Dissertation presented to the Instituto Tecnológico de Aeronáutica, in partial fulfillment of the requirements for the degree of Master of Science in the Graduate Program of Physics, Field of Atomic and Molecular Physics.

Rodrigo Pires Ferreira

VARIATIONAL QUANTUM ALGORITHM FOR ORBIT PROPAGATION WITH GRAVITATIONAL PERTURBATIONS AND ATMOSPHERIC DRAG

Dissertation approved in its final version by signatories below:

Prof. Dr. André Jorge Carvalho Chaves

Advisor

Prof. Dr. Willer Gomes dos Santos

Co-advisor

Prof[a]. Dr[a]. Emília Villani

Dean for Graduate Education and Research

Campo Montenegro
São José dos Campos, SP - Brazil
2023

Cataloging-in-Publication Data
Documentation and Information Division

Pires Ferreira, Rodrigo
Variational Quantum Algorithm for Orbit Propagation with Gravitational Perturbations and Atmospheric Drag / Rodrigo Pires Ferreira.
São José dos Campos, 2023.
101f.

Dissertation of Master of Science – Course of Physics. Area of Atomic and Molecular Physics – Instituto Tecnológico de Aeronáutica, 2023. Advisor: Prof. Dr. André Jorge Carvalho Chaves. Co-advisor: Prof. Dr. Willer Gomes dos Santos.

1. Computação quântica. 2. Mecânica orbital. 3. Equações diferenciais. 4. Algoritmos. 5. Mecânica quântica. 6. Perturbação orbital. 7. Astronomia. I. Instituto Tecnológico de Aeronáutica. II.Title.

BIBLIOGRAPHIC REFERENCE

PIRES FERREIRA, Rodrigo. **Variational Quantum Algorithm for Orbit Propagation with Gravitational Perturbations and Atmospheric Drag**. 2023. 101f. Dissertation of Master of Science in Physics – Instituto Tecnológico de Aeronáutica, São José dos Campos.

CESSION OF RIGHTS

AUTOR NAME: Rodrigo Pires Ferreira
PUBLICATION TITLE: Variational Quantum Algorithm for Orbit Propagation with Gravitational Perturbations and Atmospheric Drag
PUBLICATION KIND/YEAR: Dissertation / 2023

_____

Rodrigo Pires Ferreira
Rua Taperoá, 145, Cidade Monções
04.571-060 – São Paulo - SP

VARIATIONAL QUANTUM ALGORITHM FOR ORBIT PROPAGATION WITH GRAVITATIONAL PERTURBATIONS AND ATMOSPHERIC DRAG

Rodrigo Pires Ferreira

Thesis Committee Composition:

Profª. Drª.	Lara Kühl Teles	Chairperson	- ITA
Prof. Dr.	André Jorge Carvalho Chaves	Advisor	- ITA
Prof. Dr.	Willer Gomes dos Santos	Co-advisor	- ITA
Prof. Dr.	Tobias Frederico	Internal Member	- ITA
Prof. Dr.	Diego Rabelo da Costa	External Member	- UFC

ITA

To my family and friends.

Acknowledgments

These last few years have been a remarkable journey. I was lucky to meet many people who made me grow in numerous ways. On that note, I want to thank my family for their immense support throughout all these years. My mother and father created all the conditions for me to thrive and pursue my boldest dreams. I cannot convey their importance and how grateful I am for their love.

I am also thankful for my friends and girlfriend, who were always there for me when I needed them. As a young researcher, it is challenging to address complex questions on Quantum Computing. Their support was crucial, and I would really not get this far without them.

Academically, I am grateful to have worked under Dr. André Chaves and Dr. Willer Gomes. They are always available and happy to discuss any topic of our project. They taught me many aspects of the scientific method, but more importantly, they enhanced my critical thinking substantially. Their guidance and companionship were fundamental during this work.

Finally, I am thankful to CAPES for funding this research for the past few years. I also want to thank ITAEx for the financial support and for encouraging me to foster Quantum Computing research at ITA.

"There are two possible outcomes:
if the result confirms the hypothesis, then you've made a measurement;
if the result is contrary to the hypothesis, then you've made a discovery."
— Enrico Fermi

Resumo

Algoritmos Variacionais Quânticos (VQAs) são métodos híbridos que usam otimizadores clássicos e circuitos quânticos para resolver diferentes problemas de minimização. Devido a similaridades com as redes neurais, esses circuitos podem ser chamados de redes neurais quânticas, uma vez que são compostos por múltiplas camadas de portas lógicas quânticas cujos parâmetros devem ser otimizados a fim de minimizar uma dada função de perda. O objetivo deste trabalho é usar um VQA para resolver sistemas de equações diferenciais acopladas que descrevem o movimento de um satélite na presença de perturbações de órbita. Nós implementamos o VQA para propagar uma órbita com arrasto atmosférico e perturbação gravitacional considerando os efeitos do termo J_2 no geopotencial – segundo harmônico zonal da Terra usado para descrever a variação do potencial gravitacional devido à geometria oblata do planeta. Após resolver uma versão bi-dimensional simplificada do problema, nós propagamos a órbita 3D e comparamos a solução via VQA com a numérica. Finalmente, nós discutimos os desafios e oportunidades existentes neste método e como melhorá-lo em trabalhos futuros.

Abstract

Variational quantum algorithms (VQAs) are hybrid methods that use classical optimizers and quantum circuits to solve different minimization problems. Due to similarities with neural networks, these circuits might be called quantum neural networks, as they are composed of many layers of quantum gates whose parameters must be optimized to minimize a given loss function. This work's objective is to use a VQA for solving systems of coupled differential equations that describe the satellite's motion in the presence of orbit perturbations. We implement the VQA for propagating an orbit with atmospheric drag and gravitational perturbation considering the effects of the J_2 term – Earth's second zonal harmonic that's used to describe the variation of the gravitational potential due to the planet's oblate geometry. After solving a simplified two-dimensional version of the problem, we propagate the actual 3D orbit and compare the VQA solution to the numerical one. Finally, we discuss the existing challenges and opportunities of this method and how to improve it in future works.

List of Figures

List of Tables

List of Abbreviations

BP	Barren Plateau
CV	Continuous-Variable
FDM	Finite Difference Method
GA	Genetic Algorithm
HEO	High Earth Orbit
HHL	Harrow-Hassidim-Lloyd
LDE	Linear Differential Equation
LEO	Low Earth Orbit
LSODE	Livermore Solver for Ordinary Differential Equations
MEO	Medium Earth Orbit
MREC	Mean Relative Error per Cycle
NISQ	Noisy Intermediate-Scale Quantum
NLDE	Non-Linear Differential Equation
NN	Neural Network
PSO	Particle Swarm Optimization
QEC	Quantum Error Correction
QNN	Quantum Neural Network
VQA	Variational Quantum Algorithm
VQC	Variational Quantum Circuit
VQE	Variational Quantum Eigensolver

List of Symbols

Quantum Computing and Variational Quantum Algorithms

α	Amplitude related to the state $\|0\rangle$
β	Amplitude related to the state $\|1\rangle$
$\|\psi\rangle$	Generic quantum state
θ	Bloch sphere's angle in the xy-plane
θ_b	Bias vector's free parameter
θ_W	Weight matrix's free parameter
Θ	Parameter set
ϕ	Bloch sphere's angle relative to the z-axis
ϕ_f	Nonlinear activation function
σ_k	Pauli operator
ζ	Global phase
$\vec{a}$	Answer to be determined by the neural network
$\vec{b}$	Bias vector
$A(t)$	Time-dependent amplitude in the x-axis
$B(t)$	Time-dependent amplitude in the y-axis
$C(t)$	Time-dependent amplitude in the z-axis
$C_{2D,GP}$	Amplitudes' 1^{st} and 2^{nd} derivatives terms neglected in the loss function $\mathcal{L}_{2D,GP}$
$C_{3D,GP}$	Amplitudes' 1^{st} and 2^{nd} derivatives terms neglected in the loss function $\mathcal{L}_{3D,GP}$
$C_{3D,AD}$	Amplitudes' 1^{st} and 2^{nd} derivatives terms neglected in the loss function $\mathcal{L}_{3D,AD}$
B	Beamsplitter gate
D	Displacement gate
E_N	Euclidean norm
H	Hamiltonian
L_k	Neural network's k-th layer
k_{Num}	Numerical result
k_{VQC}	VQC's result
K	Kerr gate
$\mathcal{L}$	Loss function
$\mathcal{L}_D$	Loss function corresponding to the differential equation

$\mathcal{L}_I$	Loss function corresponding to the initial conditions
$\mathcal{L}_{2D,GP}$	Loss function of the 2D orbit propagation with gravitational perturbation
$\tilde{\mathcal{L}}_{2D,GP}$	Previous loss function without neglecting the amplitudes' time derivatives
$\mathcal{L}_{3D,GP}$	Loss function of the 3D orbit propagation with gravitational perturbation
$\tilde{\mathcal{L}}_{3D,GP}$	Previous loss function without neglecting the amplitudes' time derivatives
$\mathcal{L}_{3D,AD}$	Loss function of the 3D orbit propagation with atmospheric drag
$\tilde{\mathcal{L}}_{3D,AD}$	Previous loss function without neglecting the amplitudes' time derivatives
M	Symplectic matrix
$\hat{p}$	Momentum operator
p_i	Probability of the i-th outcome
R	Rotation gate
S	Squeeze gate
$\vec{x}$	Input vector
$\vec{y}$	Output vector
$\vec{v}$	Generic vector
$\hat{x}$	Position operator
X	Quantum NOT gate
W	Weight matrix

Orbital Mechanics

θ_{TA}	True anomaly
μ	Standard gravitational parameter
ρ	Atmospheric density
ω	Argument of the perigee
$\vec{\omega}_E$	Earth's angular velocity
Ω	Right ascension of the ascending node
A	Satellite's frontal area
C_D	Drag coefficient
$\vec{D}$	Drag force
e	Eccentricity
$\vec{F}_G$	Gravitational force
G	Universal gravitational constant
h	Specific angular momentum
i	Inclination
J_k	K-th zonal harmonic
M	Central body's mass
m	Satellite's mass
N	Node line

n_p	Number of periods within the total simulation time
P_k	K-th Legendre polynomial
$\vec{p}$	Net perturbative acceleration
p_{LG}	Perturbative acceleration due the lunar gravity
p_{RP}	Perturbative acceleration due the solar radiation pressure
R	Earth's equatorial radius
$\vec{r}$	Position vector
$\vec{r}_0$	Initial position vector
r_a	Apogee radius
r_p	Perigee radius
$\vec{v}$	Velocity vector
$\vec{v}_{atm}$	Inertial velocity of the atmosphere at a specific point
$\vec{v}_{rel}$	Satellite's relative velocity to the atmosphere
$\vec{v}_0$	Initial velocity vector
x_0	Position's x-component at $t = 0$
$\dot{x}_0$	Velocity's x-component at $t = 0$
y_0	Position's y-component at $t = 0$
$\dot{y}_0$	Velocity's y-component at $t = 0$
z_0	Position's z-component at $t = 0$
$\dot{z}_0$	Velocity's z-component at $t = 0$

Contents

1 Introduction

Orbital Mechanics can be defined as the study of the motion of artificial bodies. Such a broad field includes the analysis of orbital maneuvers (FEHSE, 2003), effects of perturbations in non-controlled orbits (CHOBOTOV, 2002), and the cost of delta-v in orbital maintenance (KÖNIGSMANN *et al.*, 1996) – the change in velocity of a satellite to keep it in its desired orbit. Among these problems, we are primarily interested in orbit propagation, which is subject to different perturbations depending on the type of orbit.

For instance, the orbit altitude is a factor that might determine which perturbation effects will be more or less important. In low Earth orbits (LEOs) – ranging from a few hundred to 2,000 kilometers – the atmospheric drag is generally the most prominent effect (PRIETO *et al.*, 2014). Meanwhile, a medium Earth orbit (MEO), which goes from 2,000 to approximately 35,800 kilometers, is subject to multiple perturbations (*e.g.*, Earth's oblateness, solar radiation pressure, atmospheric drag), and the prevailing one depends on the specific altitude, inclination, and eccentricity of the orbit (JOHNSON, 2010). Finally, in high Earth orbits (HEOs), defined by altitudes higher than 35,800 km, the lunar and solar gravitational pulls are arguably the most crucial effects – although other perturbations might be relevant as well (LARA *et al.*, 2012).

Such highly complex satellite orbits require accurate simulations to guarantee they will work properly and reduce costs. However, the classical orbital propagation model can be computationally expensive – regarding memory and execution time – when we are trying to simulate real systems (LUO; YANG, 2017). This is particularly true when our models include all relevant orbit perturbations (*e.g.*, atmospheric drag, solar radiation pressure, Earth's oblateness, third-body perturbation, and tide forces) and when it consider the interaction between multiple satellites (satellite constellation) (CHOBOTOV, 2002).

On that note, it is necessary to investigate enhanced ways of simulating orbital propagation scenarios, which involve solving differential equations accurately. In this dissertation, we will analyze if variational quantum algorithms (VQAs) – sometimes called quantum neural networks (QNNs) (LIU *et al.*, 2023) – are a feasible approach to this problem.

VQAs are algorithms based on quantum bits, also known as qubits. Analogous to

bits, qubits are the fundamental units of information in quantum computing (NIELSEN; CHUANG, 2002). Unlike bits, however, qubits have unique properties – which we will discuss throughout this dissertation – that allow them to perform some tasks in fewer steps than their classical counterparts. This phenomenon is known as "quantum advantage" (DEUTSCH; JOZSA, 1992).

David Deutsch and Richard Jozsa formally described the quantum advantage for the first time in 1992 (DEUTSCH; JOZSA, 1992). Their algorithm showed how to classify a specific set of binary functions with just one evaluation. From then on, researchers demonstrated the quantum speed up in many situations (BENNETT *et al.*, 1997; SIMON, 1997; SESHADREESAN *et al.*, 2015), including the famous Shor's algorithm, which factors integers in polynomial time (SHOR, 1999). Since most of our current cryptography relies on the intrinsic difficulty of factoring large numbers, Shor's algorithm might threaten data privacy when large-scale quantum computers are available in the future (GERJUOY, 2005).

It is worth noticing, however, that the current quantum hardware's capabilities are still orders of magnitude below the minimum requirement (millions of qubits) for factoring large numbers via Shor's algorithm (SAFFMAN, 2019). Today's quantum computers are usually called Noisy Intermediate-Scale Quantum (NISQ) devices (PRESKILL, 2018).

The "intermediate-scale" component of NISQ devices refers to their number of qubits, which ranges from fifty to a few hundred (PRESKILL, 2018). Even though it is far away from the million-qubit milestone, this scale is already an accomplishment, given that a hundred-qubit device cannot be simulated even by the most powerful classical computers (BOIXO *et al.*, 2018). NISQ's "noisy" characteristic means that we have limited control over qubits, resulting in a cumulative noise that severely affects information processing (PRESKILL, 2018). One can mitigate this problem via quantum error correction (QEC) techniques (CHIAVERINI *et al.*, 2004).

As for solving differential equations via quantum algorithms, there are a couple of ways to address this challenge, such as the algorithm based on the truncated Taylor series (XIN *et al.*, 2020) and the HHL algorithm (HARROW *et al.*, 2009). Linear differential equations (LDEs) are easier to solve, considering quantum mechanics' intrinsic linear formulation (FEYNMAN *et al.*, 1965). Therefore, there are some options to map linear differential equations into quantum systems that can be simulated in a quantum computer (MONTANARO; PALLISTER, 2016).

Conversely, solving nonlinear differential equations (NLDEs) is not as straightforward. Although it is possible to linearize NLDEs into LDEs and then apply previously discussed techniques, this process would be either computationally expensive or prone to substantial approximation errors. Recent results (KILLORAN *et al.*, 2019a; KYRIIENKO *et al.*, 2021) have demonstrated that VQAs are a promising approach to this question.

Based on the same principles of neural networks (NNs), VQAs can be applied to many problems, including function fitting, image generation, and data classification (KILLORAN *et al.*, 2019a). In this case, the NNs' hyperparameters correspond to angles that define the quantum gates of the variational quantum circuit (VQC). Therefore, when such angles change, the circuit's output also changes. We then vary the angles (arguments) in order to minimize the correspondent cost function, resulting in a more accurate solution (LUBASCH *et al.*, 2020; CEREZO *et al.*, 2021a). The variational aspect of such algorithms is obtained via a classical optimizer – thus, characterizing a hybrid method – which iteratively proposes new values for the angles.

One of the most famous variational algorithms is the Variational Quantum Eigensolver (VQE), which aims to find the approximate ground state and the corresponding wave function of a given quantum system (PERUZZO *et al.*, 2014). In this case, we encode the Hamiltonian of the system in a parametrized quantum circuit. The measurement of this circuit is the expectation value of the energy. Therefore, as we wish to determine the ground state, *i.e.*, the one with the lowest energy, we use a classical optimizer to update the circuit parameters to reduce the value of its measurement (TILLY *et al.*, 2022).

On that same note, there's also the quantum annealing: an optimization algorithm designed for finding the lowest energy configuration of a system (MORITA; NISHIMORI, 2008). Analogously to the VQE, we encode the problem via the system's Hamiltonian, which depends on several parameters (*e.g.*, the strength of interaction between qubits and an external magnetic field). We then evolve such parameters adiabatically and measure the qubits to find the most likely solution corresponding to the lowest energy state (SANTORO; TOSATTI, 2006). More than a method with many applications in quantum chemistry and materials science, quantum annealing is also a promising quantum computing architecture being developed by some research groups and companies (YARKONI *et al.*, 2022).

Considering all these discussions, it is evident that there is room for improvement in the simulation of a complete satellite's orbital model, *i.e.*, a system with all the orbit perturbations. The computational requirements could be in the orders of hundreds of terabytes of memory and quadrillions of floating-point operations per second (SAINI *et al.*, 2012). Even though multiple tools (NICHOLSON *et al.*, 2010) were developed in the last decades that allow us to perform orbit propagation more effectively (*e.g.*, Orekit, NASA's GMAT and SPICE, Analytical Graphics' STK, FreeFlyer), a complete satellite's orbital model remains a highly challenging problem that would benefit from new approaches.

To start addressing this question, we propose a proof-of-concept based on VQCs. Specifically, the problem we want to solve is how to use VQCs for solving the nonlinear differential equations of the orbital motion in the presence of gravitational perturbations and atmospheric drag. The objective is to determine if a specific circuit architecture combined with a given classical optimizer (Adam) can satisfactorily solve a particular set

of nonlinear differential equations – those that describe orbits with atmospheric drag and gravitational perturbations.

To achieve this goal, we start the second chapter presenting a literature review of the relevant subjects. We review the fundamentals of quantum computing (*e.g.*, quantum gates, different qubit representations) and discuss previous works on variational quantum algorithms. The chapter ends with a brief discussion on barren plateaus and how they affect the performance of VQAs.

Chapter three explains the method we used to solve the NLDEs. It contains all details about the circuit's parameters, cost functions, and precisely which equations and conditions were analyzed. We start with a simplified two-dimensional model and then expand the idea to the original orbit propagation problem in three dimensions. The fourth chapter presents the obtained results, including a comparison with classical numerical methods. All implementations were performed in quantum simulators executed locally in a classical computer.

Finally, the last chapter contains the conclusions of this work, including the current limitations and opportunities. We also comment on possible next steps that could be taken to expand the capabilities of our method. In addition to that, Appendix A derives and presents all the theoretical reasoning behind the nonlinear differential equations we solve in this dissertation. Moreover, Appendix B has all the relevant Python code we used throughout this work.

2 Literature Review

This chapter presents the necessary theoretical background to solve the terminal rendezvous problem with orbit perturbations via VQCs. Firstly, we introduce the foundations of quantum computing and perform a thorough review of works on VQAs developed up until this moment. We then discuss barren plateaus, an important phenomenon that affects VQAs' performance and scalability.

2.1 Quantum Computing

Before exploring the previous works on VQAs, we must review some introductory quantum computing concepts. The bit can be defined as the basic unit of information in classical computation. According to the probability theory (AARONSON, 2013), we can describe the bit as an object with a probability p of being 0 and $1 - p$ of being 1. This happens because such a theory establishes that the norm of the probability vector $\{p_0, p_1\}$ must be one.

Therefore, if an event has n possible outcomes, we can write the following expression regarding the probabilities:

$$\sum_{i=1}^{n} p_i = 1, \tag{2.1}$$

where p_i is the probability of the i-th outcome ($p_i \geq 0$) and $0 \leq i \leq n$.

Equation 2.1 is valid because the probability theory is based on the 1-norm, *i.e.*, the sum of the absolute values of the components of a given vector. From this core idea, we can deduce the mathematical formulation of the qubit (AARONSON, 2013) – also known as the quantum bit.

To achieve this goal, we must consider the Euclidean norm (2-norm), E_N, which is defined as

$$E_N = \sqrt{v_1^2 + v_2^2 + ... + v_n^2} \tag{2.2}$$

for a given vector $\vec{v} = \{v_1, v_2, ..., v_n\}$.

Since we are still describing a bit – now in the Euclidean norm – the probability vector

$\{p_0, p_1\}$ must have a unitary norm. This means that we must have two squares that add up to one. By taking $p_0 = |\alpha|^2$ and $p_1 = |\beta|^2$, we have that

$$|\alpha|^2 + |\beta|^2 = 1, \tag{2.3}$$

where α and β can be positive, negative, or even complex numbers.

In quantum mechanics, such values are the amplitudes of the associated outcomes 0 and 1 (BAYM, 2018). On that note, one can represent any qubit via the Dirac (bra-ket) notation ($|.\rangle$) (KASIRAJAN, 2021), such that

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle, \tag{2.4}$$

where $|\psi\rangle$ is a generic qubit written in the $\{|0\rangle, |1\rangle\}$ basis. Each state $|\psi\rangle$ is also called a "statevector", given its vectorial representation. For instance, $|0\rangle$ and $|1\rangle$ are given by:

$$\begin{cases} |0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \\ |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \end{cases} . \tag{2.5}$$

For specific purposes (*e.g.*, geometric representation), it is helpful to rewrite Equation 2.4 in terms of real numbers. In this case, one may write that

$$|\psi\rangle = \cos\frac{\theta}{2}|0\rangle + e^{i\phi}\sin\frac{\theta}{2}|1\rangle, \tag{2.6}$$

where $\theta \in [0, 2\pi]$ and $\phi \in [0, 2\pi]$. Therefore, when θ and ϕ vary in such intervals, we obtain the qubit's geometric locus (Bloch sphere) illustrated by Figure 2.1.

From Equations 2.4 and 2.6, one can notice that the relative phase between α and β (also known as the local phase) determines the probabilities of different measurement outcomes. Thus, if we apply a global phase ζ such that

$$|\psi'\rangle = e^{i\zeta}|\psi\rangle, \tag{2.7}$$

it doesn't affect any observable quantities because it changes the state uniformly. For that reason, we say the global phase is irrelevant in the qubit representation, *i.e.*, qubits that differ only by a global phase represent the same quantum state.

The Bloch sphere allows us to see the qubit $|\psi\rangle$ as a normalized vector (unitary norm) in a complex Hilbert space that we can express in terms of any pair of orthonormal vectors – also known as basis (NIELSEN; CHUANG, 2002). In addition to $\{|0\rangle, |1\rangle\}$, we also

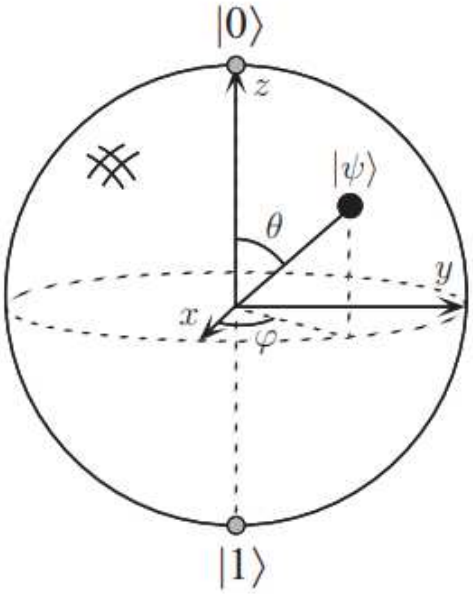

FIGURE 2.1 – Bloch sphere (NIELSEN; CHUANG, 2002).

commonly use other bases, such as $\{|+\rangle, |-\rangle\}$ and $\{|i+\rangle, |i-\rangle\}$, given by:

$$\begin{cases} |+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \\ |-\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \end{cases}, \tag{2.8}$$

$$\begin{cases} |i+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + i|1\rangle) \\ |i-\rangle = \frac{1}{\sqrt{2}}(|0\rangle - i|1\rangle) \end{cases}. \tag{2.9}$$

Once we have established the qubit's definition and main representations, we can understand how to encode and process information with qubits. Analogously to bits, we also start with an initial state and apply several logical gates to get to the next desired state (AARONSON, 2013). For instance, when one applies the NOT gate to a bit, it flips from 0 to 1 and vice-versa. The same rule is valid for qubits, *i.e.*, the NOT quantum gate (X) acts such that:

$$X|0\rangle = |1\rangle, \tag{2.10}$$

where, in this specific case, the NOT quantum gate X can be written as a 2×2 matrix

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}. \tag{2.11}$$

Even though there are multiple use cases for qubit architectures (MARTIN-LOPEZ *et al.*, 2012; MOLL *et al.*, 2016), certain aspects make it difficult to employ them for encoding neural networks. The following subsection discusses such elements and introduces a new

framework that better fits the neural networks model.

