Final Study Project
Master 2 Recherche ATSI

Neural network surrogate modelling for Earth observation missions

Author: Marcel CACERES
Supervisor: Kristen LAGADEC
Emmanuel RACHELSON

Preliminary Study

September 12, 2012
Abstract

The first part of the document is a review of the main characteristics of surrogate models and a special attention has been given to Artificial Neural Networks (ANN) in order to develop these models. Their architecture, the input and output spaces dimension, the learning process or their generalisation capabilities are some of these characteristics. In the second part of the document we try to figure out how to apply these techniques in the context of an Earth observation mission. In a first application we are able to reduce the computational cost of the model employed on the ground during the planification of the mission. A second application is developed to be implemented onboard in order to recompute the manoeuvres with a reduced computational cost. It is shown that this could increment the satellite’s reactivity and pointing accuracy.

**Key Words:** Surrogate modelling, feedforward neural network, satellite attitude manoeuvre, Earth observation mission scheduling.
Contents

Introduction 1

1 Earth observation and agile satellites 5
   1.1 Scheduling observation problem . . . . . . . . . . . . . . . . . 6
   1.2 Onboard computational limitations . . . . . . . . . . . . . . . . 8
   1.3 Summary . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 8

2 Surrogate models 11
   2.1 Input data sampling . . . . . . . . . . . . . . . . . . . . . . . 11
   2.2 Modelling process . . . . . . . . . . . . . . . . . . . . . . . . 12
   2.3 Different surrogate models structures . . . . . . . . . . . . . . 14
   2.4 Discontinuous objective functions . . . . . . . . . . . . . . . . 16
   2.5 Summary . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 16

3 Artificial Neural Networks 19
   3.1 Why Artificial Neural Networks? . . . . . . . . . . . . . . . . . 19
   3.2 Neuron model and neural network . . . . . . . . . . . . . . . . 20
   3.3 Supervised learning . . . . . . . . . . . . . . . . . . . . . . . 23
   3.4 Architecture selection . . . . . . . . . . . . . . . . . . . . . . 24
   3.5 Other important aspects . . . . . . . . . . . . . . . . . . . . . 25
   3.6 Applications . . . . . . . . . . . . . . . . . . . . . . . . . . . . 25

4 Implementation 27
   4.1 Attitude manoeuvres and surrogate models . . . . . . . . . . . 27
      4.1.1 Key variables to model attitude manoeuvres . . . . . . . 28
      4.1.2 Data generation . . . . . . . . . . . . . . . . . . . . . . 30
      4.1.3 Feedforward neural network . . . . . . . . . . . . . . . 30
      4.1.4 Performance computation . . . . . . . . . . . . . . . . 33
      4.1.5 Methodology . . . . . . . . . . . . . . . . . . . . . . . 35
   4.2 On the ground . . . . . . . . . . . . . . . . . . . . . . . . . . . 35
      4.2.1 Agility model . . . . . . . . . . . . . . . . . . . . . . . 35
      4.2.2 Design variables . . . . . . . . . . . . . . . . . . . . . 36
      4.2.3 Data analysis . . . . . . . . . . . . . . . . . . . . . . . 38
      4.2.4 Surrogate overview . . . . . . . . . . . . . . . . . . . . 40

Preliminary study 5
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.2.5</td>
<td>Results</td>
<td>40</td>
</tr>
<tr>
<td>4.3</td>
<td>Onboard</td>
<td>41</td>
</tr>
<tr>
<td>4.3.1</td>
<td>Manoeuvre computation</td>
<td>42</td>
</tr>
<tr>
<td>4.3.2</td>
<td>Design variables</td>
<td>42</td>
</tr>
<tr>
<td>4.3.3</td>
<td>Data analysis</td>
<td>43</td>
</tr>
<tr>
<td>4.3.4</td>
<td>Surrogate overview</td>
<td>45</td>
</tr>
<tr>
<td>4.3.5</td>
<td>Results</td>
<td>45</td>
</tr>
<tr>
<td>5</td>
<td>Conclusions</td>
<td>47</td>
</tr>
<tr>
<td>6</td>
<td>Future work</td>
<td>49</td>
</tr>
<tr>
<td>A</td>
<td>Satellite kinematics</td>
<td>51</td>
</tr>
<tr>
<td>B</td>
<td>Backpropagation algorithm</td>
<td>53</td>
</tr>
</tbody>
</table>
List of Figures

1. Pleiades Satellite ............................................ 2
   1.1 Roll, pitch and yaw rotation axis ................. 6
   1.2 Agile Vs non agile ....................................... 7

3.1 Single-Input neuron ........................................ 21
3.2 Multiple-Input neuron ..................................... 21
3.3 Layer of S Neurons ........................................ 22
3.4 Three-Layer Network ..................................... 22

4.1 Bang-bang manoeuvre ..................................... 30
4.2 Various sigmoid transfer functions .................. 31
4.3 Error evolution when increasing network size .... 32
4.4 Onboard manoeuvre control law ....................... 35
4.5 Agility model ............................................ 36
4.6 Design variables for the first application ........ 37
4.7 Relation between variables for the agility model 39
4.8 Relation between variables, including $h_b$ .......... 39
4.9 Models the agility model ................................ 40
4.10 Data analysis for rest-to-rest manoeuvres ....... 44
4.11 Data analysis for non rest-to-rest manoeuvres, $w_i = \frac{w_i}{4}$ ... 44
4.12 Data analysis for non rest-to-rest manoeuvres, $w_i = h_b$ .... 45
4.13 Models for onboard $T_{man}$ and $h_b$ estimation .... 45
4.14 Model for onboard $T_{man}$ estimation .............. 45
List of Tables

4.1 $T_{\text{man}}$ estimation . . . . . . . . . . . . . . . . . . . . 40
4.2 $T_{\text{man}}$ and $h_b$ estimation . . . . . . . . . . . . . . . . . . 41
4.3 $T_{\text{man}}$ and $h_b$ estimation onboard . . . . . . . . . . . . 46
4.4 $h_b$ estimation onboard . . . . . . . . . . . . . . . . . . . . . . . 46
Introduction

The title of this study is quite general and it reflects perfectly the objective of this internship which could be divided in two main parts. Firstly, Earth observation missions and secondly, neural network surrogate modelling which will be applied in the context of Earth observation missions.

This internship has been developed as an M2R project, so it will have a part focused on research. This is why the first part of the document is intended to be a bibliographic review. Research has been mainly focused on surrogate models, and more specifically on artificial neural network models. We started studying the basic theory and afterwards we focused on searching for examples of application that might have been previously done in the space domain.

With all of this information we tried to find out how surrogate models could help in the various computations and simulations related with Earth observation missions and more specifically for an Agile Earth Observation Satellite. Planning the day-to-day life of this kind of mission involves a large-scale optimization due to the huge combinatorial complexity. Some of the CPU-intense models involved in this process might be replaced by CPU-efficient approximations that would be done with surrogate models. During the project we will design those surrogate models with neural networks. An example of agile earth observation satellite is the Pleiades\(^1\) program, which can be seen in Fig. 1. This second part, which involves the application of the bibliographic review results, is also included in this document as it is going to validate the SUPAERO course too. Actually, this document is a preliminary version of the final report because the internship will continue for two and a half months more during which we will keep on working on this subject.

Trying to summarize the objective of the internship in a sentence, we could say that we must create a solid basic knowledge about the building process of neural networks surrogate models and find out if they could be

---

\(^1\)CNES satellite, mainly developed by ASTRIUM SAS
NEURAL NETWORK SURROGATE MODELLING FOR EARTH OBSERVATION MISSIONS

a realistic solution to be implemented for the approximation of more computationally complex algorithms that are involved in the Earth observation mission planing.

In the first chapter, we will briefly introduce the main characteristics of an Earth observation mission and the most important differences introduced by agile satellites. In the second chapter, the concept of surrogate model and some of its possible applications are presented. In the third chapter, feedforward neural networks are described as one possible way of creating a surrogate model. The architecture of the network, the dimensions and its generalisation capabilities are more deeply discussed. In the fourth chapter we will show how this tool can help in various steps of an agile Earth observation satellite day-to-day scheduling. This last chapter is not a bibliographic review but how we apply the previous concepts in order to see if neural networks would be suitable for this problematic. At the end we will find some conclusions and the future work that could be done.

Figure 1: Pleiades Satellite

Presentation of the company

The European Aeronautic Defence and Space company (EADS) is a global European aerospace and defence corporation and a leading defence and military contractor worldwide. The company was created in 2000 by the merger of three companies: Daimler Chrysler Aerospace AG (Germany), Aérospatiale Matra (France) and Construcciones Aeronuticas SA (Spain). It is now led by the CEO Tomas Enders. The group includes:

- Airbus as the leading manufacturer of commercial aircraft, with Airbus Military covering tanker, transport and mission aircraft,
- Eurocopter as the world’s largest helicopter supplier,
0. INTRODUCTION

- Astrium as the European leader in space programmes,
- Cassidian as a provider of comprehensive and integral systems solutions for aerial, land, naval and civilian security applications.

Overall, the company develops on markets like civil and military aircraft, as well as communication systems, missiles, space rockets, satellites and related systems. EADS is headquartered in Leiden, the Netherlands, and operates under Dutch law.

ASTRIUM

Astrium is an aerospace subsidiary of EADS and provides civil and defence space systems and services. In 2008, the company had 15,000 employees in France, Germany, the United Kingdom, Spain and the Netherlands. Its three main areas of activity are:

- **ASTRIUM SATELLITES**, involved in the manufacture of spacecraft used for science, Earth observation and telecommunication, as well as the equipment and subsystems used therein and related ground systems.
- **ASTRIUM SPACE TRANSPORTATION** for launchers and orbital infrastructure.
- **ASTRIUM SERVICES** for the development and delivery of satellite services.

Department

My internship is taking place in the ACE64 department, which means AOCS/GNC and Flight Dynamics Advanced Studies and is located in Toulouse. The ACE64 department leads R&D in the field of AOCS/GNC and aims at developing new techniques for automatic control, developing innovative AOCS/GNC algorithms and actuators/sensors, prototyping innovative functional chains.

The ACE64 department is divided into two specialized clusters:

- **Automatic Control, GNC (Guidance, Navigation, and Control) and Flights Dynamics**
- **Micro dynamics, AOCS (Attitude and Orbit Control Service)**, where my project is taking place
Earth observation and agile satellites

During the last decades, the presence of Earth observation satellites has increased and they are of great interest for various fields like cartography, environmental study or defence. Earth observation satellites are platforms that usually follow a circular, sun-synchronous and quasi-polar \( (i = 98^\circ) \) orbit. These platforms are equipped with various optical instruments which work at different wavelengths. During their mission they capture images of specific zones on the Earth depending on the requests from various users, including government, research institutions and corporations.

An example of this kind of satellites would be the SPOT program. This family of satellites are equipped with optical instruments which are pointed to Earth with the help of mobile mirrors. The rotation of these mirrors is done about the roll axis of the local orbit reference frame\(^1\). Thanks to the mirrors, images from either side of the satellite’s ground path can be taken at the same time that the satellite advances along the orbit.

