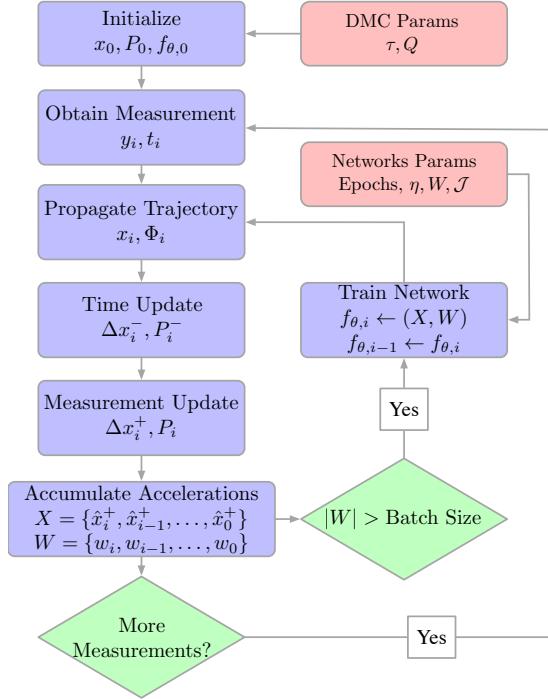


Figure 6.4: PINN-GM Kalman Filter

the PINN-GM can update its current model using stochastic gradient descent for a fixed number of epochs, and immediately redeployed into the filter's EOM.

Explicitly, as the spacecraft orbits the celestial body in question, the zero-order DMC will use the measurements (e.g., natural feature tracking, range and range-rate, etc.) to estimate the spacecraft position  $\bar{\mathbf{x}}$ , velocity  $\bar{\mathbf{v}}$ , and any accelerations not captured by the current dynamics model  $\mathbf{w} = \delta\bar{\mathbf{a}}$ . Consequently, these residuals in the dynamics can then be added to the PINN-GM's current estimate of the acceleration at estimated position,  $\mathbf{a}(\bar{\mathbf{x}})$ , to produce the best estimate of the true acceleration  $\mathbf{a}$  through

$$\mathbf{a}(\bar{\mathbf{x}}) = \bar{\mathbf{a}}(\bar{\mathbf{x}}) + \mathbf{w} \quad (6.10)$$

Together, the estimated position  $\bar{\mathbf{x}}$  and corrected acceleration vector  $\mathbf{a}(\bar{\mathbf{x}})$  then form an intermediate sets of training data for the PINN-GM. Note how the parameters of the PINN-GM are not represented as part of the state and estimated directly. This choice is purposeful, as it could lead to an exorbitantly large state vector depending on the size of the network, leading to pro-

hibitively expensive operations, particularly given the necessary covariance matrix inversions. By leaving the network to be updated using traditional machine learning techniques like stochastic gradient descent, the PINN-GM-KF avoids matrix inversions and can be updated at a frequency and computational expense level deemed appropriate by the navigation team.

As highlighted earlier, Kalman filters require a way to propagate the STM forward in time to then be used for updating the covariance and state error. This requires a way to compute the Jacobian of the equations of motion,  $\frac{\partial \dot{x}}{\partial x}$ . One might assume that when a PINN-GM is used within the EOM, the Jacobian must be computed numerically. However, the PINN-GM-KF enables the Jacobian to be computed exactly thanks to automatic differentiation as covered in Chapter 2 and leveraged in Chapter 5. This ensures that the PINN-GM-KF maintains the same high-accuracy and rapid executability of a traditional filter while using a more accurate and flexible gravity model.

Multiple advantages exist by leveraging a PINN-GM within the filter's EOM rather than a spherical harmonic model. Foremost, PINN-GMs are not prone to the same regressive difficulties as spherical harmonics. Spherical harmonics require carefully distributed data to estimate high-order frequencies, and the signal of these frequencies are extremely difficult to detect from high-altitudes. By using a PINN-GM instead, many of these challenges can be circumvented, as the PINN-GM can be trained with arbitrarily distributed data; iteratively improving its model based on where new information is gathered.

Another advantage of using the PINN-GM Kalman filter comes in the form of its efficient state representation. When using spherical harmonics models to represent an unknown gravity field, dynamicists must append many spherical harmonic coefficients to the state vector to then be estimated. This increases the dimensionality and corresponding computational burden on the filter. For context, estimating a simple degree and order 4 spherical harmonic gravity model requires an additional 26 additional state variables. This computationally penalty of this growth is exacerbated when considering the covariance matrix and associated matrix inversions required for the time update. The PINN-GM Kalman filter, in contrast, only requires appending three acceleration state variables,  $w$ , thereby maintaining a relatively small state space and computational efficiency.

Finally, the PINN-GM is uniquely designed to handle erroneous training data. As shown with the PINN-GM-II, the network cost function can be augmented to include additional differential constraints to ensure that the regressed model does not incorporate non-physically measurements into its solution. By requiring the PINN-GM to satisfy properties like  $\nabla^2 U = 0$  and  $\nabla \times \nabla U = 0$ , this model avoids accidentally incorporating non-conservative perturbations captured by DMC from corrupting the regressed model.

## 6.4 Problem Setup

Using the PINN-GM-KF framework, a small-body scenario is proposed where a spacecraft enters orbit around the asteroid 433-Eros and the quality of the learned PINN-GM is assessed. The spacecraft dynamics are influenced by the gravitational perturbations of the asteroid, solar radiation pressure, and 3rd body effects from the Sun. The spacecraft is assumed to have onboard sensors and algorithms capable of resolving noisy measurements of relative position with respect to the body to approximately 1 meter precision.

### Asteroid Gravity Model

For this scenario, the truth gravity field is constructed by superimposing three mass heterogeneities over a constant density polyhedral model. Specifically, a under-dense region is carved out of the center of the asteroid, and two over-dense regions are added to the two lobes of the asteroid. The over-dense mass elements are placed symmetrically about the center of the asteroid and the masses are chosen such that the center of mass and bulk properties / gravitational parameter of the asteroid remain fixed. The resulting heterogeneities are shown in Figure 6.5. This configuration is chosen to showcase how existing analytic models fail to capture heterogeneities. Figure 6.6 provides a visual depiction of this phenomenon, showing the error of a point mass gravity model approximation, a spherical harmonic model of degree and order 16, and a polyhedral gravity model with a constant density assumption. These models are considered “perfect”, i.e. given the assumptions of each model, the plotted results are the most accurate solutions that can be regressed. These

models will serve as baselines for the PINN-GMs trained within the PINN-GM-KF.

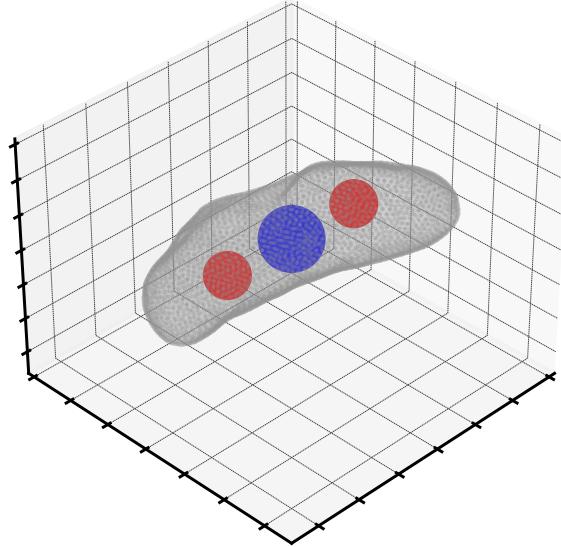


Figure 6.5: Heterogeneities of Asteroid

## Initial Conditions

The spacecraft is placed in orbit about the heterogeneous density asteroid with the initial orbital elements specified in Table 6.1. The asteroid is located 1.0 AU from the sun, has a gravitational parameter equal to  $446310.441 \text{ m}^3/\text{s}^2$  and rotates with a frequency of  $3.318\text{E-}4 \text{ rads/s}$ . The initial state is propagated for three orbit periods (corresponding trajectory shown in Figure 6.7), and position measurements are collected once every 60 seconds. The initial state, covariance, process noise matrix, and measurement noise matrix used to initialize the filter are listed in Table 6.2.

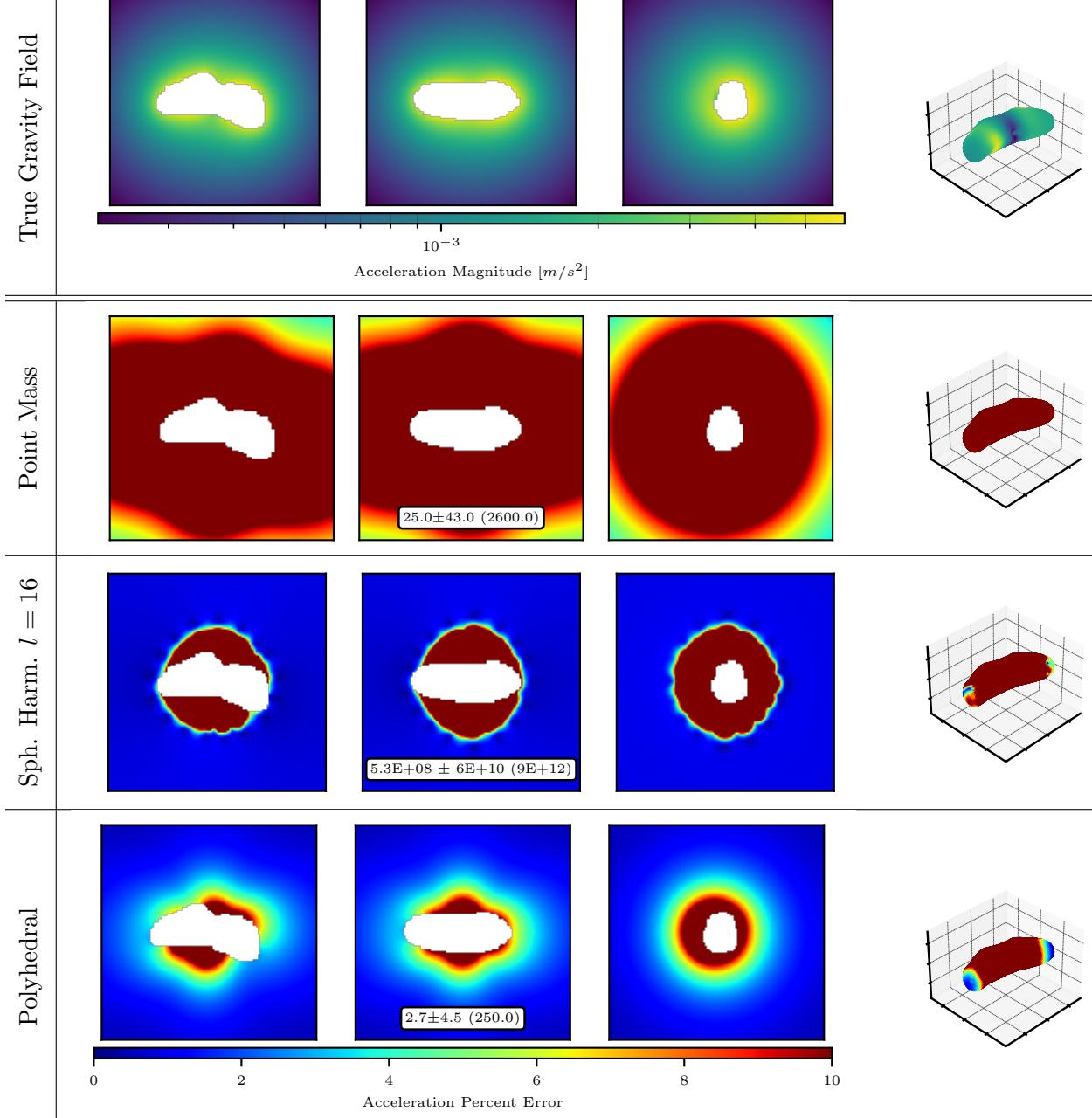


Figure 6.6: Top Row: The true heterogeneous density asteroid gravity field accelerations assessed along cartesian planes from  $[-2R, 2R]$ . Bottom Rows: Acceleration percent error for different gravity models capped at 10% error.

Table 6.1: Initial orbital elements

<b>a</b> [m]	<b>e</b> [-]	<b>i</b> [deg]	$\omega$ [deg]	$\Omega$ [deg]	$M$ [deg]
34,000	0.35	45	48.2	347.8	85.3

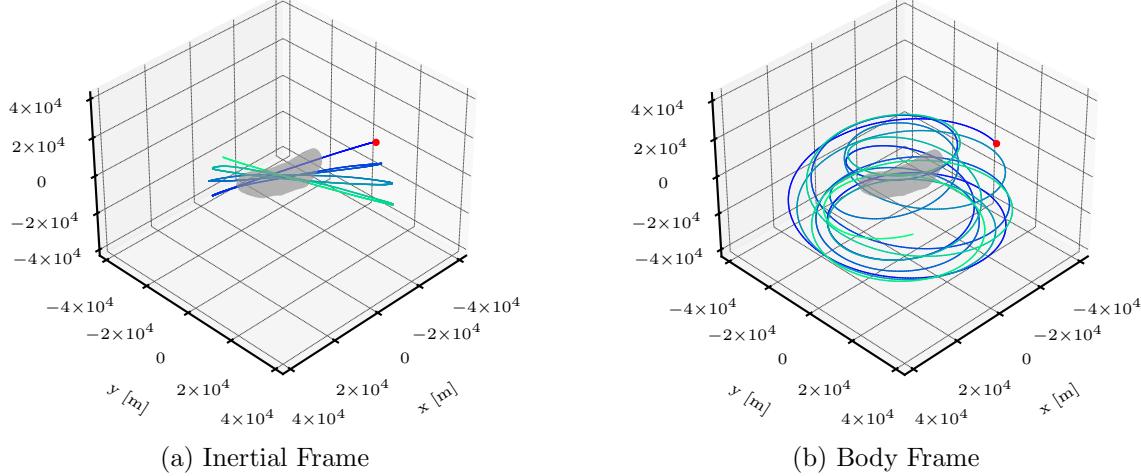


Figure 6.7: Spacecraft trajectory about the asteroid

## 6.5 Metrics

Four metrics are proposed to characterize the accuracy of the PINN-GM trained within the PINN-GM-KF. The first metric assesses the average acceleration percent error of the learned gravity model along the three cartesian planes (XY, XZ, YZ) extended between [-2R, 2R] where R is the radius of the asteroid. The field is evaluated on a 100x100 grid of points along each plane, and the average percent error is computed as

$$P = \frac{1}{N} \sum_{i=1}^N \frac{|\mathbf{a}_{\text{true}} - \mathbf{a}_{\text{PINN}}|}{|\mathbf{a}_{\text{true}}|} \quad (6.11)$$

The second and third metric evaluate the average acceleration percent error as a function of altitude. In this case, 1,000 data point are sampled from 0-3R, and the percent error of the acceleration vector is computed at each point. The average error within the Brillouin sphere 0-1R is computed and referred to as the interior error, and the average error outside the Brillouin sphere (1R-3R) is referred to as the exterior error.

Table 6.2: Initial state, covariance, process noise matrix, and measurement noise matrix

State	Value	Unit
$x_0$	[ -19243.60, 21967.51, 17404.75 ]	[m]
$v_0$	[ -2.9396, -1.1707, -1.7654 ]	[m/s]
$w_0$	[ 0 , 0, 0 ]	[m/s <sup>2</sup> ]
$P_{x_0}$	diag([100, 100, 100])	[m <sup>2</sup> ]
$P_{v_0}$	diag([0.01, 0.01, 0.01])	[m <sup>2</sup> /s <sup>2</sup> ]
$P_{w_0}$	diag([1E-14 , 1E-14, 1E-14])	[m <sup>2</sup> /s <sup>4</sup> ]
$Q$	diag([1E-14, 1E-14, 1E-14])	[m <sup>2</sup> /s <sup>2</sup> ]
$R$	diag([1E-3, 1E-3, 1E-3])	[m <sup>2</sup> ]

The fourth metric evaluates the accumulated trajectory propagation error over one orbital period for four distinct orbits through:

$$\mathcal{S} = \sum_{i=0}^4 \int_{t=0}^T \Delta \mathbf{X}_i(t) dt \approx \sum_{i=0}^4 \sum_{j=0}^N \Delta \mathbf{X}_i(t_j) \quad (6.12)$$

This provides a broad assessment of the generalizability of the learned gravity model to different orbital regimes. The four orbits tested are reported in Table 6.3.

These metrics are evaluated for the three most common gravity models — point mass, spherical harmonics, and polyhedral — and are reported in Table 6.4. Of the analytic models, it is clear that the polyhedral gravity model is the most robust option, with an average error in the cartesian planes of 2.6% and an average error of less than 5.6% within the Brillouin sphere. The spherical harmonic model is the next best option; however, the average error over the cartesian planes is heavily biased by the diverging behavior within the Brillouin sphere. Instead, the spherical harmonics true utility is in the higher altitude regimes  $> 1R$  for which the average error is 0.85%. The trajectory experiment furthers these conclusions, showing how the polyhedral gravity model accumulates 234 kilometers of error over the four orbits, whereas the spherical harmonics and the point mass model are approximately 4 to 20 times worse.

Orbit	$\mathbf{a}$ [m]	$\mathbf{e}$ [-]	$\mathbf{i}$ [deg]	$\omega$ [deg]	$\Omega$ [deg]	$M$ [deg]
Inclined Low-Altitude	34,000	0.001	45	48.2	347.8	85.3
Equatorial Low-Altitude	34,000	0.001	0	0.0	180.0	85.3
Polar Low-Altitude	34,000	0.001	90	48.2	347.8	85.3
Polar High-Altitude	68,000	0.001	90	48.2	347.8	85.3

Table 6.3: Orbits for Trajectory Metric

Model	Planes %	Interior %	Exterior %	dX Sum [km]	Time [s]
Point Mass	24.8	50.4	2.52	4304.6	0.06
Sph. Harm. ( $l = 16$ )	1.7E8	1.3E6	0.85	946.9	0.09
Polyhedral	2.7	5.6	0.24	234.1	106.02
PINN PM-Init	8.1	17.8	2.91	2197.0	22.7
PINN Poly-Init	0.51	1.57	0.07	40.4	34.1

Table 6.4: Table of the metrics for the standard gravity models.

## 6.6 Experiments

Two experiments are proposed to characterize the performance of the PINN-GM-KF. First is a hyperparameter experiment that investigates which parameters of the PINN-GM-KF have the greatest impact on the learned gravity model quality. The second experiment investigates the impact of the orbit geometry on model performance.

### 6.6.1 Hyperparameter Search

There is considerable design choice when determining when to train the PINN-GM within the Kalman filter which can be represented in the form of hyperparameters. Specifically, the hyperparameters studied in this experiment are listed in Table 6.5 and can be separated into two groups: the PINN-GM parameters and the EKF parameters.

The PINN-GM parameters include the learning rate, mini-batch size, epochs, measurement batch, and loss function. The learning rate dictates the step size of the gradient descent update for the network. Small values are associated with slower learning, but more stable gradient descent whereas large learning rates can lead to faster learning that may be less stable. The mini-batch size determines how many data are used to compute an estimate of the gradient. Large mini-batch sizes are more representative of the true cost landscape and corresponding gradient, but can lead to local minima whereas small batch sizes can lead to a more stochastic search of the cost landscape helping to escape local minima. The number of epochs are how many times the dataset is iterated over. Too many epochs can lead to overfitting to the training data, whereas too few

can lead to underfitting. The measurement batch dictates how many position and acceleration estimates are required before updating the PINN-GM. The filter can be configured to train the PINN-GM after every measurement update, or it can be configured to train the PINN-GM after every  $n$  measurement updates. The former is more computationally expensive, but immediately integrates new information of the system into the dynamics model, whereas as the latter is more computationally efficient, but prolongs the period in which the filter is using an erroneous model. For filters which are better suited to capture non-linear dynamics and modeling errors (extended and unscented Kalman filters) the later concern may be less important, but for filters like the traditional Kalman filter which are more sensitive to modeling errors, the former may be more appropriate. Finally, the loss function is the function used to compute the error between the network prediction and the training data. Past work with the PINN-GM highlights how single physics-informed loss constraints can lead to more accurate models in the presence of perfect data, whereas multiple physics-informed loss constraints can lead to more robust models in the presence of noisy data.

