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.) manuever. 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 manuever 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, γ) 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 γ is the discount factor. Reinforcement learning seeks to learn a policy $\pi : \mathcal{S} \rightarrow \mathcal{A}$ which maximizes the expected return

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

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

$$\boldsymbol{s} = (\bar{\boldsymbol{r}}, \bar{\boldsymbol{v}}, \bar{m}_f) \tag{4.2}$$

where $\bar{\boldsymbol{r}}$ 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 $\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}}, \hat{\boldsymbol{z}}$ respectively. Likewise $\bar{\boldsymbol{v}}$ 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:

$$v_{\text{norm}} = \sqrt{2\frac{\mu}{R}} \tag{4.3}$$

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 ΔVs that can be applied instantaneously at the beginning of every simulation step. The magnitude of ΔV is constrained to ± 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

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

where failure is defined as:

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

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 ΔV generated by the action is applied instantaneously. The discount factor γ 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_ψ. A second function approximator forms the actor, or policy π_ϕ, 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

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

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 ψ and θ are the trainable weights and biases of the network. π_ϕ 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:

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

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

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

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:

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

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

$$\tag{4.12}$$

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

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

where ϵ_t is a noise vector. This allows the cost function to be rewritten as

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

and solved via

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

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

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

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 δV impulses. While not representative of the true system

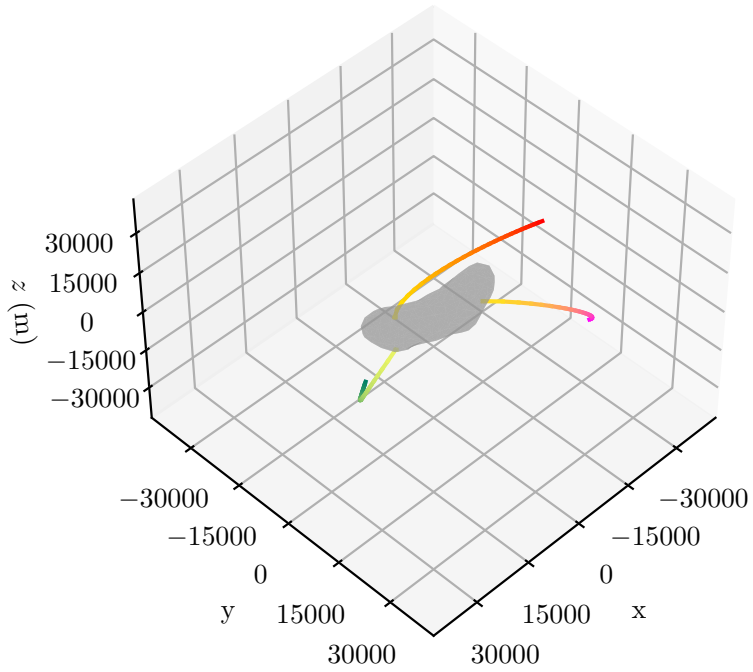


Figure 4.1: Trajectories taken **without** the Enhanced Safe Mode agent enabled. The darker shades correspond with earlier parts of the trajectory.

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 ΔV 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

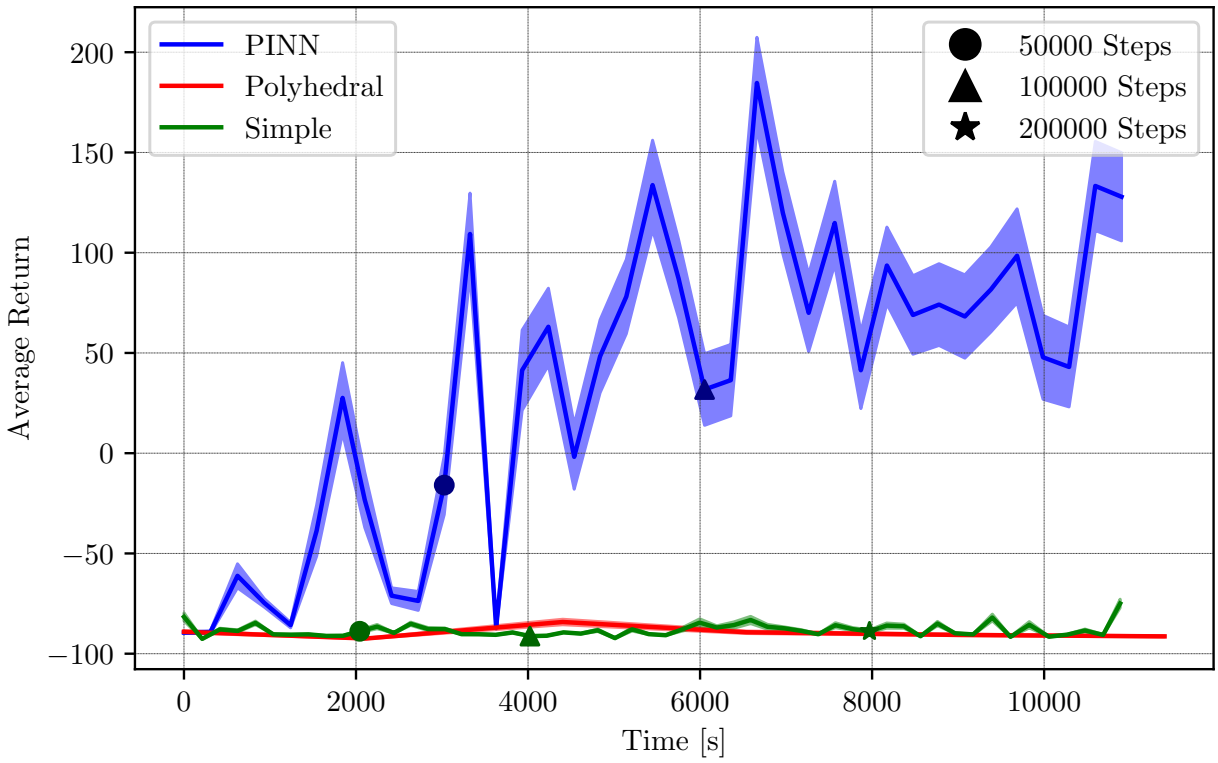


Figure 4.2: Average return as a function of clock time

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.

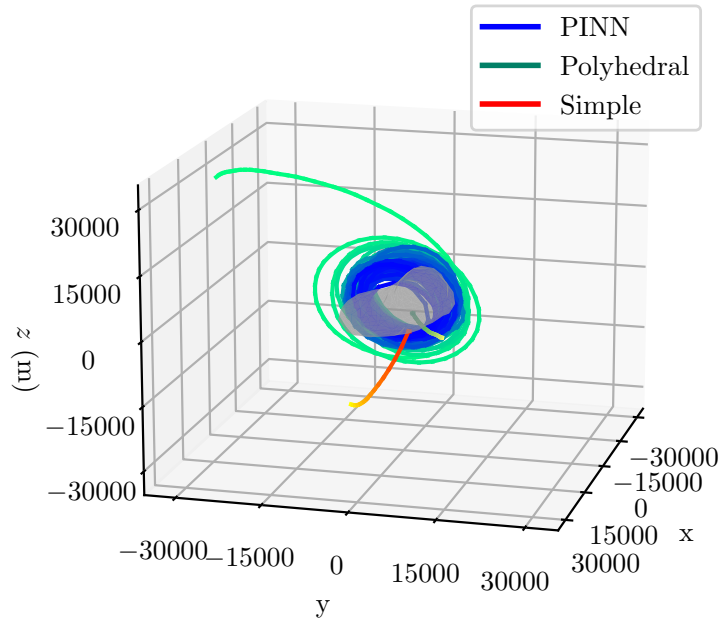


Figure 4.3: Trajectories taken **with** Enhanced Safe Mode agent enabled.

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.

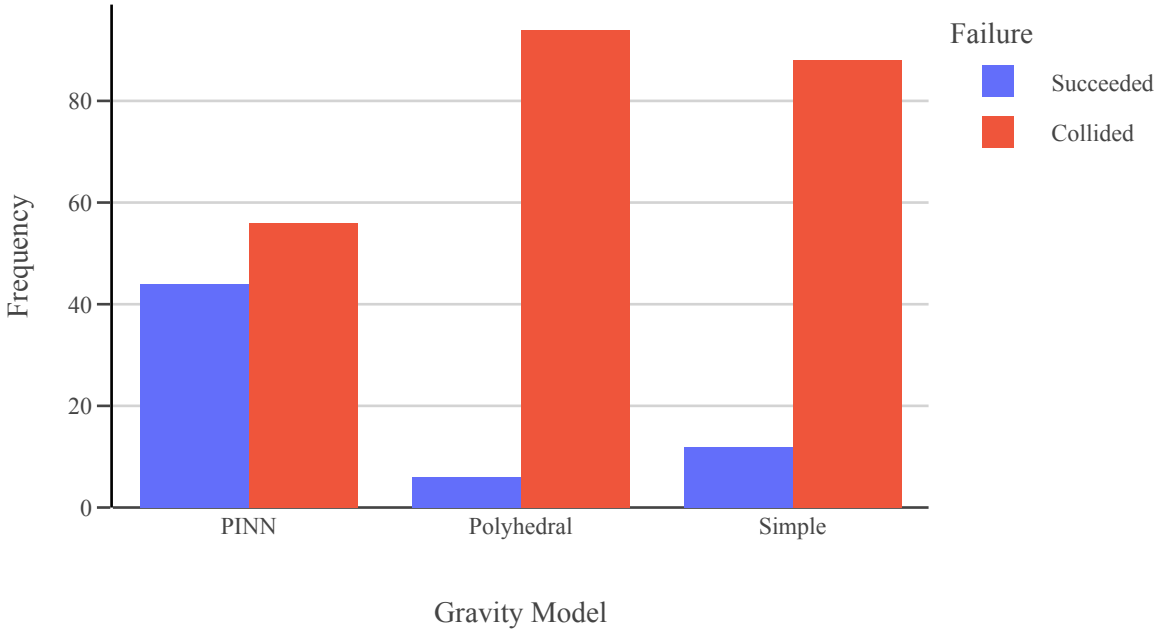


Figure 4.4: Success rates of the agents trained in different environments

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-

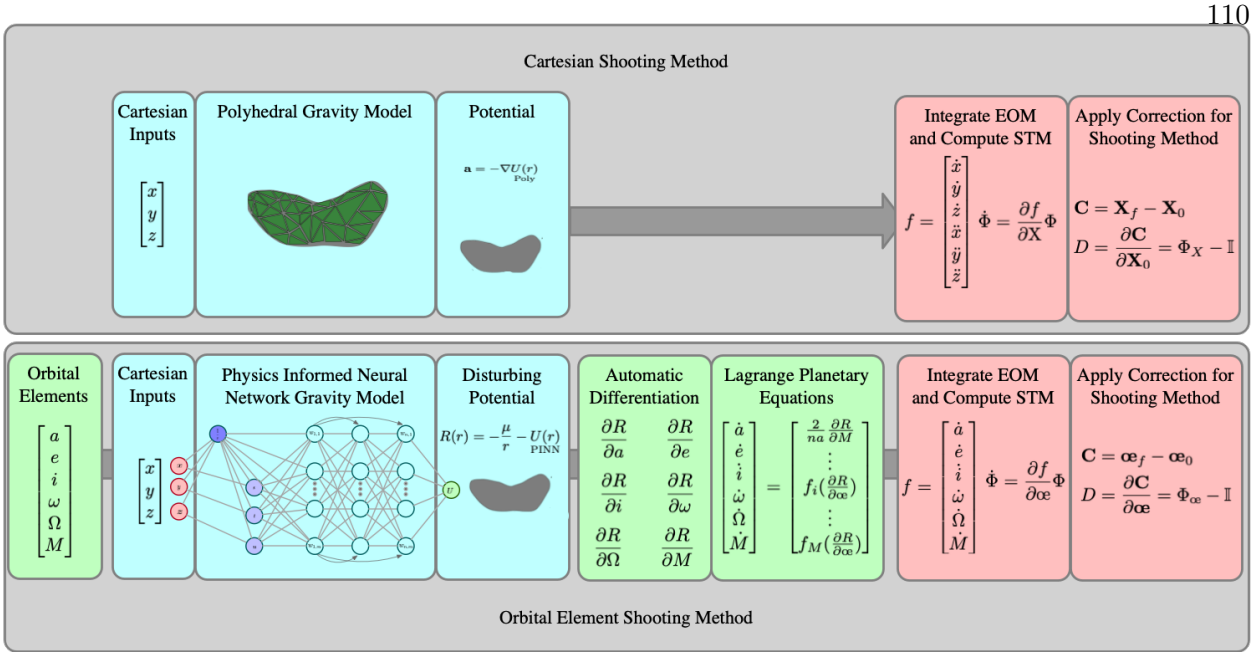


Figure 5.1: Outline of the former cartesian shooting method (top) and the novel orbital element shooting method (bottom).

