Generating Orbital Transfers with Differential Programming

Graduate Thesis

Submitted in partial fulfillment of the requirements of
BITS F422T Thesis

By

Komal Gupta
ID No. 2015B5A30330G

Under the supervision of:

Prof. Russell Boyce
&
Dr. Kinjal Banerjee

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE PILANI, K. K. BIRLA GOA CAMPUS
May 2020

Declaration of Authorship

I, Komal Gupta, declare that this Graduate Thesis titled, 'Generating Orbital Transfers with Differential Programming' and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: Komal Gupta

Date: 25 May 2020

Certificate

This is to certify that the thesis entitled, "*Generating Orbital Transfers with Differential Programming*" and submitted by Komal Gupta ID No. 2015B5A30330G in partial fulfillment of the requirements of BITS F422T Thesis embodies the work done by her under my supervision.

Supervisor	*Co-Supervisor*
Prof. Russell Boyce	Dr. Kinjal Banerjee
Chair for Space Engineering,	Assistant Professor,
UNSW Canberra	BITS-Pilani, K. K. Birla Goa Campus
Date:	Date:

"I don't study to know more, but to ignore less."

Sor Juana Inés de la Cruz

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE PILANI, K. K. BIRLA GOA CAMPUS

Abstract

Master of Science (Hons.)

Generating Orbital Transfers with Differential Programming

by Komal Gupta

Generating heteroclinic connections in three-body systems is a computationally intensive process consisting of multiple steps and integrations. Once the orbits and their associated manifolds are computed and their intersections with a suitably placed surface of section are found, discontinuity between the two trajectories - one in the unstable manifold and another in the stable manifold - is removed by means of an iterative correction process. This process, known as differential correction, linearises the system of equations to iteratively correct initial conditions. As a result, it requires hundreds of steps to converge to the solution, and often diverges or saturates. This dissertation explores an alternate approach to finding heteroclinic connections by leveraging the power of automatic differentiation to find gradients, that are then used in the correction process. We describe both methods and compare them in terms of their computing time and efficiency.‘

Acknowledgements

First and foremost, I must thank my parents, for their constant love and support throughout my life and for giving me the freedom to pursue my ambitions. I must also thank my mentor, Mr. Rasit Abay, for teaching me everything I know about Astrodynamics and for his continuous guidance, patience and encouragement. I am grateful to my supervisor, Prof. Russell Boyce, for giving me the opportunity to work with his brilliant group and to Dr. Kinjal Banerjee, for supporting me in this endeavour. I am also grateful to Dr. Sudantha Balage, for involving me in valuable discussions and for providing new perspectives on the problem at hand. Lastly, I am thankful for all my friends and family who never gave up on me and supported me throughout the course of this project.

Contents

List of Figures

Abbreviations

CR3BP	**C**ircular **R**estricted **Three** **B**ody **P**roblem
LPO	**L**iberation **P**oint **O**rbit
CNN	**C**onvolutional **N**eural **N**etworks
RNN	**R**ecurrent **N**eural **N**etworks
MSE	**M**ean **S**quared **E**rror
AD	**A**utomatic **D**ifferentiation
DP	**D**ifferentiable **P**rogramming

Physical Constants

Gravitational constant	G	=	$6.67\ 408 \times 10^{-11}\ \mathrm{m}^3\ \mathrm{kg}^1\ \mathrm{s}^{-2}$
Mass of the Earth	M_1	=	$3.986 \times 10^{14}\ kg$
Mass of the Moon	M_2	=	$4.903 \times 10^{12}\ kg$
Distance between Earth and Moon	D	=	$385\ 000\ 600\ m$

For my parents.

Chapter 1

Introduction

Orbital transfers are routinely employed in space missions to transfer spacecrafts from one trajectory to another. Different types of transfer orbits exist, which vary in their speed and energy efficiency. Most prominent examples include Hohmann transfers and bi-elliptic transfers. While designing these transfers, various forces that act on a spacecraft on its trajectory must be modelled using various models and approximations. The motion of a spacecraft is studied using the restricted N+1 body problem, where N celestial bodies move under the influence of one another and the spacecraft moves in the field of these bodies without influencing their motion [11]. Modern computers are powerful enough to take into account all the forces acting on the spacecraft and create extremely accurate simulations. Nevertheless, the process of designing orbital transfers is very computer intensive and simplifications are required to get a qualitative understanding of the scenario before the final trajectory design phase. The simplest model which grants insight into the motion of the spacecraft is one where the two most massive bodies are assumed to be moving in circular orbits around their common centre of mass. This is known as the *circular restricted three-body problem* or CR3BP.

In a three-body system, several points exist where the gravitational, centripetal and Coriolis forces on the third body add up in a way that it maintains a fixed position relative to the two larger bodies. These points were named *Lagrange points* after the Italian mathematician Joseph-Louis Lagrange who discovered them in 1772. Families of quasi-periodic orbits called *Lissajous orbits* are known to exist near these points, and are utilised for various space missions such as SOHO, Genesis, WMAP, Herschel and Planck. Koon et. al. [10] numerically demonstrated the existence of cost free transfers between planar Lissajous orbits (known as *Lyapunov orbits*). These transfers are known as *heteroclinic connections* and they asymptotically connect two Lyapunov orbits. Since their computation is a compute-intensive process, it has prompted several researchers to look for alternate ways of discovering them. Some of these works are discussed in the next section.

1.1 Literature review

A necessary step in the numerical computation of heteroclinic connection is to compute invariant manifolds of Lyapunov orbits. Haapala and Howell [8] use invariant manifolds alongside periapse Poincaré maps to determine heteroclinic connections between Lyapunov orbits of same energy. They choose a pair of trajectories from the unstable and stable manifolds of the two orbits and use an iterative correction process to remove the discontinuity between them at a suitably chosen surface of section. Davis et. al. [4] also use invariant manifolds and Poincaré maps to design optimal transfers between unstable periodic orbits of different energies. The computation of invariant manifolds is the most compute-intensive part of the process, prompting several authors to look for alternate methods to expedite the process. Beeson et. al. [1] use various approximation algorithms for quick evaluation of invariant manifolds. Shah and Beeson [14] attempt to use machine learning models to rapidly approximate invariant manifolds of Lyapunov orbits. De Smet and Scheeres [5] employ a similar approach, but leverage artificial neural networks to predict states on a periapse Poincaré map rather than compute the entire manifold.

