POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Physics-informed neural network for gravity field modeling around Didymos binary system

Tesi di Laurea Magistrale in
Space Engineering

Author: **Marco Galeazzi**

Student ID: 991479
Advisor: Prof. Francesco Topputo
Co-advisors: Mattia Pugliatti, Prof. Fabio Ferrari
Academic Year: 2022-23

Abstract

High-fidelity representations of the gravity field could be necessary in order to precisely predict the orbit of a spacecraft in proximity of a body. Different methods have been introduced during the past years in order to construct analytically the gravity field of a body, like the spherical harmonics, mascon and polyhedral model. However, despite being convenient in theory, each method comes with its unique disadvantages when it comes to real applications. They could start to diverge near the surface of a body or require some assumptions on the mass distribution inside the object. Furthermore, they could be computationally very expensive and so, preventing their use for on-board purposes.
In order to deal with these problems, this work aims to model the gravity field with a machine learning representation instead of using a purely analytic formulation. In the case of single bodies, the use of physics-informed neural networks seem to solve the problem of divergence near the body while being computationally efficient and compact. Physics-informed neural networks combine the flexibility of deep learning models with centuries of analytic insight to learn new basis functions that are uniquely suited to represent these complex environments.
This work investigates the usage of physics-informed neural networks to model the gravity field of a binary system, more specifically of the Didymos asteroid system. The physics-informed neural networks in this work are trained with data generated from existing models or with simulated real data and the best configuration of physics-informed neural networks to obtain the best accuracy is investigated. The results show that the gravity field generated by the physics-informed neural networks gravity model can offer advantages over its analytic counterparts in computational time while having a better accuracy with respect to other models. The physics-informed neural networks are also able to model the gravity field from acceleration measurements taken in-situ even when considering errors on the measurements.

Keywords: Binary asteroids, Gravity field modeling, Artificial intelligence, Physics-informed neural network

Abstract in lingua italiana

Al fine di prevedere con precisione l'orbita di un veicolo spaziale in prossimità di un corpo celeste, possono essere necessarie rappresentazioni ad alta fedeltà del campo gravitazionale. Nel corso degli anni sono stati introdotti diversi metodi per costruire in modo analitico il campo gravitazionale di un corpo, come le armoniche sferiche, il modello mascon e il modello poliedrico. Tuttavia, nonostante siano convenienti in teoria, ciascun metodo presenta svantaggi unici quando si tratta di applicazioni reali. Possono iniziare a divergere vicino alla superficie di un corpo o richiedere alcune ipotesi sulla distribuzione della massa all'interno dell'oggetto. Inoltre, possono essere computazionalmente molto costosi, impedendo quindi il loro utilizzo a bordo.

Al fine di affrontare questi problemi, questo lavoro mira a modellare il campo gravitazionale utilizzando una rappresentazione basata sul machine learning anziché una formulazione puramente analitica. Nel caso di corpi singoli, l'utilizzo di physics-informed neural network sembra risolvere il problema della divergenza vicino al corpo, garantendo al contempo efficienza computazionale e compattezza. Le physics-informed neural network combinano la flessibilità dei modelli di deep learning con secoli di intuizione analitica per apprendere nuove funzioni di base che sono particolarmente adatte a rappresentare questi ambienti complessi.

Questo lavoro investiga l'uso delle physics-informed neural network per modellare il campo gravitazionale di un sistema binario, più specificamente del sistema asteroidale Didymos. Le physics-informed neural network in questo lavoro vengono addestrate con dati generati da modelli esistenti o con misurazioni reali simulate, e viene studiata la migliore configurazione delle physics-informed neural network per ottenere la miglior precisione possibile. I risultati mostrano che il campo gravitazionale generato dal modello gravitazionale physics-informed neural network può offrire vantaggi in termini di tempo computazionale rispetto ai suoi omologhi analitici, mantenendo una maggiore precisione rispetto ad altri modelli. Le physics-informed neural network sono anche in grado di modellare il campo gravitazionale a partire da misurazioni di accelerazioni effettuate in situ, anche andando a considerare errori sulle misurazioni.

Contents

1 | Introduction

1.1. Context

The study of small solar system objects, such as asteroids and comets, are a key instrument to understand the formation of our solar system and the origin of life. They are residual debris from the creation of the solar system. As their chemical structure has remained unchanged since their formation, such objects are thought to have a composition that is very similar to that of the Earth during its early stages of formation [10].
The binary asteroid environment is also the ideal place to study gravitational dynamics, to better understand how celestial bodies in the Solar System were formed and how they evolved. In addition, it could represent an ideal place for technology demonstration missions. For these reasons, the study of the dynamical environment near an asteroid pair has become a relevant topic for future missions [14].

Current estimates indicate that 16 % of the Near Earth Asteroid (asteroids whose orbit is very close to intersecting Earth's orbit) population is made of binary systems [21]. Thus, it is not surprising that a couple of missions in the near future have been planned to explore asteroid pairs in our Solar System.
For example, in 2027, Hera will rendezvous with the binary asteroid 65803 Didymos as the European contribution to AIDA (Asteroid Impact and Deflection Assessment). The objective of Hera is to investigate the effect of the Double Asteroid Redirection Test (DART) occurred in October 2022 and to investigate the Didymos binary asteroid, including the very first assessment of its internal properties [11].
Lucy will explore the Jupiter Trojan asteroids with a series of targeted close flybys of seven Trojans and one main-belt asteroid. The asteroids studied have diameters ranging from roughly 1 km to 100 km. A near equal-mass binary asteroid system will also be investigated. The objective of the mission will be to determine the surface composition, assess the geology and to determine the bulk properties [10]. A flyby at the start of November 2023 also discovered that the first asteroid under study Dinkinesh is actually a binary pair.

Some asteroid missions could require planning a set of robust close proximity operations around small bodies. Such operations could be challenging and can be complicated by numerous factors such as the irregular shape and mass distribution of a body and its weak and uncertain gravitational field. Due to such factors, the orbital dynamics around small bodies could deviate from the ideal Keplerian motion [12]. The knowledge of the dynamics driving the motion of a body in the vicinity of a binary system is then a key point for the success of the mission.

1.2. Objectives and research questions

In this work we will focus on modeling the gravitational field around a binary system. Different strategies can be adopted to model the gravity field of the asteroids, for example spherical harmonics, the polyhedral and mascon model could be used. These models can then be used to represent the dynamics of the restricted three body problem (where a body of negligible mass moves under the influence of two massive bodies) as described in [3, 15].
In [22–24] a new method to investigate the gravity field of an asteroid using a physics-informed neural network (PINN) is presented. The model seems to have a fast computation time once the PINN is trained, it can describe the potential of an asteroid with the same accuracy of other models while using a smaller number of parameters to do so and can be also modeled using only the acceleration measurements (without knowing the exact gravitational potential of the asteroid). The PINN model could be also theoretically used in-situ to model the gravitational acceleration.

This thesis main goal then will be to extend the work done by Martin to a binary asteroid system. In particular, the Didymos-Dimorphos system gravitational field is studied. This work aims to answer the following research question:

1. *To what extend can the physics-informed neural networks provide a characterization of the gravitational field of the Didymos binary system?*

The research question is also divided in the following sub-questions in order to address the main question:

1.1 *How accurately does the physics-informed neural network represent the gravity field compared to state of the art approaches? And what is the gain in terms of computational time of PINN with respect to them?*

1.2 *What is the best physic-informed architecture and methodology?*

1.3 *How many acceleration data are required to estimate the acceleration of the binary system using physics-informed neural network without any information about the potential? How does the error vary as function of the size of the training dataset? How sensitive it is to noise in the training data?*

1.3. Structure of the thesis

The thesis is organized as follows:

- Chapter 2 offers an overview of the literature on the investigated topics, providing a background for a complete understanding of the rest of the work;
- Chapter 3 focuses on the methodology used;
- In Chapter 4 an analysis of the results obtained is made;
- Chapter 5 draws conclusions and presents insight on how the work could be expanded.

2 | Theoretical background

In order to model the acceleration induced by a binary asteroid system, several factors must be taken in account. In this chapter is reported all the theoretical framework used to build up the models adopted. The traditional formulations for modeling the gravitational potential of small bodies are reported in section 2.1, an overview on the artificial neural networks (ANN), on the PINN and the state of the art for modeling the gravity field of small bodies with the usage of PINN is reported in section 2.2, and finally some considerations of which model is optimal is described in section 2.3.

2.1. Traditional gravitational models

The gravitational potential can be generalized for any kind of body as illustrated in Equation (2.1): [17]

$$U = -G \int_M \frac{1}{r} dm \tag{2.1}$$

Where G is the gravitational constant, M is the total mass of the body and r is the distance of the field point from the infinitesimal mass dm. It can be noticed that U is always negative and it tends to 0 when the field point is far from the body (r tends to ∞).
The acceleration $\boldsymbol{a}$ generated by the body can be found once the potential is known as reported in Equation (2.2):

$$\boldsymbol{a} = -\nabla U \tag{2.2}$$

The Laplacian of the potential depends on the position of the field point and, more specifically, if the field point is outside the body, the Laplace's equation is valid:

$$\nabla^2 U = 0 \tag{2.3}$$

While if the field point is inside the body, the Poisson's equation applies:

$$\nabla^2 U = -4\pi\rho G \tag{2.4}$$

Where ρ is the density of the body. Finally the curl of the gravitational acceleration is null because the force of gravity is a conservative force:

$$\nabla \times \nabla U = 0 \tag{2.5}$$

The acceleration provided by a multi-body system can be simply calculated as a summation of the acceleration imposed by each body (after rotating them in the same reference frame). We can then represent the potential for each asteroid individually to later compute the total acceleration provided [3, 15]. The gravity field around small bodies can be represented with many dynamical models. The most used are: point mass, spherical harmonics expansion, polyhedral gravity field and mass concentration (mascon) model. In the following subsections an overview on each of the mentioned models is reported with its main pros and cons.

2.1.1. Point mass model

The point mass model assumes that an object can be treated as if all its mass was concentrated at a single point, at its center of mass. It simplifies complex gravitational interactions but makes calculations more manageable. The gravitational potential U of a point mass can be defined as [17]:

$$U = -\frac{GM}{r} \tag{2.6}$$

Where G is the gravitational constant, M is the total mass of the body and r is the distance of the field point from the body.
It's important to note that the point mass model is an approximation and may not accurately represent the actual mass distribution of a real object as for example in case the body has an irregular shape. Due to this factor, the model could deviate from the true gravity field of a body, especially in proximity of the surface [12].

2.1.2. Spherical harmonics model

The spherical harmonics model is the most popular choice to represent the gravity model. It is an analytic solution to Laplace's equation as demonstrated in [19] and it is defined as:

$$U(r,\phi,\lambda) = \frac{GM}{r}\left[1 + \sum_{n=1}^{\infty}\left(\frac{R^*}{r}\right)^n \sum_{m=0}^{n} P_{nm}(sin\phi)[C_{nm}cos(m\lambda) + S_{nm}sin(m\lambda)]\right] \tag{2.7}$$

Where R^* is the reference radius that nominally is the radius of the Brillouin sphere (a sphere that circumscribe the entire body), P_{nm} is the associated Legendre function of degree n and order m defined as [20]:

$$P_{n0} = \frac{1}{2^n n!}\frac{d^n}{du^n}(u^2-1)^n \tag{2.8}$$

$$P_{nm} = (1-u^2)^{m/2}\frac{d^m}{du^m}P_{n0} \tag{2.9}$$

It could also be computed using recurrent relationships as described in [36]. r, ϕ and λ are the radius, latitude, and longitude of the field point relative to the coordinate origin and C_{nm} and S_{nm} are the spherical harmonic coefficients.

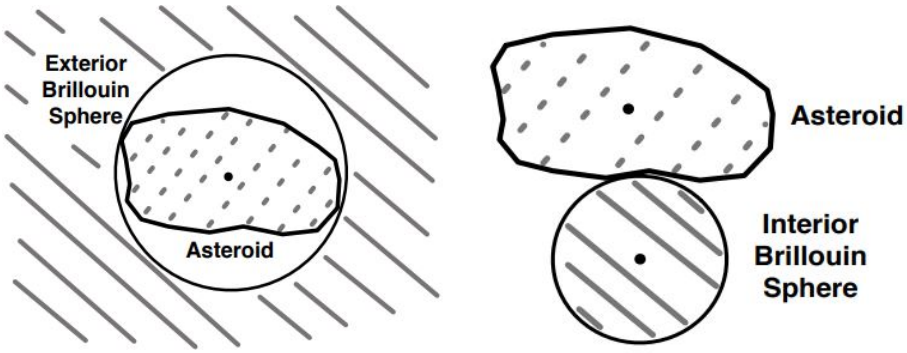
Usually, the model accuracy tends to gets better as the order of n increases and the potential can be easily computed with a low computational cost once the spherical harmonic coefficients are known. The spherical harmonic coefficients can be derived for example from real measurement [6] or from the shape model of the asteroid [36]. The acceleration can be computed for example using the Pines' method as described in [13].

This model however is valid only outside the Brillouin sphere while inside of it the model diverges. Therefore, the spherical harmonics model cannot model the dynamical environment within the Brillouin sphere, which can create some problems when we want to operate inside of it.

Another analytical solution to Laplace's equation is derived in [33] and it is the interior gravity field expressed in (2.10):

$$U^i(r,\phi,\lambda) = \frac{GM}{R_i^*}\sum_{n=0}^{\infty}\sum_{m=0}^{n}\left(\frac{r}{R_i^*}\right)^n P_{nm}(sin\phi)[C_{nm}^i cos(m\lambda) + S_{nm}^i sin(m\lambda)] \tag{2.10}$$

Where R_i^* is the radius of the interior Brillouin sphere (it is the biggest sphere centered in a point outside the body that does not contain any mass inside of it as depicted in Figure 2.1) and C_{nm}^i and S_{nm}^i are the interior spherical harmonic coefficients. The region of convergence of the two different methods is depicted in Figure 2.1 with the solid lines.



(a) (b)

Figure 2.1: Exterior (a) and interior (b) convergence regions [33]

A major drawback of the interior gravity field model is that only a small region of space corresponds to a converging region so, multiple internal models will be necessary in order to map out the entire space of the gravity field around the body. Also for each different internal model a different set of spherical harmonics coefficients must be computed. However, the interior gravity field could be efficiently used when we are interested only in a small region near the body, for example in a Touch-And-Go maneuver (TAG).

In order to map the entire region inside the exterior gravity field Brillouin sphere, in [31, 32], is reported a method called interior Bessel gravity field to compute the gravity field starting from the Poisson's equation. It consist in a redistribution of the mass of the body inside the Brillouin sphere in order to cover it all. A representation of the method is shown in Figure 2.2.

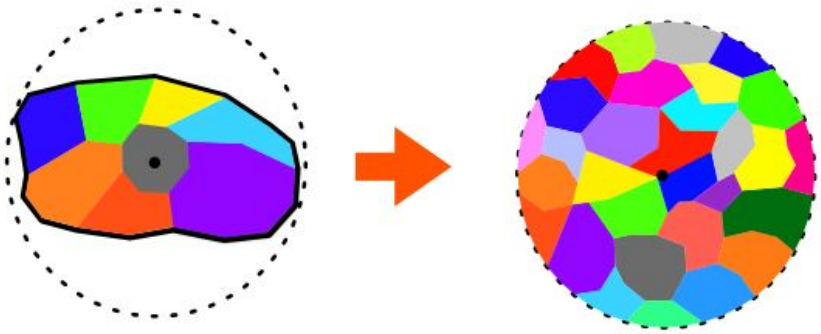


Figure 2.2: Redistribution of mass within the exterior Brillouin sphere [31]

The original density distribution on the left is redistributed throughout the exterior Brillouin sphere to model the sensed gravity field. For sake of brevity, the procedure to

implement it is not reported here. The method converges all inside the Brillouin sphere and could be combined with the exterior gravity field model expressed in Equation (2.7) in order to represent the potential of the body in his totality. In this case, at the boundary of each method (at the Brillouin sphere), the potential must match in order to have a smooth representation of the gravity field in the entire domain. This will also mean that the interior Bessel gravity field performance will be correlated to the performance of the exterior spherical harmonics at the boundary. This method seems less accurate with respect to the interior gravity field method. However, unlike the interior gravity field model, it can map the entire domain inside the Brillouin sphere. [31, 32]

2.1.3. Polyhedral gravity model

The polyhedral gravity model consists in computing the gravitational potential of a body by modeling the asteroid as a constant-density polyhedron. The polyhedron is a three dimensional solid body whose surface consists of planar faces meeting along straight edges (exactly two faces meet at each edge) or at isolated points called vertices. An example of this can be seen in Figure 2.3.

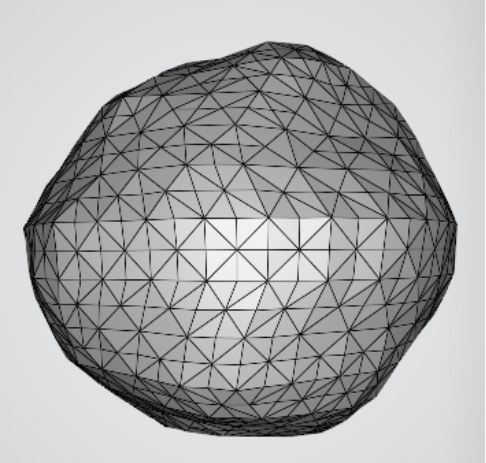


Figure 2.3: Asteroid Didymos polyhedral model. [25]

In [35], it is described the procedure to compute the gravitational potential in a field point given the polyhedral model of a body. In Equation (2.11) is reported the final result. If the reader is interested in the full demonstration of the formula, he is invited to read [35].

$$U = \frac{1}{2} G\rho \sum_{e \in edges} \boldsymbol{r}_e \cdot \boldsymbol{E}_e \cdot \boldsymbol{r}_e \cdot L_e - \frac{1}{2} G\rho \sum_{f \in faces} \boldsymbol{r}_f \cdot \boldsymbol{F}_f \cdot \boldsymbol{r}_f \cdot \omega_f \tag{2.11}$$

$\boldsymbol{r}_e$ is the position vector between the center of the edge and the field point, $\boldsymbol{r}_f$ is the distance between the face normal and the field point, the face dyad is expressed as $\boldsymbol{F}_f = \hat{\boldsymbol{n}}_f \hat{\boldsymbol{n}}_f$ where $\hat{\boldsymbol{n}}_f$ is the outward-pointing face normal vector, the edge dyad is $\boldsymbol{E}_{12} = \hat{\boldsymbol{n}}_A \hat{\boldsymbol{n}}_{12}^A + \hat{\boldsymbol{n}}_B \hat{\boldsymbol{n}}_{21}^B$ where the normal vectors are depicted in Figure 2.4 and L_e and ω_f are:

$$L_e = ln \frac{r_i + r_j + e_{ij}}{r_i + r_j - e_{ij}} \tag{2.12}$$

$$\omega_f = 2\ arctan\left(\frac{\boldsymbol{r}_i \cdot \boldsymbol{r}_j \times \boldsymbol{r}_k}{r_i r_j r_k + r_i(\boldsymbol{r}_j \cdot \boldsymbol{r}_k) + r_j(\boldsymbol{r}_k \cdot \boldsymbol{r}_i) + r_k(\boldsymbol{r}_i \cdot \boldsymbol{r}_j)}\right) \tag{2.13}$$

Where $r_i = ||\boldsymbol{r}_i||$, $\boldsymbol{r}_i$ is the vector from the variable field-point location to polyhedron vertex P_i and where e_{ij} is the length between two edge connecting vertices.

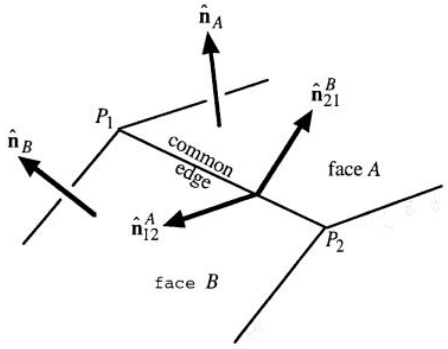


Figure 2.4: Schematic of two faces with a common edge and their respective normal [35]

The main disadvantages of the polyhedral model are its high computational cost and the fact that the density is assumed constant for the entire body. However, unlike the spherical harmonics model, the gravitational potential can be computed up to the body's surface without encountering divergence [35]. The gradient and Laplacian of the potential can also be computed easily as [35]:

$$\nabla U = -G\rho \sum_{e \in edges} \boldsymbol{E}_e \cdot \boldsymbol{r}_e \cdot L_e + G\rho \sum_{f \in faces} \boldsymbol{F}_f \cdot \boldsymbol{r}_f \cdot \omega_f \tag{2.14}$$

$$\nabla^2 U = -G\rho \sum_{f \in faces} \omega_f \tag{2.15}$$

Where all the terms inside the summation were already computed in Equation (2.11). The polyhedral model is also the only model between the ones reported here that is able to tell if a field point is inside or outside the body thanks to the laplacian and to the Laplace's and Poisson's equations.

2.1.4. Mascon model

The mascon model consists in a simple discretization of the total mass of the body in multiple point masses. The resulting potential and acceleration given by the body then is the summation of the contribution given by each single mass [27]:

$$U = -\sum_{i}^{N_i} \frac{Gm_i}{r_i} \tag{2.16}$$

$$\boldsymbol{a} = \sum_{i}^{N_i} \frac{Gm_i}{r_i^3} \boldsymbol{r}_i \tag{2.17}$$

Where m_i is the mass of the single point mass and where $\boldsymbol{r}_i$ is the vector from the field point to the single mass.
The total mass of the body shall be the same of the sum of all the point masses. However, the distribution inside the body can differ from model to model. One common approach employs an evenly spaced grid where the mass of the single point mass is the volume of the single grid cell multiplied for its density as depicted in Figure 2.5. Another approach consists in dividing the body represented with a polyhedron in a collection of tetrahedra and assigning the point mass in the center of each tetrahedra [5].

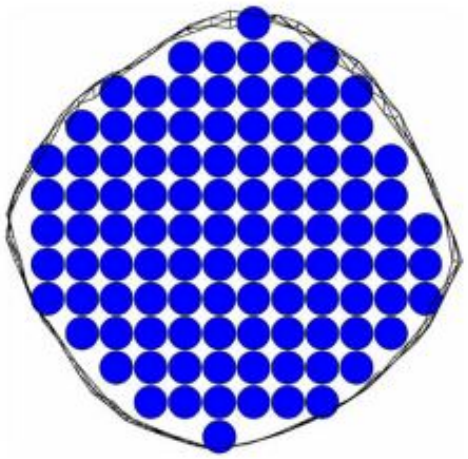


Figure 2.5: Bennu mascon model with an uniform grid [27]

The mascon model can also be used in order to estimate the distribution of density inside the body as reported in [18]. A drawback of the mascon model however, is that the precision of the model depends on the number of point masses considered. The model tends to give a good approximation only when the number of point masses considered

is high enough (especially near the surface) and the number of masses considered must be chosen depending on the requirements of the mission [35]. In order to increase the accuracy of the model, instead of increasing the number of point masses in the whole body, one could simply increase the number only near the surface as reported for example in [29]. In this way the accuracy of the model remains the same while the computational cost becomes overall smaller.

2.2. Data-driven gravitational models

In this section, a new model to compute the gravity field of a body based on PINN is discussed. In the subsection 2.2.1 an introduction to ANN is made, while in subsection 2.2.2 PINN are discussed.

2.2.1. Basis of artificial neural networks

An ANN is a computational learning system, based on the mechanisms underlying the way neurons and synapses in the human brain recognize patterns. It consists in a network of functions able to understand and translate a data input into a numeric output. [16]

In Figure 2.6, a representation of a typical ANN is shown. As we can see, the building blocks of the networks structure are simple nodes, grouped in layers. In this thesis we will only consider feedforward neural networks, where the flux of data goes from input to output.

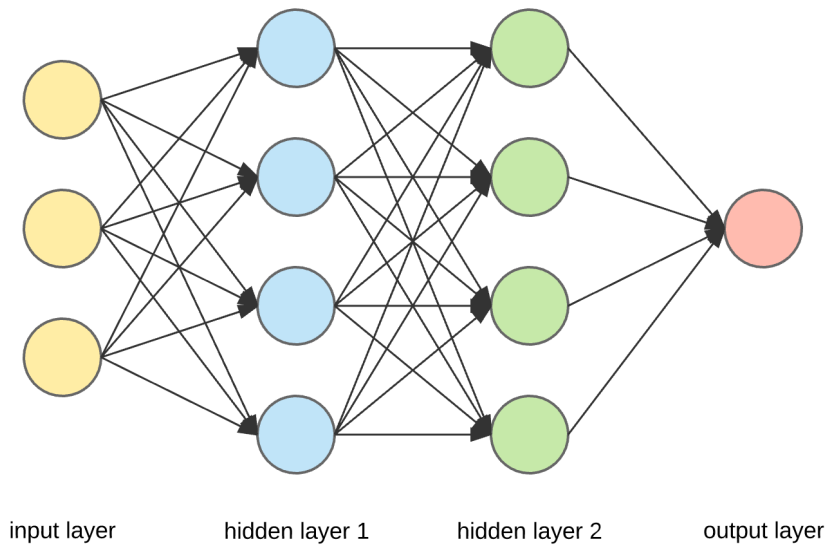


Figure 2.6: Example of a deep neural network

The left-most layer in Figure 2.6 represents the input layer where the known input data are inserted, while the right-most layer is the output layer where the ANN returns the estimation result. The remaining layers are the hidden layers. We will see later how these layers work and how to train them. In case more than one hidden layer is present, the network is called deep neural network.
Each node, or artificial neuron, belonging to one layer is connected with all the neurons that are part of the previous layer, as well as with all the neurons forming the following layer. The neurons belonging to the same layer are not interconnected. At each neuron is associated a bias b_i where i indicates the i-th neuron in the layer and a weight w_{ij} is associated to the neuron-to-neuron connection between the node i of the current layer and the node j of the previous layer. The value of the current node $h_i^{(k)}$ can then be expressed as:

$$h_i^{(k)} = w_{ij} h_j^{(k-1)} + b_i \tag{2.18}$$

Where $h_j^{(k-1)}$ is the value of the node at the previous layer k-1.
If we build our network in this way however, the ANN would be only capable of approximating linear functions due to the fact that our ANN would only be composed of linear functions. In order to avoid that, an activation function (a nonlinear transformation) σ is associated to each neuron. The current node value will then be:

$$h_i^{(k)} = \sigma(w_{ij} h_j^{(k-1)} + b_i) \tag{2.19}$$

Different activation functions exist in literature, some examples of them are reported in Table 2.1.