2.2 Variational Quantum Algorithms

Recent results have demonstrated that we can achieve a quantum advantage in some machine learning algorithms (BIAMONTE *et al.*, 2017), including those for data fitting (WIEBE *et al.*, 2012), Bayesian interference (WIEBE; GRANADE, 2015), and recommendation systems (KERENIDIS; PRAKASH, 2016), for instance. Among these algorithms, we are particularly interested in a set of techniques based on NNs, forming a new field called deep learning (LECUN *et al.*, 2015).

Broadly speaking, one of the main components that leverage deep learning methods' capabilities is the fact that its fundamental computational units are continuous vectors and tensors – rather than conventional bit registers in digital computing (KILLORAN *et al.*, 2019a). This alternative framework allows deep learning algorithms to improve state-of-the-art performance in multiple tasks, such as speech recognition (KAMATH *et al.*, 2019) and object detection (ZHAO *et al.*, 2019).

Although deep learning's continuous computation has been traditionally implemented in digital computers (LECUN, 2019), recent studies are frequently proposing new specialized hardware which is fundamentally analog (HAENSCH *et al.*, 2018). On that note, quantum hardware's unique characteristics (*e.g.*, superposition, entanglement, interference) might also provide an advantage over digitally implemented NNs (KYRIIENKO *et al.*, 2021).

However, analogously to classical bit registers, digital quantum hardware (PARRA-RODRIGUEZ *et al.*, 2020) relies on qubits, which encode information in a discrete way (KILLORAN *et al.*, 2019a). This happens because each qubit measurement results in one of two possible orthonormal states. Therefore, the qubit-based circuits we normally observe in digital quantum computers may not be the best choice for solving continuous-valued problems (BENEDETTI *et al.*, 2018).

To address such problems, we shall focus on the continuous-variable (CV) framework (KILLORAN *et al.*, 2019a). As quantum information is fundamentally encoded in quantum states of fields (*e.g.*, electromagnetic fields), it should be natural to use such states – instead of qubits – in quantum computing. This approach became progressively popular and successful in recent years, as illustrated by many publications on photonic quantum hardware (SLUSSARENKO; PRYDE, 2019; TAKEDA; FURUSAWA, 2019).

Summarily, the diffence between digital and CV quantum computing lies in the way we encode and manipulate information using quantum systems (KENDON *et al.*, 2010). As

we have discussed, digital quantum computing is based on discrete two-level qubits and can be physically implemented via superconducting circuits, trapped ions, and topological systems (MOUNT *et al.*, 2016). On the other hand, CV quantum computing uses quantum systems with continuous variables that describe quantities (*e.g.*, position and momentum) of physical systems (GU *et al.*, 2009). Photonic systems are currently the most promising hardware approach towards practical CV quantum computing (BARTLETT; SANDERS, 2002; WU *et al.*, 2020).

This dissertation uses the CV model to build VQCs, also known as quantum neural networks (QNNs). To understand how this mapping works and the CV model's advantage over qubit-based digital quantum computing, we shall first introduce the fundamentals of NNs. One should keep in mind that this is quite a broad and expanding field; thus, the following overview's purpose is to provide the necessary tools to comprehend the VQCs we will use.

While we had multiple advances (REN *et al.*, 2016; NIU *et al.*, 2021) in deep learning in the past few years, its fundamental structure is still based on feedforward NNs – also known as multilayer perceptron (SVOZIL *et al.*, 1997). Such networks consist of the collection of multiple layers (which depend on the parameter set Θ) $L_1, ..., L_N$ that map an input $\vec{x} \in \mathbb{R}^n$ to an output $\vec{y} \in \mathbb{R}^n$. Mathematically,

$$\vec{y} = f_\theta(\vec{x}) = L_N \circ ... \circ L_1(\vec{x}), \tag{2.12}$$

where the operation $\circ$ indicates that the output of a given layer is used as input for the next one. In fact, the "deep" aspect of deep learning refers to the act of stacking several layers together (KILLORAN *et al.*, 2019a).

Each layer L in Equation 2.12 can be described as

$$L(\vec{x}) = \phi_f(W\vec{x} + \vec{b}), \tag{2.13}$$

where ϕ_f is the nonlinear activation function, $W \in \mathbb{R}^{m \times n}$ is the weight matrix, and $\vec{b} \in \mathbb{R}^m$ is the bias vector. The free parameters θ_W and θ_b define W and $\vec{b}$, respectively – demonstrating how each layer depends on the parameter set.

Ultimately, the goal is to use the neural network to solve a computational problem that has an answer $\vec{a} \in \mathbb{R}^n$. To achieve this objective, we need to adjust the parameter set Θ such that the output $\vec{y}$ gets progressively closer to $\vec{a}$. One measures the difference between $\vec{y}$ and $\vec{a}$ via the loss function $\mathcal{L}$ – its mathematical definition depends on the nature of the problem (WANG *et al.*, 2020).

Considering differential equations in the format:

$$\frac{d\vec{x}}{dt} = g(\vec{x}, t), \tag{2.14}$$

where g is a generic function, the NN's loss function is defined as

$$\mathcal{L} = \mathcal{L}_D + \mathcal{L}_I, \tag{2.15}$$

where $\mathcal{L}_D$ is the loss function relative to the differential equation itself and $\mathcal{L}_I$ is the loss function corresponding to the equation's initial conditions.

We then proceed to define such loss functions as (LECUN *et al.*, 2015)

$$\mathcal{L}_D = \sum_{i=1}^{N} (\vec{a}(x_i, t) - \vec{y}(x_i, t))^2, \tag{2.16}$$

$$\mathcal{L}_I = (\vec{a}(\vec{x}, 0) - \vec{y}(\vec{x}, 0))^2, \tag{2.17}$$

where $\vec{a}(x_i, t)$ and $\vec{y}(x_i, t)$ correspond to the solution and the NN's output at a given instant t and position x_i. It is worth noting that we have considered the space discretization of N slots $(x_1, x_2, ..., x_N)$. As for the initial conditions' loss function $\mathcal{L}_I$, we simply compute the squared difference between the initial condition and the NN's output at the initial instant – in this case, $t = 0$.

Each parameter set (composed of θ_b and θ_W) determines the values of W and b, which also defines the NN's output. The loss function's value depends directly on the parameter set. Hence, one must choose the optimal parameters to minimize the loss function. We can do this optimization process in multiple ways. One of the most famous approaches is the stochastic gradient descent algorithm (BOTTOU *et al.*, 1991), which proposes the following iteration routine:

$$\begin{cases} W^j_{new} = W^j - \alpha \frac{d\mathcal{L}}{dW^j} \\ b^j_{new} = b^j - \alpha \frac{d\mathcal{L}}{db^j} \end{cases}, \tag{2.18}$$

where W^j_{new} and b^j_{new} are, respectively, the new weight matrix and bias vector in the $j-$th layer, and α is an exponential decay factor that determines the contribution of the current and previous gradients (BOTTOU, 2012).

In this dissertation, we will primarily use the Adaptive Moment Estimation (Adam) optimizer (KINGMA; BA, 2014) already implemented in Python's Numpy library. Adam is a state-of-the-art optimization method that many deep learning researchers are widely using (ZHANG, 2018). Fundamentally, Adam is a gradient descent algorithm with momentum, *i.e.*, it accelerates the gradient descent process by calculating an exponentially weighted average of the gradients (KINGMA; BA, 2014) at each iteration.

To run the algorithm, one must define a stepsize α, exponential decay rates β_1 and β_2 for the moment estimates, the initial parameter set Θ_0, the loss function $f(\theta)$, and initialize the first and second moment estimates as $m_0 = 0$ and $v_0 = 0$, respectively. Then, for each timestep t, we compute the parameter set Θ_t (until it converges within a given interval), which is composed by parameters θ_t, and also calculate the quantities β_1^t and β_2^t, which are simply the exponentiation of β_1 and β_2 to the power of t. We use the following routine (KINGMA; BA, 2014):

$$\begin{cases} t \leftarrow t + 1, \\ g_t \leftarrow \nabla_\theta f_t(\theta_{t-1}), \\ m_t \leftarrow \beta_1 m_{t-1} + (1 - \beta_1) g_t, \\ v_t \leftarrow \beta_2 v_{t-1} + (1 - \beta_2) g_t^2, \\ \hat{m}_t \leftarrow m_t / (1 - \beta_1^t), \\ \hat{v}_t \leftarrow v_t / (1 - \beta_2^t), \\ \theta_t \leftarrow \theta_{t-1} - \alpha \hat{m}_t / (\sqrt{\hat{v}_t} + \epsilon), \end{cases} \tag{2.19}$$

where g_t is the gradient at a given timestep t, $\hat{m}_t$ and $\hat{v}_t$ are, respectively, the bias-corrected first and second moment estimate, and ϵ is a small positive constant. In most deep learning problems, we tend to take values close to $\alpha = 0.001$, $\epsilon = 10^{-8}$, $\beta_1 = 0.9$, and $\beta_2 = 0.999$ (ZHANG, 2018). We chose such values not only because of their frequent use in the literature, but also due to the fact that they represent a good balance between speed and stability, *i.e.*, our model won't take too many iterations to converge and it also won't skip promising configurations (local minima) when exploring the parameter space.

Once we have covered the basics of NNs and optimization methods, we shall discuss the CV approach to quantum computing and how we use it for creating VQCs that solve differential equations. From the hardware perspective, although it is not the traditional qubit-based digital quantum computing, there are many approaches to realize the CV model, including optical systems (ANDERSEN *et al.*, 2015) and ion traps (MEEKHOF *et al.*, 1996).

The CV model does not rely on qubits; instead, it uses quantum states of bosonic modes (MICHAEL *et al.*, 2016) – also known as qumodes – to store and process information. Roughly speaking, the quantum mechanical definition of mode (FEYNMAN *et al.*, 1965) comes from the Fourier transform expansion of a free quantum field $\phi(x)$. Said expansion contains the creation and annihilation operators we apply to the particle associated with the field. If these operators commute, we have a boson – and the operatores correspond to the bosonic modes. If they anti-commute, we have the ferminon as a particle and the operators as fermionic modes.

We will represent the qumode using the phase-space representation of quantum mechanics (GROENEWOLD; GROENEWOLD, 1946). In this formulation, we write the state of a qumode as a vector $(x, p) \in \mathbb{R}^2$, where x and p are, respectively, the position and momentum of the corresponding particle (PFISTER, 2019). A system with N qumodes is represented via $(\vec{x}, \vec{p}) \in \mathbb{R}^{2N}$.

Just like qubits, one also applies quantum gates to change the states of qumodes. Regarding qumodes, however, there is an important classification of quantum gates: they are either Gaussian or non-Gaussian (KILLORAN *et al.*, 2019a). Gaussian gates are generally easier to implement in CV quantum computers (LARSEN *et al.*, 2021). Mathematically, a Gaussian transformation is described as

$$\begin{pmatrix} \vec{x} \\ \vec{p} \end{pmatrix} \to M \begin{pmatrix} \vec{x} \\ \vec{p} \end{pmatrix} + \begin{pmatrix} \alpha_r \\ \alpha_i \end{pmatrix}, \tag{2.20}$$

in which M is a symplectic matrix, and α_r and α_i are the real and imaginary parts of a complex displacement vector $\vec{\alpha} \in \mathbb{C}^N$, respectively. Symplectic matrices (MACKEY; MACKEY, 2003) are those that satisfy the relation

$$M^T \Omega M = \Omega, \tag{2.21}$$

where

$$\Omega = \begin{pmatrix} 0 & \mathbb{I} \\ -\mathbb{I} & 0 \end{pmatrix}. \tag{2.22}$$

We can apply many Gaussian transformations to change the states of qumodes. Throughout this work, we will predominantly use the Displacement $D(\alpha)$, Rotation $R(\phi)$, and Squeeze $S(r)$ for single-mode operations. They are expressed by:

$$D(\alpha) \begin{pmatrix} x \\ p \end{pmatrix} \to \begin{pmatrix} x + \mathrm{Re}(\alpha) \\ p + \mathrm{Im}(\alpha) \end{pmatrix}, \tag{2.23}$$

$$R(\phi) \begin{pmatrix} x \\ p \end{pmatrix} \to \begin{pmatrix} \cos\phi & \sin\phi \\ -\sin\phi & \cos\phi \end{pmatrix} \begin{pmatrix} x \\ p \end{pmatrix}, \tag{2.24}$$

$$S(r) \begin{pmatrix} x \\ p \end{pmatrix} \to \begin{pmatrix} e^{-r} & 0 \\ 0 & e^{r} \end{pmatrix} \begin{pmatrix} x \\ p \end{pmatrix}, \tag{2.25}$$

where $\alpha \in \mathbb{C}$, $\phi \in [0, 2\pi]$, and $r \in \mathbb{R}$.

We will also use a two-mode Gaussian gate known as Beamsplitter ($BS(\theta), \theta \in [0, 2\pi]$), which we can perceive as a rotation between two qumodes. To simplify the notation, we

shall call it $B(\theta)$. Mathematically, we have:

$$B(\theta)\begin{pmatrix} x_1 \\ x_2 \\ p_1 \\ p_2 \end{pmatrix} \rightarrow \begin{pmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & \cos\theta & -\sin\theta \\ 0 & 0 & \sin\theta & \cos\theta \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \\ p_1 \\ p_2 \end{pmatrix}. \tag{2.26}$$

Additionally, we commonly use non-Gaussian gates when creating VQCs. In this dissertation, we will use the Kerr gate ($K(\phi)$), given by

$$K(\phi)\begin{pmatrix} \hat{x} \\ \hat{p} \end{pmatrix} \rightarrow \begin{pmatrix} \hat{x}\cosh\phi - i\hat{p}\sinh\phi \\ \hat{p}\cosh\phi + i\hat{x}\sinh\phi \end{pmatrix}, \tag{2.27}$$

in which the operators $\hat{x}$ and $\hat{p}$ are given respectively by

$$\hat{x} = \int_{-\infty}^{\infty} x|x\rangle\langle x|dx, \tag{2.28}$$

$$\hat{p} = \int_{-\infty}^{\infty} p|p\rangle\langle p|dp. \tag{2.29}$$

Once we have defined the quantum gates we will use in VQCs, we can understand their connection with classical NNs. Comparing Equations 2.13 and 2.20, one deduces that the symplectic matrix M acts like the weight matrix W and the displacement vector $\vec{\alpha}$ corresponds to the bias vector $\vec{b}$. Analogously, the weight and bias parameters are equivalent to the gates' arguments. Therefore, the goal is to optimize the parameter set $\Theta_j = \{\theta_{1j}, \theta_{2j}, ..., \theta_{Nj}\}$ – for each j layer containing N gates – such that it minimizes the loss function described by Equations 2.15 to 2.17.

It is possible to demonstrate the universality of VQCs (KILLORAN *et al.*, 2019a), *i.e.*, we can parametrize every transformation using a combination of gates such as:

$$L = \mathcal{U}_1 \circ \mathcal{S} \circ \mathcal{U}_2 \circ \mathcal{D} \circ \Phi, \tag{2.30}$$

where L is a layer, $\mathcal{U}_1$ and $\mathcal{U}_2$ are generic N-modes transformations containing rotation or beamsplitter transformations, $\mathcal{S}$ and $\mathcal{D}$ are the collective squeeze and displacement operators. Additionally, Φ is the combination of some non-Gaussian gates – they could be the cubic gates or Kerr, for instance.

Since we are dealing with N qumodes, the mathematical representation of those col-

lective gates is

$$\begin{cases} \mathcal{U}_i = \mathcal{U}_i(\vec{\theta}, \vec{\phi}) \\ \mathcal{D} = D(\alpha_1) \otimes ... \otimes D(\alpha_N) \\ \mathcal{S} = S(r_1) \otimes ... \otimes S(r_N) \\ \Phi = \phi(\lambda_1) \otimes ... \otimes \phi(\lambda_N) \end{cases}, \tag{2.31}$$

in which $\vec{\theta}$ and $\vec{\phi}$ are vectors of arguments and $\otimes$ represents the tensor product. Figure 2.2 illustrates the graphical representation of such VQC's layer. Combining multiple layers – followed by measurement at the end – constitutes the VQC itself. The VQC's output is the result of such a measurement.

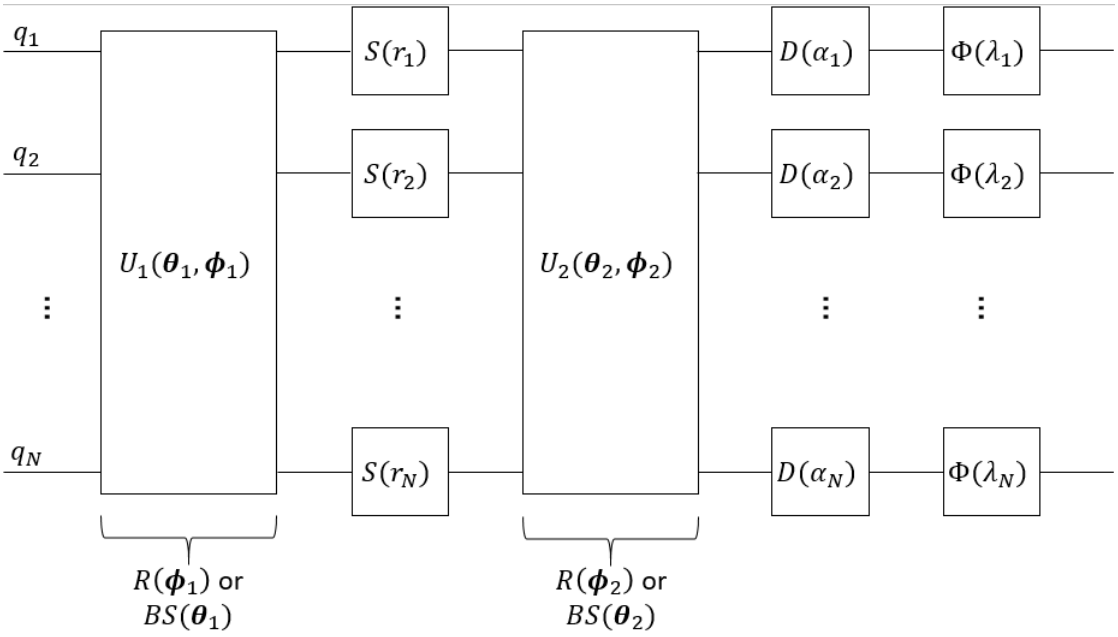

FIGURE 2.2 – General representation of each layer of the VQC (KILLORAN *et al.*, 2019a), composed of N input qumodes $(q_1, ..., q_N)$, Gaussian gates (U_1, U_2, S, D), and non-Gaussian gates (Φ).

Considering the description of VQCs, one can understand why they are also called "hybrid approaches" (CEREZO *et al.*, 2021a). They are made of classical (gradient optimizer) and quantum (circuit) components that work together to solve a specific problem. Figure 2.3 is a general scheme of how VQCs operate: given a parameter set Θ that acts as the VQC's input, the circuit's output feeds a loss function that computes how distant is the output compared to a given reference. Considering this difference between the VQC's output and the reference, a classical optimizer updates the parameter set Θ aiming to minimize the loss function, which translates to a more accurate model.

As the quantum analog of NNs – highly successful in classical machine learning – VQAs have emerged as promising methods to be implemented in today's NISQ devices (CEREZO *et al.*, 2021a). Additionally, VQAs' flexible framework and universality allow them to be

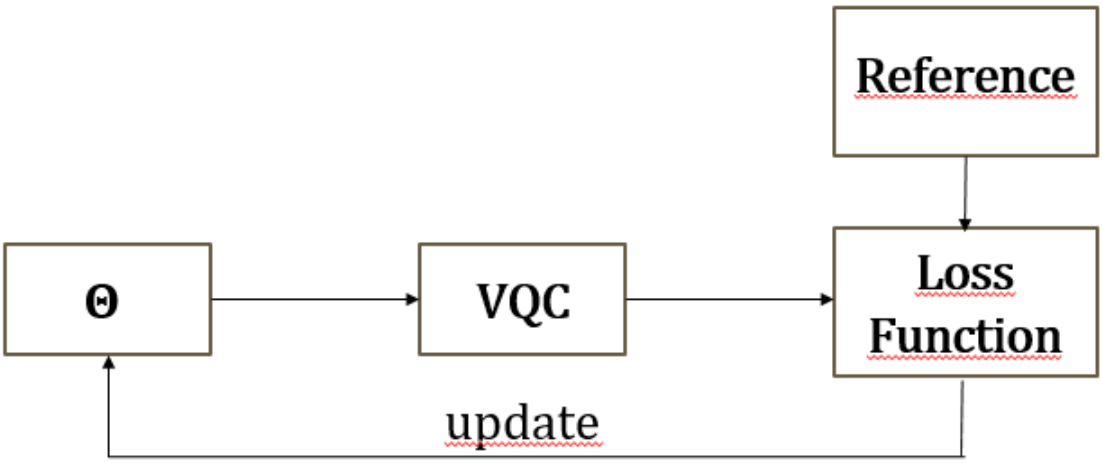

FIGURE 2.3 – General representation of the VQC containing a parameter set Θ as input, a loss function that compares its output to a given reference, and a classical optimizer that updates Θ at each iteration.

used in various problems, including (but not limited to) dynamical simulations (YAO *et al.*, 2021), combinatorial optimization (AMARO *et al.*, 2022), and quantum metrology (KOCZOR *et al.*, 2020).

Even though such VQAs' characteristics are undeniably positive, some challenges must be addressed to leverage VQAs' applications. In this work, we are primarily interested in barren plateaus (BPs), their impact on VQAs' performance, and how to mitigate their effects.

2.3 Barren Plateaus

As previously discussed, we aim to vary the gates' parameters via Adam optimizer to minimize the loss function $\mathcal{L}$. Although this is a valid method, it is well-known that the proposed VQC architecture is limited in terms of the reduction of $\mathcal{L}$ (CEREZO *et al.*, 2021b). The process eventually reaches a "barren plateau" (MCCLEAN *et al.*, 2018), in which the loss function does not decrease substantially even if we update the parameter set in many more iterations.

Formally, the barren plateau is a phenomenon where the optimizer's gradients vanish exponentially as a function of the circuit's depth (CEREZO *et al.*, 2021b). The depth of the circuit (ROMERO *et al.*, 2017) depends directly on its number of layers. Thus, a shallow circuit (few layers) will tend to have its loss function's landscape filled with peaks and valleys. Conversely, circuits with more layers – also known as deep circuits – will have landscapes progressively flatter (WANG *et al.*, 2021).

Such problems directly result from how we define the loss function in each situation,

which may be challenging (CEREZO *et al.*, 2021b). Some cases, such as the Variational Quantum Eigensolver (VQE), are relatively straightforward. VQE aims to find the ground state of a given Hamiltonian H (WANG *et al.*, 2019). Therefore, we write it as a combination of Pauli operators σ_k:

$$H = \sum_k a_k \sigma_k, \tag{2.32}$$

where a_k are complex coefficients. Since we're trying to determine the Hamiltonian's ground state, the loss function $\mathcal{L}$ at a given trial state $|\psi\rangle$ corresponds directly to:

$$\mathcal{L} = \langle\psi|H|\psi\rangle, \tag{2.33}$$

which is equivalent to the energy of the system for a given Hamiltonian. Therefore, minimizing the value of this expression implies determining the Hamiltonian's ground state.

The VQE, however, is the most basic example of VQA (WANG *et al.*, 2019). This means that other VQA's use cases are often associated with less obvious loss functions. Applications like quantum error correction (XU *et al.*, 2021), quantum metrology (KOCZOR *et al.*, 2020), solving linear systems (PAN *et al.*, 2014), and differential equations (LUBASCH *et al.*, 2020) are not directly connected to a physical Hamiltonian – making it harder to propose an appropriate loss function.

There are a few ways to mitigate the BP effects when training our VQCs. One approach is to use structured initial guesses for the parameters (MCCLEAN *et al.*, 2018) – rather than randomly assigned values – based on some characteristics of the problem. Another possible idea is to perform classical training beforehand and then use this setting's learning as input to the VQC (BENGIO *et al.*, 2006). A recent method proposes local loss functions, *i.e.*, based on some observables of the circuit as opposed to the measurement of all qumodes (CEREZO *et al.*, 2021b).

Although not the main interest of this dissertation, it is crucial to acknowledge the existence of barren plateaus and analyze their impact on the performance of our models. In particular, we apply some discussed techniques (*e.g.*, structured initial guesses and local loss functions) to our loss functions, aiming to mitigate the BP effects. Just like they substantially contributed to the "winter of deep neural networks" (HOCHREITER *et al.*, 2001), BPs might affect the overall success and adoption of VQCs if we do not address this issue adequately.

