DOI: 10.13009/EUCASS2022-4761

Assessment of Machine Learning Techniques for Time-Optimal Landing Trajectories

Pablo Solano-López, Adrián Barea Vilar*, Roger Gutiérrez-Ramon** and Hodei Urrutxua Cereijo**

**Aerospace Systems and Transport Research Group (GISAT-ASTRG) | Universidad Rey Juan Carlos, Camino del Molino 5, 28943, Fuenlabrada, Madrid, Spain.*

*** The Graduate University for Advanced Studies, SOKENDAI Tsuda Lab, ISAS/JAXA | 3-1-1 Yoshinodai, Chuo-ku, Sagamihara City, Kanagawa Prefecture, 252-5210, JAPAN.*

Abstract

In this contribution, a Machine Learning methodology is presented for predicting unknown properties of time-optimal landing problems, defined in the indirect formulation. The chosen problem to validate this methodology is the landing of a satellite on an ellipsoid-like asteroid within the two-body problem Keplerian framework. The mathematical formulation of the problem, together with the dataset generation procedure is presented to the show the learning process and success criteria. Finally, several NN schemes are validated with part of the generated dataset and benchmarked with test cases different from it to check the boundaries of their application.

1. Introduction

The advent of Machine Learning (ML) and its applications at engineering in general and to space engineering in particular, has allowed to rethink complex problems associated with mission planning and the Attitude Determination and Control System (ADCS) of the spacecraft in study. It is more and more common to find such techniques as helping tools for the guidance, navigation, and control of aerospace vehicles **[1]** as well as algorithms and tools to deal with unknown physics usual problems, such as: orbit or trajectory determination, noise and error filtering, perturbation analysis and optimization problems **[2]**.

As an interesting example, it has been proven that, using ML, orbital trajectories can be learnt from existent or artificial (modelled) data, making possible to predict the future states of a Spacecraft without the need of a mathematical approximated model (*i.e.*, data driven prediction). Also, another application area for these mathematical tools is the treatment of optimal control problems for rendezvous between orbital objects **[2]** or landing in partially known environments, as it can be an asteroid or a planetary surface **[3, 5]**. Neural Networks (NN) models can be constructed in such a way they learn these optimal solutions to then perform parametrical continuation or they provide unknown initial/final states of the problem. Additionally, thanks to the exponential grow on the development of new families of NN and related tools, together with mathematical techniques for Deep Learning (DL) common problems such as overfitting or biasing are being tackled successfully. Finally, another point of interest for ML in space engineering is the parametrical identification for uncomplete models, *e.g.*, asteroid shape determination, perturbation of unknown orbits, etc. Where NN can be used to predict the ODEs or mathematical laws of the problem with a certain level of accuracy **[1, 4-6]**.

In this contribution, we focus on the applications of ML techniques for optimal control problems, aiming to present a general working methodology that is applied to a particular scenario as a demonstrator. Specifically, we study look at the determination of a time-optimal landing trajectory in a particular environment: A Satellite approaching a gravitational object aiming to land at its surface, *i.e.*, to reach a prescribed point of a surface with null touchdown velocity. This problem can be modelled as an optimal control problem with the thrust orientation as the control variable aiming to perform the manoeuvre at minimal cost (or, optimal time), as it will be presented in the following sections.

As the aim of this work is to show the methodology and the possibilities ML offers to optimal control problems, we select a simplified demonstrator for this contribution. Considering the growing interest on flybys and touchdowns on asteroids and meteoroids **[7, 8]**, we will study the landing over a simplified asteroid, an elliptic, non-rotating one, leaving the extension of this application for future contributions.

DOI: 10.13009/EUCASS2022-4761

The resolution of such an optimal problem, following what is usually called the indirect approach, involves the determination of the so-called co-state variables **[9]**, whose rates of change are defined by the desired optimality conditions. In turn, the initial values of the co-state variables must be tuned to comply with the desired boundary conditions. This can prove to be a difficult task, since these co-states are mathematical entities without a clear physical transcription and the behaviour of the trajectory can be especially sensitive to small changes of the initial values of the co-states. Moreover, the range of values these co-states might take is highly variable, oscillating in various orders of magnitude. Consequently, the finding of proper initial co-states with enough precision calls for a mathematical problem itself, requiring an iterative process or similar that tends to fail without an adequate initial guess **[9]**.

This is the standard problematic under indirect optimal control problems, in contraposition with the direct ones, where the trajectory or other element of the cost function is mathematically treated (often, discretized in a finite number of nodes). Although easier to approach numerically, the direct approach is biased as it requires this discretization, an imposed final time-of-flight value, or similar constraints and often, it might hide the actual optimal result.

This work proposes a NN-based approach to obtain an initial guess of said initial values of the costate variables in the Hamiltonian or indirect formulation of the problem by means of a general ML methodology to solve indirect optimal control problems and then applying it to the simplified asteroid landing benchmark to show some of its peculiarities.

The paper is organized as it follows: first a mathematical description of the indirect optimization problem is presented in Section 2, with a dedicated subsection where the formulation of the asteroid landing is described together with its working hypothesis and simplifications, then Section 3 introduces the mathematical methodology for the ML initial guess calculation, detailing the test matrix and the details involved, Section 4 presents the numerical results of this approach together with some additional remarks and finally, Section 5 collects the conclusion of this work.

2. Mathematical description of the problem

This section presents the mathematical formulation of an indirect optimal problem with boundary conditions together with the selected benchmark to test the ML methodology.

A general optimization Bolza Problem can be casted in the form

$$\begin{aligned} \min\ & J = \mathcal{M}\left[\boldsymbol{s}(0), \boldsymbol{s}(t_f), t_0, t_f\right] + \int_{t_0}^{t_f} \mathcal{L}\left[\boldsymbol{s}(t), \boldsymbol{u}(t_f), t\right] dt \\ s.\ t \quad & \dot{\boldsymbol{s}} = \boldsymbol{f}\left[\boldsymbol{s}(t), \boldsymbol{u}(t), t, \boldsymbol{\beta}\right], \qquad \forall t \in \left[t_0, t_f\right] \\ & \boldsymbol{h}\left[\boldsymbol{s}(t), \boldsymbol{u}(t_f), t\right] = 0 \\ & \boldsymbol{g}\left[\boldsymbol{s}(t), \boldsymbol{u}(t_f), t\right] \leq 0, \qquad \forall t \in \left[t_0, t_f\right] \end{aligned} \tag{1}$$

where the state of the dynamical system is described by the vector $\boldsymbol{s}(t)$ and whose first-order evolution with respect to the independent variable t is governed by the vector field $\boldsymbol{f}\left[\boldsymbol{s}(t), \boldsymbol{u}(t), t, \boldsymbol{\beta}\right]$ with $\boldsymbol{\beta}$ a set of parameters and $\boldsymbol{u}(t)$ is the control vector field.