Hyperparameter	Values
$q$	[1e-9, 1e-8, 1e-7]
Epochs	[10, 100, 1000]
Learning Rate	[1e-4, 1e-5, 1e-6]
Mini-Batch Size	[256, 2048, 32768]
Measurement Batch	[1024, 2048, 4096]
Training Fcn	[A, AL]
Measurement Quality	[Perfect, Noisy]
PINN Init	[Point Mass, Polyhedral]

Table 6.5: Hyperparameters

The hyperparameters specific to the EKF are the process noise and measurement noise. The process noise is used to account for unmodelled dynamics and ensure the filter does not become overconfident in the state estimate. A large process noise magnitude will provide more conservative state estimates with greater uncertainty bounds, whereas small amounts of process noise will have more precise state estimates that could potentially be overconfident. Similarly the measurement

noise is used to tune how much uncertainty to place on the incoming measurements. Large amounts will cause the filter and corresponding state estimates to rely more heavily on the dynamics, whereas small amounts will encourage the filter to override the expectation of the dynamics and place greater weight on the incoming measurements.

The last hyperparameter is the PINN initialization scheme. Two initialization options exist: (1) a point mass initialization and (2) a polyhedral initialization. For the point mass initialization, the PINN-GM is trained under the assumption that the bulk parameter of the asteroid has been estimated a-priori. This assumes ground-based radar measurements have produced a coarse shape model of the asteroid, and its composition / class has been assessed from spectroscopy. With this information, the PINN is pre-trained to arbitrarily levels of precision with artificial data generated by a point mass approximation. This choice, combined with design choices introduced in PINN-GM-III, guarantees that the PINN-GM produces an estimate of the gravitational dynamics that is no worse than the point mass gravity model <sup>1</sup>.

The polyhedral initialization is a more optimistic initialization. In this case, the PINN-GM is trained under the assumption that the asteroid has been characterized with a high-fidelity shape model. Specifically, the PINN-GM is pre-trained on a constant density polyhedral gravity model to arbitrary levels of accuracy. Note that the polyhedral will not be accurate due to the constant density assumption, so the responsibility of the PINN-GM becomes to learn the discrepancies between the constant density assumption and the true heterogeneous density distribution.

Two hyperparameter searches are performed. The first hyperparameter experiment studies the ideal circumstances for training the PINN-GM-KF. The search assumes that perfect position measurements exist, and that the PINN-GM is initialized / pre-trained with a constant density polyhedral model. In contrast, the second hyperparameter search is conducted assuming that only noisy measurements exist, and the PINN-GM is only initialized with the point mass approximation. The corresponding results are plotted for the best and worse cases in Figures 6.8 and 6.10

---

<sup>1</sup> Recall that the PINN-GM-III embeds the point mass approximation directly into model, such that the model is guaranteed to converge to a point mass solution in the limit of  $r \rightarrow \infty$

respectively.

## Hyperparameter Results

Figure 6.8 demonstrates that there exist many hyperparameter combinations of the PINN-GM-KF for which the learned PINN-GM outperforms the best point mass, spherical harmonic, and polyhedral gravity models across all three baseline metrics. In the case of the planes metric, the PINN-GM achieves errors as low as 0.51%, a 80% improvement over the constant density polyhedral model. In the trajectory experiment, the best performing model accumulates approximately 40 km of error, nearly an order of magnitude less error than 230km of the constant density model. Inside the Brillouin sphere, the PINN-GM achieves an average error of 1.57% in contrast to the 5.6% of the constant density polyhedral model, and outside the sphere the error is as low as 0.07% rather than the original 0.24%.

In this more challenging and realistic scenario of Figure 6.10, where the PINN-GM-KF only has access to noisy measurements and a point mass approximation, similar results are shown. For nearly all hyperparameter configuration, the PINN-GM trained within the filter achieves considerably better performance than the point model it is pre-trained on. For the planes experiment, the PINN-GM achieves a lowest average error of approximately 8.1% average error, whereas the point mass baseline is 24.8% — a 67% improvement. Inside the Brillouin sphere, the error decreased from 50.4% to 17.8%. Outside the Brillouin sphere, the error did increase from 2.5% to 2.91% — an effect that is currently attributed to the noisy measurements. Despite this increase in error, the trajectory experiment the point mass model accumulates approximately 2200 km of trajectory error across the four orbits whereas the PINN-GM accumulates 4300 km. While the PINN-GM pretrained on the point mass model does not perform as well as the polyhedral model, this still demonstrates a sizable improvement in the gravity model performance over a mere three orbits, suggesting it can be used as an intermediate solution until a shape model is resolved, at which point the model can be retrained with the available data to achieve performance on par with the polyhedral initialization.

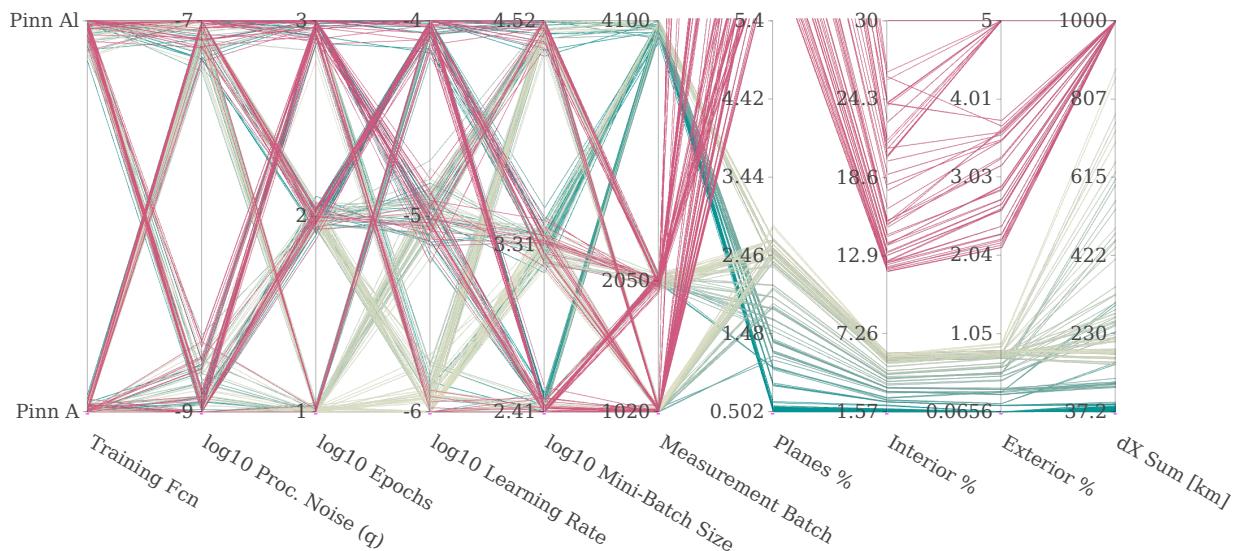


Figure 6.8: Hyperparameter search of the PINN-GM initialized with the constant density polyhedral model.

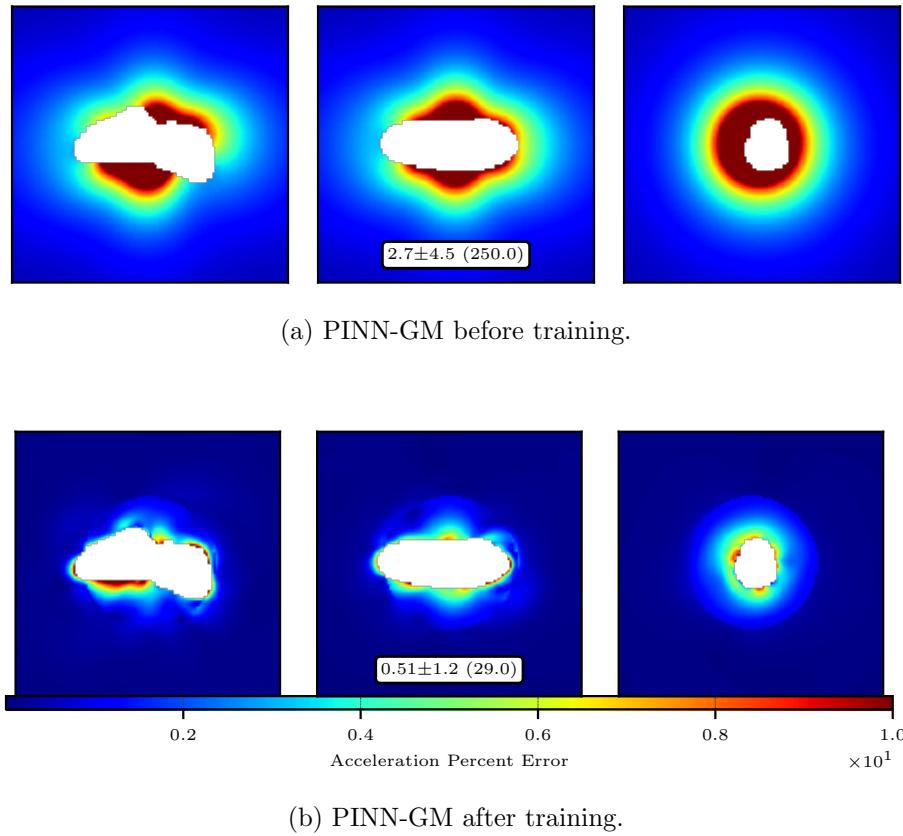


Figure 6.9: Best performing PINN-GM initialized with constant density polyhedral gravity model

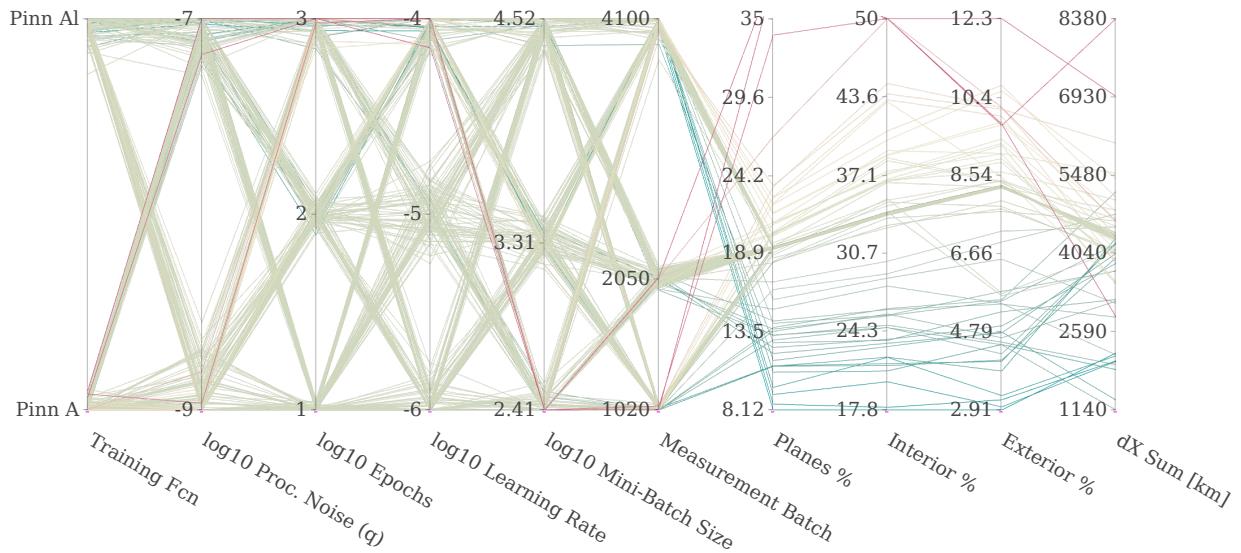


Figure 6.10: Hyperparameter search of the PINN-GM initialized with the point mass model.

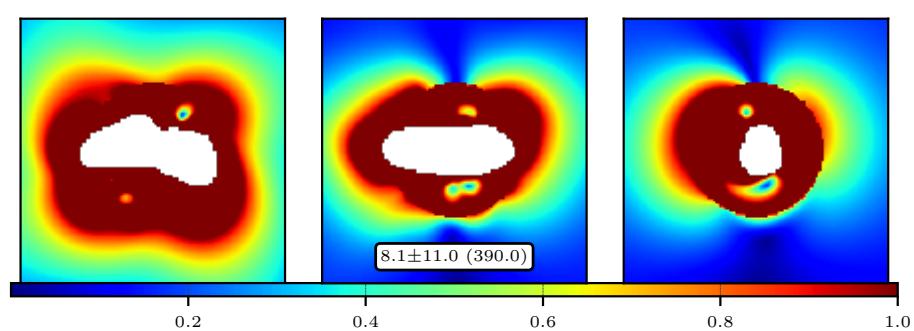
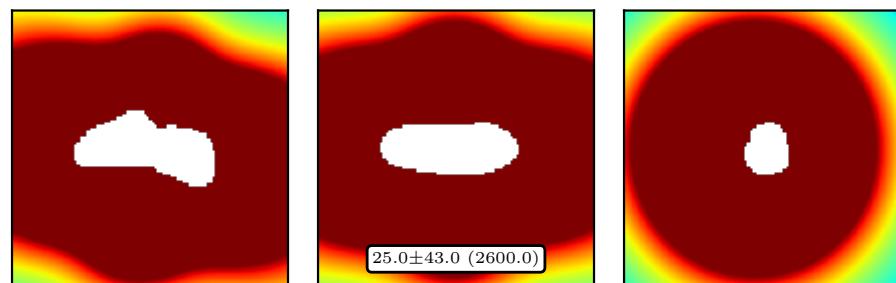


Figure 6.11: Best performing PINN-GM initialized with point mass gravity model

These hyperparameter searches demonstrate that the PINN-GM-KF is most sensitive to the size of the measurement batch size. Larger measurement batches ensure that the PINN-GM-KF has enough data to accurately constrain the network during gradient descent. Smaller measurement batches can still produce reliable models, however, this requires smaller learning rates to ensure the model does not accidentally descend into a local minima. There also exist hyperparameter configurations for which the learned PINN-GM performs worse than the baselines. In particular, large learning rates with small batch sizes and many training epochs are prone to decreasing model accuracy. Small batch sizes paired with large learning rates are generally frowned upon, for reasons previously mentioned, so this matches expectation. These results suggest that less frequent updates with larger batch sizes will generally be considered the safer choice.

These results are particularly encouraging given the constraints placed on the experiment. These hyperparameter searches are conducted over only three orbits with measurements of the position taken once every minute. This totals to less than 3,000 data points, and yet even given these relatively sparse and noisy data, the learned PINN-GM is capable of improving its performance in each case.

### 6.6.2 Sensitivity to Orbit Geometry

The final experiment studies how the learned PINN-GM performance is effected by the initial orbit geometry. In particular, this study investigates how sensitive the learned PINN-GM solution is to the altitude and eccentricity of the initial orbit. Explicitly, the spacecraft's initial semi-major is varied between  $2R$  and  $3R$  in increments of  $0.1R$ , and its eccentricity is varied between 0.0 and 0.5 in increments of 0.05. These values are chosen to ensure that no orbit ever intersects the surface of the asteroid. The PINN-GM-KF tested is initialized with the polyhedral model using the best hyperparameters found in Figure 6.10 and is run for three orbit periods for each respective orbit geometry. The corresponding results of the planes, interior, exterior, and trajectory experiment are plotted as histograms in Figure 6.12.

Figure 6.12 shows that When the orbit is eccentric, the PINN-GM tends to perform bet-

ter than in near circular regimes. This is attributed the fact that the radial component of the acceleration vector experiences the greatest variability and observability of all the components. Consequently, when orbiting at different radii, the PINN-GM is given a much more diverse set of training from which it can infer the true field. When the spacecraft orbits at a near constant radius, the acceleration data will appear more homogeneous and the PINN-GM struggles to resolve more productive basis functions to represent the entire field. The semi-major axis also plays a role in model accuracy, as lower altitudes will produce larger gravity signals in the more exotic regimes helping to constrain a model at both low and high-altitudes. For each geometry, however, the PINN-GM solution achieves lower error across all four metrics than the constant density polyhedral gravity model on which the PINN-GM was originally trained demonstrating the robustness of the PINN-GM-KF in a variety of orbital conditions.

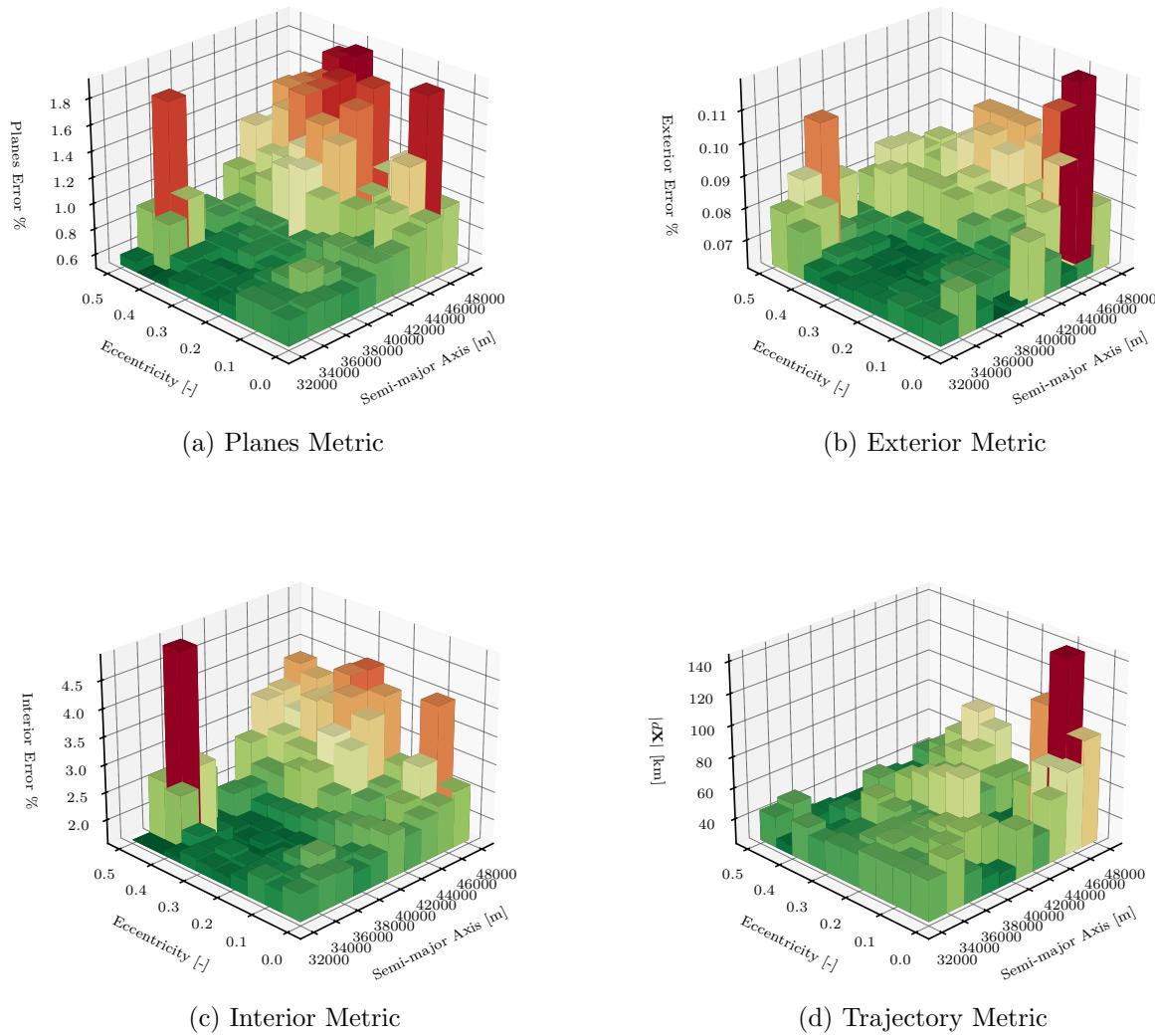


Figure 6.12: Metrics as a function of orbit geometry.

## Chapter 7

### Conclusions

The Physics-Informed Neural Network gravity model (PINN-GM) is a powerful new way to efficiently and accurately model complex gravity fields. Unlike past data-driven gravity models, the PINN-GM blends traditional deep learning techniques with dynamical systems theory to ensure that the solution learned is encouraged to comply with the underlying physics of the system. These novel gravity models are powerful without modification; however, additional design improvements enable greater accuracy, data efficiency, and robustness that is currently unparalleled by other numerical models. As these models continue to mature, there exist many valuable applications to which they can be applied including reinforcement learning, orbit discovery, and state estimation.

Explicitly, this thesis demonstrates that the PINN-GM is able to produce high-accuracy models for the gravity fields of the Earth, Moon, and the asteroid 433-Eros. By learning unique basis functions, rather than prescribing them, the PINN-GM is able to construct solutions which use orders-of-magnitude fewer parameters than their conventional counterparts. In turn, these more compact models are also much faster to evaluate, offering benefits for on-board capabilities and simulation. The design of these PINN-GMs has considerable impact on their performance and robustness, as demonstrated through the careful feature engineering, choice of cost function, enforcement of boundary conditions, and architecture design put forth in this work. In combination, these design choices ensure that the PINN-GM always produces gravity field estimates are, at worst, equally performant to the analytic models used today.