3 Orbit Propagation with Perturbations via Variational Quantum Algorithm

The present chapter discusses the specific problems and their respective solutions proposed by this dissertation. We start by explaining the simplified two-dimensional (2D) version of the orbit propagation problem considering the effects of the J_2 term and how to solve it via VQA. We then apply the same principle to the three-dimensional (3D) orbit propagation considering the J_2 term and atmospheric drag individually. The J_2 term is the Earth's second zonal harmonic, and it is used to describe the variation of the gravitational potential due to the planet's oblateness. Both phenomena (gravitational perturbation and atmospheric drag) are explained in greater detail in Appendix A. We also describe a numerical solution based on SciPy's ODEINT integrator. Finally, we discuss alternative ways for solving such problems. Appendix A contains the demonstrations of every NLDE and Appendix B has all the Python codes referring to the VQA and numerical implementation.

3.1 2D Case with Gravitational Perturbation

Before performing the 3D orbit propagation in the presence of atmospheric drag (Equation A.14) or gravitational perturbation (Equation A.27), it is convenient to analyze a simplified version of such problems. Thus, if we are able to model and solve this simpler case, we can extrapolate the same ideas to the original problem.

As a simplified version of the orbit propagation, we adopt the 2D case of the gravitational perturbation problem. We formulate this situation by considering a satellite confined to the XY-plane while orbiting the Earth. Mathematically, this situation is expressed in the geocentric equatorial frame (CURTIS, 2013) by:

$$\ddot{x} = -\mu \frac{x}{(x^2+y^2)^{3/2}} - \frac{3}{2} J_2 \mu R^2 \frac{x}{(x^2+y^2)^{5/2}}, \tag{3.1}$$

$$\ddot{y} = -\mu \frac{y}{(x^2+y^2)^{3/2}} - \frac{3}{2} J_2 \mu R^2 \frac{y}{(x^2+y^2)^{5/2}}, \tag{3.2}$$

where μ is the standard gravitational parameter, R is the Earth's equatorial radius, and J_2 is the Earth's second zonal harmonic. We derive these expressions in Appendix A, culminating in Equation A.27 (with the assumption that $z = \dot{z} = \ddot{z} = 0$).

To solve the system defined by Equations 3.1 and 3.2, one should consider particular characteristics of the problem, hoping to simplify the solution via VQA. We shall apply the *Ansatz* method (LINKS *et al.*, 2003) – widely used in many problems in Physics and Mathematics.

Derived from the German language, *ansatz* can be defined as an educated guess or assumption we make to solve a problem – such a guess must be verified afterwards (FOCK, 1930). In Mathematics, a famous example of the *Ansatz* method is assuming an exponential function as the solution of a homogeneous linear differential equation (GUNER; BEKIR, 2018). Another renowned case of the *Ansatz* approach is the Ritz method ("trial wavefunction"), which proposes a wavefunction as close as possible to the actual solution (MÜLLER; ZEINHOFER, 2022) of boundary value problems.

Since we are dealing with closed orbits, our *ansatz* is that their projections onto each axis are periodic functions of time. Therefore, we can write that:

$$x(t) = A(t)\cos(\omega t + \alpha), \tag{3.3}$$

$$y(t) = B(t)\cos(\omega t + \beta), \tag{3.4}$$

where $A(t)$ and $B(t)$ are time-dependent amplitudes, ω is the angular frequency of the satellite, α and β are initial phases.

Using this *Ansatz*, our problem is translated into finding the coefficients $A(t), B(t)$ and the initial phases α, β via VQA. The present formulation substantially reduces the possibilities of solutions – which should, ideally, lead to an accurate VQA output in fewer iterations.

The next step is to create the VQC to encode such variables and solve the 2D problem. First of all, we will provide a high-level overview of the circuit followed by detailed discussions of its layer and loss function. We use the standard procedure proposed by the literature (KILLORAN *et al.*, 2019a), *i.e.*, each variable is encoded into a single qumode (wire) – as illustrated by Figure 3.1 – which is initialized in the vacuum state.

We then apply a displacement gate D to each wire with random arguments $\alpha_0, \alpha_1, \alpha_2$, and α_3. These transformations make each qumode go from the vacuum to a new quantum state. After that, we apply n layers – which we will define later – and finally measure the qumodes in the position quadrature observable $\hat{x}$ represented by the X symbol. Math-

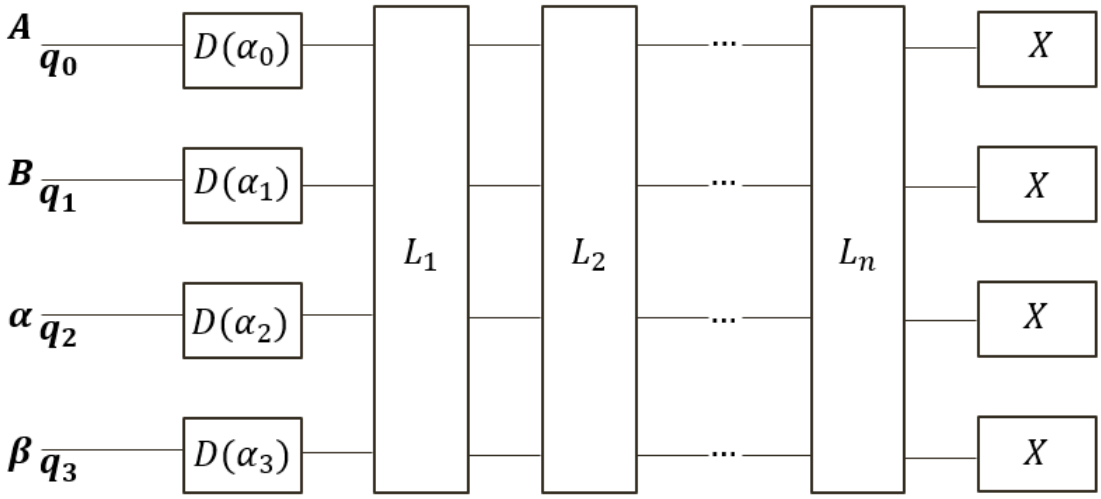

FIGURE 3.1 – VQC used to solve the 2D orbit propagation with gravitational perturbation. It is composed of four qumodes $(q_0, ..., q_3)$, each one associated to a displacement gate D followed by n layers, and a measurement gate X applied to each qumode.

ematically, this measurement corresponds to the mean displacement in the phase space along the x-axis.

One should realize that we start with random arguments for the displacement gates and the layers. At each iteration, the Adam optimizer updates the arguments of all gates, intending to minimize the loss function. The next step involves proposing a layer and defining the loss function that best represents the problem.

There are numerous possibilities for choosing a VQC layer. As discussed in Chapter 2, Equations 2.30 and 2.31 constitute a universal model that can parametrize every transformation. In principle, one could use the layer proposed by Figure 2.2 to propagate the orbit in the presence of any perturbations. However, it is essential to acknowledge that using more quantum gates implies optimizing more parameters. Therefore, the Adam optimizer would likely require more iterations to minimize the loss function successfully.

As a result, we shall use the universal layer from Figure 2.2 as a baseline and remove some unnecessary gates. The idea is to represent the Equations 3.3 and 3.4 using as few gates as possible. As we know, there is a connection between the motions in different axes – clearly expressed by the coupled differential Equations 3.1 and 3.2. Thus, we should have a quantum gate that connects the qumodes encoding corresponding variables, *i.e.*, $A(t)$ with $B(t)$ and α with β. For this purpose, we use a beamsplitter $BS(\theta)$.

Additionally, considering the nonlinear aspect of the coupled differential equations, it is convenient to add a non-Gaussian gate to our layer as well. Following similar layer designs from the literature (KILLORAN *et al.*, 2019a), we employ the Kerr gate to achieve this goal. In principle, we do not need to use a squeezing or a rotation gate once they perform transformations that are not crucial for solving our problem. The strictly necessary gates

are those that: promote a shift from the vacuum state ($D(\alpha)$), connect equivalent variables ($BS(\theta)$), and introduce non-Gaussian behavior ($K(\phi)$).

Taking that into account, we build our VQC with displacement gates at the beginning, followed by layers and the final measurement (Figure 3.1). The layer comprises beamsplitters connecting corresponding qumodes (q_0 with q_1 and q_2 with q_3) and a non-Gaussian gate (Kerr) at each wire, as represented in Figure 3.2.

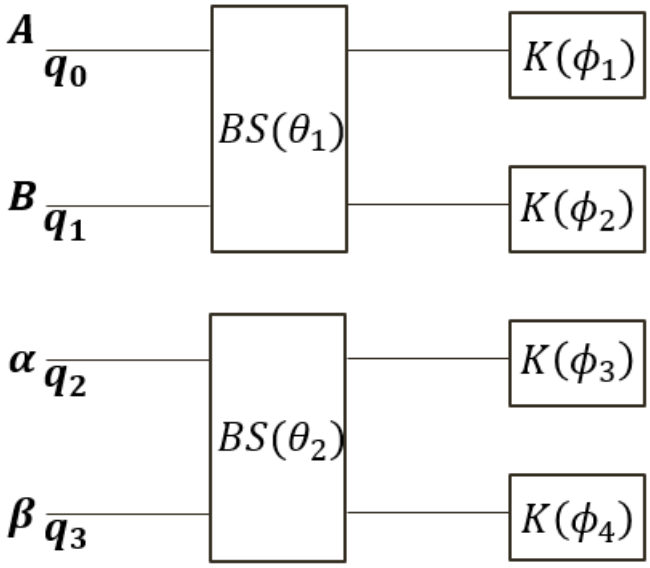

FIGURE 3.2 – Representation of the VQC layer for the 2D orbit propagation with gravitational perturbation. It is composed of four qumodes ($q_0, ..., q_3$), associated in pairs to beamsplitter gates BS, followed by Kerr gates K applied to each qumode.

Once we have defined the layer and the circuit, we proceed to design the loss function $\mathcal{L}$. Using the same principles from Equation 2.15, we consider two components. The first one refers to the initial conditions $\mathcal{L}_I$. On the other hand, the second component ($\mathcal{L}_D$) is about the differential equation itself. We will mathematically define both parts.

Since we need to solve second-order coupled differential equations, $\mathcal{L}_I$ involves differences in each axis' initial position and velocity. The problem has the following satellite's inputs: $x_0, y_0, \dot{x}_0, \dot{y}_0$, and T (orbit's period). To compute the differences, we must first evaluate our *ansatz*'s estimation for initial position and velocity. The initial position is easily obtained by setting $t = 0$ in Equations 3.3 and 3.4. This leads to the loss functions:

$$\begin{cases} \mathcal{L}_{x0} = (x_0 - (A(0)\cos\alpha))^2 \\ \mathcal{L}_{y0} = (y_0 - (B(0)\cos\beta))^2 \end{cases} . \tag{3.5}$$

As for the initial velocities in x and y-axis, one must differentiate the *ansatz*'s $x(t)$ and $y(t)$ (with respect to the time) from Equations 3.3 and 3.4 and set $t = 0$. As an

approximation, if we neglect the derivatives of $A(t)$ and $B(t)$, it follows that

$$\begin{cases} \mathcal{L}_{vx0} = (\dot{x}_0 - (-\omega A(0)\sin\alpha))^2 \\ \mathcal{L}_{vy0} = (\dot{y}_0 - (-\omega B(0)\sin\beta))^2 \end{cases}. \tag{3.6}$$

Even though the accelerations in the x and y-axis are unknown, we can still compute the differences between their predicted value and the *ansatz*'s output. This segment corresponds to the $\mathcal{L}_D$ loss function. To calculate it, we use the problem's definition from Equations 3.1 and 3.2 as a benchmark. We obtain the *ansatz*'s corresponding acceleration by differentiating $x(t)$ and $y(t)$ in time twice, and neglecting the derivatives of $A(t)$ and $B(t)$. Therefore,

$$\mathcal{L}_{ax,2D} = \left[\left(-\mu\frac{x}{(x^2+y^2)^{3/2}} - \frac{3}{2}J_2\mu R^2\frac{x}{(x^2+y^2)^{5/2}}\right) - (-\omega^2 A(t)\cos(\omega t+\alpha))\right]^2, \tag{3.7}$$

$$\mathcal{L}_{ay,2D} = \left[\left(-\mu\frac{y}{(x^2+y^2)^{3/2}} - \frac{3}{2}J_2\mu R^2\frac{y}{(x^2+y^2)^{5/2}}\right) - (-\omega^2 B(t)\cos(\omega t+\beta))\right]^2, \tag{3.8}$$

where x and y are given by Equation 3.3 and 3.4 at a given time t.

With all those components, the loss function of the 2D orbit propagation with gravitational perturbation $\mathcal{L}_{2D,GP}$ is given by

$$\mathcal{L}_{2D,GP} = \sqrt{\mathcal{L}_{x0} + \mathcal{L}_{y0} + \mathcal{L}_{vx0} + \mathcal{L}_{vy0} + \mathcal{L}_{ax,2D} + \mathcal{L}_{ay,2D}}. \tag{3.9}$$

It is worth noticing that in all analyzed cases (2D and 3D), the loss function's components that correspond to the acceleration ($\mathcal{L}_{ax,2D}$ and $\mathcal{L}_{ay,2D}$) are calculated in the entire simulation period, *i.e.*, from $t = 0$ to $t = 12$.

As the *ansatz*'s outputs ($x(t)$ and $y(t)$ from Equations 3.3 and 3.4) depend on the VQC's parameters, it is evident how they determine the value of the loss function $\mathcal{L}_{2D,GP}$. We then start the solution process with random parameters and use the Adam optimizer's routine (Equation 2.19) to generate new parameter sets that minimize the loss function at each iteration.

Moreover, it is also necessary to investigate the impacts of neglecting the derivatives of $A(t)$ and $B(t)$. If we compute such amplitudes numerically, we can also estimate their first and second derivatives via finite differences, *i.e.*,

$$\frac{dA(t)}{dt} = \frac{A(t+h_A) - A(t-h_A)}{2h_A}, \tag{3.10}$$

$$\frac{d^2A(t)}{dt^2} = \frac{A(t+h_A) - 2A(t) + A(t-h_A)}{h_A^2}, \tag{3.11}$$

$$\frac{dB(t)}{dt} = \frac{B(t+h_B) - B(t-h_B)}{2h_B}, \tag{3.12}$$

$$\frac{d^2B(t)}{dt^2} = \frac{B(t+h_B) - 2B(t) + B(t-h_B)}{h_B^2}, \tag{3.13}$$

where h_A and h_B are infinitesimal intervals ($h_A \to 0$ and $h_B \to 0$).

By using such approximations and recalculating the loss functions' components that correspond to velocity and acceleration (Equations 3.6, 3.7, and 3.8), we can estimate the contribution of the first and second derivatives we neglected in the the loss function as a whole (Equation 3.9). This analysis will allow us to understand whether neglecting such derivatives severely impacts our model's accuracy or not. We denote as $\tilde{\mathcal{L}}_{2D,GP}$ the new estimated loss function where no derivatives have been neglected. Thus,

$$\tilde{\mathcal{L}}_{2D,GP} = \sqrt{\mathcal{L}_{2D,GP}^2 + C_{2D,GP}}, \tag{3.14}$$

where $C_{2D,GP}$ is the expression (for the 2D case with gravitational perturbation) that contains all the first and second derivatives terms that should be in the loss function if we haven't neglected them.

3.2 3D Case with Gravitational Perturbation

Analogously to the 2D case from the previous section, we apply the *Ansatz* method for performing the 3D orbit propagation with gravitational perturbation. However, instead of solving the simplified version (Equations 3.1 and 3.2), we will use VQA to solve the three-dimensional problem represented by:

$$\begin{cases} \ddot{x} = -\mu x/r^3 + (3/2)J_2\mu(R^2/r^4)(x/r)(5z^2/r^2 - 1) \\ \ddot{y} = -\mu y/r^3 + (3/2)J_2\mu(R^2/r^4)(y/r)(5z^2/r^2 - 1) \\ \ddot{z} = -\mu z/r^3 + (3/2)J_2\mu(R^2/r^4)(z/r)(5z^2/r^2 - 3) \end{cases}, \tag{3.15}$$

which is derived in Appendix A, Equation A.27.

To accomplish this goal, we will once again use the fact that we are modeling a closed orbit. In addition to Equation 3.3 (x-axis) and 3.4 (y-axis), we can also write for the z-axis:

$$z(t) = C(t)\cos(\omega t + \gamma). \tag{3.16}$$

Representing this problem in a quantum circuit involves adding an extra qumode (wire) for the amplitude $C(t)$ and for the initial phase γ. Figure 3.3 illustrates the VQC we have designed for propagating orbits in the presence of gravitational perturbation, considering

the J_2 effects.

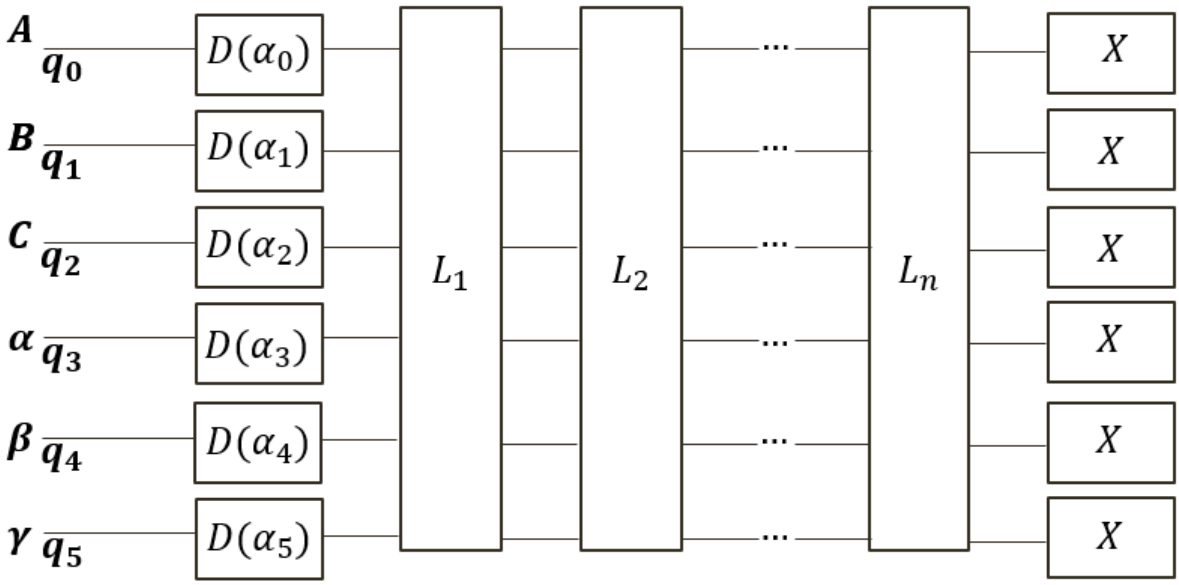

FIGURE 3.3 – VQC used to solve the 3D orbit propagation with perturbations. It is composed of six qumodes $(q_0, ..., q_5)$, each one associated to a displacement gate D followed by n layers, and a measurement gate X applied to each qumode.

Once we have established a broad overview of the VQC that solves the three-dimensional problem, we can define its layer and loss function. Similarly to the 2D case, the layer shall contain a connection between the amplitudes (via beamsplitter) and non-Gaussian component via the Kerr gate. Figure 3.4 shows one possibility for achieving this objective.

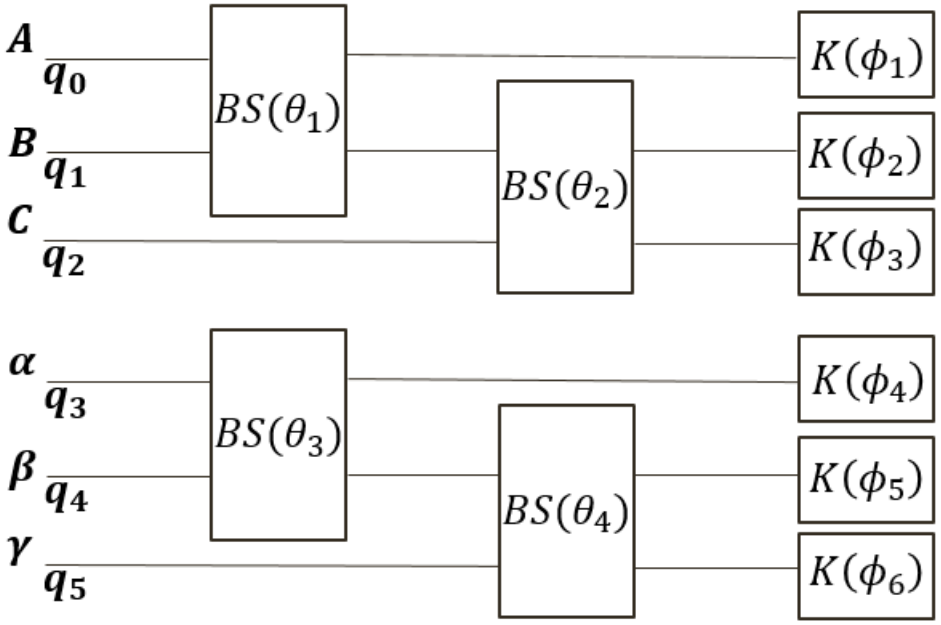

FIGURE 3.4 – Representation of the VQC layer for the 3D orbit propagation with perturbations. It is composed of six qumodes $(q_0, ..., q_5)$, associated in pairs to beamsplitter gates BS, followed by Kerr gates K applied to each qumode.

As for the loss function, we use the same idea as the 2D case. This time, however, the inputs regarding the satellite's motion are the initial position (x_0, y_0, z_0) and velocity $(\dot{x}_0, \dot{y}_0, \dot{z}_0)$. Hence, the loss functions' components relative to the initial position are:

$$\begin{cases} \mathcal{L}_{x0} = (x_0 - (A(0)\cos\alpha))^2 \\ \mathcal{L}_{y0} = (y_0 - (B(0)\cos\beta))^2 \\ \mathcal{L}_{z0} = (z_0 - (C(0)\cos\gamma))^2 \end{cases}. \tag{3.17}$$

Analogously to Equation 3.6, we also neglect the derivatives of $A(t), B(t)$, and $C(t)$ to obtain the following:

$$\begin{cases} \mathcal{L}_{vx0} = (\dot{x}_0 - (-\omega A(0)\sin\alpha))^2 \\ \mathcal{L}_{vy0} = (\dot{y}_0 - (-\omega B(0)\sin\beta))^2 \\ \mathcal{L}_{vz0} = (\dot{z}_0 - (-\omega C(0)\sin\gamma))^2 \end{cases}. \tag{3.18}$$

Finally, the part corresponding to the differential equation itself can be obtained using the same idea behind Equations 3.7 and 3.8. This case involves the z-axis as well, which leads to the following:

$$\mathcal{L}_{ax,GP} = \left[\left(-\mu\frac{x}{r^3} + \frac{3}{2}J_2\mu R^2\frac{x}{r^5}\left(5\frac{z^2}{r^2} - 1\right)\right) - (-\omega^2 A(t)\cos(\omega t + \alpha))\right]^2, \tag{3.19}$$

$$\mathcal{L}_{ay,GP} = \left[\left(-\mu\frac{y}{r^3} + \frac{3}{2}J_2\mu R^2\frac{y}{r^5}\left(5\frac{z^2}{r^2} - 1\right)\right) - (-\omega^2 B(t)\cos(\omega t + \beta))\right]^2, \tag{3.20}$$

$$\mathcal{L}_{az,GP} = \left[\left(-\mu\frac{z}{r^3} + \frac{3}{2}J_2\mu R^2\frac{z}{r^5}\left(5\frac{z^2}{r^2} - 3\right)\right) - (-\omega^2 C(t)\cos(\omega t + \gamma))\right]^2, \tag{3.21}$$

where x, y, z are given by Equations 3.3, 3.4, and 3.16, respectively, and $r = \sqrt{x^2 + y^2 + z^2}$.

Once we have calculated the components of the loss function for the gravitational perturbation problem, we can compute it as

$$\mathcal{L}_{3D,GP} = \sqrt{\mathcal{L}_{x0} + \mathcal{L}_{y0} + \mathcal{L}_{z0} + \mathcal{L}_{vx0} + \mathcal{L}_{vy0} + \mathcal{L}_{vz0} + \mathcal{L}_{ax,GP} + \mathcal{L}_{ay,GP} + \mathcal{L}_{az,GP}}. \tag{3.22}$$

Similarly to our analysis in the 2D case with gravitational perturbation, we are interested in checking whether ignoring the first and second derivatives of the amplitudes has a substantial impact in our model's output. Thus, in addition to the derivatives of $A(t)$ and $B(t)$ given by Equation 3.10 to 3.13, we can also write for $C(t)$ that:

$$\frac{dC(t)}{dt} = \frac{C(t + h_C) - C(t - h_C)}{2h_C}, \tag{3.23}$$

$$\frac{d^2C(t)}{dt^2} = \frac{C(t+h_C) - 2C(t) + C(t-h_C)}{h_C^2}, \tag{3.24}$$

where h_C is an infinitesimal interval ($h_C \to 0$). In this case, the estimated loss function in which we haven't neglected the derivatives is represented by $\tilde{\mathcal{L}}_{3D,GP}$. Considering that $C_{3D,GP}$ is the collection of first and second derivatives terms that should be in the loss function (if not neglected), we can then write that:

$$\tilde{\mathcal{L}}_{3D,GP} = \sqrt{\mathcal{L}_{3D,GP}^2 + C_{3D,GP}}. \tag{3.25}$$

3.3 3D Case with Atmospheric Drag

To propagate the 3D orbit in the presence of atmospheric drag, we use the same principles of the previous problem (gravitational perturbation considering J_2 effects). We can make that assumption because this case also involves the closed orbit of a satellite around the Earth. The problem we want to solve is mathematically described in the geocentric equatorial frame by the following system:

$$\begin{cases} \ddot{x} = -\mu x/r^3 - (1/2)\rho v_{rel} B(\dot{x} + y\omega_E) \\ \ddot{y} = -\mu y/r^3 - (1/2)\rho v_{rel} B(\dot{y} - x\omega_E) \\ \ddot{z} = -\mu z/r^3 - (1/2)\rho v_{rel} B\dot{z} \end{cases}, \tag{3.26}$$

in which ρ is the atmospheric density, $\vec{v}_{rel}$ is the satellite's relative velocity to the atmosphere, ω_E is the value of Earth's angular velocity, and $B = C_D A/m$ (C_D is the drag coefficient, A and m are the satellite's frontal area and mass, respectively). We derived Equation 3.26 in Appendix A, obtaining the Equation A.14.