Activation function	$\boldsymbol{\sigma}$
Hyperbolic Tangent	$tanh(x)$
Sigmoid	$1/(1+e^{-x})$
ReLu	$max(0,x)$
Signum	$sgn(x)$
Heaviside step	$(sgn(x)+1)/2$
GELU	$0.5x(1+erf(x/\sqrt{2}))$

Table 2.1: Examples of activation functions used. [30]

The universal approximation theorem states that a feedforward network with a linear output layer and at least one hidden layer with any activation function can approximate

any Borel measurable function[1] from one finite-dimensional space to another with any desired non-zero amount of error, provided that the network is given enough hidden units [16]. In other words the ANN should be able to represent any Borel measurable function in order to have an error below ϵ with $\epsilon > 0$ if enough hidden nodes are considered. [30]

We have seen how a neural network works. We now see how to choose the parameters in order to make it work.
First of all, we need something in order to tell how the neural networks performs. To do that, an objective function can be used. In literature exists multiple objective functions for an ANN, one of the most used is the mean squared error (MSE):

$$J(\boldsymbol{x}, \boldsymbol{W}, \boldsymbol{b}) = \frac{1}{N_f} \sum_{i=1}^{N_f} (y_i^{true} - y_i^{out}(\boldsymbol{x}_i, \boldsymbol{W}, \boldsymbol{b}))^2 \tag{2.20}$$

Where y_i^{true} represents the output we desire, $y_i^{out}(\boldsymbol{x}_i, \boldsymbol{W}, \boldsymbol{b})$ is the output we obtain from the ANN and N_f is the total number of points used to train the network. In order to perform well then, the weights $\boldsymbol{W}$ and biases $\boldsymbol{b}$ of the ANN must be chosen in order to minimize the objective function J.
To choose them, we start by selecting randomly all the weights and biases (they are generally chosen as small numbers). We can then compute the variation of all the weights and biases using the gradient descent as reported in Equation (2.21):

$$w_{ij}^{new} = w_{ij}^{old} - \eta \frac{\partial J}{\partial w_{ij}} \tag{2.21}$$

Where η is the learning rate. There exist also different types of gradient descent [30], for example the stochastic gradient descent where only one random training example from a batch of data is used to calculate the gradient and update the parameters at each iteration or the batch gradient descent where the error for each example in the training dataset is calculated, but only updates the model after all training examples have been evaluated.
In order to compute the partial derivative a chain rule can be used:

$$\frac{\partial J}{\partial w_{ij}} = \frac{\partial J}{\partial y_i} \frac{\partial y_i}{\partial z_i} \frac{\partial z_i}{\partial w_{ij}} \tag{2.22}$$

Where z_i is simply the term at which the activation function is imposed and y_i is the output of the node. We can use a back-propagation in order to compute all the partial

[1] Any continuous function on a closed and bounded subset of $\mathbb{R}^n$ is Borel measurable function and can be approximated by a neural network. [16]

derivatives of the system by computing first the right-most terms and then by proceeding going to the left. In this way only the last term of the partial derivative obtained with the chain rule must be computed while all the other terms are already known [16].

The selection of the learning rate will represent how the ANN will respond to a new set of data, a smaller learning rate will mean that the variation of the weights will be smaller and the system is less prone to changes while a higher value will mean that the system will adapt more to the latest data. We can also make the learning rate change depending on how far we are from the minima. As the training goes on, the learning rate can be reduced in order to avoid jumping over the minima.
We can notice however that with the gradient descent the convergence to the global minimum is not guaranteed as it only guarantees convergence to a local minima.

Every ANN is characterized by a set of hyperparameters. Unlike the parameters $\boldsymbol{W}$ and $\boldsymbol{b}$, they usually do not change during the training. They can be defined as settings that define the ANN architecture and that we can use to control the behavior of the learning algorithm. Some examples could be the number of hidden layers, the numbers of neurons in an hidden layer, the number of data used to train the ANN or the algorithm used to train the ANN. The selection of the hyperparameters is not unique and changes from network to network. In order to choose them, a trial and error approach is usually used and, depending on the choice of the hyperparameters, the ANN will perform better or worse.

Another important aspect is that we need to validate the ANN. In order to do that, two different sets of data must be used. One will be used for the training of the ANN while a different set of data is used to check the performance of the ANN when it is training. It is important that the set of data used for the validation is different from the one used for the training. Overfitting is a common problem in neural networks and other machine learning models. It occurs when a model learns to perform very well on the training data but does not generalize well to unseen or new data. An example of it can be seen in Figure 2.7. The validation set will be used as a metric to understand if a training event has produced an overfit or not by comparing the training error with the validation error. The weights that generated the lowest validation error will then be chosen as a reference model.

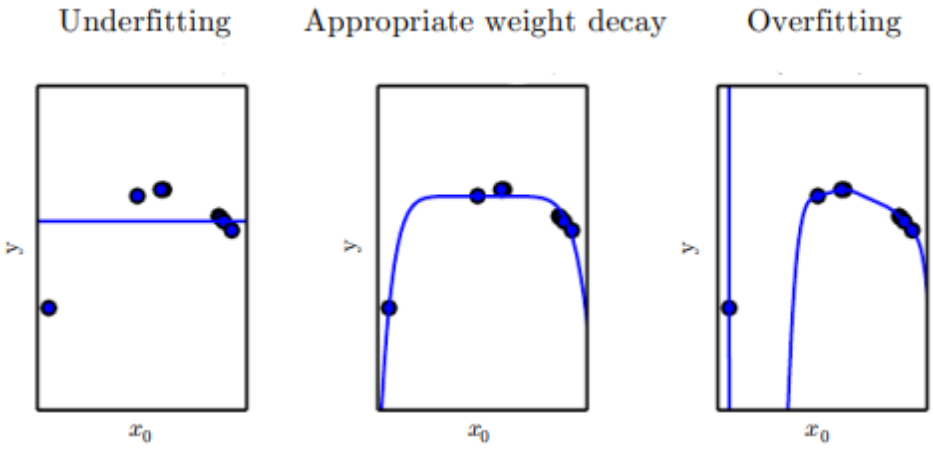


Figure 2.7: Overfitting of the training set [16]

2.2.2. Physics-informed neural networks

One of the disadvantages of using ANN to represent a function found in physics is that the learned representation may not satisfy the fundamental properties of said function. In [28] this problem is addressed and a method to train the ANN to ensure that the learned representations obey the differential equations that govern the system is proposed. To this end, the PINN are introduced. PINNs add the differential equations into the cost function of a traditional neural network and use automatic differentiation to ensure that these equations are respected by the function learned by the network. For example, consider the following differential equation:

$$f''(\boldsymbol{x}) + f'(\boldsymbol{x}) + f(\boldsymbol{x}) = 0 \tag{2.23}$$

Assume there exist measurements of $\boldsymbol{x}$ and the corresponding values of $f(\boldsymbol{x})$. A traditional neural network can use these observations as training data to learn a mapping from $\boldsymbol{x}$ to $f(\boldsymbol{x})$ by minimizing the cost function $J = (f^{real}(\boldsymbol{x}) - f^{output}(\boldsymbol{x}, \boldsymbol{W}, \boldsymbol{b}))^2$. The risk of training the network with this particular cost function is that the network does not know that the mapping must also satisfy Equation (2.23). PINNs solve this problem by inserting the original differential equation into the cost function:

$$\begin{aligned} J(\boldsymbol{x}, \boldsymbol{W}, \boldsymbol{b}) = &\frac{1}{N_f}\Big(\sum_{i=1}^{N_f} \left(f_i^{real}(\boldsymbol{x}) - f_i^{output}(\boldsymbol{x}, \boldsymbol{W}, \boldsymbol{b})\right)^2 + \\ &+ \left(f_i^{''output}(\boldsymbol{x}, \boldsymbol{W}, \boldsymbol{b}) + f_i^{'output}(\boldsymbol{x}, \boldsymbol{W}, \boldsymbol{b}) + f_i^{output}(\boldsymbol{x}, \boldsymbol{W}, \boldsymbol{b})\right)^2\Big) \end{aligned} \tag{2.24}$$

This cost function, not only penalizes erroneous values of $f^{output}(x)$, but also penalizes when the learned function violates the differential form of the problem. This extra term serves as a form of regularization in the training process which can lead to improved solutions that conveniently also satisfy important physics properties. The PINN model could also be useful to obtain a better approximation of the function in the presence of noisy training data. However, despite this additional robustness, these constraints could increase the amount of training time necessary for the PINN to converge given the computational complexity of calculating the derivatives of the function (especially when second order derivatives or beyond are considered). Also, in case multiple physics objectives are considered in the loss function, they could have competing gradient flow dynamics (i.e. the different objectives have different learning behaviors which may prevent some objectives from being leveraged during training). In order to avoid this, a weight could be associated at each physic objective and also a learning rate annealing algorithm can be implemented in order to adapt the weights during the training [34].

The derivatives of the network $f^{output}(x)$ are taken with automatic differentiation. There exist two different modes in order to compute the gradient with automatic differentiation, forward and reverse mode. The reverse mode is similar to the back-propagation but instead of computing the gradient of the loss function with respect to the weights, the partial derivative of the output (the potential U) with respect to all the inputs is computed using the chain rule. In the case of forward mode, instead of going from right to left we go from left to right, we compute the derivative of all the outputs with respect to one of the inputs. Notice that in case of the reverse mode we can compute only the derivative of a single output with respect to all the inputs in a single iteration, while for the forward mode is valid the opposite. The computation of the laplacian and of the curl can be obtained with a combination of the two modes. An example of the two models is reported in Tables 2.2 and 2.3, where the function studied is $y = f(x_1, x_2) = x_1 + e^{x_2} + sin(x_1 x_2)$ and its derivatives are studied in the point (4,1).

v_1	$= x_1$	$= 4$	$\dot{v}_1$	$= \dot{x}_1$	$= 1$
v_2	$= x_2$	$= 1$	$\dot{v}_2$	$= \dot{x}_2$	$= 0$
v_3	$= e^{v_2}$	$= 2.718$	$\dot{v}_3$	$= \dot{v}_2\ e^{v_2}$	$= 0$
v_4	$= v_1 v_2$	$= 4$	$\dot{v}_4$	$= \dot{v}_1 v_2 + v_1 \dot{v}_2$	$= 1$
v_5	$= \sin\ v_4$	$= -0.757$	$\dot{v}_5$	$= \dot{v}_4 \cos\ v_4$	$= -0.654$
v_6	$= v_1 + v_3 + v_5$	$= 5.961$	$\dot{v}_6$	$= \dot{v}_1 + \dot{v}_3 + \dot{v}_5$	$= 0.346$
y	$= v_6$	$= 5.961$	$\dot{y}$	$= \dot{v}_6$	$= 0.346$

Table 2.2: Forward mode example. On the left are reported the intermediate variables, on the right the partial derivatives with respect to the input x_1

v_1	$= x_1$	$= 4$	$\bar{v}_1$	$= \partial y/\partial x_1$	$= 0.346$
v_2	$= x_2$	$= 1$	$\bar{v}_2$	$= \partial y/\partial x_2$	$= 0.104$
v_3	$= e^{v_2}$	$= 2.718$	$\bar{v}_1$	$= \bar{v}_4\ \partial v_4/\partial v_1 + \bar{v}_6\ \partial v_6/\partial v_1$	$= 0.346$
v_4	$= v_1 v_2$	$= 4$	$\bar{v}_2$	$= \bar{v}_3\ \partial v_3/\partial v_2 + \bar{v}_4\ \partial v_4/\partial v_2$	$= 0.104$
v_5	$= \sin\ v_4$	$= -0.757$	$\bar{v}_3$	$= \bar{v}_6\ \partial v_6/\partial v_3$	$= 1$
v_6	$= v_1 + v_3 + v_5$	$= 5.961$	$\bar{v}_4$	$= \bar{v}_5\ \partial v_5/\partial v_4$	$= -0.654$
y	$= v_6$	$= 5.961$	$\bar{v}_5$	$= \bar{v}_6\ \partial v_6/\partial v_5$	$= 1$
			$\bar{v}_6$	$= \bar{y}$	$= 1$

Table 2.3: Reverse mode example. On the left are reported the intermediate variables, on the right the partial derivatives with respect to the output

While no PINN method has been applied for modeling the gravity field of a binary asteroid yet, John Martin in [22–24] tries to model the gravity field of single celestial bodies like Earth and Eros. The first model that the author implemented was a simple ANN with the relative position with respect to the body as input and the acceleration in each direction as output. This model however required a lot of data and was slow to train. As a solution, in [22] is proposed a model where, instead of trying to model the acceleration directly, the gravitational potential is found with the neural network and the acceleration is simply found using Equation (2.2). In this way the neural network will need to learn only one output instead of three. As a bonus, by computing the potential, Equation (2.3) and (2.5) can be put as constraint increasing in this way the robustness of the method. The architecture of the PINN used in [23] is shown in Figure 2.8.

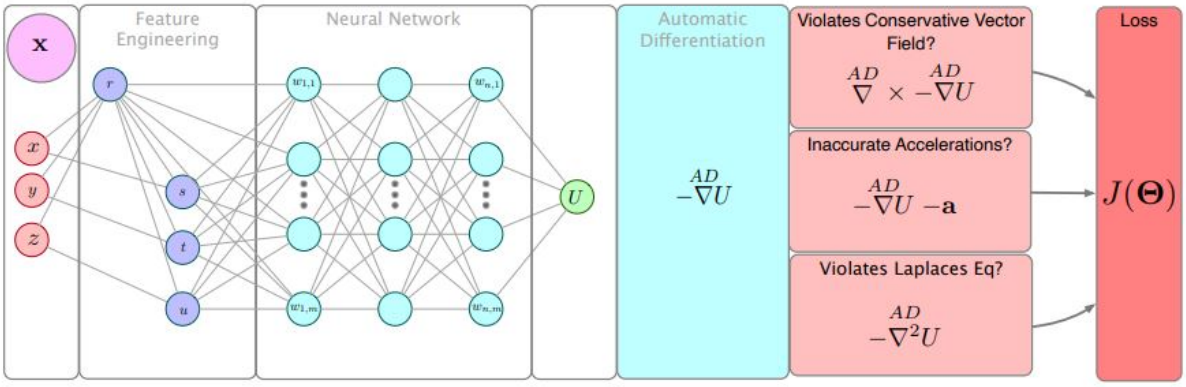


Figure 2.8: PINN for gravity field modeling of single celestial bodies [24]

The loss function used for the model is:

$$J(\boldsymbol{x}, \boldsymbol{W}, \boldsymbol{b}) = \frac{1}{N_f} \sum_{i=1}^{N_f} \|\boldsymbol{a}_i^{real}(\boldsymbol{x}_i) + \nabla f^{out}(\boldsymbol{x}_i, \boldsymbol{W}, \boldsymbol{b})\|^2 + \\ + \left(\nabla^2 f^{out}(\boldsymbol{x}_i, \boldsymbol{W}, \boldsymbol{b})\right)^2 + \|\nabla \times \nabla f^{out}(\boldsymbol{x}_i, \boldsymbol{W}, \boldsymbol{b})\|^2 \tag{2.25}$$

Where $\|...\|$ is the Euclidean norm. In case the exact potential of the body is known, it can be added at the loss function as:

$$J_{total}(\boldsymbol{x}, \boldsymbol{W}, \boldsymbol{b}) = J(\boldsymbol{x}, \boldsymbol{W}, \boldsymbol{b}) + \frac{1}{N_f} \sum_{i=1}^{N_f} \left(\boldsymbol{U}_i^{real}(\boldsymbol{x}_i) - f^{out}(\boldsymbol{x}_i, \boldsymbol{W}, \boldsymbol{b})\right)^2 \tag{2.26}$$

The gradient of the potential (and the laplacian and curl) can be found using automatic differentiation [1].

In [24], some modifications to the model of [23] are proposed. For example all the inputs and outputs are properly re-scaled in order to have a value between -1 and 1. In this way the PINN shows a faster converging time.
It can be noticed that, far from a body, the potential can be reduced to the one of a point mass without introducing a relevant error. Martin uses this information and impose that far from the body the potential transitions to the potential of the point mass with the following equation:

$$U^{output} = (1 - H(r))U_{NN}(r) + H(r)U_{BC} \tag{2.27}$$

Where r is the distance between the field point and the center of the body, U_{BC} is the

potential of the point mass, U_{NN} is the potential given by the PINN and $H(r)$ is:

$$H(r) = \frac{1 + tanh(k(r - r_{ref}))}{2} \tag{2.28}$$

Where r_{ref} is a learned (via gradient descent) or user-prescribed radius, and k is a learned or user-prescribed smoothing parameter to control for a more continuous or discrete transition.

In this way the potential not only converges in the set of data given, but can also converge outside of them. One of the main advantages of the model implemented in [22] is certainly the computational time. We can see in Figure 2.9 that the time needed to evaluate 10,000 random data using the PINN is order of magnitudes lower with respect to the other methods.

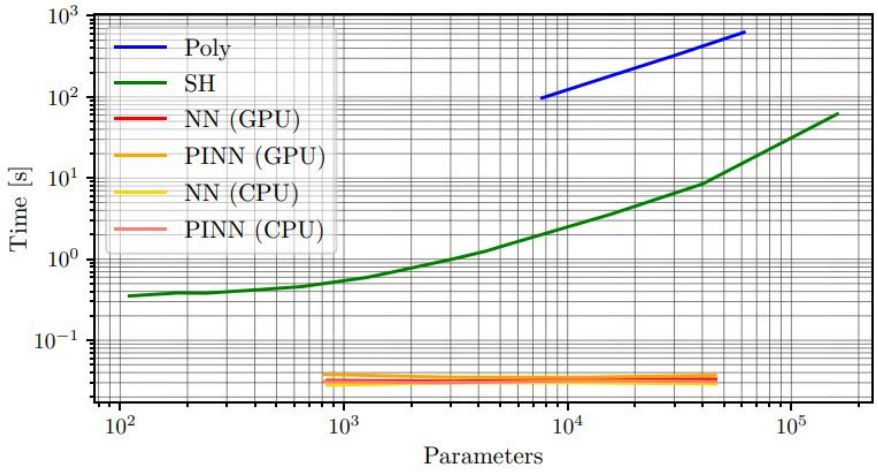


Figure 2.9: Total evaluation time using the various gravity models [22]

In [22], it can be also seen that, with the same number of parameters for the PINN and the spherical harmonics, the potential of the Earth at low altitudes is better represented by the PINN. In [23] it is shown that the PINN can obtain small errors with respect to the true values even in presence of noisy data and that the PINN could be theoretically trained even in-situ from real acceleration measurements. One drawback of this model is that, in order to converge to a good solution, some data in the proximity of the body must be taken (the distance from the body needs to be similar to twice the radius of the asteroid). However it is also shown that the number of measurement needed in order to converge is minimal and, after that, the accuracy of the model improves greatly.

2.3. Selection of the optimal model

All the models reported above have some pros and cons. The selection of the best gravity field model is not simple and depends also on the situation. When the field points to be computed are far away from the body, the spherical harmonics or the mascon model can be used, the choice between the two models will depend on the number of spherical harmonics known. At these distances the polyhedral model can be computationally inefficient despite its precision. The point model could also be used in order to reduce the computational cost if it approximate well enough the body. Near the body, the polyhedral model seems the best choice as shown in [35]. However, it is shown in [26] that, if the number of point masses is increased until the computational time is similar to the polyhedral model, the accuracy of the two models can be the same or the mascon model could be even superior. Obviously the number of masses considered depends also on the resolution of the surface of the body. Finally, the PINN model could be used in substitution of all the other models due to its fast computational time. Obviously, this will depend on the performances of the PINN to correctly evaluate the gravity field as it could theoretically reach the precision of the polyhedral model in proximity of the surface or could match the best model available for a celestial body in general. It could also be used in combination with other models as discussed before in order to reach the best accuracy. So, in summary, the model chosen is not defined a priori but needs some careful consideration before choosing it.
A quick recap of all the models is made in the next two pages.

Name of the method	Point mass model	Spherical harmonics	Polyhedral
Citations	[17]	[19, 32, 33]	[35]
Methodology	All the mass of a body is concentrated at a single point, at its center of mass.	The gravitational potential is approximated using spherical harmonics, at each order of the spherical harmonics is associated a spherical coefficient. Usually, the higher the order used to describe the potential, the higher the accuracy. The coefficient can be determined from the shape of the object or from acceleration measurements.	The exact potential of an object is analytically computed given its shape.
Advantages	Fastest computational time between all models.	Represent very well near-spherical celestial bodies. More specifically its oblateness. The order of the spherical harmonics can be chosen depending on how accurate we want the model.	If the exact shape of a body is known, the exact potential can be computed. The potential is exactly computed down to the surface and also inside the body. Once the potential is computed, the acceleration and the laplacian computation are trivial.
Disadvantages	Do not accurately represent the actual mass distribution of a real object as for example in case the body has an irregular shape.	Diverges inside the Brillouin sphere, this can be solved with other spherical harmonics model but in turns create other sort of problems. It is always an approximation of the true potential as it is a truncated series. Irregular components of the body are not so easily represented.	This is the model with the highest computational cost. If the exact shape of an object is not known, the potential can not be computed, the evaluation of the shape of an object can be problematic.

Name of the method	Mascon	PINN
Citations	[5, 27, 29, 35]	[22–24, 28]
Methodology	The total mass is discretized in multiple point masses. The total mass of the body must be equal to the sum of all the individual masses. The potential and acceleration is given by the summation of each individual mass contribution.	The potential is computed with a PINN and it is trained with acceleration data. The output of the PINN is the potential of the body and acceleration is obtained with automatic differentiation. Physics properties of the gravity field are taken in account in the loss function.
Advantages	The accuracy of the model can be changed by incrementing the number of masses considered. The body can be modeled with a varying density.	Once it is trained, it is the fastest method to compute the acceleration aside the point mass model. With the same number of parameters for PINN and spherical harmonics, the potential is better represented by the PINN. It obtains small errors with respect to true values even in presence of noisy data, can be trained in-situ from real acceleration measurements. Despite being a NN, can still follow physics properties of the potential.
Disadvantages	An high number of masses must be considered in order to have a good approximation.	It requires a lot of time and data in order to train it. The training with multiple losses could be problematic. In order to converge to a good solution, some data in the proximity of the body must be taken.

3 | Methodology

In this work, the dynamical environment near Didymos is studied. In particular, the gravity field is computed using a PINN approach. In the following sections, the dynamical environment is described in Section 3.1, the model used to train the PINN is described in section 3.2, the generation of the dataset is discussed in Section 3.3 and a recap on how to train the model is made in Section 3.4.

3.1. Dynamical environment

The dynamical environment of the binary asteroid system is defined in order to generate the data necessary to train the PINN. In Table 3.1, a list of the physical parameters used to model the binary system post DART impact is reported.

	Value	Unit
Didymos shape	Polyhedron	
Dimorphos shape	Triaxial ellipsoid	
Asteroids density	2170 ± 350	kg/m^3
Didymos maximum radius	422.7	m
Dimorphos maximum radius	104.0	m
Dimorphos axes	(104.0, 80.0, 66.5)	m
Didymos mass	$5.393\ 10^{11} \pm 8.698\ 10^{10}$	kg
Dimorphos mass	$5.008\ 10^{9} \pm 8.077\ 10^{8}$	kg
Semi-major axis of binary orbit	1170	m
Didymos spinning period	2.26	h
Dimorphos spinning period	11.37 (assumed tidally locked)	h
Secondary orbital period	11.37	h
Secondary Orbital Inclination	0	rad
Eccentricity	0.03-0.07	

Table 3.1: Physical parameters of the Didymos–Dimorphos system, [8, 9]

The shape of Didymos is modelled using a polyhedron[1], while for Dimorphos a triaxial ellipsoid is used as a reference with the dimension of the axes as reported in Table 3.1. The density of each asteroid is considered constant and the mass is obtained from multiplying the assumed density by the volume of the shape of the asteroid. In terms of relative orientation, it is assumed that the spin vector of Didymos and Dimorphos is aligned to the angular momentum vector of Dimorphos orbit around the primary. Dimorphos is assumed tidally locked. This model is simplified with respect to the possible real orientation of Dimorphos after the DART impact. In fact, Dimorphos oscillations on roll-pitch-yaw should be expected and will vary depending on the impact [9]. A schematic representation of the orbits of the asteroids around the center of mass of the binary system is reported in Figure 3.1.

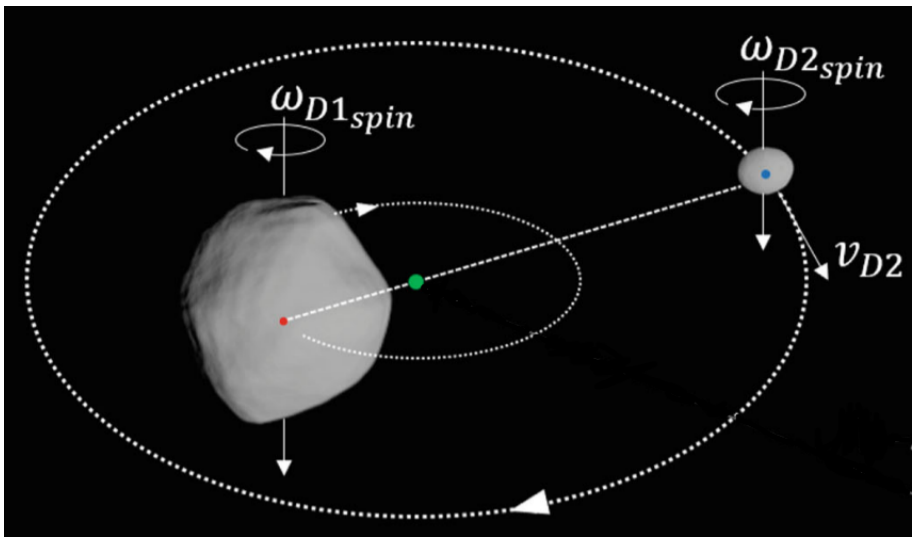


Figure 3.1: Didymos binary system model [15]

Three reference frames are considered in this work, the two body reference frame of each asteroid and a quasi-inertial reference frame centered in the barycenter of the binary system. The prefix quasi is to highlight that the system can be considered inertial for characteristic times shorter than those related to Didymos heliocentric motion, which is typically the case of spacecraft orbit design [15]. From now on it will be referred as an inertial frame.

[1]The polyhedron model for both asteroids can be found at https://dart.jhuapl.edu/DART-Proposal-Reference/.

3.1.1. Reference models

In order to train the PINN to map the acceleration and potential in a field point, a reference model shall be imposed first. In this work, two different cases are considered to train the PINN to represent the gravitational acceleration provided by the asteroid binary system. In the first case, the PINN is trained from some known model while in the second case it is trained from simulated total gravitational acceleration measurements.