The optimal solution is given by the determination of the phase space flow $\boldsymbol{s}^*(t)$ and control application $\boldsymbol{u}^*(t)$ minimizing the cost function J while satisfying the boundary conditions equalities $\boldsymbol{h}\left[\boldsymbol{s}(t), \boldsymbol{u}(t_f), t\right]$ and path constraints $\boldsymbol{g}\left[\boldsymbol{s}(t), \boldsymbol{u}(t_f), t\right]$.

DOI: 10.13009/EUCASS2022-4761

2.1 Lander Dynamics

The proposed problem involves the dynamics of a lander subjected to the gravitational attraction of a uniform ellipsoidal body as well as the thrust generated by the lander itself. The reason under this choice, *i.e.*, a Keplerian 2-body motion is to allow to focus not only on asteroid landings but, as mentioned in the introduction, the landing in any kind of gravitational body, regardless of its mass. It is important to recall that although this is a test case for benchmarking the ML methodology, it is generalizable to landings in n-body problems or rendezvous manoeuvres **[3]**.

Thus, the differential equations that describe the motion of the lander are the following:

$$\begin{cases} \dfrac{d\boldsymbol{r}}{dt} = \boldsymbol{v} \\ \dfrac{d\boldsymbol{v}}{dt} = \boldsymbol{g} + f\boldsymbol{u} \end{cases} \tag{2}$$

Where $\boldsymbol{r}$ and $\boldsymbol{v}$ are the cartesian position and velocity of the lander, respectively, in a body-fixed reference frame, $\boldsymbol{g}$ is the gravitational acceleration experienced by the lander, $\boldsymbol{u}$ is a unitary vector that defines the thrust orientation and f is the magnitude (modulus) of the thrust acceleration.

This set of axes has been chosen such its origin is at the centre of mass of the body and its axes are aligned with the ellipsoid semiaxes, as it is shown in Figure 1. In a situation of a rotating asteroid, the choice of reference frame might vary, and so the equations defining the lander motion. The combination of position and velocity defines the state vector of the lander.

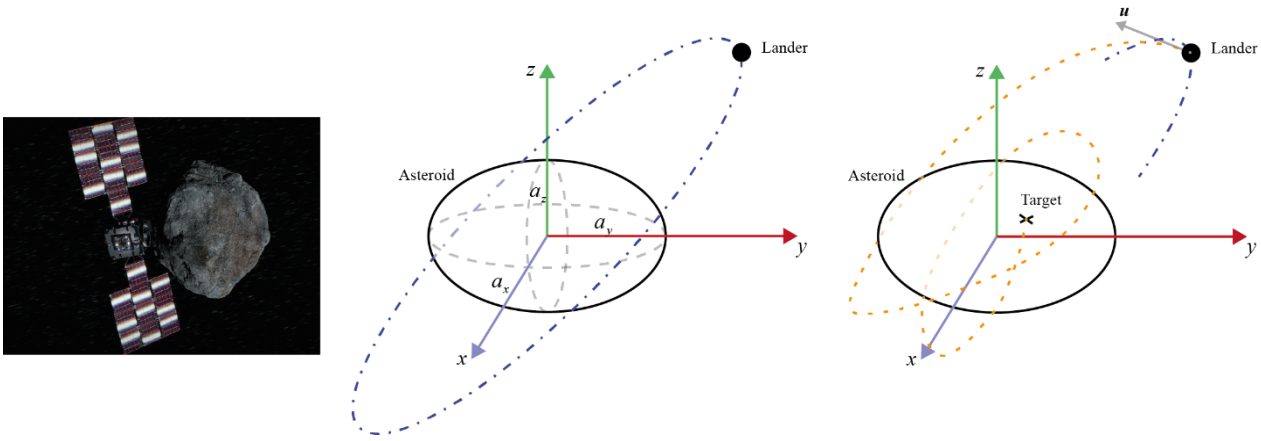

Figure 1: Schematics of the studied Problem, from left to right: Artist's impression of Phaeton, from Destiny+ mission [8], simplified Keplerian initial 2-body problem and example of a time-optimal landing trajectory

According to **[10]**, the gravitational attraction of the ellipsoid body can be computed as follows:

$$g_i = -2\pi G\rho a_x a_y a_z r_i \int_{\xi}^{\infty} \frac{d\alpha}{\left(a_i^2+\alpha\right)\phi} \qquad \forall i \in \{x,y,z\}$$
$$\phi = \sqrt{\left(a_x^2+\alpha\right)\left(a_y^2+\alpha\right)\left(a_z^2+\alpha\right)} \tag{3}$$

DOI: 10.13009/EUCASS2022-4761

Where the subscript i denotes the components of its associated vector with respect to the body-fixed reference frame. The variable α works as the integration variable, G is the universal gravitational constant, ρ the body density (considered uniform for this test case) and a_i stands for the magnitude of each of the body semiaxes.

The variable ϕ is introduced to simplify notation and works as a sort of geometrical measure of the asteroid for the integration. Regarding the limits of the integration, as indicated in **[10]**, ξ represents the positive confocal ellipsoidal coordinate associated to $\boldsymbol{r}$, defined by the single positive root of the flowing equation:

$$\frac{r_x^2}{a_x^2+\xi}+\frac{r_y^2}{a_y^2+\xi}+\frac{r_z^2}{a_z^2+\xi}=1 \tag{4}$$

Finally, the problem at hand requires the computation of the partial derivatives of $\boldsymbol{g}$ with respect to the components of $\boldsymbol{r}$, which can be obtained from the following expression:

$$\frac{\partial g_i}{\partial r_j}=\delta_{ij}\frac{g_i}{r_i} \quad \forall i\in\{x,y,z\},\forall j\in\{x,y,z\} \tag{5}$$

Where δ_{ij} is the Kronecker delta function

2.2 Optimality conditions

As already discussed, we are aiming to obtain a time-optimal trajectory for the landing for prescribed initial and final conditions, *i.e.*, the initial state vector of the orbit around the ellipsoid and the landing point at zero velocity in its surface. Hence, Equation 1 is translated into:

$$\begin{aligned} J &= \min \int_{t_0}^{t_f} dt \\ \boldsymbol{r}(t_0) &= \boldsymbol{r}_0 \quad \boldsymbol{v}(t_0)=\boldsymbol{v}_0 \\ \boldsymbol{r}(t_f) &= \boldsymbol{r}_f \quad \boldsymbol{v}(t_f)=0 \end{aligned} \tag{6}$$

Where $\boldsymbol{t}_0$ is the fixed initial time and $\boldsymbol{t}_f$ is a free final time.