Instead of solving the system in Equation A.14 directly, we calculate the variables $A(t), B(t), C(t), \alpha, \beta, \gamma$ from Equations 3.3, 3.4, and 3.16. As a result, we shall apply the VQC presented in Figure 3.3 with the layer from Figure 3.4. As for the loss function, the components relative to the initial conditions are identical to the previous section, *i.e.*, we need to take into account Equations 3.17 and 3.18. The main difference is the loss function's part that corresponds to the acceleration vector $\ddot{\vec{r}} = \{\ddot{x}, \ddot{y}, \ddot{z}\}$.

Computing this component of the loss function uses the same principles of Equations 3.20, 3.21, and 3.22. However, the benchmark value (against which we will compare the ansatz's output) is given by Equation A.14. Once again, by neglecting the derivatives of $A(t), B(t)$, and $C(t)$, we can write that:

$$\mathcal{L}_{ax,AD} = \left[\left(-\mu\frac{x}{r^3} - \frac{1}{2}\rho v_{rel} B(\dot{x} + y\omega_E)\right) - (-\omega^2 A(t)\cos(\omega t + \alpha))\right]^2, \tag{3.27}$$

$$\mathcal{L}_{ay,AD} = \left[\left(-\mu\frac{y}{r^3} - \frac{1}{2}\rho v_{rel} B(\dot{y} - x\omega_E)\right) - (-\omega^2 B(t)\cos(\omega t + \beta))\right]^2, \tag{3.28}$$

$$\mathcal{L}_{az,AD} = \left[\left(-\mu\frac{z}{r^3} - \frac{1}{2}\rho v_{rel} B\dot{z}\right) - (-\omega^2 C(t)\cos(\omega t + \gamma))\right]^2, \tag{3.29}$$

where r and v_{rel} are given by Equation A.15 and the subscript "AD" refers to atmospheric drag.

Hence, the total loss function for the 3D orbit propagation with atmospheric drag via VQA is given by

$$\mathcal{L}_{3D,AD} = \sqrt{\mathcal{L}_{x0} + \mathcal{L}_{y0} + \mathcal{L}_{z0} + \mathcal{L}_{vx0} + \mathcal{L}_{vy0} + \mathcal{L}_{vz0} + \mathcal{L}_{ax,AD} + \mathcal{L}_{ay,AD} + \mathcal{L}_{az,AD}}. \tag{3.30}$$

Once again, we repeat the same analysis as before via Equations 3.10, 3.11, 3.12, 3.13, 3.23, and 3.24. The objective is to estimate the impact of our approximations in the loss function as a whole. Similarly to previous cases, we denote the new estimated loss function as $\tilde{\mathcal{L}}_{3D,AD}$. Therefore,

$$\tilde{\mathcal{L}}_{3D,AD} = \sqrt{\mathcal{L}^2_{3D,AD} + C_{3D,AD}}, \tag{3.31}$$

where $C_{3D,AD}$ contains all first and second derivative terms that would be in the loss function expression if we haven't neglected them beforehand.

3.4 3D Orbit Propagation with Perturbations via Numerical Methods

After solving the orbit propagation problem with gravitational perturbation and atmospheric drag via VQA, we need to verify our solution's accuracy. As the systems described by Equations 3.15 and 3.26 cannot be solved analytically, we will use their numerical solution as a benchmark.

We use Python's open-source library SciPy (VIRTANEN *et al.*, 2020), which is vastly used in many scientific projects worldwide. It provides numerous mathematical algorithms for optimization, integration, differential equations, and many more. In particular, we will use SciPy's ODEINT – an integrator based on FORTRAN's Livermore Solver for Ordinary Differential Equations (LSODE) library (RADHAKRISHNAN; HINDMARSH, 1993).

Alongside the parameter values and initial conditions (described in Chapter 4) for each problem, we also define some aspects of the integrator. The total simulation time is 12 hours (*i.e.*, we simulate the satellite's motion between $t_0 = 0$ and $t = 12$ hours), which is discretized in 1200 points. Moreover, we define the "atol" and "rtol" parameters that

determine the error control performed by the solver. Table 3.1 summarizes the integrator parameters.

Integrator Parameters	**Value**
Simulation Time	12
Number of Points	1200
atol	10^{-8}
rtol	10^{-6}

TABLE 3.1 – SciPy's ODEINT parameters.

3.5 Alternative Approaches

As previously discussed, we chose the *Ansatz* method because it substantially reduces the possible solutions that can be explored by the optimizer. Instead of fitting the solution by considering infinite possibilities, the optimizer only needs to fit some pre-determined parameters. Therefore, the optimizer should be able to reduce the loss function and achieve a reasonable solution in fewer iterations.

Nevertheless, we should acknowledge the existence of other approaches and comment on their advantages and shortcomings. One possibility is to linearize the equations and solve them via a quantum algorithm based on the corresponding truncated Taylor series (XIN *et al.*, 2020; FERREIRA, 2022). Although this approach makes it easier to solve simple nonlinear differential equations, the linearization process is not straightforward for systems with coupled equations. On top of that, the solution's accuracy increases with the order of the truncated Taylor series. This means that one would need to write the expansion in multiple terms to obtain a valid approximation – which would not simplify the problem.

Another idea is to do the orbit propagation by combining VQA and the Finite Difference Method (FDM) (HEINRICH, 1987). We shall consider a sample problem to explain this approach in greater detail. Let us assume that we wish to solve the following system of coupled differential equations:

$$\begin{cases} dx/dt = -y \\ dy/dt = x \end{cases}, \tag{3.32}$$

knowing that $x(0) = 0$ and $y(0) = 1$. We want to solve it for $x(t)$ and $y(t)$.

The analytical solution is

$$\begin{cases} x(t) = -\sin t \\ y(t) = \cos t \end{cases}, \tag{3.33}$$

which we will adopt as a benchmark to analyze the accuracy of our solution. To solve this example via VQA with FDM, we build two VQCs: the first one, $\mathrm{VQC}_1(t)$, encodes $x(t)$, while $\mathrm{VQC}_2(t)$ encodes $y(t)$. Each circuit has its own loss function called $\mathcal{L}_1$ and $\mathcal{L}_2$, respectively, which should be minimized by updating their parameter set Θ_1 and Θ_2. Figure 3.5 represents a high-level diagram that solves this problem.

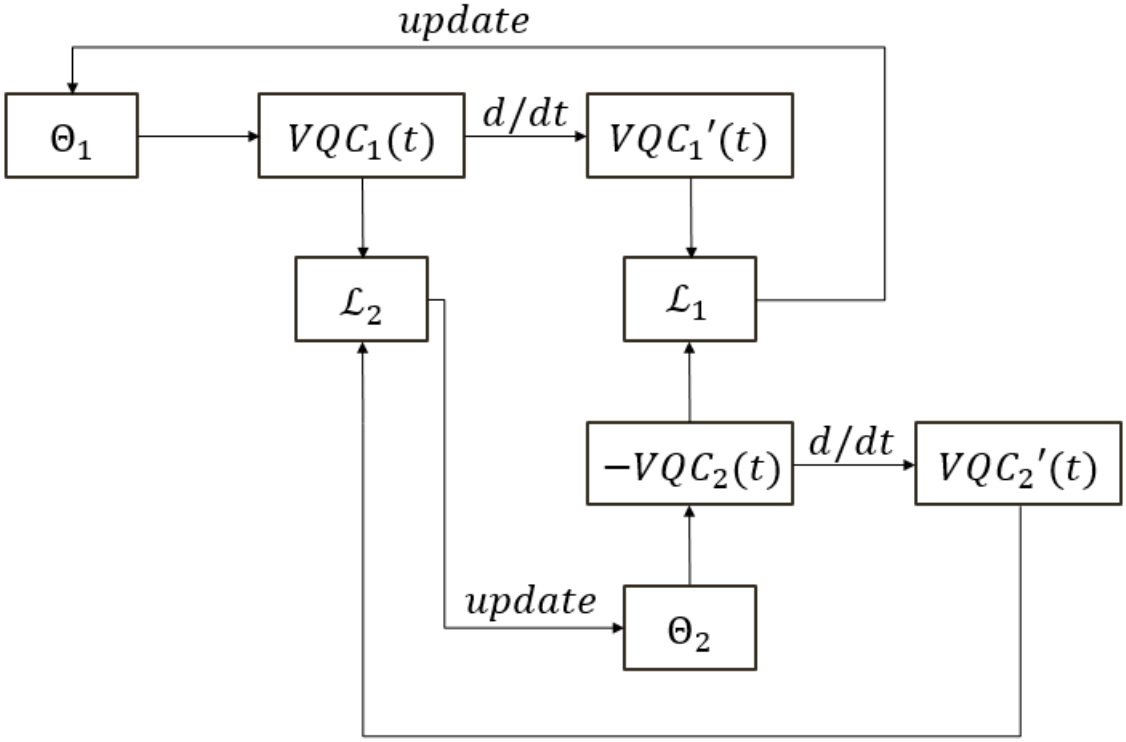

FIGURE 3.5 – Diagram of the method that uses VQA and FDM for solving the proposed system of coupled differential equations. It contains two VQCs, each of which receives a parameter set (Θ_1, Θ_2) as an input and aims to minimize a loss function $(\mathcal{L}_1, \mathcal{L}_2)$ by updating the corresponding parameter set iteratively via a classical optimizer.

As indicated in Figure 3.5, the parameter set defines the gates of its respective VQC. In this case, as we seek to have $\mathrm{VQC}_1(t) = x(t)$ and $\mathrm{VQC}_2(t) = y(t)$, $\mathcal{L}_1$ should involve the difference between $\mathrm{VQC'}_1(t)$ and $-\mathrm{VQC}_2(t)$, while $\mathcal{L}_2$ compares $\mathrm{VQC}_1(t)$ to $\mathrm{VQC'}_2(t)$. As a consequence, when the optimizer updates Θ_1 and Θ_2 such that both loss functions tend to zero, we would have $x'(t) \to -y(t)$ and $y'(t) \to x(t)$, which are the conditions imposed by the system (Equation 3.32). Therefore, this construction allows us – at least in principle – to solve the proposed system of coupled differential equations.

The FDM component of this solution consists of defining the differentiation procedure (d/dt) from Figure 3.5 as finite differences. Since we are dealing with a first-order differentiation with respect to time, we can use the following approximations:

$$\mathrm{VQC}'_1(t) = \frac{\mathrm{VQC}_1(t + h_1) - \mathrm{VQC}_1(t - h_1)}{2h_1}, \tag{3.34}$$

$$\mathrm{VQC}_2'(t) = \frac{\mathrm{VQC}_2(t+h_2) - \mathrm{VQC}_2(t-h_2)}{2h_2}, \tag{3.35}$$

where h_1 and h_2 are arbitrarily small numbers called step sizes. It is worth noticing that we could also use higher-order approximations if we want to reduce the truncation error. In particular, the first-order's error is proportional to the step size, *i.e.*, $O(h)$.

Even though this approach is correct in principle, further tests revealed that it requires a high amount of iterations to minimize the loss function effectively. Due to the fact that one loss function depends on the other, the optimization process is not as straightforward (AMES *et al.*, 1997) as in the *Ansatz* approach. Additionally, a complex system of coupled equations – such as the ones we are solving in this dissertation – would enhance all these limitations and make it unfeasible to use this technique.

4 Results

This chapter presents the results we obtain when performing the orbit propagation in the presence of perturbations via VQA and numerical integrator. We start showing the results for the two-dimensional case of the gravitational perturbation problem. We then discuss the 3D orbit propagation in the presence of the effects of the J_2 term and the atmospheric drag. Each discussion includes a comparison with the corresponding numerical solution.

4.1 2D Orbit Propagation with Gravitational Perturbation

We start applying our method to a simplified 2D version of the gravitational perturbation problem. We choose Example 10.2 (CURTIS, 2013) as a baseline. It states that, at time $t = 0$, there is a satellite orbiting Earth with some known orbital parameters: perigee radius (r_p), apogee radius (r_a), right ascension of the ascending node (Ω), inclination (i), argument of the perigee (ω), and true anomaly (θ_{TA}). Table 4.1 contains such parameters.

Parameter	**Value**	**Unit**
r_p	6678	km
r_a	9940	km
Ω	45	degree
i	28	degree
ω	30	degree
θ_{TA}	40	degree

TABLE 4.1 – Orbital parameters from the Example 10.2 (CURTIS, 2013).

Using the Appendix A procedure, we can obtain the initial position and velocity vector, *i.e.*, $\vec{r_0} = (x_0, y_0, z_0)$ and $\vec{v_0} = (\dot{x}_0, \dot{y}_0, \dot{z}_0)$. In our adapted 2D version, we set $z = \dot{z} = \ddot{z} = 0$ for the entire orbit propagation process. Tables 4.2 and 4.3 present the remaining parameters and initial conditions, respectively.

We perform all simulations in a development environment with an Intel(R) Celeron(R) N4000 CPU 1.10GHz processor and 4GB RAM. We run the VQA simulation in PennyLane, a software framework in Python for differentiable quantum computing, designed

Parameter	Value	Unit
μ	398,600	km^3/s^2
J_2	1.08263×10^{-3}	–
R	6378	km
T	2	h
Simulation Time	12	h

TABLE 4.2 – Parameters used in the two-dimensional orbit propagation considering the effects of the J_2 term.

Condition	Value	Unit
x_0	-2384.46	km
y_0	5729.01	km
$\dot{x}_0$	-7.36138	km/s
$\dot{y}_0$	-2.98997	km/s

TABLE 4.3 – Initial conditions used in the two-dimensional orbit propagation considering the effects of the J_2 term.

and distributed by the Canadian company Xanadu (BERGHOLM *et al.*, 2018). PennyLane substantially facilitates the training of VQCs, and for that reason, it has been used in multiple projects (KWAK *et al.*, 2021; DELGADO *et al.*, 2021; KONAR *et al.*, 2023). In all cases, we also compare both classical solution's and VQA's running times in this specific hardware.

PennyLane provides multiple backends for running the quantum circuit. We chose the StrawberryFields simulator (KILLORAN *et al.*, 2019b) due to its specific design for photonic quantum computing – which matches the continuous-variable formalism of this work, as explained in Chapter 2. Each simulation involves 5000 iterations with a 0.01 learning rate and 4 layers. We use this configuration in all problems (2D and 3D gravitational perturbation and 3D atmospheric drag), and we chose it based on data from literature (KILLORAN *et al.*, 2019a) and toy models inspired by these problems.

Having established the technical specifications, we shall proceed to the analysis of the results. Firstly, we plot ODEINT's numerical results together with VQA's outputs. Figure 4.1 demonstrates a satisfactory fit between the VQA model and the numerical solution in both the x and y-axis. An alternative analysis involves comparing both orbits (generated numerically and via VQA) in the XY plane, considering the total simulation period of 12 hours (Figure 4.2). We use the parameters associated with the lowest loss function – *i.e.*, not necessarily the last iteration's parameters – to plot these results.

Figures 4.1 and 4.2 suggest the satellite's orbit is a Medium-Earth Orbit (MEO), given the altitude – more details are provided in the Appendix A. Additionally, one should note that the VQA results were obtained after 5000 iterations and by using the parameters

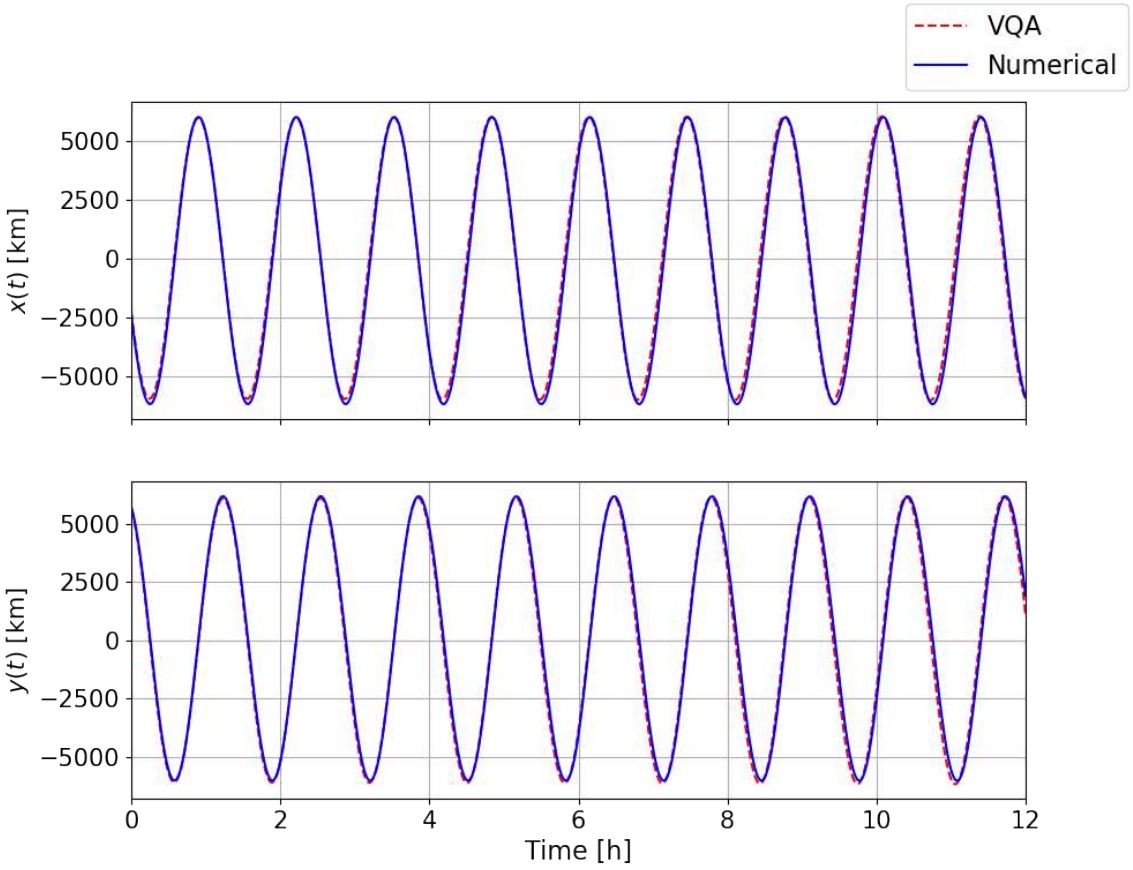

FIGURE 4.1 – Numerical (continuous blue line) and VQA's (dashed red line) components $(x(t), y(t))$ of the 2D orbit propagation with gravitational perturbation (effects of the J_2 term) as a function of time.

associated with the lowest loss function. The numerical approach took 328 milliseconds, while the VQA solution took 2 hours, 55 minutes and 53 seconds. It is worth mentioning that this running time difference is mainly due to the fact that we executed the VQA in a classical simulator, and not in a quantum device.

Additionally, it is interesting to understand how the model's convergence depends on the number of iterations (Figure 4.3). As expected, generally speaking, our model becomes more accurate as we increase the number of iterations. Graphically, this means that the loss function tends to decrease as we perform more iterations (Figure 4.3), going from $\mathcal{L}_{2D,GP} = 1.655 \times 10^5$ to $\mathcal{L}_{2D,GP} = 1.452 \times 10^2$ at the minimum value. Although this is the general trend, the loss function might eventually grow from one iteration to another. Such a phenomenon happens because the gradient-based algorithm does not know *a priori* which trajectory in the parameter landscape will lead to the minimization of the loss function.

Moreover, one could argue whether the approximation that the derivatives of $A(t)$ and $B(t)$ being small enough to be neglected was actually valid or not. We can address this

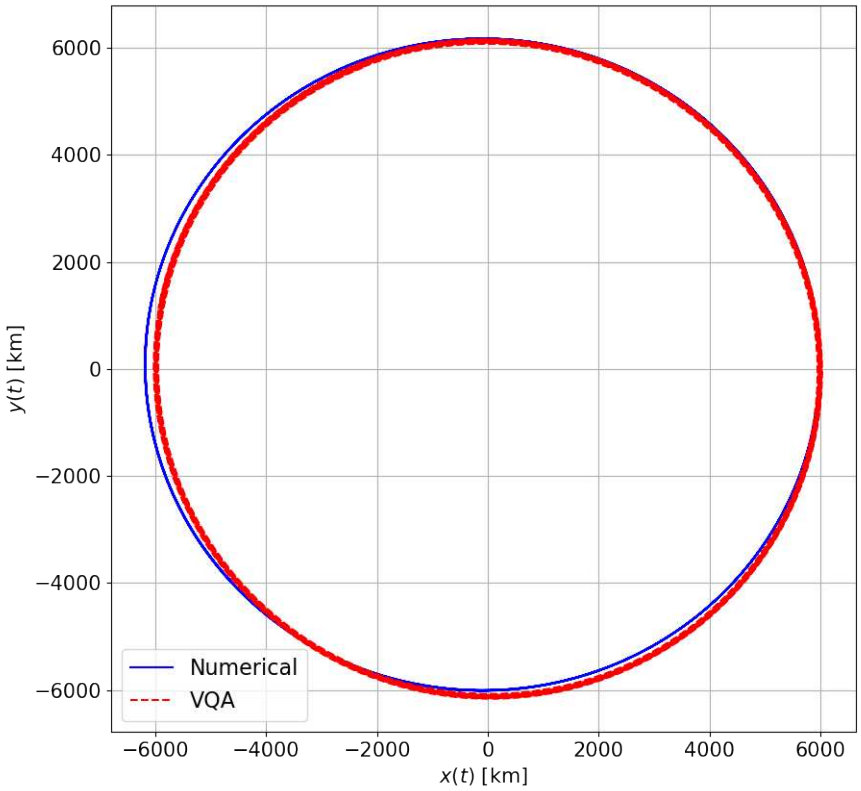

FIGURE 4.2 – Numerical (continuous blue line) and VQA's (dashed red line) 2D orbits in the presence of the gravitational perturbation (effects of the J_2 term).

question by computing the loss function variation had we not neglected such terms – as indicated in Equation 3.14. After performing those calculations, we estimate that the loss function considering such derivatives would be $\tilde{\mathcal{L}}_{2D,GP} = 1.436 \times 10^2$ at its minimum value. Given the loss function's reduction of roughly 1.12%, we can say that our approximation was valid in this case.

Figure 4.3 illustrates that our method can consistently reduce the loss function up to approximately 1000 iterations. From this point until around 3700 iterations, the VQA reaches a barren plateau in which the loss function remains roughly the same. We can explain the subsequent behavior (from 3700 onward) as the Adam optimizer exploring the parameter landscape in regions near the previous local minimum. As the optimizer tries these neighborhoods, the loss function increases and decreases abruptly.

Adam optimizer eventually finds a region associated with a lower loss function – marked with the inferior dashed line in Figure 4.3. In the last iterations, the optimizer appears to be stuck around some local minima in the parameter landscape, as indicated by the loss function's oscillation between fixed points. Such behavior suggests that the

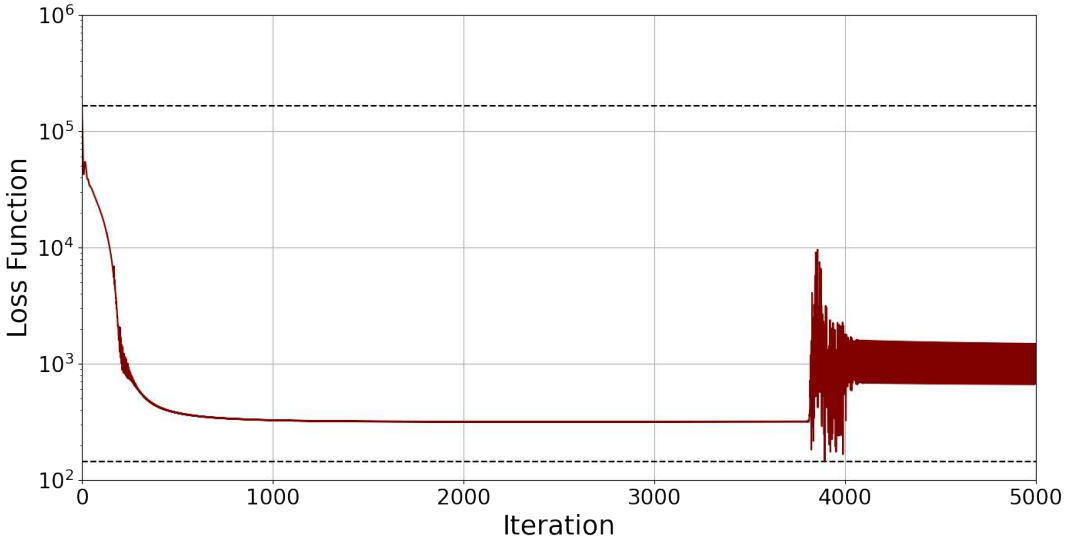

FIGURE 4.3 – Loss function *versus* the number of iterations for the 2D orbit propagation with gravitational perturbation (effects of the J_2 term). The dotted lines mark the minimum and maximum values of the loss function.