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, $\boldsymbol{X}(t_0)$ or $\boldsymbol{X}_0$, and the state after some period T, $\boldsymbol{X}(t_f)$ or $\boldsymbol{X}_f$:

$$\boldsymbol{X}_f - \boldsymbol{X}_0 = \boldsymbol{0} \tag{5.1}$$

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

$$\dot{\boldsymbol{X}} = \begin{bmatrix} \boldsymbol{v} \\ \boldsymbol{a} \\ 0 \end{bmatrix} \tag{5.2}$$

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.

$$C(\boldsymbol{X}_{\boldsymbol{0}}^{\star}) = C(\boldsymbol{X}_0 + \delta\boldsymbol{X}_0) \tag{5.3}$$

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

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

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

where

$$\frac{\partial C}{\partial \boldsymbol{X}_0} = \frac{\partial}{\partial \boldsymbol{X}_0}(\boldsymbol{X}_f - \boldsymbol{X}_0) \tag{5.6}$$

$$= \left[\frac{\partial \boldsymbol{X}_f}{\partial \boldsymbol{X}_0} - \frac{\partial \boldsymbol{X}_0}{\partial \boldsymbol{X}_0}, \quad \frac{\partial \boldsymbol{X}_f}{\partial T}\right] \tag{5.7}$$

$$= \left[\Phi(t_f, t_0) - \mathbb{I}, \quad \frac{\partial \boldsymbol{X}_f}{\partial T}\right] \tag{5.8}$$

and Φ 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 $|\delta\boldsymbol{X}_0| < \epsilon$. This numerical procedure will converge for initial guesses of $\boldsymbol{X}_0$ that are sufficiently close to the true periodic solution, $\boldsymbol{X}_0^{\star}$, assuming that a reliable gravity model exists which can provide accurate values of the acceleration vector, $\boldsymbol{a}$, 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, $\boldsymbol{r}$. 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, ω, Ω 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, $\dot{\text{oe}}$, and the jacobian of these derivatives, $\frac{\partial \dot{\text{oe}}}{\partial \text{oe}}$, are required, but notoriously difficult to compute. The time derivatives of the orbital elements can technically be computed through the Lagrange Planetary Equations (LPE):

$$\frac{\partial a}{\partial t} = \frac{2}{na}\frac{\partial R}{\partial M} \tag{5.9}$$

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

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

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

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

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

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

$$R(\boldsymbol{r}) = -\frac{\mu}{r} - U(\boldsymbol{r}) \tag{5.15}$$

where U is the total gravitational potential and μ 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 $\partial\dot{\text{œ}}/\partial\text{œ}$ 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, $\frac{\partial \hat{R}}{\partial \alpha}$, 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

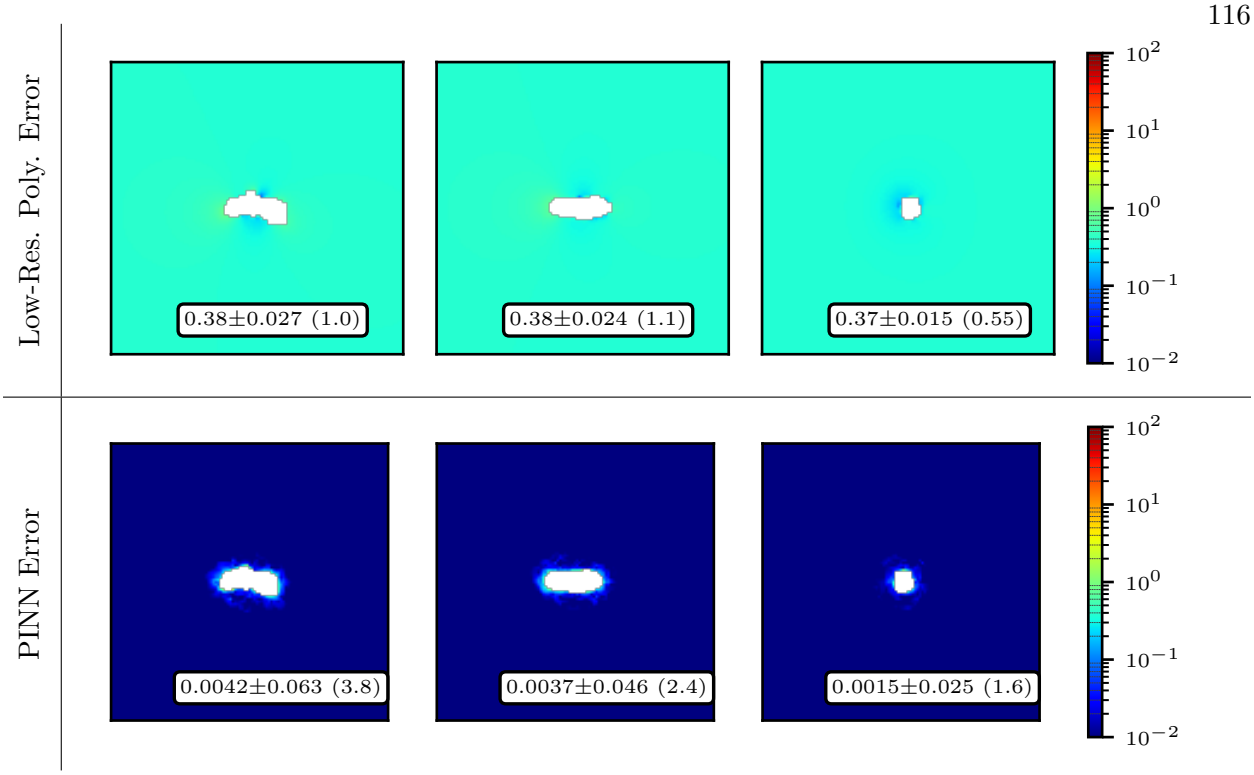


Figure 5.2: Percent error of the acceleration vector for each gravity model

Table 5.1: Initial Orbital Element Distribution

Parameter	**Value**
Semi-major axis	$\mathcal{U}[3R_{\text{Eros}}, 7R_{\text{Eros}}]$
Eccentricity	$\mathcal{U}[0.1, 0.3]$
Inclination	$\mathcal{U}[-\frac{\pi}{2}, \frac{\pi}{2}]$

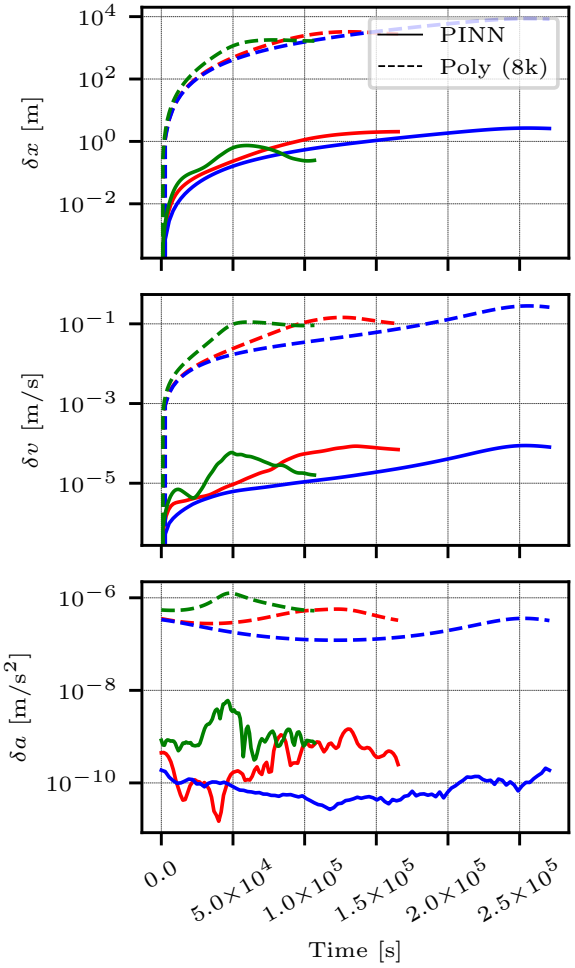
Parameter	**Value**
Argument of periapsis	$\mathcal{U}[0, 2\pi]$
Longitude of the Ascending Node	$\mathcal{U}[0, 2\pi]$
Mean Anomaly	$\mathcal{U}[0, 2\pi]$

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

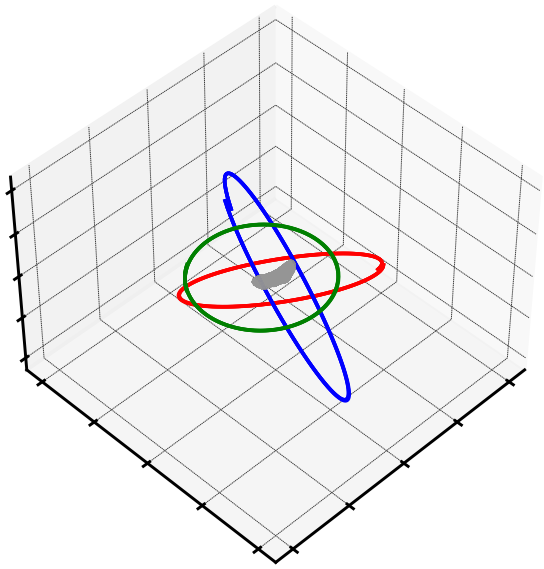
$$T = 2\pi\sqrt{\frac{a^3}{\mu}} \tag{5.16}$$

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



(a) State error of PINN vs 8k Polyhedral



(b) Integrated PINN and Polyhedral Orbits

Orbit	**Model**	$\boldsymbol{\delta r}$ [m]	$\boldsymbol{\delta v}$ [m/s]	$\boldsymbol{T}$ [s]	**Speedup** (T_{200k}/T_{model})
Red	Poly 200	0.0	0.0e+00	481.7	1.0
	Poly 8k	2741.5	1.0e-01	18.8	25.6
	PINN	2.1	7.0e-05	4.8	99.5
Blue	Poly 200	0.0	0.0e+00	471.8	1.0
	Poly 8k	8307.2	2.6e-01	17.8	26.6
	PINN	2.6	8.1E-06	1.2	378.1
Green	Poly 200	0.0	0.0e+00	492.4	1.0
	Poly 8k	1658.6	9.2e-02	18.5	26.6
	PINN	0.2	1.6E-05	1.2	422.4

(c) Metrics comparing the PINN and Polyhedral Integrated Orbits

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

$$A = \frac{\partial \dot{\boldsymbol{X}}}{\partial \boldsymbol{X}} \tag{5.17}$$

to then propagate the STM through

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

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, $|\delta \boldsymbol{X}_0|$, is less than $\epsilon_x(\epsilon_x + |\boldsymbol{X}_0|)$ where $\epsilon_x = 10^{-8}$.