In addition to its robust performance, another significant property of the PINN-GM is its

universality. Until now, no other gravity model could simultaneously avoid assumptions about the body in question, operate across all orbital regimes, achieve small memory footprints, be rapidly executable, and maintain exact differentiability. The PINN-GM achieves all of these things without needing large amounts of unbiased training data to regress. This make the model equally functional in both large- and small-body contexts. While the PINN-GM is a compelling new solution to this problem, it should be recognized that this could not have been accomplished without the many efforts of past dynamicists and their corresponding gravity models. The spherical harmonics and polyhedral gravity models are foundational to this work and remain important models to the community. Moreover, the work of authors who study different forms of data-driven models are equally important to recognize, as they provide further context highlighting how machine learning can prove to be a valuable tool for astrodynamacists.

The PINN-GM poses a number of advantages for applications within astrodynamics. By leveraging the PINN-GM within the environments used to train reinforcement learning agents deployed on spacecraft, researchers can produce higher-quality agents in a fraction of the time required when using alternative models. For mission designers seeking to identify stable, periodic orbits around irregularly shaped bodies, the PINN-GM's differentiability makes it trivial to conduct this search in a variety of orbital element descriptions. The PINN-GM even finds use in orbit determination pipelines, and its data efficiency allows for the rapid estimation of a gravity field in less mission time than current approaches.

While the findings presented in this thesis are encouraging, work remains to establish communal standards for measuring the efficiency and accuracy of a gravity model. This work proposes experiments that measure average integration error, model accuracy as a function of model parameters, percent error in the cartesian planes or as a function of altitude, and others; however, these are not the only ways to measure model performance. It is possible that other researchers may develop experiments in the future which identify potential pitfalls of this model, and these alternative measures are welcomed, as they will inform the next generations of gravity models. In fact, others are already beginning to explore different machine learning methods to solve the gravity

modeling problem. The work of Reference (7) is of particular note, offering investigations for how tools like Neural Radiance Fields can supply candidate density distributions and gravity models when integrated. As machine learning models continue to develop, there will inevitably be more sophisticated, and likely accurate, ways to produce high-fidelity gravity models that warrant attention. In the meantime, the PINN-GM offer a compelling intermediate solution. With relatively little modification, these models can be deployed within simulation or even exported to flight code. Pending thorough testing and validation, these models may one day offer ephemeris-like accuracy with point mass-like speed.

Beyond the work presented in this thesis, questions remain about how PINNs can be improved further. For example, it is well known that the ocean tides, ground water reservoirs, and glacial melt all have detectable effects on Earth's gravity field. It remains an open question if PINNs can be designed to capture these temporal variations, and what sorts of science might be enabled with these data products. Given sufficient data, it is also possible these models can potential help identify candidate mineral deposits, or resolve geometries of unknown rock formations within the lithosphere by blending gravitational and magnetic field data. For the planetary scientists, these gravity models can even offer candidate density estimates for planetesimals thanks to their exact differentiability and Poisson's equation.

Beyond planetary science, there are also questions within the machine learning community that may further benefit these gravity models. For example, how might orthogonality constraints be leveraged in the learned basis functions? Can larger networks and Fourier feature mapping help resolve small-scale features with greater ease? Can convolutional structures be embedded within the model to better preserve geometric information about the gravitational perturbations? There also remains the important question of how uncertainty be quantified for these models. These questions, among others, are all worthy of investigation before these models should be considered fully mature.

Finally, there also exists the much broader set of problems that can potentially benefit from the lessons gathered through this work. While gravity modeling is interesting dynamical system that

previously lacked a universal model, many other complex dynamical systems exist that continue to leverage cumbersome analytics. Within astrodynamics, the force of solar radiation pressure (SRP) is one such example. SRP is notoriously expensive to represent to high-fidelity; requiring ray-traced solutions with long compute times, or gross oversimplifications like is found with the cannonball or flat plate models. PINNs and other machine learning tools could be used to find more efficient alternatives for these systems as well.

Taken together, PINNs are an exciting tool emerging out of the scientific machine learning community that hold tremendous potential in their ability to produce high-fidelity dynamics models of complex systems. This thesis presents one such application for astrodynamicists in the form of the PINN gravity model. Requiring relatively little training data, PINN gravity models are able to regress impressively high-accuracy models that can be deployed across a variety of environments and problems within the community. While the careful curation of these models have offered high yield thus far, future work remains to realize their full potential. These efforts, among others, may continue to prove fruitful for the next aspiring dynamicist who shares an enthusiasm for machine learning.

## Bibliography

- [1] J. Martin and H. Schaub, “Physics-Informed Neural Networks for Gravity Field Modeling of Small Bodies,” *Celestial Mechanics and Dynamical Astronomy*, p. 28, Sept. 2022.
- [2] W. M. Kaula, *Theory of Satellite Geodesy: Applications of Satellites to Geodesy*. Waltham, Mass.: Blaisdell Publishing Co, 1966.
- [3] G. Romain and B. Jean-Pierre, “Ellipsoidal harmonic expansions of the gravitational potential: Theory and application,” *Celestial Mechanics and Dynamical Astronomy*, vol. 79, no. 4, pp. 235–275, 2001.
- [4] S. Tardivel, “The Limits of the Mascons Approximation of the Homogeneous Polyhedron,” in *AIAA/AAS Astrodynamics Specialist Conference*, no. September, (Reston, Virginia), pp. 1–13, American Institute of Aeronautics and Astronautics, Sept. 2016.
- [5] R. Werner and D. Scheeres, “Exterior gravitation of a polyhedron derived and compared with harmonic and mascon gravitation representations of asteroid 4769 Castalia,” *Celestial Mechanics and Dynamical Astronomy*, vol. 65, no. 3, pp. 313–344, 1997.
- [6] R. Furfaro, R. Barocco, R. Linares, F. Topputo, V. Reddy, J. Simo, and L. Le Corre, “Modeling irregular small bodies gravity field via extreme learning machines and Bayesian optimization,” *Advances in Space Research*, no. June, 2020.
- [7] D. Izzo and P. Gómez, “Geodesy of irregular small bodies via neural density fields,” *Communications Engineering*, vol. 1, p. 48, Dec. 2022.
- [8] G. E. Karniadakis, I. G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, and L. Yang, “Physics-informed machine learning,” *Nature Reviews Physics*, vol. 3, pp. 422–440, May 2021.
- [9] M. Raissi, P. Perdikaris, and G. Karniadakis, “Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations,” *Journal of Computational Physics*, vol. 378, pp. 686–707, Feb. 2019.
- [10] R. Reinhard, “The Giotto encounter with comet Halley,” *Nature*, vol. 321, pp. 313–318, May 1986.
- [11] T. Johnson, C. Yeates, and R. Young, “Space science reviews volume on Galileo Mission overview,” *Space Science Reviews*, vol. 60, May 1992.
- [12] L. Prockter, S. Murchie, A. Cheng, and J. Trombka, “THENEAR SHOEMAKEMR IS-SIONTO ASTEROID433 EROS,”

- [13] M. D. Rayman, P. Varghese, D. H. Lehman, and L. L. Livesay, “Results from the Deep Space 1 technology validation mission,” *Acta Astronautica*, vol. 47, pp. 475–487, July 2000.
- [14] D. E. Brownlee, P. Tsou, D. Burnett, B. Clark, M. S. Hanner, F. Horz, J. Kissel, J. A. M. McDonnell, R. L. Newburn, S. Sandford, Z. Sekanina, A. J. Tuzzolino, and M. Zolensky, “The STARDUST Mission: Returning Comet Samples to Earth,” *Meteoritics and Planetary Science Supplement*, vol. 32, p. A22, Jan. 1997.
- [15] W. H. Blume, “Deep Impact Mission Design,” *Space Science Reviews*, vol. 117, pp. 23–42, Mar. 2005.
- [16] K.-H. Glassmeier, H. Boehnhardt, D. Koschny, E. Kührt, and I. Richter, “The Rosetta Mission: Flying Towards the Origin of the Solar System,” *Space Science Reviews*, vol. 128, pp. 1–21, Feb. 2007.
- [17] J. Kawaguchi, A. Fujiwara, and T. Uesugi, “Hayabusa—Its technology and science accomplishment summary and Hayabusa-2,” *Acta Astronautica*, vol. 62, pp. 639–647, May 2008.
- [18] S.-i. Watanabe, T. Saiki, and S. Nakazawa, “Hayabusa2 Mission Overview,”
- [19] D. S. Lauretta, S. S. Balram-Knutson, E. Beshore, W. V. Boynton, C. Drouet d’Aubigny, D. N. DellaGiustina, H. L. Enos, D. R. Golish, C. W. Hergenrother, E. S. Howell, C. A. Bennett, E. T. Morton, M. C. Nolan, B. Rizk, H. L. Roper, A. E. Bartels, B. J. Bos, J. P. Dworkin, D. E. Highsmith, D. A. Lorenz, L. F. Lim, R. Mink, M. C. Moreau, J. A. Nuth, D. C. Reuter, A. A. Simon, E. B. Bierhaus, B. H. Bryan, R. Ballouz, O. S. Barnouin, R. P. Binzel, W. F. Bottke, V. E. Hamilton, K. J. Walsh, S. R. Chesley, P. R. Christensen, B. E. Clark, H. C. Connolly, M. K. Crombie, M. G. Daly, J. P. Emery, T. J. McCoy, J. W. McMahon, D. J. Scheeres, S. Messenger, K. Nakamura-Messenger, K. Righter, and S. A. Sandford, “OSIRIS-REx: Sample Return from Asteroid (101955) Bennu,” *Space Science Reviews*, vol. 212, pp. 925–984, Oct. 2017.
- [20] K. Berry, K. Getzandanner, M. Moreau, P. Antreasian, A. Polit, M. Nolan, H. Enos, and D. Lauretta, “REVISITING OSIRIS-REX TOUCH-AND-GO (TAG) PERFORMANCE GIVEN THE REALITIES OF ASTEROID BENNU,”
- [21] NASA/JPL/JHUAPL, “Eros.jpg.”
- [22] ISAS/JAXA, “Itokawa.jpg.”
- [23] B. D. Tapley, M. M. Watkins, F. Flechtner, C. Reigber, S. Bettadpur, M. Rodell, I. Sasgen, J. S. Famiglietti, F. W. Landerer, D. P. Chambers, J. T. Reager, A. S. Gardner, H. Save, E. R. Ivins, S. C. Swenson, C. Boening, C. Dahle, D. N. Wiese, H. Dobslaw, M. E. Tamisiea, and I. Velicogna, “Contributions of GRACE to understanding climate change,” *Nature Climate Change*, vol. 9, pp. 358–369, May 2019.
- [24] J. L. Chen, C. R. Wilson, B. D. Tapley, J. S. Famiglietti, and M. Rodell, “Seasonal global mean sea level change from satellite altimeter, GRACE, and geophysical models,” *Journal of Geodesy*, vol. 79, pp. 532–539, Dec. 2005.
- [25] B. Wouters, A. S. Gardner, and G. Moholdt, “Global Glacier Mass Loss During the GRACE Satellite Mission (2002–2016),” *Frontiers in Earth Science*, vol. 7, p. 96, May 2019.

- [26] B. D. Vishwakarma, “Monitoring Droughts From GRACE,” *Frontiers in Environmental Science*, vol. 8, p. 584690, Dec. 2020.
- [27] J. T. Reager, B. F. Thomas, and J. S. Famiglietti, “River basin flood potential inferred using GRACE gravity observations at several months lead time,” *Nature Geoscience*, vol. 7, pp. 588–592, Aug. 2014.
- [28] M. Mandea, V. Dehant, and A. Cazenave, “GRACE—Gravity Data for Understanding the Deep Earth’s Interior,” *Remote Sensing*, vol. 12, p. 4186, Dec. 2020.
- [29] V. Mikhailov, S. Tikhotsky, M. Diament, I. Panet, and V. Ballu, “Can tectonic processes be recovered from new gravity satellite data?,” *Earth and Planetary Science Letters*, vol. 228, pp. 281–297, Dec. 2004.
- [30] E. D. Kaplan and C. Hegarty, eds., *Understanding GPS: Principles and Applications*. Artech House Mobile Communications Series, Boston: Artech House, 2nd ed ed., 2006.
- [31] R. H. Rapp, “Past and Future Developments in Geopotential Modeling,” in *Geodesy on the Move* (K.-P. Schwarz, R. Forsberg, M. Feissel, and R. Dietrich, eds.), vol. 119, pp. 58–78, Berlin, Heidelberg: Springer Berlin Heidelberg, 1998.
- [32] F. J. Lerch, C. A. Wagner, D.-E. Smith, J. E. Brovnd, and S. A. Richardson, “GRAVITATIONAL FIELD MODELS FOR THE EARTH (GEM 1 & 2),”
- [33] F. J. Lerch, O. M. Klosko, and G. B. Patel, “Technical Memorandum 84986,”
- [34] J. G. Marsh, F. J. Lerch, B. H. Putney, D. C. Christodoulidis, T. L. Felsentreger, B. V. Sanchez, D. E. Smith, S. M. Klosko, V. Martin, E. C. Pavlis, J. W. Robbins, R. G. Williamson, N. L. Chandler, K. E. Rachlin, G. B. Patel, S. Bhati, and D. S. Chinn, “An Improved Model of the Earth’s Gravitational Field: \*GEM-TI,”
- [35] R. S. Nerem, F. J. Lerch, J. A. Marshall, E. C. Pavlis, B. H. Putney, B. D. Tapley, R. J. Eanes, J. C. Ries, B. E. Schutz, C. K. Shum, M. M. Watkins, S. M. Klosko, J. C. Chan, S. B. Luthcke, G. B. Patel, N. K. Pavlis, R. G. Williamson, R. H. Rapp, R. Biancale, and F. Nouel, “Gravity model development for TOPEX/POSEIDON: Joint Gravity Models 1 and 2,” *Journal of Geophysical Research*, vol. 99, no. C12, p. 24421, 1994.
- [36] B. D. Tapley, M. M. Watkins, J. C. Ries, G. W. Davis, R. J. Eanes, S. R. Poole, H. J. Rim, B. E. Schutz, C. K. Shum, R. S. Nerem, F. J. Lerch, J. A. Marshall, S. M. Klosko, N. K. Pavlis, and R. G. Williamson, “The Joint Gravity Model 3,” *Journal of Geophysical Research: Solid Earth*, vol. 101, pp. 28029–28049, Dec. 1996.
- [37] F. G. Lemoine, S. C. Kenyon, J. K. Factor, R. G. Trimmer, N. K. Pavlis, D. S. Chinn, C. M. Cox, S. M. Klosko, S. B. Luthcke, M. H. Torrence, Y. M. Wang, R. G. Williamson, E. C. Pavlis, R. H. Rapp, and T. R. Olson, “The Development of the Joint NASA GSFC and the National Imagery and Mapping Agency (NIMA) Geopotential Model EGM96,” 1998.
- [38] N. K. Pavlis, S. A. Holmes, S. C. Kenyon, and J. K. Factor, “The development and evaluation of the Earth Gravitational Model 2008 (EGM2008): THE EGM2008 EARTH GRAVITATIONAL MODEL,” *Journal of Geophysical Research: Solid Earth*, vol. 117, pp. n/a–n/a, Apr. 2012.

- [39] “Mascons: Lunar Mass Concentrations,”
- [40] S. Nozette, P. Rustan, L. P. Pleasance, J. F. Kordas, I. T. Lewis, H. S. Park, R. E. Priest, D. M. Horan, P. Regeon, C. L. Lichtenberg, E. M. Shoemaker, E. M. Eliason, A. S. McEwen, M. S. Robinson, P. D. Spudis, C. H. Acton, B. J. Buratti, T. C. Duxbury, D. N. Baker, B. M. Jakosky, J. E. Blamont, M. P. Corson, J. H. Resnick, C. J. Rollins, M. E. Davies, P. G. Lucey, E. Malaret, M. A. Massie, C. M. Pieters, R. A. Reisse, D. E. Smith, T. C. Sorenson, R. W. V. Breugge, and M. T. Zuber, “The Clementine Mission to the Moon: Scientific Overview,” *Science*, vol. 266, pp. 1835–1839, Dec. 1994.
- [41] C. R. Tooley, M. B. Houghton, R. S. Saylor, C. Peddie, D. F. Everett, C. L. Baker, and K. N. Safdie, “Lunar Reconnaissance Orbiter Mission and Spacecraft Design,” *Space Science Reviews*, vol. 150, pp. 23–62, Jan. 2010.
- [42] M. T. Zuber, D. E. Smith, M. M. Watkins, S. W. Asmar, A. S. Konopliv, F. G. Lemoine, H. J. Melosh, G. A. Neumann, R. J. Phillips, S. C. Solomon, M. A. Wieczorek, J. G. Williams, S. J. Goossens, G. Kruizinga, E. Mazarico, R. S. Park, and D.-N. Yuan, “Gravity Field of the Moon from the Gravity Recovery and Interior Laboratory (GRAIL) Mission,” *Science*, vol. 339, pp. 668–671, Feb. 2013.
- [43] M. Brillouin, “équations aux dérivées partielles du 2e ordre. Domaines à connexion multiple. Fonctions sphériques non antipodes,” vol. 4, pp. 173–206, 1933.
- [44] D. J. Scheeres, A. S. French, P. Tricarico, S. R. Chesley, Y. Takahashi, D. Farnocchia, J. W. McMahon, D. N. Brack, A. B. Davis, R. L. Ballouz, E. R. Jawin, B. Rozitis, J. P. Emery, A. J. Ryan, R. S. Park, B. P. Rush, N. Mastrodemos, B. M. Kennedy, J. Bellerose, D. P. Lubey, D. Velez, A. T. Vaughan, J. M. Leonard, J. Geeraert, B. Page, P. Antreasian, E. Mazarico, K. Getzandanner, D. Rowlands, M. C. Moreau, J. Small, D. E. Highsmith, S. Goossens, E. E. Palmer, J. R. Weirich, R. W. Gaskell, O. S. Barnouin, M. G. Daly, J. A. Seabrook, M. M. Al Asad, L. C. Philpott, C. L. Johnson, C. M. Hartzell, V. E. Hamilton, P. Michel, K. J. Walsh, M. C. Nolan, and D. S. Lauretta, “Heterogeneous mass distribution of the rubble-pile asteroid (101955) Bennu,” *Science Advances*, vol. 6, no. 41, 2020.
- [45] A. Gao and W. Liao, “Efficient gravity field modeling method for small bodies based on Gaussian process regression,” *Acta Astronautica*, vol. 157, pp. 73–91, Apr. 2019.
- [46] L. Cheng, Z. Wang, Y. Song, and F. Jiang, “Real-time optimal control for irregular asteroid landings using deep neural networks,” *Acta Astronautica*, vol. 170, pp. 66–79, May 2020.
- [47] Ch. Reigber, H. Lühr, and P. Schwintzer, “CHAMP mission status,” *Advances in Space Research*, vol. 30, pp. 129–134, July 2002.
- [48] B. D. Tapley, S. Bettadpur, M. Watkins, and C. Reigber, “The gravity recovery and climate experiment: Mission overview and early results: GRACE MISSION OVERVIEW AND EARLY RESULTS,” *Geophysical Research Letters*, vol. 31, pp. n/a–n/a, May 2004.
- [49] F. Frappart and G. Ramillien, “Monitoring Groundwater Storage Changes Using the Gravity Recovery and Climate Experiment (GRACE) Satellite Mission: A Review,” *Remote Sensing*, vol. 10, p. 829, May 2018.