Adam optimizer might not be the best option for obtaining a highly-accurate solution.

In future works, we should instead consider optimizers that explore multiple pathways in the parameter space simultaneously, such as the Particle Swarm Optimization (PSO) (KENNEDY; EBERHART, 1995). The advantage of PSO over Adam is that it uses a group of particles to explore the search space. Each particle represents a potential solution and moves through the search space to find the optimal solution (CLERC, 2010). This trait maintains a balance between exploration – searching for new regions – and exploitation – focusing on promising areas – which is crucial for avoiding premature convergence to local optima (SHANKAR *et al.*, 2016). Another possibility would be genetic algorithms (GAs), as they keep a population of possible solutions exploring the search space throughout the entire process (WRIGHT, 1991). Mechanisms like crossover and mutation allow GAs to avoid local optima and find the best solution globally (SRINIVAS; PATNAIK, 1994).

In addition to the loss function in terms of the iterations, we should also analyze the behavior of our model's core components, *i.e.*, amplitude and initial phase. Since our *ansatz* is based on determining these features for each axis ($x(t)$ and $y(t)$), it is crucial to comprehend how the results of amplitude and phase evolve as a function of the number of iterations.

Achieving this goal involves creating a metric that reflects the proximity between the numerical and VQC's solution as a function of amplitude or phase. Thus, we introduce

the mean relative error per cycle (which we will abbreviate as "MREC"), defined as:

$$\mathrm{MREC}(k) = \frac{1}{n_p}\left|\frac{k_{\mathrm{Num}} - k_{\mathrm{VQC}}}{k_{\mathrm{Num}}}\right|, \tag{4.1}$$

where k is the feature we are analyzing (amplitude or phase), n_p is the number of periods within the total simulation time (12 hours), and k_{Num} and k_{VQC} are the numerical and VQC's results for that feature, respectively.

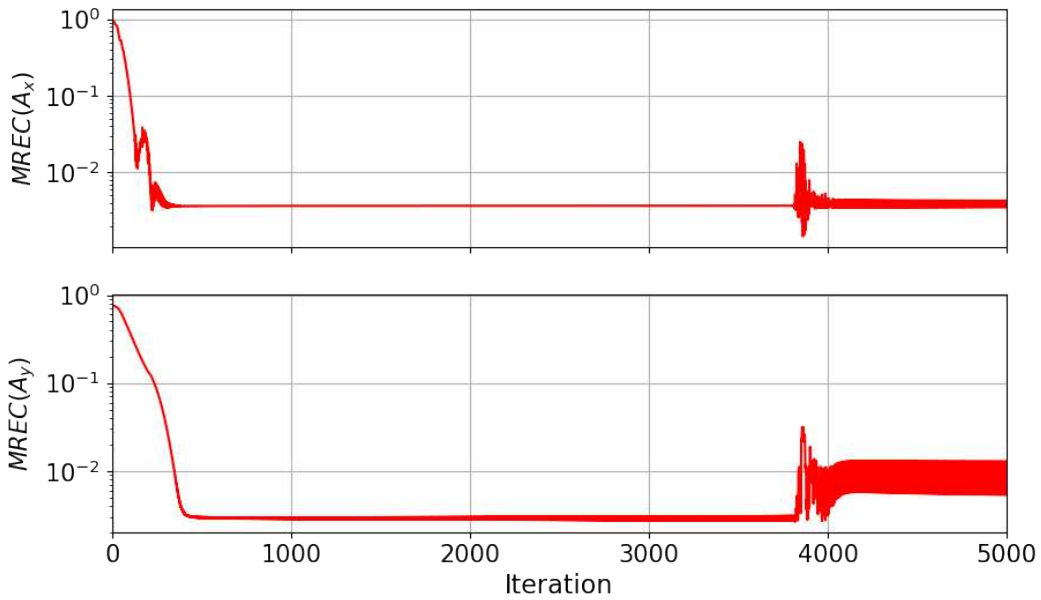

FIGURE 4.4 – Mean Relative Error per Cycle (MREC) of amplitudes as a function of the number of iterations for the 2D orbit propagation with gravitational perturbation (effects of the J_2 term).

We are interested in understanding how the MREC of amplitude and phase change over the simulation process – as illustrated in Figures 4.4 and 4.5. For the sake of simplicity, we adopt the following notation:

$$\begin{cases} A_x = A \\ A_y = B \\ \phi_x = \alpha \\ \phi_y = \beta \end{cases}, \tag{4.2}$$

for amplitudes and phases, respectively.

As expected, the curves of $\mathrm{MREC}(A_x), \mathrm{MREC}(A_y), \mathrm{MREC}(\phi_x)$, and $\mathrm{MREC}(\phi_y)$ have similar behavior when compared to the loss function. This conformity is essential because MREC converging to zero implies that our model's accuracy is increasing, which is associated with a decreasing loss function.

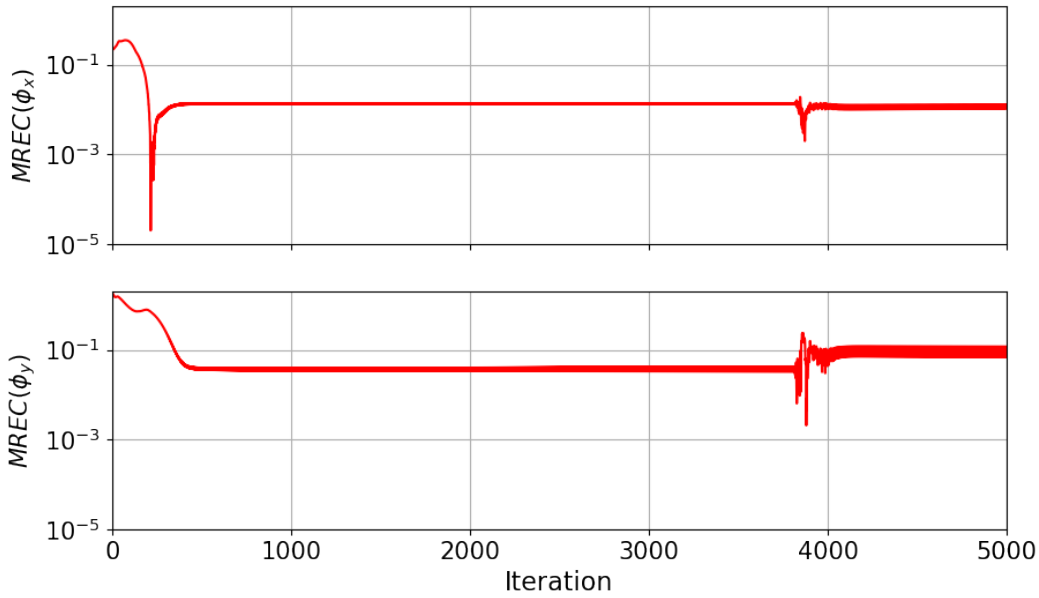

FIGURE 4.5 – Mean Relative Error per Cycle (MREC) of phases as a function of the number of iterations for the 2D orbit propagation with gravitational perturbation (effects of the J_2 term).

For that reason, the MREC curves intuitively should have a decreasing trend within the first iterations, followed by a plateau up until the 3700$^{\text{th}}$ repetition – as we can attest in Figures 4.4 and 4.5. In particular, we can observe that the curves of MREC(A_x) and MREC(ϕ_x) have abrupt variations within the initial 500 iterations before reaching the plateau state. Even though the value of MREC(ϕ_x) increases afterward, VQA's output as a whole becomes more accurate.

Moreover, we can also explain the sudden drop in the MREC(ϕ_x) curve. That happened because the optimizer found a specific set of parameters that produced that effect. However, as we update the parameters at each iteration, the MREC value might increase in the following iterations. Indeed, the lowest values of MREC(ϕ_x) and MREC(ϕ_y) are 2.007×10^{-5} and 2.086×10^{-3}, respectively. On the other hand, at the end of the iterations, their results are MREC(ϕ_x) $= 1.217 \times 10^{-2}$ and MREC(ϕ_y) $= 0.1067$.

Immediately after the 3700$^{\text{th}}$ iteration, as the Adam optimizer starts to explore other regions aiming for another minimum, we naturally observe sudden variations in MREC curves, which stabilize around a given value in the last iterations. In particular, for the amplitudes, these values are MREC(A_x) $= 3.527 \times 10^{-3}$ and MREC(A_y) $= 1.100 \times 10^{-2}$. Additionally, the minima are MREC(A_x) $= 1.474 \times 10^{-3}$ and MREC(A_y) $= 2.653 \times 10^{-3}$.

Based on those results, we can attest that the VQA model does not provide a satisfactory approximation to the numerical solution of the problem under the analyzed

conditions. As indicated in Figure 4.2, there are some points in the orbit in which the deviation between the numerical and VQA's solution is up to 200 kilometers, approximately. Even though the altitude tolerance error highly depends on the mission requirements (JR, 2017), an error of 200 kilometers for a MEO satellite is considered too elevated, even for less strict missions (WERTZ *et al.*, 2011).

Moreover, there is no evidence that performing more iterations would result in better results. Although there is a general trend that the model gets more accurate with the iterations, the classical optimizer appears to be stuck in a local minimum from the 3700^{th} iteration onward. As we cannot guarantee that the Adam optimizer will eventually find a new minimum, we conclude that the proposed combination of circuit architecture and optimizer is not able to solve the problem considering practical tolerance errors. One should notice, however, that alternative optimizers, such as PSO and GA, or circuit architectures could potentially lead to better solutions.

4.2 3D Orbit Propagation with Gravitational Perturbation

Once we have analyzed the 2D case, we can move on to the orbit propagation with gravitational perturbations considering the J_2 term effects. We use the same development environment, framework (PennyLane), and backend simulator (StraberryFields) presented in section 4.1.

Once again, we will solve the Example 10.2 (CURTIS, 2013) – described in section 4.1. The parameters are present in Tables 4.1 and 4.2. Using the formulas from the Appendix A, we can determine the initial conditions (position and velocity at $t = 0$), as we can see in Table 4.4.

Condition	**Value**	**Unit**
x_0	-2384.46	km
y_0	5729.01	km
z_0	3050.46	km
$\dot{x}_0$	-7.36138	km/s
$\dot{y}_0$	-2.98997	km/s
$\dot{z}_0$	1.64354	km/s

TABLE 4.4 – Initial conditions used in the three-dimensional orbit propagation considering the effects of the J_2 term.

Like the previous problem, we plot the numerical and the VQC's solution together. As shown in Figure 4.6, we analyze the match of our model compared to the numerical solution considering the curves of $x(t), y(t)$ and $z(t)$. Moreover, Figure 4.7 illustrates the trajectory obtained after the total simulation time. Similarly to the 2D problem, we save

the parameters that correspond to the lowest loss function and use them in our VQC to obtain these results. As for the running time, in this case, we had 344 milliseconds for the numerical solution and 3 hours, 34 minutes and 57 seconds for the VQA, which can be attributed to our classical simulator backend once again.

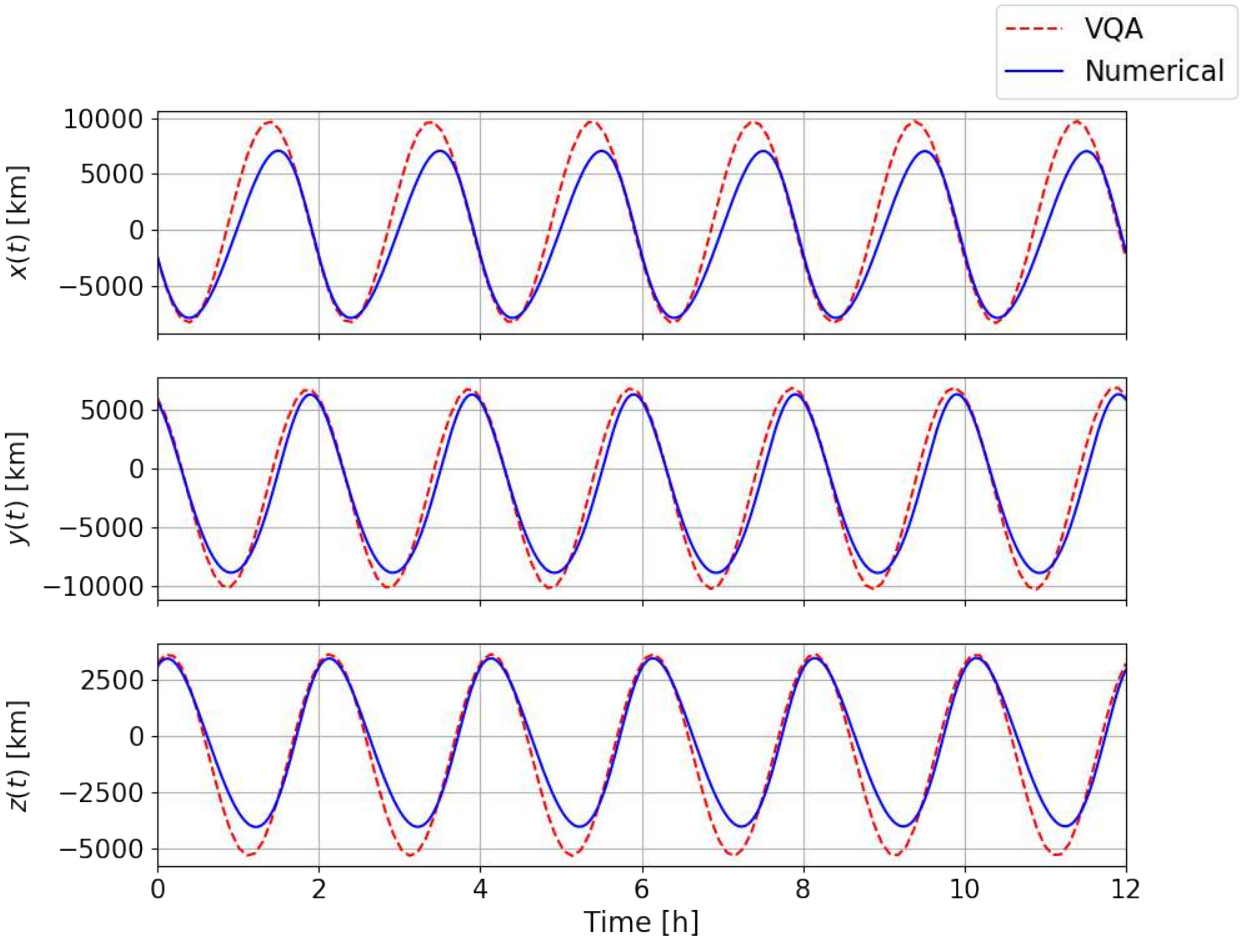

FIGURE 4.6 – Numerical (continuous blue line) and VQA's (dashed red line) components $(x(t), y(t), z(t))$ of the 3D orbit propagation with gravitational perturbation (effects of the J_2 term) as a function of time.

As we can see from Figures 4.6 and 4.7, there is not a reasonable fit between the numerical output and the VQA's solution. We can visually attest that the proximity between the models is greater on the y-axis; for the x and z-axis, however, there is room for improvement, especially in the extremes (local maxima in $x(t)$ and minima in $z(t)$). Such differences generate orbits that are relatively close to each other in some points, but have higher disparities in other positions, as illustrated in Figure 4.7.

Analogously to our previous analysis, it is vital to comprehend the evolution of the loss function in terms of the number of iterations. Therefore, it is evident in Figure 4.8 that the loss function starts at a high value ($\mathcal{L}_{3D,GP} = 3.541 \times 10^5$), abruptly falls within the first 1000 iterations, and then decreases at a slower pace from this point up until

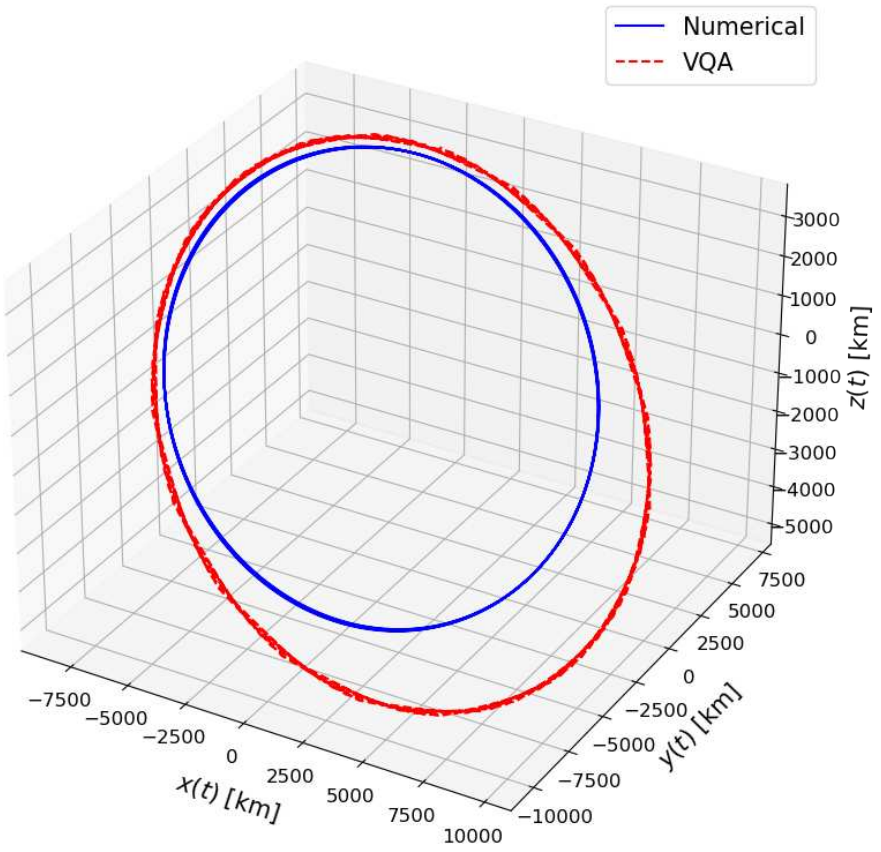

FIGURE 4.7 – Numerical (continuous blue line) and VQA's (dashed red line) 3D orbits in the presence of the gravitational perturbation (effects of the J_2 term).

the end of the simulation, reaching $\mathcal{L}_{3D,GP} = 4.838 \times 10^2$. The analysis of the impact of neglecting the derivatives of amplitudes $A(t), B(t)$, and $C(t)$ represented in Equation 3.25 reveals that $\tilde{\mathcal{L}}_{3D,GP} = 4.598 \times 10^2$, which corresponds to a 4.96% reduction in terms of the lowest loss function. Hence, neglecting those derivatives was a valid approach indeed.

Unlike the 2D problem, however, this case has not exhibited a barren plateau in 5000 iterations. Rather, the loss function slowly decreases during the process, indicating that the Adam optimizer is exploring the parameter landscape and converging at least to a local minimum.

Additionally, we should comment on a sudden loss function's oscillation around the 400^{th} iteration. Similar to what we have discussed in the 2D case, we can understand this event as the Adam trying different directions around a given point to minimize the loss

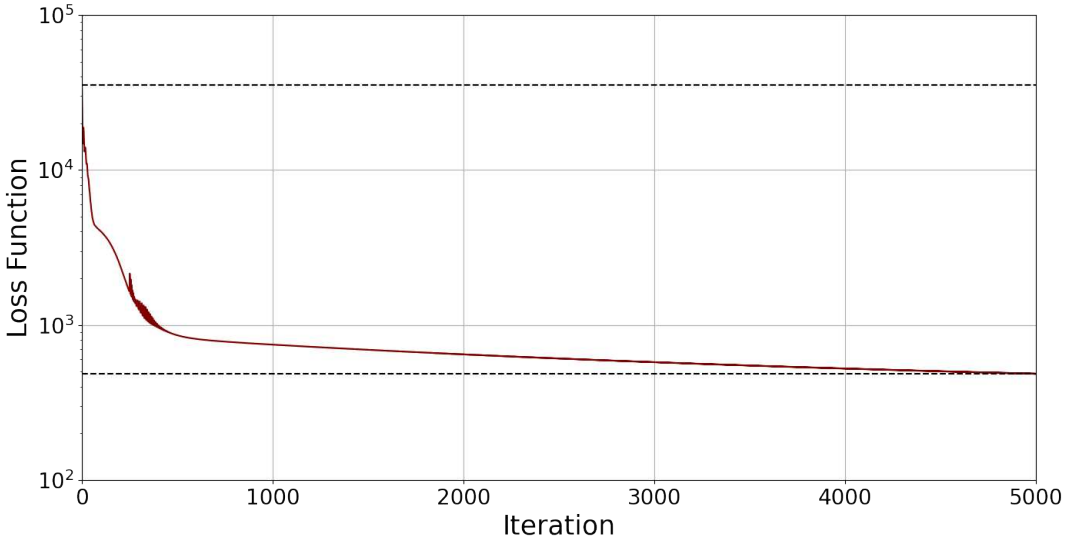

FIGURE 4.8 – Loss function *versus* the number of iterations for the 3D orbit propagation with gravitational perturbation (effects of the J_2 term). The dotted lines mark the minimum and maximum values of the loss function.

function. Eventually, the optimizer finds the best path that will decrease the loss function according to the calculations expressed in Equation 2.19.

Since we have analyzed the behavior of the loss function in terms of the number of iterations, we can proceed to investigate how our ansatz's features (amplitude and phase) evolve throughout the simulation process. We adopt the same notation expressed in Equation 4.2 with an additional definition that $A_z = C$ and $\phi_z = \gamma$. The results are presented in Figures 4.9 and 4.10.

As indicated by Figure 4.9, the values of MREC for A_x, A_y, and A_z reach values as low as $9.157 \times 10^{-4}, 6.286 \times 10^{-5}$, and 5.869×10^{-2}, respectively. At the end of the process, the MREC curves of amplitudes converge to the following values: $\text{MREC}(A_x) = 0.2868$, $\text{MREC}(A_y) = 6.955 \times 10^{-2}$, and $\text{MREC}(A_z) = 5.939 \times 10^{-2}$. As for the phases, the MREC curves reach 1.251×10^{-2}, 6.670×10^{-4}, and 0.2423 for ϕ_x, ϕ_y, and ϕ_z, respectively, as their lowest values. After 5000 iterations, they converge to $\text{MREC}(\phi_x) = 1.337 \times 10^{-2}$, $\text{MREC}(\phi_y) = 1.998 \times 10^{-2}$, and $\text{MREC}(\phi_z) = 0.2481$.

We shall discuss the reason why the MREC curves in the 3D problem do not converge to values as low as in the two-dimensional case. The former situation has more parameters than the latter, resulting in a parameter landscape with more dimensions. Therefore, the optimizer needs to update more variables to minimize the loss function. As a consequence, 5000 iterations are not sufficient to find the correct parameters' values that will reduce all MREC values to match the results from the 2D problem. We can extend the same argument to the loss function comparison between the two and three-dimensional cases.

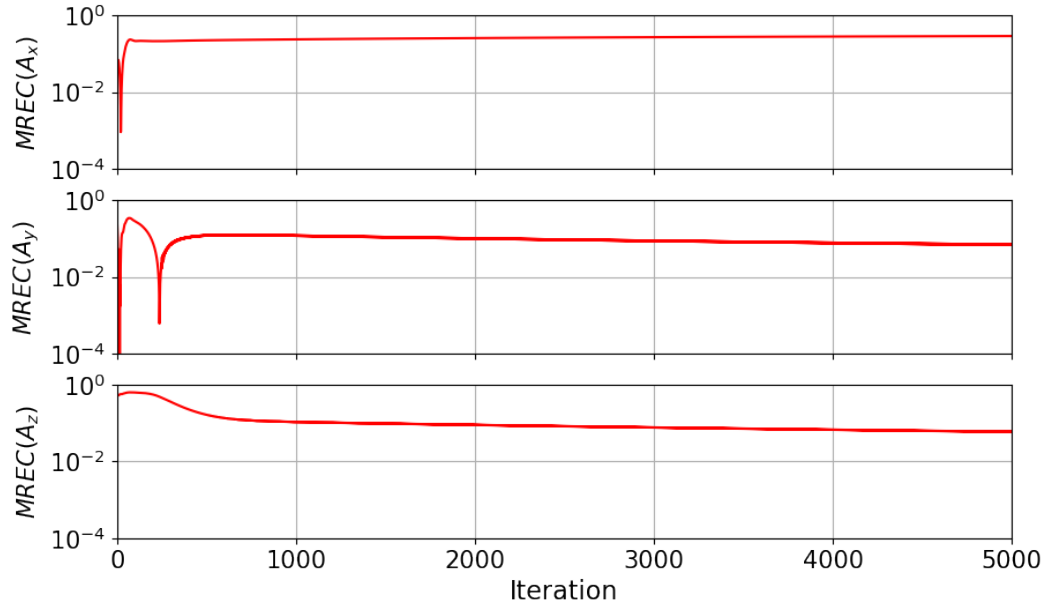

FIGURE 4.9 – Mean Relative Error per Cycle (MREC) of amplitudes as a function of the number of iterations for the 3D orbit propagation with gravitational perturbation (effects of the J_2 term).