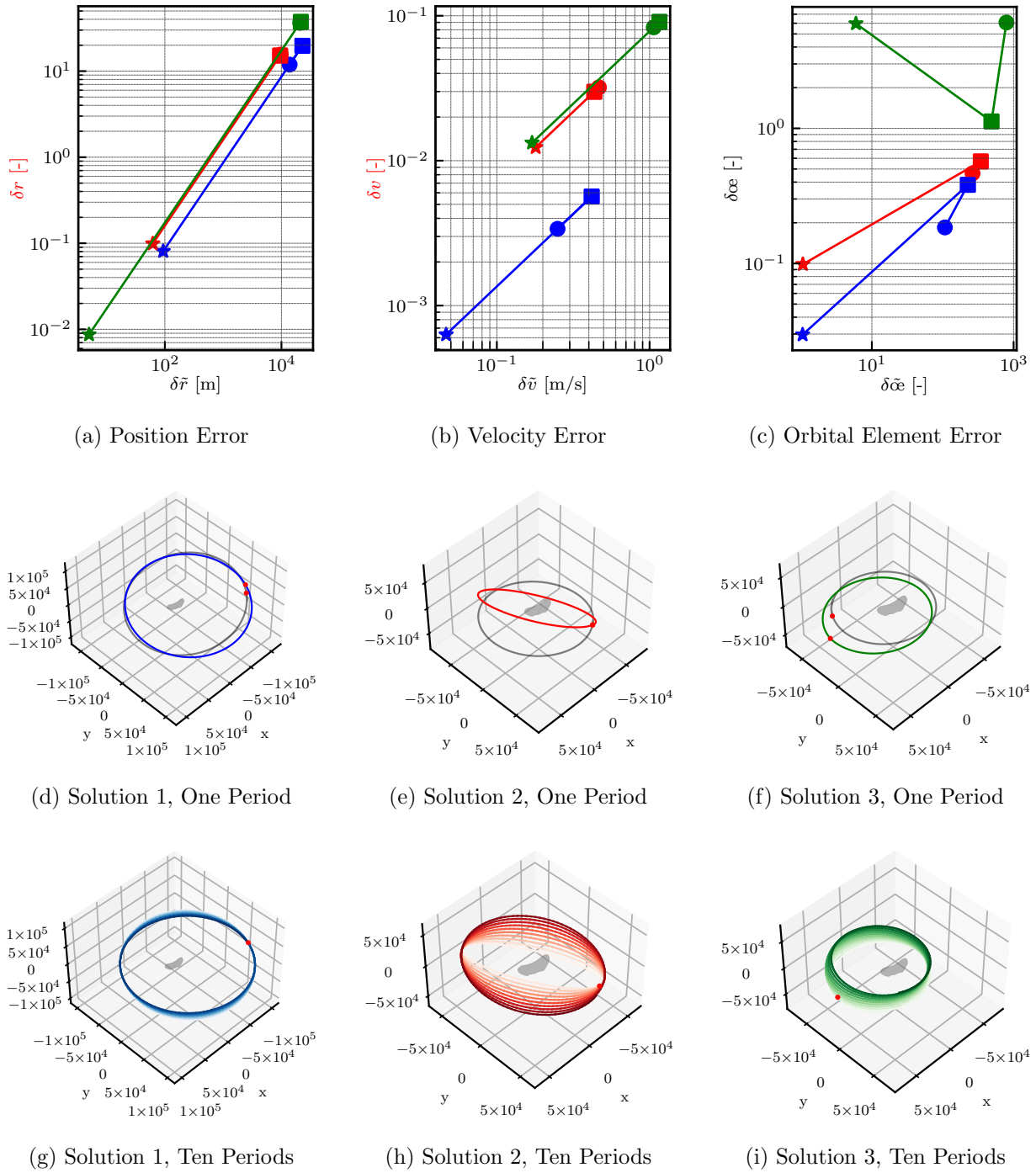
(2) The normalized change in the cost function, $\delta F/F$, is less than ϵ_f where $\epsilon_f = 10^{-8}$.

(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 $l^\star = |\boldsymbol{r_0}|$ and normalize time by the initial period $t^\star = 2\pi\sqrt{a_0^3/\mu}$. 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 $l^\star = |\boldsymbol{r_0}|$, but instead scale the velocity such that $t^\star = l^\star/|\boldsymbol{v}|$. 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.



(a) Position Error (b) Velocity Error (c) Orbital Element Error

(d) Solution 1, One Period (e) Solution 2, One Period (f) Solution 3, One Period

(g) Solution 1, Ten Periods (h) Solution 2, Ten Periods (i) Solution 3, Ten Periods

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 $\boldsymbol{X}_f$ and $\boldsymbol{X}_0$ in both dimensionalized ($\tilde{\boldsymbol{r}}, \tilde{\boldsymbol{v}}$, œ) and non-dimensionalized ($\boldsymbol{r}, \boldsymbol{v}$, œ) 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 δr and δv 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, ω, Ω, 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, ω, Ω and M are all scaled by 2π. In addition, the δi, $\delta\omega$, $\delta\Omega$, and δM 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

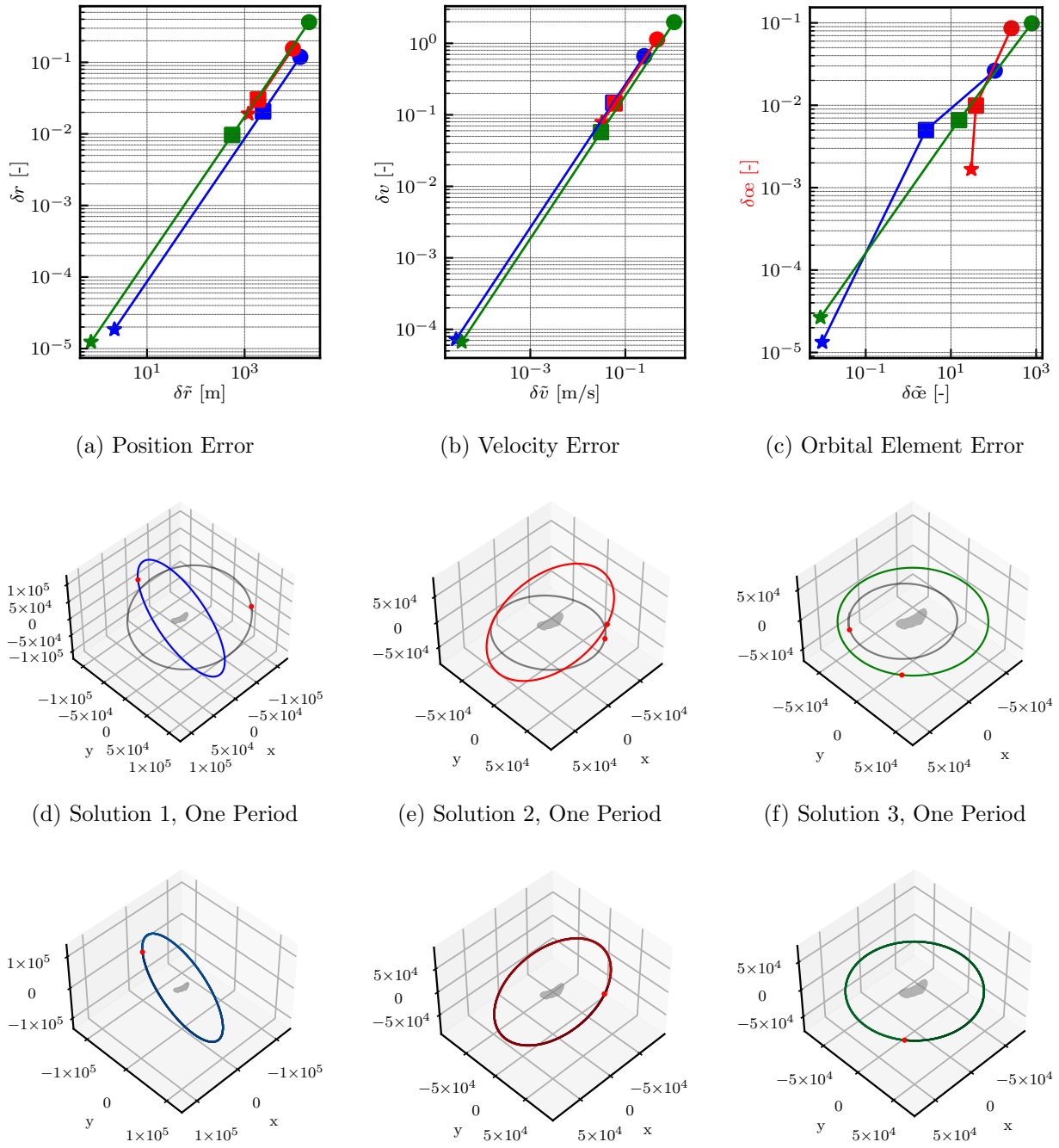
For the experiment, the same three initial conditions are used but are instead propagated and corrected in orbital element space. All elements remain unconstrained in the minimization, and the corresponding improvement in orbital element state error (both dimensionalized and non-dimensionalized) are 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 δr, δv, and $\delta \text{œ}$ 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 $\boldsymbol{r}_0$ and $\boldsymbol{r}_f$ 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 form the optimization set ($\delta \text{œ}' = \delta \text{œ}_0 \ / \ \{\delta a_0, \delta i_0\}$). 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,

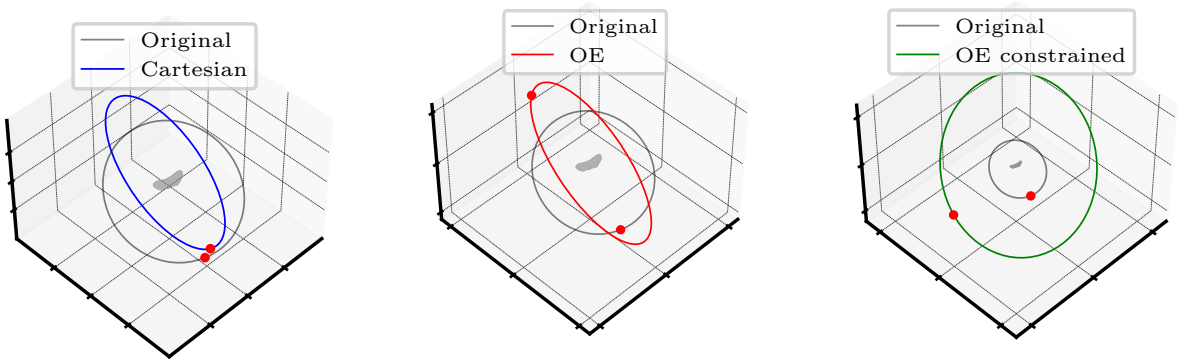


(a) Position Error (b) Velocity Error (c) Orbital Element Error

(d) Solution 1, One Period (e) Solution 2, One Period (f) Solution 3, One Period

(g) Solution 1, Ten Periods (h) Solution 2, Ten Periods (i) Solution e, Ten Periods

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.



(a) Cartesian Solution (b) OE Solution (c) OE Constrained Solution

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	**a**$[m]$	**e**	**i** [rad]	ω [rad]	$\boldsymbol{\Omega}$ [rad]	**M** [rad]	**T** [s]	$\lvert d\boldsymbol{X}\rvert[m]$
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 $|d\boldsymbol{X}|$, 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.

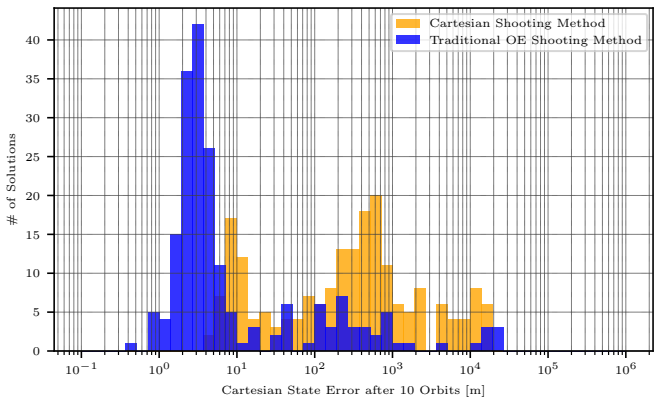


Figure 5.7: Average Percent Error of Solutions using the Cartesian LPE and OE LPE

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.