In case the network is trained from some known model, the potential and the acceleration of the asteroid can be computed individually in a field point with coordinates in the body frame of the same body. The reference models used to compute the acceleration and the potential are a polyhedral model for Didymos and a triaxial ellipsoid model for Dimorphos. Didymos acceleration and potential in a field point can be computed as discussed in Section 2.1.3 [35]:

$$\hat{\boldsymbol{a}}_{didymos} = G\rho \sum_{e \in edges} \boldsymbol{E}_e \cdot \boldsymbol{r}_e \cdot L_e + G\rho \sum_{f \in faces} \boldsymbol{F}_f \cdot \boldsymbol{r}_f \cdot \omega_f \tag{3.1}$$

$$U_{didymos} = \frac{1}{2} G\rho \sum_{e \in edges} \boldsymbol{r}_e \cdot \boldsymbol{E}_e \cdot \boldsymbol{r}_e \cdot L_e - \frac{1}{2} G\rho \sum_{f \in faces} \boldsymbol{r}_f \cdot \boldsymbol{F}_f \cdot \boldsymbol{r}_f \cdot \omega_f \tag{3.2}$$

The Dimorphos acceleration can be computed using the triaxial ellipsoid approximation as described in [7]:

$$\begin{cases} \hat{a}_{x,dimorphos} = -2\pi G\rho abcx \displaystyle\int_k^\infty \frac{ds}{(a^2+s)\Delta(s)} & (3.3) \\ \hat{a}_{y,dimorphos} = -2\pi G\rho abcy \displaystyle\int_k^\infty \frac{ds}{(b^2+s)\Delta(s)} & (3.4) \\ \hat{a}_{z,dimorphos} = -2\pi G\rho abcz \displaystyle\int_k^\infty \frac{ds}{(c^2+s)\Delta(s)} & (3.5) \end{cases}$$

Where:

$$\Delta(s) = \sqrt{(a^2+s)(b^2+s)(c^2+s)} \tag{3.6}$$

The parameter k is the algebraically largest root of the equation:

$$\frac{x^2}{a^2+k} + \frac{y^2}{b^2+k} + \frac{z^2}{c^2+k} = 1 \tag{3.7}$$

ρ is the homogeneous density of the ellipsoid, while a, b and c are the axes of the ellipsoid ($a > b > c$). x, y and z are the coordinates in a Cartesian reference frame of the field

point with respect to the center of mass of the ellipsoid.
The potential of the triaxial ellipsoid can be computed as:

$$U_{dimorphos} = \pi G \rho abc \int_k^\infty \left(1 - \frac{x^2}{a^2+s} - \frac{y^2}{b^2+s} - \frac{z^2}{c^2+s}\right) \frac{ds}{\Delta(s)} \tag{3.8}$$

In case the potential and the acceleration of an interior field point are required, the procedure is the same as the one discussed above with the exception of k that is put equal to 0. In order for the integral to converge, the parameter s can be substituted with $\frac{s}{a^n}$ where n is a tuned parameter.

In case the PINNs are trained from these known models, the acceleration and potential in a field point are simply computed in this way. However, if the network is trained with simulated total gravitational acceleration measurements, the total gravitational acceleration in a field point must be computed as it is assumed that only this component is known. The total gravitational acceleration $\ddot{\boldsymbol{r}}$ provided by the binary system can be computed as the summation of the acceleration provided by each asteroid as [15]:

$$\ddot{\boldsymbol{r}} = \boldsymbol{a}_{didymos} + \boldsymbol{a}_{dimorphos} \tag{3.9}$$

Where $\boldsymbol{a}_{didymos}$ and $\boldsymbol{a}_{dimorphos}$ represent the gravitational acceleration provided by each asteroid in the inertial frame. As the accelerations computed before are obtained in the body frame, they must be rotated in the inertial frame [15]:

$$\boldsymbol{a}_{didymos} = \boldsymbol{R}_{didymos} \hat{\boldsymbol{a}}_{didymos} \tag{3.10}$$

$$\boldsymbol{a}_{dimorphos} = \boldsymbol{R}_{dimorphos} \hat{\boldsymbol{a}}_{dimorphos} \tag{3.11}$$

Where $\boldsymbol{R}_{didymos}$ represents the rotation between Didymos body-fixed frame and the inertial frame and where $\boldsymbol{R}_{dimorphos}$ represents the rotation between Dimorphos body-fixed frame and the inertial frame. The acceleration obtained with Equation 3.9 will be used as a reference model to train the network when only measurements of the total gravitational acceleration in a field point are assumed known.

The contributions of the solar radiation pressure (SRP) and of the third-body effects of the Sun were not considered unlike in [15] because for the training of the PINN only the gravitational acceleration provided by the asteroids are required. In case the PINN is trained using measurements of the total acceleration as in Section 4.2, it is assumed that the instrument used is able to measure directly the gravity field as it is in line with real space instruments used [4]. The contribution of these components omitted with respect

to Didymos and Dimorphos gravitational acceleration can be seen in Figure 3.2. The acceleration due to SRP is computed for the Milani CubeSat case, while all other effects are independent from the spacecraft mass [15].

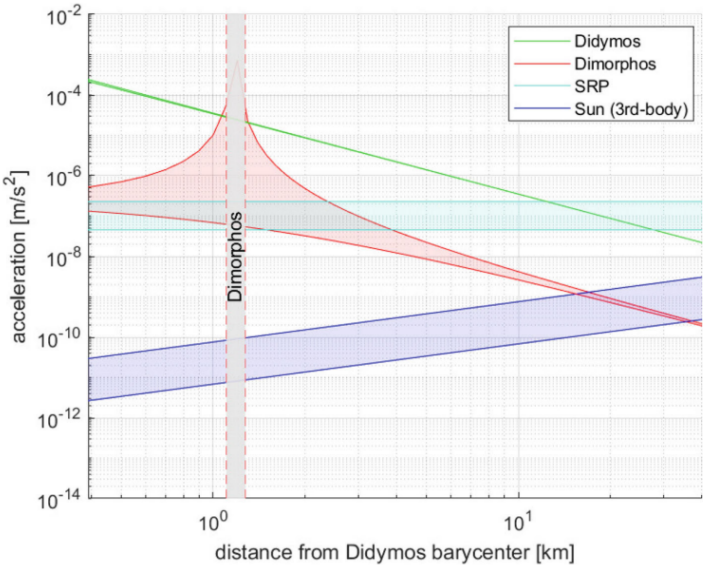


Figure 3.2: Main accelerations in the proximity of Didymos binary system. Gravity of Didymos (primary) and Dimorphos, Sun (third body) and SRP as function of the distance from the barycenter of Didymos system [15]

3.2. PINN implementation

In order to train the PINN to represent the gravitational acceleration provided by the asteroid binary system, the two different cases discussed in Section 3.1.1 are considered. The architecture of the model in both cases is discussed in this section.

3.2.1. Model architecture

The model used in this work takes inspiration from [22–24] where a PINN was used to represent the gravity field of a single celestial body and adapts it to a binary system. The model consists of two PINNs. Each PINN will be used to map the potential of a singular asteroid. In case the PINNs are trained from some known model, two different networks can be trained independently as shown in Figure 3.3. In case only total gravitational acceleration measurements are known, the PINNs must be trained using a common loss function as shown in Figure 3.4. The only difference between the two methods is the final loss function and how the dataset is generated. The model were implemented in Python

with the usage of TensorFlow 2.10[2].

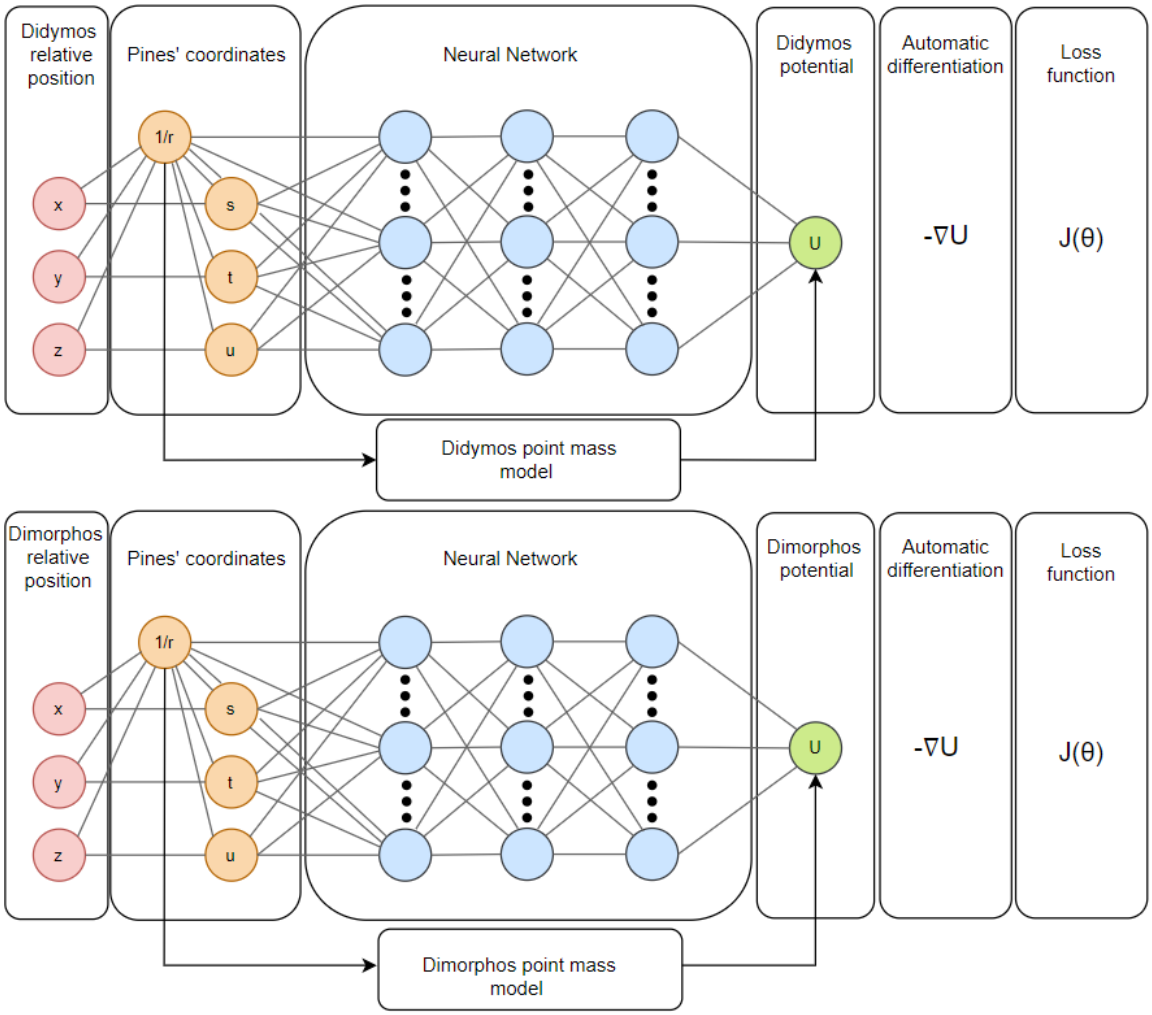


Figure 3.3: PINNs architecture to approximate the gravity field of a binary system from a known model

[2]https://www.tensorflow.org/

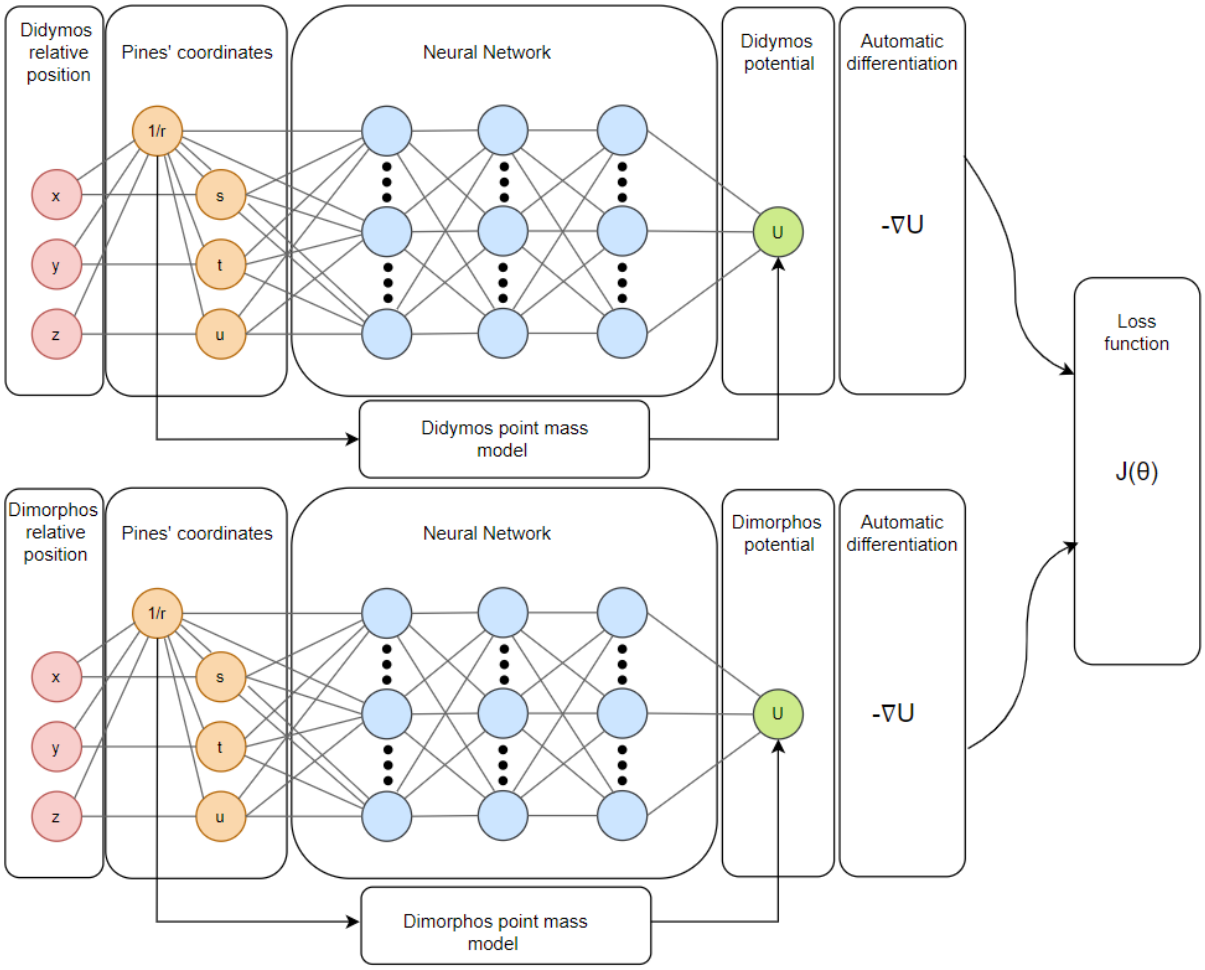


Figure 3.4: PINNs architecture to approximate the gravity field of a binary system from total gravitational acceleration measurements

The architecture of the singular network is pretty similar to the one implemented in [24]. It takes as an input the relative position of the field point with respect to the corresponding asteroid with coordinates in the body frame of the same celestial body and returns as an output the potential of the asteroid in the field point.

Another approach was considered to model the gravity field of the asteroid binary system. Instead of modeling two different PINNs (one for each asteroid), a single PINN was trained with the total potential of the field point as a single output. However, this model did not performed well and it is not presented in this work. A quick recap of its performances is reported in Appendix A.

3.2.2. Input definition

The input position is initially given in Cartesian coordinates in the body frame of each asteroid. A transformation of the input positions is made to increment the performances of the network. The position is expressed using the same coordinates used in Pines' acceleration computation (r, s, t, u):

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

The coordinates s, t and u will be bounded between the values of $[-1, 1]$. The same can not be said for the radius that can scale between $[0, \infty]$. To avoid that, the radial coordinate is inverted. In this way, the value is bounded between $[0, 1]$ outside of the body thank to a dimensionalization discussed later. The coordinates $1/r, s, t, u$ are then used as inputs of the Neural Network.

3.2.3. Potential and acceleration computation

The output of the network is the potential of a single body in a field point. In order to increment the performances of the PINN, the potential obtained from the point model of an asteroid can be added at the output of the network. In this way, the PINN does not need to learn this prominent and easily observable contribution and can focus on mapping the potential's perturbations.

Once the potential is computed and the potential of the point mass model is added to it, the acceleration is obtained through automatic differentiation as explained in Section 2.2.2. The gradient of the potential is computed with respect to the x, y and z coordinates in order to obtain the acceleration of the field point in the body frame of the corresponding asteroid. It is then mandatory to keep the Cartesian coordinates within the TensorFlow graph as shown in Figure 3.3 and 3.4.

By using again automatic differentiation and by computing the gradient of each component of the acceleration with respect to the Cartesian coordinates, it is also possible to compute the Laplacian of the potential and the curl of the acceleration. These components will also be used in the loss function as discussed in the next subsection.

3.2.4. Loss function

In order to train the PINNs, the two different cases are taken in account. In the first case, where the model is simplified and the PINNs can be trained independently as the acceleration of the single asteroid in a certain field point can be computed, each neural network will have its own loss function and it will be:

$$J(\boldsymbol{\Theta}) = \frac{1}{N}\sum_{i=0}^{N}\frac{\|\boldsymbol{a}_{true,i} - (-\boldsymbol{\nabla}U_{NN,i})\|}{\|\boldsymbol{a}_{true,i}\|} \tag{3.12}$$

Where N represents the number of samples in a batch and the term in the summation is the percentage acceleration error where $a_{true,i}$ is the acceleration obtained from the reference model, where $U_{NN,i}$ is the potential obtained from the PINN model and where $\|...\|$ is the Euclidean norm. The percentage error is used as a loss function over the MSE as it increases the performances of the network far away from the asteroid [24]. The gradient of the potential is computed with respect to the Cartesian coordinates to obtain the acceleration and the double minus sign is due to Equation 2.2. The subscript i indicates the i-th data of the batch used for training at a given epoch.

Some other additional physics properties can be learned and can be implemented in the loss function. The curl of the acceleration and the laplacian of the potential are imposed equal to zero. It can also be imposed that the potential obtained from the PINN is equal to the potential obtained from the model. By taking all this in account, the original cost function becomes:

$$\begin{aligned} J(\boldsymbol{\Theta}) =& \frac{1}{N}\sum_{i=0}^{N}\frac{\|\boldsymbol{a}_{true,i} - (-\boldsymbol{\nabla}U_{NN,i})\|}{\|\boldsymbol{a}_{true,i}\|} + \frac{|U_{true} - (U_{NN,i})|}{|U_{true}|} + \\ &+ \left(\boldsymbol{\nabla}^2 U_{NN,i}\right)^2 + \|\boldsymbol{\nabla}\times\boldsymbol{\nabla}U_{NN,i}\|^2 \end{aligned} \tag{3.13}$$

Where $|...|$ is the absolute value.
In the second case, only the total acceleration provided by the binary system is known. It is then not possible to model each PINN independently as the potential or acceleration for a singular asteroid in a certain field point is not known. In order to solve this problem, a singular loss function will be used to train both PINNs. The total acceleration obtained in a certain field point from a measurement will be imposed equal to the summation of the contribution of both PINNs using the following loss function:

$$J(\boldsymbol{\Theta}) = \frac{1}{N}\sum_{i=0}^{N}\frac{\|\boldsymbol{a}_{tot,i} - (-\boldsymbol{R}_{1,i}\boldsymbol{\nabla}U_{NN,1,i} - \boldsymbol{R}_{2,i}\boldsymbol{\nabla}U_{NN,2,i})\|}{\|\boldsymbol{a}_{tot,i}\|} \tag{3.14}$$

In a similar way as before, the percentage acceleration error is computed. However, in this case, the accelerations obtained from each PINN shall be rotated from the body frame to the inertial frame with the rotational matrix as discussed in Section 3.1.1. Once they are rotated, they can be summed together and can be imposed equal to the total gravitational acceleration. The subscripts 1 and 2 indicates respectively the potential of Didymos and Dimorphos. This is valid also for the rotational matrix.
As before, the laplacian of the potential and the curl of the acceleration of both Didymos and Dimorphos can be imposed equal to zero. The following terms can then be added to the loss function reported in Equation 3.14:

$$\frac{1}{N}\sum_{i=0}^{N}\left(\boldsymbol{\nabla}^2 U_{NN,1,i}\right)^2 + \left(\boldsymbol{\nabla}^2 U_{NN,2,i}\right)^2 + \|\boldsymbol{\nabla}\times\boldsymbol{\nabla}U_{NN,1,i}\|^2 + \|\boldsymbol{\nabla}\times\boldsymbol{\nabla}U_{NN,2,i}\|^2 \tag{3.15}$$

In this work, all the different terms of the loss function are tested in both cases to see which ones produce the better performances. Different loss functions are then created and from this point on they are referred as:

1. NN_A: only the acceleration term of the loss function is considered
2. NN_P: only the potential term of the loss function is considered
3. NN_AP: both the acceleration and potential terms of the loss function are considered
4. NN_ALC: the acceleration term is considered with the laplacian and the curl
5. NN_ALCP: all the terms of the loss function are considered

In addition to these loss functions, an ANN called NN_0 is also considered. In this case, the output of each Neural Network is not the potential of the body but instead are the three components of the acceleration. Also in this case the point mass model is added to the output.

The loss function for this term in case the ANN is trained from some known model will look as follow:

$$J(\boldsymbol{\Theta}) = \frac{1}{N}\sum_{i=0}^{N}\frac{\|\boldsymbol{a}_{true,i} - (\boldsymbol{a}_{NN,i})\|}{\|\boldsymbol{a}_{true,i}\|} \tag{3.16}$$

Where $\boldsymbol{a}_{NN,i}$ is the acceleration obtained from the ANN. In case the ANN are trained

from acceleration measurements, the total acceleration measured is imposed equal to the sum of the contribution of each ANN once they are rotated in the inertial reference frame as before.

3.2.5. Non-dimensionalization

All inputs and outputs of the neural network are normalized in order to be between the values of $[-1, 1]$ in order to achieve better performances [16]. To do that, the following dimensionalization are made:

$$\begin{aligned}\hat{x} &= R^* \\ \hat{U} &= max(|U - U_{point}|) \\ \hat{a} &= \frac{\hat{U}}{\hat{x}}\end{aligned}$$

Where R^* is the Brillouin sphere radius, $max(|U - U_{point}|)$ is the maximum difference between the potential of the reference model and of the point model and where the parameters with the hat symbol represent the quantity for which the position, the potential and the acceleration must be normalized. The normalization takes in account the common units of measurements of each component [24]. Each model will have its own dimensionalization.

In case the model is trained from real measurements, the potential of the model is not known. Then, another dimensionalization is made as reported below:

$$\begin{aligned}\hat{x}_1 &= R_1^* \\ \hat{x}_2 &= R_2^* \\ \hat{a} &= max(\|\boldsymbol{a}_{tot} - \boldsymbol{a}_{point,1} - \boldsymbol{a}_{point,2}\|) \\ \hat{U}_1 &= \hat{a}\hat{x}_1 \\ \hat{U}_2 &= \hat{a}\hat{x}_2\end{aligned}$$

The subscripts 1 and 2 indicate the dimensionalization for Didymos and Dimorphos PINNs respectively and all the accelerations are expressed in the inertial reference frame.

3.2.6. Output rescaling

By applying the dimensionalization discussed in the previous subsection and when summing the point mass potential at the output of the PINN, it can be observed that the output tends only to zero as the distance from the asteroid increases. In order to avoid

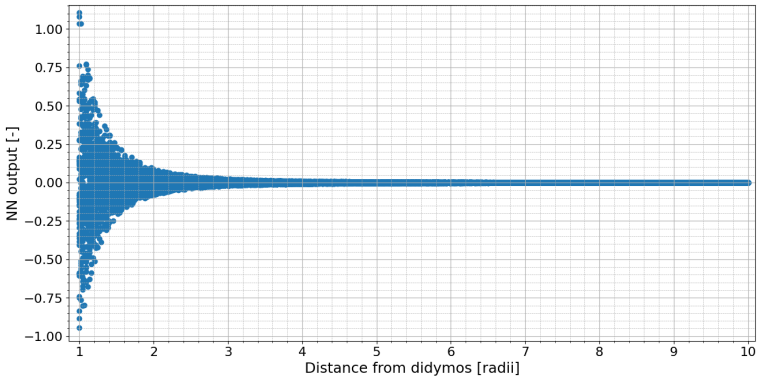
that, the output of the Neural Network is rescaled as:

$$U_{NN} = \frac{U^*_{NN}}{r^3} \tag{3.17}$$

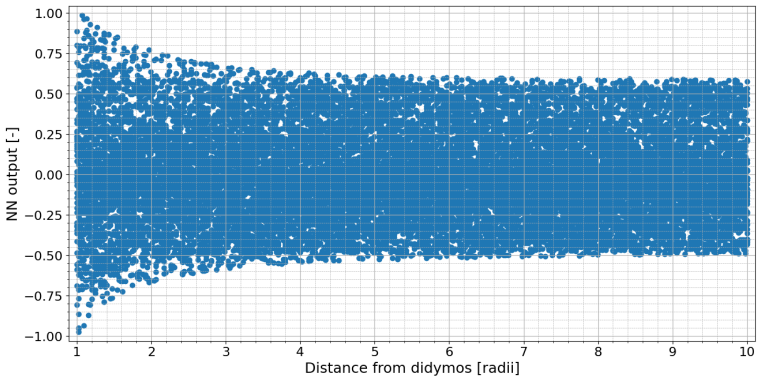
Where U^*_{NN} indicate the output of the PINN before rescaling.
In a similar way, the output of NN_0 is rescaled as:

$$a_{NN} = \frac{a^*_{NN}}{r^4} \tag{3.18}$$

The rescaling effect on the output can be seen in Figure 3.5. The field points considered in the figure are part of the test set described in Section 3.3.1. By implementing this rescaling, the performance of the PINNs increment at field points far away from the surface.



(a) Without rescaling



(b) With rescaling

Figure 3.5: Rescaling of the output of the neural network

3.2.7. Hyperparameters definition

A list of the hyperparameters used to train the PINNs is reported in Table 3.2.

Hyperparameters	Value
Optimizer	Adam
Initializer	Glorot normal
Activation function	GELU
Number of layers	8
Number of nodes	80
Number of epochs	10000
Learning rate Didymos	0.003
Learning rate Dimorphos	0.001
Batch size	5000

Table 3.2: Hyperparameters used to model and train the PINN

The GELU activation function is used due to its smooth high-order derivatives. If this condition is not true, the gradients of the network that are taken using automatic differentiation to enforce the physics constraints will no longer be well-behaved for gradient descent. It is then suggested to not use activation functions such as ReLu or Signum and instead the usage of activation functions such as GELU or Hyperbolic Tangent is recommended [22].
The learning rate is reduced with the increasing number of epochs in order to avoid jumping over the minima as suggested in [16, 22].
All the hyperparameters will be the same in the following simulations unless directly stated in singular cases. The hyperparameters are also common between the PINN models of Didymos and Dimorphos unless specified.
The number on nodes for each hidden layer of the network is the same. The number of data used for the training, validation and test is discussed in Section 3.3.

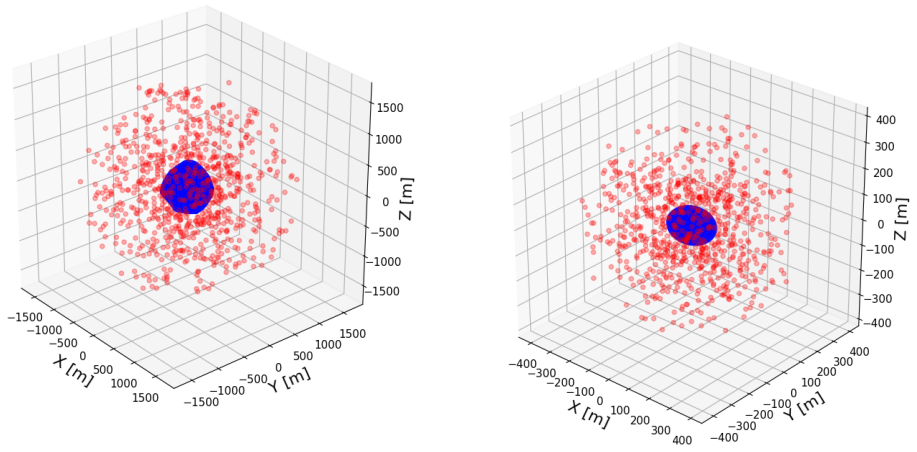
3.3. Data generation

In this section the data used to train the PINN are described. In particular, the two different cases described in Section 3.1.1 are taken into account.