1.2 Present work

The present work is concerned with numerical propagation of orbits to determine a particular class of orbital transfers known as heteroclinic connections. The process of finding these connections consists of several steps and iterative algorithms. In particular, once the orbits and their associated manifolds have been computed, the equations of motion are linearised and an iterative differential correction scheme is applied to find a connection. We use this scheme to find heteroclinic connections between L1 and L2 liberation point orbits within the earth-moon circular restricted three body problem. Moreover, we present an alternative method of finding these connections, one that leverages the power of automatic differentiation to find the required gradients and hence eliminates the need for using the linearised equations of motion. Finally, we also compare the two methods in terms of their speed and accuracy.

This dissertation is structured as follows. Chapter 2 covers the background for this research including the three body problem and its approximation, the circular restricted three body problem. Chapter 3 provides an overview of machine learning, and discusses automatic differentiation as well as the broader topic of differentiable programming. Chapter 4 describes the process of finding heteroclinic connections, using both classical and differentiable algorithms and compares the two. Finally, chapter 5 concludes the dissertation.

Chapter 2

Background

2.1 The general three-body problem

A problem that has perplexed physicists for centuries - the general three body problem - was first formulated in its current form by Issac Newton in 1687. It describes the motion of three bodies under the influence of their mutual gravitational field. Their motion in an inertial frame is described by the following set of coupled ODEs:

$$\frac{d^2\vec{r_3}}{dt^2} = G\sum_{j=1}^{2}\frac{M_j}{r_{j3}^3}(\vec{r_j} - \vec{r_3}) \tag{2.1}$$

where M_i and $\vec{r_i}$ are the mass and distance of the i^{th} object from the origin of the frame, and G is the universal gravitational constant. No analytical closed form solution exists for these equations, which must be integrated using numerical methods, making the problem difficult. Many special case solutions have been discovered by various scientists over the years. These include family of periodic solutions by Euler (Fig. 2.1) and Lagrange (Fig. 2.2) where the objects are collinear and form an equilateral triangle, respectively [12]. A simplification of this problem known as the circular restricted three-body problem is often used for theoretical analysis of real world scenarios. This simplification is called the *circular* or *elliptical restricted three-body problem* depending on the shape of the orbit, the former of which is studied as part of this research and described in the next section.

2.2 The circular restricted three-body problem

The circular restricted three body problem (CR3BP) is a simplification of the general three body problem in which one of the bodies is considered massless compared to the other two.

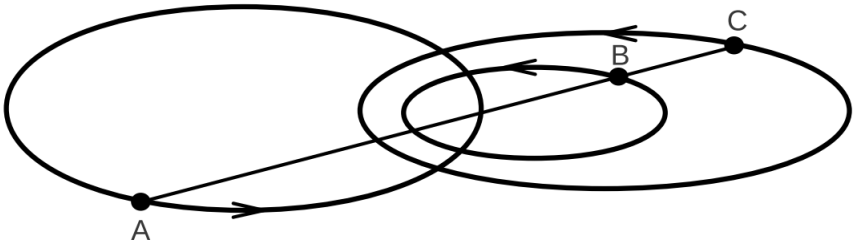


FIGURE 2.1: Euler's solution to the three body problem. Source: [12]

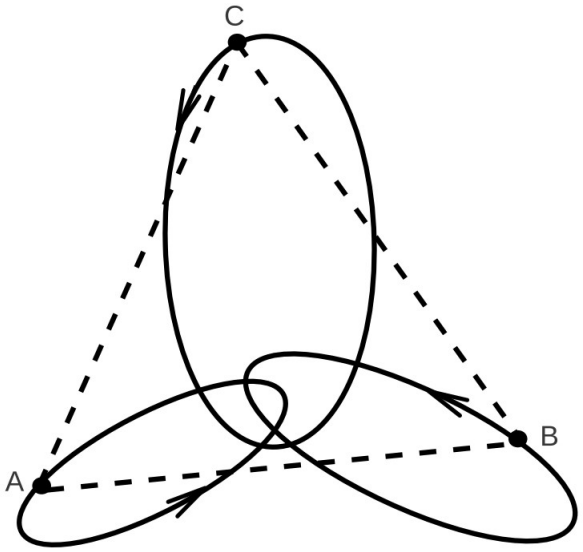


FIGURE 2.2: Lagrange's solution to the three body problem. Source: [12]

The secondary body is free to move whereas the two primary objects move in circular orbits around their centre of mass at a constant angular velocity. Since the smaller body exerts no force on the other two large objects, the motion of the two primaries can be approximated by two-body problem and can be solved analytically. A *synodic* or *co-rotating frame*, that has its origin at the centre of mass of the system and is rotating at the same angular velocity as the two primaries, is used to determine the dynamics of the smaller mass. The x-axis of this frame is defined as the line pointing from the larger to the smaller primary, the z-axis is defined as being parallel to their angular momentum, and the y-axis completes the right-handed coordinate system (Fig. 2.3). The problem is simplified further by using non-dimensional coordinates for time and length, such that the time period of their circular orbit is 2π and the distance between the two primaries is one non-dimensional unit. The masses are expressed as fractions of the total mass of the system using a quantity knwon as the mass parameter μ, such that $\mu = \frac{M_2}{M_1+M_2}$.