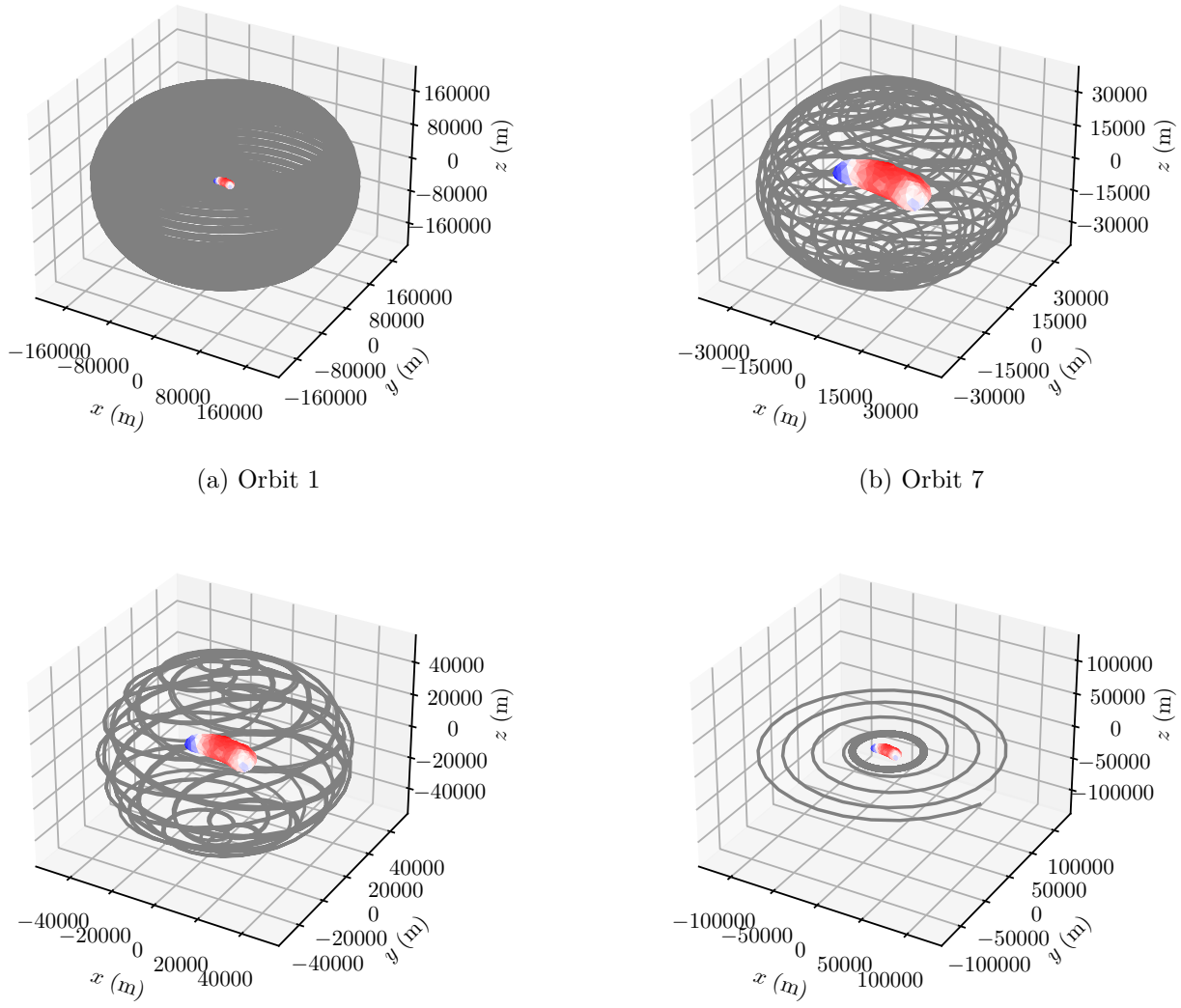
As discussed throughout, the Physics-Informed Neural Network gravity model (PINN-GM) offers a compelling alternative to representing the gravity field of small-bodies. By learning, rather than prescribing, basis functions, the PINN-GM is able to produce high-fidelity gravity models without making any 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.



(a) Orbit 1

(b) Orbit 7

(c) Orbit 9

(d) Orbit 19

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.

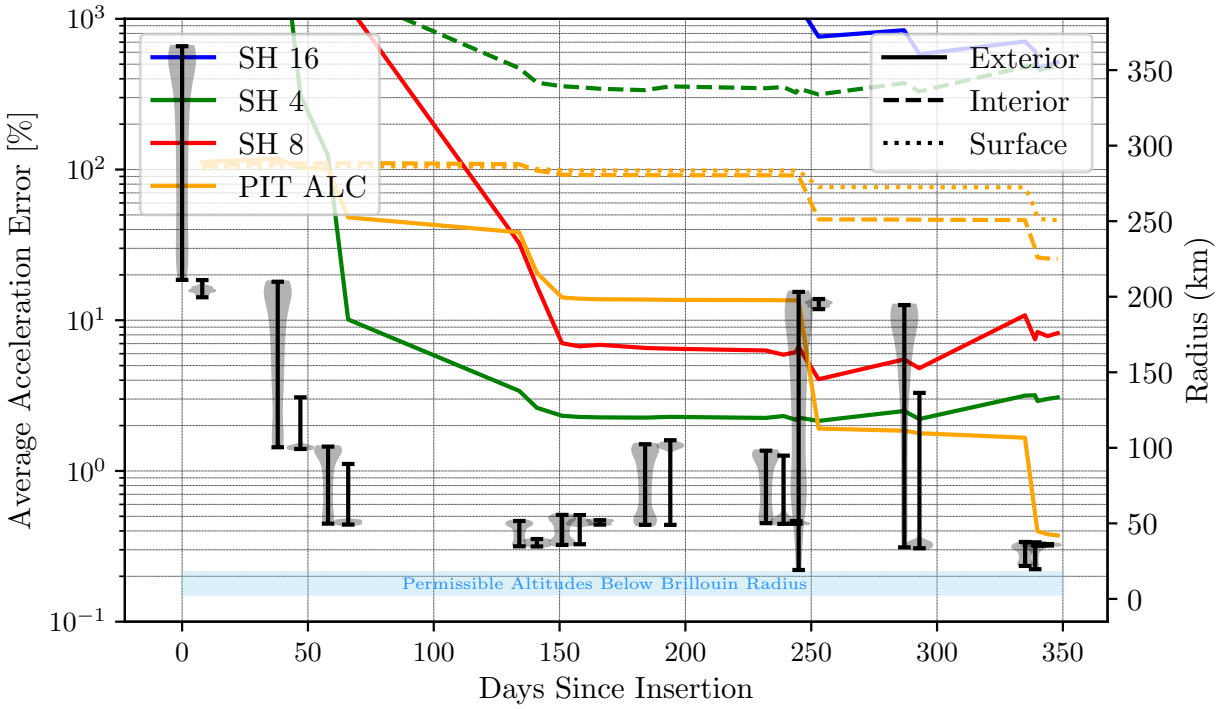


Figure 6.2: Error of Regressed PINN-GM Gravity Model of Eros without Ejecta

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

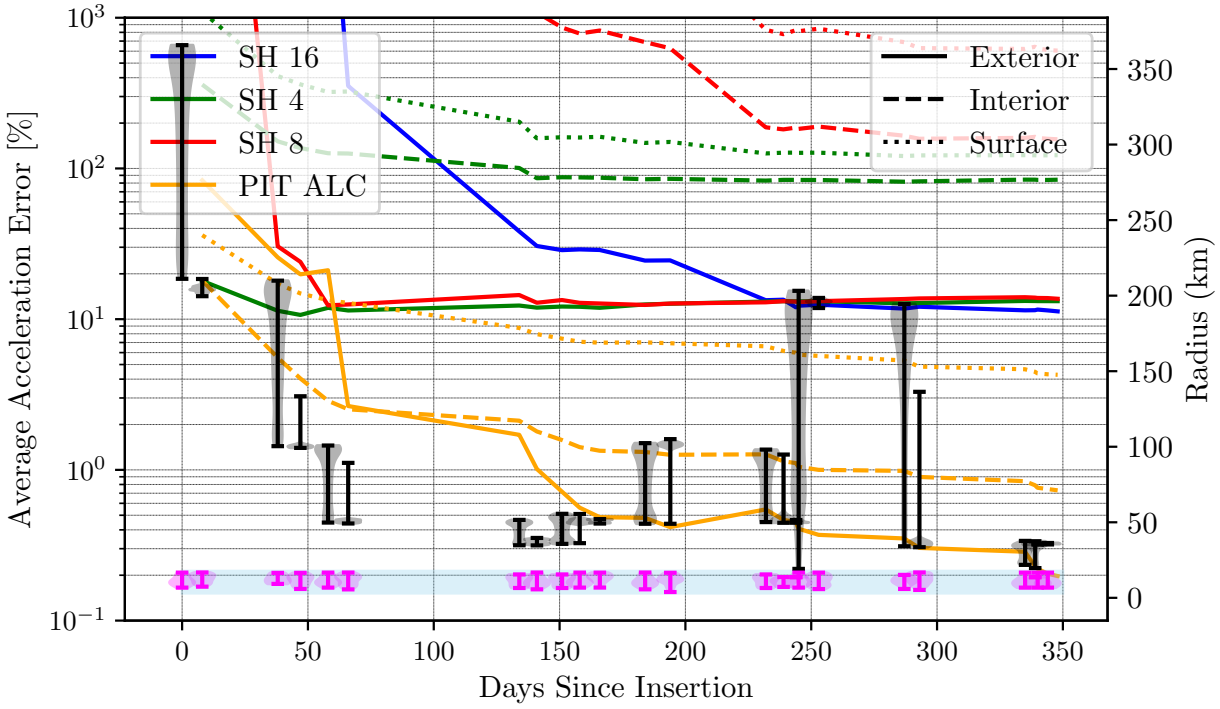


Figure 6.3: Error of Regressed PINN-GM Gravity Model of Eros with Ejecta Measurements (Magenta)

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 convienence:

(1) Initialize the filter with an initial reference state $\boldsymbol{x}_0$, the initial state deviation / error $\Delta\boldsymbol{x}_0$, the initial state covariance P_0, process noise covariance matrix Q, measurement noise covariance matrix R.

(2) Obtain a measurement $\boldsymbol{y}_i$

(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

$$\dot{\boldsymbol{x}} = \left[\dot{x}, \dot{y}, \dot{z}, \ddot{x}, \ddot{y}, \ddot{z}\right]^T, \quad \dot{\Phi} = \frac{\partial \dot{\boldsymbol{x}}}{\partial \boldsymbol{x}}\Phi \tag{6.1}$$

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:

$$\Delta x_i^- = \Phi \Delta x_{i-1}^+ \tag{6.2}$$

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

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

surement through:

$$r_i = y_i - \hat{y}_i \tag{6.4}$$

$$H_i = \frac{\partial h}{\partial x}\Big|_{x_i} \tag{6.5}$$

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

$$\Delta x_i^+ = \Delta x_i^- + K_i(r_i - H_i \Delta x_i^-) \tag{6.7}$$

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

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:

$$\dot{\mathbf{w}} = -\frac{1}{\tau}\mathbf{w} \tag{6.9}$$

where $\mathbf{w}$ is the noise vector, and τ 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,

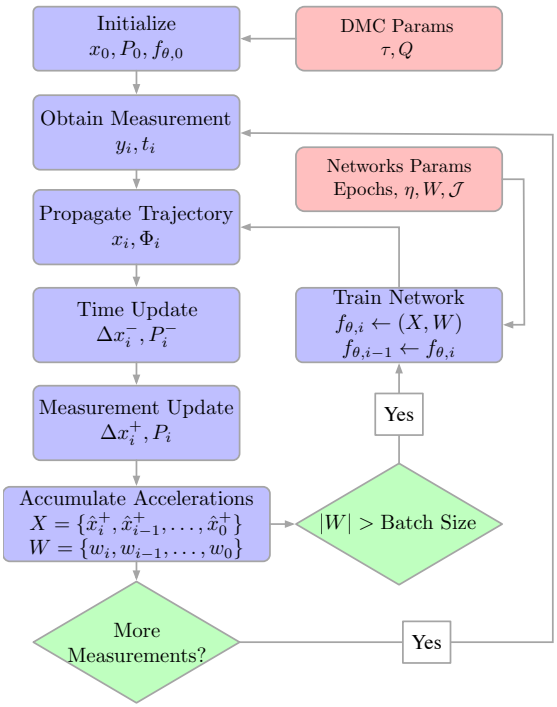


Figure 6.4: PINN-GM Kalman Filter

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 $\bar{\boldsymbol{x}}$, velocity $\bar{\boldsymbol{v}}$, and any accelerations not captured by the current dynamics model $\mathbf{w} = \delta\bar{\boldsymbol{a}}$. Consequently, these residuals in the dynamics can then be added to the PINN-GM's current estimate of the acceleration at estimated position, $\boldsymbol{a}(\bar{\boldsymbol{x}})$, to produce the best estimate of the true acceleration $\boldsymbol{a}$ through

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

Together, the estimated position $\bar{\boldsymbol{x}}$ and corrected acceleration vector $\boldsymbol{a}(\bar{\boldsymbol{x}})$ 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 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.

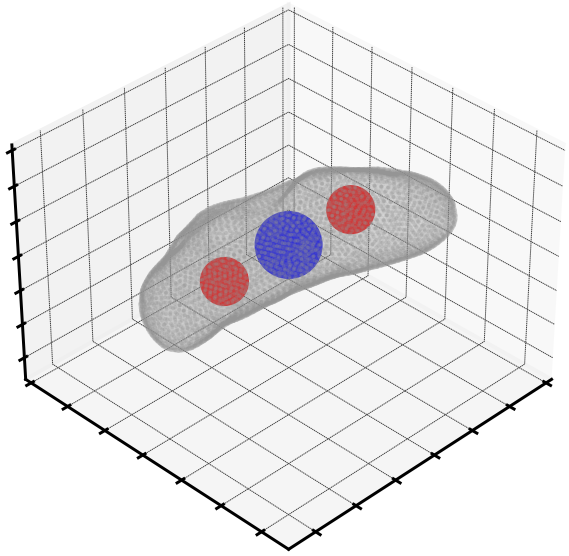


Figure 6.5: Heterogeneities of Asteroid

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 m^3/s^2 and rotates with a frequency of 3.318E-4 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.