3.3.1. Data generated from a known model

In case data are generated from a known model, the potential and the acceleration in a certain field point can be computed for each asteroid. In this case, two separate different dataset can be generated to train the PINNs. One dataset will be used to train the Didymos PINN while the other one will be used to train Dimorphos PINN.

Both dataset are built in a similar way. Some field points are generated randomly in proximity of the asteroid and the potential and acceleration is computed in those specific field points. In order to generate the field points, some random spherical coordinates are generated. Then, once these coordinates are computed, they are transformed in Cartesian coordinates expressed in the body frame. The radius is generated using a random uniform distribution from a minimum radius to a maximum one. More specifically, the radius of the field points used for the training in Chapter 4 are all generated between the minimum radius of the asteroid and 5 Brillouin radii of distance from the asteroid. The azimuthal angle is also generated randomly using a random uniform distribution between $[0, 2\pi]$. A random value between -1 and 1 is generated using a random uniform distribution and the arccosine of this value is computed. The polar angle will then be equal to this value. A representation of the field points generated in proximity of the asteroids is shown in Figure 3.6. Both figures are represented in the body frame of each asteroid.



(a) Didymos (b) Dimorphos

Figure 3.6: 1000 different field points used for the dataset in proximity of the asteroids

Once all the field points of the dataset are generated, the acceleration and potential are computed in all the field points for each asteroid individually using the reference models discussed in Section 3.1.1. An acceleration and a potential value will then be associated

to each field point generated and will be used for the training.

The training set, the validation set and the test set are all built using this method. The only difference is the number of field points considered. For the training set 10000 different field points were considered unless directly specified. In order to validate the model while training it, 5000 different data are generated. A total of 10000 points are generated in order to test the model. In this case, the radius is randomly generated using a random uniform distribution between the minimum radius of the asteroid and 10 Brillouin radii of distance from the asteroid in order to check if the PINNs are able to map the gravity field even when outside of the domain of the training. Two different training, validation and test set are generated with these number of data, one for each asteroid. All the field points are generated only in the exterior of the asteroids. A representation of the number of field points of the dataset as a function of the distance from the asteroid is made in Appendix B.

3.3.2. Data generated from simulated acceleration measurements

In case the PINNs are trained from simulated acceleration measurements, it is assumed that only the total gravitational acceleration is known in a field point. The PINNs would need to be trained using the same field points in order to impose that the sum of each acceleration provided by each asteroid is equal to the total acceleration in a certain field point. For this reason, the dataset used to train and validate both PINNs will be the same.
The field points are generated in a similar way as before. Two different sets of field points are considered. A set is composed of some field points generated randomly in proximity of Didymos while the other set is composed of field points generated in proximity of Dimorphos. In both cases, a random uniform distribution is used for the radius, for the azimuthal angle and for the arccosine of the polar angle. The ranges are the same used in the known model case. Once the field points are generated in proximity of both asteroids, a new single set is formed containing all the generated field points. This new single set of data will be the training set (or validation set). Each single data will contain the relative position of a field point with respect to both Didymos and Dimorphos with coordinates in the body frame of the same asteroid. The total acceleration and the angles of orientation of both asteroids are also added to the single data.

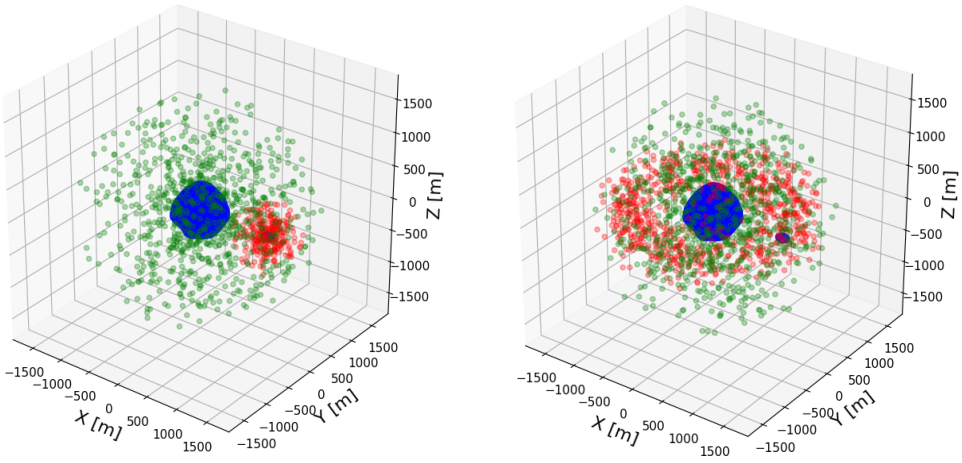
The PINNs trained with this method considers as a training set of data 10000 field points generated in proximity of Didymos plus 10000 field points generated in proximity of Dimorphos for a total of 20000 field points. The validation set is composed of 5000 data

taken in proximity of Didymos and 5000 data in proximity of Dimorphos. Finally, two different test set are considered. The first one is composed of 10000 field points taken only in proximity of Didymos while the second is composed of 10000 field points in proximity of Dimorphos.

The orientation and position of the asteroids is integrated in time using the parameters described in Section 3.1. Dimorphos is assumed at the pericenter of the orbit at the initial time and the angles of orientation are equal to zero degrees at the initial time instant. The total acceleration in a certain field point is computed as described in Section 3.1.1.

Each measurement is taken at a different time instant (the position of the asteroids and their orientation is different for every measurement). All the training data and the validation set are generated in a time domain between 0 and 250 hours to simulate the time needed to sample all the measurements. The time needed is chosen by assuming that a 30 seconds measurement time is required in order to measure each single acceleration. This value is in line with real instruments used in space applications [4]. The two test sets are generated in a time domain between 250 and 1000 hours. The time domain for the test set is chosen in this way in order to understand if the PINN is able to map the gravity field even for positions and orientations of the asteroids never seen before as they will vary with time.

A representation of the field points used to train the network is shown in Figure 3.7. Two different cases are represented. In the figure of the left, all the field points are taken at the initial time instant while on the right is indicated the case where the samples are taken in the time domain studied. In green are indicated the field points in proximity of Didymos while in red the field points in proximity of Dimorphos. The field points are represented in the inertial frame. As we can see, due to the motion of Dimorphos around Didymos, a torus of samples around Didymos is formed when considering field points in different time instances. The two test sets can be seen as the field points of a singular colour. A representation of the number of field points of the dataset as a function of the position in the inertial reference frame and as a distance from each asteroid is reported in Appendix B.



(a) same time instant (b) 250 hours domain

Figure 3.7: 1000 samples in proximity of Didymos and 1000 samples in proximity of Dimorphos

3.4. Summary of how to train

A quick recap on how to train the PINNs is reported below:

Algorithm 3.1 How to train the Network

1: Collect training data from a known model or from acceleration measurements
2: Non-dimensionalize the training data
3: **for** *epoch in* n_{epochs} **do**
4: Convert the input position (r=$norm([x, y, z])$, x/r=s, y/r=t, z/r=u)
5: Compute the output of both networks
6: Add the point mass model to both networks output
7: Re-scale the output of both networks
8: Auto differentiate the potential with respect to the cartesian coordinates expressed in the body frame to compute the accelerations
9: Compute the loss function
10: Update the parameters of the network
11: **end for**

4 | Analysis of the results

In the following sections, all the results of the training will be reported. All the models presented in this work are trained in TensorFlow on a NVIDIA Quadro P1000 graphic card.

4.1. Training with a known model

In this section it will reported the case where the models are trained from a known model. Different analysis are made in order to understand the performance of the PINNs in this case.

4.1.1. Comparison between the different loss fuctions

First of all, the different loss functions introduced in Section 3.2.4 are used to train the model in order to see which loss function makes the model perform best. This is done by training the same model with the same training and validation set described in Section 3.3.1 while changing only the loss function used for the training. All the models trained are then tested on the same test set in order to see which one performs best. This is shown in Figure 4.1 and 4.2. The gray region in the figures indicate the training data domain region. In the figures is represented the percentage acceleration error between the acceleration obtained by the network and the acceleration obtained from the reference model of the singular asteroid as a function of the distance from the same asteroid.

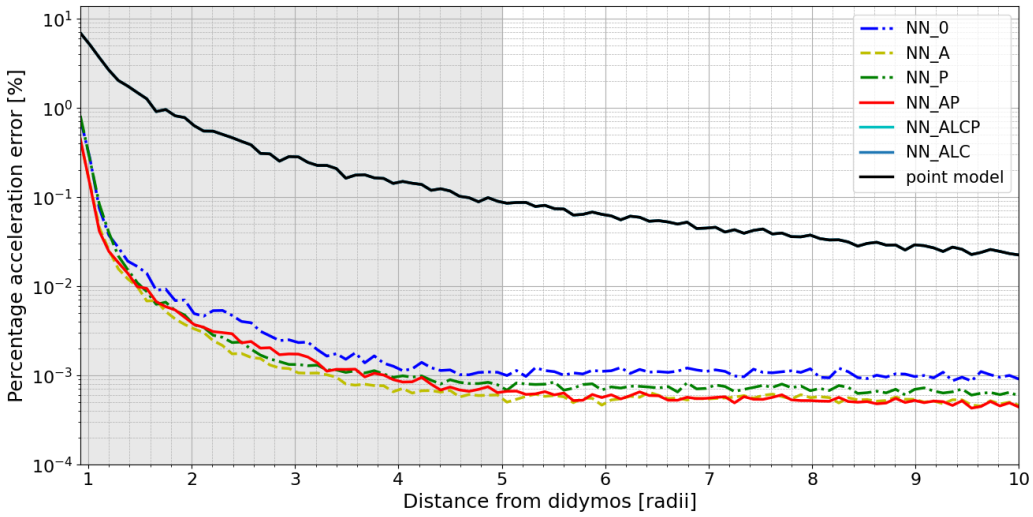


Figure 4.1: Comparison of different loss functions to map the gravity field of Didymos

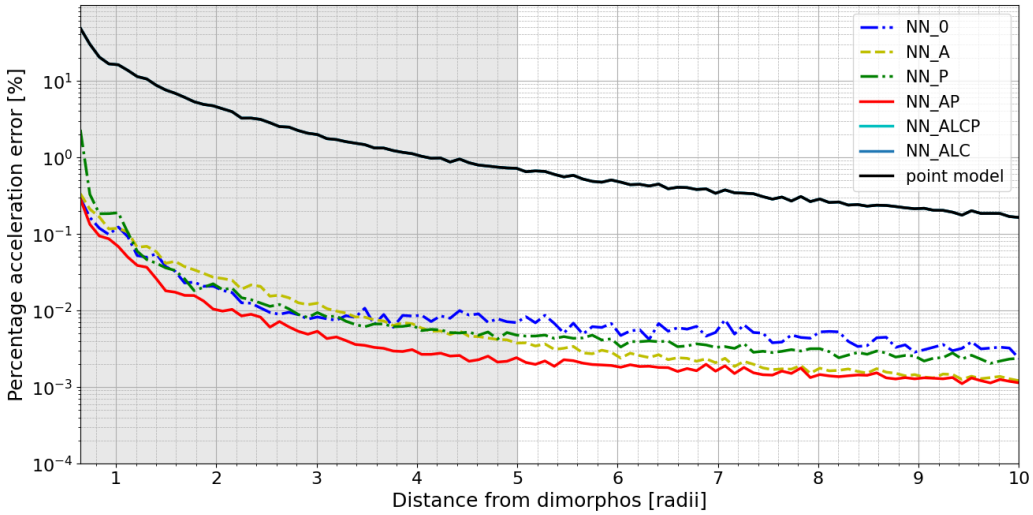


Figure 4.2: Comparison of different loss functions to map the gravity field of Dimorphos

As we can see, in all cases the model can approximate the reference model well even when considering field points outside the domain used for the training. However, in case the point mass model is included in the model, both NN_ALCP and NN_ALC converge at the same result and perform much worse with respect to other models. In particular they both converge at the point mass model. Between all the model used, NN_AP seems to perform slightly better with respect to the other models while the ANN seems to perform

worse with respect to all the loss functions considered (with the exception of NN_ALCP and NN_ALC).

It seems like that the NN_ALCP and NN_ALC models converge to the point mass model independently on the number of data and epochs used for the training. This behaviour can be understood once the variation of the laplacian and of the acceleration is studied as a function of time.

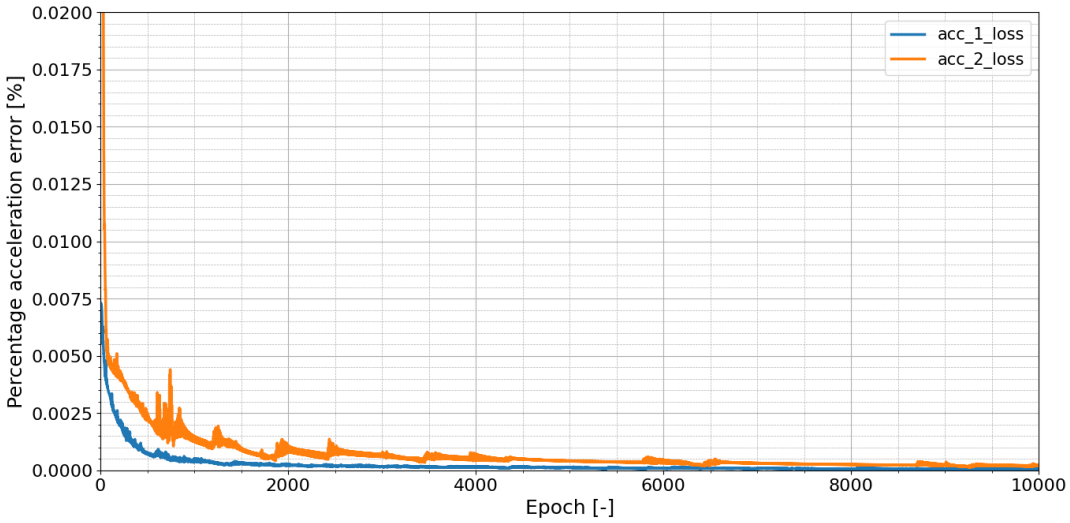


Figure 4.3: Variation of acceleration error with the number of epochs for NN_AP

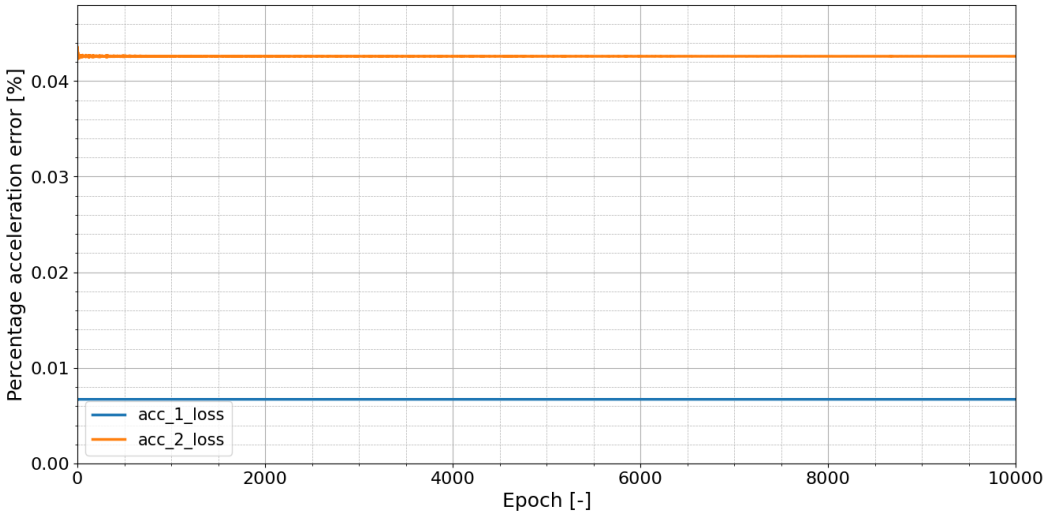


Figure 4.4: Variation of acceleration error with the number of epochs for NN_ALCP

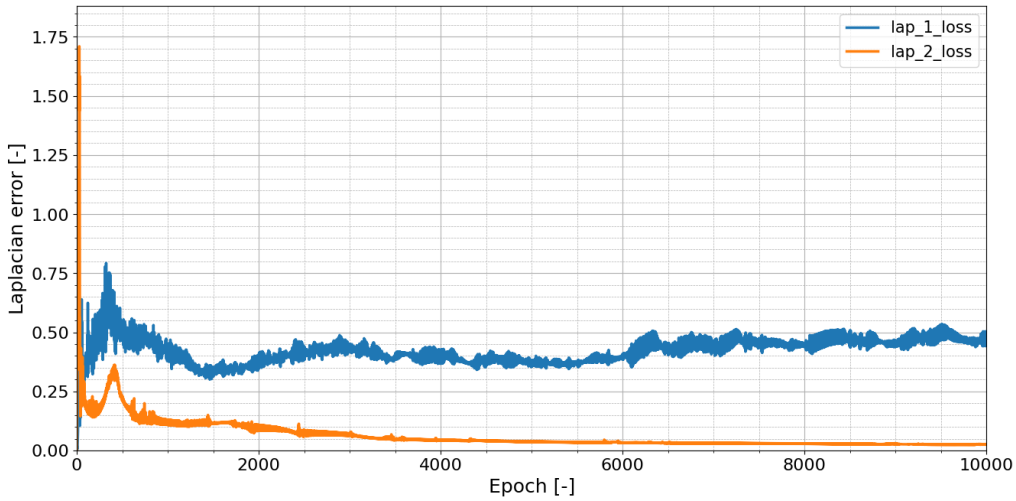


Figure 4.5: Variation of laplacian error with the number of epochs for NN_AP

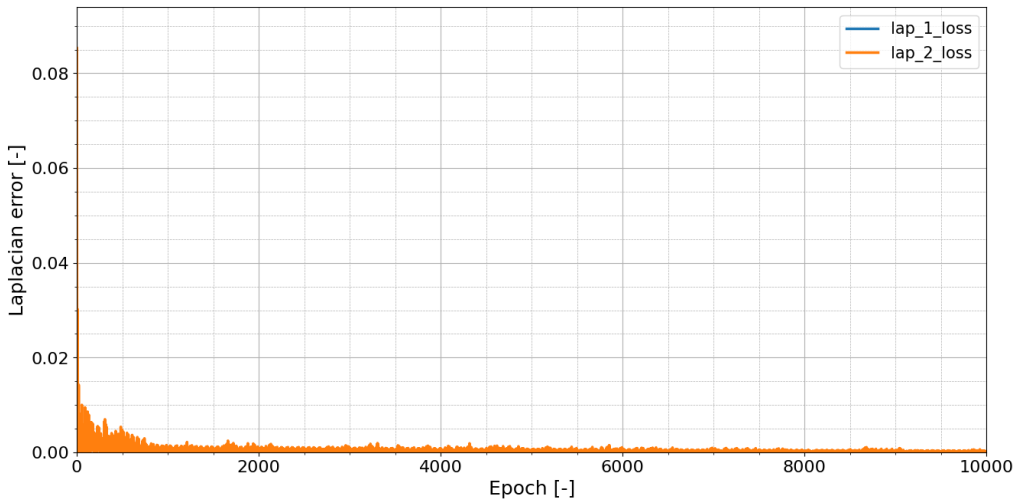


Figure 4.6: Variation of laplacian error with the number of epochs for NN_ALCP

It seems that the laplacian term in the loss function makes the point model a local minima as it satisfies all the physical properties of a gravity field. This will make the NN_ALC and the NN_ALCP models converge to the point mass model.

4.1.2. Adaptive weight

In order to avoid that the NN_ALC and NN_ALCP models converge to the point mass model, a system of self adapting weights applied to each term of the loss function can be

implemented. Different approaches to this problem are introduced in [2, 34]. However, by applying them at the loss function, worse results with respect to the other models were obtained. Some considerations on the variation of the weights can be made in order to understand how to vary them at each epoch. To start, the laplacian and curl weights are imposed equal to zero at the initial epoch in order to leave the point model local minima. The weights of the potential and of the acceleration loss term are imposed equal to 1 at the start. Using a trial and error approach, it seems like that using a small variation of the weight of the laplacian at each epoch increases the performances (a variation between 0 to 0.001 to the weight is considered). Using these considerations, the following adaptive loss weight is introduced:

$$\hat{\lambda}_i(t) = \frac{\left|\overline{\nabla_\theta L_a(t)}\right|}{\left|\overline{\nabla_\theta L_i(t)}\right|}, \quad i \in \{1, ..., k\} \tag{4.1}$$

$$\lambda_i(t) = \alpha\lambda_i(t-1) + (1-\alpha)\hat{\lambda}_i(t) \tag{4.2}$$

$\left|\overline{\nabla_\theta L_a(t)}\right|$ is the mean of the gradient of the acceleration loss function term with respect to the weights and biases of the network and where |...| denotes the elementwise absolute value. The subscript i indicates all the other terms of the loss function. t indicates the current epoch, $\lambda_i(t)$ is the weight for the corresponding loss term and α is an hyperparameter chosen based on how fast the variation of the weights shall be. This method derives from a modification of the adaptive loss weight introduced in [34]. A value between $[0.995 - 0.999995]$ is recommended in order to train the PINN.
A new model of PINN will be trained using the adaptive loss weight and it will referred from now on as PINN_custom. The variation of the laplacian with this method is represented in Figure 4.7 and a zoom of it is shown in Figure 4.8. The laplacian error in this case tends to zero although with worse performances with respect to NN_ALCP as in that case the error is zero due to the point model.

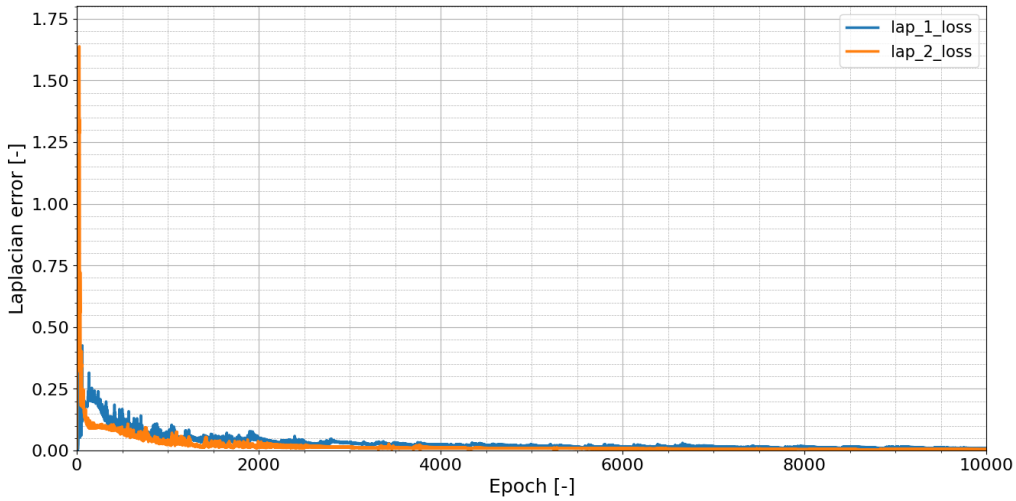


Figure 4.7: Variation of laplacian error with the number of epochs for NN_custom

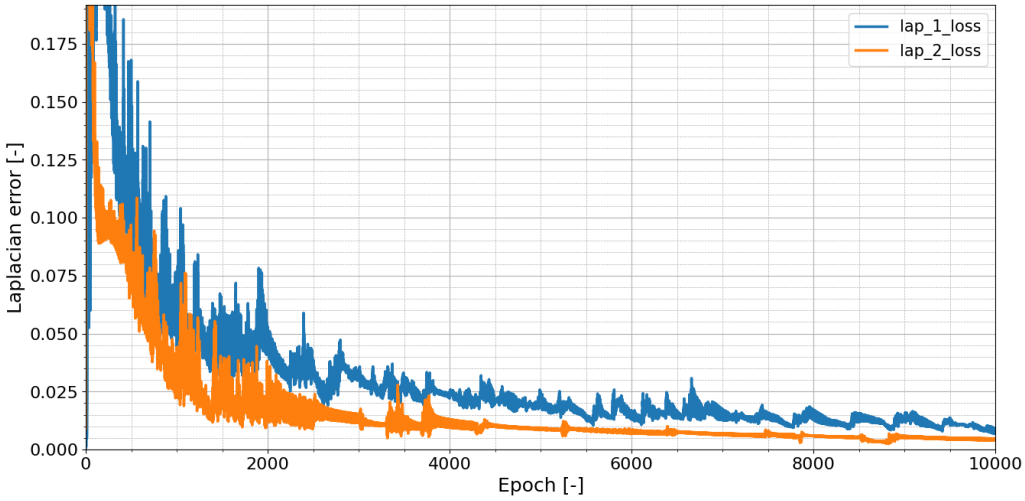


Figure 4.8: Zoom of the variation of laplacian error with the number of epochs for NN_custom

The performances of the model with PINN_custom included is shown in Figure 4.9 and 4.10.