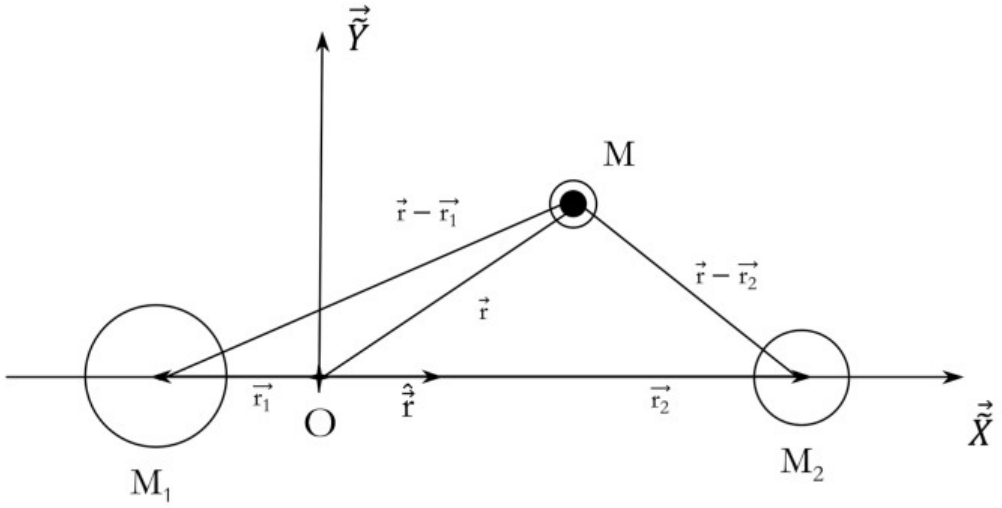


FIGURE 2.3: Co-rotating reference frame used to describe the circular restricted three body problem. The z-axis is directed out of the plane. Source: [6]

The equations of motion of the secondary in the rotating frame can now be written as

$$\begin{aligned} \ddot{x} - 2\dot{y} &= x - \frac{1-\mu}{r_{13}^3}(x+\mu) - \frac{\mu}{r_{23}^3}(x-1+\mu) \\ \ddot{y} + 2\dot{x} &= y - \frac{1-\mu}{r_{13}^3}y - \frac{\mu}{r_{23}^3}y \\ \ddot{z} &= -\frac{1-\mu}{r_{13}^3}z - \frac{\mu}{r_{23}^3}z \end{aligned} \tag{2.2}$$

A single conserved quantity exists for this system in this frame, known as the *Jacobi constant*. In terms of the position and velocity of the secondary object, it is defined as

$$J = x^2 + y^2 + \frac{2(1-\mu)}{r_{13}} + \frac{2\mu}{r_{23}} - v^2 \tag{2.3}$$

The Jacobi constant is related to the energy of the system by $J = -2H$. In terms of velocity, the equation 2.3 can be written as

$$\begin{aligned} v^2 &= x^2 + y^2 + \frac{2(1-\mu)}{r_{13}} + \frac{2\mu}{r_{23}} - J \\ v &= \sqrt{x^2 + y^2 + \frac{2(1-\mu)}{r_{13}} + \frac{2\mu}{r_{23}} - J} \end{aligned} \tag{2.4}$$

The term inside the root in equation 2.4 cannot be less than zero as the velocity of an object cannot be imaginary, allowing us to define a region in space outside of which the object will never be found. This region is bounded by surfaces where the velocity of the object is zero, known as *zero velocity surfaces* (or zero velocity curves, in two dimensions).

2.2.1 Lagrange points and orbits

The CR3BP has five equilibrium points, known as *Lagrange or Liberation points*, at which the object does not experience a net acceleration. These points can be determined by setting the velocity and acceleration terms to zero in equation 2.2. The locations of the equilibrium points in the rotating frame is shown in figure 2.4

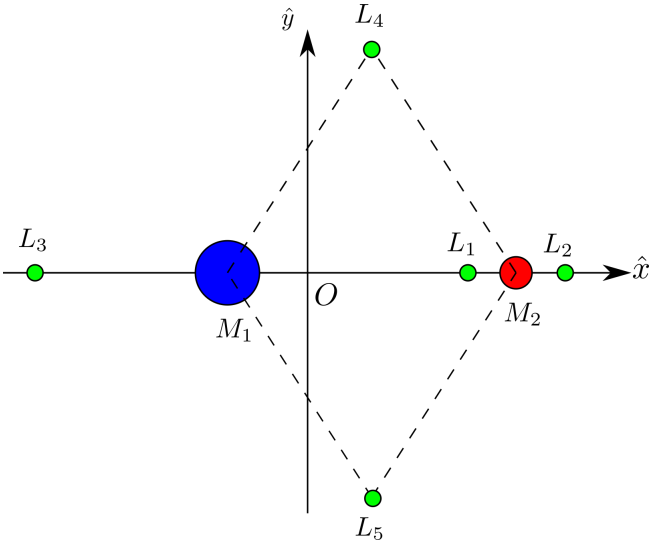


FIGURE 2.4: Locations of the five Lagrange points in the synodic frame.

Quasi-periodic orbits known as *Lissajous orbits* or *Liberation point orbits* exist in the vicinity of the Lagrange points. They are quasi-periodic because they do not close in finite time but remain close to the Lagrange points. Such orbits that lie entirely in the plane of the primary bodies are called *Lyapunov orbits*. The evolution of an object's state vector $\vec{X}$ along such an orbit is described by the following system of ODEs:

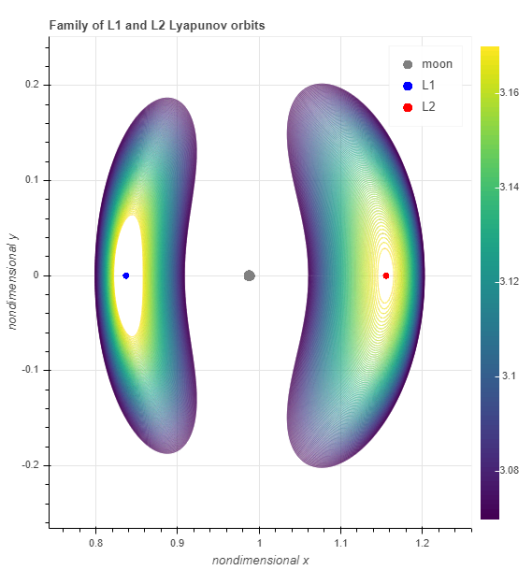
$$\begin{aligned} \dot{\vec{X}} &= f(\vec{X}) \\ \dot{\vec{\Phi}}(t, t_0) &= Df(\vec{X})\vec{\Phi}(t, t_0) \end{aligned} \tag{2.5}$$