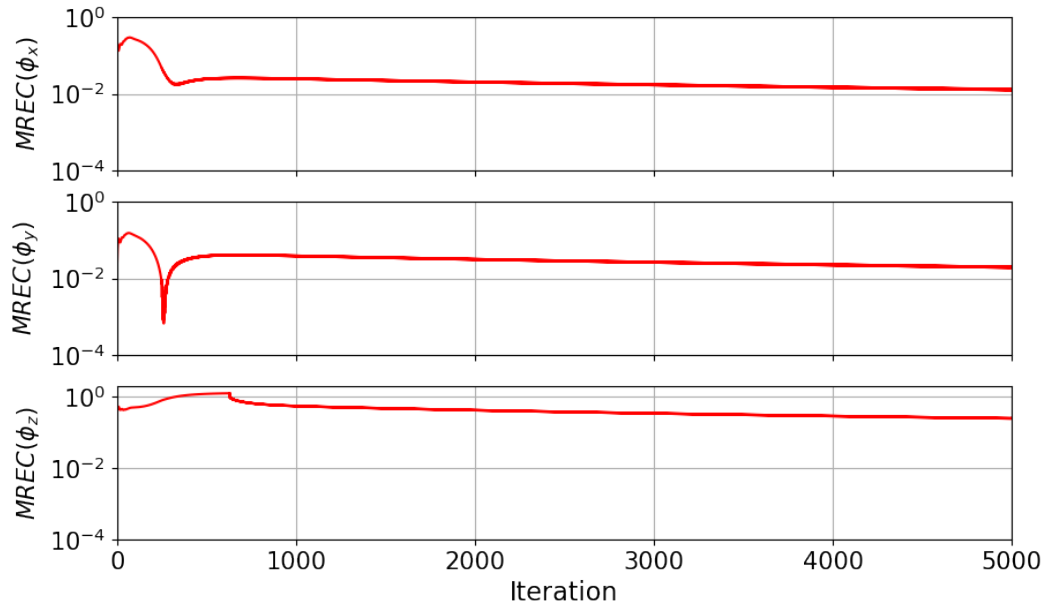

FIGURE 4.10 – Mean Relative Error per Cycle (MREC) of phases as a function of the number of iterations for the 3D orbit propagation with gravitational perturbation (effects of the J_2 term).

We can verify that applying the VQA model to the 3D orbit propagation with gravitational perturbation has not provided satisfactory results – the deviation illustrated in

Figure 4.7 (hundreds of kilometers) is greater than the acceptable for missions in MEO (WERTZ *et al.*, 2011). Over the iterations, A_y, A_z, ϕ_x, ϕ_y and ϕ_z became progressively more accurate (although in a low slope), while A_x remained virtually the same. This behavior suggests the Adam optimizer was continuously exploring regions that minimize all the variables of our model except for A_x.

As a consequence, the loss function also decreases at a slow pace throughout the process. Yet, 5000 iterations were insufficient to reduce the loss function to a magnitude similar to the two-dimensional problem. As we need more parameters to model the 3D scenario, the optimizer requires more iterations to adjust all these values in a specific way to minimize the loss function. Considering the loss function's decreasing behavior, it might be possible to obtain more accurate solutions through more iterations in this case. Other strategies include replacing the Adam optimizer with PSO or GA, or modeling the problem with fewer parameters – which would reduce the search space the optimizer has to explore.

4.3 3D Orbit Propagation with Atmospheric Drag

In this case, we will solve Example 10.1 from the same source (CURTIS, 2013). It describes a small spherical satellite (mass $m = 100$ kilograms and diameter $D = 1$ meter) revolving around the Earth in a closed orbit with atmospheric drag, which is extensively discussed in the Appendix A. Tables 4.5 and 4.6 contain the parameters of the problem. Analogously to the sections 4.1 and 4.2, we apply the formulas from the Appendix A to obtain the satellite's initial position and velocity (Table 4.7).

Parameter	**Value**	**Unit**
r_p	6593	km
r_a	7317	km
Ω	340	degree
i	65.1	degree
ω	58	degree
θ_{TA}	332	degree

TABLE 4.5 – Orbital parameters from the Example 10.1 (CURTIS, 2013).

As the previous cases, we solve this problem using the same development environment used in sections 4.1 and 4.2, PennyLane and StrawberryFields. We start by plotting the orbit's components in function of time ($x(t), y(t)$, and $z(t)$) in a comparison between the VQA and the numerical approach. The following step is to plot the full orbit – considering the total simulation time of 12 hours – in a three-dimensional plot, considering both solutions as well (Figures 4.11 and 4.12). As for the running time, in this case, the

Parameter	Value	Unit
μ	398,600	km^3/s^2
ρ	10^{-11}	kg/m^3
ω_E	72.9211×10^{-6}	rad/h
C_D	2.2	–
A	0.25π	m^2
m	100	kg
T	96.207	min
Simulation Time	12	h

TABLE 4.6 – Parameters used in the three-dimensional orbit propagation considering atmospheric drag.

Condition	Value	Unit
x_0	5873.40	km
y_0	-658.522	km
z_0	3007.49	km
$\dot{x}_0$	-2.89641	km/s
$\dot{y}_0$	4.94010	km/s
$\dot{z}_0$	6.14446	km/s

TABLE 4.7 – Initial conditions used in the three-dimensional orbit propagation considering atmospheric drag.

numerical solution took 336 milliseconds, while the VQA solution took 3 hours, 4 minutes and 28 seconds for the VQA due to its classical simulator backend.

As we can visually attest, the specific VQA we used model does not constitute a reasonable approximation to the numerical solution – given the hundreds of kilometers of deviation in Figures 4.11 and 4.12. Even though the period of the motion is visually the same in both approaches, the VQA method seems to overestimate the amplitude – especially in the x and z-axis. Consequently, when we analyze the full orbit, we observe the numerical output contained within the VQA result. Despite this amplitude difference, there is a region where both orbits coincide in Figure 4.12.

To complement our analysis, we shall consider the plot of the loss function *versus* the number of iterations. As indicated by Figure 4.13, there is a general decreasing trend of the loss function over the course of 5000 iterations, going from $\mathcal{L}_{3D,AD} = 2.540 \times 10^5$ to $\mathcal{L}_{3D,AD} = 8.328 \times 10^2$. This problem does not present any evident barren plateau behavior, but it has some slow variations between the 800^{th} and the 2200^{th} iteration and from the 2800^{th} iteration onward. As for the idea of neglecting the derivatives of the amplitudes ($A(t), B(t)$, and $C(t)$) conveyed by Equation 3.31, we obtain that $\tilde{\mathcal{L}}_{3D,AD} = 8.259 \times 10^2$, corresponding to a 0.8% loss function reduction, approximately. Therefore, neglecting such derivatives was a valid approximation in this case as well.

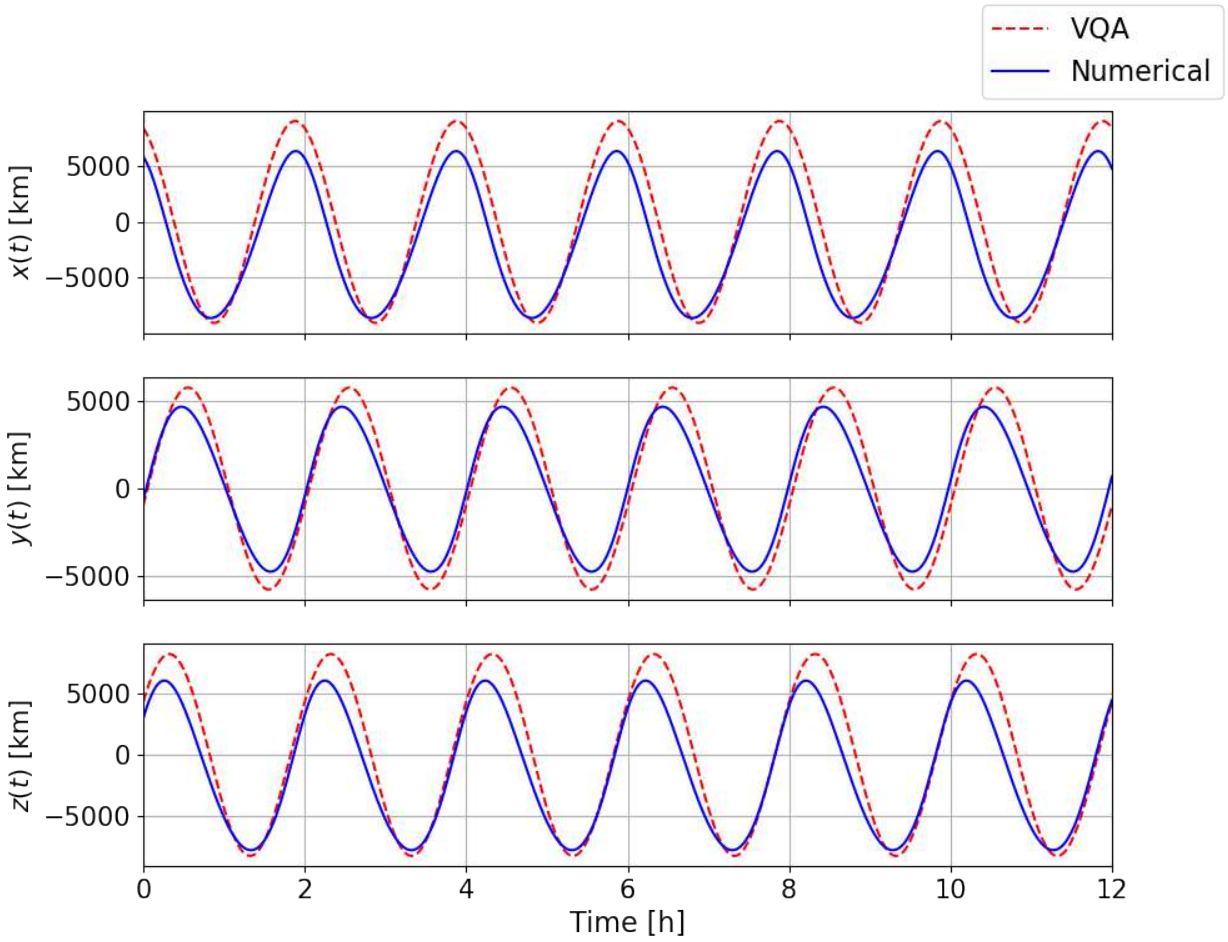

FIGURE 4.11 – Numerical (continuous blue line) and VQA's (dashed red line) components $(x(t), y(t), z(t))$ of the 3D orbit propagation with atmospheric drag as a function of time.

In addition to the previous comments, it is important to notice the rapid decrease of the loss function within the initial 800 iterations and between the 2200^{th} and 2800^{th} repetition. The latter case also presents a sudden oscillation that suggests that the Adam optimizer was exploring adjacent regions in a local minimum in the parameter landscape. Eventually, the optimizer chooses a direction associated with a steady and slow decrease of the loss function, as observed in the last portion of the Figure 4.13.

Finally, we investigate how the amplitude and phase estimates evolve throughout the simulation process (Figures 4.14 and 4.15). As expected, the MREC curves of the amplitudes have a general decreasing trend along the iterations. However, in all three axes, we see a sudden drop followed by an increase and slow decrease of their respective MREC values. We can explain such behavior by observing that the optimizer updates all parameters at each iteration. Therefore, it might find a particular value that reduces a given MREC abruptly; but since it keeps looking for new regions in the parameter landscape, the MREC will tend to grow again before it resumes its decreasing behavior.

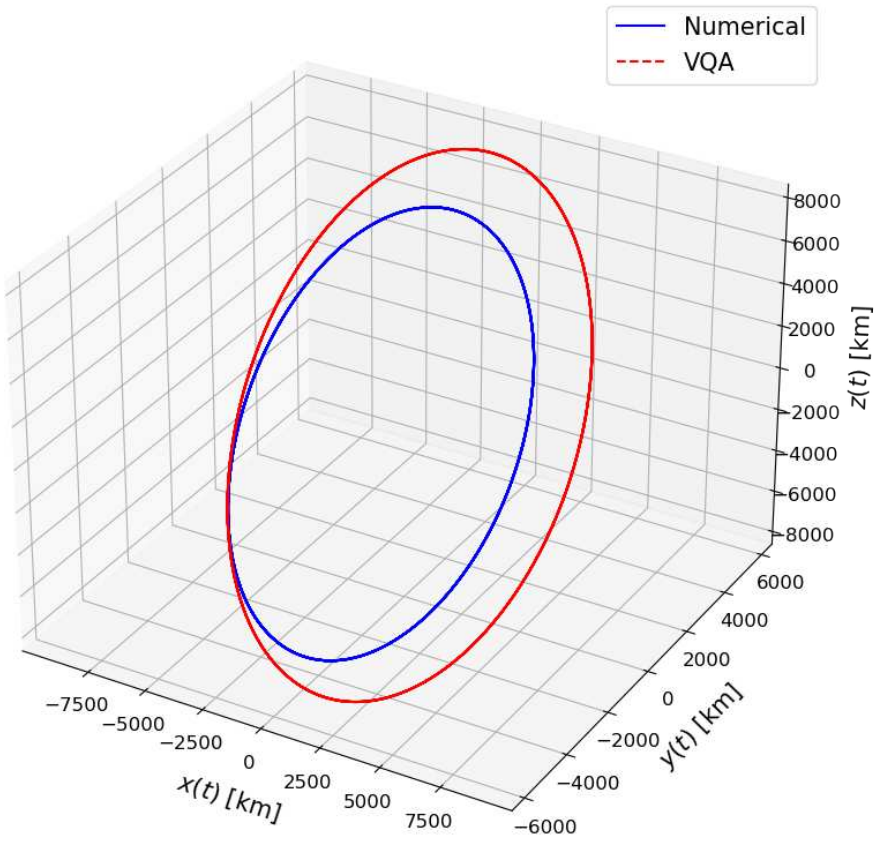

FIGURE 4.12 – Numerical (continuous blue line) and VQA's (dashed red line) 3D orbits in the presence of atmospheric drag.

Quantitatively, the lowest values these curves assume during the simulation process are $\mathrm{MREC}(A_x) = 1.332 \times 10^{-3}$, $\mathrm{MREC}(A_y) = 3.102 \times 10^{-4}$, and $\mathrm{MREC}(A_z) = 6.124 \times 10^{-4}$. Considering the previous discussion, we know these values are not the same as those obtained at the end of the simulation. After 5000 iterations, the MREC curves converge to 0.1217, 0.2308, and 0.1612 (for A_x, A_y, and A_z, respectively).

As for the MREC curves of the phases (Figure 4.15), we mainly observe a decreasing trend in $\mathrm{MREC}(\phi_x)$ and $\mathrm{MREC}(\phi_y)$. In both cases, we have a sudden decrease around the 2500^{th} iteration, followed by a slow reduction – coinciding with the loss function plot. This suggests that the parameters explored by the optimizer at that moment reduced $\mathrm{MREC}(\phi_x)$, $\mathrm{MREC}(\phi_y)$, and the loss function simultaneously. Moreover, there are some fluctuations in $\mathrm{MREC}(\phi_y)$ within the first 1000 iterations, which we might connect to the

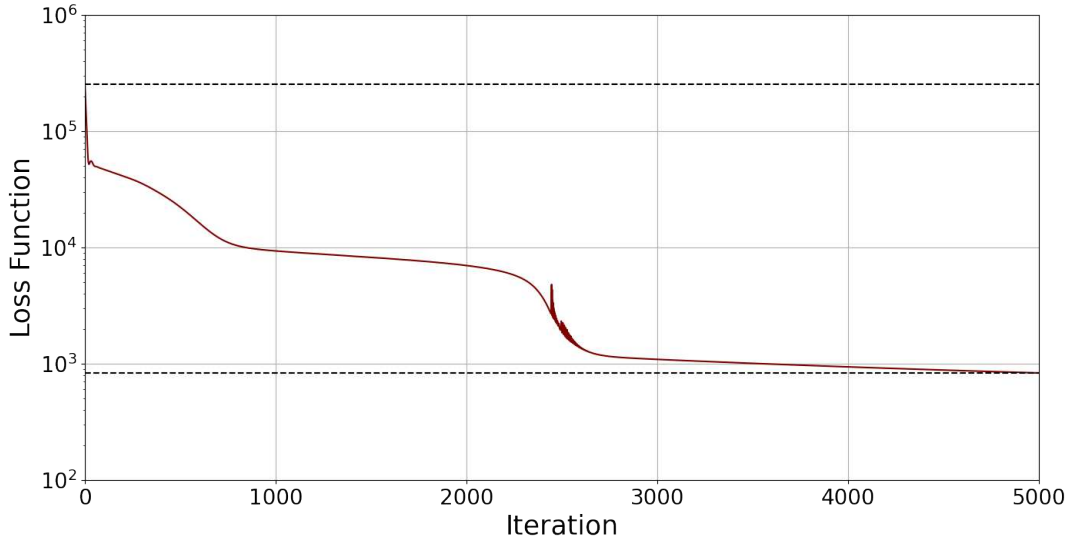

FIGURE 4.13 – Loss function *versus* the number of iterations for the 3D orbit propagation with atmospheric drag. The dotted lines mark the minimum and maximum values of the loss function.

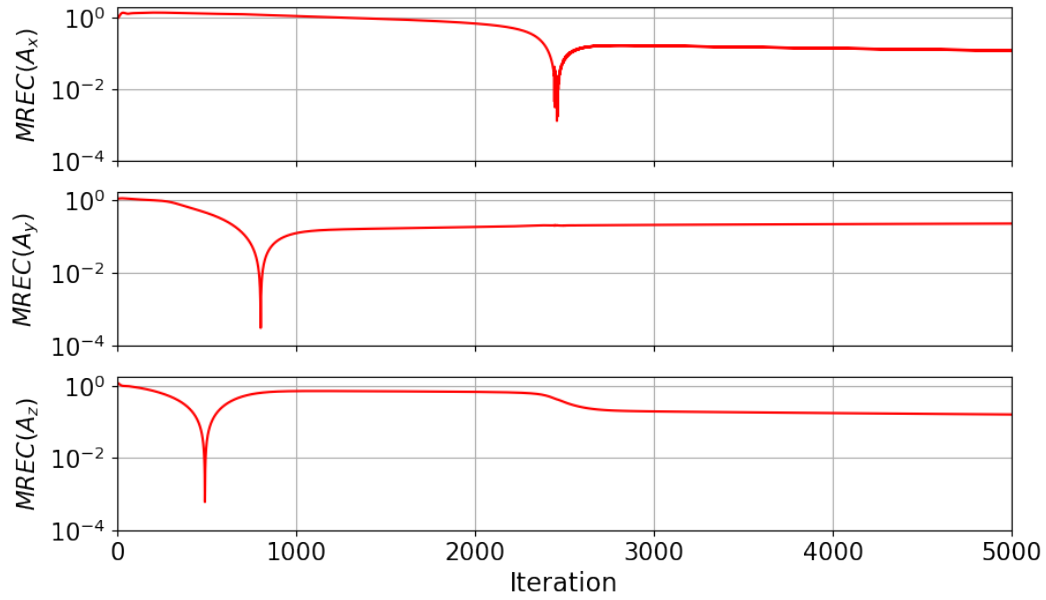

FIGURE 4.14 – Mean Relative Error per Cycle (MREC) of amplitudes as a function of the number of iterations for the 3D orbit propagation with atmospheric drag.

loss function decreasing during the same period.

Conversely, MREC(ϕ_z) does not present a clear decreasing trend. Instead, it has a relatively constant behavior except for a few abrupt reductions at the 200$^{\text{th}}$ and 2500$^{\text{th}}$ iterations, approximately. This observation indicates that the optimizer was unable to

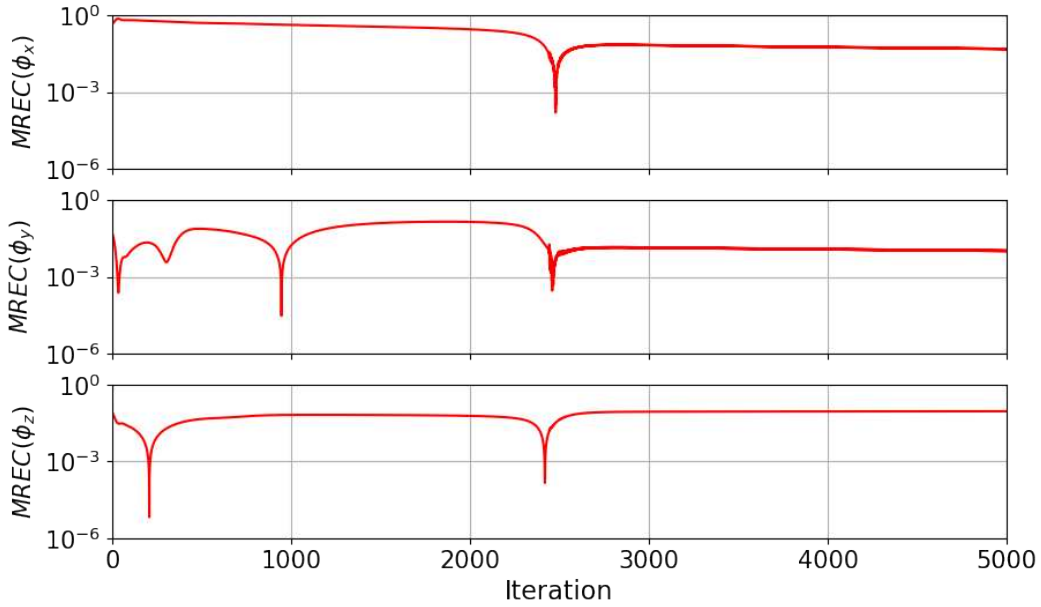

FIGURE 4.15 – Mean Relative Error per Cycle (MREC) of phases as a function of the number of iterations for the 3D orbit propagation with atmospheric drag.

consistently find parameters values that culminated in more accurate predictions of ϕ_z. Although Adam optimizer could sporadically locate some points that produced a better solution – represented by the two valleys in MREC(ϕ_z) plot – it probably needs more iterations to achieve a steady MREC reduction, as observed in MREC(ϕ_x) and MREC(ϕ_y).

Given those reasons, one should expect the value obtained at the end of the 5000 iterations to be relatively closer to the lowest value in the case of MREC(ϕ_x) and MREC(ϕ_y). Indeed, such lowest values are MREC(ϕ_x) $= 1.591 \times 10^{-4}$ and MREC(ϕ_y) $= 3.028 \times 10^{-5}$, and these MREC cuvers converge to 4.777×10^{-2} and 1.004×10^{-2} at the end of the process. On the other hand, MREC(ϕ_z) gets as low as 6.722×10^{-6} and converges to 9.065×10^{-2} after 5000 iterations.

We can draw some insights about this solution similar to section 4.2. In this case, the components A_x, A_z, ϕ_x, and ϕ_y became more accurate at a slow pace throughout the iterations; A_y and ϕ_z, on the other hand, remained stable during the entire process. We can deduce that the Adam optimizer was probably exploring a region with progressively lower values of A_x, A_z, ϕ_x, and ϕ_y while stagnating in A_y and ϕ_z.

As for the loss function, it decreases in a slight slope over 5000 iterations – not enough to obtain amplitudes and phases that would generate accurate solutions, as we can see in Figures 4.11 and 4.12. Again, in this case, the error in the order of hundreds of kilometers is superior to the acceptable in the literature for MEO missions (WERTZ *et al.*, 2011). Since the loss function continues in a decreasing trend around the 5000^{th} iteration, it is

possible to achieve more accurate results via more iterations. We could also try different optimizers (*e.g.*, PSO, GA) or represent the problem with fewer parameters.

5 Conclusions

This dissertation presented a hybrid method – partially quantum, partially classical – for performing orbit propagation of an artificial satellite around the Earth with perturbations. After explaining the fundamentals of Quantum Computing (in particular, Variational Quantum Algorithms), we defined the problem and derived the equations that describe the satellite motion considering the effects of the J_2 term and atmospheric drag in Appendix A.

We then described the underlying methodology of doing orbit propagation with perturbations via VQA. Our core idea relies on the fact that all problems refer to closed orbits, which have periodic projections in all axes. We use that principle to propose an *Ansatz* that the solution must be defined by a given amplitude and initial phase. Since the satellite's period is an input of the problem (obtained via direct observation), we only need to find the amplitude and phase for each axis to solve the problem.