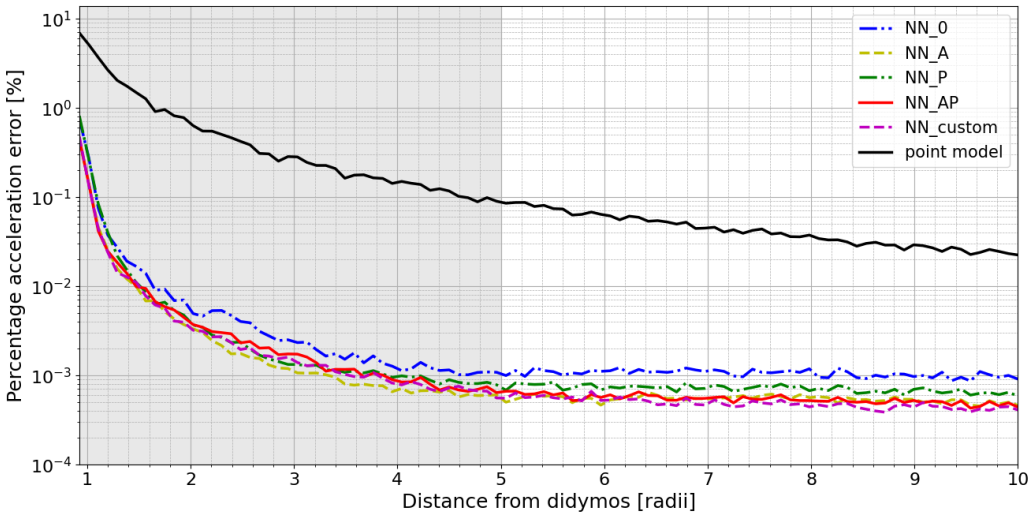


Figure 4.9: Didymos model trained with PINN_custom

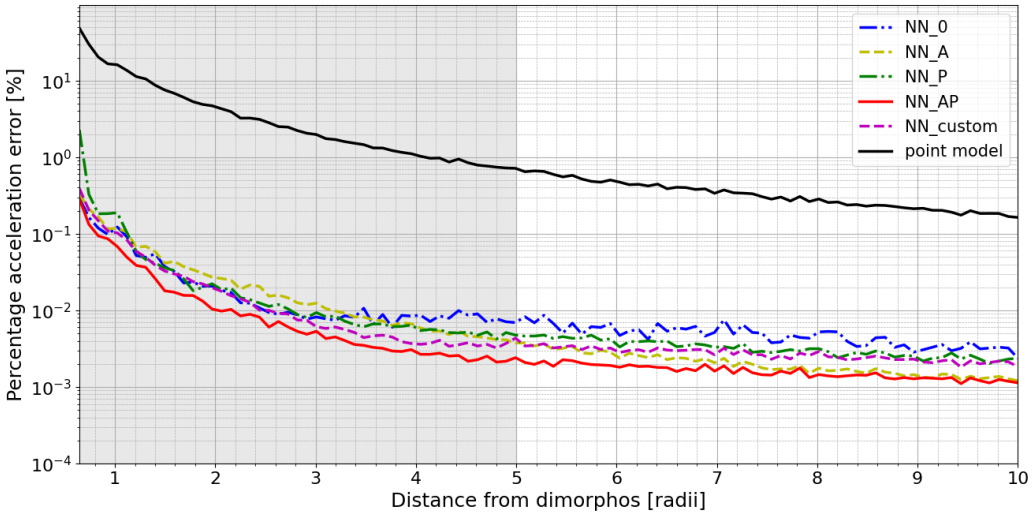


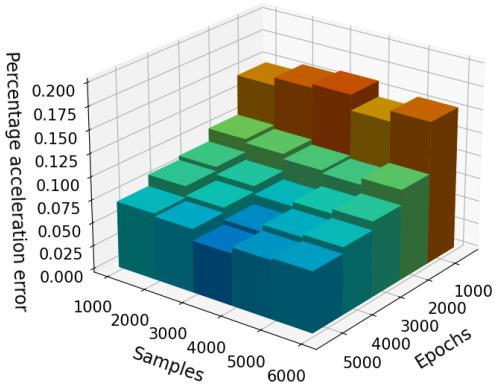
Figure 4.10: Dimorphos model trained with PINN_custom

As we can see, the new model performs in a similar way as the other methods. However, this loss function comes with a cost as the time needed for training the PINN is approximately 5 times higher with respect to the other models when using the same number of data for training. The memory needed to allocate all the data is also higher due to the computation of second order derivatives of the potential. It is then not recommended to use this loss function to train the PINN when training with a known model.

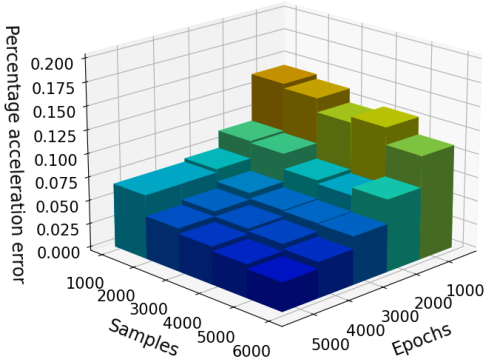
4.1.3. Hyperparameter analysis

An analysis on how many data and how many epochs are necessary to reach convergence can be made. This is done by training the same model (the network at the start has the same initial weights and biases) 25 different times by varying the number of training data used and the number of total epochs for each time. The number of nodes for each hidden layer is also varied in order to see how the performance varies with different model capacities. All the models were trained using NN_AP as a loss function. To evaluate the performance, a mean percent error is computed using 10000 different test samples generated in the training domain. The results are reported in Figure 4.11 and 4.12.

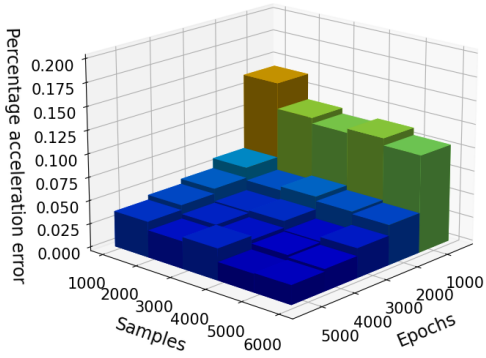
For both Didymos and Dimorphos, a small set of data is sufficient to guarantee the convergence of the model. Incrementing the number of nodes up to 40 for layer for Didymos seem to increment the performance. Between 40 and 80 nodes the performance are pretty similar. For Dimorphos this behaviour is also true considering 20 nodes for layer. Thus, the number of nodes can be reduced in order to save on memory while maintaining similar performances. It seems that 2000 training data and 2000 epochs are sufficient in both cases to reach convergence when considering enough nodes for layer. Increasing the number of training data and epoch above these values seem to slightly reduce the mean error.



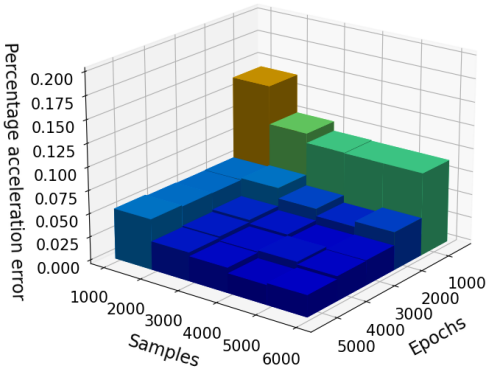
(a) 10 nodes



(b) 20 nodes

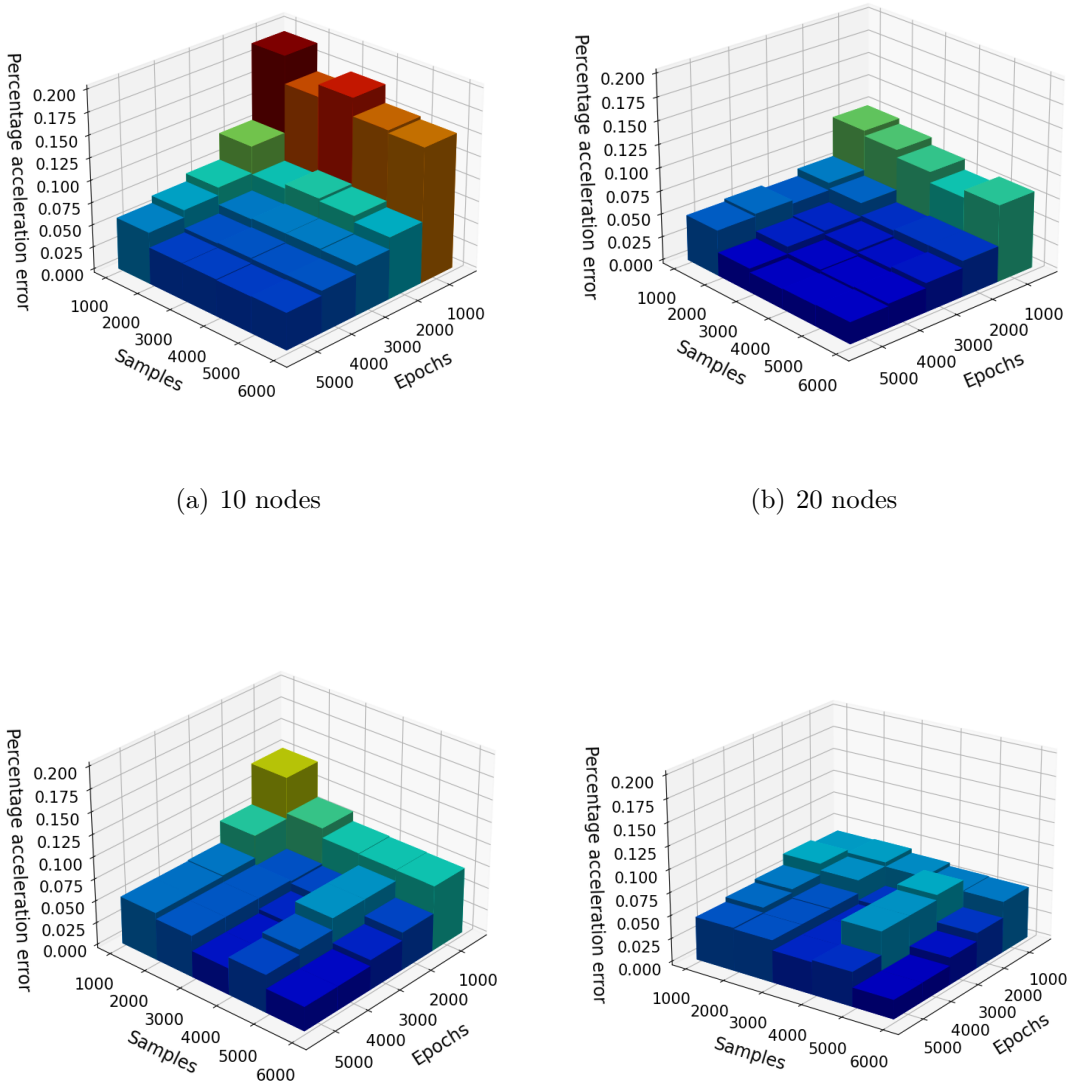


(c) 40 nodes



(d) 80 nodes

Figure 4.11: Study of hyperparameters Didymos



(a) 10 nodes

(b) 20 nodes

(c) 40 nodes

(d) 80 nodes

Figure 4.12: Study of hyperparameters Dimorphos

4.1.4. Neural networks compared to traditional methods

A comparison of the PINN methods implemented in this work with the traditional methods exposed in Section 2.1 can be made in order to understand which method can approximate better the gravity field of the asteroids. In this case, the PINNs and the ANNs are trained for 30000 epochs using 100000 different training data generated in the same

way as described in Section 3.3.2. The validation and the test set are composed by the same number of field points as before. The loss function used for the training is NN_AP in the case of PINN. The number of samples in the dataset and the number of epochs was highly increased with respect to the previous cases in order to understand how much the network increases in performances with respect to the previous case. The polyhedral and the ellipsoid models are used as a reference, the mascon model is created by dividing the polyhedron in a collection of tetrahedra and assigning the point mass in the center of each tetrahedra as described in [5]. In particular, a single mass was considered per tetrahedra for mascon model, while 3 masses where considered for mascon model 3. For the spherical harmonics, the spherical harmonics coefficients were computed using the method described in [36]. A spherical harmonics of order 2 and order 8 were considered.

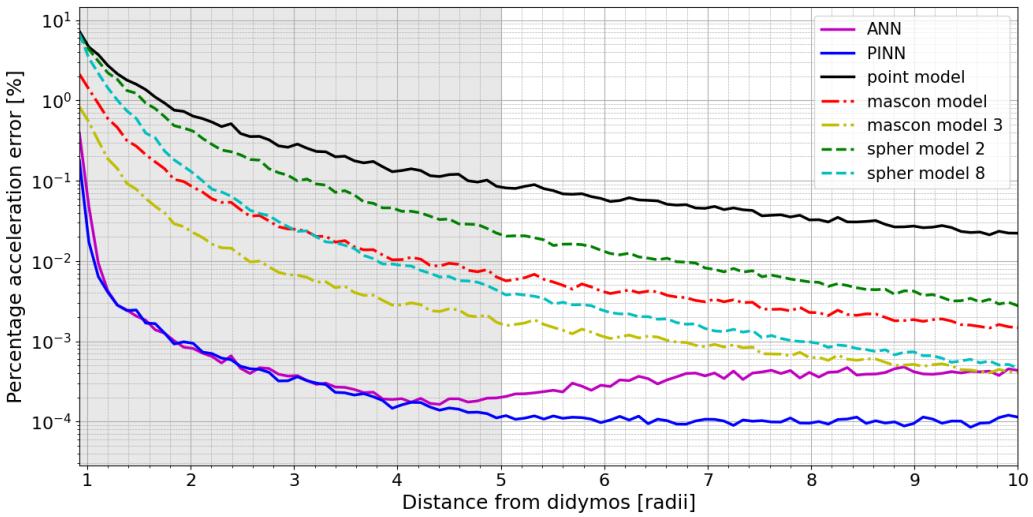


Figure 4.13: PINN Didymos model compared with traditional models

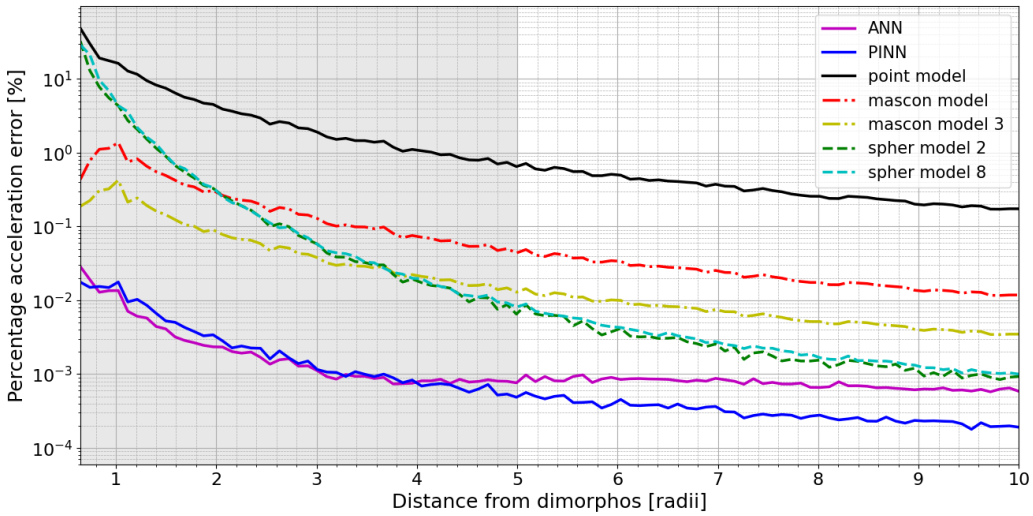


Figure 4.14: PINN Dimorphos model compared with traditional models

As we can see, the PINN methods performs better with respect to the other models in proximity of the surface. As the distance from the asteroid increases, it seems that the spherical model gets closer with the performances and at distances near 10 radii the spherical model has errors in the same order with respect to the PINN model. However, it should also pointed out that at these distances the point model could be already sufficient to plan operations [15]. The error of PINN and of ANN is the same in the training domain. However PINN performs better with respect to ANN when outside of the domain.
The same conclusions can be made when comparing the potential percentage error of the different models as shown in Figure 4.15 and 4.16.

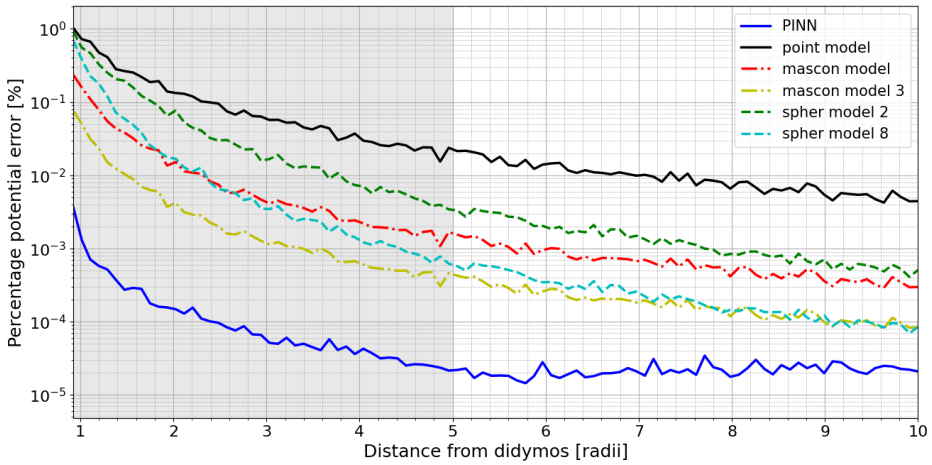


Figure 4.15: PINN Didymos model potential compared with traditional models

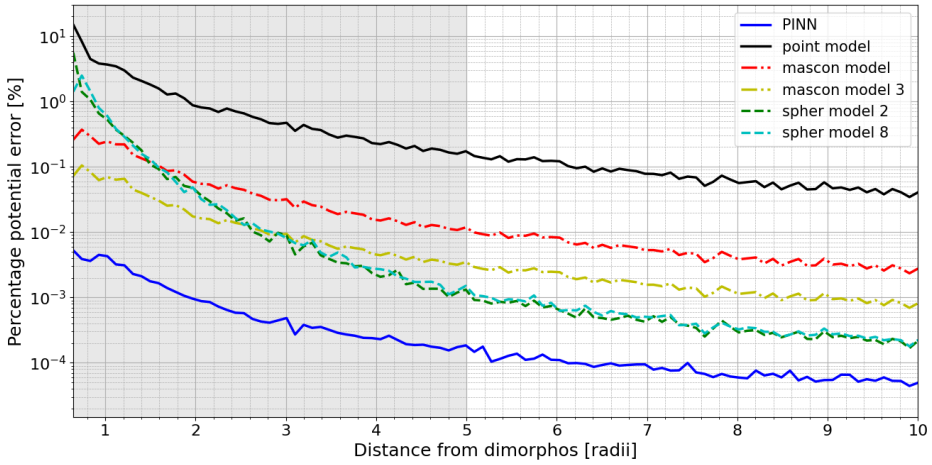
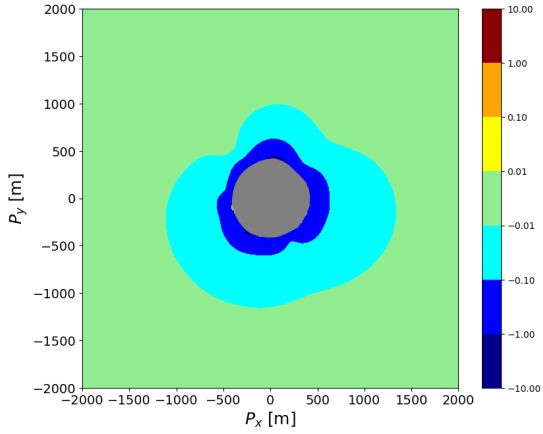
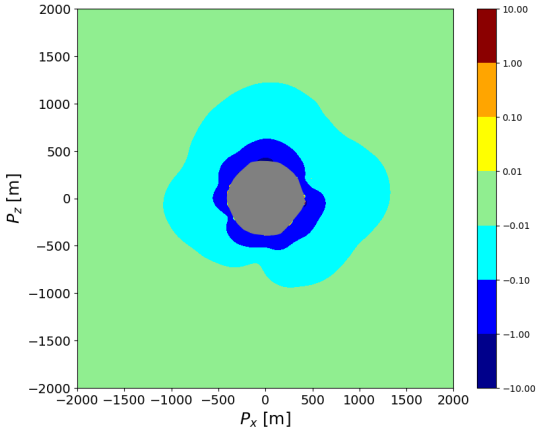


Figure 4.16: PINN Dimorphos model potential compared with traditional models

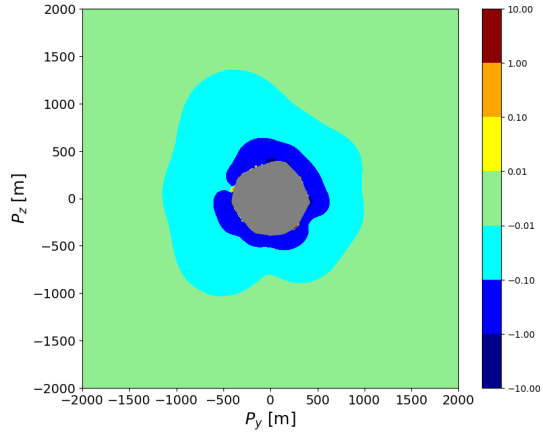
The performance of the different models can be compared also in singular field points to see if the PINN model do not map well in certain regions of space with respect to the traditional models. To do that, the percentage error of the PINN model and of the traditional methods with respect to the reference model are computed. Then, the difference between the PINN error and the traditional method error is computed in each point. If the value obtained is negative (in the graph is represented in blue), the PINN model performs better with respect to the traditional method and vice versa (red in this case in the graph). In case the same performances are obtained, it is indicated in gray. The field points are expressed in the body frame of each asteroid.



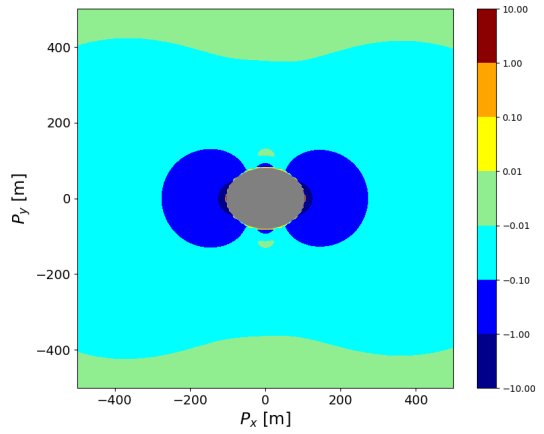
(a) xy plane Didymos



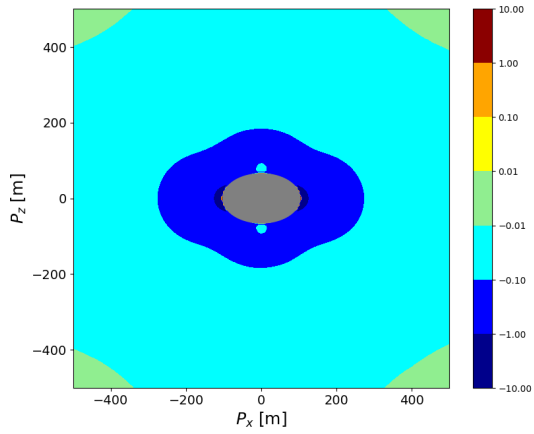
(b) xz plane Didymos



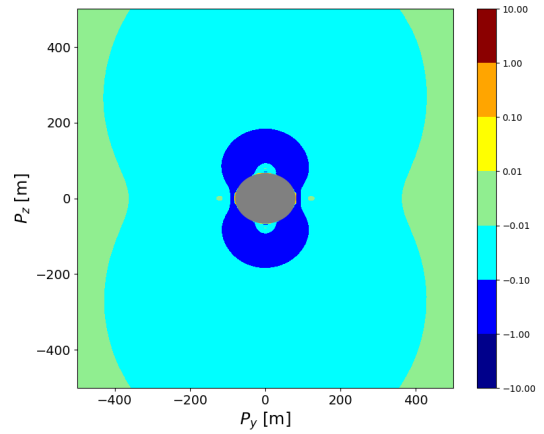
(c) yz plane Didymos



(d) xy plane Dimorphos

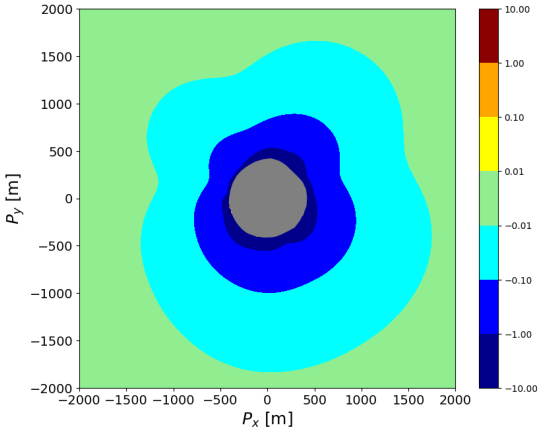


(e) xz plane Dimorphos

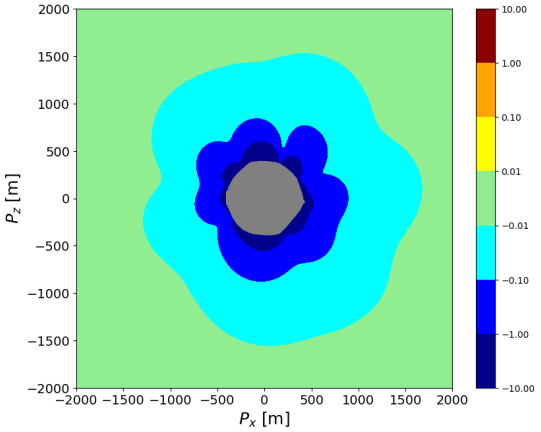


(f) yz plane Dimorphos

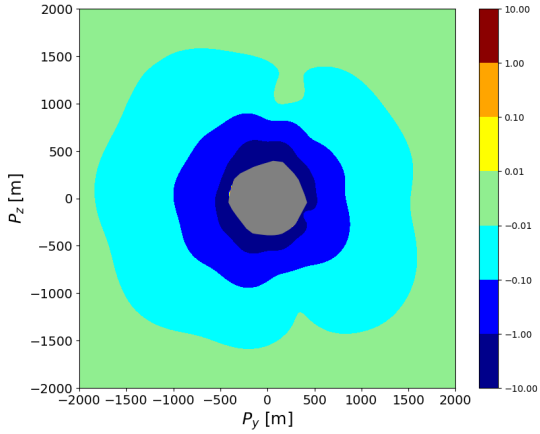
Figure 4.17: PINN-mascon model errors



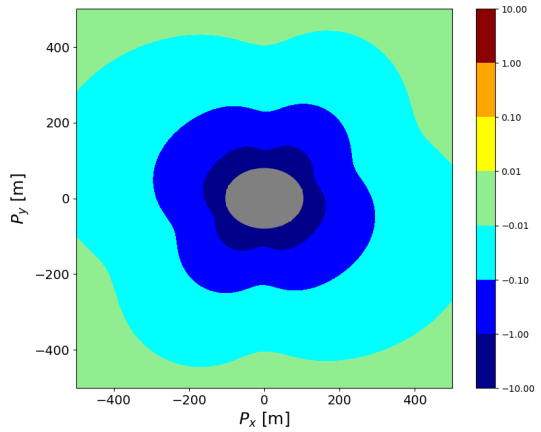
(a) xy plane Didymos



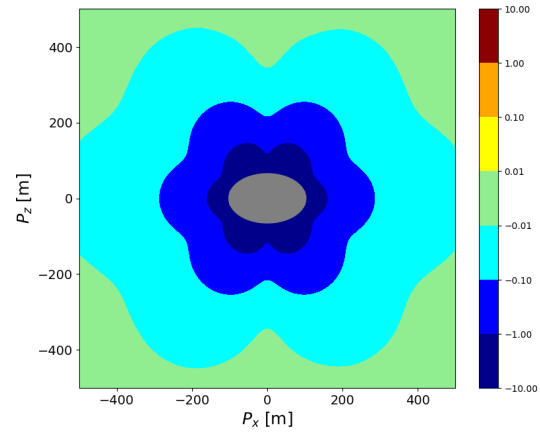
(b) xz plane Didymos



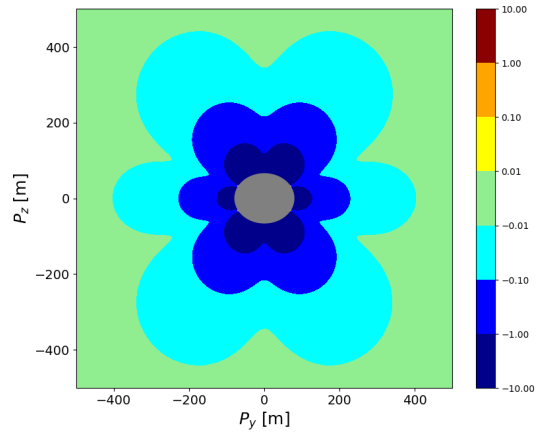
(c) yz plane Didymos



(d) xy plane Dimorphos



(e) xz plane Dimorphos



(f) yz plane Dimorphos

Figure 4.18: PINN-spherical harmonics model errors

Also in this case NN_AP is used to train the PINN. The PINN model performs better or at most equal with respect to the other models in most of the field points considered. In some small regions in proximity of the asteroids it seems that the mascon model performs slightly better. To improve the performance of the PINNs some additional samples can be taken in proximity of the surface in order to decrease the error as discussed in Section 4.1.6.