where

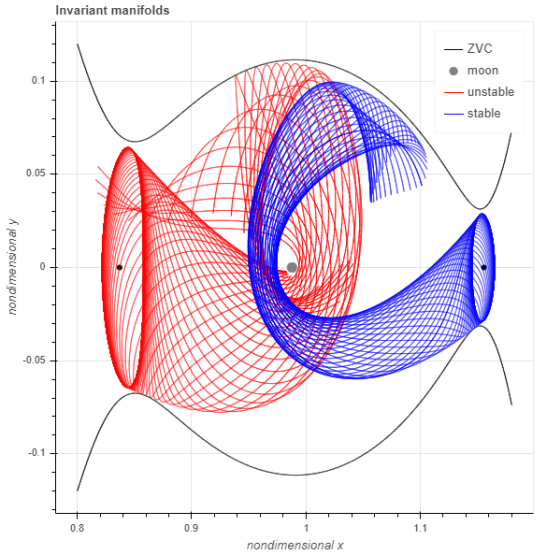
$$\vec{X} = [x \ y \ z \ \dot{x} \ \dot{y} \ \dot{z}]^T \tag{2.6}$$

$$\vec{X}(t) = \vec{\Phi}(t, t_0)\vec{X}(t_0) \tag{2.7}$$

In these equations, $f(\vec{X})$ is the set of equations of motion (2.2), $Df(\vec{X})$ is the Jacobian matrix and $\vec{\Phi}$ is the state transition matrix along the orbit. For a given value of mass parameter μ, a family of Lyapunov orbits exist around each Lagrange point, shown in Fig. 2.5a.



(A) Family of L_1 and L_2 Lyapunov orbits



(B) Unstable (red) and stable (blue) manifolds for L_1 and L_2 Lyapunov orbits respectively for J = 3.17

FIGURE 2.5: Lyapunov orbits and associated manifolds

2.2.2 Heteroclinic connections

The existence of cost free transfers between Lyapunov orbits of the same Jacobi constant were numerically proven by Koon et. al. [10] and are called *Heteroclinic connections*. Theoretically these transfers are free of cost, but in reality they require little fuel to move the object from the initial orbit to the transfer trajectory, and then from the transfer trajectory to the final orbit. The invariant manifold structure of Lyapunov orbits is exploited to compute these transfers. Invariant manifolds can be unstable or stable: an orbit's unstable manifold is the set of all trajectories that move away from the orbit in positive time along its unstable eigenvector. Conversely, an orbit's stable manifold is the set of all trajectories that move away from the orbit in negative time (or move towards the orbit in positive time) along its stable eigenvector. Invariant manifolds are shown in Fig. 2.5b. An object must depart the initial orbit on a trajectory within its unstable manifold and arrive at the final orbit on a trajectory within its stable manifold. The problem is solved by finding the points of intersection of the two manifolds in four dimensional phase space. The problem is simplified by using Jacobi constant to reduce the dimensionality of the problem by one. Further simplification is done by using a Poincaré surface of section. A Poincaré map is a map showing intersections of a flow (or trajectory in this case) with a certain predefined surface of lower dimension, known as the surface of section. By fixing a surface of section, the dimensionality of the problem is once again reduced by one. The problem is simplified greatly as it is reduced to finding intersections of stable and unstable manifolds on a two dimensional surface.

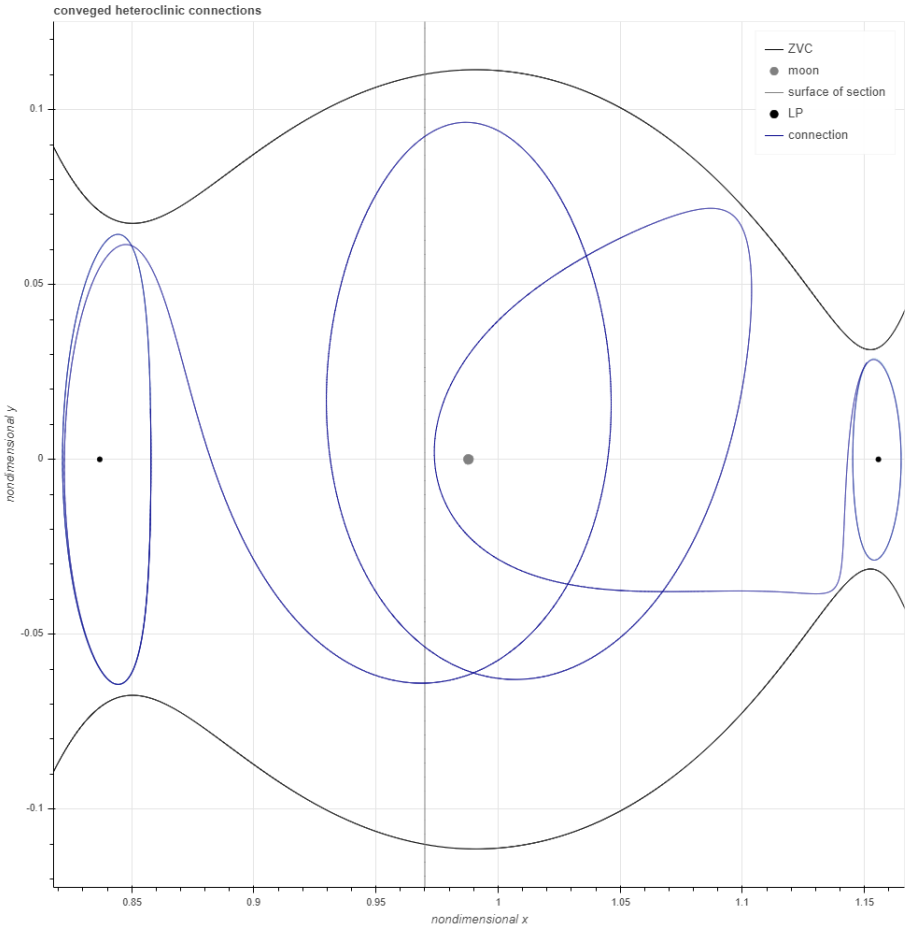


FIGURE 2.6: An example of heteroclinic connection for J = 3.17

Chapter 3

Machine Learning

Machine learning is a branch of computer science that deals with building algorithms to create statistical models of real world data [2]. These models are used to extract information from the data and/or make predictions on unseen data. A collection of examples that represent the scenario to be modelled is called a dataset. The dataset must contain enough examples of different types so that the model learns features in an unbiased manner.