Moreover, introducing the motion of the lander, described by Equation 2, it is possible to define a Hamiltonian formulation in the following shape:

$$\mathcal{H}=1+\boldsymbol{\lambda}_r\cdot\boldsymbol{v}+\boldsymbol{\lambda}_v\cdot(\boldsymbol{g}+f\boldsymbol{u}) \tag{7}$$

Where λ_r and λ_v are the co-state variables associated to Equation 2 respectively, and as already introduced, the target of our ML study.

To close the mathematical problem, the rates of change of the co-state variables must be determined. These are easily derived from the Hamiltonian, Equation 7 as the ODEs:

DOI: 10.13009/EUCASS2022-4761

$$\begin{aligned} \frac{d\lambda_{ri}}{dt} &= -\frac{\partial \mathcal{H}}{\partial r_i} = -\lambda_{vi}\frac{g_i}{r_i} \qquad \forall i \in \{x, y, z\} \\ \frac{d\lambda_{vi}}{dt} &= -\frac{\partial \mathcal{H}}{\partial v_i} = -\lambda_{ri} \qquad \forall i \in \{x, y, z\} \end{aligned} \tag{8}$$

The Pontryagin's minimum principle states that the optimal solution for a problem casted in this way is obtained when the thrust orientation (*i.e.*, the control variable) minimizes the Hamiltonian function during the whole trajectory. For this formulation of the problem, together with the aforementioned simplifications and hypothesis, it is possible to conclude that the thrust orientation vector that satisfies this condition is the following optimal law:

$$\boldsymbol{u} = -\frac{\boldsymbol{\lambda}_v}{|\boldsymbol{\lambda}_v|} \tag{9}$$

Furthermore, the partial derivative with respect to time of the Hamiltonian function for the time-optimal landing problem is null. This implies that its value will remain constant throughout the whole trajectory and, as the boundary conditions are also time-independent and t_f is a free variable, the transversality conditions require that the Hamiltonian at the final time will be zero.

It can be then concluded that the Hamiltonian function will be null during the whole time-optimal trajectory. This analytical result can be used to directly define the modulus of the co-states λ_v as a function of the rest of the problem variables, yielding:

$$|\boldsymbol{\lambda}_v| = \frac{1 + \boldsymbol{\lambda}_u \cdot \boldsymbol{v}}{\boldsymbol{u} \cdot \boldsymbol{g} + f} \tag{10}$$

Which is an additional condition to be checked, useful for validation of the ML methodology.

2.3 Training set generation: procedure and strategies

So far, the mathematical description of a time-optimal landing over a steady ellipsoid has been introduced, leaving the most crucial part, its resolution, open. This problem as indicated in Equation 6, requires for an initial and a final condition (although the final time of arrival is always left as a free parameter). Moreover, as we are following the indirect or Hamiltonian formulation, it is not enough to set these values but also the boundary conditions for the co-estates must be stated. This results into the common problematic of the indirect formulation of such an optimization problem: there is no information about the values of the initial nor final conditions for the co-states to satisfy.

This is nothing but a consequence of the lack of physical meaning or counterpart that in general present the co-states, resulting in several orders of magnitude of difference in their possible values depending on how these initial/final conditions are computed. As it was shown before, sometimes thanks to a specific shape of the Hamiltonian some extra relations might be found and used as first hints, still, this is a case-dependent situation. One of the reasons of having chosen the two-body ellipsoid landing problem is that we can use this extra information as an additional validation for the ML methodology that will be presented in Section 3. As a result, we will proceed without using this condition to avoid particularizing the general procedure and use it at the very end of the validation/result comparison.

Although some efforts are done to reach an initial or final guess on the co-states (being related by an ODE, knowing one, the other is easy to compute by integration) this tend to be a dead-end in optimization processes. The aim of this work is, as it was already introduced in Section 1, to take profit of ML techniques to shortcut this problematic, producing a NN that can learn the physics of the landing process and produce a valid initial guess for the co-states. The methodology to be followed will be fully described in the next section.

DOI: 10.13009/EUCASS2022-4761

Nevertheless, to feed the ML with useful data it is necessary to generate it by solving the time-optimal landing problem, leaving available the initial and final state of the lander together with the initial and final values of the co-states. This problem is still highly challenging for the same reason: the lack of information on initial or final co-states.

To tackle this situation and to be able to generate a training data set for the NN to learn we propose the following approach:

- The Asteroid/Ellipsoid surface will be meshed in terms of its latitude and longitude. As the ellipsoid comprises eight symmetric sectors, only one is required to be meshed, and the rest can be recovered by symmetry.
- For each of these mesh points, the problem will be propagated backwards following a time-optimal evolution, described by Equations 2 and 8.
- As only the interesting trajectories for a controlled landing are the ones in which the touch-down is performed perpendicularly to the object surface (*i.e.*, the lander velocity as well as its resultant acceleration are perpendicular to the object surface) extra restrictions can be imposed on the final value of the co-state $\lambda_v\left(t_f\right)$ leaving it unambiguously defined.
- This way, sweeping through a range of the possible values of $\lambda_r\left(t_f\right)$ for each meshed point, a consequent number of time-optimal trajectories ending in that point are recovered.
- The backwards propagation is stopped at a sphere around the asteroid of a radius equal to 10 times the ellipsoid higher semiaxis, providing the initial condition for the state vector and the co-states at the stopping point.

This process is illustrated in Figure 2.