After the success of the SPOT program, the Centre National d’Etudes Spatiales (CNES) started the creation of the new generation of Earth observation satellites, called Pleiades. This new generation of satellites belong to the called agile Earth observation satellites. The Pleiades program is formed by two satellites that follow the same circular sun-synchronous orbit. Each of the satellites is equipped with a high resolution optical instrument that allow them to provide extremely detailed images of the Earth, with a 70cm resolution and a scan swath of 20km. Each satellite is able to supply up to 450 images per day. The time-scales for accessing the information could be reduced to less than 24 hours with two satellites in operation, offering daily revisits to any point of the globe. On the other hand, the size of this optical instrument does not allow the use of mirrors to redirect the light. In this case the instrument is fixed to the satellite’s body and it is the whole satellite’s body which is in charge of doing the necessary rotations to point the instrument to the desired Earth zones.

\(^1\)Defined with the orbital plane and the satellite position and velocity.
Agile satellites sometimes employ Control Moment Gyroscope (CMG) as their actuator which allow them to achieve rapid attitude manoeuvres with high precision. In the case of Pleiades, manoeuvres of $10^\circ$ can be achieved in $10s$, or even $60^\circ$ in $21s$ [25]. In addition, this rotational performance can be achieved about any mix of the three axis of the reference frame (roll, pitch and yaw), as shown in Fig. (1.1).

1.1 Scheduling observation problem

As previously shown, agile Earth observation satellites are capable of taking an image of a fixed point over the Earth from an infinity of positions of its trajectory because all azimuths can be used to capture a scene. On the other hand, non-agile Earth observation satellites could only take an image from one unique point of their trajectory as just vertical azimuth is possible.

The example in Fig. 1.2 shows the effect of agility when choosing the images that will be taken out of all the possible ones. In the case of an Earth observation satellite (Fig. 1.2(a): only scenes 1, 3 and 5 can be taken) and for an agile Earth observation satellite (Fig. 1.2(b): the five images can be taken). The agility allows to enlarge the visibility windows of each image, so the starting date can be advanced or delayed with the objective of maximising the number of images taken.

The mission of an agile Earth observation satellite is to take images of specific zone of the Earth which are requested by the users or clients. Actually, the number of images to be taken is higher than the ones that the satellite will be able to take in a single pass. This is the origin of the problematic of choosing and scheduling the observations of an agile Earth observation satellite. In fact there are many constraints involved in this problem, which is at the end an optimization problem with the objective of maximising the profit done with the images taken by the satellite. Some of these constraints are listed below:

1. Power and thermal availability.
1. EARTH OBSERVATION AND AGILE SATELLITES

(a) Non agile satellite

(b) Agile satellite

Figure 1.2: agile Vs non agile as shown in [12]

2. Limited visibility time along the orbit. With the mostly used sun-synchronous orbit of the Earth observation satellites, each orbit takes around 100 minutes.

3. Revisiting limitations.

4. The needed time to take a whole image, which depends on its size because Earth observation satellites use the displacement along its orbit to capture the scene.

5. Reduced storage capacity.

6. Availability of ground stations to download all collected data.

7. Transition manoeuvres. The attitude change between two consecutive images is not instantaneous and requires a transition manoeuvre. Satellite’s kinematics allows to compute the minimum time needed to do a transition manoeuvre between every image pair. This minimal time does not only depend on the physical location of the images and the acquisition azimuths, but also depends on the time when the manoeuvre starts.

8. Angular constraints related to each image. For example, an image could require a pitch under $20^\circ$ and a roll under $35^\circ$ with respect to the vertical axis.
The objective of this study is not to solve the agile Earth observation satellite scheduling problem, which has already been resolved by different means and remains as a major research field. See [15], [37] and [5] for more details. What we wanted is to show the combinatorial complexity of this problem and specially, the relevance of the computation of the transition manoeuvre minimal time between images, which becomes a key factor in the optimization problem.

To compute this maneuver time, it is necessary to use the satellite’s kinematics. It is not a difficult step but it is computationally expensive as it requires numerical integration. A short summary of spacecraft dynamics is included in Appendix A. For more details see [31] and [36]. In addition, the computation method for this transition time must be time efficient because it is going to be repeated thousands of times during the optimisation for the scheduling. Some researches have been focused on this specific subject. In [13], an improvement method based on a spacial kinematic model is given to acquire the attitude changing duration. In [12], the estimation of the shortest duration of an attitude change is treated as a constraints satisfaction problem.

1.2 Onboard computational limitations

If we take the example of the Pleiades satellites, they are equipped with ERC-32 processors, which have a RISC² architecture. This processor is in charge of various on-board computations, instrument control and data processing from the payload and is also in charge of the Attitude and Orbit Control System (AOCS). However, this processor is not reasonably suitable for the trajectory computing methods that are used on the ground, in terms of computation time. In particular, the computation of the shortest duration of an attitude change is based on an iterative algorithm whose execution time is not strictly deterministic. Because of these reasons, the scheduling of the day-to-day mission is done on the ground. The dates and attitudes corresponding to the images that must be taken are uploaded to the satellite. Onboard, the AOCS is in charge of executing all the required manoeuvres.

1.3 Summary

In this chapter we have seen the complexity increase of the day-to-day scheduling task of an Earth observation mission with an agile satellite like Pleiades compared to previous generations of Earth observation satellites.

²Reduction Instruction Set Computing
1. EARTH OBSERVATION AND AGILE SATELLITES

More precisely, we have underlined the importance of the manoeuvre computation cost inside the optimization process for the scheduling problem. As the manoeuvre computation method will be executed thousands of times, it is necessary to reduce at the maximum its time consumption.

Taking into account this context, the objective of this study is to explore the possibilities and capabilities of using Artificial Neural Network (ANN) surrogate models to approximate the manoeuvres computation of an agile Earth observation satellite. These models could be applied during the optimization process associated to the mission scheduling and also in the AOCS onboard the satellite.
Surrogate models

The potential of optimisation in order to improve engineering design is growing thanks to the capability of doing more and more complex simulations (like CFD, finite elements...). One of the major barriers when optimising is the running time needed by these simulations. The content of this chapter has been largely inspired by [1] and we strongly advice the reader to refer to this document for further information.

The main idea of surrogate models or meta models is to develop fast mathematical approximations to be used instead of computationally expensive simulations. Thanks to these approximations, new questions can be considered, new optimisation methods can be developed and new conditions can be explored faster. At the end we can come back to long simulations to test these new ideas developed. We can then describe the working principle of a surrogate model as a curve fit with the available data. Theory is based under the assumption that once created, the surrogate will be several orders of magnitude faster than the simulation code and at the same time sufficiently accurate when predicting data points that may be far from the already known data.

Although the base idea of surrogate models seems intuitive or simple, some questions arise. Which points do we have to take into account to construct approximations? Which approximation method is better to be used? How do we have to use this new approximations?

2.1 Input data sampling

As we said, the mission of a surrogate model, noted \( \hat{f} \), is to emulate the response of a black box \( f \). In this case, \( f(\mathbf{x}) \) is a continuous function defined by a \( k \)-vector of design variables , \( \mathbf{x} \in D \subseteq \mathbb{R}^k \). We refer to \( D \) as the design space. Further than continuity, the only information we can get from \( f \) is through samples or discrete observations \( \{ \mathbf{x}^{(i)} \rightarrow y^{(i)} = f(\mathbf{x}^{(i)}) \}_{i=1,...,n} \).
The main task will be to use available samples to build an approximation \( \hat{f} \) that allows us to estimate with enough accuracy for any design \( \mathbf{x} \in D \).
Most of the mathematical formulations used to define the models are posed without taking into account how sampling $X = \{ X^{(1)}, X^{(2)}, \ldots, X^{(n)} \}$ affects the corresponding observations spatial distribution. However, a model being well formulated does not mean that it has a good generalisation capability, that is the capability of predicting unknown data, and this quality depends on how sampling $X$ is done.

Sometimes we will not have the choice in this step as data might be somehow limited. In the case of our study, this will be a key factor to produce a model with good generalisation capabilities as we will have as many available data as necessary.

A first important point is to realize that the higher the number of design variables involved in the modelling problem, the more samples from the objective function will be needed to build a sufficiently accurate predictor. For example, if we have a certain accuracy level when predicting with a one dimension variable space and $n$ samples, to achieve the same sampling density with a $k-$dimensions space, $n^k$ observations will be needed [1].

We can come to the conclusion that the number of design variables has a great impact in the necessary amount of samples. It is then worth trying to minimise the number of design variables $x_1, x_2, \ldots, x_3$ before modelling the objective function. This process is called screening and is based on the comparison of the influence of each design variable over the objective function. Most important variables are preferable to be kept. There is much literature on this subject [6], and one of the most important screening algorithms was written by Morris [23], and is specific for deterministic objective functions.

A shared property of most approximations is that they are more accurate around the points where the objective function was evaluated. We can intuitively say that to have a uniformly accurate model throughout the design space, uniformly spread sample points are needed. A sampling plan with this property is called space-filling. There are various techniques to obtain a space-filling sample plan, like stratification, Latin squares or Latin hypercubes. The study of this techniques is out of this review’s scope.

### 2.2 Modelling process

The creation of a surrogate model can be divided in three main parts:
Prepare data and choose the modelling method

Firstly, variables that influence $f$ must be identified and the design variable vector $\mathbf{x} = [x_1, x_2, ..., x_k]^T$ must be reduced at maximum. Just as explained in the previous section. Secondly, $n$ of these $k$-vectors that represent the design space must be obtained. It is important to emphasize that it would be a good idea to scale the entry vectors $\mathbf{x}$ into a unit cube $[0, 1]^k$, this step will avoid having scaling problems afterwards.

Next stage on the process would be the learning, which is defined with data pairs $\{(\mathbf{x}^{(1)}, y^{(1)}), (\mathbf{x}^{(2)}, y^{(2)}), ..., (\mathbf{x}^{(n)}, y^{(n)})\}$. This kind of learning is called supervised learning and searches into the space of possible functions $\hat{f}$ that can replicate the observations of $f$. This space is infinite and almost all of the possible functions do not generalise well.

The method to follow would be to write a structure $\hat{f}$ and search into the space of its parameters to tune the observations approximation. We can take a simple example with the following model structure:

$$\hat{f}(\mathbf{x}, \mathbf{w}) = \mathbf{w}^T \mathbf{x} + v$$

The fitting process will be to find the parameters $(\mathbf{w}, v)$ for which $\mathbf{w}^T \mathbf{x} + v$ better fits the data.

One risk of supervised learning process is overfitting. Overfitting occurs when the model is too flexible, usually because of the use of too many parameters in its structure, and fits too precisely to given data. If the provided data is noisy, an overfitted model will learn the noise over the real behaviour of the system.