- [50] M. Van Der Meijde, R. Pail, R. Bingham, and R. Floberghagen, “GOCE data, models, and applications: A review,” *International Journal of Applied Earth Observation and Geoinformation*, vol. 35, pp. 4–15, Mar. 2015.
- [51] R. P. Kornfeld, B. W. Arnold, M. A. Gross, N. T. Dahya, W. M. Klipstein, P. F. Gath, and S. Bettadpur, “GRACE-FO: The Gravity Recovery and Climate Experiment Follow-On Mission,” *Journal of Spacecraft and Rockets*, vol. 56, pp. 931–951, May 2019.
- [52] M. T. Zuber, D. E. Smith, D. H. Lehman, T. L. Hoffman, S. W. Asmar, and M. M. Watkins, “Gravity Recovery and Interior Laboratory (GRAIL): Mapping the Lunar Interior from Crust to Core,” *Space Science Reviews*, vol. 178, pp. 3–24, Sept. 2013.
- [53] N. Pavlis, S. Holmes, S. Kenyon, D. Schmidt, and R. Trimmer, “A Preliminary Gravitational Model to Degree 2160,” in *Gravity, Geoid and Space Missions*, vol. 129, pp. 18–23, Berlin/Heidelberg: Springer-Verlag, 2005.
- [54] S. Goossens, F. Lemoine, T. Sabaka, J. Nicholas, E. Mazarico, D. Rowlands, B. Loomis, D. Chinn, G. Neumann, D. Smith, and M. Zuber, “A Global Degree and Order 1200 Model of the Lunar Gravity Field Using GRAIL Mission Data,” in *47th Annual Lunar and Planetary Science Conference*, Lunar and Planetary Science Conference, pp. 1484–1484, Mar. 2016.
- [55] A. Genova, S. Goossens, F. G. Lemoine, E. Mazarico, G. A. Neumann, D. E. Smith, and M. T. Zuber, “Seasonal and static gravity field of Mars from MGS, Mars Odyssey and MRO radio science,” *Icarus*, vol. 272, pp. 228–245, July 2016.
- [56] G. Gravity, *The Earth’s Gravitational Field 2.1*. No. 1, 2016.
- [57] E. Hewitt and R. E. Hewitt, “The Gibbs-Wilbraham phenomenon: An episode in fourier analysis,” *Archive for History of Exact Sciences*, vol. 21, no. 2, pp. 129–160, 1979.
- [58] J. R. Martin and H. Schaub, “GPGPU Implementation of Pines’ Spherical Harmonic Gravity Model,” in *AAS/AIAA Astrodynamics Specialist Conference* (R. S. Wilson, J. Shan, K. C. Howell, and F. R. Hoots, eds.), (Virtual Event), Univelt Inc., 2020.
- [59] Y. Takahashi, D. J. Scheeres, and R. A. Werner, “Surface gravity fields for asteroids and comets,” *Journal of Guidance, Control, and Dynamics*, vol. 36, no. 2, pp. 362–374, 2013.
- [60] Y. Takahashi and D. J. Scheeres, “Small body surface gravity fields via spherical harmonic expansions,” *Celestial Mechanics and Dynamical Astronomy*, vol. 119, no. 2, pp. 169–206, 2014.
- [61] A. P. M. Muller and W. L. Sjogren, “Mascons : Lunar Mass Concentrations,” vol. 161, no. 3842, pp. 680–684, 1968.
- [62] P. T. Wittick and R. P. Russell, “Mixed-model gravity representations for small celestial bodies using mascons and spherical harmonics,” *Celestial Mechanics and Dynamical Astronomy*, vol. 131, pp. 31–31, July 2019.
- [63] P. G. Antreasian, S. R. Chesley, J. K. Miller, J. J. Bordi, and B. G. Williams, “THE DESIGN AND NAVIGATION OF THE NEAR SHOEMAKER LANDING ON EROS,”

- [64] T. G. G. Chanut, O. C. Winter, and M. Tsuchida, “3D stability orbits close to 433 Eros using an effective polyhedral model method,” *Monthly Notices of the Royal Astronomical Society*, vol. 438, pp. 2672–2682, Mar. 2014.
- [65] J. Miller, A. Konopliv, P. Antreasian, J. Bordi, S. Chesley, C. Helfrich, W. Owen, T. Wang, B. Williams, D. Yeomans, and D. Scheeres, “Determination of Shape, Gravity, and Rotational State of Asteroid 433 Eros,” *Icarus*, vol. 155, pp. 3–17, Jan. 2002.
- [66] D. Scheeres, R. Gaskell, S. Abe, O. Barnouin-Jha, T. Hashimoto, J. Kawaguchi, T. Kubota, J. Saito, M. Yoshikawa, N. Hirata, T. Mukai, M. Ishiguro, T. Kominato, K. Shirakawa, and M. Uo, “The Actual Dynamical Environment About Itokawa,” in *AIAA/AAS Astrodynamics Specialist Conference and Exhibit*, (Keystone, Colorado), American Institute of Aeronautics and Astronautics, Aug. 2006.
- [67] B. Bercovici, P. Panicucci, and J. McMahon, “Analytical shape uncertainties in the polyhedron gravity model,” *Celestial Mechanics and Dynamical Astronomy*, vol. 132, p. 29, May 2020.
- [68] L. . A. Cangahuala, “Augmentations to the Polyhedral Gravity Model to Facilitate Small Body Navigation,”
- [69] D. J. Scheeres, B. Khushalani, and R. A. Werner, “Estimating asteroid density distributions from shape and gravity information,” *Planetary and Space Science*, vol. 48, no. 10, pp. 965–971, 2000.
- [70] M. T. Zuber, D. E. Smith, A. F. Cheng, J. B. Garvin, O. Aharonson, T. D. Cole, P. J. Dunn, Y. Guo, F. G. Lemoine, G. A. Neumann, D. D. Rowlands, and M. H. Torrence, “The shape of 433 Eros from the NEAR-Shoemaker Laser Rangefinder,” *Science*, vol. 289, no. 5487, pp. 2097–2101, 2000.
- [71] M. K. Shepard, B. Timerson, D. J. Scheeres, L. A. Benner, J. D. Giorgini, E. S. Howell, C. Magri, M. C. Nolan, A. Springmann, P. A. Taylor, and A. Virkki, “A revised shape model of asteroid (216) Kleopatra,” *Icarus*, vol. 311, pp. 197–209, 2018.
- [72] J. K. Miller, A. S. Konopliv, P. G. Antreasian, J. J. Bordi, S. Chesley, C. E. Helfrich, W. M. Owen, T. C. Wang, B. G. Williams, D. K. Yeomans, and D. J. Scheeres, “Determination of shape, gravity, and rotational state of asteroid 433 Eros,” *Icarus*, vol. 155, no. 1, pp. 3–17, 2002.
- [73] Y. Takahashi and D. Scheeres, “Morphology driven density distribution estimation for small bodies,” *Icarus*, vol. 233, pp. 179–193, May 2014.
- [74] D. P. Kingma and J. Ba, “Adam: A Method for Stochastic Optimization,” *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, pp. 1–15, Dec. 2014.
- [75] L. Bottou, *Stochastic Gradient Descent Tricks*, vol. 7700. Springer, Jan. 2012.
- [76] Y. Lecun, Y. Bengio, and G. Hinton, “Deep learning,” *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

- [77] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, “Extreme learning machine: Theory and applications,” *Neurocomputing*, vol. 70, pp. 489–501, Dec. 2006.
- [78] Rui Zhang, Yuan Lan, Guang-Bin Huang, and Zong-Ben Xu, “Universal Approximation of Extreme Learning Machine With Adaptive Growth of Hidden Nodes,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 23, pp. 365–371, Feb. 2012.
- [79] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, “NeRF: Representing scenes as neural radiance fields for view synthesis,” *Communications of the ACM*, vol. 65, pp. 99–106, Jan. 2022.
- [80] P. Mertikopoulos, C. Papadimitriou, and G. Piliouras, *Cycles in Adversarial Regularized Learning*. Philadelphia, PA: Society for Industrial and Applied Mathematics, Jan. 2018.
- [81] S. Wang, Y. Teng, and P. Perdikaris, “Understanding and mitigating gradient pathologies in physics-informed neural networks,” *arXiv*, pp. 1–28, Jan. 2020.
- [82] S. Ruder, “An overview of gradient descent optimization algorithms,” *arXiv*, pp. 1–14, Sept. 2016.
- [83] X. Glorot and Y. Bengio, “Understanding the difficulty of training deep feedforward neural networks,” *Journal of Machine Learning Research*, vol. 9, pp. 249–256, 2010.
- [84] S. Pines, “Uniform Representation of the Gravitational Potential and its Derivatives,” *AIAA Journal*, vol. 11, pp. 1508–1511, Nov. 1973.
- [85] R. Swinbank and R. James Purser, “Fibonacci grids: A novel approach to global modelling,” *Quarterly Journal of the Royal Meteorological Society*, vol. 132, pp. 1769–1793, July 2006.
- [86] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.
- [87] J. Martin and H. Schaub, “Physics-informed neural networks for gravity field modeling of the Earth and Moon,” *Celestial Mechanics and Dynamical Astronomy*, vol. 134, Apr. 2022.
- [88] T. Jayalakshmi and A. Santhakumaran, “Statistical Normalization and Back Propagationfor Classification,” *International Journal of Computer Theory and Engineering*, vol. 3, no. 1, pp. 89–93, 2011.
- [89] S. R. Chesley, A. S. French, A. B. Davis, R. A. Jacobson, M. Brozović, D. Farnocchia, S. Selznick, A. J. Liounis, C. W. Hergenrother, M. C. Moreau, J. Pelgrift, E. Lessac-Chenen, J. L. Molaro, R. S. Park, B. Rozitis, D. J. Scheeres, Y. Takahashi, D. Vokrouhlický, C. W. Wolner, C. Adam, B. J. Bos, E. J. Christensen, J. P. Emery, J. M. Leonard, J. W. McMahon, M. C. Nolan, F. C. Shelly, and D. S. Lauretta, “Trajectory Estimation for Particles Observed in the Vicinity of (101955) Bennu,” *Journal of Geophysical Research: Planets*, vol. 125, no. 9, 2020.
- [90] J. W. McMahon, D. J. Scheeres, S. R. Chesley, A. French, D. Brack, D. Farnocchia, Y. Takahashi, B. Rozitis, P. Tricarico, E. Mazarico, B. Bierhaus, J. P. Emery, C. W. Hergenrother, and D. S. Lauretta, “Dynamical Evolution of Simulated Particles Ejected From Asteroid Bennu,” *Journal of Geophysical Research: Planets*, vol. 125, no. 8, pp. 1–18, 2020.

- [91] J. Villa, A. French, J. Mcmahon, D. Scheeres, and B. Hockman, “Gravity Estimation of Small Bodies via Optical Tracking of Hopping Artificial Probes,” in AAS/AIAA Astrodynamics Specialist Conference, (Big Sky, Montana), pp. 1–21, 2021.
- [92] L. Cheng, Z. Wang, and F. Jiang, “Real-time control for fuel-optimal Moon landing based on an interactive deep reinforcement learning algorithm,” Astrodynamic, vol. 3, pp. 375–386, Dec. 2019.
- [93] Q. Zhu, Z. Liu, and J. Yan, “Machine learning for metal additive manufacturing: Predicting temperature and melt pool fluid dynamics using physics-informed neural networks,” Sept. 2020.
- [94] J. R. Martin and H. Schaub, “Preliminary Analysis of Small-Body Gravity Field Estimation using Physics-Informed Neural Networks and Kalman Filters,” th International Astronautical Congress, p. 10, 2022.
- [95] N. K. Pavlis, S. A. Holmes, S. C. Kenyon, and J. K. Factor, “The development and evaluation of the Earth Gravitational Model 2008 (EGM2008),” Journal of Geophysical Research: Solid Earth, vol. 117, pp. n/a–n/a, Apr. 2012.
- [96] M. KANAMARU and S. SASAKI, “Estimation of Interior Density Distribution for Small Bodies: The Case of Asteroid Itokawa,” Transactions of the Japan Society for Aeronautical and Space Sciences, Aerospace Technology Japan, vol. 17, no. 3, pp. 270–275, 2019.
- [97] S. Aradi, “Survey of Deep Reinforcement Learning for Motion Planning of Autonomous Vehicles,” IEEE Transactions on Intelligent Transportation Systems, vol. 23, pp. 740–759, Feb. 2022.
- [98] M. M. Afsar, T. Crump, and B. Far, “Reinforcement learning based recommender systems: A survey,” June 2022.
- [99] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, T. Lillicrap, K. Simonyan, and D. Hassabis, “A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play,” Science, vol. 362, pp. 1140–1144, Dec. 2018.
- [100] G. Carleo, I. Cirac, K. Cranmer, L. Daudet, M. Schuld, N. Tishby, L. Vogt-Maranto, and L. Zdeborová, “Machine learning and the physical sciences,” Reviews of Modern Physics, vol. 91, p. 045002, Dec. 2019.
- [101] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, “A Brief Survey of Deep Reinforcement Learning,” IEEE Signal Processing Magazine, vol. 34, pp. 26–38, Nov. 2017.
- [102] J. Broida and R. Linares, “SPACECRAFT RENDEZVOUS GUIDANCE IN CLUTTERED ENVIRONMENTS VIA REINFORCEMENT LEARNING,”
- [103] K. Hovell and S. Ulrich, “Deep Reinforcement Learning for Spacecraft Proximity Operations Guidance,” Journal of Spacecraft and Rockets, vol. 58, pp. 254–264, Mar. 2021.

- [104] C. J. Sullivan and N. Bosanac, "USING MULTI-OBJECTIVE DEEP REINFORCEMENT LEARNING TO UNCOVER A PARETO FRONT IN MULTI-BODY TRAJECTORY DESIGN,"
- [105] N. B. LaFarge, D. Miller, K. C. Howell, and R. Linares, "Guidance for Closed-Loop Transfers using Reinforcement Learning with Application to Libration Point Orbits," in AIAA Scitech 2020 Forum, (Orlando, FL), American Institute of Aeronautics and Astronautics, Jan. 2020.
- [106] A. Harris, T. Valade, T. Teil, and H. Schaub, "Generation of Spacecraft Operations Procedures Using Deep Reinforcement Learning," Journal of Spacecraft and Rockets, vol. 59, pp. 611–626, Mar. 2022.
- [107] A. Herrmann and H. Schaub, "A COMPARISON OF DEEP REINFORCEMENT LEARNING ALGORITHMS FOR EARTH-OBSERVING SATELLITE SCHEDULING,"
- [108] W. Zhao, J. P. Queralta, and T. Westerlund, "Sim-to-Real Transfer in Deep Reinforcement Learning for Robotics: A Survey," in 2020 IEEE Symposium Series on Computational Intelligence (SSCI), pp. 737–744, Dec. 2020.
- [109] T. Imken, T. Randolph, M. DiNicola, and A. Nicholas, "Modeling spacecraft safe mode events," IEEE Aerospace Conference Proceedings, vol. 2018-March, pp. 1–13, 2018.
- [110] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," 35th International Conference on Machine Learning, ICML 2018, vol. 5, pp. 2976–2989, 2018.
- [111] D. J. Scheeres, Orbital Motion in Strongly Perturbed Environments. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012.
- [112] Y. Yu and H. Baoyin, "Generating families of 3D periodic orbits about asteroids: Generating 3D periodic orbits about asteroids," Monthly Notices of the Royal Astronomical Society, vol. 427, pp. 872–881, Nov. 2012.
- [113] A. G. Baydin, B. A. Pearlmutter, and J. M. Siskind, "Automatic Differentiation in Machine Learning: A Survey," Journal of Machine Learning Research, vol. 18, pp. 1–43, 2018.
- [114] D. J. Scheeres, "Analysis of Orbital Motion around 433 Eros," The Journal of the Astronautical Sciences, vol. 43, no. 4, pp. 427–452, 1995.
- [115] D. J. Scheeres, S. J. Ostro, R. S. Hudson, E. M. DeJong, and S. Suzuki, "Dynamics of Orbits Close to Asteroid 4179 Toutatis," Icarus, vol. 132, no. 1, pp. 53–79, 1998.
- [116] K. C. Howell, "Three-Dimensional, Periodic, 'Halo' Orbits," Celestial Mechanics and Dynamical Astronomy, pp. 53–71, 1984.
- [117] E. J. Doedel, R. C. Paffenroth, H. B. Keller, D. J. Dichmann, J. Galán-Vioque, and A. Vanderbauwhede, "Computation of periodic solutions of conservative systems with application to the 3-body problem," International Journal of Bifurcation and Chaos in Applied Sciences and Engineering, vol. 13, no. 6, pp. 1353–1381, 2003.
- [118] A. Abad, A. Elipe, and E. Tresaco, "Analytical model to find frozen orbits for a lunar orbiter," Journal of Guidance, Control, and Dynamics, vol. 32, no. 3, pp. 888–898, 2009.

- [119] Y. Lan and P. Cvitanović, “Variational method for finding periodic orbits in a general flow,” 2003.
- [120] D. J. Scheeres, “Orbit mechanics about asteroids and comets,” *Journal of Guidance, Control, and Dynamics*, vol. 35, no. 3, pp. 987–997, 2012.
- [121] N. Baresi, *Spacecraft Formation Flight on Quasi-periodic Invariant Tori*. PhD thesis, University of Colorado Boulder, 2017.
- [122] M. A. Branch, T. F. Coleman, and Y. Li, “A Subspace, Interior, and Conjugate Gradient Method for Large-Scale Bound-Constrained Minimization Problems,” *SIAM Journal on Scientific Computing*, vol. 21, pp. 1–23, Jan. 1999.
- [123] S. ichiro Watanabe, Y. Tsuda, M. Yoshikawa, S. Tanaka, T. Saiki, and S. Nakazawa, “Hayabusa2 Mission Overview,” *Space Science Reviews*, vol. 208, no. 1-4, pp. 3–16, 2017.
- [124] D. S. Lauretta, S. S. Balram-Knutson, E. Beshore, W. V. Boynton, C. Drouet d’Aubigny, D. N. DellaGiustina, H. L. Enos, D. R. Golish, C. W. Hergenrother, E. S. Howell, C. A. Bennett, E. T. Morton, M. C. Nolan, B. Rizk, H. L. Roper, A. E. Bartels, B. J. Bos, J. P. Dworkin, D. E. Highsmith, D. A. Lorenz, L. F. Lim, R. Mink, M. C. Moreau, J. A. Nuth, D. C. Reuter, A. A. Simon, E. B. Bierhaus, B. H. Bryan, R. Ballouz, O. S. Barnouin, R. P. Binzel, W. F. Bottke, V. E. Hamilton, K. J. Walsh, S. R. Chesley, P. R. Christensen, B. E. Clark, H. C. Connolly, M. K. Crombie, M. G. Daly, J. P. Emery, T. J. McCoy, J. W. McMahon, D. J. Scheeres, S. Messenger, K. Nakamura-Messenger, K. Righter, and S. A. Sandford, “OSIRIS-REx: Sample Return from Asteroid (101955) Bennu,” *Space Science Reviews*, vol. 212, no. 1-2, pp. 925–984, 2017.
- [125] C. T. Russell, F. Capaccioni, A. Coradini, M. C. De Sanctis, W. C. Feldman, R. Jaumann, H. U. Keller, T. B. McCord, L. A. McFadden, S. Mottola, C. M. Pieters, T. H. Prettyman, C. A. Raymond, M. V. Sykes, D. E. Smith, and M. T. Zuber, “Dawn mission to vesta and ceres,” *Earth, Moon and Planets*, vol. 101, no. 1-2, pp. 65–91, 2007.
- [126] D. Y. Oh, S. Collins, T. Drain, W. Hart, T. Imken, K. Larson, D. Marsh, D. Muthulingam, J. S. Snyder, D. Trofimov, L. T. Elkins-Tanton, I. Johnson, P. Lord, and Z. Pirlk, “Development of the Psyche Mission for NASA’s Discovery Program,” in *36th International Electric Propulsion Conference*, (Vienna, Austria), 2019.
- [127] D. J. Scheeres, J. W. McMahon, E. B. Bierhaus, J. Wood, L. A. Benner, C. Hartzell, P. Hayne, J. Hopkins, R. Jedicke, L. LeCorre, S. Naidu, P. Pravec, and M. Ravine, “Janus: A NASA SIMPLEx mission to explore two NEO binary asteroids,” *Proceedings of the International Astronautical Congress, IAC*, vol. 2020-Octob, 2020.
- [128] A. S. French, *Precise Orbit Determination And Gravity Field Estimation During Small Body Missions*. PhD thesis, University of Colorado Boulder, 2020.
- [129] J. M. Leonard, F. G. Nievinski, and G. H. Born, “Gravity Error Compensation Using Second-Order Gauss-Markov Processes,” *Journal of Spacecraft and Rockets*, vol. 50, pp. 217–229, Jan. 2013.
- [130] P. Xu and R. Rummel, “Generalized ridge regression with applications in determination of geopotential fields,” *manuscripta geodaetica*, pp. 8–20, 1994.