4.1.5. Time comparison between neural networks and traditional methods

The computational time can be assessed in order to understand which method compute faster. In Section 2.2.2, it was said that, given 10000 different samples all at once, PINNs seem to compute much faster with respect to other SOTA methods. However, in real space mission applications, usually the acceleration of a field point is computed individually instead of giving a batch of field points. To simulate this, 1000 different field points are generated and the acceleration is computed for each point individually. The time to compute each single acceleration is measured and the mean time to compute the acceleration is retrieved. The comparison of the computational speed for all the methods is reported for both Didymos and Dimorphos in the tables below. The computer used for the time comparison has an NVIDIA Quadro P1000 as a graphic card and an Intel core i7-8850h CPU. These computer components are not representative of a space-qualified processors. Obviously the computational times reported here could vary depending on the specifics of the computer used and different conclusions could emerge.

Model	Time needed [s]
Point	3.52 e-6
Polyhedral	2.49 e-1
Mascon	1.07 e-2
Mascon 3	3.19 e-2
Spherical 2	1.01 e-4
Spherical 8	6.09 e-4
NN_0	3.21 e-3
PINN	5.50 e-3

Table 4.1: Didymos time computation

Method	Time needed [s]
Point	3.84 e-6
Ellipsoid	5.63 e-3
Mascon	1.66 e-2
Mascon 3	4.97 e-2
Spherical 2	1.00 e-4
Spherical 8	6.12 e-4
NN_0	3.19 e-3
PINN	5.51 e-3

Table 4.2: Dimorphos time computation

The tables take only in consideration the time needed to compute the acceleration once the field point is given. The time of training for the network, the time needed to compute

the spherical harmonics coefficients and to compute the masses of the mascon model is not taken in account. In the tables, PINN refers to all the models that compute the acceleration through automatic differentiation of the potential. Due to automatic differentiation needed, they perform slightly slower with respect to the ANN. Changing the number of parameters of the network do not seem to modify the computational time too much (the time order remains the same). The mascon models in the case of Dimorphos takes more time with respect to Didymos as the number of point masses considered is greater with respect to Didymos as the polyhedral model used for Dimorphos has more faces. It seems that, by computing accelerations of the field points one at a time, the neural networks are no more the fastest model. Besides the point model, the spherical harmonics perform the fastest. It should also be noted that the computational time of the PINN and of the ellipsoid model is pretty similar. It would then be unnecessary to train the PINN in the case of Dimorphos as the PINN model would take the same amount of time to compute the accelerations with respect to the reference model and would only perform worse with respect to it. A combination of PINN and spherical harmonics or point model could be used in order to compute the acceleration of a field point. PINN could be used to compute the acceleration of field points in proximity of the surface while faster models could be used when they are located far away from the asteroid. In this way, the accuracy would still be high while the overall computational speed is increased.

4.1.6. Proximity error

One problem that remains is that in proximity of the asteroid the performance of the network becomes worse, especially in the case of Didymos. The difference in the performances for Didymos and Dimorphos in proximity of the surface could be explained by the model used as a reference. In fact, Dimorphos is modeled as a symmetrical object (unlike Didymos that is modeled as a non-symmetric polyhedron). This could cause an easier mapping for the network.
It could be that the network gets worse just because is in proximity of a boundary. Increasing the size of the boundary might result in improved mapping of the network. Instead of generating data up to the surface, some field points taken from inside the body were added to the domain of the training data with the domain of the radius between 0.5 and 5 Brillouin radii. The laplacian loss function term tries to approximate the Poisson's equation instead of the Laplace's equation when inside of the asteroid.
However, it seems like that the expansion of the domain does not increase the performances of the network in proximity of the surface. In particular, in the case of NN_custom, the performances gets worse as it needs to approximate a discontinuous function due to the

variation of the laplacian term when inside and outside of the asteroid.

Another way to approach the problem could be to increase the number of data in proximity of the surface. For example this can be done using an exponential distribution of the field points in proximity of the surface. The maximum of the distribution is at the surface of the asteroids. The data generated from the exponential distribution are added to the ones generated normally. An exponential distribution is used in order to avoid a jump in the number of samples. Two different networks can be trained using NN_AP, one network will be trained using 100000 data generated uniformly as described in Section 3.3.2 and it is called PINN constant, while the other network is trained with a training set containing 20000 data generated uniformly and 80000 data that are generated in proximity of the surface (PINN proximity). The performance of the network is shown in Figure 4.19 and 4.20. Both networks were tested on the same test set generated as usual.

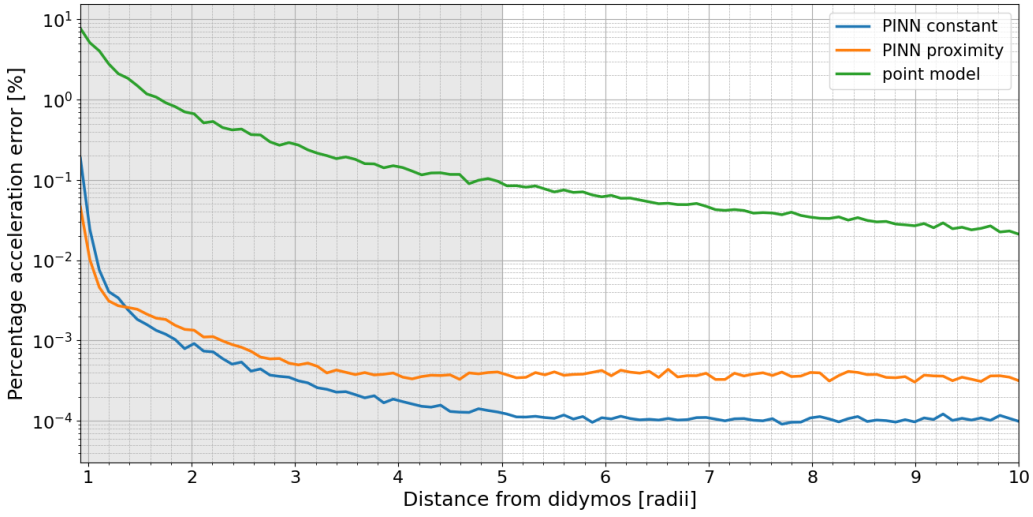


Figure 4.19: Didymos model trained with the addition of proximity data

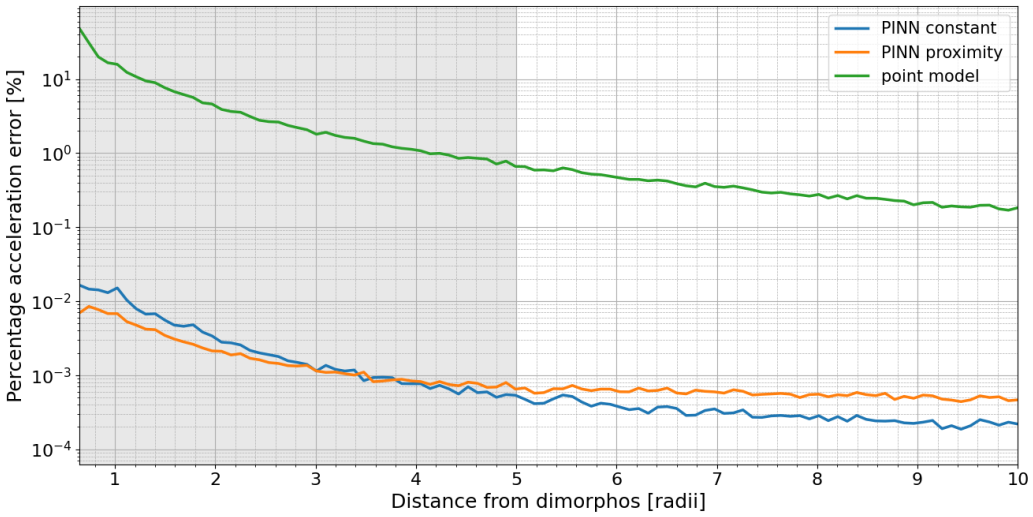


Figure 4.20: Dimorphos model trained with the addition of proximity data

In this example, an exponential distribution of the radius was defined as:

$$f(r) = 0.015e^{-0.015(r-r_{min})}$$

Where r_{min} is the minimum radius of the asteroid. The distribution of the samples as a function of the distance of the asteroid is shown in Appendix B. As we can see, using data in proximity of the surface does increase the performance in that region. However, an obvious drawback is that the performance in the rest of the domain gets worse when using the same number of data. By increasing the coefficient 0.015, the performance in proximity of the surface increases while it decrease the performance when far away from the asteroid. The generation of data in proximity of the surface would then depend on the requirements of the mission. If operations in close proximity of the surface are required, it is suggested to generate them, otherwise not. Similar results could be obtained by changing the loss function in order to give more weight to field points in proximity of the surface. This happens for example when using a MSE instead of the percentage error as a loss function.

4.2. Training with acceleration measurements

In this section the PINNs are trained from total acceleration measurements. The data used for the training and for the testing are discussed in Section 3.3.2.

4.2.1. Model training with and without a point mass reference

First of all, the model is trained without any information on the mass and is compared to the model with mass assigned. This is done in order to understand if it would be feasible to map the gravity field of a binary system from real measurements without any information about the mass of the asteroids and to see if there are any benefits to study the asteroids properties before the mission. This could be useful for future missions where no prior estimation of the properties of the binary system is made. All the different loss functions are tested in order to understand which model is the best one (only the loss function that do not contain the potential are tested as it is assumed unknown). In the figures below, the total percentage acceleration error is computed on the two test set discussed in Section 3.3.2. The model is trained and validated with the dataset presented in Section 3.3.2. The gray area in the graph indicates also in this case the region of the training data domain.

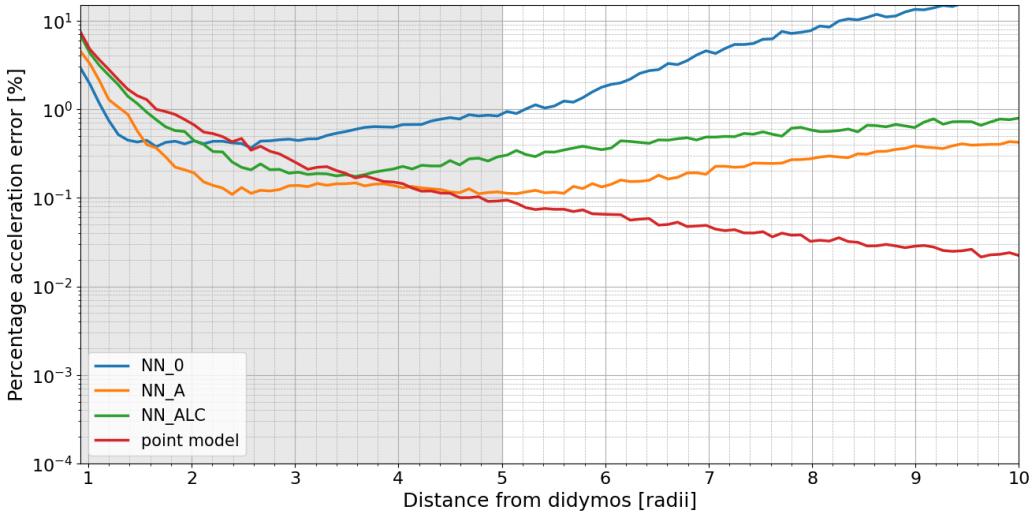


Figure 4.21: Model trained without the point mass model in proximity of Didymos

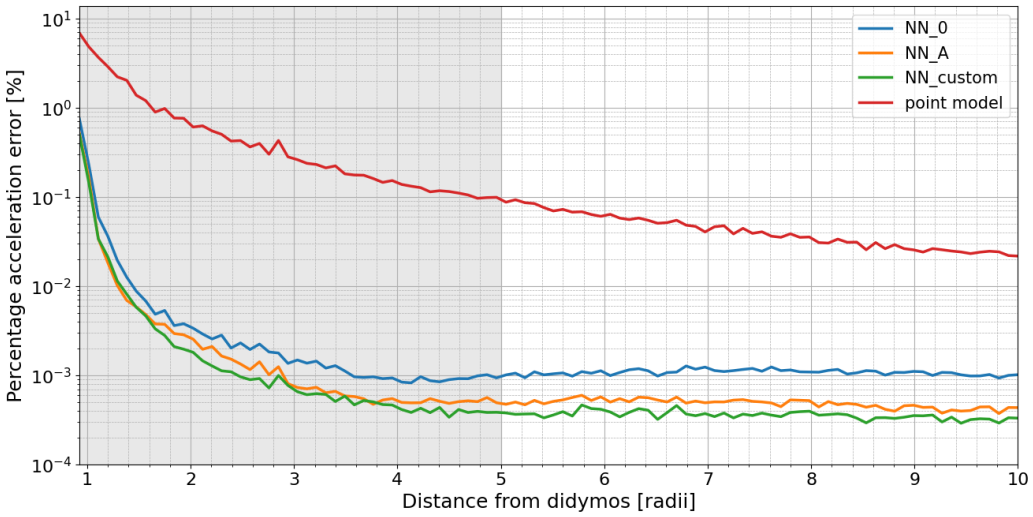


Figure 4.22: Model trained with the point mass model in proximity of Didymos

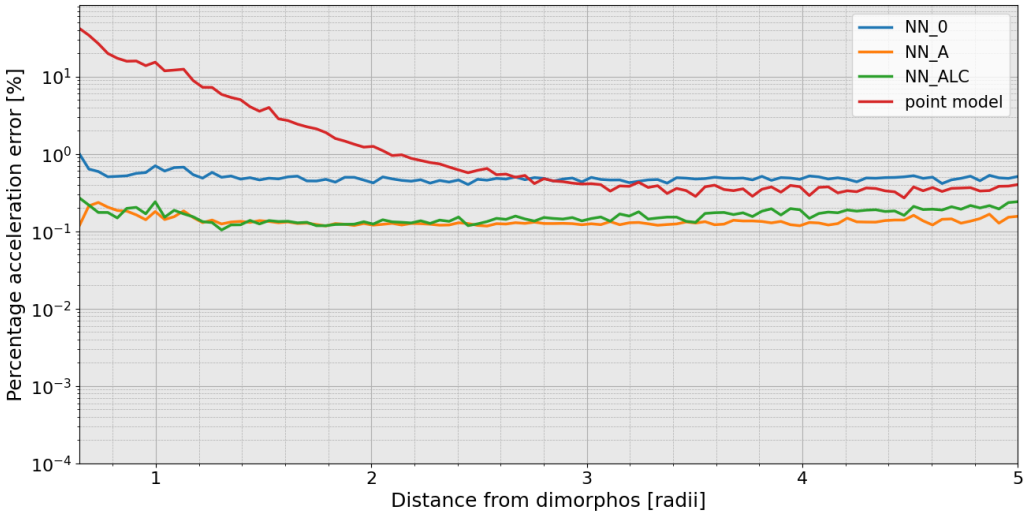


Figure 4.23: Model trained without the point mass model in proximity of Dimorphos

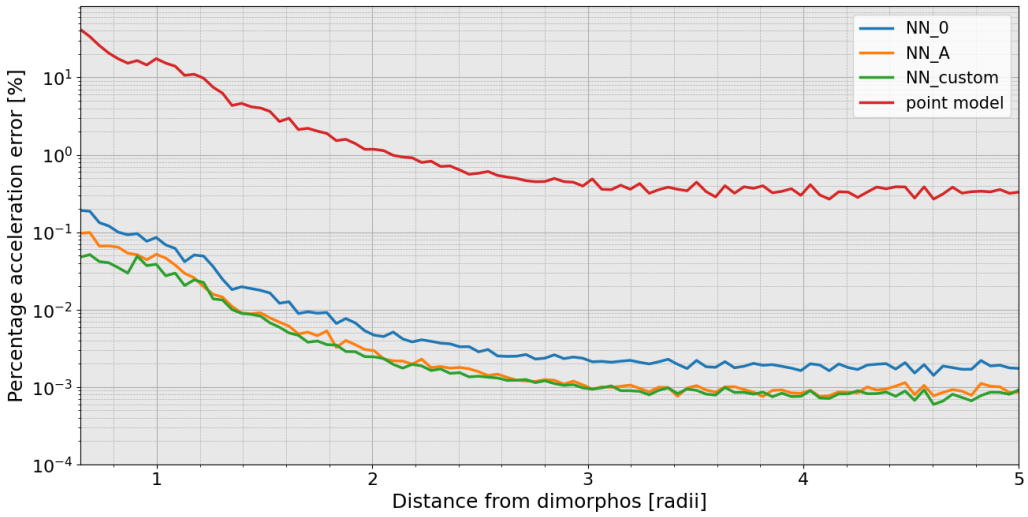


Figure 4.24: Model trained with the point mass model in proximity of Dimorphos

As we can see, when the mass is assigned, the model performs much better. Estimation of the mass of the binary system would then be advised in order to obtain better performances. Also in this case, NN_ALC converges at the point mass model when the point model is prescribed due to the same reasons explained in Section 4.1 and it is omitted. NN_A and NN_custom seem to have the same kind of performance and both models result in a better approximation with respect to NN_0 in the case of the mass assigned. NN_A would then be recommended in order to train the PINNs as NN_custom has higher training time and more memory is needed for the training as discussed in Section 4.1. The small peak in error near 3 radius from Didymos is due to Dimorphos. The plot in proximity of Dimorphos can be seen as a zoom of that region.

The results obtained are pretty similar to the ones obtained from the case where each network was trained with its own model as shown in Figure 4.25 and 4.26. Both models are trained with the same number of data and are tested with the same test set. In this case the models are tested only on the percentage acceleration error due to a singular asteroid in the same way done in Section 4.1 in order to see if, when PINNs are trained from total gravitational acceleration measurements, they are still capable of mapping the gravity field of the singular asteroid.

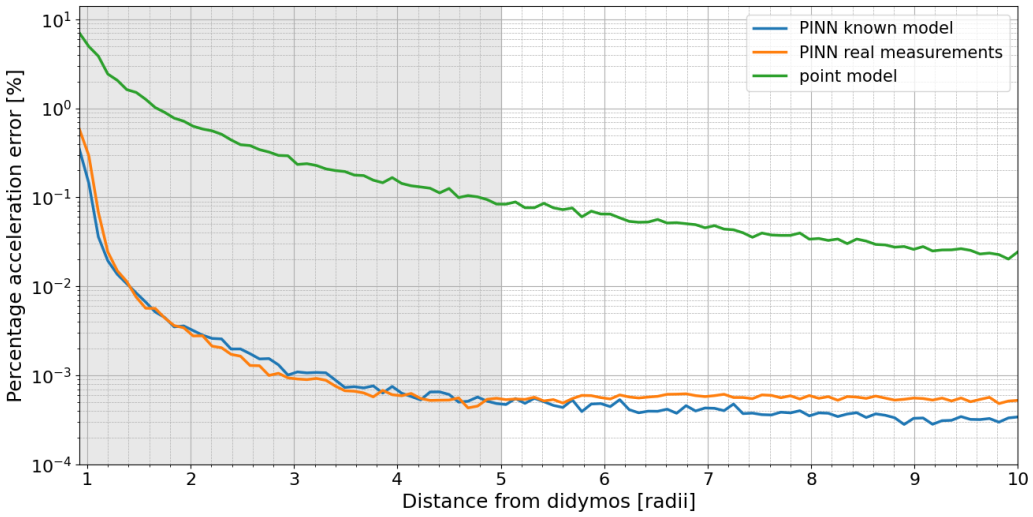


Figure 4.25: Comparison between PINN trained with and without a known model in proximity of Didymos

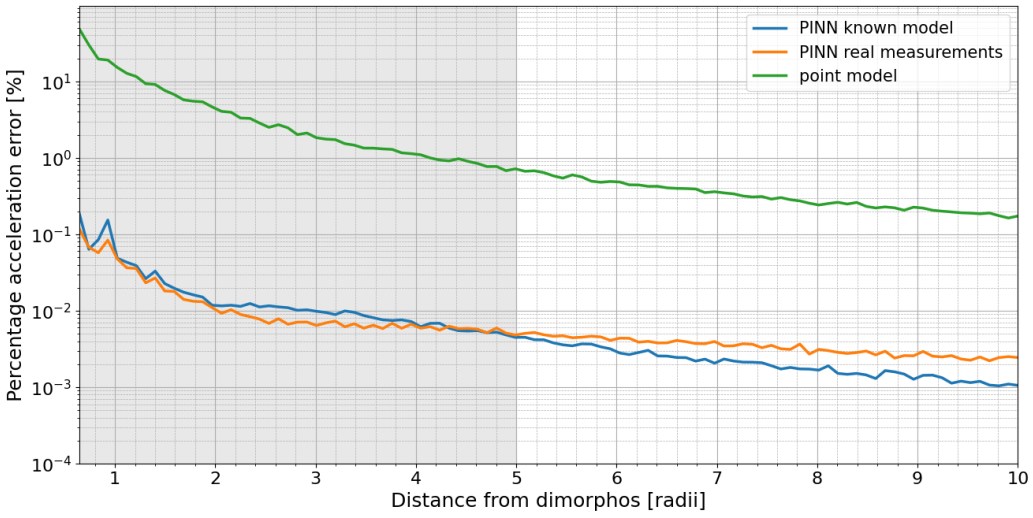


Figure 4.26: Comparison between PINN trained with and without a known model in proximity of Dimorphos

4.2.2. Mass estimation errors

By training the PINNs using the point model as a reference, one problem arises. It could be that the mass that we have used is not the same of the real asteroid. By assuming an error on the density of 350 kg/m^3 for each asteroid [9], and by computing the mass from

the shape of the reference model with this new density, the model can be trained with this new wrong mass. The effect of this error on the performances of the PINNs will be as follows:

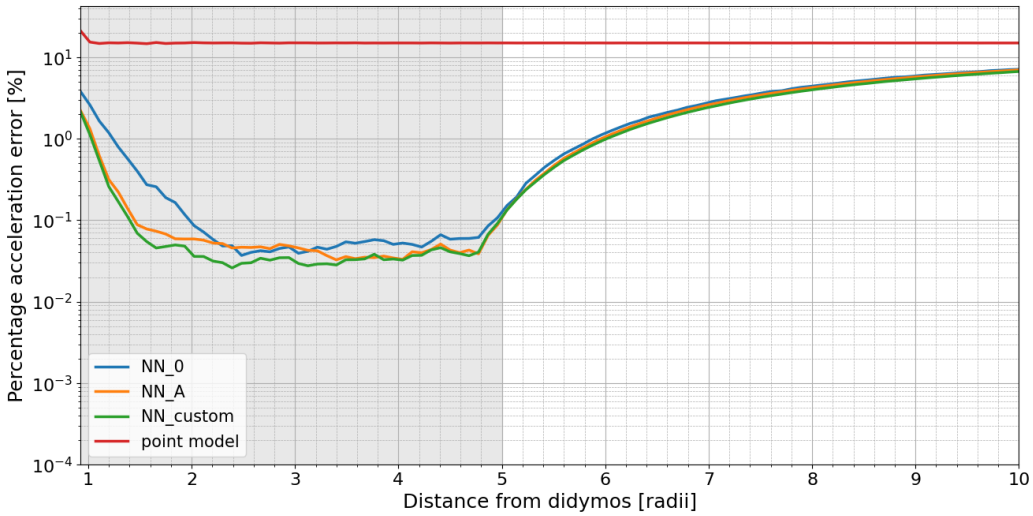


Figure 4.27: Model trained with a wrong point mass model in proximity of Didymos

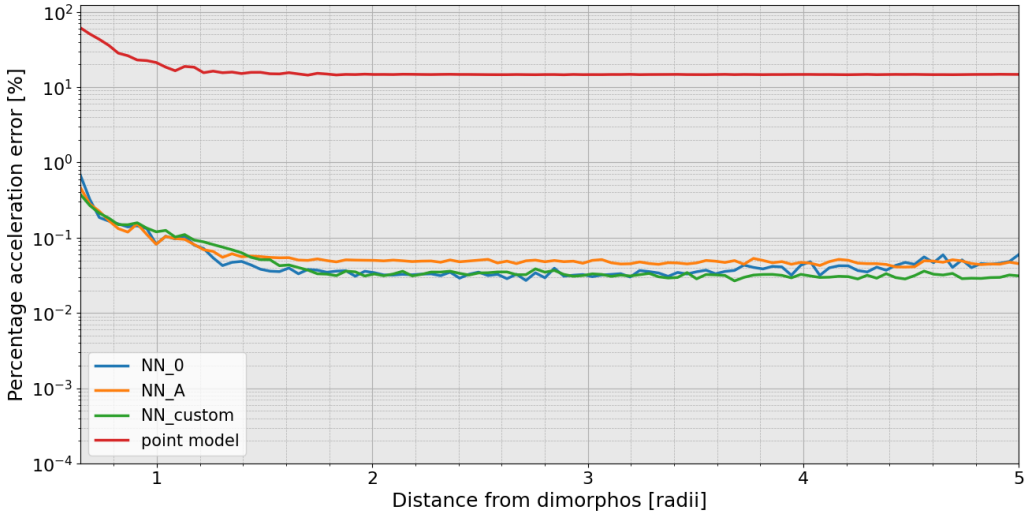


Figure 4.28: Model trained with a wrong point mass model in proximity of Dimorphos

As we can see, the performance of the PINNs gets worse, especially when outside of the training data domain. This is due to the output of the network that it is no more between $[-1, 1]$ as described in Section 3.2. This can be seen in Figure 4.29.