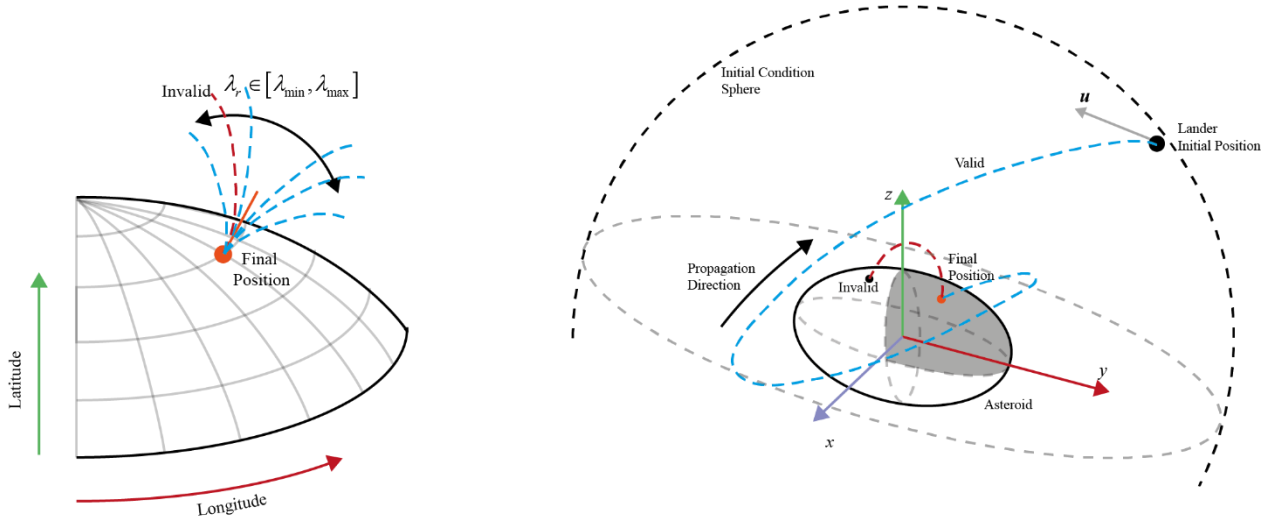

Figure 2: Meshing of the Asteroid Surface (Left) and backward propagation example with a Valid and an Invalid Trajectory (Right)

As it was mentioned, only those trajectories with velocity and acceleration perpendicular to the surface in the moment of the landing are of interest. And so, a relation between the thrust orientation and the rest of the values in Equation 2 appears:

$$\boldsymbol{u}\left(t_f\right)=\frac{1}{f}\left[k\boldsymbol{n}\left(t_f\right)-\boldsymbol{g}\left(t_f\right)\right] \tag{11}$$

Where $\boldsymbol{n}\left(t_f\right)$ is the unitary normal vector of the ellipsoidal surface at the landing point and k is a proportionality coefficient that must be chosen for the thrust to remain a unitary vector, yielding the expression:

DOI: 10.13009/EUCASS2022-4761

$$k = \sqrt{f^2 - \boldsymbol{g}(t_f)\cdot\boldsymbol{g}(t_f) + \left[\boldsymbol{g}(t_f)\cdot\boldsymbol{n}(t_f)\right]^2} + \boldsymbol{g}(t_f)\cdot\boldsymbol{n}(t_f) \quad \textbf{(12)}$$

As a result, it is possible to obtain a value for $\lambda_v(t_f)$ defined by Equations (9-12) for a certain landing point and thrust orientation.

This way it is possible to recover a training data set of trajectories that start over the "Initial Conditions Sphere" (ICS) and ends in a point of the meshed one eight of asteroid. If the training is successful, it should be possible to feed the NN scheme with points outside the sphere (yet, not very distant) and a landing point that doesn't coincide with the meshed grid points.

Although following the backwards propagation previously described it is in principle possible to obtain a number of families equal to: $N_{latitudes} \times N_{longitudes} \times N_{\lambda r's}$, in principle, enough for a solid training set of a Neural Network, there are some extra details to be taken into account regarding the backwards propagation.

Indeed, as only one eight of the ellipsoid surface was meshed, it might happen that the propagated trajectories "collide" with the asteroid before reaching the initial position sphere. These trajectories must be discarded as they will not provide a representative set for the training. Consequently, an extra step must be added to detect the collided trajectories, *i.e.*, the ones whose state vector gives a position that is under the ellipsoid surface.

An additional inconvenient of this procedure that must be checked to produce valuable data is that the orientation of the initial thrust vector must be pointing at the asteroid to avoid extra turns. This orientation will be fixed, as presented by Equation 9 by the initial co-state vector $\lambda_v(t_0)$, which is an output of the backwards propagation. Consequently, to properly bias the NN those time-optimal trajectories that output an initial co-state pointing outwards the asteroid are also discarded.

For this problem, a total of 380 million valid trajectories were generated following this procedure, leaving out a total of 35,62% of "colliding" or "bad initial pointing" ones, called, from now on, invalid trajectories. For each of the trajectories, the initial and final states vectors where registered together with the initial and final co-states. As the final velocities will be always 0, only 6+3+6 double precision scalars require to be saved per valid trajectory. This information is presented with more detail in Table 1.

Table 1: Initial and final orbital elements of the transfer

Meshed Parameter	**Variation Range**	**Sampling Rate**
Latitude	$[0^o, 90^o]$	1^o
Longitude	$[0^o, 90^o]$	1^o
$\lambda_{r,x}(t_f)$	$[-10, 10]$	0.5
$\lambda_{r,y}(t_f)$	$[-10, 10]$	0.5
$\lambda_{r,z}(t_f)$	$[-10, 10]$	0.5

Consequently, an important number of time-optimal trajectories were generated to properly train the networks. The final choice on sampling and variation range presented in Table 1 is the result of an iterative process altogether with different ML schemes, always having as a success criterion the minimization of the error at the learning process.

DOI: 10.13009/EUCASS2022-4761

Although a randomly generated mesh and sweep would in principle be more beneficial for the training processes of the NN to avoid biasing, random sampling ends up providing very similar results in terms of successful learning with the inconvenience that higher number of trajectories are required to obtain the same order of magnitude of valid ones.

Another issue regarding the trajectory set generation is the variation range for the final co-states. The presented range in Table 1 is a result from a heuristic iterative process, although it has not been studied to its last consequences. Indeed, the results with a simple ranging produced fairly good results and did not provide biasing when compared to variation ranges of higher or lower orders of magnitude. The in-depth analysis of how different ranges of final position co-states affect the quality of the learning process is left for future work.