- [131] P. Xu, “The value of minimum norm estimation of geopotential fields,” *Geophysical Journal International*, vol. 111, pp. 170–178, Oct. 1992.
- [132] P. Xu, “Truncated SVD methods for discrete linear ill-posed problems,” *Geophysical Journal International*, vol. 135, pp. 505–514, Nov. 1998.
- [133] P. Xu, Y. Fukuda, and Y. Liu, “Multiple parameter regularization: Numerical solutions and applications to the determination of geopotential from precise satellite orbits,” *Journal of Geodesy*, vol. 80, pp. 17–27, Apr. 2006.
- [134] G. Welch, “An Introduction to the Kalman Filter,” tech. rep., 1997.
- [135] B. Schutz, B. Tapley, and G. H. Born, *Statistical Orbit Determination*. Elsevier, 2004.
- [136] D. Hendrycks and K. Gimpel, “Gaussian Error Linear Units (GELUs),” July 2020.
- [137] S. Wang, X. Yu, and P. Perdikaris, “When and why PINNs fail to train: A neural tangent kernel perspective,” July 2020.
- [138] N. Rahaman, A. Baratin, D. Arpit, F. Draxler, M. Lin, F. A. Hamprecht, Y. Bengio, and A. Courville, “On the Spectral Bias of Neural Networks,”
- [139] M. Tancik, P. P. Srinivasan, B. Mildenhall, S. Fridovich-Keil, N. Raghavan, U. Singhal, R. Ramamoorthi, J. T. Barron, and R. Ng, “Fourier Features Let Networks Learn High Frequency Functions in Low Dimensional Domains,”
- [140] S. Wang, H. Wang, and P. Perdikaris, “On the eigenvector bias of Fourier feature networks: From regression to solving multi-scale PDEs with physics-informed neural networks,” *Computer Methods in Applied Mechanics and Engineering*, vol. 384, p. 113938, Oct. 2021.
- [141] V. Sitzmann, J. N. P. Martel, A. W. Bergman, D. B. Lindell, and G. Wetzstein, “Implicit Neural Representations with Periodic Activation Functions,” June 2020.
- [142] B. A. Jones, G. H. Born, and G. Beylkin, “AAS 10-237 ORBIT DETERMINATION WITH THE CUBED-SPHERE GRAVITY MODEL,” no. 1, pp. 1–20.
- [143] F. G. Lemoine, S. Goossens, T. J. Sabaka, J. B. Nicholas, E. Mazarico, D. D. Rowlands, B. D. Loomis, D. S. Chinn, G. A. Neumann, D. E. Smith, and M. T. Zuber, “GRGM900C: A degree 900 lunar gravity model from GRAIL primary and extended mission data,” *Geophysical Research Letters*, vol. 41, no. 10, pp. 3382–3389, 2014.
- [144] V. Volkov, “Understanding Latency Hiding on GPUs — EECS at UC Berkeley,” tech. rep., Berekely, CA, 2016.
- [145] P. W. Kenneally, *Faster than Real-Time GPGPU Radiation Pressure Modeling Methods*. PhD thesis, 2019.
- [146] W. Kefan and L. Ge, “The gravity parallel computation based on GPU,” *2017 3rd IEEE International Conference on Computer and Communications, ICCC 2017*, vol. 2018-Janua, pp. 2409–2413, 2018.
- [147] I. O. Hupca, J. Falcou, L. Grigori, and R. Stompor, “Spherical harmonic transform with GPUs,” vol. 7155 LNCS, no. PART 1, pp. 355–366, 2012.

- [148] A. Atallah and A. Bani Younes, “Parallel Chebyshev Picard Method,” in AIAA SciTech Forum, no. January, (Orlando, FL), AIAA, 2020.
- [149] J. B. Lundberg and B. E. Schutz, “Recursion formulas of Legendre functions for use with nonsingular geopotential models,” Journal of Guidance, Control, and Dynamics, vol. 11, no. 1, pp. 31–38, 1988.
- [150] J. Kessenich, D. Baldwin, and R. Rost, “The OpenGL Shading Language Version 4.60.7,” tech. rep., 2019.
- [151] M. Mantor and M. Houston, “AMD Graphics Core Next,” AMD Fusion Developer Summit, 2011.
- [152] NVIDIA, “CUDA C Best Practices Guide (v11.0),” Tech. Rep. July, 2020.
- [153] S. Jones, “Cuda Optimization Tips, Tricks and Techniques,” in GPU Technology Conference, (Sillicon Valley), NVIDIA, 2017.
- [154] A. Ilic, F. Pratas, and L. Sousa, “Cache-aware roofline model: Upgrading the loft,” IEEE Computer Architecture Letters, vol. 13, no. 1, pp. 21–24, 2014.

## Appendix A

### PINN-GM-III Training Details

This section aims to highlight the various training and implementation details of the PINN gravity model. Like the PINN-GM-II, the PINN-GM-III is built around a core network which shares similarities to a transformer architecture. The network itself is composed of a feed-forward multilayer perception, preceded by an embedding layer. Skip connections are attached between the embedding layer and each of the hidden layers, and the final layer uses linear activation functions to produce the network’s prediction of the proxy potential.

Unlike the PINN-GM-II, all of the experiments tested in this work do not make use of the multi-constraint loss function. Specifically, in the PINN-GM-II implementation, the loss function included penalties for  $\nabla^2 U = 0$  and  $\nabla \times \nabla U = 0$  in addition to  $-\nabla U - \mathbf{a} = 0$ . While the multi-objective loss function offers value when training networks with noisy data, it is not necessary for the experiments in this work as the data is noise-free. Appendix B discusses the multi-objective loss function and this choice in more detail.

The default hyperparameters used to train PINN-GM-III are listed in Table A.1. The network is trained using the Adam optimizer with a learning rate of 0.001. The learning rate is decayed when the validation loss plateaus for 2,500 epochs. The default batch size is set to  $2^{20}$  although many of the training data sizes are less than this value, so typically the batch size is automatically reduced to the size of the training dataset. The networks are trained for 10,000 epochs unless otherwise specified as in Section 3.4. The network is initialized using the Xavier uniform initialization scheme (83), and the network activation functions are GELU (136). Note that the

final layers weights are initialized to zero, which heuristically leads to faster convergence and better performance.

Table A.1: Default Hyperparameters for PINN-GM-III

Hyperparameter	Value	Hyperparameter	Value
Batch Size	$2^{20}$	Activation	GeLU
Learning Rate	0.001	Network Architecture	Transformer Inspired
Optimizer	Adam	Non-Dimension.	Section 3.3.1
LR Scheduler	On Plateau	Preprocessing Layers	Pines + $\frac{1}{r}$
Patience	2500 Epochs	Hidden Layer $\theta$ Init.	Glorot Normal
Loss Function	Percent	Final Layer $\theta$ Init.	Zeros

## Appendix B

### Learning Rate Annealing Algorithm

Reference (1) proposed the use of a multi-objective loss function for the PINN-GM which leverages knowledge of three constraints: (1)  $-\nabla U = \mathbf{a}$ , (2)  $\nabla^2 U = 0$ , and (3)  $\nabla \times \nabla U = \mathbf{0}$  through:

$$L(\theta) = \frac{1}{N} \sum_{i=0}^N \left| -\nabla \hat{U}(x_i|\theta) - \mathbf{a} \right|^2 + \left| \nabla^2 \hat{U}(x_i|\theta) \right|^2 + \left| \nabla \times \nabla \hat{U}(x_i|\theta) \right|^2 \quad (\text{B.1})$$

The inclusion of constraints (2) and (3) into Equation (B.1) benefit network performance in the presence of noisy training data. While this additional robustness is typically welcomed when training on imperfect datasets, these constraints also present new challenges for network training. Foremost, these constraints introduce a trade between model robustness and accuracy. Past work has shown that these constraints can act as a form of regularization and can inadvertently degrade performance of PINN-GMs when trained on perfect position and acceleration measurements (1). These constraints also increase total training time as the second order derivatives of the potential are particularly expensive to compute using automatic differentiation. Finally, the multi-objective nature of the cost function can lead to competition among terms in Equation (B.1), where one constraint can dominate the learning process at the expense of the others.

PINN-GM-III proposes multiple modifications to address these challenges. To alleviate some of the computational cost, PINN-GM-III only leverages the multi-objective cost function for part of its training cycle. PINN-GM-III begins by only using the  $-\nabla \hat{U}(x_i|\theta) - \mathbf{a} = 0$  constraint. This allows the network to identify a candidate solution without the large computational cost of the  $\nabla^2 U$  or  $\nabla \times \nabla U$  operations. Once a candidate solution is identified, the loss function changes to

incorporate the Laplacian  $\nabla^2 U = 0$  constraint and continues training<sup>1</sup>.

PINN-GM-III also leverages an adaptive learning rate annealing algorithm to change the weights of the loss components during training to minimize competing objectives. The algorithm takes inspiration from a theoretical analysis of the learning dynamics of neural networks using Neural Tangent Kernels to highlight how different components in a loss function can have competing gradient flow dynamics. Further details of this effect and the annealing rate algorithm can be found in Reference (137).

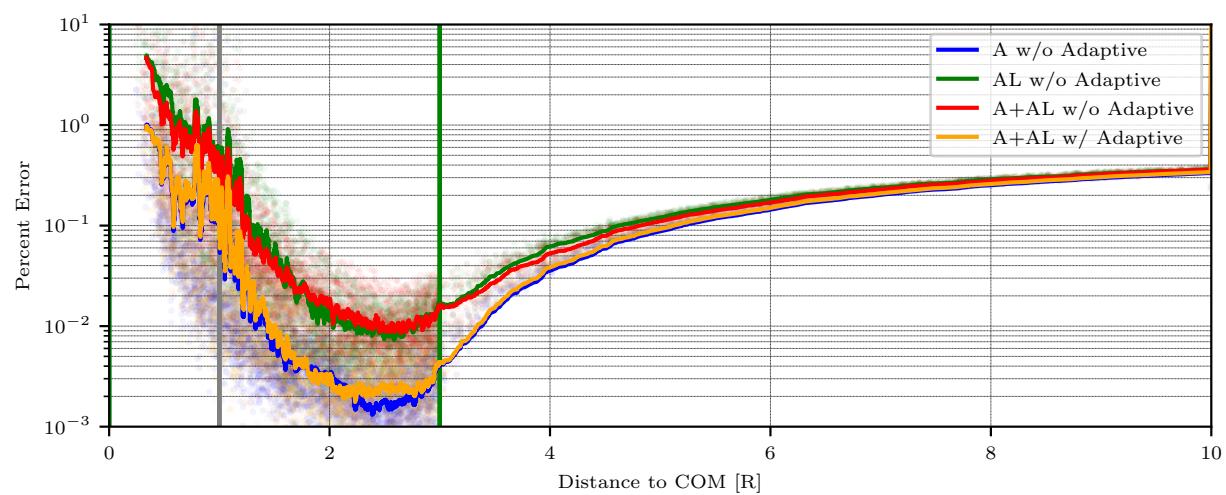
To measure the effect of these changes, four experiments are run. First, a PINN-GM-III is trained with only the  $-\nabla U = \mathbf{a}$  constraint for a total of 20,000 epochs. Second, a PINN-GM-III is trained with both the  $-\nabla U = \mathbf{a}$  and  $\nabla^2 U = 0$  constraint for 20,000 epochs without the learning rate annealing algorithm. Third, a PINN-GM is trained for 10,000 epochs using only  $-\nabla U = \mathbf{a}$  followed by 10,000 epochs with both A and L constraints without the learning rate annealing. Finally, model is trained with 10,000 epochs using constraint A followed by 10,000 epochs with constraints A and L with the learning rate annealing algorithm. These experiments are labeled as “A w/o Adaptive”, “AL w/o Adaptive”, “A+AL w/o Adaptive”, and “A+AL w/ Adaptive” respectively.

These four experiments are run using 45,000 data randomly distributed between 0 - 3R about the asteroid Eros. The model error is assessed on a separate test set of 50,000 data points distributed from 0-10R. The experiment is performed once with no noise in the training data (Figure B.1a), and again with noise applied — 10% relative error added in a random direction to each acceleration vector (Figure B.1b).

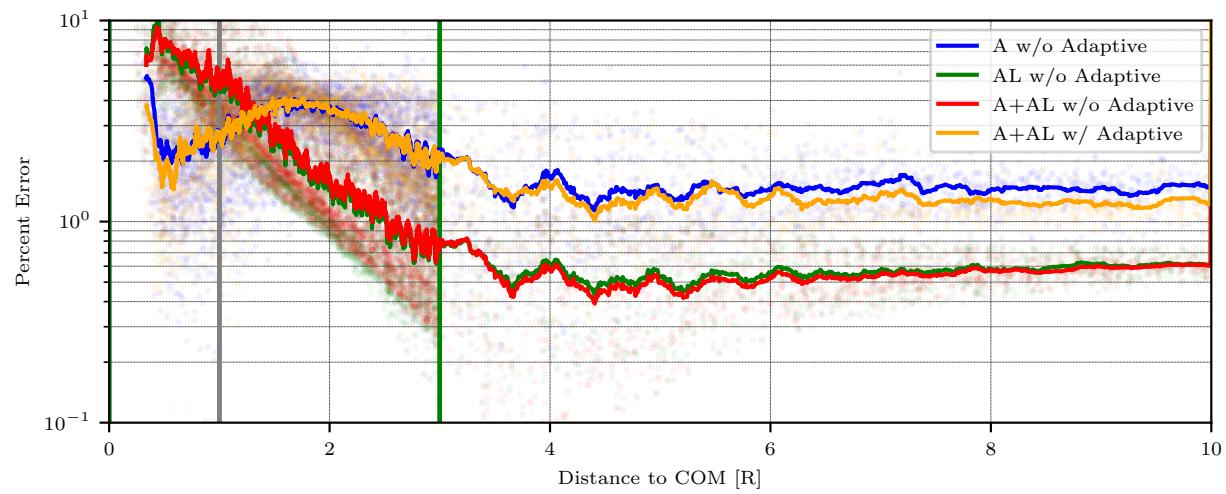
When noise is **not** added to the training data, Figure B.1a showcases three noteworthy behaviors. First, there is no practical drawbacks to pre-training a network with the A constraint, and later adding additional constraints to the loss function. The green and the red error averages are nearly identical, which demonstrates that the training time can be considerably reduced by

---

<sup>1</sup> Note that the  $\nabla \times \nabla U = 0$  constraint is no longer included within PINN-GM-III because the expression often evaluates to numbers that near machine precision making its contribution negligible to the loss.



(a) No error in training data



(b) 10% error in training data

Figure B.1: Model % error as a function of altitude using different combinations of pre-training and adaptive learning rate annealing algorithms.

initially bypassing the computationally expense of  $\nabla^2 U$  operation during the beginning of training. Second, as is consistent with past results, the introduction of the L constraint into the loss function **without** the adaptive learning rate algorithm hurts model accuracy. Third and finally, when the adaptive learning rate algorithm is incorporated, the performance gap between the A and AL constraints disappears.

Despite the apparent benefit of the adaptive learning rate algorithm, when noise **is added** to the training data, Figure B.1b demonstrates a less favorable picture. The AL loss function **without** the adaptive algorithm showcases better performance. Consistent with past results, the AL constraint without the adaptive algorithm is more robust to the noisy data than its isolated A constraint counterpart. However, when the adaptive learning rate algorithm is introduced, that robustness is lost and the model error closely tracks that of the singular A constraint. This suggests that the learning rate annealing algorithm, in this setting, appears to consistently favor or up-weight the A constraint over the L constraint.

Taken together, these adaptive learning rate annealing algorithm is an interesting method to circumvent competing objectives in the loss function; however, this demonstrates it can have inadvertent drawbacks. In the presence of noiseless, perfect training data, the inclusion of this algorithm offers no immediate disadvantages, and does improve model performance when using a multi-objective loss function. However, in the presence of noisy data, it is advised to default to the non-adapting algorithm.

## Appendix C

### Avoiding Spectral Bias with Fourier Feature Mapping

A well-documented challenge neural networks face is an effect called spectral bias (138). The gradient flow dynamics of traditional and physics-informed neural networks are known to preferentially learn low frequencies before high frequencies during training. This bias towards lower frequencies can lead to excessively slow convergence rates when attempting to model high frequencies. This problem cannot be ignored when constructing PINN-GMs, particularly when considering the discontinuous mountain ranges on the Earth, the craters on the surface of the Moon, or the boulders on the surface of asteroids. The locality and scale of these features manifest as high frequencies and can require long training periods to be sufficiently regressed.

Multiple research efforts aim to address this bias. One effort proposes the introduction of a fourier feature mapping layer to aid in making the neural tangent kernel stationary (139). By randomly projecting the inputs of a neural network into fourier space, it is shown that the spectral bias can be mitigated if the inputs are projected into the correct range of frequencies. Another effort extends these findings to demonstrate how multi-scale problems in scientific machine learning may require multiple projections into fourier space to efficiently learn both high and low frequency functions (140). Separately, the introduction of Sinusoidal Representation Networks (SIRENs) proposes exclusively leveraging sine functions for the network activations along with a principled initialization scheme to construct more compelling implicit neural representations of complex signals (141).

For PINN-GM-III, a fourier mapping layer is proposed inspired by Reference (139). The

fourier mapping layer is implemented immediately after the Pine's projection layer. The values of  $s, t$ , and  $u$  are first scaled to exist in  $[0, 1]$  and then run through the following mapping:

$$\hat{x}_i = \left[ \sin\left(2\pi B_{i,j}^{(x)} x_j + \phi_j^{(x)}\right), \cos\left(2\pi B_{i,j}^{(x)} x_j + \phi_j^{(x)}\right) \right] \forall i = 1, \dots, N \quad (\text{C.1})$$

where  $x \in \mathbb{R}^{1 \times N} \forall x \in [s, t, u]$  is the feature to be projected into fourier space,  $B^{(x)} \sim \mathcal{N}(0, \sigma_f^2) \in \mathbb{R}^{N_f \times 1}$  is the set of random frequencies to projected onto, and  $\phi^{(x)} \sim \mathcal{N}(0, 1) \in \mathbb{R}^{N_f}$  are the phase offsets of the projection.

There exist many possible initialization schemes for this mapping. The user is ultimately responsible for deciding how many frequencies to project onto,  $N_f$ , the variance of the frequency distribution,  $\sigma_f$ , if the frequencies and offsets should be trainable variables or fixed, and if each feature should have their own frequencies and offsets or if they should be shared for each layer (i.e.  $B^{(s)} = B^{(t)} = B^{(u)}$ ). An experiment is run which investigates the increase in model size versus model performance across these design choices and reported in Table C.1. Each network was trained on the asteroid Eros using the default hyperparameters, and a  $\sigma = 1$  is used. The baseline performance was that of a PINN-GM-III with no Fourier layer and a model capacity of 40 nodes per layer.

<b>Fourier Layer Without Trainable <math>\sigma</math> and <math>\phi</math></b>				
Parameters	$N_{FF}$	Avg. % Error	$\Delta$ Model Size [%]	$\Delta$ Avg. % Error
13923	5	0.191	14.84	-24.11
15723	10	0.206	29.69	-18.08
19323	20	0.192	59.39	-23.92
<b>Fourier Layer With Trainable <math>\sigma</math> and <math>\phi</math></b>				
Parameters	$N_{FF}$	Avg. % Error	$\Delta$ Model Size [%]	$\Delta$ Avg. % Error
13953	5	0.185	15.09	-26.60
15783	10	0.200	30.19	-20.60
19443	20	0.191	60.38	-24.25
<b>Shared Fourier Layer With Trainable <math>\sigma</math> and <math>\phi</math></b>				
Parameters	$N_{FF}$	Avg. % Error	$\Delta$ Model Size [%]	$\Delta$ Avg. % Error
13933	5	0.185	14.93	-26.40
15743	10	0.208	29.86	-17.61
19363	20	0.195	59.72	-22.42

Table C.1: Hyperparameter search for Fourier feature mapping.

At best, the PINN-GM-III with a learned fourier mapping reduced average error about the asteroid 433-Eros from 0-3R by 24% as compared to a base PINN-GM-III without this mapping. This learned layer does increase the model size (number of parameters in the network) by 14.8%. Thus, it should be noted that there is an implicit trade introduced between model size and accuracy through the introduction of these features. Because spacecraft computing resources are typically quite limited, further work must be conducted to assess if this trade is consistently worth the cost.

## Appendix D

### Comments on Past Machine Learning Performance

Table 3.4 highlights the general performance of past and present machine learning gravity models. All values are taken from their corresponding reference, but further context is warranted as each model assessed accuracy in different ways and on different asteroids. This section aims to provide relevant details regarding these metrics for completeness.

For the Gaussian process gravity model reported in Reference (45), the number of model parameters are not explicitly reported, but can be deduced. Gaussian processes are defined by their covariance matrix and kernel function. The covariance matrix scales as  $\mathcal{O}(N^2)$  and the maximum number of data points used were  $N = 3,600$ . This suggests that the minimum number of parameters used in the model is 12,960,000. The accuracy for these models are also reported at fixed radii from the center of mass for each asteroid rather than across the full domain, so the values can be considered upper-bounds. Moreover, the model is shown to diverge at high altitudes hence is not valid globally.