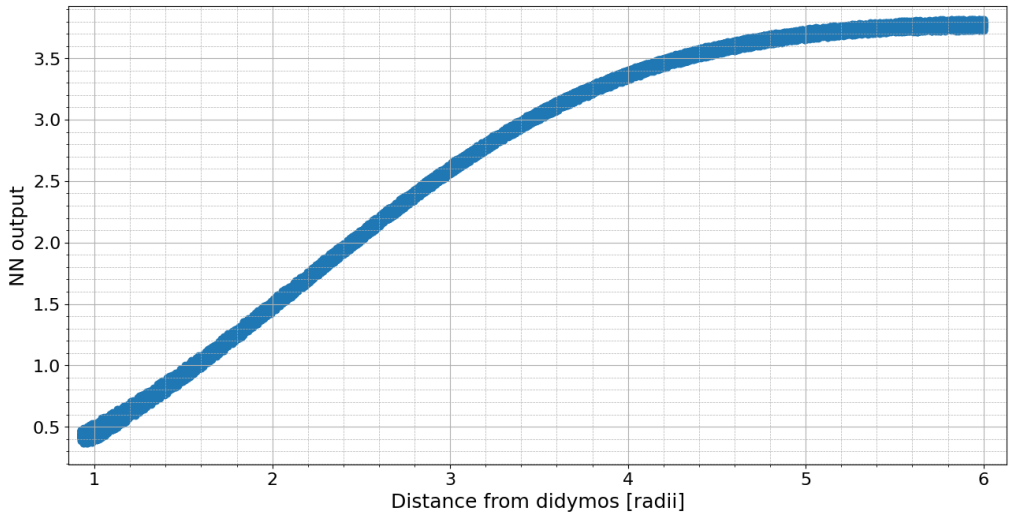


Figure 4.29: Output of the network with a wrong mass assigned

Before training the model, an estimation of the mass of each asteroid shall be made in order to reduce this error. For example, the error between the acceleration computed with only the contribution of the point mass models with a guessed mass and the acceleration obtained from real data measurements can be imposed as small as possible by varying the masses. In Figure 4.30 and 4.31, the performance of the network with a prior estimation of the mass (PINN mass corrected) is compared with the model trained with the exact mass estimation (PINN exact mass). The PINN considered in this training is NN_A.

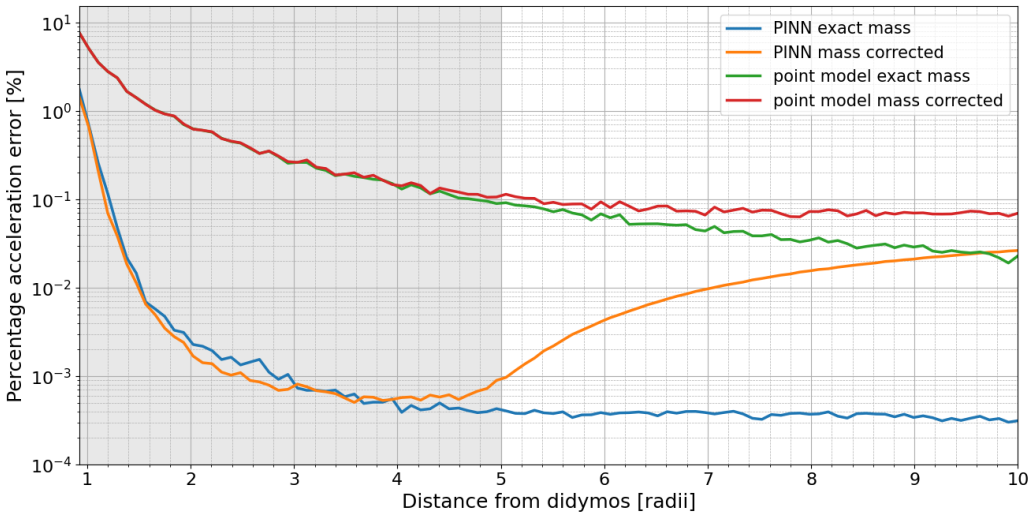


Figure 4.30: Model trained with the estimated mass model in proximity of Didymos

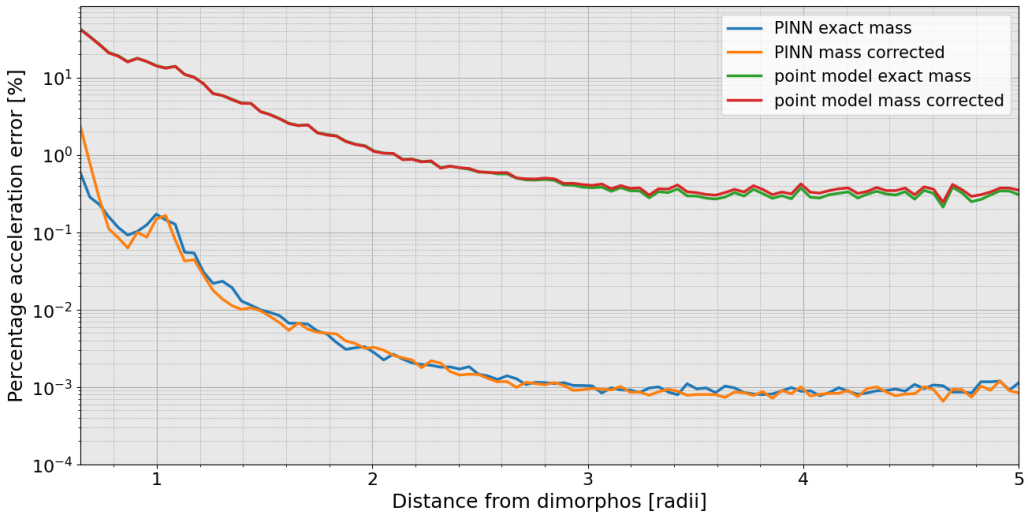
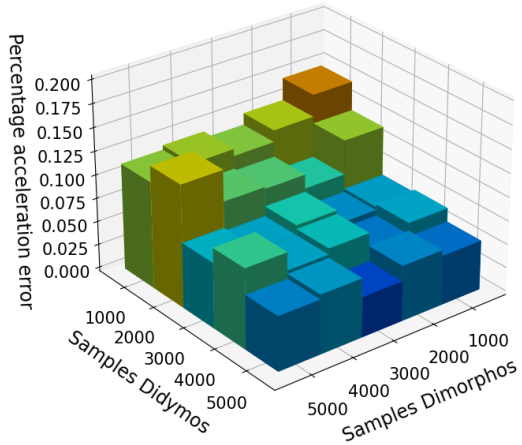


Figure 4.31: Model trained with the estimated mass model in proximity of Dimorphos

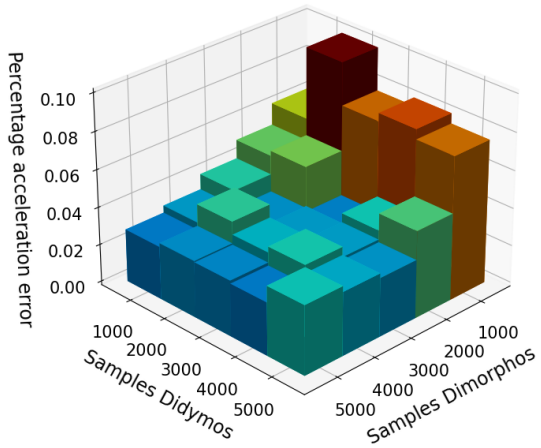
As we can see, the performance of the model gets better and similar results with respect with the model trained with the correct mass are obtained in the domain of the dataset. However, once outside the domain studied, the model in this case performs worse and it tends to the approximated mass model. This is due to the error on the estimation on the mass. If the mass model would be approximated better, the error would decrease outside of the training data domain.

4.2.3. Model performance with different training set sizes

In order to understand if the sampling of data in proximity of Didymos influences negatively with the performances of the model in proximity of Dimorphos and vice-versa (this could be due to the fact that the same field point is used to train both networks with the same loss function), the number of data used for the training set in proximity of both asteroids is varied as we can see in Figure 4.32. The same network (with same weight and biases) is trained using NN_A and varying only the number of data in the training set. In particular the number of samples in proximity of Didymos and Dimorphos is changed. The validation and the test set is the same for all the different models trained. The test set has a domain between the surface of the asteroid and 5 radii. The percentage error represented in the figures is the mean percent error of all the test set.



(a) Error in proximity of Didymos



(b) Error in proximity of Dimorphos

Figure 4.32: Mean error of the model with the variation of the number of data in the training set

As we can see, only when a low number of samples are considered in proximity of an asteroid the mapping in proximity of the same asteroid gets worse. It seems that sampling in proximity of an asteroid does not degrade the performance in the vicinity of the other one. In both cases, it seems that when considering 3000 or more samples the model starts to converge in both regions. A total sampling time of 50 hours would be sufficient to

compute all the accelerations required for the training (the sampling for each measurement is assumed of 30 seconds).

4.2.4. Sampling time domain

In this simulation, the same model is trained using NN_A using different time domain for the training and validation set (between 0 and 1, 100 and 1000 hours). This is done in order to understand if the time domain of the training data influences on the performance of the model. The test set has a domain between 1000 to 3000 hours from the initial time instant. By changing the sampling time domain, the performance of the network do not change much. This is because the model is built in such a way that all the measurements could be theoretically taken in the same time instance. In fact, the acceleration of the asteroids is modelled individually and, given the same position of the field point in the body frame, the acceleration would be the same independently on the position and on the orientation of the asteroids. This can be seen in the figures below:

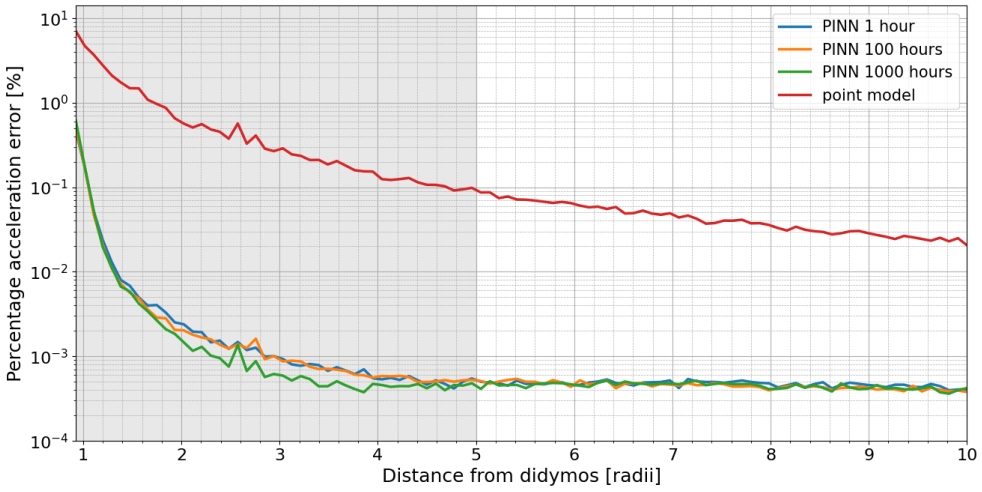


Figure 4.33: Error in proximity of Didymos with different time domains

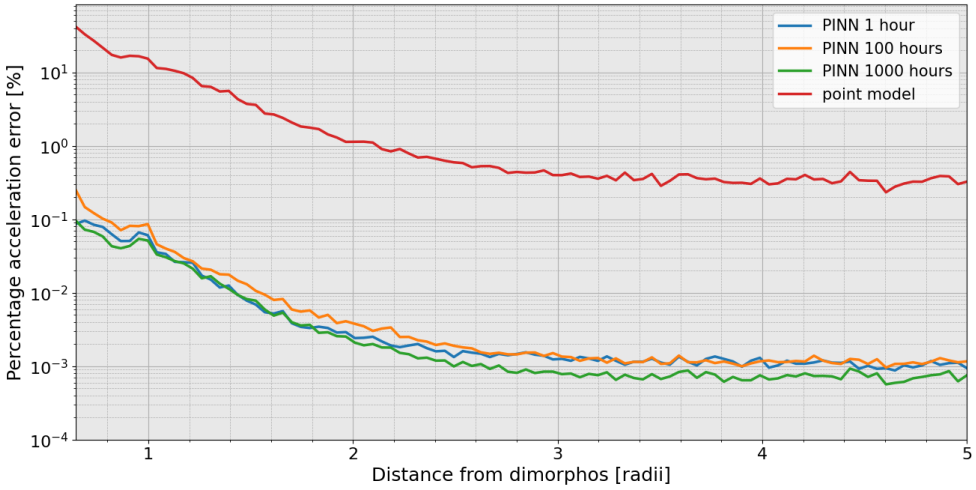


Figure 4.34: Error in proximity of Dimorphos with different time domains

For the same reason as before, by changing the eccentricity of the orbit (it could be that the current eccentricity of the orbit is between 0.03 and 0.07 [8, 9]), the results do not change. For this reason, by perturbing the orbits and the orientation of the asteroids, the performance of the networks should not change much.

4.2.5. No proximity data

It could be that the measurements near the surface of an asteroid are not permitted. To simulate this case, a domain from 2 to 5 radii of distance from both asteroids is considered for the training and validation set. This case is compared with the model trained with the radii domain described in Section 3.3.2. As we can see in Figure 4.35 and 4.36, only the performance of the model in proximity of the surface seems to be affected. Measurements in proximity of the surface would then be advised in order to map better the total acceleration in that region. In the region with a corresponding domain for both cases the performance seems pretty similar. In particular, the PINN trained with a training data domain between 2 and 5 radii performs slightly better at increasing distances when the same number of data and epochs are considered. This could be due to the fact that this PINN does not need to model data in proximity of the surface and will concentrate on mapping only field points far away from the surface. The blue area in the figures indicate the region where both training data domain are considered. In grey is indicated the region where only the training data generated in the whole domain are used.

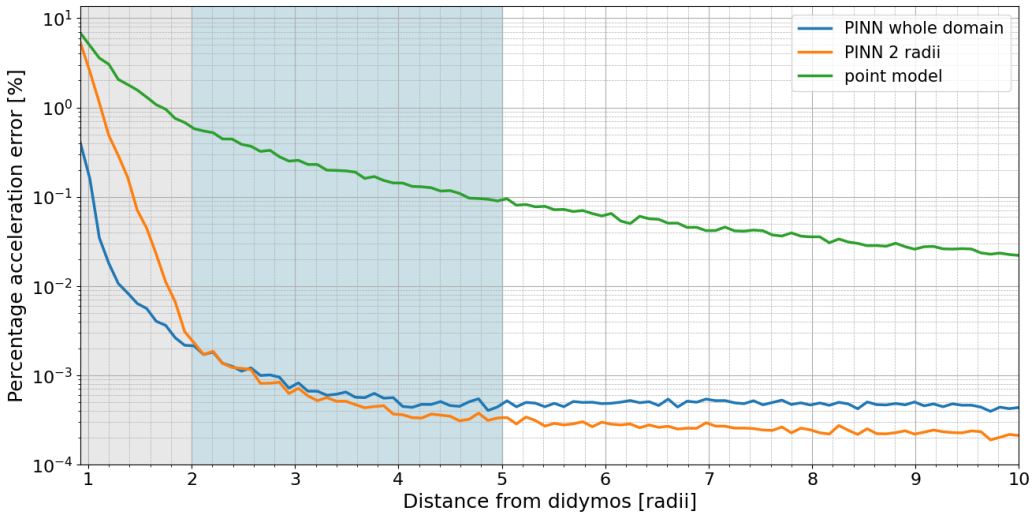


Figure 4.35: Model trained with domain starting from 2 radii in proximity of Didymos

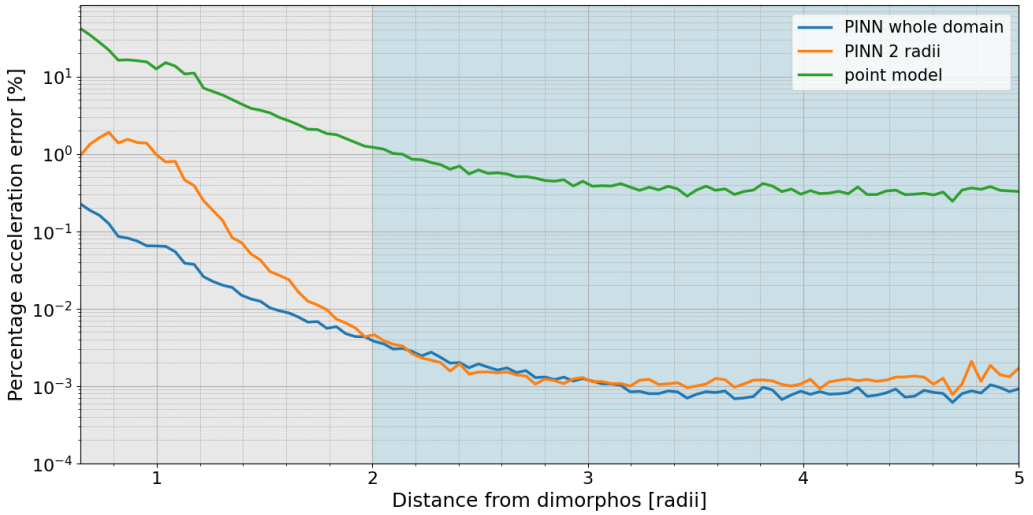
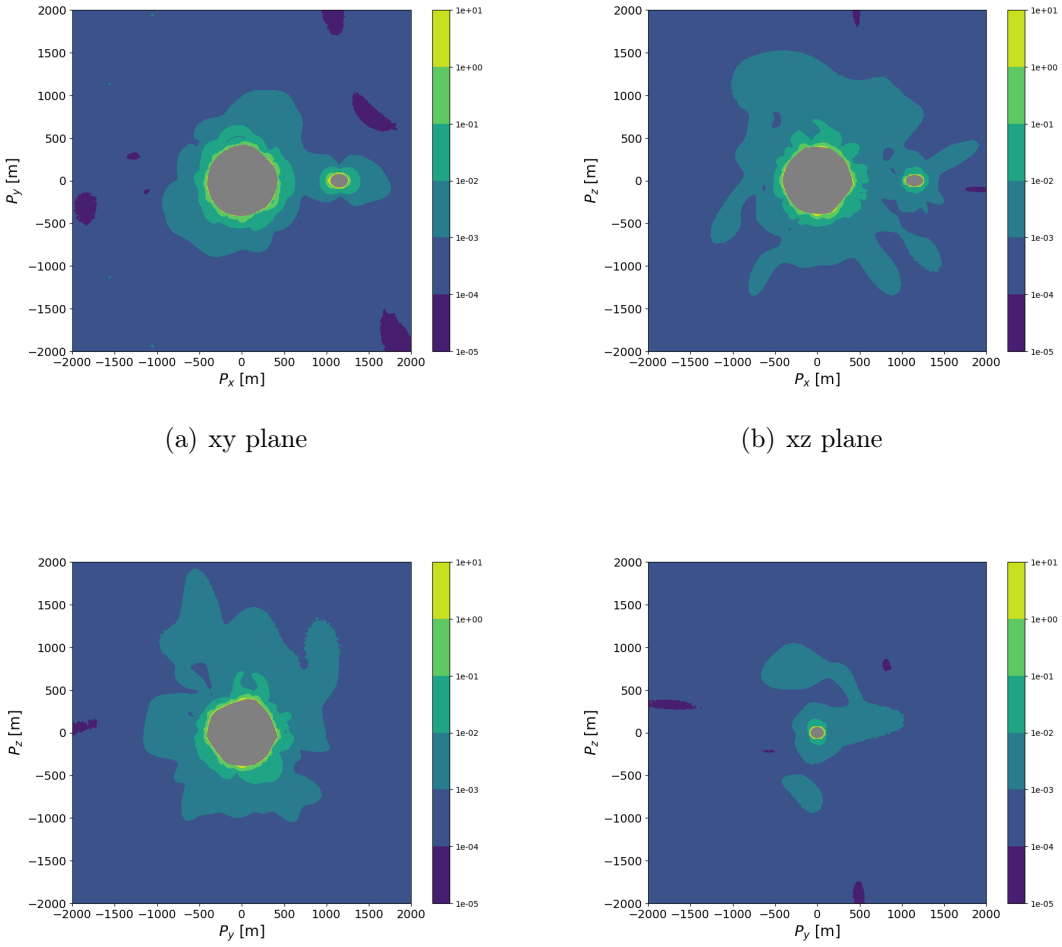


Figure 4.36: Model trained with domain starting from 2 radii in proximity of Dimorphos

4.2.6. Networks mapping in singular field points

In order to check the performance of the PINNs in case they are trained from real acceleration measurements, the total percentage acceleration error can be studied in each field point individually. In this way, it can be investigated if any region is poorly mapped by the PINNs. In Figure 4.37 the total percentage acceleration error is reported for each field point. The reference frame considered is the inertial one. The graphs represent the

error at the initial time instant for sake of clarity. Similar results are obtained at different time instants. As we can see, the errors are in line with what we have seen so far and no particular region seems poorly mapped. The two asteroids are represented in gray. The loss function used for the training was NN_A.



(a) xy plane

(b) xz plane

(c) yz plane centered in Didymos

(d) yz plane centered in Dimorphos

Figure 4.37: Total percentage acceleration error for each field point at the initial time instant

4.2.7. Error on the measurements

In order to understand better the possible performances of the model in real life applications, some errors are added to the measurements. A root mean square error (RMSE) in the range of $[10^{-6}, 10^{-8}]$ is assumed. The error is added to all the total acceleration measurements of the training and of the validation set. The error is not added to the

test set in order to verify the performances of the model. The error used is in line with errors of some gravimeters used in space applications [4]. As we can see in Figure 4.38 and 4.39, the model is still capable of mapping the gravity field even in the presence of errors although with worse performances as the error gets bigger. All the models in the figure where trained with NN_A.

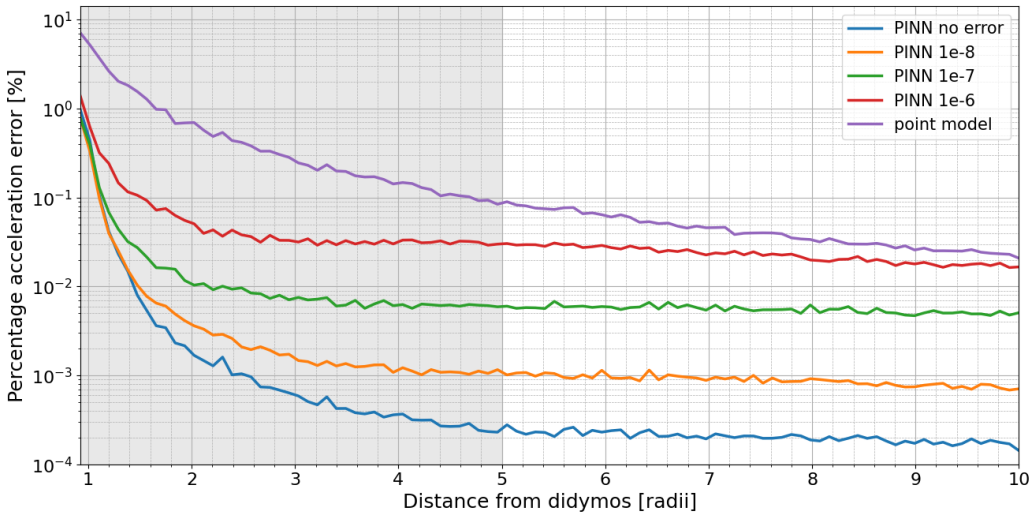


Figure 4.38: Error comparison in proximity of Didymos

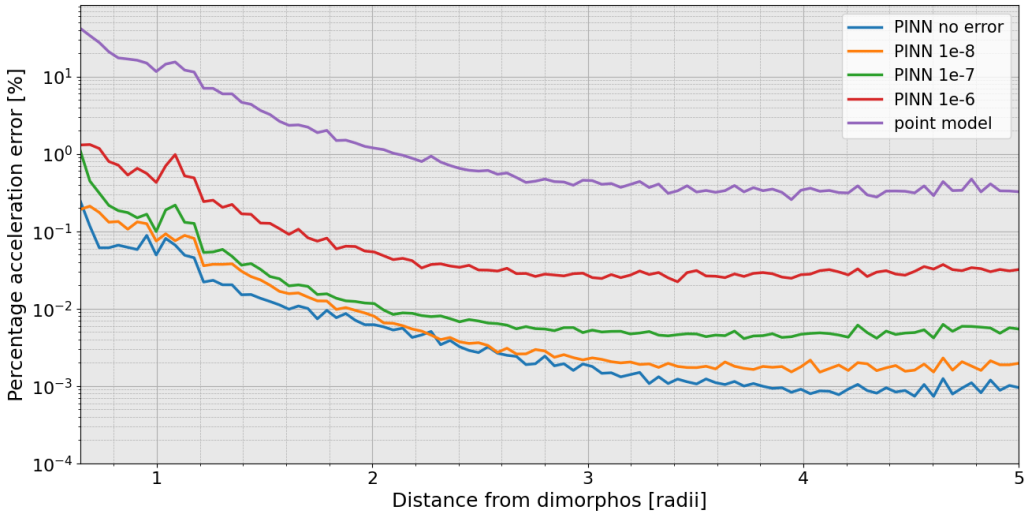


Figure 4.39: Error comparison in proximity of Dimorphos

As the error in the measurements is increased, the performance of NN_custom gets better

with respect to NN_A when trained with the same data. With an error of 10^{-8}, the performance of the model are practically the same. However, as we can see in Figure 4.40 and 4.41, in case the error considered is 10^{-6}, NN_custom performs visibly better resulting in the best model.

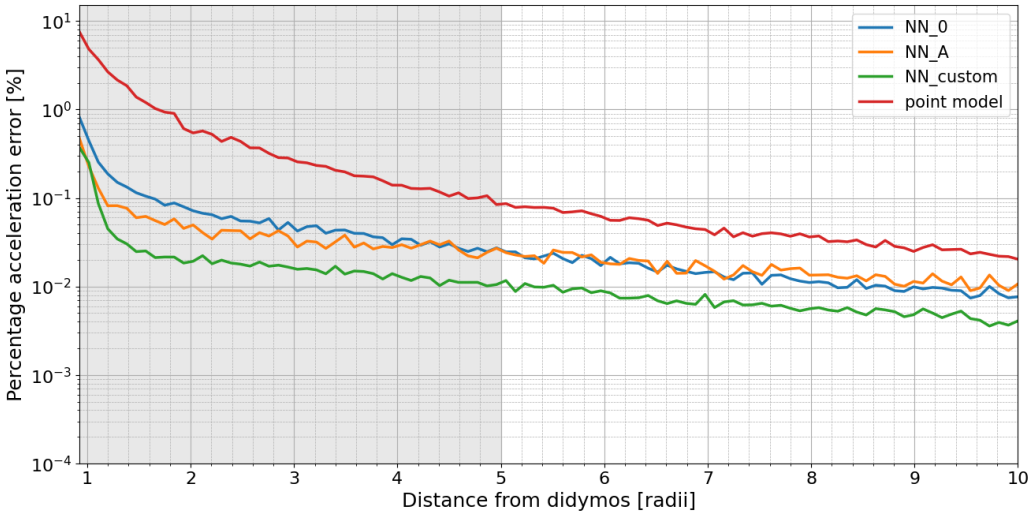


Figure 4.40: Performance with 10^{-6} RMSE in proximity of Didymos

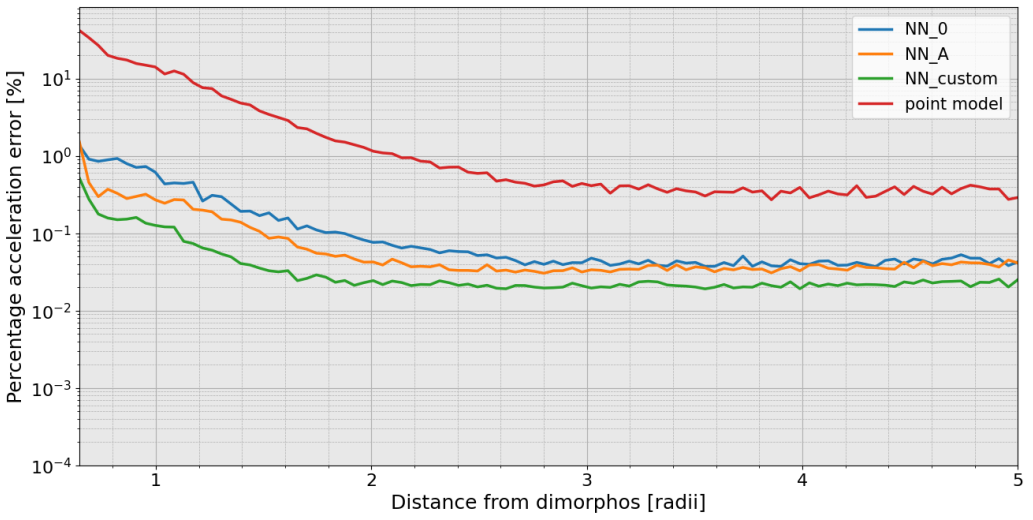


Figure 4.41: Performance with 10^{-6} RMSE in proximity of Dimorphos

In case an error of 10^{-5} or higher is considered, the model starts to diverge and it performs worse with respect to the point model as we get farther from the surface. This is due

to an high percentage error that is even over 100% on measurements far away from the body.