3. Machine Learning Methodology

As it has been already described, the aim of this work is to take profit of ML schemes to provide unknown a-priori information in the indirect optimization formulation, particularly: the initial values of the co-states. Until the exponential growth of ML it was somehow common to address time-dependent problems (optimal, or not) in two categories: direct or classical, where from a certain set of initial condition an equation was solved marching forward in time to reach some final state; and inverse problems, where it was possible to deduce some of the system properties from its time evolution.

We follow in this work a similar approach for ML schemes. It has already been shown that NN can be used to model mathematical functions and, moreover, to model differential equations as predictors **[1-5]**. Particularly, Deep Learning schemes such as the Long-Short Term Memory networks (LSTM) have proven to be a powerful and precise tool as time series predictors **[11]**. These applications could be considered the direct approach in ML, and although their precision will always be far from the highly optimized and precise numerical methods, they can be used as data-driven predictors or even, as noise filters in real trajectories **[3]**.

Nevertheless, it is at the inverse problems that ML really is capable to compete and offer new solutions and shortcuts to highly complex problems. They are indeed mathematical correlators, and they do not require for time-ordered data **[12]**. Consequently, it is possible to feed a NN with the final state of a solution of an ODE to march backwards and predict its initial state, or even, mix initial and final partially known states to predict the rest of them. And, in the context of the optimal indirect formulation, it should be possible to feed a NN scheme with known initial and final states to predict the corresponding initial unknown co-states.

Throughout this section, a consistent implementation of this idea will be described and detailed, paying special attention to the way inputs and outputs of the network can be defined. Additionally, a test matrix to show the hyperparametric study of the ML scheme is justified and presented. Finally, a subsection will be dedicated to discussing the precision of the results and their sensibility to define the different success criteria for comparison between different schemes.

3.1 Neural Network models and training data processing

The training and validation data is recovered from the valid trajectories generated with the procedure described on subsection 2.3. As it was introduced, the initial and final states (in position and velocity) and co-states are available for time-optimal landing trajectories, starting at the ICS and ending in one of the meshed points of the asteroid's surface. This leaves for 15 variables available for training per trajectory: 6 for initial position and velocity, 3 for final position (as final velocity will be 0) and 6 initial and final co-states pairs.

The idea under the use of this NN scheme is that it will serve as a black-box predictor for the initial co-states (or a fair initial guess of them) from an initial state will reach a final position on the surface. Consequently, the NN scheme will have 9 channels or features as inputs and 6 outputs. The final co-states should not be used as input parameters as they are also unknown in a general problem.

Having a low output-input features ratio is often detrimental for NNs, as they then work in a higher-dimensional parametric space and the fitting accuracy decreases substantially. This issue is often tackled by separating the network in independent ones, with the same number of inputs and one or few output features. Nevertheless, for this problem, being all the features physically related by ODEs, it might be beneficial to leave them related to help the NN scheme "learn their physical relation" so the choice is not as straightforward as it might seem. As a result, three different schemes are already defined:

DOI: 10.13009/EUCASS2022-4761

- Model 1: One NN scheme that includes all inputs and outputs, *i.e.*, a 9x6 scheme. It is expected that the NN captures the physical relation between all variables.
- Model 2: The output features are separated into the velocity related initial co-states, $\lambda_v(t_0)$, and the position ones $\lambda_r(t_0)$. Resulting in a Model with two independently trained NN with 9 inputs and 3 outputs each.
- Model 3: All 6 of the output features are separated, resulting in a Model with six independently trained NN with 9 inputs and 1 output each. In this model, the reduction of the parametric space is prioritized.

The schematics of these 4 models is presented in Figure 3, assuming each of the NN schemes can be composed by any kind of NN layers (Standard or Fully Connected, LSTM, Convolutional, etc.).

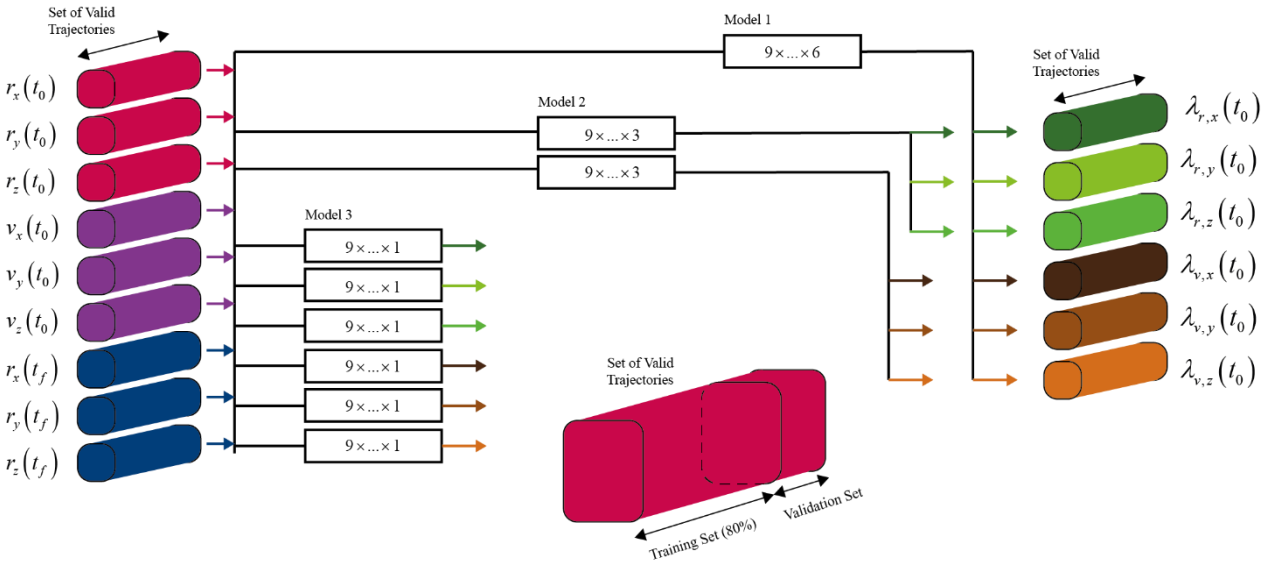

Figure 3: Schematics of the 3 different Models tested, showing inputs and outputs for each case.

As it can be observed, it could be possible to join all independent outputs from Model 2 or 3 with a closing, independent NN scheme that trains on how the output features are related between themselves. This idea, although initially considered is left for future work.