For the extreme learning machine gravity model reported in Reference (6), the model size is determined by the fact that there are 50,000 hidden nodes in the ELM. The first 50,000 parameters constitute the random weights connecting the inputs to the hidden layer, and the second 50,000 parameters are the learned weights from the hidden layer to the output layer, totalling to 100,000 total model parameters. The asteroid modeled is 25143 Itokawa. The error in the Reference (6) is reported in terms of absolute terms rather than relative terms, however using Figure 9 from the reference, it can be approximated that the relative error varies between 1% and 10%.

In Reference (46), the neural network gravity model is reported to use 512 nodes per hidden layer for 6 hidden layers ( $512^2 * 6 = 1,572,864$  parameters). The paper reports 1,000,000 training data were generated and divided into an 8:2 ratio between training and testing data, totalling to 800,000 training data. The asteroid investigated is also 433-Eros and the average relative error of the test set is reported as 0.35% in their Table 3.

For GeodesyNets (7), SIRENs of 9 hidden layers with 100 nodes each are used. Four asteroids are studied: Bennu, Churyumov-Gerasimenko, Eros, and Itokawa. In their supplementary materials (Table S4), the relative error about Eros is reported at three characteristic altitudes. At their lowest altitude, the average error is 0.571% and their highest altitude is 0.146%. In attempts to quantify error across the entire high and low altitude regime, these values are averaged for the reported value of 0.359%.

Finally for the PINN-GM-III, the average error reported is taken from Section 3.4.2 Figure 3.31d. With a 20-node per hidden layer network for 8 layers, trained on 4,096 data points and trained for 32,768 epochs the model was able to achieve an average relative error of < 0.2% about Eros.

## Appendix E

### Pines Algorithm

#### E.1 Introduction

As the age of large, highly-coordinated satellite constellations grows closer to reality, the need for fast, analytic orbit propagation is paramount to efficient satellite simulation and planning. To achieve efficient propagation, however, simulations are often burdened by the fidelity of the gravity model used. With a coarse gravity model, the simulation may run efficiently, but trajectories are only valid over short time scales. Alternatively, with a high-fidelity gravity model, trajectories will become more accurate, but at the cost of slow runtimes – inhibiting larger sensitivity studies or Monte Carlo analysis.

Explicitly, analytic calculation of the gravitational acceleration imparted by a heterogeneous mass is a computationally expensive task when using high-fidelity gravity field models. Traditionally this calculation is done by first representing the gravitational potential as a spherical harmonic series expanded to a finite degree  $l$  and order  $m$ , converting this expansion to non-singular representation, and then taking the gradient to compute the gravitational acceleration. A popular implementation of this process is Pines' formulation (84).

Despite its popularity within the astrodynamics community, Pines' formulation has a high computational cost – scaling as  $\mathcal{O}(l^2)$  where  $l$  maximum degree of the gravity model (see Figure E.2) (142). For low-fidelity gravity models, this computational inefficiency is negligible, and the ability to propagate orbits for one or many spacecraft is unaffected. However, when scientists and engineers use high-fidelity representations of the gravity field like Earth's EGM2008 model

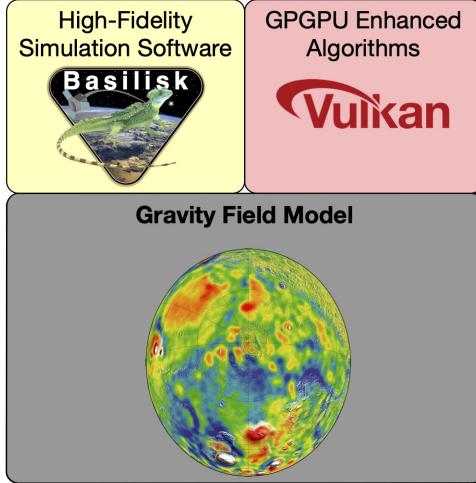


Figure E.1: GPGPU enhanced gravity algorithms for high-fidelity astrodynamics software enables support for high accuracy orbit propagation with lower runtimes.

(reaching degree and order 2160) or the Moon’s model produced by GRAIL (degree 900) – Pines’ formulation can demand millions of computations per timestep to produce the corresponding acceleration (53; 143). Consequently, high-fidelity gravity models impose a large computational bottleneck for astrodynamics simulation – often requiring trajectory designers and researchers to choose either simulation speed or accuracy.

A traditional solution to this computational bottleneck is to use a truncated gravity model – one that provides sufficiently many terms in the spherical harmonic expansion to capture the coarsest gravitational perturbations but few enough to prevent exorbitant amounts of compute time. Such a solution is often acceptable when generating trajectories over short time intervals or for missions that do not require precise orbits. In the case of longer simulations, however, this solution is untenable as the effect of unaccounted gravitational perturbations will accumulate through the dynamics and negatively impact the trajectory (Figure E.2). This work proposes that, in principle, given the state of modern day computing, astrodynamicists need not compromise between simulation speed and accuracy. Explicitly, the computational overhead of Pines’ formulation might be significantly reduced if transitioned off of traditional CPUs and onto alternative hardware like Graphics Processing Units (GPUs).

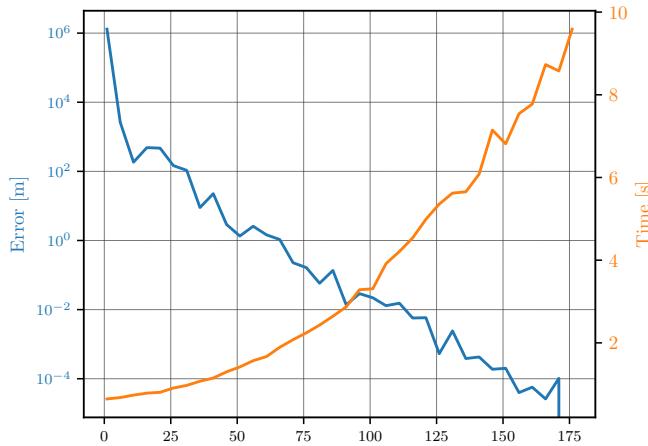


Figure E.2: Compute time and final error associated with simulating a spacecraft at 600km altitude orbiting for four hours real-time using Pines' formulation as a function of spherical harmonic degree.

GPUs are designed to solve problems in parallel, unlike CPUs which demand serial execution. As such, if a problem can be properly decomposed into parallelized pieces, GPU algorithms can offer order of magnitude performance gains over their serial CPU counterpart (144). Performance gains of this magnitude have opened up entirely new domains physical modeling within the scientific community<sup>1</sup> — but their application for gravity modeling remains relatively unsaturated (145).

Kefan et. al. presented a CUDA implementation of a spherical harmonic gravity model claiming positive speed up ratios on the GPU, but fail to provide repeatable or verifiable results (146). Moreover, the algorithm presented requires vendor specific hardware and does not provide details into the model used. Hupca et. al. demonstrated that inverse spherical harmonic transforms can be evaluated rapidly on multi-core systems or GPUs by but only when evaluating the transform across a 2D grid rather than a single point location (147). Atallah et. al. propose a GPGPU implementation of the Chebyshev Picard Method which is quick to evaluate, but is only an approximation of the field and only valid over a finite domain (148). None of these paper provide an explicit, analytic computation of the gravitational force experienced at a single point in a cross-platform, GPGPU compatible manner. This work attempts to fill this hole by providing an implementation of Pines' formulation on a GPU using Vulkan, a modern GPGPU compute and graphics API.

<sup>1</sup> <https://www.nvidia.com/content/gpu-applications/PDF/gpu-applications-catalog.pdf>

Vulkan is the only API that is simultaneously cross-platform, officially supported, and does not have hardware specific stipulations<sup>2</sup>. Alternative GPU APIs like CUDA, OpenGL/CL, Metal, and DirectX each fail in at least one of these three categories as of 2020. In addition to its broader applicability, Vulkan is also considered a low-level GPU API providing developers with direct access and control of the GPU, opening opportunities for powerful optimization. Developers are given near-complete control of the graphics and compute pipelines allowing for careful design of command buffers and their dispatch. Coupling these features with traditional tuning of dispatch calls, memory transfer, and thread barriers allows developers to accumulate maximum speed gains. Discovering the optimal permutations of these features and design choices require extensive testing and careful formulations of the underlying algorithm at hand. This work discusses various ways to decompose Pines' formulation and quantitatively explores how these optimization affect performance.

## E.2 Pines' Formulation

Exploiting GPU hardware to efficiently evaluate Pines' formulation requires refactoring the equations from the original algorithm. As such, it is advantageous to provide the unperturbed algorithm before investigating specific optimization strategies.

### E.2.1 Gravitational Potential

Pines' formulation provides an analytic formula that computes the acceleration imparted by a non-homogenous, massive body. This is done by first expressing the potential as a series expansion of spherical harmonics.

$$U(\mathbf{r}) = \frac{\mu}{r} \sum_{l=0}^{\infty} \sum_{m=0}^l \left(\frac{1}{r}\right)^l P_{l,m}[\sin(\phi)] [C'_{l,m} \cos(m\lambda) + S'_{l,m} \sin(m\lambda)] \quad (\text{E.1})$$

where  $r$  is the magnitude of the position vector with respect to the center of mass of the gravitational body,  $\mu$  is the gravitational parameter of the body,  $P_{l,m}$  are the associated legendre polynomials,  $\phi$  is the geodetic latitude,  $\lambda$  is the geodetic longitude, and  $C'_{l,m}$  and  $S'_{l,m}$  are the Stokes' coefficients.

---

<sup>2</sup> <https://www.khronos.org/vulkan/>

While the potential can technically remain in this form, it is more commonly expressed with non-dimensional coefficients

$$C_{l,m} = \frac{C'_{l,m}}{R_{\text{ref}}^l m} \quad (\text{E.2})$$

$$S_{l,m} = \frac{S'_{l,m}}{R_{\text{ref}}^l m} \quad (\text{E.3})$$

where  $R_{\text{ref}}$  is a chosen reference radius, typically defined as the radius of the sphere which encloses all mass elements of the body (the Brillouin sphere) (43). Simplifying the expression provides

$$U(\mathbf{r}) = \frac{\mu}{r} \sum_{l=0}^{\infty} \sum_{m=0}^l \left( \frac{R_{\text{ref}}}{r} \right)^l P_{l,m}[\sin(\phi)] [C_{l,m} \cos(m\lambda) + S_{l,m} \sin(m\lambda)] \quad (\text{E.4})$$

Again, the equation can remain in this form, however the terms in the series grow exponentially with the degree  $l$ . To retain numerical stability, a normalization factor is introduced by Lundberg and Schutz (149)

$$N_{l,m} = \sqrt{\frac{(l-m)!(2-\delta_m)(2l+1)}{(l+m)!}} \quad (\text{E.5})$$

such that the coefficients and the associated legendre polynomials become

$$\bar{C}_{l,m} = \frac{C_{l,m}}{N_{l,m}} \quad (\text{E.6})$$

$$\bar{S}_{l,m} = \frac{S_{l,m}}{N_{l,m}} \quad (\text{E.7})$$

$$\bar{P}_{l,m}[x] = P_{l,m}[x] N_{l,m} \quad (\text{E.8})$$

altogether providing

$$U(\mathbf{r}) = \frac{\mu}{r} \sum_{l=0}^{\infty} \sum_{m=0}^l \left( \frac{R_{\text{ref}}}{r} \right)^l \bar{P}_{l,m}[\sin(\phi)] [\bar{C}_{l,m} \cos(m\lambda) + \bar{S}_{l,m} \sin(m\lambda)] \quad (\text{E.9})$$

### E.2.2 Gravitational Acceleration

To compute the gravitational acceleration, the gradient of Equation (E.9) must be taken. However, in the case of  $\phi = -\frac{\pi}{2}$  or  $\frac{\pi}{2}$ , the gradient diverges. As such, Pines introduced an alternative formulation which bypasses this numerical instability by changing to dimensionless coordinates

within the cartesian coordinate frame where

$$\mathbf{r} = r \begin{pmatrix} s \\ t \\ u \end{pmatrix} \quad \hat{\mathbf{i}} = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \quad \hat{\mathbf{j}} = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \quad \hat{\mathbf{k}} = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \quad (\text{E.10})$$

such that

$$r = \sqrt{x^2 + y^2 + z^2} \quad (\text{E.11})$$

$$s = \frac{x}{r} \quad (\text{E.12})$$

$$t = \frac{y}{r} \quad (\text{E.13})$$

$$u = \frac{z}{r} \quad (\text{E.14})$$

Using these alternative coordinates, the associate Legendre polynomials can be rewritten as

$$P_{l,m}[\sin(\phi)] = P_{l,m}[u] = (1 - u^2)^{\frac{m}{2}} A_{l,m}[u] \quad (\text{E.15})$$

where

$$A_{l,m}[u] = \frac{d^m}{du^m} P_l[u] = \frac{1}{2^l l!} \frac{d^{l+m}}{du^{l+m}} (u^2 - 1)^l \quad (\text{E.16})$$

Moreover, if one defines  $\xi$  as

$$\xi = \cos(\phi) \cos(\lambda) + j \cos(\phi) \sin(\lambda) = \frac{x}{r} + j \frac{y}{r} = s + jt \quad (\text{E.17})$$

then

$$\xi^m = \cos^m(\phi) e^{jm\lambda} = (s + jt)^m \quad (\text{E.18})$$

such that

$$R_m[s, t] = \operatorname{Re}\{\xi^m\} \quad (\text{E.19})$$

$$I_m[s, t] = \operatorname{Im}\{\xi^m\} \quad (\text{E.20})$$

then the potential can be rewritten as

$$U(\mathbf{r}) = \frac{\mu}{r} \sum_{l=0}^{\infty} \sum_{m=0}^l \left( \frac{R_{\text{ref}}}{r} \right)^l \bar{A}_{l,m}[u] \{ \bar{C}_{l,m} R_m[s, t] + \bar{S}_{l,m} I_m[s, t] \} \quad (\text{E.21})$$

Defining

$$D_{l,m}[s,t] = \bar{C}_{l,m}R_m[s,t] + \bar{S}_{l,m}I_m[s,t] \quad (\text{E.22})$$

$$\rho_l[r] = \frac{\mu}{r} \left( \frac{R_{\text{ref}}}{r} \right)^l \quad (\text{E.23})$$

the potential simplifies further to

$$U(\mathbf{r}) = \sum_{l=0}^{\infty} \sum_{m=0}^l \rho_l[r] \bar{A}_{l,m}[u] D_{l,m}[s,t] \quad (\text{E.24})$$

where the constituent terms abide by the following recursion relationships

$$R_m[s,t] = sR_{m-1}[s,t] - tI_{m-1}[s,t] \quad (\text{E.25})$$

$$I_m[s,t] = sI_{m-1}[s,t] + tR_{m-1}[s,t] \quad (\text{E.26})$$

$$\bar{A}_{l,l}[u] = \sqrt{\frac{(2l+1)(2-\delta_l)}{(2l)(2-\delta_{l-1})}} \bar{A}_{l-1,l-1}[u] \quad (\text{E.27})$$

$$\bar{A}_{l,l-1}[u] = u \sqrt{\frac{(2l)(2-\delta_{l-1})}{2-\delta_l}} \bar{A}_{l,l}[u] \quad (\text{E.28})$$

$$\bar{A}_{l,m}[u] = \frac{N_{l,m}}{l-m} ((2l-1)uA_{l-1,m}[u] - \quad (\text{E.29})$$

$$(l+m-1)A_{l-2,m}[u]) \quad (\text{E.30})$$

with the following initial conditions

$$R_0[s,t] = 1 \quad (\text{E.31})$$

$$I_0[s,t] = 0 \quad (\text{E.32})$$

$$\bar{A}_{0,0}[u] = 1 \quad (\text{E.33})$$

Equation (E.30) can be further simplified by expanding  $N_{l,m}$  for cases where  $l \geq (m+2)$ , into  $N_{1_{l,m}}$  and  $N_{2_{l,m}}$  such that

$$\bar{A}_{l,m}[u] = N_{1_{l,m}} u A_{l-1,m}[u] - N_{2_{l,m}} A_{l-2,m}[u] \quad (\text{E.34})$$

where

$$N_{1_{l,m}} = \sqrt{\frac{(2l+1)(2l-1)}{(l-m)(l+m)}} \quad (\text{E.35})$$

$$N_{2_{l,m}} = \sqrt{\frac{(l+m-1)(2l+1)(l-m-1)}{(l-m)(l+m)(2l-3)}} \quad (\text{E.36})$$

To compute the acceleration, the gradient of the potential must be taken with respect to the non-dimensional coordinates  $u$ ,  $t$ ,  $s$ , and  $r$ .

$$\nabla U(\mathbf{r}) = \frac{\partial U}{\partial r} \frac{\partial r}{\partial \mathbf{r}} + \frac{\partial U}{\partial s} \frac{\partial s}{\partial \mathbf{r}} + \frac{\partial U}{\partial t} \frac{\partial t}{\partial \mathbf{r}} + \frac{\partial U}{\partial u} \frac{\partial u}{\partial \mathbf{r}} \quad (\text{E.37})$$

$$\frac{\partial r}{\partial \mathbf{r}} = \frac{1}{r} \hat{\mathbf{r}} \quad (\text{E.38})$$

$$\frac{\partial s}{\partial \mathbf{r}} = \frac{1}{r} \hat{\mathbf{i}} - \frac{s}{r} \hat{\mathbf{r}} \quad (\text{E.39})$$

$$\frac{\partial t}{\partial \mathbf{r}} = \frac{1}{r} \hat{\mathbf{j}} - \frac{t}{r} \hat{\mathbf{r}} \quad (\text{E.40})$$

$$\frac{\partial u}{\partial \mathbf{r}} = \frac{1}{r} \hat{\mathbf{k}} - \frac{u}{r} \hat{\mathbf{r}} \quad (\text{E.41})$$

$$\mathbf{g} = \left( \frac{\partial U}{\partial r} - \frac{s}{r} \frac{\partial U}{\partial s} - \frac{t}{r} \frac{\partial U}{\partial t} - \frac{u}{r} \frac{\partial U}{\partial u} \right) \hat{\mathbf{r}} + \frac{1}{r} \frac{\partial U}{\partial s} \hat{\mathbf{i}} + \frac{1}{r} \frac{\partial U}{\partial t} \hat{\mathbf{j}} + \frac{1}{r} \frac{\partial U}{\partial u} \hat{\mathbf{k}} \quad (\text{E.42})$$

The partials of the potential can be applied directly to their interior variables

$$\frac{\partial U}{\partial r} = \sum_{l=0}^{\infty} \sum_{m=0}^l \frac{\partial \rho_l[r]}{\partial r} \bar{A}_{l,m}[u] D_{l,m}[s, t] \quad (\text{E.43})$$

$$\frac{\partial U}{\partial u} = \sum_{l=0}^{\infty} \sum_{m=0}^l \rho_l[r] \frac{\partial \bar{A}_{l,m}[u]}{\partial u} D_{l,m}[s, t] \quad (\text{E.44})$$

$$\frac{\partial U}{\partial s} = \sum_{l=0}^{\infty} \sum_{m=0}^l \rho_l[r] \bar{A}_{l,m}[u] \frac{\partial D_{l,m}[s, t]}{\partial s} \quad (\text{E.45})$$

$$\frac{\partial U}{\partial t} = \sum_{l=0}^{\infty} \sum_{m=0}^l \rho_l[r] \bar{A}_{l,m}[u] \frac{\partial D_{l,m}[s, t]}{\partial t} \quad (\text{E.46})$$

$$(\text{E.47})$$

Evaluating the partials:

$$\frac{\partial \rho_l[r]}{\partial r} = -\frac{(l+1)}{R_{\text{ref}}} \rho_{l+1}[r] \quad (\text{E.48})$$

$$\frac{\partial \bar{A}_{l,m}[u]}{\partial u} = \frac{N_{l,m}}{N_{l,m+1}} \bar{A}_{l,m+1}[u] \quad (\text{E.49})$$

$$\frac{\partial D_{l,m}[s,t]}{\partial s} = m(\bar{C}_{l,m} R_{m-1}[s,t] + \bar{S}_{l,m} I_{m-1}[s,t]) \quad (\text{E.50})$$

$$\frac{\partial D_{l,m}[s,t]}{\partial t} = m(\bar{S}_{l,m} R_{m-1}[s,t] - \bar{C}_{l,m} I_{m-1}[s,t]) \quad (\text{E.51})$$