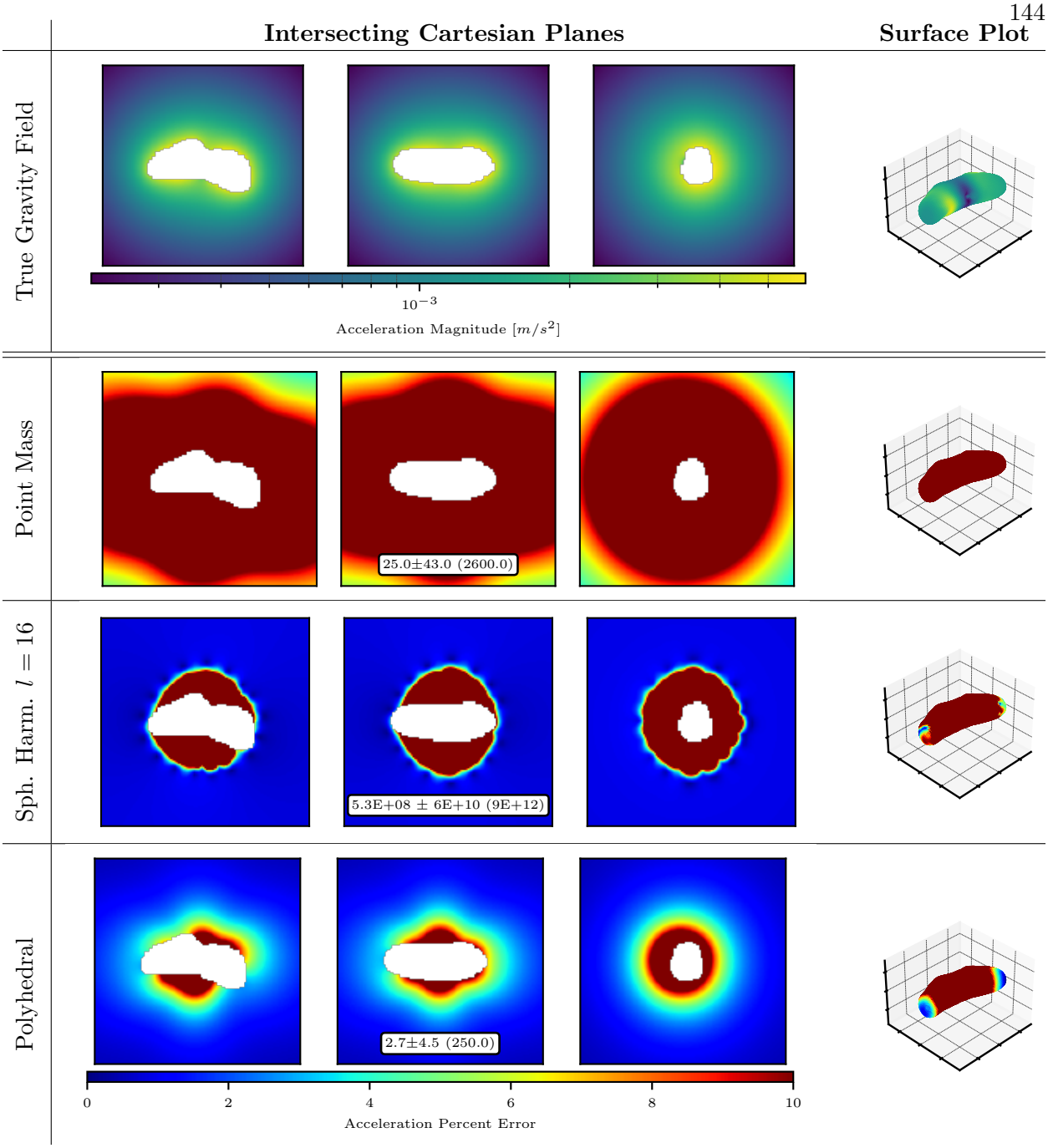
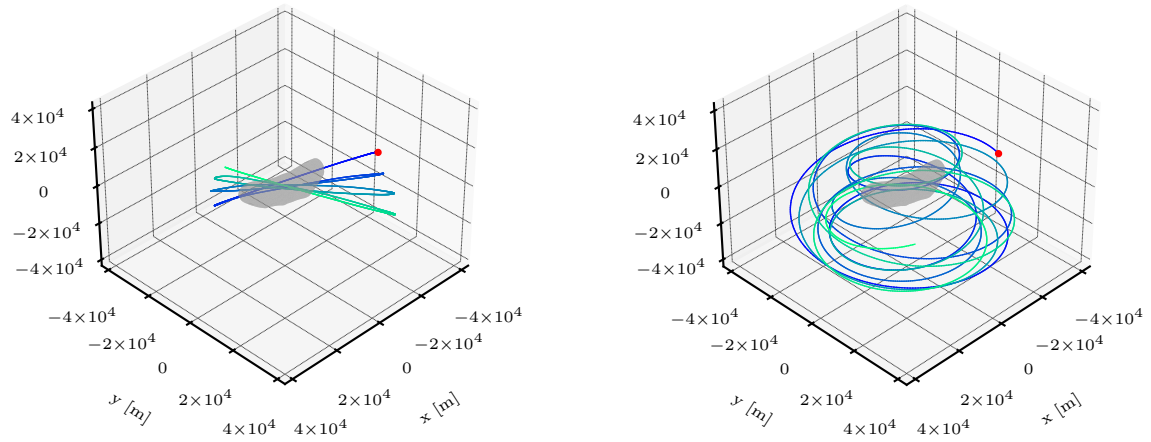


Figure 6.6: Top Row: The true heterogeneous density asteroid gravity field accelerations assessed along cartesian planes from $[-2R, 2R]$. Bottom Rows: Acceleration percent error for different gravity models capped at 10% error.

Table 6.1: Initial orbital elements

a [m]	**e** [-]	**i** [deg]	ω [deg]	$\boldsymbol{\Omega}$ [deg]	**M** [deg]
34,000	0.35	45	48.2	347.8	85.3



(a) Inertial Frame (b) Body Frame

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

$$P = \frac{1}{N} \sum_{i=1}^{N} \frac{|\boldsymbol{a}_{\text{true}} - \boldsymbol{a}_{\text{PINN}}|}{|\boldsymbol{a}_{\text{true}}|} \tag{6.11}$$

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^2]
P_{x_0}	diag([100, 100, 100])	[m^2]
P_{v_0}	diag([0.01, 0.01, 0.01])	[m^2/s^2]
P_{w_0}	diag([1E-14 , 1E-14, 1E-14])	[m^2/s^4]
Q	diag([1E-14, 1E-14, 1E-14])	[m^2/s^2]
R	diag([1E-3, 1E-3, 1E-3])	[m^2]

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

$$\mathcal{S} = \sum_{i=0}^{4} \int_{t=0}^{T} \Delta \boldsymbol{X}_i(t) \mathrm{d}t \approx \sum_{i=0}^{4} \sum_{j=0}^{N} \Delta \boldsymbol{X}_i(t_j) \tag{6.12}$$

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	**a** [m]	**e** [-]	**i** [deg]	ω [deg]	**Ω** [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 [1] .

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

[1] 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 $r \to \infty$

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.

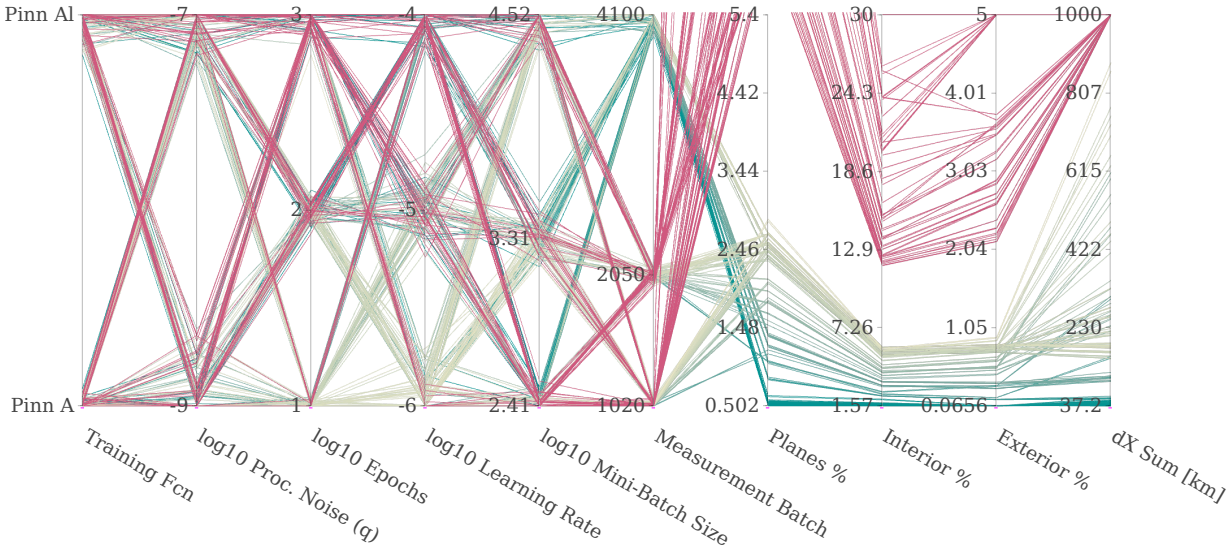
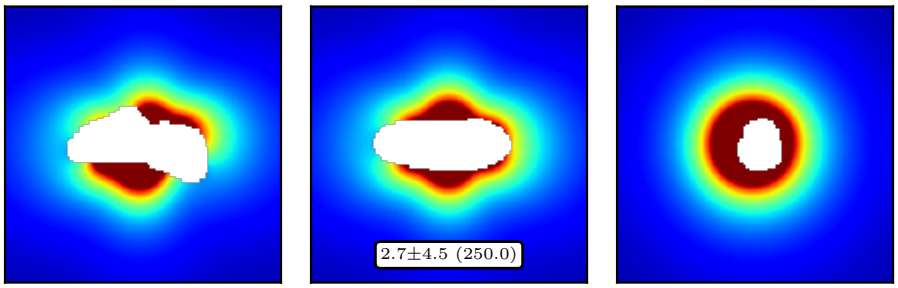
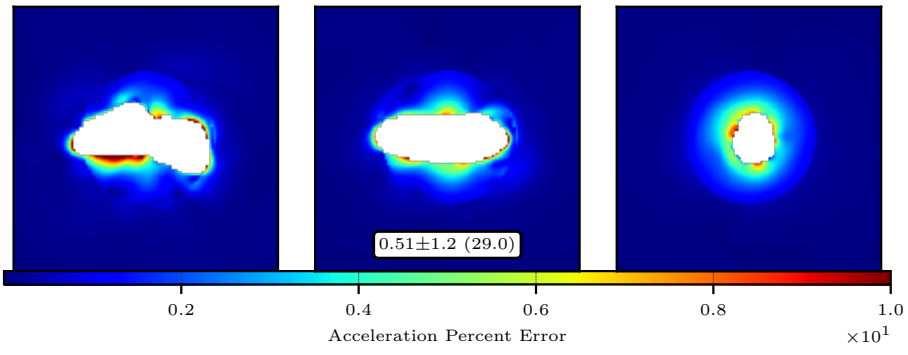


Figure 6.8: Hyperparameter search of the PINN-GM initialized with the constant density polyhedral model.



(a) PINN-GM before training.



(b) PINN-GM after training.