3.1 Types of Machine Learning

Machine learning algorithms can be divided in four classes: supervised, unsupervised, semi-supervised and reinforcement [2]. These are described below.

Supervised learning Supervised learning is the art of making models that learn from labelled data. The dataset used to train the model consists of (feature, label) pairs of the form $\{(\vec{x_i}, y_i)\}_{i=1}^{N}$ that are used to alter parameters of the model. Each $\vec{x_i}$ is a vector of D dimensions where each of this vector is a feature of the example. Each y_i is a *label*, which can be a real number or an element belonging to a finite set of classes, or a more complex structure such as a vector or matrix [2]. Once the model is trained, it is tested by using a set of examples that were not used to train the model and have not been seen by the model. The accuracy of the model is measured by taking the difference between the predicted and actual labels. Examples of supervised learning include linear regression, logistic regression, decision trees, support vector machines and neural networks.

Unsupervised learning Unsupervised learning is the art of making models that learn from unlabeled data. The dataset is a collection of unlabeled examples of the form $\{\vec{x_i}\}_{i=1}^{N}$. Examples

of unsupervised learning include clustering, dimensionality reduction, anomaly detection and generative adversarial networks.

Semi-supervised learning Semi-supervised learning algorithms lie within the intersection of supervised and unsupervised learning, in which a model is trained on both labeled and unlabeled examples. The addition of unlabeled examples provides more information about the problem that using only labeled examples would, resulting in a better model.

Reinforcement learning Reinforcement learning is applied to problems where sequential decision making is required by a model to maximize the long term reward or goal. It aims to learn a *policy* that directs the actions of an *agent*. The agent is an algorithm that exists in an *environment* and perceives its *state* as a vector of features. Reinforcement learning models are used in robotics, video games, resource management in computers etc.

3.2 How Supervised Learning Works

This section describes the learning process of a model, in particular, a model that learns via labeled data. The dataset for supervised learning comprises of (input, label) pairs. The process starts with gathering the data and allotting appropriate labels to it. Next, the architecture of the model is chosen which depends on the problem at hand, for example, Convolutional Neural Networks (CNNs) are used for graphical inputs whereas Recurrent Neural Networks (RNNs) are suitable for textual inputs. The model takes in a series of inputs $\{\vec{x}_1, \vec{x}_2, ...\vec{x}_N\}$ and outputs the predicted labels $\{\tilde{y}_1, \tilde{y}_2, ...\tilde{y}_N\}$. The error between the predicted and actual label is then calculated by using an appropriate cost function. The most commonly used cost function is the Mean Squared Error (MSE), which is defined by the following equation:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \tilde{y}_i)^2 \tag{3.1}$$

Until now, the model has learned no information about the data it is trying to predict. This information is learned by the model via an algorithm known as *Backpropagation*. The learning process, in turn, is known as *training*. Its details in the context of a simple feed-forward neural network are presented below.

In essence, neural networks are complex functions that act on an input to generate an output. They comprise of multiple interconnected units that are logically organised in one or more layers. The relationship between the inputs and outputs are dictated by the parameters of the neural network, i.e., weights and biases. It is these parameters that are changed during the training

process. The parameters are changed in such a way that the neural network 'learns' the essential features in the dataset and thus gains the ability to make predictions about it. Exactly how much each parameter is changed depends on how much it influences the final error, and is determined by taking the gradients of the cost function with respect to each parameter in the neural network. This is done iteratively; in each iteration, the gradients of the cost function are computed and the parameters are changed, until the cost falls below certain accepted tolerance.

The computation of gradients, i.e., the partial derivatives of the cost function with respect to the parameters is not trivial. Doing so using either forward mode differentiation or finite differences would imply a significant increase in computational burden as the number of parameters in the network increases, and would make deep learning entirely infeasible. Instead, backpropagation relies on *automatic differentiation* (AD), which allows the gradients to be computed in a single backward pass. AD works by decomposing the differentials of a function according to the chain rule and then accumulating them in reverse. As an example, we consider the function z and write it in terms of elementary arithmetic operations and functions:

$$\begin{aligned} z &= f(x_1, x_2) \\ &= x_1 x_2 + sin(x_1) \\ &= w_1 w_2 + sin(w_1) \\ &= w_3 + w_4 \\ &= w_5 \end{aligned}$$

This can be represented as a graph, as shown in Fig. 3.1. In order to compute the derivatives, this graph is traversed from top to bottom, i.e., the chain rule is traversed from outside to inside. The seed value is set equal to 1 for the dependent variable; this is the variable whose derivatives have to be computed. The derivative of the dependent variable with respect to a sub-expression is given by the adjoint, $\bar{w} = \frac{\partial z}{\partial w}$. The operations to compute the derivative using AD are given by:

$$\begin{aligned} \bar{w}_5 &= 1 \ (seed) \\ \bar{w}_4 &= \bar{w}_5 \\ \bar{w}_3 &= \bar{w}_5 \\ \bar{w}_2 &= \bar{w}_3 \cdot w_1 \\ \bar{w}_1 &= \bar{w}_3 \cdot w_2 + \bar{w}_4 \cdot cos(w_1) \end{aligned}$$