Choosing the modelling technique is important, this means its structure and the parameters that define it. This choice also depends on the nature of the function to be approximated. If for example we want to estimate the fatigue of a deformed elastic solid, a linear approximation would be enough. On the other side, if we use a quadratic approximation with a function $f$ that has various peaks, we will never be capable of estimating the value of the parameters that would give a good result and this is just because of a bad choice of the model structure.

Model parameter estimation and training

Starting from a generic structure $\hat{f} = (\mathbf{x}, \mathbf{w})$, the model is defined with the parameters $\mathbf{w}$. The actual problem is about estimating $\mathbf{w}$ so the model fits the data. There are various estimation criteria like the maximum likelihood estimation or the Cross-Validation. We will show the key points about maximum likelihood estimation.
We can compute the probability that data result from the estimation \( \hat{f} \). This data is represented by many pairs of samples \( x^i \) and their predictions \( y^i \pm \epsilon \), where \( \epsilon \) is the estimation error. So we would have:

\[
\{(x^{(1)}, y^{(1)} \pm \epsilon), ..., (x^{(n)}, y^{(n)} \pm \epsilon)\}.
\]

If we assume independent randomly distributed errors \( \epsilon \) as a normal distribution with standard deviation \( \sigma \):

\[
P = \frac{1}{(2\pi\sigma^2)^{\frac{n}{2}}} \prod_{i=1}^{n} \left\{ \exp \left[ -\frac{1}{2} \left( \frac{y^{(i)} - \hat{f}(x, w)}{\sigma} \right)^2 \right] \right\}
\]

with its inverse we obtain the likelihood of parameters from given data. We have to maximise this probability, or what is the same, minimise the negative of its natural logarithm:

\[
\min_w \sum_{i=1}^{n} \left[ \frac{(y^{(i)} - \hat{f}(x, w))^2}{2\sigma^2} - n \ln \epsilon \right]
\]

which will lead to the well known least squares criterion if we take constant \( \sigma \) and \( \epsilon \):

\[
\min_w \sum_{i=1}^{n} \left[ (y^{(i)} - \hat{f}(x, w))^2 \right]
\]

With this example we can understand the basis of most of the training methods which is the estimation of the model parameters that minimise a given training error.

**Testing the model**

If observation data is complete, it is recommended to keep apart 25% of the data to test the model. These observations can not be used during the previous steps because they are intended to evaluate the test error (difference between real function and approximation values) once the model is build. To test the model we can use de Root Mean Square Error (RMSE):

\[
RMSE = \sqrt{\frac{\sum_{i=0}^{n_t} (y^{(i)} - \hat{y}^{(i)})^2}{n_t}}
\]

where \( n_t \) is the size of the test data. We want to have a RMSE as small as possible, but it will be limited by the errors of computation of the objective function \( f \).

### 2.3 Different surrogate models structures

In this section some specific surrogate model types are shown:
2. SURROGATE MODELS

Polynomial models
This models have the following structure:

\[
\hat{f}(x, m, \mathbf{w}) = w_0 + w_1x + w_2x^2 + ... + w_mx^m = \sum_{i=0}^{m} w_i x^i
\]

An order \( m \) polynomial approximation of a function \( f \) is essentially an expansion of Taylor series of \( f \) truncated after \( m + 1 \) \[1\]. Then, for higher value of \( m \), approximation will be more accurate but at the same time the model will be more flexible and we will have higher risk of overfitting the noise or lose generalisation ability. This kind of model is unsuitable for non-linear multidimensional landscapes, which will be our case.

Radial basis function models
This models have the following structure:

\[
\hat{f}(\mathbf{x}) = w^T \psi = \sum_{i=1}^{n_c} w_i \psi(||\mathbf{x} - \mathbf{c}^{(i)}||)
\]

where \( \mathbf{c}^{(i)} \) is the \( i_{th} \) of the \( n_c \) centres of the basis function and \( \psi \) is the \( n_c \)-vector that contains the values of the basis functions \( \psi \). This expression is similar to that of neural network with a single layer and radial coordinates neurons. There are various types of basis functions like linear, cubic, spline, Gaussian, multi-quadratic and more. Under some assumptions they are shown to be universal approximations. Their flexibility is easy to control and they are quite easy to implement \[1\].

Kriging
This is a particular case of radial basis function where the function is:

\[
\psi^{(i)} = \exp\left(-\sum_{j=1}^{k} \theta_j |x_j^{(i)} - x_j|\right)
\]

One of the main characteristics of this function is that the vector \( \theta \) allows to vary the base function’s width for each variable.

Support vector regression
This surrogate model structure allows to specify or compute a margin \( \varepsilon \) within which we accept errors of sampling data without affecting the prediction \( \hat{f} \) of the SVR.
Artificial neural networks

ANN and more specifically multilayer feed-forward networks with as few as one hidden layer, provided with sufficiently hidden units, are a class of universal approximators \[32\]. This structure of surrogate model will be more deeply analysed in Chapter 3.

2.4 Discontinuous objective functions

To complete the state-of-the-art on surrogate models, some lately researches must be mentioned, like \[8\], where the problem of discontinuity of the objective function is treated. Sometimes a simple model is not enough to approximate a complicated function, specially when it presents different behaviours in different input space regions.

In fact, meta models usually assume a smooth objective function. For this reason they present important variations near discontinuities that reduces the model’s generalisation ability. One way of preventing this variation would be to divide the input space in smaller regions without discontinuities and build a model for each one of these regions. Although the idea of combining various surrogate models seems appropriate to approximate functions, there is no evidence that shows that combining them is better than choosing the best model of all of them.

This subject is deeply investigated in \[8\] where the authors present an automatic method to combine various local surrogate models to obtain a smooth and accurate approximator for discontinuous functions. The method is based on the Expectation-Maximization algorithm for Gaussian mixture modes. An improvement in the approximation accuracy is shown when applied to an engineering optimisation case.

2.5 Summary

In this chapter we have shown the concept of surrogate model and its mission in nowadays engineering design. We explained most important points to take into account to build a model. Starting with the design variables selection that will be used to define the objective function and the impact of their number over the amount of needed observations to build a sufficiently accurate model. About the observations chosen to create the model, we said they must be space-filling so that they let the model have a good generalisation capability. Once data is prepared, the model’s structure must be chosen, and it must be adapted to the complexity of the objective function to avoid being too simple or too flexible and overfit training data, which
2. SURROGATE MODELS

will make the model learn the possible noise in data samples and lose the
generalisation skill. We also showed the principles of supervised learning
and the importance and key factors about testing the model.
Artificial Neural Networks

In Chapter 2 we showed the main characteristics of a surrogate model and most important aspects to consider while building one. In this chapter we explain how can ANN be used to build a surrogate model. Along this section we will talk about some previously defined concepts like generalisation or supervised learning but directly related to the ANN case. We also describe the internal structure and most of the elements that form Feedforward Neural Network (FNN) as they are relevant for the understanding of their implementation.

The study of ANN was originated with the attempts of understanding and building mathematical models for neurobiology. Further than their origin, nowadays FNN provide the engineering community with a methodology to build non-linear models that accept a large number of inputs. Because of the learning algorithms that train networks from example of the modelled problem, neural networks can be used for an application knowing relatively few a priori information about the real process. Obviously, some information about the nature of the problem should be known, and this is going to be reflected in the design process where the input variables and the network architecture must be defined.

The notation chosen for neural network representation and most of the figures presented in this chapter are taken from [21], mainly because it is written by the developers of the Neural Network Toolbox™ software that we will use to implement our surrogate models.

3.1 Why Artificial Neural Networks?

There are various ways of implementing a surrogate model, but we decided to use ANN and more precisely FNN, a kind of neural network without feedback connections. FNN is the common choice when approximating non-linear parameters, specially after some studies that demonstrated their approximation capabilities. In [33] the authors demonstrate that FNN with just one hidden layer and a sigmoid transfer function in its neurons are capable of approximating any arbitrary continuous function if it has enough
hidden units available. In addition, [35] shows that to prove the capability of a neural network in the \( n \) dimension case, only the 1 dimension case needs to be proved.

Because of their simple implementation and the good results obtained, even in multi-dimension cases, FNN are a suitable solution for our study case. Although the good results obtained by simply applying the “brute force” for solving a problem, if we add some knowledge about the problem and about the neural networks fundamentals while designing the network, the effectiveness and efficiency of the solution could increment. This fact is shown in [22] through three different examples.

Another important point about FNN is the nature of their structure. As they are formed by connections of different nodes mostly in parallel, they can be computed nearly simultaneously. The only possible delay would be introduced by the propagation from one layer to another. However, most of the neural networks implemented have two or three layer. From this point of view, FNN could be directly implemented in hardware and be executed at high speed thanks to their massively parallel architecture. This parallel architecture is also interesting when they must be implemented in software because most of actual operative systems and programming languages allow the execution of precesses in parallel by the use of multiple threads.

### 3.2 Neuron model and neural network

**Neuron model**

The following neuron model was inspired by the real neurons behaviour. A single exit \( a \) is related to multiple inputs \([p_1, p_2, ..., p_R] \), which will be noted as the vector \( \mathbf{p} \) and to the weights \( \mathbf{w} = [w_1, w_2, ..., w_R] \) associated to the input connections. Another input, 1, is multiplied by the bias \( b \) and summed with the weights and inputs products \( \mathbf{wp} \). The summation output is a scalar, noted \( n \), and will go into the transfer function \( f \), which produces the scalar \( a \).

\[
a = f\left(\sum_{i=1}^{R} w_i p_i - b\right) = f(\mathbf{wp} - b) = f(\mathbf{Wp} - b) \quad (3.1)
\]

where \( \mathbf{W} \) is a matrix with a single row when a single neuron is represented, like in Fig. (3.1). The rows of the matrix identify the destination neuron of the weight, and the columns identify the input signal that will be affected by that weight. In Fig. (3.2), we can see the representation of a multiple input neuron (3.2(a)) and its abbreviated version (3.2(b)).
The transfer function can be a linear or non-linear function. A particular function must be chosen in order to satisfy the problem specifications that is being resolved by the neuron. For example, hard-limit transfer functions are useful for classification problems as it can only get two possible values:

$$ f(n) = \begin{cases} 
0 & \text{for } n < 0 \\
1 & \text{for } n \geq 0 
\end{cases} $$

Another kind of transfer function, and more suitable for our case, are non-linear sigmoid functions. These are increasing, continuously differentiable functions in the range \([0, 1]\) or \([-1, 1]\). An example would be the log-sigmoid:

$$ \sigma(n) = \frac{1}{1 + e^{-n}} $$

(3.2)

Thanks to these properties, sigmoid transfer functions are used in multilayer networks that are trained with the backpropagation algorithm, specially because of their differentiability.
Network architecture

Usually, one single neuron is not enough. We might need several ones working in parallel, forming a layer as shown in Fig. (3.3). In this case, each of the $R$ inputs is connected to every neuron, so the weights matrix $W$ will have $S$ rows, one for each neuron, and $R$ columns, one for each input. Each neuron has its own bias $b_i$, transfer function $f$ and output $a_i$ that. The output of the layer is the vector of outputs $a$. In the proposed abbreviated notation, the $f$ vector contains the transfer functions of each neuron, as they are not necessary the same for all neurons.