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

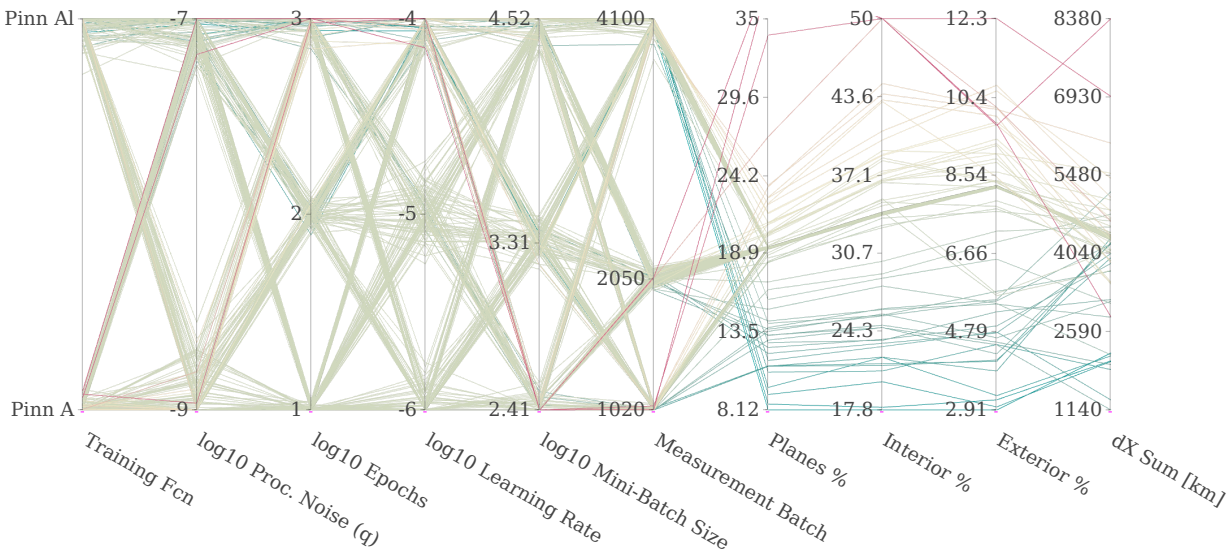
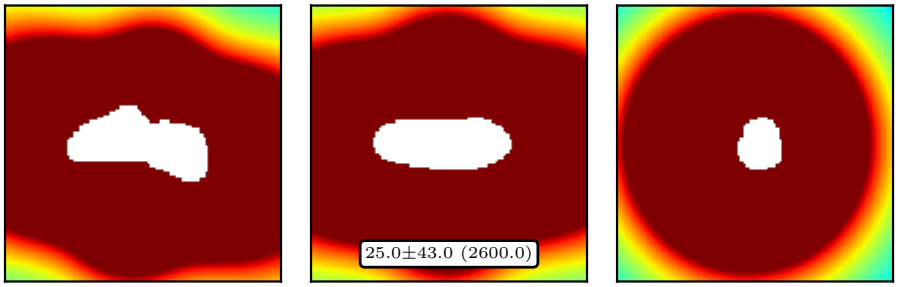
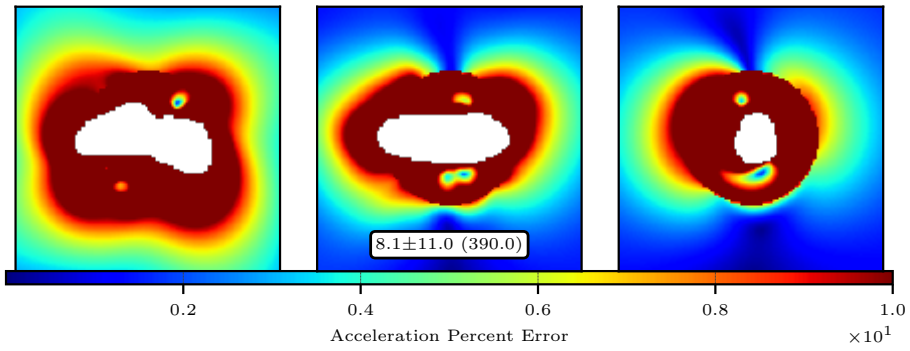


Figure 6.10: Hyperparameter search of the PINN-GM initialized with the point mass model.



(a) PINN-GM before training.



(b) PINN-GM after training.

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.

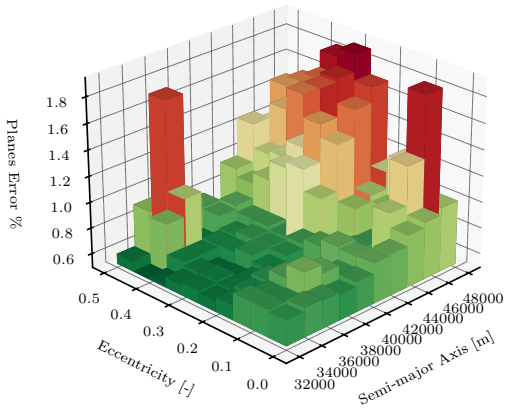
These results are particularly encouraging given the constraints placed on the experiment. These hyperparameter searches are conducted over only three orbits with measurements of the position taken once every minute. This totals to less than 3,000 data points, and yet even given these relatively sparse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

6.6.2 Sensitivity to Orbit Geometry

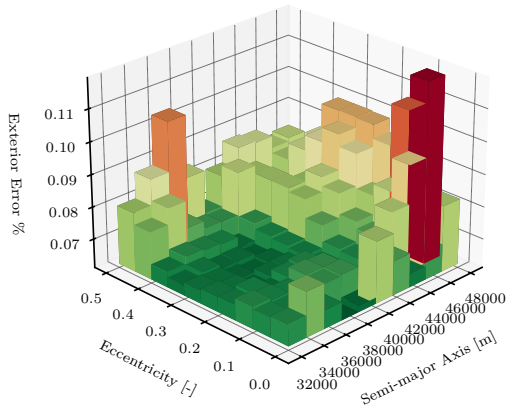
The final experiment studies how the learned PINN-GM performance is effected by the initial orbit geometry. In particular, this study investigates how sensitive the learned PINN-GM solution is to the altitude and eccentricity of the initial orbit. Explicitly, the spacecraft's initial semi-major is varied 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-

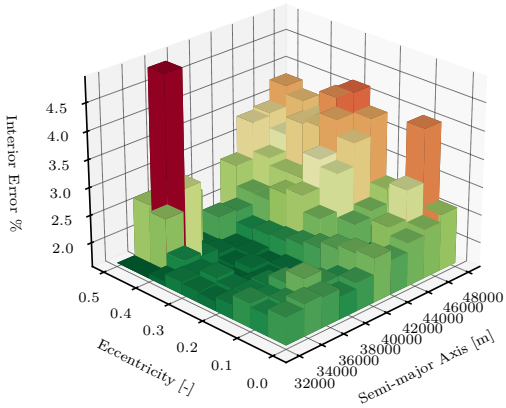
ter than in near circular regimes. This is attributed the fact that the radial component of the acceleration vector experiences the greatest variability and observability of all the components. Consequently, when orbiting at different radii, the PINN-GM is given a much more diverse set of training from which in 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.



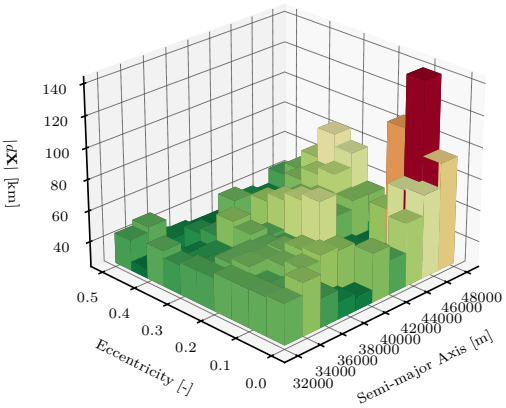
(a) Planes Metric



(b) Exterior Metric



(c) Interior Metric



(d) Trajectory Metric

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 astrodynamicists.

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 ISSIONTO 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. Brovvnd, 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," Astrodynamics, 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. Cvitanovi'c, "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. Pirkl, "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 $-\nabla U - \boldsymbol{a} = 0$. 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 θ Init.	Glorot Normal
Loss Function	Percent	Final Layer θ 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) $-\nabla U = \boldsymbol{a}$, (2) $\nabla^2 U = 0$, and (3) $\nabla \times \nabla U = \mathbf{0}$ through:

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

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 $-\nabla\hat{U}(x_i|\theta) - \boldsymbol{a} = 0$ 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[1] .

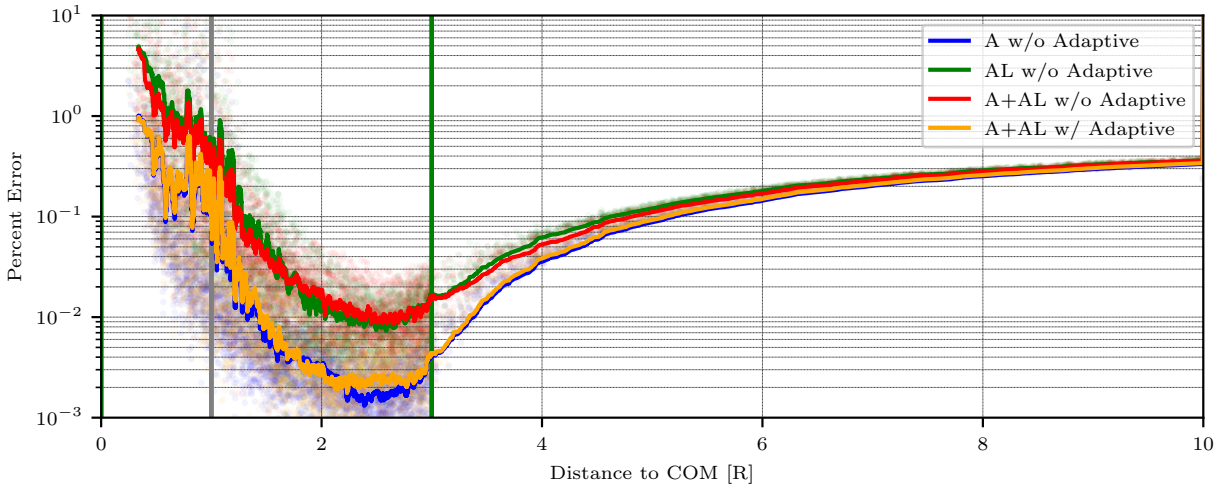
PINN-GM-III also leverages an adaptive learning rate annealing algorithm to change the weights of the loss components during training to minimize competing objectives. The algorithm takes inspiration from a theoretical analysis of the learning dynamics of neural networks using Neural Tangent Kernels 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 $-\nabla U = \boldsymbol{a}$ constraint for a total of 20,000 epochs. Second, a PINN-GM-III is trained with both the $-\nabla U = \boldsymbol{a}$ 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 $-\nabla U = \boldsymbol{a}$ 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.

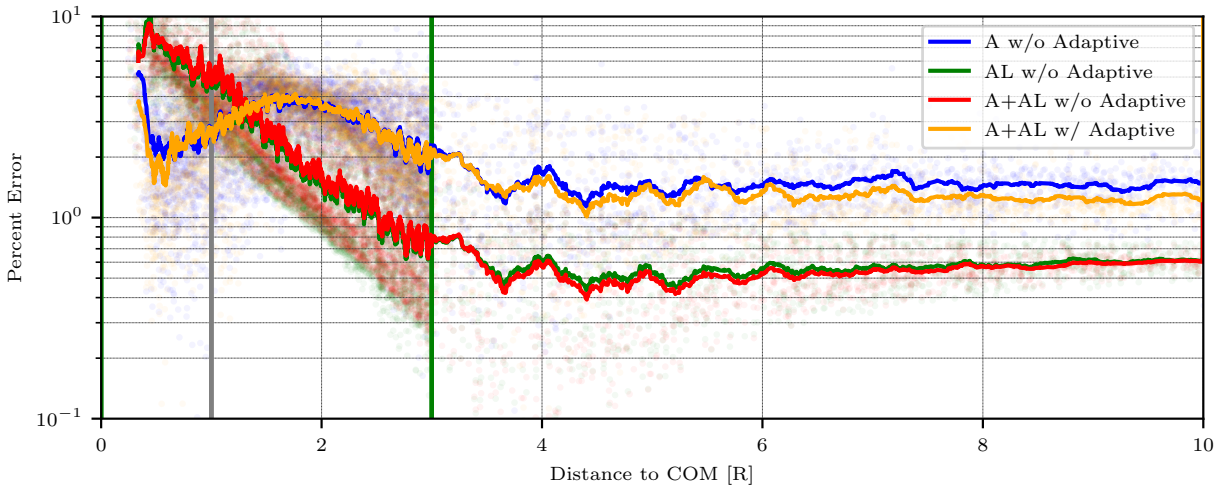
These four experiments are run using 45,000 data randomly distributed between 0 - 3R about the asteroid Eros. The model error is assessed on a separate test set of 50,000 data points distributed from 0-10R. The experiment is performed once with no noise in the training data (Figure B.1a), and again with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

[1] 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:

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

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, σ_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.