Inserting Equations (E.48) - (E.51) into Equations (E.43) - (E.46)

$$\begin{aligned} \frac{\partial U}{\partial r} &= \sum_{l=0}^{\infty} \sum_{m=0}^l -\frac{(l+1)}{R_{\text{ref}}} \rho_{l+1}[r] \bar{A}_{l,m}[u] D_{l,m}[s,t] \\ \frac{\partial U}{\partial u} &= \sum_{l=0}^{\infty} \sum_{m=0}^l \rho_l[r] \frac{N_{l,m}}{N_{l,m+1}} \bar{A}_{l,m+1}[u] D_{l,m}[s,t] \\ \frac{\partial U}{\partial s} &= \sum_{l=0}^{\infty} \sum_{m=0}^l \rho_l[r] \bar{A}_{l,m}[u] m(\bar{C}_{l,m} R_{m-1}[s,t] + \bar{S}_{l,m} I_{m-1}[s,t]) \\ \frac{\partial U}{\partial t} &= \sum_{l=0}^{\infty} \sum_{m=0}^l \rho_l[r] \bar{A}_{l,m}[u] m(\bar{S}_{l,m} R_{m-1}[s,t] - \bar{C}_{l,m} I_{m-1}[s,t]) \end{aligned} \quad (\text{E.52})$$

Setting  $a_1 = \frac{1}{r} \frac{\partial U}{\partial s}$ ,  $a_2 = \frac{1}{r} \frac{\partial U}{\partial t}$ ,  $a_3 = \frac{1}{r} \frac{\partial U}{\partial u}$ , and  $a_4 = \left( \frac{\partial U}{\partial r} - \frac{s}{r} \frac{\partial U}{\partial s} - \frac{t}{r} \frac{\partial U}{\partial t} - \frac{u}{r} \frac{\partial U}{\partial u} \right)$ , refactoring  $\rho_l[r]$ , and simplifying yields

$$a_1[r, s, t, u] = \sum_{l=0}^{\infty} \sum_{m=0}^l \frac{\rho_{l+1}[r]}{R_{\text{ref}}} m \bar{A}_{l,m}[u] (\bar{C}_{l,m} R_{m-1}[s,t] + \bar{S}_{l,m} I_{m-1}[s,t]) \quad (\text{E.53})$$

$$a_2[r, s, t, u] = \sum_{l=0}^{\infty} \sum_{m=0}^l \frac{\rho_{l+1}[r]}{R_{\text{ref}}} m \bar{A}_{l,m}[u] (\bar{S}_{l,m} R_{m-1}[s,t] - \bar{C}_{l,m} I_{m-1}[s,t]) \quad (\text{E.54})$$

$$a_3[r, s, t, u] = \sum_{l=0}^{\infty} \sum_{m=0}^l \frac{\rho_{l+1}[r]}{R_{\text{ref}}} \frac{N_{l,m}}{N_{l,m+1}} \bar{A}_{l,m+1}[u] D_{l,m}[s,t] \quad (\text{E.55})$$

$$a_4[r, s, t, u] = \sum_{l=0}^{\infty} \sum_{m=0}^l \frac{\rho_{l+1}[r]}{R_{\text{ref}}} \frac{N_{l,m}}{N_{l+1,m+1}} \bar{A}_{l+1,m+1}[u] D_{l,m}[s,t] \quad (\text{E.56})$$

where

$$\frac{N_{l,m}}{N_{l,m+1}} = \sqrt{\frac{(l-m)(2-\delta_m)(l+m+1)}{2-\delta_{m+1}}} \quad (\text{E.57})$$

$$\frac{N_{l,m}}{N_{l+1,m+1}} = \sqrt{\frac{(l+m+2)(l+m+1)(2l+1)(2-\delta_m)}{(2l+3)(2-\delta_{m+1})}} \quad (\text{E.58})$$

The final acceleration can then be expressed as:

$$\begin{aligned} \mathbf{g} = & (a_1[r, s, t, u] + s \cdot a_4[r, s, t, u])\hat{\mathbf{i}} + \\ & (a_2[r, s, t, u] + t \cdot a_4[r, s, t, u])\hat{\mathbf{j}} + \\ & (a_3[r, s, t, u] + u \cdot a_4[r, s, t, u])\hat{\mathbf{k}} \end{aligned} \quad (\text{E.59})$$

## Appendix F

### GPU Algorithm

This work aims to develop an algorithm that evaluates Equation (E.59) as efficiently as possible on GPU hardware. This requires first understanding the underlying software and hardware of GPUs.

#### F.1 GPGPU Software

Foremost, GPU programs operate on the kernel scale. A kernel is the specific algorithm dispatched to the GPU to be computed asynchronously. Each kernel invocation is assigned a unique thread, an ID, and local memory. These kernel invocations are typically dispatched in work-groups of a fixed, user-defined size of (x, y, z) (`WorkGroupSize`) up to some limit specified by the hardware. Each work-group has a unique shared memory space where kernel invocations within that work-group can exchange data with one another at faster rate than typical global access memory (150). A GPU program begins execution when the CPU dispatches one or many work-groups – also of user-specified dimensions (x, y, z) (`NumWorkGroups`). As such, if a user defines the `WorkGroupSize` as (16, 8, 4) there will be 512 kernel invocations within that work-group, and if `NumWorkGroups` is defined as (128, 256, 64) for a total of 2,097,152 work-groups, there will ultimately be 1,073,741,824 total kernel invocations sent to the GPU for execution.

## F.2 GPGPU Hardware

GPU vendors like NVIDIA and AMD use similar microarchitectures<sup>1 2 3 4</sup>. Every GPU will have a collection of Streaming Multiprocessors (SM) or Compute Units (CU) (NVIDIA and AMD language respectively) which each house hardware designed to schedule and execute a work-group. The threads (kernel invocations) within a work-group are then divided into batches called warps (NVIDIA) or wavefronts (AMD). A warp is defined as a batch of 32 threads, and a wavefront is defined as a batch of 64 threads. The remainder of this work will exclusively use AMD language and sizes as the default. Continuing with the earlier example, if a work-group is of size (16, 8, 4), or 512 threads and is sent to a CU, 8 wavefronts will be executed on that CU. By extension, if there are (128, 256, 64) work-groups distributed across 32 CUs, a total of  $16,777,216/32 = 524,288$  wavefronts must be executed per CU. This number is still large, however, each CU can also manage multiple wavefronts simultaneously to hide memory latency. I.e. if a single wavefront is waiting from a result in memory, a different wavefront within that work-group can simultaneously run on the momentarily unutilized arithmetic hardware within the same CU. On modern AMD GPUs, up to 40 wavefronts can be scheduled per CU. This ultimately brings the total “tasks” per CU to  $524,288/40 \approx 13,108$  – a much more approachable number than the individual 1,073,741,824 invocations that needed to be completed (151).

## F.3 GPGPU General Optimization Strategies

GPU optimization is a nuanced endeavour which demands attention to both GPU hardware and software. This subsection presents the optimizations considered in the development of this algorithm thus far, but should not be considered extensive. Further discussion and additional optimization strategies can be found in resources like the CUDA C++ Best Practices Guide, hardware whitepapers, and other online forums (152; 153).

---

<sup>1</sup> <https://www.nvidia.com/content/dam/en-zz/Solutions/design-visualization/technologies/turing-architecture/NVIDIA-Turing-Architecture-Whitepaper.pdf>

<sup>2</sup> <https://www.amd.com/system/files/documents/rdna-whitepaper.pdf>

<sup>3</sup> <https://www.techpowerup.com/gpu-specs/docs/amd-gcn1-architecture.pdf>

<sup>4</sup> <https://www.nvidia.com/en-us/data-center/resources/pascal-architecture-whitepaper/>

The first consideration for GPU programming is that all threads within each wavefront are executed in lockstep, meaning all threads within the wavefront are expected to perform the same instruction. If there is branching between different threads in the same wavefront, the GPU will halt all threads that do not meet the branch condition and leave them idle until the branch is complete. Consequently, branches impart a performance penalty that grows in proportion to the number of inactive threads and cycles to complete the branch. As such it is strongly recommended to minimize the number of branches in the kernel whenever possible. Common examples of branching include **if-else** statements or conditional **for** loops.

In a similar vein, bank conflicts should be minimized. If multiple kernel invocations need to access to the same shared memory bank, the store and load operations must be executed sequentially. This synchronization imparts a performance penalty as GPU cycles are often much slower than their CPU cycle counterpart. It is therefore recommended that serial work be left to the CPU whenever possible. To avoid the synchronization caused by bank conflicts, the programmer can ensure that memory accesses per thread are separated by the width of the bank through proper striding and padding.

Another optimization strategy is to use Single Instruction Multiple Data (SIMD) operations. Both CPUs and GPUs have specialized instructions that allow for the same operation to be performed on multiple data simultaneously. In the case of a 4D vector, SIMD operations allow all elements to be simultaneously loaded, stored, or operated on. E.g. it costs the same amount of cycles to compute  $a=4*6$  as it does to compute  $a=(1,2,3,4)*(5,6,7,8)$ . It is encouraged to maximize the number of SIMD operations per kernel wherever possible.

It is also necessary to maximize occupancy on a GPU. This means ensuring that most, if not all, threads remain active, and all compute units are adequately supplied with many wavefronts. If either of these criteria are not met, the GPU will not be operating at full capacity, and can become slower than if the task were executed on the CPU.

Finally, its important to consider if a kernel is compute bound (algorithm efficiency limited by how many operations need to be performed) or memory bound (algorithm efficiency limited

by bottlenecks in memory transfer and overhead). To determine which regime a compute kernel operates within one must first compute the arithmetic intensity of the algorithm, which equates to the number of operations per byte of data transferred to the GPU. If the arithmetic intensity is sufficiently large, the kernel will always be compute bound and the programmer should prioritize minimizing the number of computation cycles. Alternatively, if the arithmetic intensity is low, the programmer should prioritize throughput – optimizing efficient memory load and store requests (154).

## F.4 Pines' Formulation Core Routines

To optimize Pines' formulation for a GPU, the algorithm must be broken down into its constituent parts. This work decomposes the algorithm into two primary routines. Working backwards, the first routine is the computing the double summation for  $a_1-a_4$ .

### F.4.1 Core Routine 1: Data Reduction

Assuming the addends of the series are computed a priori, Equations (E.53) - (E.56) simply represent the summation of all terms within a lower-triangle 2D matrix (this assumes the gravity model used is of equal degree and order such that  $N = l_{\max} = m$ ). If that matrix is then flattened into a single 1D array, there exist GPU data reduction techniques that substantially decrease the total number of cycles needed to compute the sum. Explicitly, summing all terms within a lower triangular matrix with a total of  $N(N + 1)/2$  entries requires  $\mathcal{O}(n^2)$  cycles on a CPU. The same reduction algorithm takes as few as  $\mathcal{O}(\log_2 n)$  cycles on a GPU by using sequential memory access patterns and shared memory across all threads in a work-group. For brevity, the core routine is expressed in Algorithm 1 and visualized in Figure F.1. Additional optimization techniques exist beyond those expressed in Algorithm 1 like loop unrolling and removing instruction overhead, though a deeper discussion of such techniques is left to the many resources available online<sup>5</sup>.

---

<sup>5</sup> <http://developer.download.nvidia.com/assets/cuda/files/reduction.pdf>

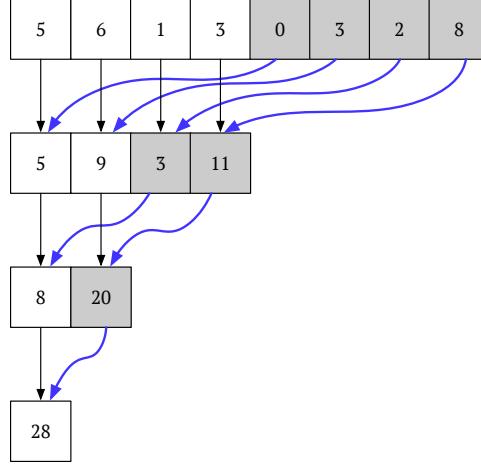


Figure F.1: Data Reduction Technique

#### F.4.2 Core Routine 2: Legendre Matrix

The more challenging routine to put on a GPU is computing the series addends prior to data reduction. Redefining the addends from Equations (E.53) - (E.56) as  $a_{i_{l,m}}$  and expanding yields:

$$a_{1_{l,m}}[r, s, t, u] = \frac{\rho_{l+1}[r]}{R_{\text{ref}}} m \bar{A}_{l,m}[u] (\bar{C}_{l,m} R_{m-1}[s, t] + \bar{S}_{l,m} I_{m-1}[s, t]) \quad (\text{F.1})$$

$$a_{2_{l,m}}[r, s, t, u] = \frac{\rho_{l+1}[r]}{R_{\text{ref}}} m \bar{A}_{l,m}[u] (\bar{S}_{l,m} R_{m-1}[s, t] - \bar{C}_{l,m} I_{m-1}[s, t]) \quad (\text{F.2})$$

$$a_{3_{l,m}}[r, s, t, u] = \frac{\rho_{l+1}[r]}{R_{\text{ref}}} \frac{N_{l,m}}{N_{l,m+1}} \bar{A}_{l,m+1}[u] (\bar{C}_{l,m} R_m[s, t] + \bar{S}_{l,m} I_m[s, t]) \quad (\text{F.3})$$

$$a_{4_{l,m}}[r, s, t, u] = \frac{\rho_{l+1}[r]}{R_{\text{ref}}} \frac{N_{l,m}}{N_{l+1,m+1}} \bar{A}_{l+1,m+1}[u] (\bar{C}_{l,m} R_m[s, t] + \bar{S}_{l,m} I_m[s, t]) \quad (\text{F.4})$$

These expressions are not immediately amenable for GPU optimization as they are idiosyncratic, each with different variables and indices. The expressions can be homogenized however by defining

**Algorithm 3:** Data Reduction Loop

**Input:** LocalInvocationIndex  $localIdx$

```

1 finalValue[localIdx] =  $a_{i_{l,m}}$ ;
2 barrier();
3 for s = WorkGroupSize.x/2; s > 0; s >>= 1 do
4   if localIdx < s then
5     | finalValue[localIdx] += finalValue[localIdx + s];
6   end
7   barrier();
8 end

```

---

the following constants:

$$Q = \frac{\rho_{l+1}[r]}{R_{\text{ref}}} \quad (\text{F.5})$$

$$c_1 = \frac{N_{l,m}}{N_{l,m+1}} \quad (\text{F.6})$$

$$c_2 = \frac{N_{l,m}}{N_{l+1,m+1}} \quad (\text{F.7})$$

$$d_1 = s\bar{C}_{l,m} + t\bar{S}_{l,m} \quad (\text{F.8})$$

$$d_2 = s\bar{S}_{l,m} - t\bar{C}_{l,m} \quad (\text{F.9})$$

$$(\text{F.10})$$

and expanding  $R_m$  and  $I_m$  in terms of  $R_{m-1}$  and  $I_{m-1}$ . Simplifying, the expressions for all  $a_{i_{l,m}}$  become significantly more consistent:

$$\begin{bmatrix} a_{1_{l,m}} \\ a_{2_{l,m}} \\ a_{3_{l,m}} \\ a_{4_{l,m}} \end{bmatrix} = \begin{bmatrix} Q \\ Q \\ Q \\ Q \end{bmatrix} \begin{bmatrix} m \\ m \\ c_1 \\ c_2 \end{bmatrix} \begin{bmatrix} A_{l,m}[u] \\ A_{l,m}[u] \\ A_{l,m+1}[u] \\ A_{l+1,m+1}[u] \end{bmatrix} \left( \begin{bmatrix} \bar{C}_{l,m} \\ \bar{S}_{l,m} \\ d_1 \\ d_1 \end{bmatrix} \begin{bmatrix} R_{m-1} \\ R_{m-1} \\ R_{m-1} \\ R_{m-1} \end{bmatrix} + \begin{bmatrix} \bar{S}_{l,m} \\ -\bar{C}_{l,m} \\ d_2 \\ d_2 \end{bmatrix} \begin{bmatrix} I_{m-1} \\ I_{m-1} \\ I_{m-1} \\ I_{m-1} \end{bmatrix} \right) \quad (\text{F.11})$$

Equation F.11 is advantageous for GPU computing for two reasons. The first is SIMD compatibility. The expressions for  $a_{1_{l,m}} - a_{4_{l,m}}$  share many common terms which can be loaded into memory simultaneously via SIMD operations. Moreover, each addend uses the same six multiplications and one addition – common instructions that can be shared across all expressions. By converting

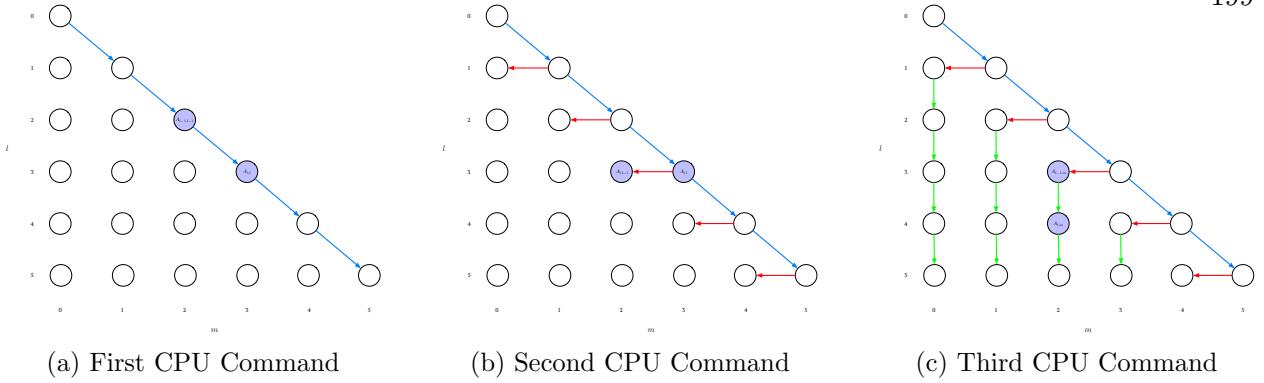


Figure F.2: CPU Data Flow

the equations into vectorized form, the shared multiply and addition instructions alone reduce the number of cycles needed to compute all addends by an additional factor of four.

The second advantage of Equation (F.11) is that  $Q, c_1, c_2, d_1, d_2, \bar{C}_{l,m}, \bar{S}_{l,m}$  are all independent variables that can be computed without any knowledge of prior operations within the algorithm. This means there is no recursion or unique branch criteria to calculate their values, and thereby no synchronization is imposed on the GPU for these variables.

Beyond these terms, there is the challenge of computing the remaining variables  $A_{l,m}, A_{l,m+1}, A_{l+1,m+1}, R_{m-1}, I_{m-1}$  which *do* require knowledge of prior terms due to their recursion formulas. Two possible solutions exist for evaluating these terms. The first is a memory-centric approach where these terms are only computed once, stored in memory, and then loaded back into registers as needed. The alternative is a compute-centric approach where the values are computed as needed. The former demands more memory operations and fewer arithmetic operations while the latter requires less memory but at the cost of redundant calculations.

#### F.4.3 Memory Bound

In the memory bound case, it is assumed that all terms within Equation (F.11) are computed individually and saved to local memory on the GPU. This requires careful sequencing to satisfy the recursion relationships of  $A_{l,m}, R_m$ , and  $I_m$ . There is little advantage to solving  $R_m$  and  $I_m$  on the

GPU as their formula have no elements of parallelization. As such those terms are solved on the CPU and transferred upon dispatch to the GPU. The  $A_{l,m}$  terms, however, have a mix of strict, serial recursion but also opportunities for parallelization. Specifically Eqs. (E.27) is explicitly recursive as seen in Figure F.2a, however Equations (E.28) and (E.30) (seen in Figures F.2b and F.2c) can be evaluated asynchronously across the columns. Each kernel invocation can be assigned to a different column to be computed independently. Despite this asynchronous opportunity, such approach suffers from loop divergence. With every cycle another thread / column will have been completed and the thread within the wavefront will go idle. Nevertheless the order of computation to the kernel invocation scales as  $\mathcal{O}(n)$  rather than the entire  $\mathcal{O}(n^2)$  algorithm necessary to compute the same terms on a CPU.

Beyond the asynchronous challenges of this strategy, there are also hardware limits to consider – namely available VRAM on the GPU. Assuming  $N_{S/C}$  total spacecraft are being simulated on the GPU, additional memory will need to be allocated. Table F.1 show the total number of buffers that must be stored on the GPU as well as their size. Constraining these parameters to the available memory limits of the hardware, Figure F.3 shows the maximum number of spacecraft that can be simultaneously simulated on a GPU given a chosen spherical harmonic model fidelity.