5 | Conclusions and future developments

In this work, a novel approach has been adopted to model the gravity field of a binary asteroid system through the usage of PINN. The PINN combines traditional deep learning techniques with physics properties to ensure that the solution learned obeys those laws. In particular, this work demonstrates that the PINN is able to produce high-accuracy models for the gravity field of the Didymos(65803) binary system.

In this work the following sub-questions were investigated:

1.1 *How accurately does the physics-informed neural network represent the gravity field compared to state of the art approaches? And what is the gain in terms of computational time of PINN with respect to them?*

The PINN model can reach performances close to the best known gravity field models of both asteroids (polyhedral for Didymos and ellipsoid model for Dimorphos) while increasing the computational speed in the case of Didymos. In the case of Dimorphos, the computational speed is close to the reference model used, making its usage for modeling the gravity field redundant.
Compared to spherical and mascon models, PINN performs better, especially in proximity of the surface. However, spherical models, when considering low order spherical harmonics coefficients, compute much faster with respect to the PINN model. In order to map the whole gravity field, a combination of PINN and spherical harmonics could be implemented. PINN will be used to map the acceleration in proximity of the surface while spherical harmonics will be operated when considering field points far away from the asteroids. In this way the overall computational speed is increased while maintaining the same precision.

1.2 *What is the best physic-informed architecture and methodology?*

Modeling the gravity field of the binary system using two different networks, where each network models the potential of a single asteroid and their contribution is summed up

together, results in performances much greater with respect to the case where the gravity field is directly modeled using a singular network. Better performances are obtained when the point model is already prescribed to the PINN. In case the PINNs are trained from real acceleration measurement, an assessment on the mass of the binary system shall be made before the training. PINNs performs slightly better with respect to ANN. The PINN should be trained imposing the potential (if known) and acceleration error equal to zero with the loss function in order to achieve the best performances. Imposing the laplacian and the curl error equal to zero increase the training time and the memory needed and shall be avoided unless errors on the measurements are present.

1.3 *How many acceleration data are required to estimate the acceleration of the binary system using physics-informed neural network without any information about the potential? How does the error vary as function of the size of the training dataset? How sensitive it is to noise in the training data?*

A fairly low number of data (around 3000 data taken in proximity of Didymos and 3000 data in proximity of Dimorphos) generated from simulated total acceleration measurements are sufficient to map the gravity field of the binary system with the usage of PINNs. The PINNs could then be used to model the gravity field in-situ from real acceleration measurements. Increasing the number of data in proximity of the asteroids above this number do not seem to give significant improvements on the networks.
When considering errors in the measurements, the PINNs are still able to map the gravity field fairly well. In this case, it is recommended to impose the laplacian of the potential and the curl of the acceleration equal to zero with the loss function while using an adapting loss weight. This will decrease the error while mapping the gravity field. However, this will increase the training time and the memory needed for the training.

Future developments of this work are linked to the approximations made. Investigation on the method performances in case the reference models have an heterogeneous density distribution shall be made.
The motion of the asteroids can be modeled considering the excited spin state of Dimorphos due to DART impact as discussed in [9] to see if it affects in any way the performances of the network when trained from total acceleration measurements.
A spacecraft trajectory can be simulated around the binary system in order to see if acceleration measurements taken during the simulated trajectory can still map the whole gravity field of the binary system. In this way, the samples will not be randomly distributed around the asteroid, but only confined in certain regions of space.

Bibliography

[1] A. G. Baydin, B. A. Pearlmutter, A. A. Radul, and J. M. Siskind. Automatic differentiation in machine learning: a survey. *Journal of Machine Learning Research*, 18 (153):1–43, 2018.

[2] R. Bischof and M. Kraus. Multi-objective loss balancing for physics-informed deep learning. 10 2021.

[3] A. Capannolo, F. Ferrari, and M. Lavagna. Families of bounded orbits near binary asteroid 65803 didymos. *Journal of Guidance, Control, and Dynamics*, 42:1–10, 2018.

[4] D. Carroll, K.and Faber. Tidal acceleration gravity gradiometry for measuring asteroid gravity field from orbit. *69th International Astronautical Congress*, (1), 2018.

[5] T. G. G. Chanut, S. Aljbaae, and V. Carruba. Mascon gravitation model using a shaped polyhedral source. *Monthly Notices of the Royal Astronomical Society*, 450 (4):3742–3749, 2015.

[6] P. Ditmar, J. Kusche, and R. Klees. Computation of spherical harmonic coefficients from gravity gradiometry data to be acquired by the goce satellite: regularization issues. *Journal of Geodesy*, 77:465–477, 2003.

[7] M. W. Duncan. *The Theory of the Potential*. McGraw-Hill, 1930.

[8] A. J. M. et al. The perturbed full two-body problem: Application to post-dart didymos. *The planetary science journal*, 4(141), 2023.

[9] H. F. A. et al. Dynamical evolution of the didymos-dimorphos binary asteroid as rubble piles following the dart impact. *The planetary science journal*, 3(7), 2022.

[10] H. F. L. et al. Lucy mission to the trojan asteroids: Science goals. *The planetary science journal*, 2(171), 2022.

[11] P. M. et al. The esa hera mission: Detailed characterization of the dart impact outcome and of the binary asteroid (65803) didymos. *The planetary science journal*, 3(7), 2022.

[12] R. F. et al. Modeling irregular small bodies gravity field via extreme learning machines and bayesian optimization. *Advances in Space Research*, 67(1), 2021.

[13] E. Fantino and S. Casotto. Methods of harmonic synthesis for global geopotential models and their first-, second- and third-order gradients. *Journal of Geodesy*, 83: 595–619, 2009.

[14] F. Ferrari and M. Lavagna. Dynamical model of binary asteroid systems through patched three-body problems. *Celest Mech Dyn Astr*, 125:413–433, 2016.

[15] F. Ferrari, V. Franzese, M. Pugliatti, C. Giordano, and F. Topputo. Trajectory options for hera's milani cubesat around (65803) didymos. *J Astronaut Sci*, 68: 973–994, 2021.

[16] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016.

[17] B. Hofmann-Wellenhof and H. Moritz. *Physical Geodesy*. Springer Vienna, 2005.

[18] M. Kanamaru, S. Sasaki, and M. Wieczorek. Density distribution of asteroid 25143 itokawa based on smooth terrain shape. *Planetary and Space Science*, 174:32–42, 2019.

[19] W. M. Kaula. *Theory of Satellite Geodesy: Applications of Satellites to Geodesy*. Courier Corporation, 2013.

[20] J. B. Lundberg and B. E. Schutz. Recursion formulas of legendre functions for use with nonsingular geopotential models. *Journal of Guidance Control Dynamics*, 11: 31–38, 1988.

[21] J. L. Margot, M. C. Nolan, L, A. M. Benner, S. J. Ostro, R. F. Jurgens, J. D. Giorgini, M. A. Slade, and D. B. Campbell. Binary asteroids in the near-earth object population. *Science*, 296(5572):1445–48, 2002.

[22] J. Martin and H. Schaub. Physics-informed neural networks for gravity field modeling of the earth and moon. *Celest Mech Dyn Astr*, 134(13), 2022.

[23] J. Martin and H. Schaub. Physics-informed neural networks for gravity field modeling of small bodies. *Celest Mech Dyn Astron*, 134(46), 2022.

[24] J. Martin and H. Schaub. Physics-informed neural network gravity model revisited, model generation iii. *AAS/AIAA Spaceflight Mechanics Meeting*, Austin, TX, 2023.

[25] S. Naidu, L. Benner, M. Brozovic, M. Nolan, S. Ostro, J. Margot, J. Giorgini, T. Hirabayashi, D. Scheeres, P. Pravec, P. Scheirich, C. Magri, and J. Jao. Radar ob-

servations and a physical model of binary near-earth asteroid 65803 didymos, target of the dart mission. *Icarus*, 348:113777, 2020.

[26] J. Pearl and D. Hitt. Comparing the computational efficiency of polyhedral and mascon gravity models. *AAS/AIAA Space Flight Mechanics Meeting*, 2017.

[27] J. Pearl and D. M. Hitt. Mascon distribution techniques for asteroids and comets. *Celest Mech Dyn Astron*, 134(58), 2022.

[28] M. Raissi, P. Perdikaris, and G. E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019.

[29] A. Rathinam and A. G. Dempster. Octree-based mascon model for small body gravity fields. *Journal of Guidance, Control and Dynamics*, 42(11), 2019.

[30] S. Silvestrini and M. Lavagna. Deep learning and artificial neural networks for spacecraft dynamics, navigation and control. *Drones*, 6(270), 2022.

[31] Y. Takahashi. *Gravity Field Characterization around Small Bodies*. PhD thesis, University of Colorado at Boulder, 2013.

[32] Y. Takahashi and D. Scheeres. Small body surface gravity fields via spherical harmonic expansions. *Celest Mech Dyn Astr*, 119:169–206, 2014.

[33] Y. Takahashi, D. Scheeres, and R. Werner. Surface gravity fields for asteroids and comets. *Journal of Guidance, Control, and Dynamics*, 36:362–374, 2013.

[34] S. Wang, Y. Teng, and P. Perdikaris. Understanding and mitigating gradient flow pathologies in physics-informed neural networks. *SIAM Journal on Scientific Computing*, 43(5):A3055–A3081, 2021.

[35] R. Werner and D. Scheeres. Exterior gravitation of a polyhedron derived and compared with harmonic and mascon gravitation representations of asteroid 4769 castalia. *Celest Mech Dyn Astr*, 65:313–344, 1996.

[36] R. A. Werner. Spherical harmonic coefficients for the potential of a constant-density polyhedron. *Computers Geosciences*, 23(10):1071–1077, 1997.

A | Appendix

In this appendix is briefly discussed the case where a new neural network system is trained. Instead of modeling two different PINNs in parallel, a single PINN was trained with the total potential of the field point as a single output. The inputs are: the positions in the inertial reference frame of the two asteroids, the position of the field point in the inertial reference frame and the angles to rotate from the body reference frame of the asteroids to the inertial reference frame. The acceleration would then be the gradient of the potential with respect to the position of the field point. The loss function would be similar to Equation 3.12 where, instead of the acceleration of the single asteroid, the total acceleration is considered. This method was studied in order to understand if it would be more beneficial to map the potential of the single asteroids with two different networks and then summing together their contribution or it would be better to map directly the contribution from both bodies with a single network. The model is trained with the same parameters used in Section 4.2. The point mass model is added also in this case to the total contribution of the Network.

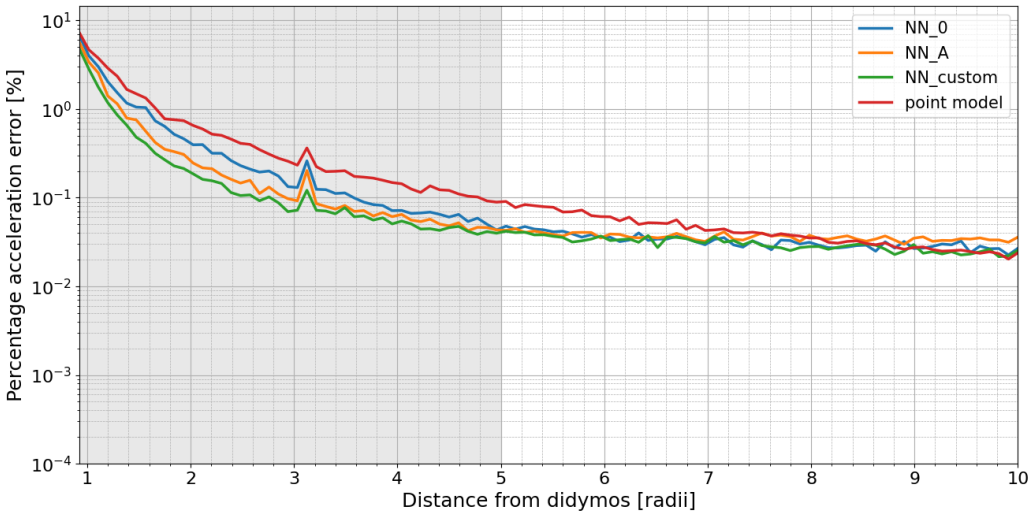


Figure A.1: Model tested in proximity of Didymos

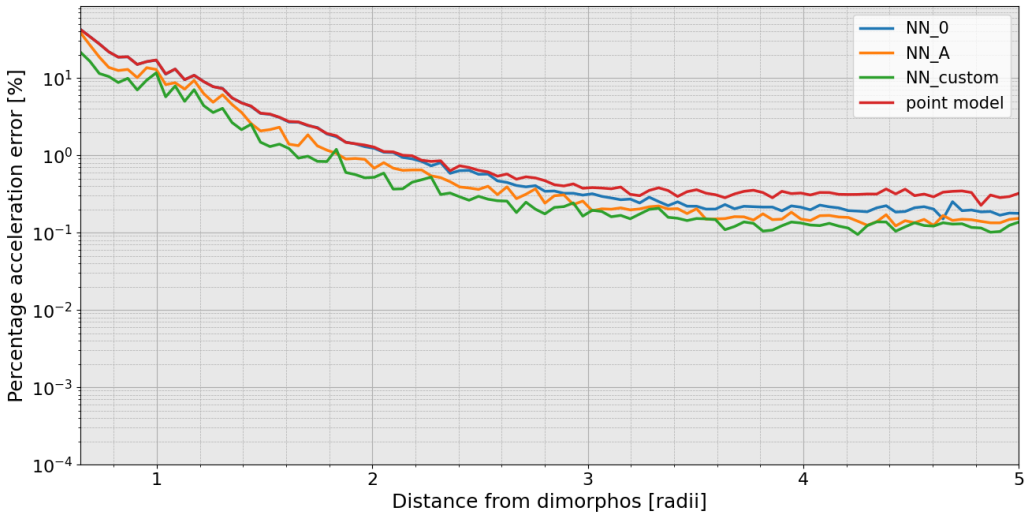
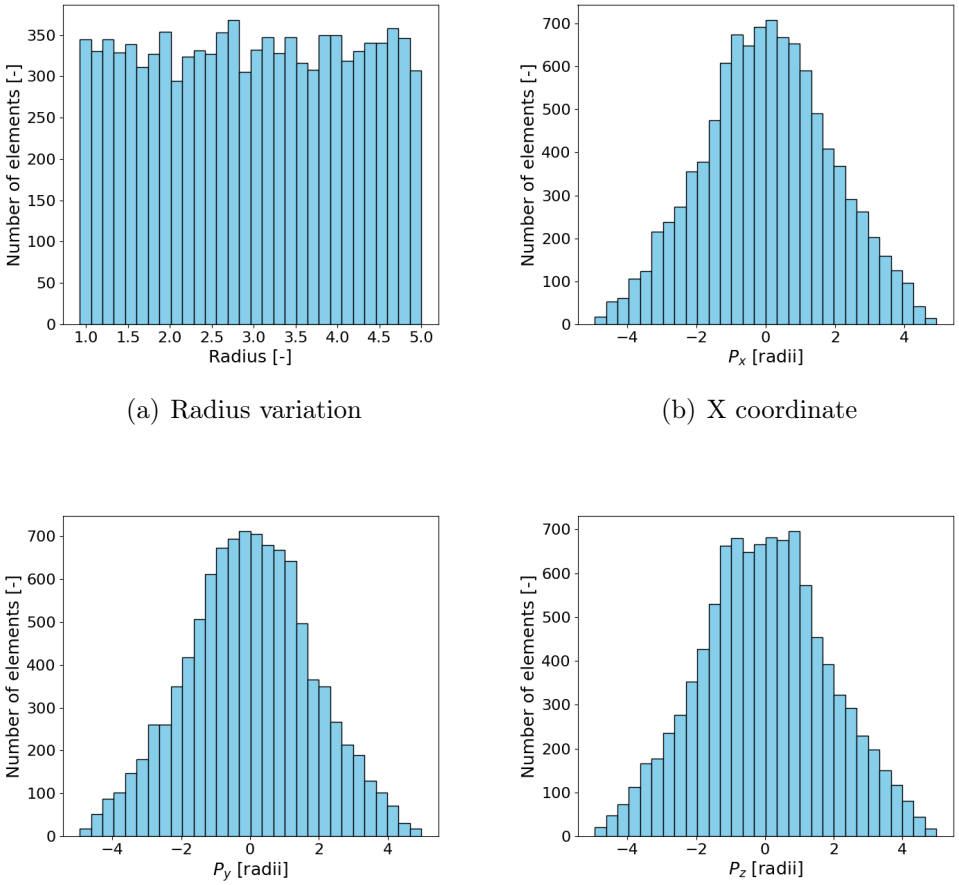


Figure A.2: Model tested in proximity of Dimorphos

The peak in error near 3 radii from Didymos is due to Dimorphos. In Figure A.2, this region is analyzed better. The grey area indicates the region where the training data are considered. As it can be seen in the figures above, the performance of the network do not come even close to the ones discussed in this work. This is probably due to the fact that some inputs in this case are time dependant as they will vary only as a function of time and not on the position of the measurements made. The position of the asteroids and their angles of rotation will in fact vary only based on the time at which the measurement was made. Because the model is tested on a new set of data that are not part of the time domain studied, the network needs to approximate the acceleration with inputs that are in a time domain never seen before and thus resulting in worse performances.

B | Appendix

In Figure B.1 is represented the field points distribution when they are generated as described in Section 3.3.1. In particular, they are represented as a function of the radial distance from the asteroid and as a distance from the asteroid in Cartesian coordinates in the body frame. When varying the number of the samples, for example from 10000 to 5000, the distribution will be similar.



(a) Radius variation

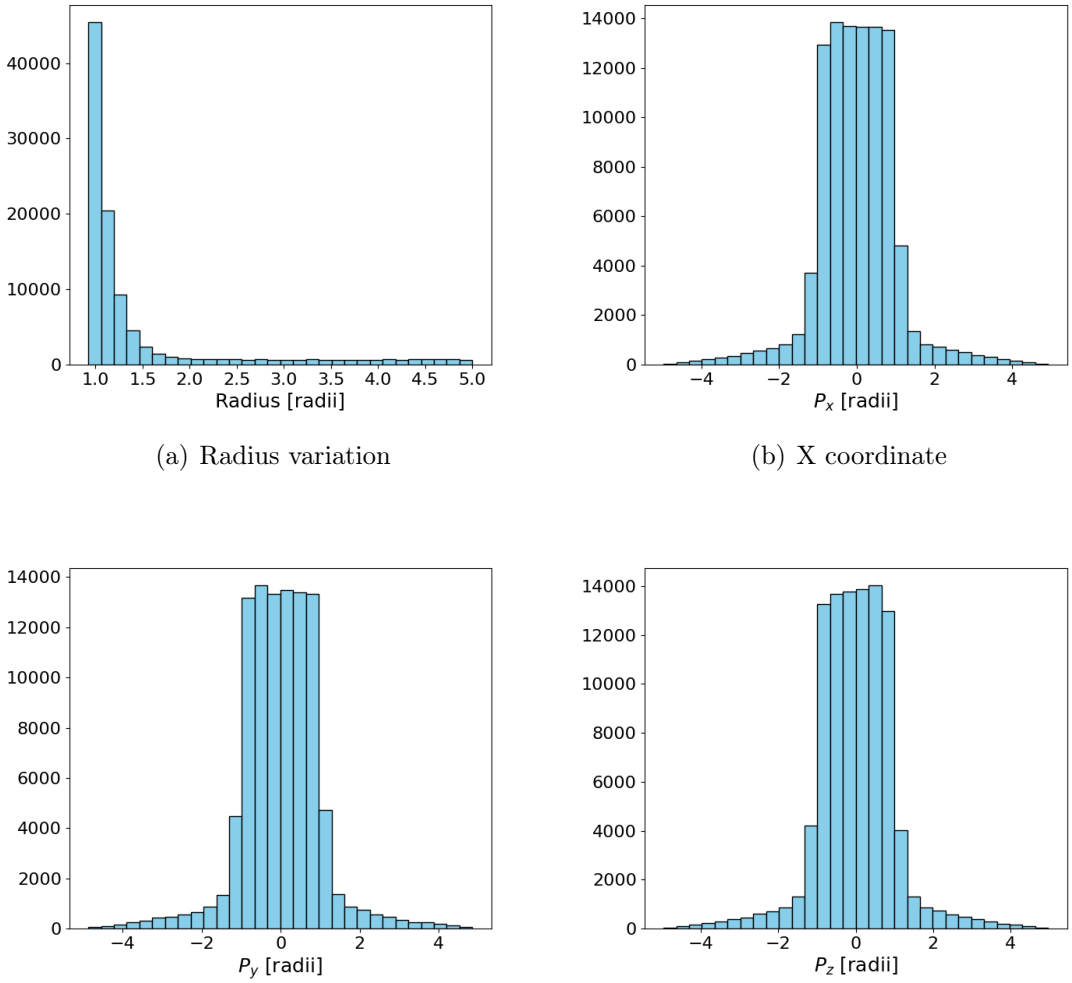
(b) X coordinate

(c) Y coordinate

(d) Z coordinate

Figure B.1: Number of field points considered as a function of the distance from Didymos for 10000 different field points

In Figure B.2 is represented the field points distribution when they are generated as described in Section 4.1.6.



(a) Radius variation

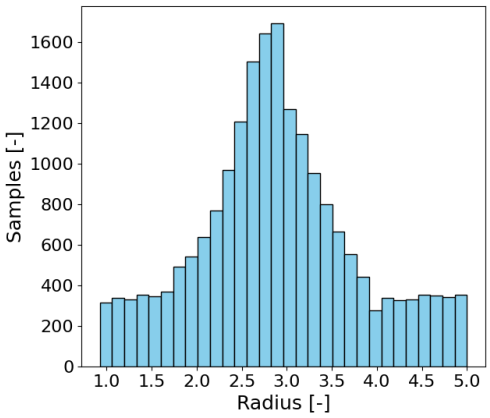
(b) X coordinate

(c) Y coordinate

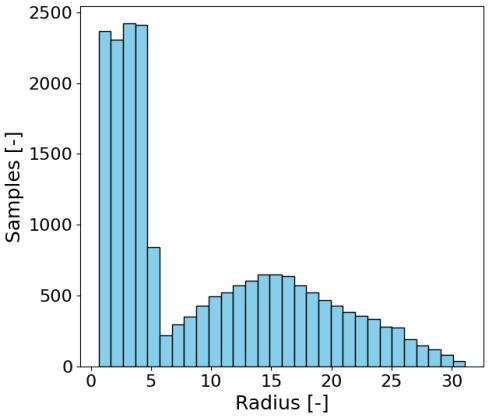
(d) Z coordinate

Figure B.2: Number of field points considered as a function of the distance from Didymos for 20000 field points generated uniformly plus 80000 field points generated in proximity of Didymos

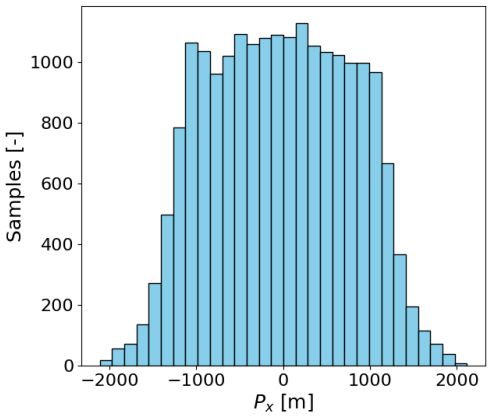
In Figure B.3 is represented the field points distribution when they are generated as described in Section 3.3.2. In particular, they are represented as a function of the radial distance from the asteroids and as a position in the inertial reference frame with Cartesian coordinates.



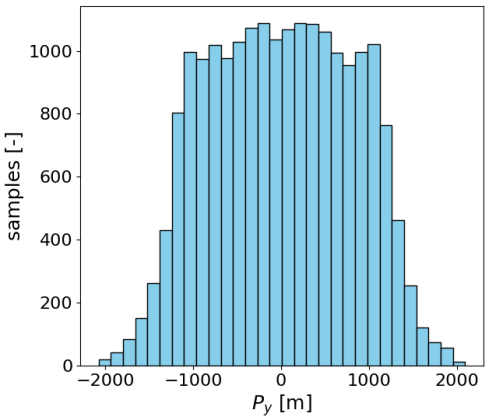
(a) Radius variation from Didymos



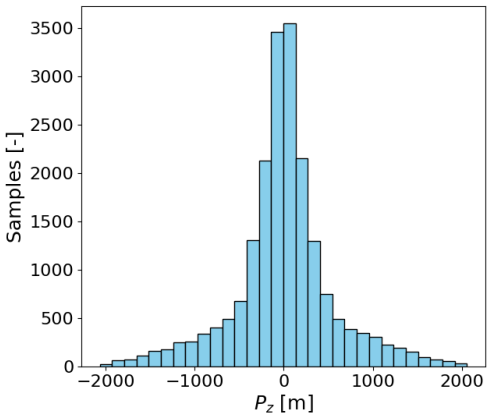
(b) Radius variation from Dimorphos



(c) X coordinate



(d) Y coordinate



(e) Z coordinate

Figure B.3: Number of field points considered as a function of the distance of the asteroids and of the position in the inertial reference frame for 10000 different field points in proximity of Didymos and 10000 field points in proximity of Dimorphos

List of Figures

List of Tables

List of acronyms

ANN Artificial Neural Network

PINN Physics-Informed Neural Network

MSE Mean Squared Error

RMSE Root Mean Squared Error

NEA Near Earth Asteroids

AIDA Asteroid Impact and Deflection Assessment

DART Double Asteroid Redirection Test

TAG Touch And Go maneuver

SRP Solar Radiation Pressure