Parameters	N_{FF}	Avg. % Error	Δ Model Size [%]	Δ Avg. % Error
Fourier Layer Without Trainable σ and ϕ				
13923	5	0.191	14.84	-24.11
15723	10	0.206	29.69	-18.08
19323	20	0.192	59.39	-23.92
Fourier Layer With Trainable σ and ϕ				
13953	5	0.185	15.09	-26.60
15783	10	0.200	30.19	-20.60
19443	20	0.191	60.38	-24.25
Shared Fourier Layer With Trainable σ and ϕ				
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

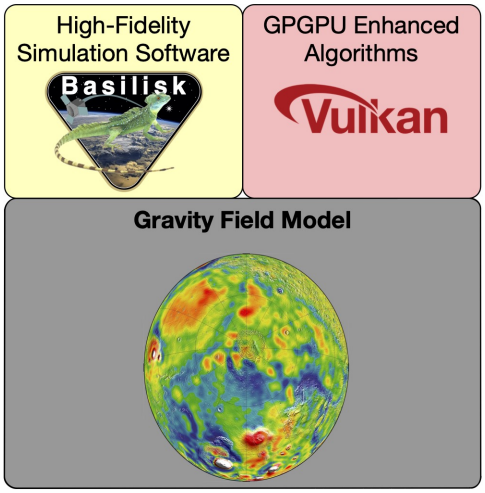


Figure E.1: GPGPU enhanced gravity algorithms for high-fidelity astrodynamics software enables support for high accuracy orbit propagation with lower runtimes.

(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).

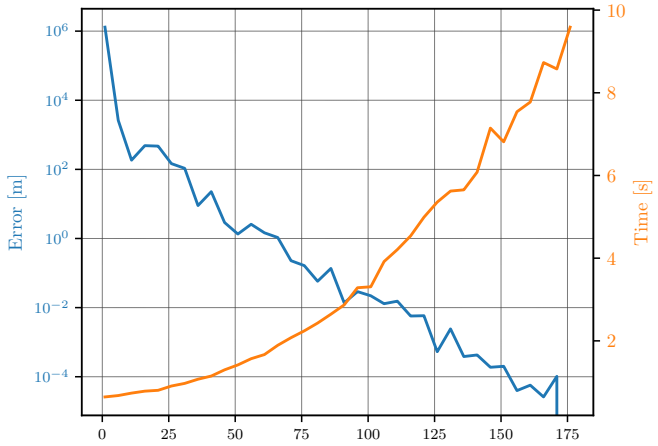


Figure E.2: Compute time and final error associated with simulating a spacecraft at 600km altitude orbiting for four hours real-time using Pines' formulation as a function of spherical harmonic degree.

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[1] — 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.

[1] 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 [2] . 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.

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

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

[2] https://www.khronos.org/vulkan/

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

$$C_{l,m} = \frac{C'_{l,m}}{R^l_{\text{ref}} m} \tag{E.2}$$

$$S_{l,m} = \frac{S'_{l,m}}{R^l_{\text{ref}} m} \tag{E.3}$$

where R_{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

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

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)

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

such that the coefficients and the associated legendre polynomials become

$$\bar{C}_{l,m} = \frac{C_{l,m}}{N_{l,m}} \tag{E.6}$$

$$\bar{S}_{l,m} = \frac{S_{l,m}}{N_{l,m}} \tag{E.7}$$

$$\bar{P}_{l,m}[x] = P_{l,m}[x] N_{l,m} \tag{E.8}$$

altogether providing

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

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

$$\boldsymbol{r} = r\begin{pmatrix} s \\ t \\ u \end{pmatrix} \qquad \hat{\boldsymbol{\imath}} = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \qquad \hat{\boldsymbol{\jmath}} = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \qquad \hat{\boldsymbol{k}} = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \tag{E.10}$$

such that

$$r = \sqrt{x^2 + y^2 + z^2} \tag{E.11}$$

$$s = \frac{x}{r} \tag{E.12}$$

$$t = \frac{y}{r} \tag{E.13}$$

$$u = \frac{z}{r} \tag{E.14}$$

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

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

where

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

Moreover, if one defines ξ as

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

then

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

such that

$$R_m[s,t] = Re\{\xi^m\} \tag{E.19}$$

$$I_m[s,t] = Im\{\xi^m\} \tag{E.20}$$

then the potential can be rewritten as

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

Defining

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

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

the potential simplifies further to

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

where the constituent terms abide by the following recursion relationships

$$R_m[s,t] = sR_{m-1}[s,t] - tI_{m-1}[s,t] \tag{E.25}$$

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

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

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

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

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

with the following initial conditions

$$R_0[s,t] = 1 \tag{E.31}$$

$$I_0[s,t] = 0 \tag{E.32}$$

$$\bar{A}_{0,0}[u] = 1 \tag{E.33}$$

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

$$\bar{A}_{l,m}[u] = N_{1_{l,m}} uA_{l-1,m}[u] - N_{2_{l,m}} A_{l-2,m}[u] \tag{E.34}$$

where

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

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

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

$$\nabla U(\boldsymbol{r}) = \frac{\partial U}{\partial r}\frac{\partial r}{\partial \boldsymbol{r}} + \frac{\partial U}{\partial s}\frac{\partial s}{\partial \boldsymbol{r}} + \frac{\partial U}{\partial t}\frac{\partial t}{\partial \boldsymbol{r}} + \frac{\partial U}{\partial u}\frac{\partial u}{\partial \boldsymbol{r}} \tag{E.37}$$

$$\frac{\partial r}{\partial \boldsymbol{r}} = \frac{1}{r}\hat{\boldsymbol{r}} \tag{E.38}$$

$$\frac{\partial s}{\partial \boldsymbol{r}} = \frac{1}{r}\hat{\boldsymbol{i}} - \frac{s}{r}\hat{\boldsymbol{r}} \tag{E.39}$$

$$\frac{\partial t}{\partial \boldsymbol{r}} = \frac{1}{r}\hat{\boldsymbol{j}} - \frac{t}{r}\hat{\boldsymbol{r}} \tag{E.40}$$

$$\frac{\partial u}{\partial \boldsymbol{r}} = \frac{1}{r}\hat{\boldsymbol{k}} - \frac{u}{r}\hat{\boldsymbol{r}} \tag{E.41}$$

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

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

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

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

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

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

(E.47)

Evaluating the partials:

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

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

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

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

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

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

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

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

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

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

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

where

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

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

The final acceleration can then be expressed as:

$$\begin{aligned} \boldsymbol{g} =&(a_1[r,s,t,u] + s \cdot a_4[r,s,t,u])\hat{\boldsymbol{\imath}} + \\ &(a_2[r,s,t,u] + t \cdot a_4[r,s,t,u])\hat{\boldsymbol{\jmath}} + \\ &(a_3[r,s,t,u] + u \cdot a_4[r,s,t,u])\hat{\mathbf{k}} \end{aligned} \tag{E.59}$$

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 venders like NVIDIA and AMD use similar microarchitectures[1] [2] [3] [4] . 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).

[1] https://www.nvidia.com/content/dam/en-zz/Solutions/design-visualization/technologies/turing-architecture/NVIDIA-Turing-Architecture-Whitepaper.pdf
[2] https://www.amd.com/system/files/documents/rdna-whitepaper.pdf
[3] https://www.techpowerup.com/gpu-specs/docs/amd-gcn1-architecture.pdf
[4] 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 is 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[5] .

[5] http://developer.download.nvidia.com/assets/cuda/files/reduction.pdf

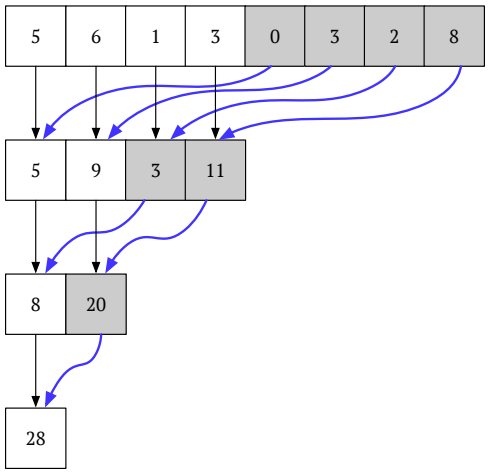


Figure F.1: Data Reduction Technique

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:

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

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

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

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

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:

$$Q = \frac{\rho_{l+1}[r]}{R_{\text{ref}}} \tag{F.5}$$

$$c_1 = \frac{N_{l,m}}{N_{l,m+1}} \tag{F.6}$$

$$c_2 = \frac{N_{l,m}}{N_{l+1,m+1}} \tag{F.7}$$

$$d_1 = s\bar{C}_{l,m} + t\bar{S}_{l,m} \tag{F.8}$$

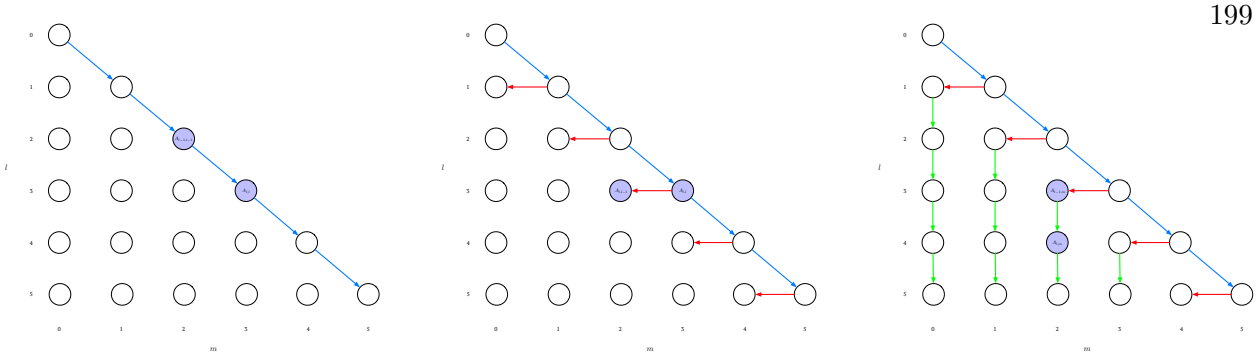
$$d_2 = s\bar{S}_{l,m} - t\bar{C}_{l,m} \tag{F.9}$$

$$\tag{F.10}$$

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:

$$\begin{bmatrix} a_{1_{l,m}} \\ a_{2_{l,m}} \\ a_{3_{l,m}} \\ a_{4_{l,m}} \end{bmatrix} = \begin{bmatrix} Q \\ Q \\ Q \\ Q \end{bmatrix} \begin{bmatrix} m \\ m \\ c_1 \\ c_2 \end{bmatrix} \begin{bmatrix} A_{l,m}[u] \\ A_{l,m}[u] \\ A_{l,m+1}[u] \\ A_{l+1,m+1}[u] \end{bmatrix} \left(\begin{bmatrix} \bar{C}_{l,m} \\ \bar{S}_{l,m} \\ d_1 \\ d_1 \end{bmatrix} \begin{bmatrix} R_{m-1} \\ R_{m-1} \\ R_{m-1} \\ R_{m-1} \end{bmatrix} + \begin{bmatrix} \bar{S}_{l,m} \\ -\bar{C}_{l,m} \\ d_2 \\ d_2 \end{bmatrix} \begin{bmatrix} I_{m-1} \\ I_{m-1} \\ I_{m-1} \\ I_{m-1} \end{bmatrix} \right) \tag{F.11}$$

Equation F.11 is advantageous for GPU computing for two reason. 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 the same six multiplications and one addition – common instructions that can be shared across all expressions. By converting



(a) First CPU Command (b) Second CPU Command (c) Third CPU Command

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 later 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_{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	μ, R_0	float	1	1
	$l_{\max}$	int	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

Compute lower diagonal $A_{l,l-1}$;
Thread barrier;
for $m = l$ **to** $N + 2$ **do**
    Recursively solve for $A_{l,m}$;
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

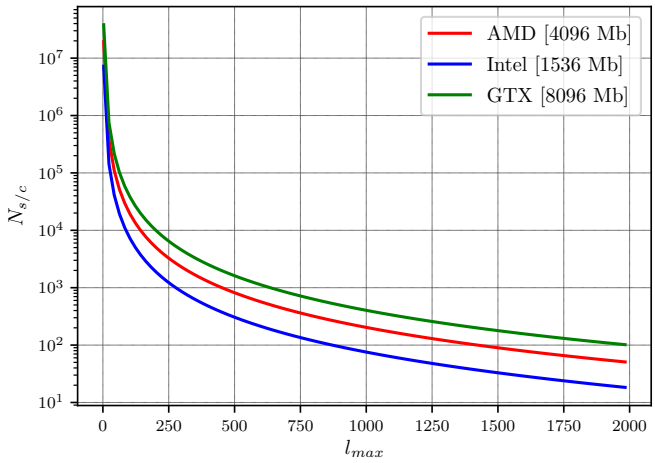


Figure F.3: Video memory consumed as a function of $l_{\max}$ and number of spacecraft, $N_{S/C}$

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 satellite 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:

$$\tau = \frac{t_{\text{CPU}}}{t_{\text{GPU}}} \tag{F.12}$$

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 a 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.