Finally we can connect various consecutive layers of neurons and each layer will have its own weights matrix $W^i$ and its own bias vector $b_i$. The superscript is used to identify the layer. Each layer can have a different number of neurons. We can see in Fig. (3.4) the notation of a three-layer network.

Multilayer networks are more powerful than single layers. In fact, a network with two layers, the first one with sigmoid functions and the second one with linear functions, can approximate most continuous functions.
3. ARTIFICIAL NEURAL NETWORKS

We can see that the number of choices to do when building a network is huge. However there are some parameters easy to determine. The number of input and output variables is defined by the external problem, so the number of neurons in the output layer will be established by the number of output variables. Finally, the characteristics of the output signals will also determine the transfer functions of the last layer neurons.

We have only presented FNN but there are other kinds of ANN with feedback connections or delays. These other ones are more suitable for applications where a certain dynamic is required, which is not our case.

3.3 Supervised learning

The learning method that we will use to train our networks will be supervised learning. This method infers a function from some training data. This training data is a collection of input and output example pairs. The inferred function should predict the right output values for any input value. The learning algorithm must be able to generalise from training data to unseen data.

The mission of the learning algorithm will be to relate the prediction error of the training data to the network parameters. There are four kinds of optimisation algorithms that make possible to minimise this error. Steepest descent, conjugate gradients and quasi-Newton are three of them and they can be understood in the context of minimising a quadratic error function. The fourth method is the Levenberg-Marquardt algorithm.

All of these methods require an efficient and repeated gradient computation, and backpropagation is the most commonly used method to do so. Backpropagation is essentially an optimisation technique based in the gradient computation. The mean square error between predicted data and the target for an input is propagated backwards from the output layer to the first one. Thanks to this fact, connections between neurons can be tuned in order to minimise the error. In Appendix B we included a detailed description of the backpropagation algorithm using the previously defined notation.

Backpropagation meant a huge revolution for the neural networks evolution. However, the basic algorithm is too slow for most of the practical applications. For this reason, some variations have appeared to speed-up the algorithm. To train our networks we use a variation of backpropagation, Levenberg-Marquardt, based in numerical optimisation techniques. For more details on this subject [21] and [10].

Preliminary study 23
In some cases, even if these variations are used, the training time can be really slow, specially for big networks. Recent study [20] shows how parallel ANN should drastically reduce the training time.

### 3.4 Architecture selection

Although backpropagation finds the structure parameters that minimise the mean square error, a good approximation of the objective function is not assured. This is mainly because the network’s capacity is inherent limited by the number of layers and neurons.

The architecture is in first place defined by each node’s transfer function. Then the problem is about choosing de architecture that will result in the minimum generalisation error knowing that it will not be the same architecture that gives the minimum training error. The training error is the distance between the predicted results and the targets specified in the training dataset. On the other hand, the generalisation error is the same distance but with unseen data. The difficulty is in the fact that we know the training error but not the generalisation error.

There have been various research and publications that deal with this problematic like [17], [3] and [11] or in the book [10]. From all of them we can summarize two ways of approaching the problem.

A first category of methods is characterised by starting with oversized networks that gradually remove unnecessary nodes or connection. They are known as pruning methods. There are two different ways of applying this method. One is based on an error function modification, and the second is based on a sensibility measure. In [4] it is shown a redundant node elimination method in order to reduce an oversized network.

The second category starts with a very simple network that will be grown. They are known as constructive methods. In [29], a Modified Cascade Correlation algorithm is developed in order to build a constructive network.

We could cite some advantages of constructive methods over pruning ones:

1. In constructive algorithms the size of the network is automatically determined. Whereas in pruning algorithms initial size must be determined.

2. Constructive algorithms are computationally cheaper.

3. Constructive algorithms are more likely to find a smaller network.
In [29], they are also investigated the benefits over generalisation when creating a committee of networks with all the networks that have been generated while searching for the best one.

3.5 Other important aspects

In addition to the structure in terms of number of layers and neurons, there are other elements that influence the generalisation ability of the network.

A first important element that will influence the final results is the input data, and more precisely its dimension. In [28], the author defines the properties that must be attended from an analysis of the input dataset. In [14], the treatment of multi-dimensional data is shown to improve the estimation capabilities of neural networks.

Another method for reducing the overfitting problem is developed in [34], where the input and output multi-dimension spaces are divided in various smaller curve surface spaces. By curve fitting these subspaces, new samples are created. These new samples can be used to train the neural network and help to reduce the overfitting effect.

Another important factor that influence the capability of the network is the output neurons transfer function. It is concluded in [16] that output functions strongly depends on the objective function in order to obtain good results when trying to approximate it.

3.6 Applications

ANN are widely used in the space domain. An example could be [19], where the author evaluates the possibility of using ANN as an abort detection technique for the landing phase of the Mars Science Laboratory in the case of a dust storm. Another example is the [35] application, where a dynamic neural network is designed for fault detection of reaction wheels in a satellite.

We are interested for a more specific application of neural networks in the space domain, we are interested in satellite attitude and manoeuvres applications. ANN have been widely applied for satellite attitude control. In [2] and [24], adaptive multilayer neural networks are used to build an optimal neuro-controller that is able of tracking an objective quaternion. In [2], another neuro-controller is created to stabilise the satellite’s attitude with magnetic actuators. This controller is created with dynamic neural units. Another example of application is shown in [30]. In this case, authors
create an evolving neural network trained with a genetic algorithm that controls the satellite’s attitude.

All precedent publications show attitude control systems based on dynamic or evolving networks. For the case that we will treat, we need to create a static network that do not change its behaviour during the satellite’s mission.

Some other important publications are [18], [27] and [26]. These articles show a new structure of multilayer neural network that is developed in the hypercomplex quaternion algebra. Their particularity is that weights and biases are quaternions instead of scalars, and quaternion product is used in the place of conventional product. It is also presented a learning algorithm for the proposed Quaternion multilayer perceptrons (QMLP) based in backpropagation. QMLP allow to approximate quaternion-valued functions more efficiently than real-valued multilayer neural networks.

This new type of networks would be interesting for applications involving satellite attitude operations. This is the case presented in [9] where authors present an algorithm that allows to estimate the satellite attitude through a QMLP. They demonstrate that this new method is as accurate as Extended Kalman Filters that are used nowadays, but with a reduced computational complexity.

All of these applications show that ANN are suitable for space applications that involve attitude operations like controllers or attitude estimators. Now we face the agile Earth observation context with a well built knowledge about surrogate models and ANN. Our work will be to imagine of possible problems and applications into which a surrogate model can be used in order to reduce the computational cost. We will apply the techniques reviewed in order to prove if neural networks are a valid solution for these problems.
Implementation

In this chapter we will show our vision on how surrogate models can be used in the space industry, and more precisely in the agile satellites manoeuvre computation process that is present in various Earth observation mission phases.

We first analyse the nature of the data that we have and discuss about the best choice on the design variables \( x \) that will define our final objective function \( f \). As mentioned in Chapters 2 and 3 this is a very important step which will have a great impact in our surrogates performance. Some words about the FNN used and the employed methodology are also included.

Two different applications are presented, the first one is intended to be applied in the scheduling problem of an Earth observation mission, and the second would be suitable to perform onboard manoeuvres computation without being computationally expensive.

As we already said in the introduction, this internship will continue for some more than two months, so these applications and their results are not the final solutions, and will surely continue evolving.

4.1 Attitude manoeuvres and surrogate models

Our efforts will be focused on finding possible applications of ANN surrogate models in the domain of an Earth observation mission and on the building and testing of those models. To be more precise, we will focus on applications that involve attitude manoeuvre computation. This is not a very time consuming computation when just one manoeuvre is being treated, but it becomes an important factor when iterative algorithms need this information and it has to be computed thousands of times. This would be the case of the optimisation process involved in the scheduling problem. Another situation where the execution time would be critical is onboard of the satellite, where computational budget might be limited.

Before talking about the specific applications that we will work in, we must describe some aspects about the physics involved in our model.
We will treat manoeuvres of a satellite considered as a rigid body. We consider three actuators, reactions wheels, and the axis of rotation of each one of them is aligned with the axis of the satellite’s body frame. The satellite describes a circular orbit of 800 km of altitude and an inclination angle of $98^\circ$.

The computation of the manoeuvre is done by a numerical integration of the differential kinematics equations shown in Appendix A. For more details about manoeuvres, two complete reference would be [36] and [31].

Now we can go through the first steps to build our models. First choices would be about the variables that are involved in the process, and also discuss about the sampling plan to obtain the observations that will be used to train our models. In a second step, the model structure must be defined and trained by supervised learning. Finally, the testing method must be defined too.

### 4.1.1 Key variables to model attitude manoeuvres

In this first analysis we will identify the most important variables involved in the manoeuvring process. However, it could happen that not all of them are used when we create a surrogate (i.e. if the application where we want to implement our model does not have all the measurements available) or that some other new variables appear because they give some important information, depending on the concrete problem being treated by our model.

We will list all the variables and parameters that are involved in the manoeuvre process:

- $h_{tot}$, total angular momentum of the satellite;
- $q$, quaternion that describes the satellite’s attitude;
- $h_b$, the maximum angular momentum given by reaction wheels considering bang-bang manoeuvres;
- $w$, the angular velocity of the satellite’s body frame with respect to the inertial reference frame;
- $T_{man}$, the time duration of the manoeuvre.

This variables define the state of the satellite at any moment. For the rest of the document we will use subindices to distinguish the starting and ending value of these variables. $i$ would mean the initial value, before starting the manoeuvre, and $f$ would mean the final value, at the end of the manoeuvre. We will develop the main characteristics of each of these variables.
Total angular momentum

\( h_{\text{tot}} \) is an important variable as it has a direct impact on the manoeuvre duration (i.e. for a given manoeuvre, \( h_{\text{tot}} \) can make a wheel saturate earlier or later, making the manoeuvre last more or less in order to do the same rotation). We will consider it as constant during the whole manoeuvre as the satellite is isolated. From this point of view, it can be seen as an initial condition of the manoeuvre. In a real application, \( h_{\text{tot}} \) would slowly change with external disturbances affecting the satellite. \( h_{\text{tot}} \subset \mathbb{R}^3 \) and \( h_{\text{tot}}^{(i)} \in (-\frac{1}{4}h_{\text{max}}, \frac{1}{4}h_{\text{max}}) \) | \( i = 1, 2, 3 \), where \( h_{\text{max}} = 10[Nm.s] \).