The strategy is to start with a simplified version of the orbit propagation and then apply the same idea to the real problems we are aiming to solve. We started by solving the two-dimensional version of the gravitational perturbation problem, considering the effects of the J_2 term. The following steps were about using the same method for performing the actual 3D orbit propagation with J_2 effects and atmospheric drag separately. We also solve those three problems numerically in order to have an accuracy benchmark for the VQA solution.

After defining the problems and methodology, we proceeded to analyze the obtained results. In summary, the 2D orbit propagation with the effects of the J_2 term (section 4.1) constitutes a reasonable proof-of-concept. Even though it doesn't solve the problem within the acceptable tolerance errors, it could still be a valid method for less strict situations. The loss function was substantially reduced, but there was also a noticeable barren plateau, indicating the Adam optimizer got stuck in a local minimum for many iterations. Thus, in this case, there is no evidence that more iterations would produce better VQA's outputs.

Regarding the three-dimensional orbit propagation with the effects of the J_2 term or atmospheric drag (section 4.2 and 4.3, respectively), the loss function has not converged

to values as low as in the 2D case due to more parameters to be optimized in the 3D situation. The analysis of the amplitudes and phases showed that, although most of these variables were slowly converging to the target value, some were still stagnant (A_x in section 4.2, A_y and ϕ_z in section 4.3). Such behavior illustrates a limitation in the Adam optimizer's performance: as a gradient-descent algorithm, it explores only one region at a time, which might lead it to focus on a region that doesn't minimize all parameters involved. It is worth noticing that we haven't observed barren plateaus in the 3D cases. Moreover, the loss function's behavior in Figures 4.8 and 4.13 suggests more iterations could improve the solution's accuracy – although marginally, given its small slope.

Based on these results, we cannot prove if VQCs can satisfactorily perform orbit propagation with perturbations. In particular, we demonstrated that our specific combination of circuit architecture and classical optimizer is not able to do so. Nevertheless, there might be circuit architectures and optimizers (*e.g.*, PSO, GA) that could lead to orbit propagation with acceptable altitude fluctuations.

Therefore, there are a couple of ways to address those issues and to plan possible future directions of this work. The first idea is to use a different classical optimizer – preferentially PSO and GAs – for reasons explained in Chapter 4, section 4.1. By doing so, we could explore the parameter landscape (search space) more effectively, avoiding local minima and keep searching for the best solution globally. This approach could drastically reduce the number of iterations spent on barren plateaus.

Additional ideas may include performing more iterations. The loss function generally decreases with the increase in the number of iterations. Therefore, in principle, we could obtain more accurate solutions if we allow more than 5000 repetitions. However, one should be aware of the potential barren plateaus that might appear, as well as the feasibility of this approach, *i.e.*, a solution that requires hundreds of thousands of iterations will hardly have any practical value.

One could also model the problem with fewer parameters. This approach is highly promising because the convergence of the Adam optimizer depends directly on the number of parameters that we have in our VQC – thus explaining why the 2D model converges faster than the 3D one. The main difficulty is to find a way to build a circuit with fewer gates such that it represents all possible solutions without losing generality. Unless we have additional information about the problem, this task seems highly challenging.

Lastly, we should acknowledge that, as of 2023, quantum information is a vibrant field that is filled with new results and promising methods that are published at an accelerated pace. We hope this dissertation will inspire students to expand this use case of VQA and investigate further applications of quantum algorithms in applied and fundamental sciences.

Appendix A - Orbital Mechanics

In this Appendix, we proceed to derive the nonlinear differential equations that describe closed orbits with atmospheric drag and gravitational perturbations. We start by defining the inertial frame as the cartesian coordinate system XYZ, in which the origin O may move with constant velocity relative to the fixed stars, but the axes do not rotate. In this case, such a frame is also known as the geocentric equatorial frame (CURTIS, 2013).

Additionally, the moving frame of reference xyz is such that the origin O can rotate and translate freely. This frame is usually attached to a moving physical object – in this work, to an artificial satellite around the Earth in a closed orbit.

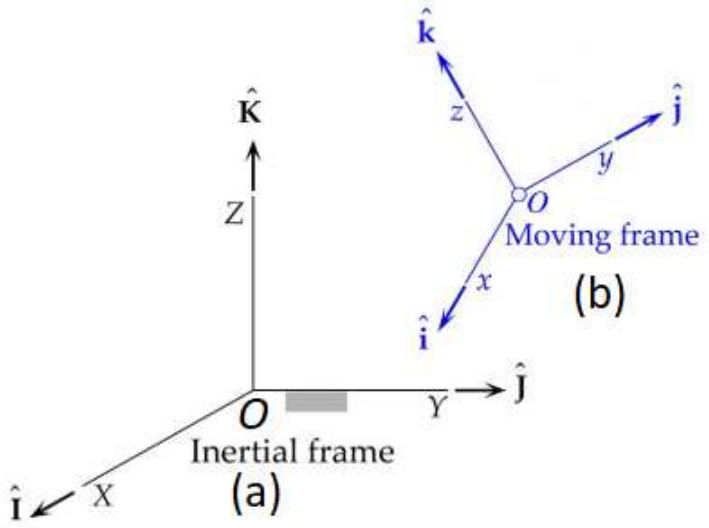

FIGURE A.1 – Diagrams corresponding to the (a) inertial frame and (b) moving frame (CURTIS, 2013).

According to Newton's law of gravitation, the motion of a body under the influence of a central gravity field is mathematically described as (FEHSE, 2003):

$$\vec{F}_G(\vec{r}) = -G\frac{Mm}{r^2}\frac{\vec{r}}{r} = -\mu\frac{m\vec{r}}{r^3}, \tag{A.1}$$

where $\vec{F}_G$ is the gravitational force, G is the universal gravitational constant ($G = 6,67 \times 10^{-11}$ $\mathrm{Nm^2/kg^2}$), M is the mass of the central body, m is the spacecraft's mass, $\vec{r}$ is the

radius vector ($\vec{r} = x\hat{i} + y\hat{j} + z\hat{k}$), and μ is the standard gravitational parameter ($\mu = GM$).

On the other hand, we can also write that

$$\vec{F}_G(\vec{r}) = m\vec{r}. \tag{A.2}$$

Comparing Equations A.1 and A.2, we deduce that

$$\ddot{\vec{r}} = -\mu\frac{\vec{r}}{r^3}. \tag{A.3}$$

We derived Equation A.3 assuming there are only those two objects in space (the Earth and a satellite around it) and the only interaction between them is due to their symmetric gravitational fields. Thus, any additional effect is considered an orbit perturbation, which may include the gravitational interaction with other bodies, atmospheric drag, solar radiation pressure, propulsive thrust, and effects of an oblate central body (CURTIS, 2013).

To represent such cases, we add a vector $\vec{p}$ to the right side of Equation A.3:

$$\ddot{\vec{r}} = -\mu\frac{\vec{r}}{r^3} + \vec{p}, \tag{A.4}$$

where $\vec{p}$ is the net perturbative acceleration considering all sources of orbit perturbation.

Generally, the perturbative acceleration p is significantly lower than the gravitational one ($a_0 = \mu/r^2$). For instance, at 1,000 km altitude, the relative magnitudes of the perturbative accelerations due the lunar gravity (p_{LG}), and solar radiation pressure (p_{RP}) are approximately (FORTESCUE *et al.*, 2011):

$$\begin{cases} p_{LG} = 10^{-7} a_0 \\ p_{RP} = 10^{-9} a_0 \end{cases}. \tag{A.5}$$

One possible exception to this trend is the atmospheric drag – it can easily deorbit satellites below 100 km of altitude. Nonetheless, its effects decrease rapidly and become negligible above 1,000 km, reaching $p_{drag} \approx 10^{-10} a_0$ (CURTIS, 2013).

We shall now derive the expression for the atmospheric drag's perturbative acceleration (CHOBOTOV, 2002). Starting with the spacecraft's relative velocity (relative to the atmosphere) $\vec{v}_{rel}$, we have that:

$$\vec{v}_{rel} = \vec{v} - \vec{v}_{atm}, \tag{A.6}$$

where $\vec{v}$ is the spacecraft's inertial velocity and $\vec{v}_{atm}$ is the inertial velocity of the atmosphere at that specific point.

Supposing that the atmosphere rotates with the Earth with angular velocity $\vec{\omega}_E$, we can also write that

$$\vec{v}_{atm} = \vec{\omega}_E \times \vec{r}, \tag{A.7}$$

where $\vec{v}_{rel}$ can be expressed in terms of its unitary vector $\hat{v}_{rel}$:

$$\vec{v}_{rel} = v_{rel}\hat{v}_{rel}. \tag{A.8}$$

In this context, the drag force $\vec{D}$ can be written as

$$\vec{D} = -D\hat{v}_{rel}, \tag{A.9}$$

where the magnitude D depends on the atmospheric density ρ, the spacecraft's frontal area A, and the drag coefficient C_D, such that

$$D = \frac{1}{2}\rho v_{rel}^2 C_D A. \tag{A.10}$$

Therefore, given a spacecraft with mass m, the atmospheric drag's perturbing acceleration $\vec{p}$ is given by $\vec{p} = \vec{D}/m$, *i.e.*,

$$\vec{p} = -\frac{1}{2}\rho v_{rel}\left(\frac{C_D A}{m}\right)\vec{v}_{rel}. \tag{A.11}$$

By substituting Equation A.11 into Equation A.4, and taking $B = C_D A/m$, we obtain the NLDE that describes the orbital motion of an artificial satellite around the Earth in the ECI frame under the effects of atmospheric drag (CHOBOTOV, 2002):

$$\ddot{\vec{r}} = -\mu\frac{\vec{r}}{r^3} - \frac{1}{2}\rho v_{rel} B\vec{v}_{rel}. \tag{A.12}$$

If we define the velocity vector as $\vec{v} = \{v_x, v_y, v_z\} = \{\dot{x}, \dot{y}, \dot{z}\}$, the relative velocity $\vec{v}_{rel}$ from Equation A.6 can be written as (CHOBOTOV, 2002):

$$\vec{v}_{rel} = \vec{v} - \vec{\omega}_E \times \vec{r} = (\dot{x} + y\omega_E)\hat{i} + (\dot{y} - x\omega_E)\hat{j} + \dot{z}\hat{k}, \tag{A.13}$$

where $\vec{\omega}_E = \omega_E\hat{k}$. Therefore, we can expand Equation A.12 as the following system of coupled equations in x, y and z:

$$\begin{cases} \ddot{x} = -\mu x/r^3 - (1/2)\rho v_{rel} B(\dot{x} + y\omega_E) \\ \ddot{y} = -\mu y/r^3 - (1/2)\rho v_{rel} B(\dot{y} - x\omega_E) \\ \ddot{z} = -\mu z/r^3 - (1/2)\rho v_{rel} B\dot{z} \end{cases}, \tag{A.14}$$

in which r and v_{rel} are given by:

$$\begin{cases} r = \sqrt{x^2 + y^2 + z^2} \\ v_{rel} = \sqrt{(\dot{x} + y\omega_E)^2 + (\dot{y} - x\omega_E)^2 + \dot{z}^2} \end{cases} . \tag{A.15}$$

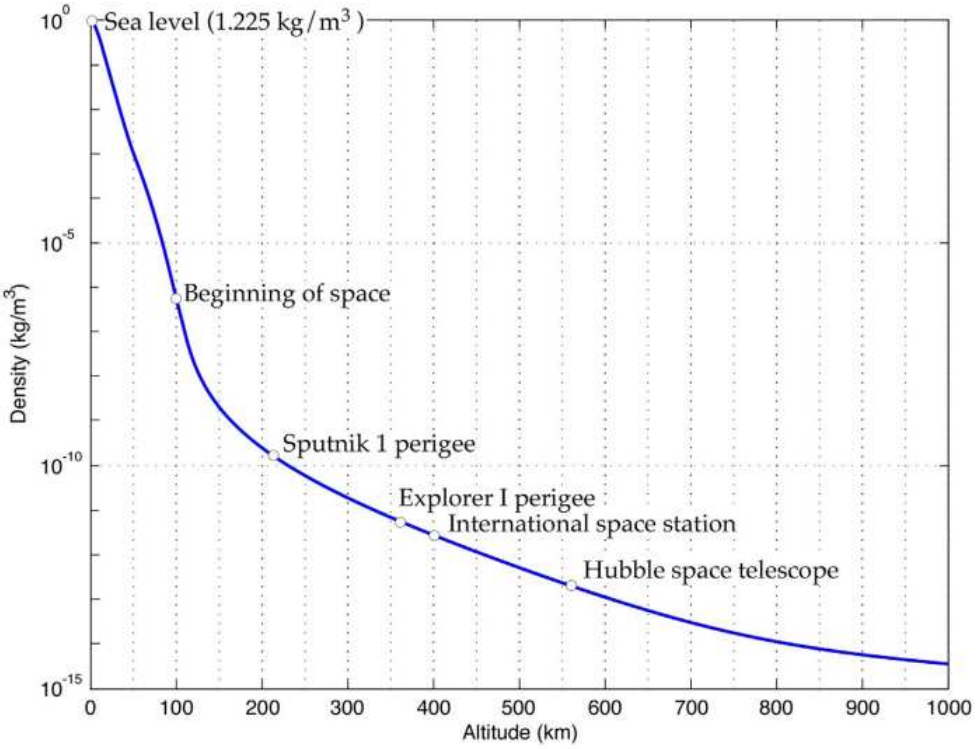

FIGURE A.2 – Atmospheric density as a function of altitude (ATMOSPHERE, 1976).

It is crucial to realize that the acceleration vector (Equation A.14) depends directly on the atmospheric density ρ. However, ρ itself is also a function of multiple factors, including the z-component. As represented by Figure A.2, the density substantially decreases as we increase the altitude. This work will assume a circular orbit with constant atmospheric density for simplicity.

In addition to the atmospheric drag, gravitational perturbations might also affect the orbit severely. This kind of perturbation derives from Earth's asymmetric mass distribution (CHOBOTOV, 2002). A perfect sphere would produce the following gravitational potential energy (per unit mass)

$$V(r) = -\frac{\mu}{r} \tag{A.16}$$

for a given point at a distance r outside of the sphere.

Like any other celestial body, the Earth is not a perfect sphere. Instead, it is an irregularly shaped ellipsoid (JEKELI, 1981), as illustrated by Figure A.3. This figure uses a spherical coordinate system with the origin O coinciding with the planet's center of mass. Moreover, the z-axis is the axis of rotational symmetry. Using such definitions, we

can compute the polar angle ϕ as

$$\phi = \arctan\left(\frac{\sqrt{x^2 + y^2}}{z}\right). \tag{A.17}$$

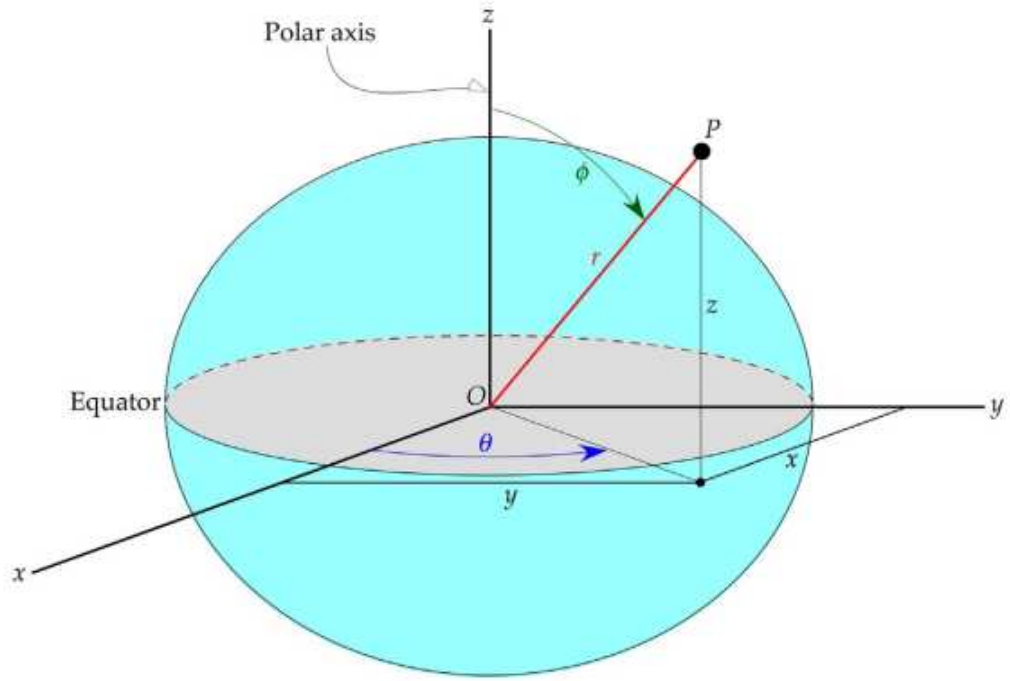

FIGURE A.3 – Representation of Earth's geometry (CURTIS, 2013).

We can determine the corresponding gravitational perturbation by acknowledging the Earth's asymmetric shape. Since the gravitational field is rotationally symmetric, it does not depend on the azimuth angle θ. Thus, we correct Equation A.16 by adding the gravitational perturbation term $f(r, \phi)$ due to the planet's oblateness:

$$V(r) = -\frac{\mu}{r} + f(r, \phi). \tag{A.18}$$

We can expand the perturbation $f(r, \phi)$ as the infinite series (BATTIN, 1999)

$$f(r, \phi) = \frac{\mu}{r} \sum_{k=2}^{\infty} J_k \left(\frac{R}{r}\right)^k P_k(\cos\phi), \tag{A.19}$$

where R is the Earth's equatorial radius, J_k is the k-th zonal harmonic, and P_k is the k-th Legendre polynomial.

Zonal harmonics are dimensionless values that can be measured from observations of satellites around a given planet. Since the coordinate system's center coincides with the planet's center of mass, we have that $J_1 = 0$. Table A.1 contains Earth's zonal harmonics

from J_2 to J_7 (VALLADO; MCCLAIN, 2007).

Zonal Harmonic	**Value**
J_2	1.08263×10^{-3}
J_3	-2.53266×10^{-6}
J_4	-1.61963×10^{-6}
J_5	-2.27298×10^{-7}
J_6	5.40676×10^{-7}
J_7	3.52364×10^{-7}

TABLE A.1 – Earth's zonal harmonics.

From J_8 and beyond, the zonal harmonics become more than three orders of magnitude smaller than J_2 (VALLADO; MCCLAIN, 2007). Therefore, the J_2 effects are predominant in this specific set of zonal harmonics. For this reason, we can approximate (CURTIS, 2013) the infinite series from Equation 2.20 to a single term when $k = 2$:

$$f(r, \phi) = \frac{\mu}{r} J_2 \left(\frac{R}{r}\right)^2 P_2(\cos\phi). \tag{A.20}$$

To compute Legendre polynomials, we use Rodrigues' formula (MCCARTHY *et al.*, 1993)

$$P_k(x) = \frac{1}{2^k k!} \frac{d^k}{dx^k}(x^2 - 1)^k, \tag{A.21}$$

with $k = 2$, to obtain

$$P_2(x) = \frac{1}{2}(3x^2 - 1). \tag{A.22}$$

Substituting Equation A.22 in Equation A.20, we derive the formula of perturbation corresponding to the J_2 effects, *i.e.*,

$$f(r, \phi) = \frac{\mu}{r} \frac{J_2}{2} \left(\frac{R}{r}\right)^2 (3\cos^2\phi - 1). \tag{A.23}$$

We then use Equation A.23 to compute the net perturbative acceleration vector $\vec{p}$ from Equation A.4. Such a vector is simply the negative gradient of $f(r, \phi)$:

$$\vec{p} = -\nabla f = -\frac{\partial f}{\partial x}\hat{i} - \frac{\partial f}{\partial y}\hat{j} - \frac{\partial f}{\partial z}\hat{k}. \tag{A.24}$$

Using the chain rule, we have that:

$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial r}\frac{\partial r}{\partial x} + \frac{\partial f}{\partial \phi}\frac{\partial \phi}{\partial x}, \tag{A.25}$$

and analogous expressions for y and z.

Calculating the partial derivatives from Equation A.25, we show that Equation A.24 becomes

$$\vec{p} = \frac{3}{2}\frac{J_2 \mu R^2}{r^4}\left[\frac{x}{r}\left(5\frac{z^2}{r^2}-1\right)\hat{i} + \frac{y}{r}\left(5\frac{z^2}{r^2}-1\right)\hat{j} + \frac{z}{r}\left(5\frac{z^2}{r^2}-3\right)\hat{k}\right]. \tag{A.26}$$

Therefore, the total acceleration (from Equation A.4) can be obtained by solving the following system of coupled differential equations:

$$\begin{cases} \ddot{x} = -\mu x/r^3 + (3/2)J_2\mu(R^2/r^4)(x/r)(5z^2/r^2-1) \\ \ddot{y} = -\mu y/r^3 + (3/2)J_2\mu(R^2/r^4)(y/r)(5z^2/r^2-1) \\ \ddot{z} = -\mu z/r^3 + (3/2)J_2\mu(R^2/r^4)(z/r)(5z^2/r^2-3) \end{cases}. \tag{A.27}$$

One should note that to obtain the differential equations that represent orbit propagation with both gravitational perturbations and atmospheric drag simultaneously, it is necessary to sum both perturbing acceleration vectors $\vec{p}$ from Equations A.11 and A.26. Thus, once we have the resulting perturbing vector, we substitute it in Equation A.4 and can write the acceleration in three components $x(t), y(t)$, and $z(t)$.

Additionally, we should mention one practical aspect of orbit propagation models. Equations A.14 and A.27 represent systems of second-order NLDEs, implying that one needs to know the initial conditions (position and velocity at a time instant) to solve such systems. Nonetheless, in practical terms, we gather different data types from the satellite's motion around the Earth.

The following discussion assumes the general case, *i.e.*, a satellite that describes an elliptic orbit (eccentricity e) around the Earth. Through observations, we can measure the orbit's perigee (r_p) and apogee radius (r_a) – the shortest and longest distance to Earth, respectively.

We then define the first Euler angle. It is called the right ascension of the right node (Ω), which is the angle between the positive X axis and the node line N, which is defined by the intersection of the orbital plane with the equatorial plane (XY). The second Euler angle is the inclination i, which we measure between the positive Z-axis and the normal to the plane of the orbit.

The third angle of Euler is the argument of perigee (ω), which we define as the angle between the node line vector $\vec{N}$ and the eccentricity vector $\vec{e}$, measured in the plane of the orbit. Finally, the true anomaly θ_{TA} is located between the satellite's position vector and the eccentricity vector. Figure A.4 is a graphic representation of these geometric relations.

Lastly, we define angular momentum h as the product between the satellite-Earth

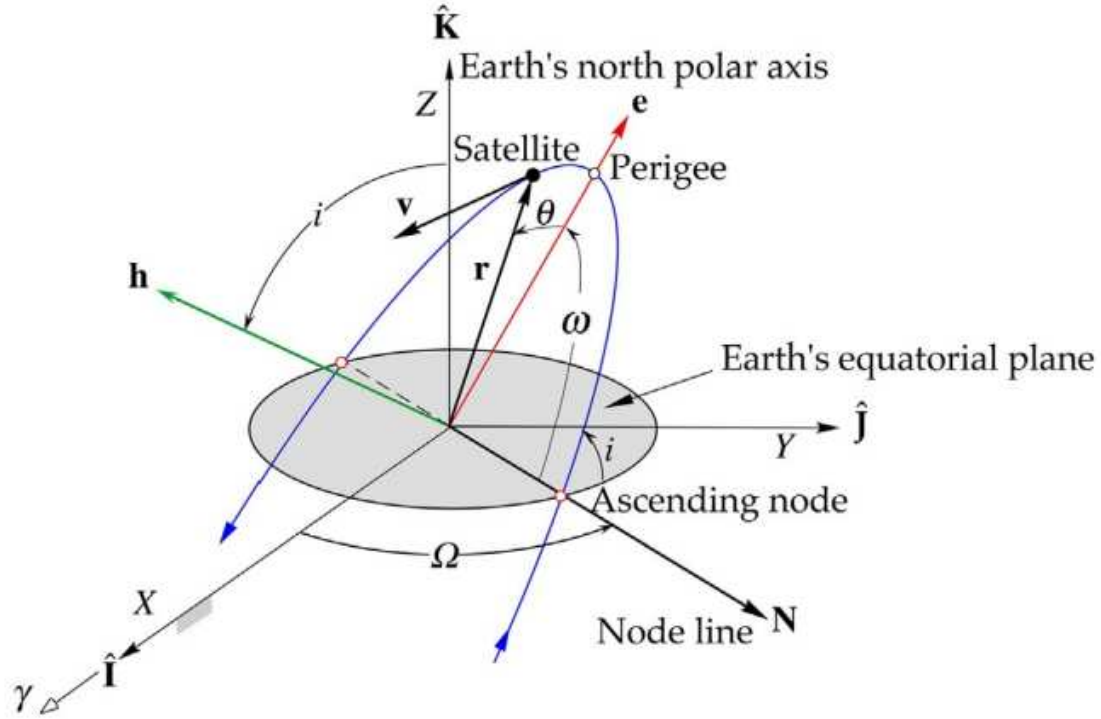

FIGURE A.4 – Diagram representing the geocentric equatorial frame and the orbital elements (CURTIS, 2013).

distance (r) and the radial component of velocity $v_\perp$. Mathematically, we can write that:

$$h = rv_\perp. \tag{A.28}$$

By doing so, we have discussed the six orbital elements (CHOBOTOV, 2002), *i.e.*, specific angular momentum (h), inclination (i), right ascension of the ascending node (Ω), eccentricity (e), the argument of perigee (ω), and true anomaly (θ_{TA}). As for the problems discussed in this dissertation, these are the inputs we use to determine the initial conditions of the NLDEs we wish to solve.