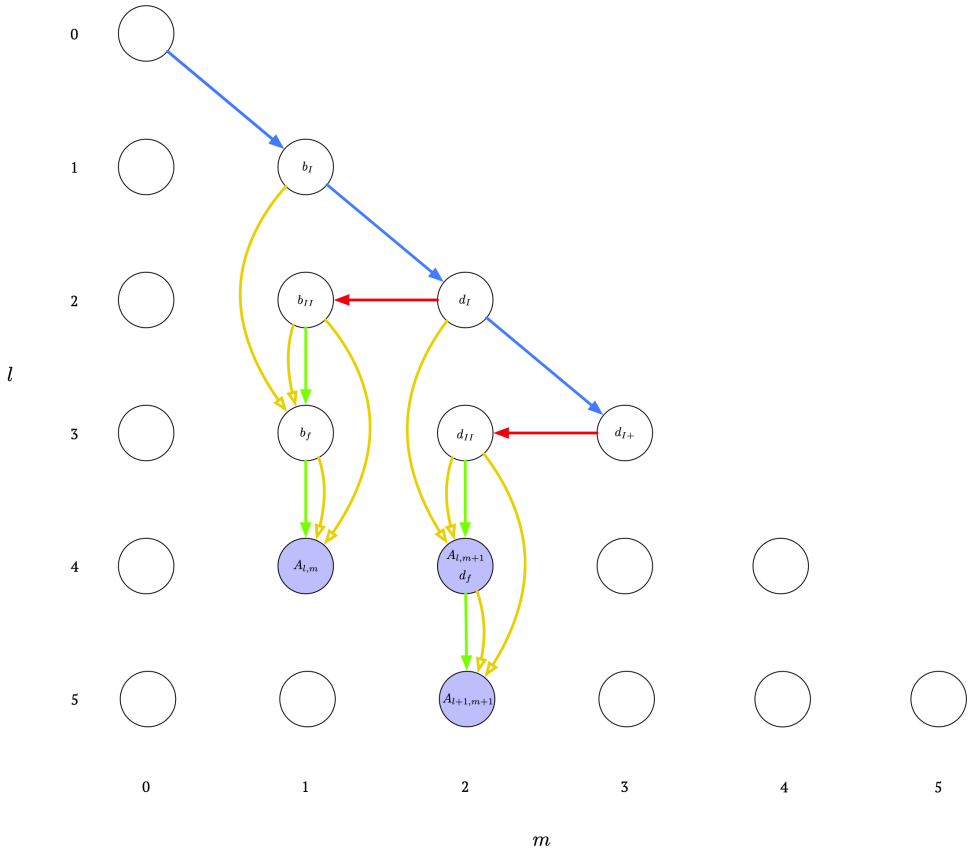
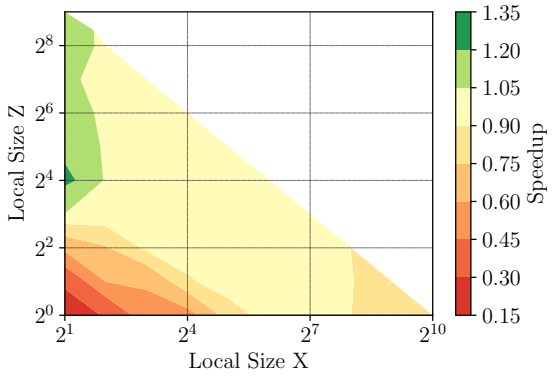
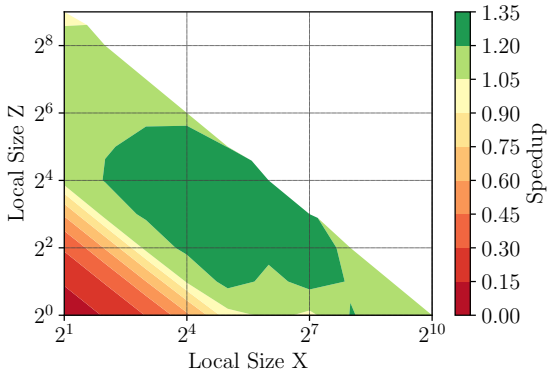


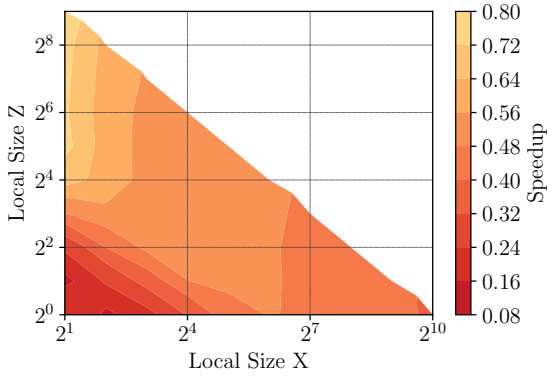
Figure F.4: Data Flow of GPU Kernel Invocation



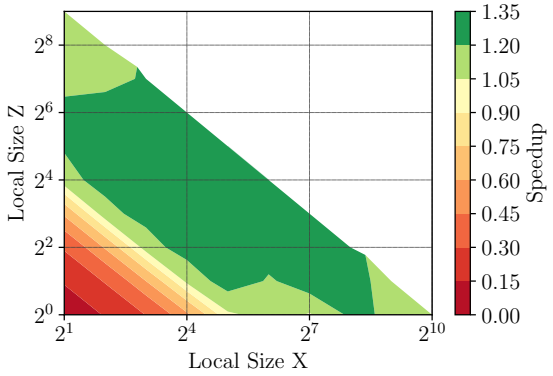
(a) 1024 Spacecraft; $l = 32$



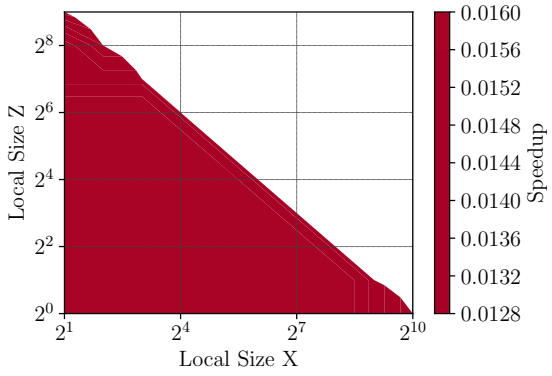
(b) 1024 Spacecraft; $l = 128$



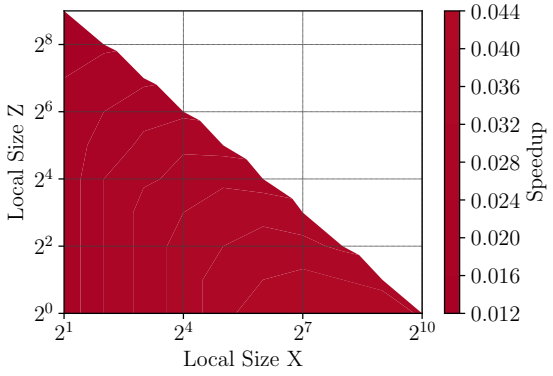
(c) 256 Spacecraft; $l = 32$



(d) 256 Spacecraft; $l = 128$

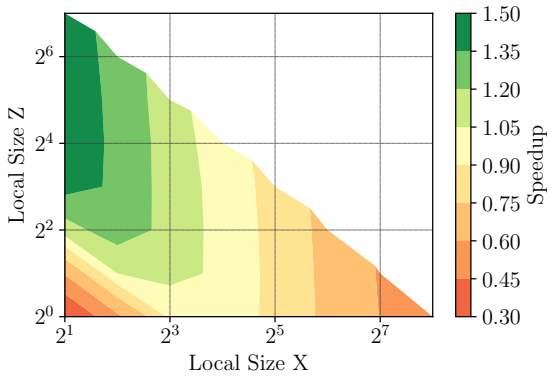


(e) 1 Spacecraft; $l = 32$

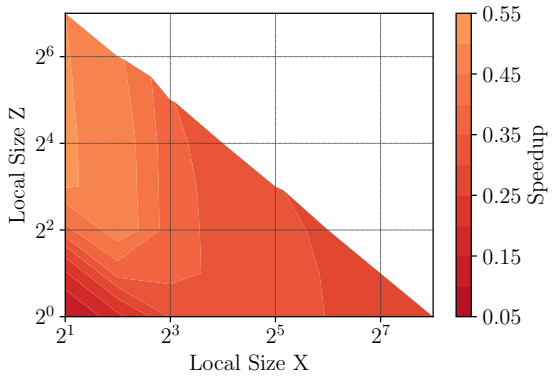


(f) 1 Spacecraft; $l = 128$

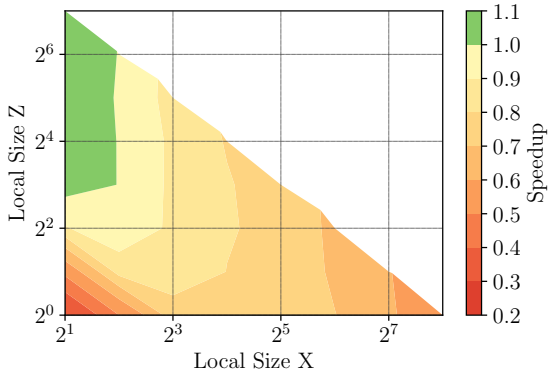
Figure F.5: Current speed up ratios for the Legendre matrix computation on AMD Radeon Pro 560 GPU using the compute bound method.



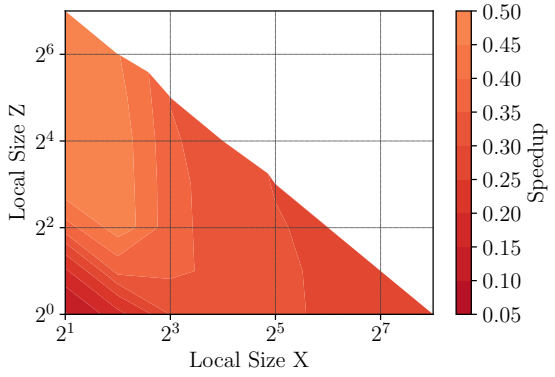
(a) 1024 Spacecraft; $l = 32$



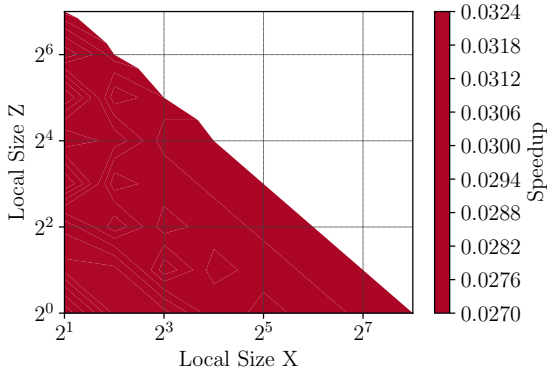
(b) 1024 Spacecraft; $l = 128$



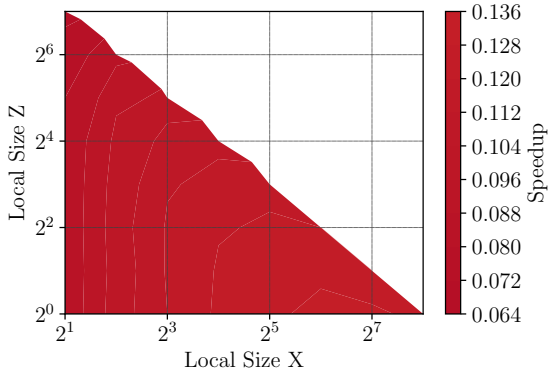
(c) 256 Spacecraft; $l = 32$



(d) 256 Spacecraft; $l = 128$



(e) 1 Spacecraft; $l = 32$



(f) 1 Spacecraft; $l = 128$

Figure F.6: Current speed up ratios for the Legendre matrix computation on Intel HD Graphics 630 GPU using the compute bound method.