To produce a proper training, each feature must be non-dimensionalized independently (whether it is an input or output). This is problematic, specially considering that the dataset was generated with backwards propagation schemes and the orders of magnitude of the initial conditions are highly sensitive to a variation of the selected final co-states. Consequently, an extra trim of the data must be performed, removing trajectories that produce initial co-states that are several orders of magnitude higher or lower than the rest. This is achieved by setting a threshold value for the co-state, filtering the very separated ones. The result is a loss of around 300.000 trajectories of the 380 million that were considered initially valid. Once the final trim is performed, each of the features is non-dimensionalized as the whole set varies between -1 and 1. Also, to avoid possible biasing from ordered data, the set of trajectories is shuffled arbitrarily before the trim and then, during every epoch of the training.

Once the set of trajectories is separated into inputs and outputs and preprocessed, it must be divided into the information that will be used to train and test the NN scheme and the one left for validation. For this problem, as illustrated in Figure 3, 80% of the data is used for training while the 20% left is kept out of the process for validation. Additionally, as the velocity co-states must satisfy the conditions presented through Equations (9-12) this extra validation step is included.

For each of the NN Schemes on the model, an hyperparametric study on the layers and number of connexions has been performed, selecting 3 possible configurations:

- Classic: The scheme is composed by Fully Connected Layers (FCL).
- Convolutional: The scheme includes a 1D Convolutional NN with a small filter size that precedes FCL.
- Deep Learning (DL): The scheme is composed by LSTM layers.

Additionally, the number of connexions is also varied, leaving, for each configuration inside each model:

- HQ: at least 3 Layers containing at least 500 connexions each of the FCL and at least 200 for the LSTM.

DOI: 10.13009/EUCASS2022-4761

- LQ: 1 or 2 Layers containing less than 100 connexions each of the FCL and less than 20 for the LSTM.

With this information, the test matrix for each of the models is fully defined. To avoid extra complications, all the NN schemes inside a NN model are chosen to be the same, although they could be left open to try different combinations.

Regarding the training process, all the models were trained through 10.000 epochs with a piecewise variable learning rate that, starting in 0.01, decreased a factor of 0.95 every 100 epochs. The NN training process was performed by means of Matlab's *trainNetwork* function **[13]**.

3.2 Errors and success criteria for the NN Models

Although there is enough data and variety of Models to train, and it is possible to reach high precision with the proper NN combinations **[12]** the results will be obtained with a certain error depending on the training process and the shape of the data. For this application, we deal with highly sensitive equations at which, a small error in the initial co-states produces a different trajectory and landing point.

Nevertheless, it is enough to produce a fair initial guess for the initial co-states as then, a non-lineal iterative optimal programming problem (iterative NLP) can be proposed to reach the correct landing point **[14]**. Usually, without such an "educated" initial guess it is highly difficult to reach convergence, so apart from the error on the predicted co-states from the NN Models, a more accurate criterium can be defined: A model will have a better performance or, betted accuracy if it produces a set of initial co-states that, fed into the iterative NLP requires for the less number of iterations to converge into the actual solution.

For this problem, we select the modified Leven-Marquardt (LM) as implemented in MINPACK **[15]**, with a tolerance of 10^{-3} for both the sum of squares and the approximate solution, and we retrieve the number of evaluations of the objective function as a measure of the required iterations to reach convergence.

Therefore, two measures of the error of the approach are presented:

- RMSE: The RMSE of the prediction that the trained Model produces using the validation set as input/output.
- ITER: The number of iterations that the LM algorithm (with a tolerance of 10^{-3}) requires to converge at the desired landing point.

Finally, we can recover the relation the velocity co-states must satisfy to obtain an estimated value of them by computing their final co-state and then propagating it backwards with the predicted model values of the spatial co-states. This measure of the approximation is not a rigorously defined one, still it is an extra indicator, and it might be useful for future error improvement.

Figure 4 illustrates the different errors that can be obtained for the presented methodology:

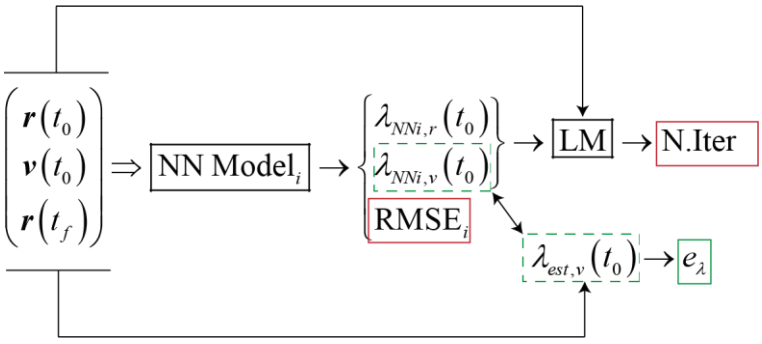

Figure 4: Scheme of the Errors and measures of the accuracy of the NN Model.

DOI: 10.13009/EUCASS2022-4761

4. Model results for the Landing Problem

Throughout this section the training and the results of the selected benchmark, *i.e.*, the co-state prediction for a time-optimal landing trajectory over an ellipsoid, are presented together with some extra tests that have been performed over the trained network.

Once the test matrix and the error criteria are defined, we carry out an hyperparametric study for each of the models looking to reduce the values of the RMSE and the number of iterations to reach convergence. As the numerical cost of each one of these processes (training and NLP iteration) was computationally very expensive, the number of hyperparameters to be analyzed was only 2: the number of layers and the connections per layer, for both HQ and LQ configurations.

The starting point for each kind of NN (the 1D Convolutional is not subjected to this study) is:

- FCL: 500 and 10 connections and 2 and 1 layers for HQ and LQ respectively
- LSTM: 200 and 10 connections and 2 and 1 layers for HQ and LQ respectively

Table 2: Results of the hyperparametric analysis

Model	Configuration	Quality	Number of Layers	Number of Connections	Improvement of RMSE
1	Classic	HQ	5	1000	25%
		LQ	2	100	6%
	Conv	HQ	6	500	10%
		LQ	2	100	2.5%
	DL	HQ	3	250	13%
		LQ	2	50	6%
2	Classic	HQ	4	1200	17%
		LQ	2	100	3%
	Conv	HQ	3	500	11%
		LQ	2	100	3%
	DL	HQ	2	300	9.5%
		LQ	2	50	2%
3	Classic	HQ	5	2000	17%
		LQ	2	100	5%
	Conv	HQ	5	500	6.5%
		LQ	2	200	3%
	DL	HQ	2	200	7%
		LQ	2	20	6%

All the connections are left the same for each of the layers during the analysis, and it is important to recall that for the convolutional configuration, the layers that are varied are the FCL that are present after the 1D Convolution Layer.