Quaternions

\( q_i \) and \( q_f \) define the attitude of the satellite, they are one of the most important variables. We are interested in manoeuvres changing satellite’s attitude from \( q_i \) to \( q_f \), but we can use quaternions properties to describe the attitude change driven by the manoeuvre with one only quaternion defined as the product of these two. With this we will reduce the input space of our models. \( q \subset \mathbb{R}^4 \) and \( q^{(i)} \in (-1, -1) \) | \( i = 1, ..., 4 \)

Angular velocity

In a rest-to-rest manoeuvre, \( \mathbf{w}_i \) and \( \mathbf{w}_f \) are zero, so these variables do not give relevant information for our model. On the other hand, for some kind of manoeuvres typically done by agile Earth observation satellites, like yaw-steering, kinematic angular velocity will be crucial to ensure that the image is taken with the right conditions. \( \mathbf{w} \subset \mathbb{R}^3 \) and \( \mathbf{w}^{(i)} \in (-\frac{h_{\text{max}}}{I_{\text{sat}}}, \frac{h_{\text{max}}}{I_{\text{sat}}}) \) | \( i = 1, 2, 3 \) where \( I_{\text{sat}} \) is the satellite’s inertia matrix.

Bang-bang manoeuvres

Manoeuvre duration The time \( T_{\text{man}} \) that it takes to achieve the attitude manoeuvre to be completed is very important. It is a key factor in Earth observation missions, and specially for agile satellites because it is this that defines the agility of the satellite. The faster the satellite can make a given manoeuvre, the more efficient it would be for the mission. \( T_{\text{man}} \subset \mathbb{R}^1 \) and \( T_{\text{man}} \in (2, 60) \).

Wheel’s momentum profile We will work with bang-bang manoeuvres, where each of the 3 wheels has the angular momentum profile shown in Fig. {4.1(b)}. We will consider that all wheels accelerate and decelerate at maximum torque, \( C_{\text{max}} \), and they will be saturated at \( h_{\text{max}} \). We can see in Fig. {4.1}, the relation between acceleration, velocity and position profiles of a bang-bang manoeuvre. In order to define the manoeuvre we will use \( h_{b} \) from the velocity profile, as shown in Fig. {4.1(b)}. As there are
three wheels that are aligned with the satellite’s body frame, \( h_b \subset \mathbb{R}^3 \) and \( h_b^{(i)} \in (-h_{\text{max}}, h_{\text{max}}) \mid i = 1, 2, 3 \)

### 4.1.2 Data generation

We pointed out in Chapter 2 that sometimes, when we try to implement a surrogate model and train it with supervised learning, the number of observations available is limited. Actually, this is not our case because we can produce as many observations as wanted.

With the algorithm used to compute manoeuvres, we need to define \( q_i, h_{\text{tot}}, w_i, T_{\text{man}} \) and \( h_b \). In fact, \( q_i \) is not relevant as we will force it to be the unit quaternion without affecting the manoeuvre. About the rest of variables, they are randomly generated but always in the previously defined possible values.

Once initial variables are defined, a numeric integration is done to obtain the final attitude \( q_f \). By generating the manoeuvres this way, we make sure that the randomly generated data will be spread through the whole space of possible values. However, this doesn’t mean that the input or output space are homogeneously distributed through their space. This will depend on which variables are chosen to define the problem. Input and output samples should be analysed to make sure that our sampling plan was really space-filling in input and output spaces.

### 4.1.3 Feedforward neural network

We chose this model structure because the supervised learning method employed is easy to implement. Moreover, their capability to work with multidimensional input and output spaces makes FNN really flexible to solve a huge variety of problems. In our case, this flexibility will be really useful as
we will add or remove variables from the input and output easily, depending on the application done.

**Transfer function**

We will always work with various layers. The output layer will always have linear neurons as they are the most appropriate to solve approximation problems, like our case. Output variables of our problems are expected to have an infinity of possible values.

For the rest of neurons in hidden layers, we will apply a sigmoid type transfer function. A typical sigmoid function would be:

\[
\sigma(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}
\]

We have decided to create our own sigmoid function that will be less computationally expensive:

\[
\sigma(x) = \frac{x}{1 + |x|}
\]

We can see in Fig. 4.2 a comparison of our sigmoid function with one typically used in FNN architectures for function approximation.

It is important to emphasize that our function is differentiable, without singularities, an important property to apply some learning algorithms like backpropagation. We have changed the transfer function in order to make our models less time consuming when computing an output. In fact this will reduce both the training and execution time of the network, and execution time might be specially important for applications where the computational capabilities of the processor are limited.
Network architecture

As we already said, we will be using FNN with an output layer and one or more hidden layers. Network’s architecture will be defined by the number of hidden layers and the number of neurons in each of these layers.

The method used to choose the best architecture is not computationally optimal, but we obtained good results. It is a constructive algorithm that starts with a simple network and progressively adds neurons in the hidden layers. In order to choose the best architecture, the algorithm do not use the training error but an estimation of the generalization error. To compute this generalization error we use a completely different test dataset. The performance index that is used for this error is shown in the next section.

We can see in Fig. (4.3) the evolution of the error of a trained network when we increment its size. In Fig. (4.3(a)) the curve shows the evolution of the error in a single hidden layer network, when the number of neurons goes from 1 to 100. As we can see, the error decreases very fast for the first neurons but after 20 of them it decreases very slowly. In Fig. (4.3(b)) we can see the same evolution but in the case of two hidden layers, so an error surface is shown. From these two examples we can see that there is a network size from which it is not worth to continue growing because the performance gain obtained would not compensate the computational cost increment for the network training. We confirm with this observations what we saw in the literature, where it was recommended to keep the smallest network that validated the required performance.

Training the model  To train the model we will use the Levenberg-Marquardt algorithm, a much faster variation of backpropagation. We will use manoeuvres files with (input,output) pairs to train the network. We did various trials with files containing different amounts of manoeuvres in
order to see the influence of this parameter over the final performance of
the model. We noticed that over 3000 manoeuvres, the performance did
not increase significantly but the training time kept on increasing. Taking
into account the performance and training time cost trade-off, we decided to
use 3000 manoeuvres files. Moreover, from those 3000 manoeuvres, 33% is
used for training, 33% for validation and 33% for testing the generalisation
capability of the network.

4.1.4 Performance computation

After the training is done, the performance of the network is computed with
the testing dataset. This performance index is the mean square error index:

\[ mse = \frac{1}{N} \sum_{k=1}^{N} (t(k) - a(k))^2 \]

where \( N \) is the number of test samples used, \( t \) is the vector of real process
outputs and \( a \) is the vector of outputs estimated by our model.

This index is useful to determine the network’s approximation accuracy
and its generalisation performance from the point of view of a data fitting
problem. However, we will profit our knowledge in the modelled process to
define a new performance index which will be more related with the global
problem needed precision.

As we are computing manoeuvres, we will always be capable of getting
a final estimated attitude, \( \hat{q}_f \), from the data coming out of the model. We
can then define the new performance index as follows:

\[ \delta q = \frac{1}{N} \sum_{i=1}^{N} \left| \frac{\delta \theta_i}{\theta_i} \right| \]  \hspace{1cm} (4.1)

where \( \delta \theta_i \) is the angle error between \( q_f \) and \( \hat{q}_f \) and \( N \) is the number of
testing points. This error is defined as the rotation angle associated to the
quaternion \( q \) that would be necessary to rotate from \( q_f \) to \( \hat{q}_f \). In an ideal case
they would be identical and the rotation would be zero. The \( q \) quaternion
is defined as:

\[ q = q_f \hat{q}_f = \cos \left( \frac{\delta \theta_i}{2} \right) + v \sin \left( \frac{\delta \theta_i}{2} \right) \]

So the rotation angle \( \delta \theta_i \) can be computed as follows:

\[ \delta \theta_i = 2 \text{arctan} \left( \frac{|v|}{|q_0|} \right) \]
In some of the applications that we have developed, the estimated variable is the manoeuvre time, $T_{\text{man}}$. For these cases, the performance index used will be much simple:

$$\delta T_{\text{man}} = \frac{1}{N} \sum_{i=1}^{N} |T_{\text{man}}^{(i)} - \hat{T}_{\text{man}}^{(i)}|$$ (4.2)

Once the indexes that will be used are defined, we need to define the specifications that will determine if our models are accurate enough or not. In the case of the time estimation we will distinguish two specifications. The first one, and more restrictive, is given by the onboard AOCS precision that works at 8Hz, so $\Delta T_{\text{AOCS}} = 125\,\text{ms}$. If $\delta T$ is smaller than the onboard $\Delta T_{\text{SCAO}}$, it will not influence to the AOCS performance. In the case that this specification is not respected, a softer second one can be defined, relative to the real manoeuvre time. We will accept an approximation error of 1% of the real $T_{\text{man}}$. We will write these specifications as follows:

$$\delta T_{\text{man}} < \Delta T_{\text{AOCS}} = 125\,\text{ms}$$ (4.3)
$$\delta T_{\text{man}} < \frac{T_{\text{man}}}{100}$$ (4.4)

For the applications where we can obtain an estimation of the final attitude $\hat{q}_f$, we need to take a look to how the AOCS is implemented in order to determine the performance specification. In Fig. (4.4), a simple representation of the onboard control law is shown.

$C_{ff}$ is the feedforward command used to control the dynamics of the system in order to achieve the desired final attitude $q_c$. This $C_{ff}$ torque is computed with an estimated system dynamics that has an uncertainty compared to the real system dynamics. Because of this uncertainty, the real output attitude of the satellite, $q_r$, is not equal to $q_c$. This difference results in an error quaternion, $q_e$, which will be used by the controller to cancel it. In order to avoid the approximation error to affect the AOCS performance, we need to respect the following specification:

$$\delta q < 0.1\%$$ (4.5)

Another important point of the performance evaluation is the manoeuvres file used to compute this index. If we use the same file used to train the network, the 33% of the manoeuvres would have already been used during the learning process, so the performance in this exact points will be much higher than unknown points. This would not really be representative of the generalisation capability of our model. To avoid this problem we will always
4. IMPLEMENTATION

Figure 4.4: Onboard manoeuvre control law

generate two separate files with 3000 manoeuvres each, and the second one will just be used to compute the network’s performance.

4.1.5 Methodology

We will use MATLAB to implement our algorithms and SIMULINK to create the blocks that compute the manoeuvres. To implement the neural networks, and specially the training algorithms, the Neural Network Toolbox of MATLAB has been used. This tool is very complete and flexible at the same time as it allows to implement and integrate in the toolbox any element involved in a neural network as we did when changing the transfer function.

4.2 On the ground

As we previously said, the scheduling problem for agile Earth observation satellites has a big combinatorial complexity. One of the most important points involved in the optimisation process is the computation of the manoeuvre transition time to change the attitude from the actual scene to the next one.

4.2.1 Agility model

The agility model is in charge of deciding which scenes must be taken from all the possible scenes, and the order into which they will be taken. In Fig. 4.5 we show a graph to make this process more understandable. In this example, the agility model scheduled to take images 2, 4 and 5. This images will start to be taken from the satellite’s trajectory at dates $T_2$, $T_4$ and $T_5$ with attitudes $q_2$, $q_4$ and $q_5$ respectively.