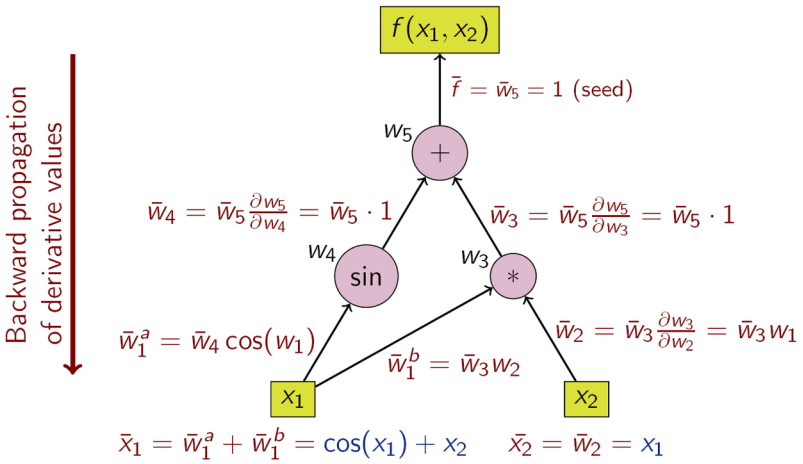


FIGURE 3.1: Computational graph of reverse-mode automatic differentiation

3.3 Differentiable Programming

Differentiable programming (DP) is a programming paradigm in which software are built by assembling networks of parameterized functional blocks, and are trained from examples using gradient-based optimization. Rather than being a new construct, this represents a shift in the way one thinks of deep learning, i.e., from a purely programming perspective. Within this paradigm, neural networks are not simply tools that learn from data, but they are programming constructs and the building blocks of larger software. In short, differentiable programming aims to build programs that can be differentiated throughout using automatic differentiation and therefore optimized using data itself.

Conventionally, programs are composed of functions, which take some inputs and return some outputs. Differentiable programming also consists of functions, although, these functions must be differentiable. Thus, all conditional statements, loops, arithmetic and reading and writing from memory must be made differentiable . Moreover, in modern programming practice, functions are accompanied by one or more unit tests, that evaluate it on a series of use cases. In machine learning terminology, this is a test set; previously unseen data on which a model is evaluated. The differentiability of the testing procedure must also be ensured; once all these constrains are satisfied, we can leverage the power of automatic differentiation and numerical optimization to optimize any program directly from data [13].

Chapter 4

Finding heteroclinic connections

This chapter describes the methods used to find heteroclinic connections in the CR3BP.

4.1 Data generation

The data for this work was generated in two parts. L_1 and L_2 Lyapunov orbits are generated for Jacobi constants between 3.07 and 3.17. Then, the invariant manifolds are generated for all orbits. All integrations are performed using SciPy's DOP853 integrator [9] with relative and absolute tolerances of 10^{-13} and 10^{-16} respectively. Finally, the intersections of unstable and stable manifolds is determined at suitable Poincaré surface of section and corrected to find heteroclinic connections. The process is described in more detail in the following sections.

4.1.1 Generating orbits

Lyapunov orbits for this work were generated using a single shooting algorithm called differential correction [11]. These orbits, symmetrical about the x-axis, lie in two dimensions and and intersect with this plane perpendicularly twice per period. The states at half- and full period are given by:

$$\vec{X}(t_0) = [x_0 \ 0 \ 0 \ 0 \ \dot{y}_0 \ 0]^T \tag{4.1}$$

$$\vec{X}(t_{T/2}) = [x_{T/2} \ 0 \ 0 \ 0 \ \dot{y}_{T/2} \ 0]^T \tag{4.2}$$

The algorithm start with an initial guess $\vec{X}(t_0)$, $\hat{\vec{X}}(t_0)$ which is close to the desired initial condition. This state is integrated until the next y-crossing, where the state is of the form:

$$\hat{\vec{X}}(t_{\hat{T}/2}) = [x_{\hat{T}/2} \ 0 \ 0 \ \dot{x}_{\hat{T}/2} \ \dot{y}_{\hat{T}/2} \ 0]^T \tag{4.3}$$

The correct initial conditions $\vec{X}(t_0)$ are obtained by 'correcting' the initial guess for the Lyapunov orbit until the final state takes the same form as equation (4.2). Small deviations in the initial and final states are related by the linearised system of equations (2.7):

$$\delta\vec{X}(t_{T/2}) \approx \vec{\Phi}(t_{T/2}, t_0)\delta\vec{X}(t_0) \tag{4.4}$$

where $\delta\vec{X}(t_{T/2}) = \hat{\vec{X}}(t_{\hat{T}/2}) - \vec{X}(t_{T/2})$. The initial guess can now be perturbed according to the equation:

$$\vec{X}(t_0) = \hat{\vec{X}}(t_0) - \delta\vec{X}(t_0) \tag{4.5}$$

This process is called *differential correction*. A modified version of this process was adopted to generate the orbits for this work. The initial conditions were obtained from Richardson's approximation, and were corrected alongside the Jacobi constant in order to converge to an orbit of desired J (J*). We use a cost function as follows:

$$G = [\dot{x}\ J]^T \tag{4.6}$$

and change the initial guess $\hat{\vec{X}}(t_0)$ such that $G = [0\ J^*]^T = G^*$ [15]. Small changes in final state $\vec{X}(t_{T/2})$ lead to deviations in G, according to

$$\delta G = \frac{\partial G}{\partial \vec{X}(t_f)}\delta\vec{X}(t_f) \tag{4.7}$$

where $\delta G = G - G^*$ is the error. In terms of small deviations in initial conditions this becomes

$$\delta G = \frac{\partial G}{\partial \vec{X}(t_f)}\vec{\Phi}(t_f, t_0)\delta\vec{X}(t_0) \tag{4.8}$$

$$\delta\vec{X}(t_0) = \left[\frac{\partial G}{\partial \vec{X}(t_f)}\vec{\Phi}(t_f, t_0)\right]^{-1}\delta G \tag{4.9}$$

The states can now be corrected using equation (4.5). Multiple iterations are required to converge to the solution due to the non-linear nature of the problem. Richardson's approximation was used as the initial guess first determine an orbit with J=3.17. After this the J was changed incrementally (with step size of 0.001) and the modified differential correction scheme was used with the initial condition corrected in the previous iteration as the starting guess for next iteration. The process was continued until all the orbits were generated, shown in Fig. 2.5a.