The results shown on Table 2 might look contradictory as the Model 3 is supposed to be the one with higher precision. Yet, what is presented is the improvement achieved on the RMSE after the hyperparametric analysis, not the actual values of the error, that are shown in Table 3 and are consistent with what has been presented so far.

An interesting conclusion of this analysis is that the LQ configurations present a lower improvement than the HQ when increasing their complexity (layers and connections). One can then conclude that, for this specific set of data, the NN Models require for higher order of complexity to show an actual improvement on its learning process.

Once the optimal hyperparameters have been selected, the resulting test matrix with its errors is presented in Table 3. It is important to recall that all the errors in this Table are obtained by predicting the validation set with the trained network. Consequently, the presented RMSE is the total error for all the set for all the features (although it is presented non-dimensionalized and might result misleading). On the same manner, the number of iterations is an average of the converged solutions for the validation set and does not consider the non-converged cases. Therefore, an extra column is added showing the % of non-converged cases.

DOI: 10.13009/EUCASS2022-4761

It is important to recall that the total size of the validation set is the 20% of the full valid trajectory set, giving a total of 78 million validation trajectories, randomly extracted from the full set. Consequently, the value of the total RMSE, added for all the features, will inevitably decrease if the number of trajectories does so, therefore the same number of trajectories has been maintained throughout the study and the results although for the DL schemes this was translated into a very computationally expensive training.

The computations have been performed using a 10th Gen Intel(R) Core(TM) i9-10900X @ 3.70 GHz with 128 GB of RAM, equipped with a GPU NVIDIA GeForce RTX 2080Ti. All the training processes have been performed in GPU while the NLP problem resolution runed on CPU.

Table 3: Comparison Model Performance for the Validation Set

Model	Configuration	Quality	RMSE	% Of Converged Cases	Iterations	
					Average	Min
1	Classic	HQ	$2.1x10^{-1}$	63%	530	24
		LQ	$3.0x10^{-1}$	59%	620	13
	Conv	HQ	$2.2x10^{-1}$	63%	510	30
		LQ	$3.5x10^{-1}$	60%	540	41
	DL	HQ	$1.8x10^{-1}$	65%	612	22
		LQ	$5.2x10^{-1}$	59%	700	16
2	Classic	HQ	$1.7x10^{-2}$	71%	517	5
		LQ*	$2.4x10^{-2}$	71%	550	28
	Conv	HQ	$1.2x10^{-2}$	72%	570	14
		LQ*	$4.7x10^{-2}$	70%	530	30
	DL	HQ	$2.2x10^{-2}$	71%	601	19
		LQ	$1.3x10^{-1}$	61%	653	57
3	Classic	HQ	$8.3x10^{-3}$	81%	468	43
		LQ	$1.0x10^{-2}$	72%	505	21
	Conv	HQ	$9.2x10^{-3}$	79%	531	27
		LQ*	$2.1x10^{-2}$	71%	576	12
	DL	HQ*	$1.9x10^{-2}$	74%	1030	6
		LQ	$9.2x10^{-2}$	65%	890	15

It is possible to categorize the results obtained in Table 3 into several aspects of interest, regarding the different models, their associated error, and the percentage of converged cases.

- For this specific problem, increasing the ratio output features vs input features is the way to improve the quality of the training and the accuracy of the NN Model. Indeed, the training results with Model 3, that is, 1 independent training process per output feature reaches almost two orders of magnitude on the approximation of the error and provide a reasonably high % of converged cases.
- The DL configuration will require for more epochs (or an additional hyperparametric analysis) to reach the order of convergence that the Classic one shows. Moreover, the computational cost of the DL configurations is 4 times higher (in some cases, even more) than the other two. Although this configuration underperforms for the study, it is possible that better computational resources and a more thorough hyperparametric analysis will succeed in obtaining similar results than the other.
- The Convolutional configuration helps to avoid overfitting the data, although, in general provides similar results than the Classic study.
- The cases presented with an asterisk in Table 3 correspond to overfitted solutions. Meaning that the RMSE of the validation data starts to grow after a certain epoch rather than keep decreasing with the training process. Special attention in this aspect goes to the DL HQ results from Model 3, that technically should avoid overfitting, yet the selected configuration of layers and connections still yields overfitting. It is left for future work to improve the learning procedure, tackling the overfitting problematic with cross-validation techniques.
- The LM problem is as sensitive as it was expected, staying always far from the 100% of convergence. The number of iterations show a certain gaussian behavior in its distribution around an average value that ranges from 500 to 700 evaluations of the objective function this is somehow an expected result as there are 78 million trajectories being tested and so, the central limit theorem will appear if there is no hard bias on the results.

Once the Model 3, Classical Configuration, HQ (M3CHQ) has been identified as the best candidate to work in this problem, three independent test cases are selected to show the performance of the numerical tool, that is, the trained NN Model.

To improve the performance of the network, an extra 10.000 epochs are added to the training process. Both the HQ and the LQ configurations of the M3C are used for this problem

Test Case 1 (TC1): Landing in a position outside the surface grid used for the training and validation process, starting inside the ICS.
Test Case 2 (TC2): Landing in a position outside the surface grid used for the training and validation process, starting in an initial condition perturbed a 10% outside of the ICS
Test Case 3 (TC3): Landing in an asteroid with extra spherical harmonics (the first 9) to simulate different geometric shape, being the first one the ellipsoid of the training process and the perturbation of the extra harmonics a 10% on its gravitation field. Density is still left constant, and the initial condition is part of the ICS

Table 4 summarizes the results in terms of errors and convergence

Table 4: Extrapolation Test Cases

Test Case	Models	RMSE	Convergence?	Number of Iterations
1	M3CHQ	7.11×10^{-3}	Y	954
	M3CLQ	9.37×10^{-3}	Y	1135
2	M3CHQ	2.55×10^{-2}	Y	5154
	M3CLQ	8.36×10^{-1}	N	-
3	M3CHQ	1.03×10^{-1}	Y	7623
	M3CLQ	1.34×10^{0}	N	-

Consequently, the performance of the Model 3 remains consistent for slight variations on the initial and final conditions, increasing however the number of iterations required for convergence in the NLP problem. This shows the potential of NN schemes as generators of good initial guesses for the co-states of optimal control problems.