In order to schedule the images, the agility model must compute the manoeuvre time to acquire an image when starting from an specific position in orbit. At this point of the mission design, the in orbit position of the satellite is a known parameter. Its initial attitude, $q_i = q_0$ at $T_0$, can then
be easily obtained from the coordinates of the last scene acquired. Initial angular velocity can also be obtained as they will be related to the conditions under which the image was taken.

In fact, the output of the agility model are the dates when the chosen scenes must start to be taken. In order to find this dates, the model uses iterative algorithms. Once these dates are defined, the satellite’s in orbit position can be computed and so does the attitude to point to the following image.

An important point about the agility model is the uncertainty on $h_{tot}$ due to the fact that the scheduling is done more than 24 hours in advance, so the real satellite’s $h_{tot}$ is unknown because it depends on external perturbations of the satellite. Because of this reason, the dates given by the agility model must be an upper bound, to take into account the effects of this uncertainty and ensure that the satellite will always be able to get into the final attitude in the given time.

In a second phase of the mission design, programming, the real manoeuvres are computed from a first estimation of $h_{tot}$. This information is transmitted to the satellite in order to be executed by the AOCS.

Our first application will estimate the minimal manoeuvre time needed to do the transition from an initial attitude to an unknown final attitude that enables the satellite to point to the next image that must be taken. To do so, the ground coordinates of the last image that was taken and of the next image to be taken are needed. Also the initial in orbit position of the satellite is needed to determine its initial attitude.

### 4.2.2 Design variables

We can see in Fig. 4.6 a representation of the variables used to define our problem. $\phi$ and $\psi$ represent respectively the roll and yaw angles needed to point to the images. $\delta_{iop}$ is the distance between the satellite’s in orbit position and the last image, projected over the ground path. Finally, $\text{Dist}$...
is also the projection over the ground path, but in this case we describe the distance between last images and the next one. As we can see, the final attitude is not known as the image can be taken from various in orbit positions. With these 6 variables, the problem’s geometry is defined.

The first objective will be to compute the minimal time needed to arrive to a final attitude that enables the satellite to take the next image. If we would like to generate this manoeuvres directly with this variables, we should implement an optimisation iterative algorithm that searches the closest possible position along the satellite’s trajectory, from which the satellite can point to the second image with an attitude that must be reachable from its initial attitude in the same time that it took to the satellite to get to this position.

Actually we used another method. What we did was to create random manoeuvres without taking into account the images context. We need to create a random bang-bang profile and integrate the kinematics equations to obtain the final attitude. By looking at the initial and final attitude, we filtered those that where out of the visibility limits of the satellite, keeping all manoeuvres that would be suitable for the imaging context. We defined these limits in terms of roll, pitch and yaw angles as following:

\[-30^\circ < \phi < 30^\circ\]
\[-30^\circ < \theta < 30^\circ\]
\[-4^\circ < \psi < 4^\circ\]

These angles can be directly obtained from the quaternions of the manoeuvre and we can obtain the rest of the 6 design variables by using the satellite’s Earth trace.
The manoeuvres computed by our algorithm are created with the maximum torque of the wheels, so we can assume that they correspond to the minimal time manoeuvre to change from \( q_i \) to \( q_f \). Because of this reason, we will not find two different manoeuvres that enable the satellite to take the same final image, starting with the same initial attitude and having different final attitudes. The only one that we will find must be the one done in minimal time.

Input variables will then be:
\[
[\phi_i, \psi_i, \delta_{iop}, \text{Dist}, \phi_f, \psi_f]
\]
As output variables we will have the minimal time of manoeuvre, \( T_{man} \). With this surrogate we can help the agility model in its computations.

In a second application we have created a model where we added \( h_{tot} \) as an input, and \( h_b \) as output. This model is intended to show the approximation capabilities of FNN. This would be also useful in the programming phase of the mission.

We must emphasise that these applications do not estimate the manoeuvre minimal time given a final attitude but they estimate the minimal time to point to an scene on Earth surface, without previously knowing when it will happen.

4.2.3 Data analysis
As we said, we can create as many manoeuvres as desired. On the other hand, the method that we use creates manoeuvres in absolutely random directions. Adding the fact that we have to filter the manoeuvres in order to remove those that point out of the satellites limit, the final distribution of the input variables is not homogeneously distributed through the input space. This can be observed in Fig. (4.7). We can see for example in Fig. (4.7(a)] that the \( \delta_{iop} \) is homogeneously distributed for manoeuvres that last less than 20 seconds. However, for manoeuvres where \( T_{man} \) is much longer, there are almost no samples. This is due to the manoeuvre generation process, in which we filter all the manoeuvres that make the satellite point out of the visibility limits. When the satellite does a long manoeuvre it has a higher probability to finish pointing out of this zone, hence, the manoeuvre will be deleted. In Fig. (4.7(b)] we can see the same situation but observing \( \text{Dist} \). In this case the distribution of \( \text{Dist} \) is bounded by some kind of cone due to the fact that \( \text{Dist} \) is related to the initial and final attitudes which are both filtered. The lack of samples for longer manoeuvres is also visible. The represented line shows the bias introduced by the fact that the satellite is moving forwards.
When we observe Fig. 4.8, we can see distribution of the angular momentum \( h_b \) for different manoeuvres durations. We can see the same behaviour for the three wheels. The cone shows the angular momentum bounds for a given manoeuvre time which is the consequence of the bang-bang manoeuvres profile that we described in Fig. 4.1(b). We can also see that the saturation is only achieved for manoeuvres with \( T_{\text{man}} > 40 \) s in Figs. 4.8(a) and 4.8(b). In Fig. 4.8(c), we can see that there are less manoeuvres that saturate the \( z \) axis which is normal as the angle limitation for the yaw axis is more restrictive. However, the most important information that we can see in these figures is that for longer manoeuvres we obtain barely no samples.

As a consequence of the lack of samples for long manoeuvres, we should have a worst generalisation ability for manoeuvres with \( T_{\text{man}} > 30 \) s. It would be interesting to invest some more time in developing an algorithm that allows us to create minimal time manoeuvres from an initial and final attitude, without having to filter them. This should solve this non-homogeneous distribution of samples and increment the following models performance.

Figure 4.7: Relation between variables for the agility model

(a) \( T_{\text{man}} \) and \( \delta_{\text{tip}} \) relation
(b) \( T_{\text{man}} \) and Dist relation

Figure 4.8: Relation between variables for the agility model when \( h_b \) is estimated too

(a) \( T_{\text{man}} \) and \( h_{b1} \) relation
(b) \( T_{\text{man}} \) and \( h_{b2} \) relation
(c) \( T_{\text{man}} \) and \( h_{b3} \) relation
NEURAL NETWORK SURROGATE MODELLING
FOR EARTH OBSERVATION MISSIONS

4.2.4 Surrogate overview

In this section we will formally present the inputs and outputs of the surrogate models that we have designed. The first model, in Fig. (4.9(a)), will estimate $T_{man}$, and the performance of this model will be calculated as $\delta T_{man}$ defined in Eq. (4.2).

For the second application we created another model, Fig. (4.9(b)). This model includes $h_b$ as output and $h_{tot}$ as input. For this last application, performance will be computed as $\delta q$ defined in Eq. (4.1).

4.2.5 Results

We used our algorithm to find a constructive network that solves the model problem of Fig. (4.9(a)). Four different results are shown in Table (4.1). In order to find these architectures, the constructive algorithm took around two hours in each case, and the training time of a single network with an architecture within the range of these ones is around one minute. The differences between each row of Table (4.1) are the characteristics of the manoeuvres that were used to train the networks. In fact, as the result of training the network with a normal manoeuvre file we obtain a performance of $\delta T_{man} = 0.7103$, the performance is too bad as the specification requires an error of less than $125 ms$. We then wondered if we could somehow classify the manoeuvres so that different networks will be training with each kind of manoeuvre. We then decided to generate manoeuvres by forcing

Table 4.1: Models performance for $T_{man}$ estimation of rest-to-rest manoeuvres

<table>
<thead>
<tr>
<th>Direction</th>
<th>$\delta T_{man}$ [s]</th>
<th>Architecture</th>
</tr>
</thead>
<tbody>
<tr>
<td>isotropic</td>
<td>0.7103</td>
<td>{14, 19}</td>
</tr>
<tr>
<td>$\vec{x}$ saturated</td>
<td>0.0053</td>
<td>{18, 19}</td>
</tr>
<tr>
<td>$\vec{y}$ saturated</td>
<td>0.0051</td>
<td>{6, 20}</td>
</tr>
<tr>
<td>$\vec{z}$ saturated</td>
<td>0.0055</td>
<td>{14, 9}</td>
</tr>
</tbody>
</table>
4. IMPLEMENTATION

<table>
<thead>
<tr>
<th>Direction</th>
<th>$\delta q$</th>
<th>Architecture</th>
</tr>
</thead>
<tbody>
<tr>
<td>isotropic</td>
<td>0.0483</td>
<td>${19, 15}$</td>
</tr>
<tr>
<td>$\vec{x}$ saturated</td>
<td>0.0037</td>
<td>${20, 17}$</td>
</tr>
<tr>
<td>$\vec{y}$ saturated</td>
<td>0.0041</td>
<td>${17, 20}$</td>
</tr>
<tr>
<td>$\vec{z}$ saturated</td>
<td>0.0044</td>
<td>${18, 10}$</td>
</tr>
</tbody>
</table>

Table 4.2: Models performance for $T_{man}$ and $h_b$ estimation of rest-to-rest manoeuvres

the saturation of the wheel related to a specific axis of the satellite’s body frame. When we indicate $\vec{x}$ saturated, in the second row, we mean that the manoeuvres used to train that network had always the wheel in the $\vec{x}$ axis saturated. When these models must be used in the real application, the three of them must estimate $T_{man}$ and we should keep the smaller result as it would correspond to the network that was trained with that kind of manoeuvre. The third column describes the network architecture, where each element defines the number of neurons in the hidden layer. All these networks have two hidden layers, and all of them have a third layer, the output layer, which will have a single output corresponding to $\hat{T}_{man}$.

All three final networks achieve a performance around $5 \cdot 10^{-3}$ which is enough for the desired specification.

We evaluate now the results obtained for the case of the model in Fig. (4.9(b)). Results are shown in Table (4.2). In this case we must compare the resulting error to the Eq. (4.5) specification. As happened in the previous case, if we use manoeuvres with an arbitrary axis saturated, what we call isotropic, the performance is lower than needed. In this case $\delta_q = 4.8 \cdot 10^{-2}$ which is an order of magnitude higher that the specification. By dividing the manoeuvres in three groups as previously explained, the order of magnitude is always $10^{-3}$, like the specification. All of the networks have two hidden layers, and the average training time employed was around 30 seconds.