4.1.2 Generating manifolds

Invariant manifolds were generated at different departure locations along each orbit. The manifolds are parameterized by $\tau \in [0, T)$, which were chosen randomly to generate 500 trajectories. The initial state on each trajectory was obtained from equation (4.10). Let $\vec{X}^*$

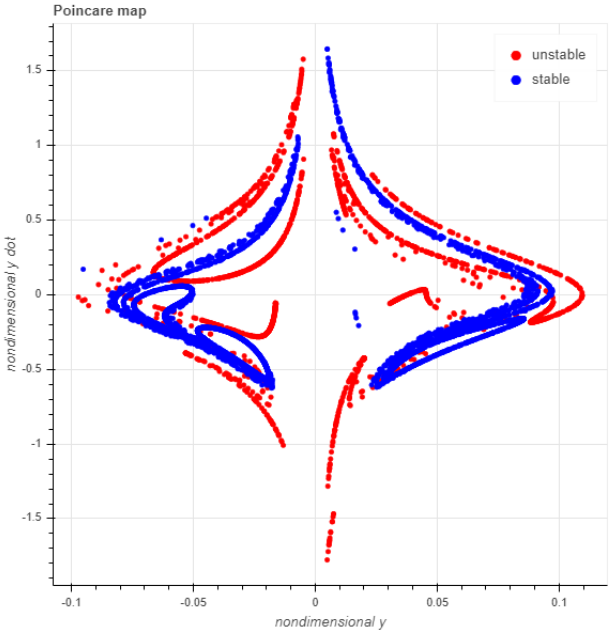


FIGURE 4.1: Poincare map for J = 3.17 and surface of section $x = 1 - \mu$

be a point on the orbit and $\vec{Y}_u(\vec{X}^*)$ and $\vec{Y}_s(\vec{X}^*)$ be the orbit's normalized unstable and stable eigenvectors at $\vec{X}^*$. Then the initial guesses for the unstable and stable manifolds at $\vec{X}^*$ are

$$\begin{aligned} \vec{X}_u(\vec{X}^*) &= \vec{X}^* \pm \epsilon \vec{Y}_u(\vec{X}^*) \\ \vec{X}_s(\vec{X}^*) &= \vec{X}^* \pm \epsilon \vec{Y}_s(\vec{X}^*) \end{aligned} \tag{4.10}$$

where ϵ is the magnitude of perturbation along the respective eigenvector. The plus of minus sign in the equation were chosen suitably to obtain the trajectories which move towards the moon. The trajectories were then integrated forward and backward in time for L_1 and L_2 Lyapunov orbits respectively, until they either escape the vicinity of the moon or impact the moon (Fig. 2.5b).

4.2 Generating connections

4.2.1 Using Linearised Equations

Once the orbits and corresponding manifolds are obtained, their intersections with a suitable Poincaré surface of section [7] is computed, which generates a Poincaré map (Fig. 4.1) In order to determine the connections, pairs of points on the Poincaré map were chosen as initial guesses for the actual trajectory, which is found by correcting these guess using a differential correction algorithm [8] and in doing so removing the discontinuity between the two parts of the trajectory that lie in separate manifolds. Although the plane at x-coordinate of the moon,

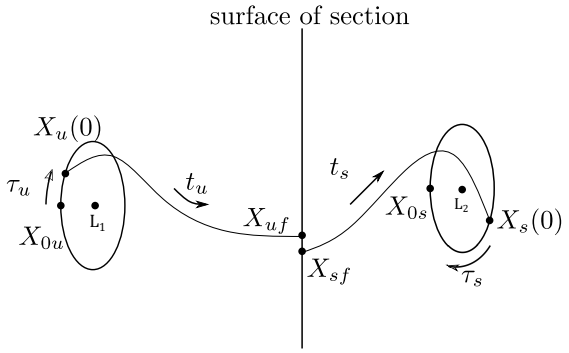


FIGURE 4.2: Differential correction scheme to obtain heteroclinic connections

i.e., $x = 1 - \mu$ is the most intuitive choice for the surface of section, we observed that the differential correction scheme diverged for initial conditions that were too close to the moon. This was because differential correction uses linearised system of equations, which are a poor approximation in the neighborhood of a massive body. This problem can be resolved by simply changing the surface of section such that the initial conditions on the Poincaré map lie farther from the moon.

Each trajectory is characterized by the coast time along the Lyapunov orbit (τ_u, τ_s) and the time spent on the manifold (t_u, t_s). As a result, each point on the Poincaré map corresponds to a trajectory defined by τ and t. Let $\vec{X}_0$ be a fixed point on each Lyapunov orbit such that $y = 0,\ \dot{y} > 0$ and $\vec{X}_0 = \vec{X}(\tau = 0) = \vec{X}(\tau = T)$. Then the initial states on the manifolds $\vec{X}_{u,s}(0)$, are defined by equation (4.10) with $\vec{X}^* = \vec{X}(\tau_{u,s})$

$$\begin{aligned} \vec{X}_u(0) &= \vec{X}(\tau_u) + \epsilon \vec{Y}_u \\ \vec{X}_s(0) &= \vec{X}(\tau_s) - \epsilon \vec{Y}_s \end{aligned} \tag{4.11}$$

and the final states along the manifolds are written as $\vec{X}_{uf,sf} = \vec{X}_{u,s}(t_{u,s})$ (Fig. 4.2). If we begin with initial conditions such that the difference between $\vec{X}_{uf}$ and $\vec{X}_{sf}$ is small, it is possible to iteratively update the parameters $[\tau_u,\ \tau_s,\ t_u,\ t_s]$ so that $|\vec{X}_{uf} - \vec{X}_{sf}|$ becomes approximately zero. As the Jacobi constant along both manifold arcs is the same, we only need to change three of the four parameters. The design parameters and cost function are defined as

$$\vec{Q} = [\tau_u\ \tau_s\ t_u]^T \tag{4.12}$$

$$\vec{F}(\vec{Q}) = [x_{uf} - x_{sf}\ y_{uf} - y_{sf}\ \dot{x}_{uf} - \dot{x}_{sf}]^T \tag{4.13}$$

Let $\vec{Q}^*$ be the desired solution such that $\vec{F}(\vec{Q}^*) = 0$. A Taylor expansion of the cost function

gives us:

$$\begin{aligned} \vec{F}(\vec{Q^*}) &= \vec{F}(\vec{Q}) + D\vec{F} \cdot (\vec{Q^*} - \vec{Q}) \\ -D\vec{F} \cdot (\vec{Q^*} - \vec{Q}) &= \vec{F}(\vec{Q}) \\ \delta\vec{Q} &= -D\vec{F}^{-1} \cdot \vec{F}(\vec{Q}) \end{aligned} \tag{4.14}$$

where $\delta\vec{Q} = \vec{Q^*} - \vec{Q}$ and $D\vec{F}$ is the Jacobian matrix of the cost function. The parameters are corrected until $|\vec{F}| < \Delta$, where Δ represents the selected tolerance for convergence. Once again, due to the non-linearity of the system, hundreds of updates are required before the cost function reduces to below an acceptable tolerance.