Table F.1: The buffers transferred to the GPU across algorithm lifetime

Struct Name	Variables	Type	Structs per Buffer	Buffers Per Sim
$N_{\text{Params}}$	$N_1, N_2, N_{q1}, N_{q2}$	float	$(l+1)(l+2)/2$	1
A	$a_{l,m}$	float	$(l+1)(l+2)/2$	$N_{S/C}$
Coef	$C_{l,m}, S_{l,m}$	float	$(l+1)(l+2)/2$	1
Misc	$\mu, R_0$ $l_{\max}$	float int	1 1	1
Location	$r, u$	float	1	$N_{S/C}$
Acceleration	$a_1, a_2, a_3, a_4$	float	$(l+1)(l+2)/2$	$N_{S/C}$
Euler	$R_m, I_m$	float	$(l+1)$	$N_{S/C}$

#### F.4.4 Compute Bound

The alternative formulation to the memory bound approach centers on computing all variables on the GPU directly on an as needed basis. This prevents shaders from needing to interface with

**Algorithm 4:** Legendre-Matrix Memory Bound Algorithm

---

**Input :** Unique thread ID  $l$

- 1 Compute lower diagonal  $A_{l,l-1}$ ;
- 2 *Thread barrier*;
- 3 **for**  $m = l$  **to**  $N + 2$  **do**
- 4 | Recursively solve for  $A_{l,m}$ ;
- 5 **end**

---

global memory for which write and read operations are particularly slow. This compute bound method leans on the fact that even in the memory bound case, there is still divergence at the column level such that the algorithm complexity will always be  $\mathcal{O}(n)$ . This is also true of the compute bound case exhibited in Figure F.4 which uses  $m + 2$  computations for traversing the diagonal, 2 computations to reach the off-diagonal terms, and  $2(l - m - 1)$  computations to acquire the necessary values in the  $A_{l,m}$  matrix – yielding the same  $\mathcal{O}(n)$  algorithm without the need for global memory access.

The added benefit of the compute bound approach is that there is not any functional limit to the total number of spacecraft that can be simulated at once. The only data that needs to be transferred to the GPU are the Stokes' coefficients ( $4 * l(l + 1)$  bytes sent once), the normalization parameters ( $16 * l^2$  bytes sent once) and the location of each spacecraft ( $16 * N_{S/C}$  bytes sent at each timestep). The disadvantage of using the compute bound approach is the redundant computation of intermediate terms. Each shader invocation will always need to traverse the diagonal of the matrix, and compute intermediate  $A_{i,j}$  on their way to the  $A_{l,m}$ . Despite this disadvantage, the cumulative advantages outweigh the cost of redundancy, so the compute bound approach is ultimately prioritized in subsequent discussion and benchmarking.

## F.5 Scalability to Constellations

In both the compute and memory bound approaches, there is no clear way to circumvent thread divergence when computing the Legendre matrix in the single spacecraft case. Some indices within the matrix will inevitably require more cycles to compute than their adjacent terms due

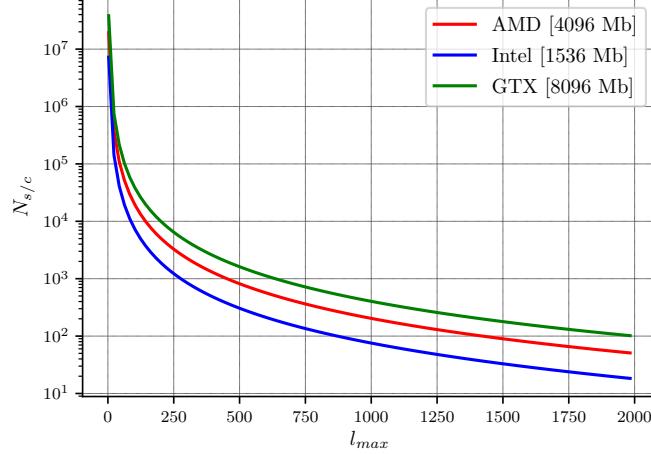


Figure F.3: Video memory consumed as a function of  $l_{\max}$  and number of spacecraft,  $N_{s/c}$

to the recursion. This ultimately compromises the lockstep nature of the wavefronts and incurs a performance penalty. Despite this challenge for the single spacecraft case, there is a workaround when simulating multiple spacecraft with the same kernel. Namely, if the Legendre matrix of each spacecraft is stacked along the z work-group dimension, and the local size of the z dimension in each work-group is sufficiently large, each wavefront can be assigned a specific index,  $l, m$ , such that all threads within that wavefront execute the exact same instructions with no divergence. Such approach requires sufficiently many satellites in the simulation to maximize occupancy on the GPU, but should avoid the divergence penalty.

## F.6 Benchmarks

Optimization of this algorithm is ongoing; however preliminary results and benchmarking methodology are presented to demonstrate how this work is being verified. Foremost this algorithm is tested on two different GPUs. The first is the Intel HD Graphics 630 with 1536 MB of VRAM, the second is a AMD Radeon Pro 560 with 4096 MB of VRAM. The former is a consumer grade integrated GPU available on modern CPU processors, while the latter is a more performant discrete graphics card at the high-end of the consumer spectrum. Both GPUs are run on a 15" 2017 Macbook Pro with a 3.1 GHz Quad-Core Intel Core i7 CPU with 16 Gb of 2133 MHz LPDDR3 RAM. The

performance of the GPGPU implementation is tested by varying the number of spacecraft simulated simultaneously as well as changing the dimensions of the work-group size by factors of 2. The maximum number of kernel invocations that can be spawned on the Intel integrated graphics card is 256 whereas the AMD card allows for as many as 1024. As such the dimensions of the work-group must always multiply such that they remain less than or equal to 256 or 1024 respectively.

Such benchmarks are important to coarsely characterize GPU occupancy and thread divergence. As discussed, the Legendre matrix computation has unavoidable thread divergence along the columns, but otherwise has coalesced memory access and shares many of the same instructions for intermediate computations. Therefore, when decreasing the work-group size x-dimension (which corresponds to how many elements of  $A_{l,m}$  are evaluated in the work-group) thread divergence is reduced as fewer columns are seen (performance gain), but the number of shared intermediate instructions among the work-group decrease (performance penalty). By extension, when varying the local group size in z-dimension (corresponding to the number of simultaneously computed Legendre matrices), these tradeoffs grow more exaggerated though the general wavefront efficiency should increase for reasons mentioned in the prior section. By searching across local work-group size permutations, empirically optimal work-group dimensions can be found which maximize shared instructions and occupancy while minimizing thread divergence.

The current results for the Legendre matrix calculation are presented as speed-up ratios. These ratios are measured by averaging the time taken to submit and complete the Legendre matrix calculation on the GPU during a Basilisk scenario. The Basilisk scenario simulated thirty minutes of real-time orbit propagation, stepping with 10 second intervals, and using an RK4 integrator. The same simulation was performed on a CPU and the time to compute the Legendre matrix was averaged. The corresponding speed-up ratio is defined as the ratio between these two time average:

$$\tau = \frac{t_{\text{CPU}}}{t_{\text{GPU}}} \quad (\text{F.12})$$

The speed-up ratios for the AMD GPU are found in Figure F.5 and for the Intel GPU in Figure F.6.

Currently the GPU Legendre matrix algorithm demonstrates consistent  $> 1$  speed-up ratios

on the AMD card when there are sufficiently many spacecraft ( $> 256$ ) operating in a sufficiently high-fidelity gravity field ( $l = 128$ ). While speed-up ratios were occasionally realized in the lower-fidelity case ( $l = 32$ ), the ratios were consistently less performant than their high-fidelity counterpart. It is assumed that this is due to the relatively small dimension of the Legendre matrix in the low fidelity case. The computational overhead to transition the initial memory onto the GPU, dispatch threads, and return a result is sufficiently high on discrete GPUs such that there exists a minimum degree model that must be used before measurable speed-up ratios can be achieved.

The narrative differs when looking at the performance on the Intel GPU in Figure F.6. Here there are no GPU performance gains when using the high-fidelity models, but there appreciable speed-ups (as high as 40%) when using the lower-fidelity models. This appears to be an artifact of the shared DRAM of the integrated graphics card with the CPU. Discrete GPUs like the AMD card require that data be transferred from the CPU to the GPU explicitly. This is often an expensive process and best performed in a single, large data transfer. Integrated graphics processors do not have this limitation and can share memory share memory directly with the CPU resulting in much faster transfers. Consequently, the overhead for dispatching work to the Intel card is lower than the overhead associated with the discrete card, ultimately allowing observable speed-up ratios in the low-fidelity gravity field. Despite this, the integrated graphics card does suffer when evaluating high-fidelity models due to its lower clock-rate. This is seen through the narrowness of the speed-up window in Figures F.6a and F.6c. Given that the highest performance is observed when the work-group x-dimension is particularly small, the thread divergence of solving multiple columns in the Legendre matrix is significantly more costly to the integrated card than the discrete card.

Together, the two GPUs offer unique advantages that span most operational conditions. The discrete graphics card is able to model high-fidelity gravity fields with measurable speed-ups, and the integrated card is able to model low-fidelity gravity fields with measurable speed-ups. The one condition for which neither GPU succeeds is in the case of too few spacecraft. When modeling a single spacecraft in either a low- or high-fidelity gravity field, it is always faster to model on a CPU. Heuristically this is intuitive – despite the integrated card having a lower overhead cost than the

discrete card, an overhead still exists. The dimensions of the single satellite problem are sufficiently small that the overhead is always too large, regardless of GPU type.

## F.7 Conclusion

This work provides a discussion of general GPGPU best practices and attempts implementation of such practices to construct an alternative form of Pines algorithm to compute the accelerations generated by high-fidelity gravity fields. Such implementation decreases the algorithmic complexity from  $\mathcal{O}(n^2)$  on a CPU into  $\mathcal{O}(n)$  for the Legendre matrix computation and to  $\mathcal{O}(\log_2 n)$  for the data reduction computation on the GPU. Moreover the GPGPU implementation that has no functional limitations on GPU memory making it advantageous for modeling large satellite constellations. While implementation efforts are still ongoing, the intermediate results do match heuristic expectation and fully-realized speed-up ratios are expected in the near future.

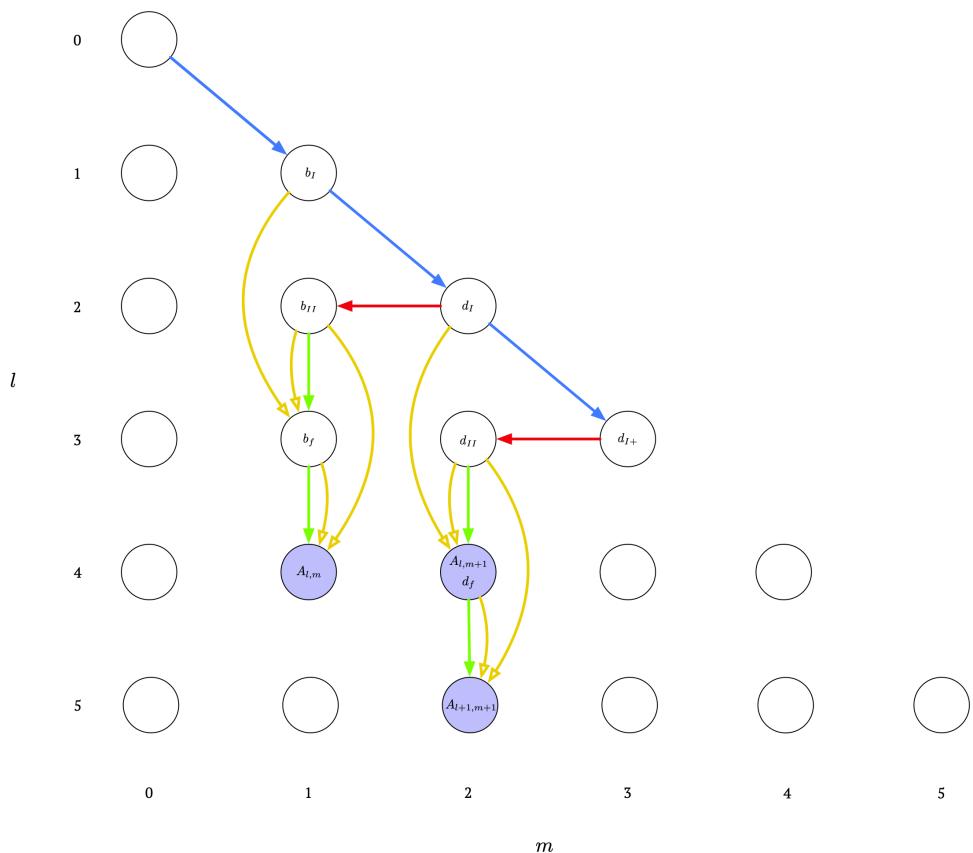


Figure F.4: Data Flow of GPU Kernel Invocation

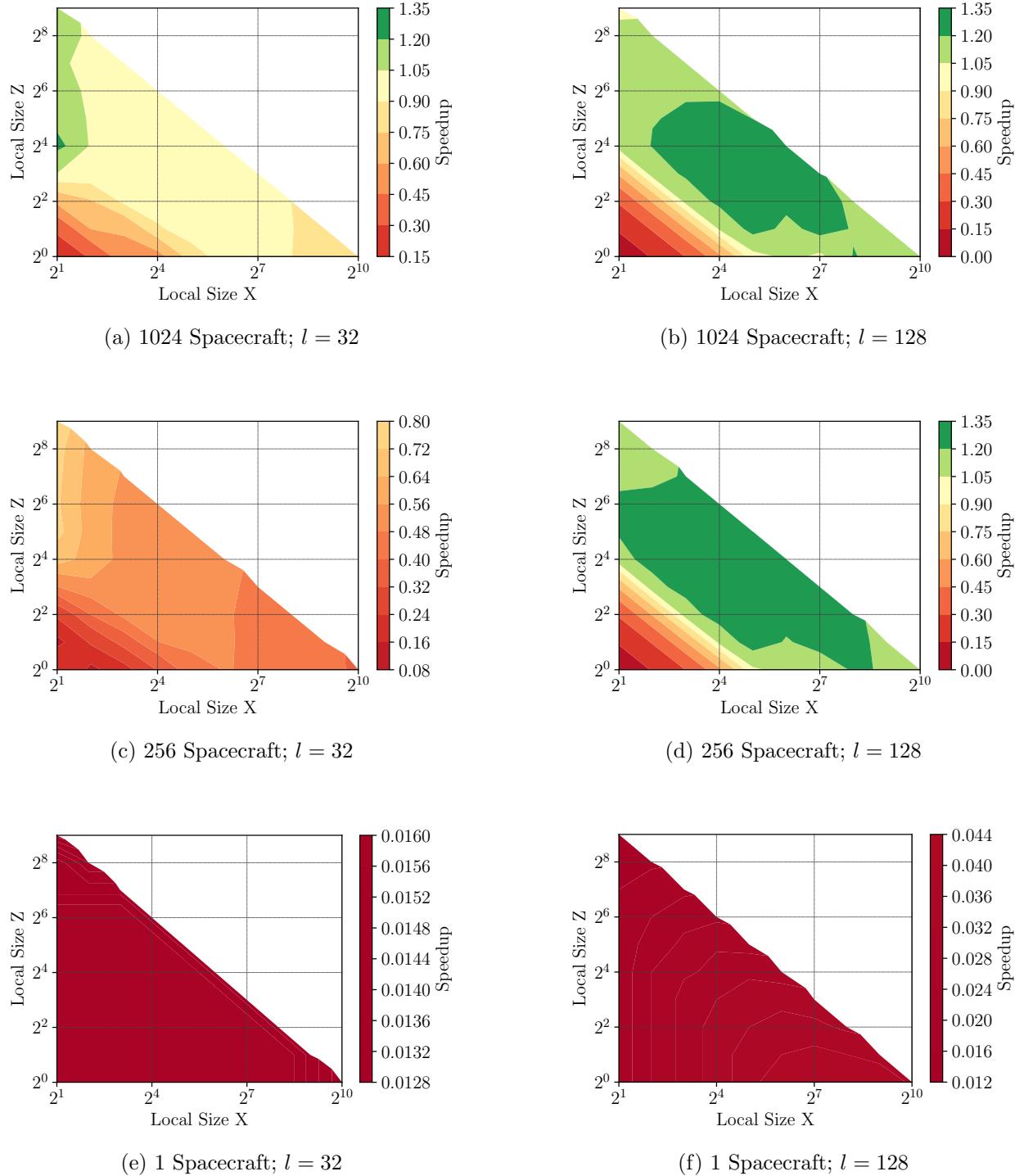


Figure F.5: Current speed up ratios for the Legendre matrix computation on AMD Radeon Pro 560 GPU using the compute bound method.

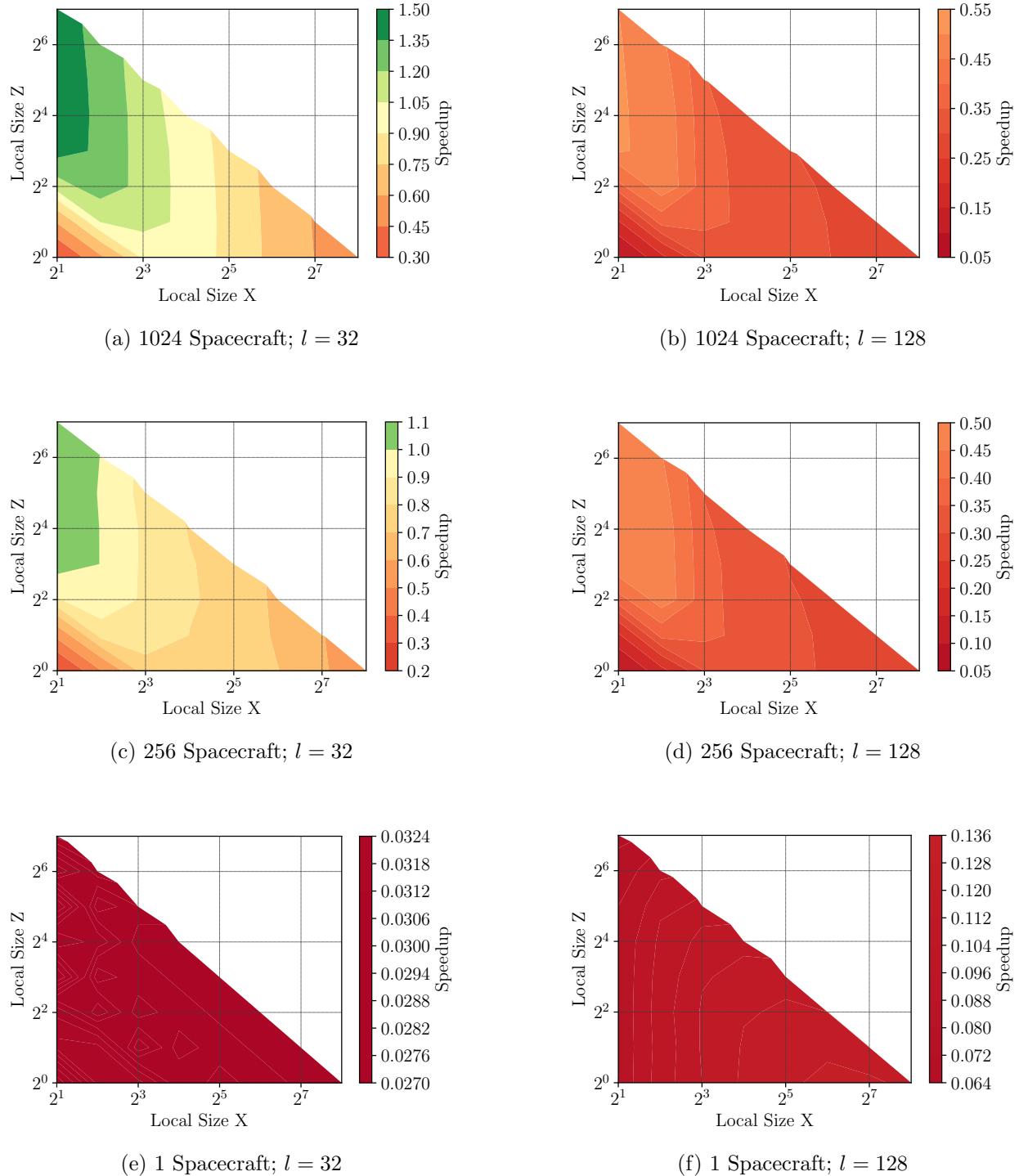


Figure F.6: Current speed up ratios for the Legendre matrix computation on Intel HD Graphics 630 GPU using the compute bound method.