4.3 Onboard

In the last section we saw the agility model and $h_{tot}$ uncertainty affects over the estimation of the minimal time needed to go from one image to another. The developed models would help the agility model by reducing its computational charge. These elements would always be implemented in the ground segment. In this section we will explore the possibilities of neural networks when the actual state of the satellite is known.
4.3.1 Manoeuvre computation

The main difference of this section with respect to the previous one is the approach of the problematic. Hereafter we will estimate the manoeuvre directly, without using the variables that were used in the agility model to make reference to the images. From this point of view, this applications could be useful for any kind of agile satellite.

Another difference with the previous cases would be the fact of adding $h_{\text{tot}}$ as an input variable. This means that we assume that the satellite state is known in the context where these models might be used. An example of this situation would be onboard the satellite, where the total angular momentum is known as it is a consequence of all the perturbations affecting the satellite.

A manoeuvre would be determined with the initial and final attitudes, and will be conditioned by the the total angular momentum of the satellite. With this information, we considered two problems in order to see if neural networks would be capable of approximating the manoeuvres respecting the specifications.

The first problem is to compute the time minimal manoeuvre to go from $q_i$ to $q_f$ with a given state of the satellite $h_{\text{tot}}$. A second problem that we consider is to find the minimal $h_b$ when the manoeuvre time $T_{\text{man}}$ is fixed and we want to go from $q_i$ to $q_f$ with a given $h_{\text{tot}}$.

4.3.2 Design variables

To start, we created a model that estimates $h_b$ and $T_{\text{man}}$ at the same time. The needed input variables for this model would be $q_i$, $q_f$ and $h_{\text{tot}}$ for a rest-to-rest manoeuvre, and $w_i$, $w_f$ must be added in any other manoeuvre.

Afterwards we have done the model with a fixed manoeuvre time. In this case, the model inputs would be $q_i$, $q_f$, $h_{\text{tot}}$, $w_i$, $w_f$ and $T_{\text{man}}$. And the output would be $h_b$.

We saw in Chapter[2] that the reduction of the design variables dimension reduces the needed amount of observations needed in order to have the same accuracy. With this objective we applied some transformations to our variables. The first one is to introduce a single quaternion that describes the rotation that enables to pass from $q_i$ to $q_f$. 
A second transformation is done to this single quaternion. A quaternion can be defined as follows:

\[ q = \cos\left(\frac{\theta}{2}\right) + u \sin\left(\frac{\theta}{2}\right) \]

where \( u \) is a unitary vector and \( \theta \) is the rotation about the direction of \( u \). As \( u \) is a unitary vector, we can use its spherical representation without loosing information about its direction. The quaternion is then represented by three parameters instead of four. These parameters would be the azimuth and elevation of \( u \), represented by \( q_{az} \) and \( q_{el} \). And the rotation angle \( \theta \).

We also applied the same method with the output variable, the angular momentum vector \( h_b \). We would then have \( h_{baz} \) and \( h_{bel} \). And the output space dimension would be reduced from 3 to 2.

### 4.3.3 Data analysis

We used the same method that we employed previously in order to see if our sampling plan was space-filling. In this case we did not filter the manoeuvres with an angle limitation because these applications do not have any direct relation with the images that will be taken by the satellite. From all involved variables, \( h_{tot} \), \( w_i \), \( w_f \), \( T_{man} \) and \( h_b \) are always homogeneously spread through their respective space. This is because they are randomly generated with this intention. What we must check is if this sampling plan fills all the space related to the final quaternion \( q_f \).

We started by analysing the distribution for rest-to-rest manoeuvres that is shown in Fig. 4.10. Fig. 4.10(a) shows the relation between the quaternion rotation \( \theta \) and the manoeuvre duration \( T_{man} \). This is the typical profile of agile manoeuvres. We can see that \( \theta \in (0, 1.4) \) [rad] which would correspond to the maximum achievable rotation with this manoeuvres characteristics. Moreover, we can see that there a similar amount of samples for any \( T_{man} \) value but the density of samples decrease for higher \( T_{man} \). This density difference might reduce the network performance for longer manoeuvres. In Fig. 4.10(b) we can see that the quaternion azimuth \( q_{az} \) and elevation \( q_{el} \) are homogeneously distributed.

In order to continue with the data analysis, we will define a variable, \( w_b \). This angular velocity corresponds to the satellite’s angular velocity when its angular momentum is \( h_b \). This is a way of comparing the initial and final angular velocity to the angular velocity achieved during the manoeuvre. We can then classify the manoeuvres as \( w_i = k w_b \) where \( k \in [0, 1] \). The previously analysed case corresponds to \( k = 0 \). This is done in order to analyse more progressively the effect of non zero \( w_i \) and \( w_f \).
NEURAL NETWORK SURROGATE MODELLING FOR EARTH OBSERVATION MISSIONS

Figure 4.10: Data analysis for rest-to-rest manoeuvres, where $w_i = 0$

Starting with $w_i = \frac{1}{4}w_b$, we can see in Fig. (4.11(a)) that in this case $\theta \in (0, 0.9) \ [\text{rad}]$ which is less than for rest-to-rest manoeuvres. $\theta$ is not filling all the space, but its distribution is quite homogeneous. Fig. (4.11(b)) shows that for $q_a$ and $q_e$ the distribution is still homogeneous as it was for rest-to-rest manoeuvres.

We will finally analyse the case where $w_i = w_b$. In Fig. (4.12(a)) we can see that in this case $\theta \in (0, 0.7) \ [\text{rad}]$ which means that smaller rotations are achieved. In addition the $\theta$ profile have lost the original distribution not being homogeneous any more. In Fig. (4.12(b)) we also see how for high $w_i$ the final attitudes are no longer homogeneously distributed but cumulated around a region of the space.

With this analysis we can conclude that if we use this data for training some neural network, the performance will get worst as we try to approximate higher $w_i$. 
4. IMPLEMENTATION

Figure 4.12: Data analysis for non rest-to-rest manoeuvres, where \( w_i = w_b \)

Figure 4.13: Models for onboard \( T_{\text{man}} \) and \( h_b \) estimation

4.3.4 Surrogate overview

As we already did in the previous applications, we show here the final overview of our models that would be implemented. In Fig. (4.13) we can see the models that estimates \( T_{\text{man}} \) and \( h_b \) at the same time. And the models that only estimate \( h_b \) are shown in Fig. (4.14).

4.3.5 Results

We first created the network as defined in Fig. (??). Results are shown in Table (4.3). For the isotropic case we have a poor performance. However, if we use more complex networks, three hidden layers, good performances are achieved for the axis saturation classification. We can see that the case of \( \vec{z} \) saturated is at the limit of the order of magnitude.

We also created the networks for the model in Fig. (??). As Table (4.4) shows, this is the application in which, for axis \( \vec{x} \) and \( \vec{y} \), we obtained the

Figure 4.14: Model for \( T_{\text{man}} \) estimation for the agility model
NEURAL NETWORK SURROGATE MODELLING
FOR EARTH OBSERVATION MISSIONS

<table>
<thead>
<tr>
<th>Direction</th>
<th>( \delta q ) [s]</th>
<th>Architecture</th>
</tr>
</thead>
<tbody>
<tr>
<td>isotropic</td>
<td>0.1242</td>
<td>{16, 18}</td>
</tr>
<tr>
<td>( \vec{x} ) saturated</td>
<td>0.0032</td>
<td>{11, 19, 19}</td>
</tr>
<tr>
<td>( \vec{y} ) saturated</td>
<td>0.0036</td>
<td>{20, 19, 12}</td>
</tr>
<tr>
<td>( \vec{z} ) saturated</td>
<td>0.0094</td>
<td>{15, 17, 10}</td>
</tr>
</tbody>
</table>

Table 4.3: Models performance for \( T_{man} \) and \( h_b \) estimation of rest-to-rest manoeuvres

<table>
<thead>
<tr>
<th>Direction</th>
<th>( \delta q ) [s]</th>
<th>Architecture</th>
</tr>
</thead>
<tbody>
<tr>
<td>isotropic</td>
<td>0.0401</td>
<td>{17, 12}</td>
</tr>
<tr>
<td>( \vec{x} ) saturated</td>
<td>0.0023</td>
<td>{18, 18}</td>
</tr>
<tr>
<td>( \vec{y} ) saturated</td>
<td>0.0016</td>
<td>{17, 16}</td>
</tr>
<tr>
<td>( \vec{z} ) saturated</td>
<td>0.0138</td>
<td>{9, 18}</td>
</tr>
</tbody>
</table>

Table 4.4: Models performance for \( h_b \) estimation of rest-to-rest manoeuvres

best results without the need of a third hidden layer. We can see in the last row, that for \( \vec{z} \) saturated we have a poor performance as the previous case. This could be the consequence of the chosen quaternion representation. As we are using spherical azimuth and elevation, there is a singularity in the \( \vec{z} \) axis. This could be solved by changing the representation of the quaternion for \( \vec{z} \) saturated manoeuvres.

We can deduce that the network performance is very influenced by the variables chosen to define the problem, which is in fact one of the points we learned in Chapters 2 and 3.

At the actual moment we have not started to develop the models to estimate manoeuvres with non zero \( w_i \) and \( w_f \), but we expect worst performances than the ones obtained in the rest-to-rest manoeuvres. In fact it would be a consequence of the samples distribution shown in the the data analysis section.
Conclusions

Along this study we could see the most relevant aspects to take into account when creating meta models based on feedforward neural networks. These networks are so flexible that they can approximate barely any function, even in various dimensions. However, the problem is not their capabilities but how they need to be designed in order to achieve the desired performance.

We can conclude that there are two main aspects that will influence the network’s performance. In first place, the way we pose the problem will strongly influence the results. When we say to pose the problem we mean the variables that will be used to represent the problem and also the representation of the variables themselves. A second influence over the performance of the network is the way we collect all the samples that will be used for training. For instance, if we want to solve a problem, if we don’t get a representative sample of all the possible situations, the resulting network will have poor performance.

As a general conclusion we could say that for the treated problems, neural networks are capable of estimating the desired results with the demanded precision. However, they are sometimes in the limit, showing that the process is more difficult than expected, mainly because results depends more on the way we pose the problem that we want to solve than on the techniques used to build the network. It is worth to spend the main part of the designing time in choosing the variables that will be used and see how this choice affects the training data in order to have an homogeneous repartition of samples.

Neural networks are specially useful for various agile Earth observation satellite’s applications because of their adaptability to many different kinds of problems. It will remain as a promising research field.
Future work

As the internship has not arrived to its end, there are some thinks that can be done during the next two months. Firstly we should check the definition of some variables as they might be the reason of the poor performance of some models. It is the case of the last two models, where we used a spherical representation of the quaternion, which introduces a singularity in the $\vec{z}$ axis. We need to find a different way to represent the quaternion in order to correct this situation.

Secondly we need to introduce the angular velocity to some models and see how this affects to their performance. As we expect a degradation in the models performances, we must find a solution to the problem. We could apply some techniques seen in the literature, where the input space is divided into smaller regions that have a similar behaviour. In that case, each group of data would be used to train a new network. In order to use all these networks to solve the problem, we will need some mechanism that chooses the best network from all the trained ones. Another network might be used for this purpose.