This study was performed using prefabricated layers and configurations included in Matlab's toolboxes, better results in terms of error and convergence can still be achieved for a more in-depth analysis and design of the Models.

The fact that the model manages to converge in a non-elliptical geometry although being trained for elliptical test cases shows that Transfer Learning is possible for at least, similar problems. Testing the limits of this transfer learning process is left for a future study with higher perturbed asteroids, with rotating frames and different density distribution.

5. Conclusions

This work presents a numerical methodology to tackle indirect optimal problems by means of NN schemes in what it can be considered an inverse approach. First, the mathematical general formulation of a time-optimal landing over an ellipsoidal-like asteroid with the thrust orientation as control variable is presented, focusing on the hypothesis and simplifications that can be performed for this test case. The problematic is analyzed, showing its parametrization and the main obstacles associated: the lack of information regarding the co-states at initial or final for the optimization problem. Additionally, some extra relations that arise in this problem due to some properties the Hamiltonian of the problem must satisfy are presented, showing that part of the co-states can be determined for this test case.

DOI: 10.13009/EUCASS2022-4761

Once the mathematical details are presented, the dataset that will be used for the training the NN Models is generated by means of a backward propagation and a sweep over the unknown final co-states. To simplify the amount of data only one sector of the asteroid is meshed, taking special attention to trajectories that, when propagated backwardly in time might crash over the surface or provide a thrust vector pointing outside the asteroid. A total number of 380 million valid time-optimal trajectories are generated, saving for the training process the initial and final state vectors and the initial and final co-states.

Using the generated data, 3 different NN Models are trained, each of them with a different strategy to deal with the output features of the network. The difference they present is the ratio input features vs output features, in this manner, 3 different models are trained to confront a more physically oriented approach vs a more data fitting approach. The higher number of output features, the higher the information the network has regarding the physical correlation between output features. On the other hand, it is often more efficient to fit/train data that goes from a high number of input features to a low number of output features.

The 3 Models are trained and a hyperparametric iterative study is performed to find the best configuration of the networks and layers composing the models. The success criteria for this training is selected as the RMSE and the number of iterations required to converge from the predicted co-states to the actual values by means of a NLP problem. This training process show that the best Models under the aforementioned success criteria are the ones that present higher input features vs output features ratio. Simple or Fully Connected Layers, Convolutional Layers and DL layers are tested as different configurations for the 3 models, showing that the best configuration is a combination of high number of connections FCL as they present the best compromise between calculation time and lower RMSE.

Finally, the best Models, already trained are tested in an environments that differ from the dataset originally generated: Starting and ending on different positions that were not present at the dataset and trying to find the co-states of a Landing problem over a slightly non-elliptical ellipsoid. The results show that, although with lower performance than the dataset environment, the NN Model is capable to provide a good initial guess for the unknown co-states.

It is left for future work the further investigation in the Transfer Learning processes altogether with the training of a Model composed by different combinations of Layers and Networks than the ones performed in this paper. Also, more complex mathematical landing problems will be studied for dataset generation: such as non-elliptic asteroids (with random shapes defined by its spherical harmonics), rotating bodies or landing environments in n-body problems.

Acknowledgments

This research was supported by Project Grant PID2020-112967GB-C33 by the Spanish State Research Agency.

References

[1] de Celis, R., Solano-López, P., & Cadarso, L. (2021). Sensor hybridization using neural networks for rocket terminal guidance. Aerospace Science and Technology, 111, 106527.Li, Haiyang, Hexi Baoyin, and Francesco
[2] Li, H., Baoyin, H., & Topputo, F. (2019). Neural networks in time-optimal low-thrust interplanetary transfers. IEEE Access, 7, 156413-156419.
[3] Urrutxua, H., Solano-López, P., Gutierrez-Ramon, R., Barea, A. (2021), Assessment of Hybrid Machine Learning Applications on Trajectories in the CR3BP, In: *Proceedings of the International Astronautical Congress, IAC*, C1.
[4] Yu, Y., Yao, H. and Liu, Y. (2019), Aircraft dynamics simulation using a novel physics-based learning method, *Aerospace Science and Technology*, Vol. 87, 2019, pp. 254–264.
[5] Cheng, L., Li, H., Wang, Z., & Jiang, F. (2020). Fast solution continuation of time-optimal asteroid landing trajectories using deep neural networks. *Acta Astronautica*, *167*, 63-72.
[6] Schiassi, E., D'Ambrosio, A., Drozd, K., Curti, F., & Furfaro, R. (2022). Physics-informed neural networks for optimal planar orbit transfers. Journal of Spacecraft and Rockets, 59(3), 834-849.
[7] Atchison, J. A., Ozimek, M. T., Kantsiper, B. L., & Cheng, A. F. (2016). Trajectory options for the DART mission. *Acta Astronautica*, *123*, 330-339.
[8] Ozaki, N., Yamamoto, T., Gonzalez-Franquesa, F., Gutierrez-Ramon, R., Pushparaj, N., Chikazawa, T., ... & Takashima, T. (2022). Mission design of DESTINY+: Toward active asteroid (3200) Phaethon and multiple small bodies. Acta Astronautica, 196, 42-56.
[9] Bryson, A. E., & Ho, Y. C. (2018). Applied optimal control: optimization, estimation, and control. Routledge.
[10] Chandrasekhar, S. Subrahmanyan Chandrasekhar. An Introduction to the Study of Stellar Structure, 2.

DOI: 10.13009/EUCASS2022-4761

[11] Gers, F. A., Eck, D., & Schmidhuber, J. (2002). Applying LSTM to time series predictable through time-window approaches. In Neural Nets WIRN Vietri-01 (pp. 193-200). Springer, London.
[12] Yadav, N., Yadav, A., & Kumar, M. (2015). *An introduction to neural network methods for differential equations* (Vol. 1, p. 114). Berlin: Springer.
[13] *trainNetwork*. MathWorks. https://es.mathworks.com/help/deeplearning/ref/trainnetwork.html
[14] Numerical Optimization: Theoretical and Practical Aspects. J. Frédéric Bonnans, J. Charles Gilbert, Claude Lemaréchal, Claudia A. Sagastizábal. SpringerLink. 2006.
[15] Moré, J. J. (1978). The Levenberg-Marquardt algorithm: implementation and theory. In Numerical analysis (pp. 105-116). Springer, Berlin, Heidelberg.