We chose closest 200 pairs of points from the Poincaré map, and corrected them until the final error in position was 1 m and in velocity was 1 mm/s. All integrations are performed using the same integrator as in the previous section.

4.2.2 Using Automatic Differentiation

In this section we discuss a novel approach that uses back-propagation and the deep learning library PyTorch to find heteroclinic connections. We use the same data for orbits and manifolds as in the previous section. Before starting the correction process, the intersections of the manifolds with the surface of section are found, and pairs of points which are close to each other are identified. We then build a model which takes the time parameters $[\tau_u, \tau_s, t_u, t_s]$ and integrates predefined initial conditions to returns the quantity $|\vec{X}_{uf} - \vec{X}_{sf}|$ (which can be interpreted as the loss L of the model), where all symbols have the same meaning as in the previous section. The ODE solvers used in the model are from the library provided by Chen et. al. [3] and are fully differentiable. This library allows loss to be back-propagated through all the operations of the solver using the adjoint method. By writing our code using this library, we can obtain gradients of $|\vec{X}_{uf} - \vec{X}_{sf}|$ with respect to the inputs by simply back-propagating it, similar to the way loss is back-propagated during the training process of neural networks. Furthermore, by framing the problem in this way, we can use optimizers from PyTorch to minimize the loss L quickly and efficiently.

In this section we solve the equations using RK8(7)-13M, a Runge-Kutta method of order 8 with 7th order error estimate and 13 stages, and relative and absolute tolerances of 10^{-13} and 10^{-16} respectively. Stochastic Gradient Descent is used as the optimizer, with variable learning rate and momentum. Not all initial conditions converge; the ones that do converge to within 100 m in position and 1 mm/s in velocity. A comparison between the two methods is outlined in the following paragraph.

Comparing the two methods The main difference between the two processes is computational time and memory requirement. Since it relies on several approximations and using the linearised system of equations, the classical algorithm needs to store less variables in memory and has lower memory requirement. On the other hand, using backpropagation requires all the intermediate variables using the process to be stored, thereby needing significantly more memory. Moreover, while the classical algorithm takes < 5 seconds for one iteration of differential correction, the novel method takes ~ 30 seconds for the same.

Chapter 5

Summary and future work

The present work investigates the process of identifying heteroclinic connections between Lyapunov orbits in the earth-moon circular restricted three body problem using techniques derived from Machine Learning. The theoretical background regarding the CR3BP is provided in chapter 2 along with basics of Machine Learning and Differentiable Programming in chapter 3. Following this, the classical approach to discovering connections is discussed in chapter 4 alongside a novel method that employs automatic differentiation for the same. Comparing the two methods for the same set of initial conditions shows that the classical method is faster and more computationally efficient than the novel method. This is because the classical method relies on several approximations and needs to store less variables in memory, while the novel method stores significantly more variables in order to back-propagate through the entire process.

Bibliography

[1] Ryne Beeson, Devin Bunce, and Victoria Coverstone. "Approximation methods for quick evaluation of invariant manifolds during global optimization". In: *AIAA/AAS Astrodynamics Specialist Conference* (2016).

[2] Andriy Burkov. *The Hundred-Page Machine Learning Book*. 1st ed. Andriy Burkov, 2019. ISBN: 9781999579500.

[3] Ricky T. Q. Chen et al. "Neural Ordinary Differential Equations". In: *Advances in Neural Information Processing Systems* (2018).

[4] Kathryn E. Davis et al. "The use of invariant manifolds for transfers between unstable periodic orbits of different energies". In: *Celestial Mechanics and Dynamical Astronomy* 107.4 (2010), pp. 471–485.

[5] Stijn De Smet and D Scheeres. "Identifying heteroclinic connections using artificial neural networks". In: *Acta Astronautica* 161 (May 2019), pp. 192–199.

[6] Thibaud Etienne. *The three-body problem*. URL: `https://thibaudetienne.wordpress.com/scientific-interests/the-three-body-problem-in-classical-and-quantum-mechanics/`.

[7] Komal Gupta et al. "Learning Orbital Transfers in the Three-Body Problem". In: *Unpublished work* (2019).

[8] Amanda Haapala and Kathleen Howell. "Trajectory design using periapse Poincaré maps and invariant manifolds". In: *Advances in the Astronautical Sciences* 140 (Jan. 2011), pp. 415–434.

[9] E. Hairer, S. P. Norsett, and G. Wanner. *Solving Ordinary Differential Equations I. Nonstiff Problems*. 2nd ed. Springer-Verlag, 1993.

[10] Wang Sang Koon et al. "Heteroclinic connections between periodic orbits and resonance transitions in celestial mechanics". In: *Chaos: An Interdisciplinary Journal of Nonlinear Science* 10.2 (2000), pp. 427–469.

[11] "Low-Energy Lunar Trajectory Design". In: John Wiley & Sons, Ltd, 2014. Chap. 2, pp. 27–115. ISBN: 9781118855065. DOI: `10.1002/9781118855065.ch2`.

[12] Zdzislaw Musielak and Billy Quarles. "The three-body problem". In: *Reports on Progress in Physics* 77 (June 2014), p. 065901. DOI: 10.1088/0034-4885/77/6/065901.

[13] Simone Scardapane. *Deep learning from a programmer's perspective (aka Differentiable Programming).* URL: https://towardsdatascience.com/deep-learning-from-a-programmers-perspective-aka-differentiable-programming-ec6e8d1b7c60.

[14] V. Shah and R. Beeson. "Rapid Approximation of Invariant Manifolds Using Machine Learning". In: *AAS17-784, AIAA/AAS Astrodynamics Specialist Conference, Stevenson, WA* (Aug. 2017).

[15] W.E. Wiesel and Air Force Institute of Technology (U.S.) *Modern Astrodynamics*. Aphelion Press, 2003. ISBN: 9780974827216.