A third possible subject to develop could be the use of radial basis functions instead of sigmoid functions as they have been proven to give good results too.

We do not think that we might have more time to go further than this, however, there are still a few options to be investigated:

An interesting line of investigation would be to introduce the performance index that we defined into the backpropagation algorithm. This would mean that instead of propagating the error in terms of target values, it would be interesting to backpropagate $\delta q$ through the whole network. The difficulty resides in the fact that the performance index should be differentiable with respect to the networks parameters, which is not a trivial problem to solve.

Another interesting future work would be to implement a quaternionic network and see if they perform better in the applications that we designed.
Finally we could always find new ways of representing the problems we treated, and new problems can be posed in order to help in some phase of an Earth observation satellite mission.
Satellite kinematics

In this appendix we will briefly present the most important points in defining a satellite’s attitude, the kinematic equations and the dynamics of a rigid body which completely define its rotational motion.

**Attitude** If we have two reference frame $[R_i]$, inertial, and $[R_{sat}]$, body, the attitude of the satellite can be defined as the orientation of $[R_{sat}]$ relative to $[R_i]$. This orientation can be defined with a direction cosine matrix, with three *body-axis* rotations (Euler angles) or with a single rotation about the called eigenaxis of rotation. We will work with this last representation using quaternions.

If $\mathbf{u}$ is a unit vector along the Euler axis or eigenaxis, relative to $[R_i]$ and $[R_{sat}]$, then:

$$\begin{align*}
q_0 &= \cos\left(\frac{\theta}{2}\right) \\
q &= \mathbf{u} \sin\left(\frac{\theta}{2}\right) \\
q &= [q_0, q_1, q_2, q_3]^T
\end{align*}$$

where $\theta$ is the rotation angle about the Euler axis. If we want to perform a frame change we have to apply the rotation of coordinates as follows:

$$\mathbf{x}^{[R_i]} = q \mathbf{x}^{[R_{sat}]} q^*$$  \hspace{1cm} (A.1)

where $\mathbf{x} = (0, \mathbf{x})$. For more on quaternion products and other properties of quaternions, [36].

**Kinematic equations** We define $\mathbf{w} = \mathbf{w}^{R_{sat}/R_i}$ as the angular velocity vector of $[R_{sat}]$ respect to $[R_i]$. The time-dependent relation between $[R_{sat}]$ and $[R_i]$ is the kinematic differential equation defined as follows with quaternions:

$$\dot{q} = \frac{1}{2} \Omega^x q,$$

where $\Omega^x = \begin{bmatrix}
0 & -w_1 & -w_2 & -w_3 \\
w_1 & 0 & w_3 & -w_2 \\
w_2 & -w_3 & 0 & w_1 \\
w_3 & w_2 & -w_1 & 0
\end{bmatrix}$  \hspace{1cm} (A.2)
In space vehicles, the body rates $w_1$, $w_2$ and $w_3$ are measured by rate gyros. Then equation [A.2] is integrated numerically onboard to determine the vehicle orientation. Quaternions have no geometric singularities as Euler angles do. Moreover, quaternions are well suited for onboard real-time computation because only products and no trigonometric relation exist in quaternion kinematic differential equations.

**Rigid body dynamics** In this part we will suppose that we have a rigid body with the reference body frame $[R_{sat}]$ in its center of mass. Then its angular momentum is defined as:

$$h_{tot} = Iw$$

where $I$ is the inertia matrix of the rigid body and $w$ is the previously defined angular velocity. We create a multi-body system by adding an element to the previous body. This element could be a reaction wheel for example. Then we have:

$$h_{tot} = Iw + h_w$$

where $h_w$ is the wheel’s angular momentum. The angular momentum theorem is:

$$\frac{\partial}{\partial t}(h_{tot})/R_i = \sum C_i$$

where $C_i$ are external forces applied on the body. If we have an isolated system, $C = 0$ and $\dot{h}_{tot} = 0$, so $h_{tot} = 0$ which is the principle of conservation of angular momentum. If we develop these equations we obtain:

$$\frac{\partial}{\partial t}(h_{tot})/R_i = I\dot{w} + w \times (Iw + h_w) = -\dot{h}_w$$

With this coupled non linear equations in [A.6] and the kinematic differential equations in [A.2], the rotational motion of a rigid body is completely described.
Backpropagation algorithm

The training process is in charge of defining the weights and biases values. This is not a trivial task, and we need the training data and a training algorithm to do it.

The most widely used training algorithm is the back-propagation. The most important points will be explained in order to understand the principles of the neural network training process. The algorithm must be provided with a set of training data which represent significant examples of the proper network behavior. The training data is a set of vectors containing the inputs and target outputs of the network.

\[(p_1, t_1), (p_2, t_2), \ldots, (p_N, t_N)\]

where \(p_i\) is the input vector and \(t_i\) is the target output associated to that input.

Each input will be applied to the network and the generated output will be compared to the target. The mission of the training algorithm is to adjust the weights and biases in order to minimize a performance index. The most usual performance index is the mean square error. We will note the performance index with \(F(x)\):

\[
F(x) = E[e^2] = E[(t - a)^2] = E[(t - a)^T(t - a)] \tag{B.1}
\]

where \(x\) is the vector of weights and biases.

Once the performance index is defined, the steepest descent algorithm is used to update the weights and biases values as follows:

\[
w_{ij}^{m}(k + 1) = w_{ij}^{m}(k) - \alpha \frac{\partial F}{\partial w_{ij}^{m}} \tag{B.2}
\]

\[
b_{i}^{m}(k + 1) = b_{i}^{m}(k) - \alpha \frac{\partial F}{\partial b_{i}^{m}} \tag{B.3}
\]

where \(\alpha\) is the learning rate. The complicated part of the algorithm is the computation of the partial derivatives.
We know that the error in the output of the network is an indirect function of the weights in the hidden layers. Because of this property, we can apply the chain rule to find the derivatives in (B.2) and (B.3) to obtain:

\[
\frac{\partial F}{\partial w_{i,j}^m} = \frac{\partial F}{\partial n_i^m} \frac{\partial n_i^m}{\partial w_{i,j}^m}
\]

(B.4)

\[
\frac{\partial F}{\partial b_i^m} = \frac{\partial F}{\partial n_i^m} \frac{\partial n_i^m}{\partial b_i^m}
\]

(B.5)

where \(n_{i,j}^m\) is the net input to layer \(m\) and is defined as follows:

\[
n_i^m = \sum_{i=1}^{S_{m-1}} w_{i,j}^m a_{j}^{m-1} + b_i^j
\]

(B.6)

where we can see the implicit relation with the weights and biases. So,

\[
\frac{\partial n_i^m}{\partial w_{i,j}^m} = a_j^{m-1}, \frac{\partial n_i^m}{\partial b_i^m} = 1
\]

(B.7)

We will define \(s_i^m\) as the sensitivity of \(F\) to changes of the \(i\) element of the input at layer \(m\),

\[
s_i^m = \frac{\partial F}{\partial n_i^m}
\]

(B.8)

Then Eq. (B.4) and Eq. (B.5) can be expressed as:

\[
\frac{\partial F}{\partial w_{i,j}^m} = s_i^m a_j^{m-1}
\]

(B.9)

\[
\frac{\partial F}{\partial b_i^m} = s_i^m
\]

(B.10)

We can now rewrite Eq. (B.2) and Eq. (B.3) as

\[
\begin{align*}
    w_{i,j}^m(k+1) &= w_{i,j}^m(k) - \alpha s_i^m a_j^{m-1} \\
    b_i^m(k+1) &= b_i^m(k) - \alpha s_i^m
\end{align*}
\]

(B.11)

(B.12)

And in matrix form:

\[
\begin{align*}
    W^m(k+1) &= W^m(k) - \alpha s^m (a^{m-1})^T \\
    b^m(k+1) &= b^m(k) - \alpha s^m
\end{align*}
\]

(B.13)

(B.14)

where \(s^m\) is the vector of sensitivities in layer \(m\).
B. BACKPROPAGATION ALGORITHM

**Back-propagation of sensitivities** is the process of computing the sensitivity at layer \( m \) from the sensitivity at layer \( m + 1 \). To obtain the relationship for the sensitivities, we must use the following Jacobian matrix:

\[
\begin{bmatrix}
\frac{\partial n_{m+1}^1}{\partial n_m^1} & \frac{\partial n_{m+1}^1}{\partial n_m^2} & \cdots & \frac{\partial n_{m+1}^m}{\partial n_m^m} \\
\frac{\partial n_{m+1}^2}{\partial n_m^1} & \frac{\partial n_{m+1}^2}{\partial n_m^2} & \cdots & \frac{\partial n_{m+1}^m}{\partial n_m^m} \\
\vdots & \vdots & \ddots & \vdots \\
\frac{\partial n_{m+1}^m}{\partial n_m^1} & \frac{\partial n_{m+1}^m}{\partial n_m^2} & \cdots & \frac{\partial n_{m+1}^m}{\partial n_m^m}
\end{bmatrix}
\]

(B.15)

We can finally write the Jacobian matrix as

\[
\frac{\partial n_{m+1}}{\partial n_m} = W^{m+1} \mathbf{F}^m(n^m)
\]

(B.16)

where

\[
\mathbf{F}^m(n^m) = \begin{bmatrix}
f^m(n_1^m) & 0 & \cdots & 0 \\
0 & f^m(n_2^m) & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots \\
0 & 0 & \cdots & f^m(n_S^m)
\end{bmatrix}
\]

(B.17)

and

\[
f^m(n_j^m) = \frac{\partial f^m(n_j^m)}{\partial n_j^m}
\]

(B.18)

We can then write the recurrence relation for the sensitivity by using the chain rule in matrix form

\[
s^m = \mathbf{F}^m(n^m(W^{m+1})^T s^{m+1}
\]

(B.19)

The sensitivities are back-propagated through the network from the last layer to the first layer

\[
s^M \rightarrow s^{M-1} \rightarrow \cdots \rightarrow s^2 \rightarrow s^1
\]

At this point, the last step to do is to calculate the first sensitivity, \( s^M \) and apply it to the relation in Eq. (B.19). This sensitivity is easy to obtain at the final layer:

\[
s_1^M = \frac{\partial F}{\partial n_1^M} = \cdots = -2(t_i - a_i)f^M(n_i^M)
\]

(B.20)

which can be expressed in matrix form as

\[
s^M = -2\mathbf{F}^M(n^M)(t - a)
\]

(B.21)
Bibliography


Acronyms

**ANN** Artificial Neural Network. [page6] [page16] [page19] [page23–27]

**AOCS** Attitude and Orbit Control System. [page8] [page9] [page33] [page34] [page36] [page42] [page43]

**CMG** Control Moment Gyroscope. [page6]

**CNES** Centre National d’Etudes Spatiales. [page5]

**FNN** Feedforward Neural Network. [page19] [page20] [page23] [page27] [page30] [page32] [page37] [page43]

**QMLP** Quaternion multilayer perceptrons. [page26]

**RMSE** Root Mean Square Error. [page14]