To obtain the initial conditions from the orbital elements, we must first define the perifocal frame in the context of an artificial satellite orbiting the Earth. The perifocal frame (CHOBOTOV, 2002) is a Cartesian coordinate system centered at the focus of the orbit (in this case, the Earth), and its plane (represented by $\overline{xy}$) coincides with the plane of the orbit. Its $\overline{x}$ axis is directed from the Earth to the perigee (periapsis), and the $\overline{y}$ axis is located at 90° true anomaly to the $\overline{x}$ axis. Finally, the $\overline{z}$ axis is normal to the plane of the orbit and has the same direction as the specific angular momentum vector $\vec{h}$. As illustrated in Figure A.5, the unit vectors of the $\overline{x}$, $\overline{y}$, and $\overline{z}$ axes are $\hat{p}, \hat{q}$, and $\hat{w}$, respectively.

Once we have defined the perifocal frame, we proceed to derivate the algebraic relations

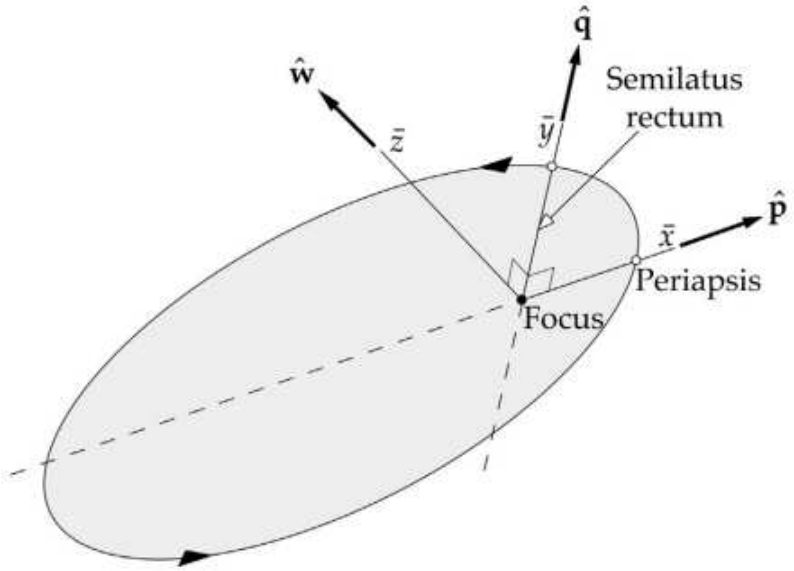

FIGURE A.5 – Perifocal frame representation (CURTIS, 2013).

that compute the orbit's initial conditions as a function of the orbital elements. The first step is to write the position ($\vec{r}_{\bar{x}}$) and velocity ($\vec{v}_{\bar{x}}$) vectors in the perifocal frame. It is possible to demonstrate (CURTIS, 2013) that such vectors are given by:

$$\vec{r}_{\bar{x}} = \frac{h^2}{\mu}\frac{1}{1+e\cos\theta_{TA}}\begin{pmatrix}\cos\theta_{TA}\\ \sin\theta_{TA}\\ 0\end{pmatrix}, \tag{A.29}$$

$$\vec{v}_{\bar{x}} = \frac{h}{\mu}\begin{pmatrix}-\sin\theta_{TA}\\ e+\cos\theta_{TA}\\ 0\end{pmatrix}. \tag{A.30}$$

After obtaining the position and velocity in the perifocal frame, we use a transformation matrix $Q_{\bar{x}X}$ that maps vectors from the perifocal to the geocentric equatorial coordinates. The transformation matrix is given by:

$$Q_{\bar{x}X} = \begin{pmatrix}Q_{11} & Q_{12} & Q_{13}\\ Q_{21} & Q_{22} & Q_{23}\\ Q_{31} & Q_{32} & Q_{33}\end{pmatrix}, \tag{A.31}$$

where

$$\begin{cases} Q_{11} = -\sin\Omega\cos i\sin\omega + \cos\Omega\cos\omega \\ Q_{12} = \cos\Omega\cos i\sin\omega + \sin\Omega\cos\omega \\ Q_{13} = \sin i\sin\omega \\ Q_{21} = -\sin\Omega\cos i\cos\omega - \cos\Omega\sin\omega \\ Q_{22} = \cos\Omega\cos i\cos\omega - \sin\Omega\sin\omega \\ Q_{23} = \sin i\cos\omega \\ Q_{31} = \sin\Omega\sin i \\ Q_{32} = -\cos\Omega\sin i \\ Q_{33} = \cos i \end{cases} . \tag{A.32}$$

We then obtain the position ($\vec{r}$) and velocity ($\vec{v}$) vectors in the geocentric equatorial frame by multiplying the transformation matrix and the corresponding vector in the perifocal coordinates. Mathematically,

$$\vec{r} = Q_{\bar{x}X}\vec{r}_{\bar{x}}, \tag{A.33}$$

$$\vec{v} = Q_{\bar{x}X}\vec{v}_{\bar{x}}. \tag{A.34}$$

Therefore, given the orbital elements at a time instant t_0, we can compute the satellite's position and velocity at that specific time. That information corresponds to the initial conditions that we need to solve the systems of NLDEs expressed in Equations A.14 and A.27.

Appendix B - Code

The following code shows the necessary packages that one needs to install in order to run the algorithms that we mention in this work.

```
python -m pip install -U pip
python -m pip install -U matplotlib

pip install numpy

pip install scipy

pip install pennylane --upgrade
```

Our first implementation is designed for solving the 2D gravitational perturbation problem.

```
#declaring packages and libraries
import pennylane as qml
import matplotlib.pyplot as plt
from pennylane import numpy as np
from pennylane.optimize import AdamOptimizer

#selecting the backend device
dev_fock = qml.device("strawberryfields.fock", wires=4, cutoff_dim=10)

#defining the VQC layer
def layer(v):
    qml.Beamsplitter(v[4], np.pi/2, wires=[0,1])
    qml.Kerr(v[5], wires=0)
    qml.Kerr(v[6], wires=1)
    qml.Beamsplitter(v[7], np.pi/2, wires=[2,3])
    qml.Kerr(v[8], wires=2)
    qml.Kerr(v[9], wires=3)
```

```
@qml.qnode(dev_fock, diff_method="parameter-shift")
def quantum_neural_net(var, x):
    # Encode input x into a quantum state
    qml.Displacement(v[0], 0.0, wires=0) #A
    qml.Displacement(v[1], 0.0, wires=1) #B
    qml.Displacement(v[2], 0.0, wires=2) #alpha
    qml.Displacement(v[3], 0.0, wires=3) #beta

    # "layer" subcircuits
    for v in var:
        layer(v)

    return [qml.expval(qml.X(0)), qml.expval(qml.X(1)), qml.expval(qml.X(2)),
        qml.expval(qml.X(3))]

#generating random quantum gates arguments
np.random.seed(0)
num_layers = 4
var_init = np.random.randn(num_layers, 10, requires_grad=True)
var = var_init

# Parameter values
mu = 398600*3600**2
J2 = 1082.63*10**(-6)
R = 6378 #km
w = 4.79632466196915 #rad/h

# Initial conditions
x0 = -2384.46 #km
vx0 = -7.36138*3600 #km/h
y0 = 5729.01 #km
vy0 = -2.98997*3600 #km/h
r0 = np.sqrt(x0**2 + y0**2) #km

#defining the variables: x(t) = A*cos(w*t+phi); y(t) = B*cos(w*t+theta)
def A(var, t):
    return x0*quantum_neural_net(var,t)[0]

def B(var, t):
    return y0*quantum_neural_net(var,t)[1]
```

```
def phi(var, t):
    return quantum_neural_net(var,t)[2]

def theta(var,t):
    return quantum_neural_net(var,t)[3]

def loss(var):
    #setting the corresponding loss functions to 0

    L = 0
    L0x = 0
    L0y = 0
    L1x = 0
    L1y = 0
    L2x = 0
    L2y = 0

    L0x = (x0 - (A(var,0)*np.cos(phi(var,0))))**2
    L0y = (y0 - (B(var,0)*np.cos(theta(var,0))))**2

    L1x = (vx0 - (-A(var,0)*w*np.sin(phi(var,0))))**2
    L1y = (vy0 - (-B(var,0)*w*np.sin(theta(var,0))))**2

    #performing the simulation for a total time of 12 hours
    for t in np.linspace(0, 12, 120):
        x = A(var,t)*np.cos(w*t + phi(var,t))
        y = B(var,t)*np.cos(w*t + theta(var,t))
        r = np.sqrt(x**2 + y**2)
        L2x = ((-mu*x/r**3 - 1.5*J2*mu*x*R**2/r**5) - (-A(var,t)*np.cos(w*t +
            phi(var,t))*w**2))**2
        L2y = ((-mu*y/r**3 - 1.5*J2*mu*y*R**2/r**5) - (-B(var,t)*np.cos(w*t +
            theta(var,t))*w**2))**2

    L = np.sqrt(L0x+L0y+L1x+L1y+L2x+L2y)
    return L

for it in range(5000):
    opt = AdamOptimizer(0.01, beta1=0.9, beta2=0.999)
    var_min = var_init
    loss_min = 10**15
```

```
    var, _loss = opt.step_and_cost(loss, var)
    if _loss < loss_min:
        loss_min = _loss #saving the lowest loss function
        var_min = var #saving the corresponding parameters

    print("Iter: {:5d} | Loss: {:0.7f}".format(it, _loss))
```

The following code solves 3D gravitational perturbation problem via VQC.

```
import pennylane as qml
import matplotlib.pyplot as plt
from pennylane import numpy as np
from pennylane.optimize import AdamOptimizer

dev_fock = qml.device("strawberryfields.fock", wires=6, cutoff_dim=10)

def layer(v):
    qml.Beamsplitter(v[6], np.pi/2, wires=[0,1])
    qml.Beamsplitter(v[7], np.pi/2, wires=[1,2])
    qml.Kerr(v[8], wires=0)
    qml.Kerr(v[9], wires=1)
    qml.Kerr(v[10], wires=2)
    qml.Beamsplitter(v[11], np.pi/2, wires=[3,4])
    qml.Beamsplitter(v[12], np.pi/2, wires=[4,5])
    qml.Kerr(v[13], wires=3)
    qml.Kerr(v[14], wires=4)
    qml.Kerr(v[15], wires=5)

@qml.qnode(dev_fock, diff_method="parameter-shift")
def quantum_neural_net(var, x):
    # Encode input x into a quantum state
    qml.Displacement(v[0], 0.0, wires=0) #A
    qml.Displacement(v[1], 0.0, wires=1) #B
    qml.Displacement(v[2], 0.0, wires=2) #C
    qml.Displacement(v[3], 0.0, wires=3) #alpha
    qml.Displacement(v[4], 0.0, wires=4) #beta
    qml.Displacement(v[5], 0.0, wires=5) #gamma

    # "layer" subcircuits
    for v in var:
        layer(v)
```

```
    return [qml.expval(qml.X(0)), qml.expval(qml.X(1)), qml.expval(qml.X(2)),
        qml.expval(qml.X(3)), qml.expval(qml.X(4)), qml.expval(qml.X(5))]

np.random.seed(0)
num_layers = 4
var_init = np.random.randn(num_layers, 16, requires_grad=True)
var = var_init

# Parameter values
mu = 398600*3600**2 #km^3/h^2
J2 = 1082.63*10**(-6)
R = 6378 #km
w = np.pi #rad/h

# Initial conditions
x0 = -2384.46 #km
vx0 = -7.36138*3600 #km/h
y0 = 5729.01 #km
vy0 = -2.98997*3600 #km/h
z0 = 3050.46 #km
vz0 = 1.64354*3600 #km/h
r0 = np.sqrt(x0**2 + y0**2 + z0**2) #km

#defining the variables: x(t) = A*cos(w*t+phi); y(t) = B*cos(w*t+theta); z(t)
    = C*cos(w*t+psi)
def A(var, t):
    return x0*quantum_neural_net(var,t)[0]

def B(var, t):
    return y0*quantum_neural_net(var,t)[1]

def C(var, t):
    return z0*quantum_neural_net(var,t)[2]

def phi(var, t):
    return quantum_neural_net(var,t)[3]

def theta(var,t):
    return quantum_neural_net(var,t)[4]
```

```
def psi(var, t):
    return quantum_neural_net(var,t)[5]

def loss(var):
    #setting the corresponding loss functions to 0

    L = 0

    L0x = 0
    L0y = 0
    L0z = 0

    L1x = 0
    L1y = 0
    L1z = 0

    L2x = 0
    L2y = 0
    L2z = 0

    #change this stuff
    L0x = (x0 - (A(var,0)*np.cos(phi(var,0))))**2
    L0y = (y0 - (B(var,0)*np.cos(theta(var,0))))**2
    L0z = (z0 - (C(var,0)*np.cos(psi(var,0))))**2

    L1x = (vx0 - (-A(var,0)*w*np.sin(phi(var,0))))**2
    L1y = (vy0 - (-B(var,0)*w*np.sin(theta(var,0))))**2
    L1z = (vz0 - (-C(var,0)*w*np.sin(psi(var,0))))**2

    for t in np.linspace(0, 12, 120):
        x = A(var,t)*np.cos(w*t + phi(var,t))
        y = B(var,t)*np.cos(w*t + theta(var,t))
        z = C(var,t)*np.cos(w*t + psi(var,t))

        r = np.sqrt(x**2 + y**2 + z**2)

        L2x = (((-mu*x)/(r**3.0) - (1.5*J2*mu*(R**2)/r**4.0)*(x/r)*(5*(z/r)**2
            - 1)) - (-x*w**2))**2
        L2y = (((-mu*y)/(r**3.0) - (1.5*J2*mu*(R**2)/r**4.0)*(y/r)*(5*(z/r)**2
            - 1)) - (-y*w**2))**2
        L2z = (((-mu*z)/(r**3.0) - (1.5*J2*mu*(R**2)/r**4.0)*(z/r)*(5*(z/r)**2
```

```
            - 3)) - (-z*w**2))**2

    L = np.sqrt(L0x+L0y+L0z+L1x+L1y+L1z+L2x+L2y+L2z)
    return L

%%time

opt = AdamOptimizer(0.01, beta1=0.9, beta2=0.999)
var_min = var_init
loss_min = 10**15

for it in range(5000):
    var, _loss = opt.step_and_cost(loss, var)
    if _loss < loss_min:
        loss_min = _loss
        var_min = var

    print("Iter: {:5d} | Loss: {:0.7f} | A: {:0.7f} | B: {:0.7f} | C:
        {:0.7f}".format(it, _loss, A(var,0), B(var,0), C(var,0)))
```

The next code solves 3D atmospheric drag problem via VQC.

```
import pennylane as qml
import matplotlib.pyplot as plt
from pennylane import numpy as np
from pennylane.optimize import AdamOptimizer

dev_fock = qml.device("strawberryfields.fock", wires=6, cutoff_dim=10)

def layer(v):
    qml.Beamsplitter(v[6], np.pi/2, wires=[0,1])
    qml.Beamsplitter(v[7], np.pi/2, wires=[1,2])
    qml.Kerr(v[8], wires=0)
    qml.Kerr(v[9], wires=1)
    qml.Kerr(v[10], wires=2)
    qml.Beamsplitter(v[11], np.pi/2, wires=[3,4])
    qml.Beamsplitter(v[12], np.pi/2, wires=[4,5])
    qml.Kerr(v[13], wires=3)
    qml.Kerr(v[14], wires=4)
    qml.Kerr(v[15], wires=5)
```

```
@qml.qnode(dev_fock, diff_method="parameter-shift")
def quantum_neural_net(var, x):
    # Encode input x into a quantum state
    qml.Displacement(v[0], 0.0, wires=0) #A
    qml.Displacement(v[1], 0.0, wires=1) #B
    qml.Displacement(v[2], 0.0, wires=2) #C
    qml.Displacement(v[3], 0.0, wires=3) #alpha
    qml.Displacement(v[4], 0.0, wires=4) #beta
    qml.Displacement(v[5], 0.0, wires=5) #gamma

    # "layer" subcircuits
    for v in var:
        layer(v)

    return [qml.expval(qml.X(0)), qml.expval(qml.X(1)), qml.expval(qml.X(2)),
        qml.expval(qml.X(3)), qml.expval(qml.X(4)), qml.expval(qml.X(5))]

np.random.seed(0)
num_layers = 4
var_init = np.random.randn(num_layers, 16, requires_grad=True)
var = var_init

# Parameter values
mu = 398600*3600**2 #km^3/h^2
J2 = 1082.63*10**(-6)
R = 6378 #km
w = np.pi #rad/h

# Initial conditions
x0 = -2384.46 #km
vx0 = -7.36138*3600 #km/h
y0 = 5729.01 #km
vy0 = -2.98997*3600 #km/h
z0 = 3050.46 #km
vz0 = 1.64354*3600 #km/h

#defining the variables: x(t) = A*cos(w*t+phi); y(t) = B*cos(w*t+theta); z(t)
    = C*cos(w*t+psi)

def A(var, t):
    return x0*quantum_neural_net(var,t)[0]
```

```
def B(var, t):
    return y0*quantum_neural_net(var,t)[1]

def C(var, t):
    return z0*quantum_neural_net(var,t)[2]

def phi(var, t):
    return quantum_neural_net(var,t)[3]

def theta(var,t):
    return quantum_neural_net(var,t)[4]

def psi(var, t):
    return quantum_neural_net(var,t)[5]

def loss(var):
    #setting the corresponding loss functions to 0

    L = 0

    L0x = 0
    L0y = 0
    L0z = 0

    L1x = 0
    L1y = 0
    L1z = 0

    L2x = 0
    L2y = 0
    L2z = 0

    L0x = (x0 - (A(var,0)*np.cos(phi(var,0))))**2
    L0y = (y0 - (B(var,0)*np.cos(theta(var,0))))**2
    L0z = (z0 - (C(var,0)*np.cos(psi(var,0))))**2

    L1x = (vx0 - (-A(var,0)*w*np.sin(phi(var,0))))**2
    L1y = (vy0 - (-B(var,0)*w*np.sin(theta(var,0))))**2
    L1z = (vz0 - (-C(var,0)*w*np.sin(psi(var,0))))**2
```

```
    for t in np.linspace(0, 12, 120):
        x = A(var,t)*np.cos(w*t + phi(var,t))
        y = B(var,t)*np.cos(w*t + theta(var,t))
        z = C(var,t)*np.cos(w*t + psi(var,t))

        r = np.sqrt(x**2 + y**2 + z**2)

        L2x = (((-mu*x)/(r**3.0) - (1.5*J2*mu*(R**2)/r**4.0)*(x/r)*(5*(z/r)**2
            - 1)) - (-x*w**2))**2
        L2y = (((-mu*y)/(r**3.0) - (1.5*J2*mu*(R**2)/r**4.0)*(y/r)*(5*(z/r)**2
            - 1)) - (-y*w**2))**2
        L2z = (((-mu*z)/(r**3.0) - (1.5*J2*mu*(R**2)/r**4.0)*(z/r)*(5*(z/r)**2
            - 3)) - (-z*w**2))**2

    L = np.sqrt(L0x+L0y+L0z+L1x+L1y+L1z+L2x+L2y+L2z)
    return L

%%time

opt = AdamOptimizer(0.01, beta1=0.9, beta2=0.999)
var_min = var_init
loss_min = 10**15

for it in range(5000):
    var, _loss = opt.step_and_cost(loss, var)
    if _loss < loss_min:
        loss_min = _loss
        var_min = var

    print("Iter: {:5d} | Loss: {:0.7f} | A: {:0.7f} | B: {:0.7f} | C:
        {:0.7f}".format(it, _loss, A(var,0), B(var,0), C(var,0)))
```

The following codes are the numerical solutions of the orbit propagation in the presence of perturbations. The first one is the numerical solution (via SciPy's ODEINT) of the 3D gravitational perturbation problem. The code solves and plots the solutions $x(t), y(t), z(t)$ as a function of time.

```
import scipy as sp
import numpy as np
from scipy.integrate import odeint
```

```
def vectorfield(var, t, p):
    """
    Defines the differential equations for the coupled spring-mass system.

    Arguments:
        var :  vector of the state variables:
                  var = [x, vx, y, vy, z, vz]
        t :  time
        p :  vector of the parameters:
                  p = [mu, k, w]
    """
    x, vx, y, vy, z, vz = var
    mu, J2, R = p

    r = np.sqrt(x**2 + y**2 + z**2)

    # Create f = (x',vx',y',vy',z',vz'):
    f = [vx,
         (-mu*x)/(r**3.0) - (1.5*J2*mu*(R**2)/r**4.0)*(x/r)*(5*(z/r)**2 - 1),
         vy,
         (-mu*y)/(r**3.0) - (1.5*J2*mu*(R**2)/r**4.0)*(y/r)*(5*(z/r)**2 - 1),
        vz,
         (-mu*z)/(r**3.0) - (1.5*J2*mu*(R**2)/r**4.0)*(z/r)*(5*(z/r)**2 - 3)]
    return f

# Parameter values
mu = 398600*3600**2 #km^3/s^2
J2 = 1082.63*10**(-6)
R = 6378 #km

# Initial conditions
x0 = -2384.46 #km
vx0 = -7.36138*3600 #km/h
y0 = 5729.01 #km
vy0 = -2.98997*3600 #km/h
z0 = 3050.46 #km
vz0 = 1.64354*3600 #km/h

# ODE solver parameters
abserr = 1.0e-8
relerr = 1.0e-6
```

```
days = 1
stoptime = 12.0
numpoints = 1200

# Create the time samples for the output of the ODE solver.
# I use a large number of points, only because I want to make
# a plot of the solution that looks nice.
t = [stoptime * float(i) / (numpoints - 1) for i in range(numpoints)]

# Pack up the parameters and initial conditions:
p = [mu, J2, R]
w0 = [x0, vx0, y0, vy0, z0, vz0]

# Call the ODE solver.
wsol = odeint(vectorfield, w0, t, args=(p,),
              atol=abserr, rtol=relerr)

f = open("3D_J2_real.txt", "w")
for t1, w1 in zip(t, wsol):
    f.write('{} {} {} {} {} {} {}'.format(t1, w1[0], w1[1], w1[2], w1[3],
        w1[4], w1[5]))
    f.write("\n")

f.close()

from numpy import loadtxt
import pylab
from pylab import figure, plot, xlabel, ylabel, grid, legend, title, savefig
from matplotlib.font_manager import FontProperties

t, x, vx, y, vy, z, vz = loadtxt('3D_J2_real.txt', unpack=True)
th = t/(3600)

figure(1, figsize=(6, 4.5))

xlabel('Time (hours)')
ylabel('Position (km)')
grid(True)
#hold(True)
lw = 1
```

```
plot(t, x, 'b', linewidth=lw)
plot(t, y, 'g', linewidth=lw)
plot(t, z, 'r', linewidth=lw)

legend((r'$x$', r'$y$', r'$z$'), prop=FontProperties(size=16))
title('Components of Relative Motion with J2 Effects')
savefig('3D_J2_real.png', dpi=100)
```

Lastly, the next code is the numerical solution of the orbit propagation with atmospheric drag. The code solves and plots the solutions $x(t), y(t), z(t)$ as a function of time.

```
import scipy as sp
import numpy as np
from scipy.integrate import odeint

def vectorfield(var, t, p):
    """
    Defines the differential equations for the coupled spring-mass system.

    Arguments:
        var :  vector of the state variables:
                  var = [x, vx, y, vy]
        t :  time
        p :  vector of the parameters:
                  p = [mu, B, w, rho]
    """
    x, vx, y, vy, z, vz = var
    mu, B, w, rho = p

    r = np.sqrt(x**2 + y**2 + z**2)
    vt = np.sqrt((vx + w*y)**2 + (vy - w*x)**2 + vz**2)

    # Create f = (x',vx',y',vy'):
    f = [vx,
         (-mu*x)/(r**3) - 0.5*rho*B*(vx + w*y)*vt,
         vy,
         (-mu*y)/(r**3) - 0.5*rho*B*(vy - w*x)*vt,
         vz,
         (-mu*z)/(r**3) - 0.5*rho*B*vz*vt]
    return f
```

```
# Parameter values
mu = 398600*3600**2 #km^3/s^2
rho = (10**(-13))*10**9 #kg/km^3 #use anything from 10**(-14) to 10**(-7)
    kg/m^3 -- 10**(-13) is the average value (Ex10.1 Curtis)
w = 72.9211*10**(-6)*3600 #rad/h
Cd = 2.2
A = 10**(-6)*np.pi*(0.5)**2 #km^2
m = 100 #kg
B = Cd*A/m

# Initial conditions
x0 = 5873.40 #km
vx0 = -2.89641*3600 #km/h
y0 = -658.522 #km
vy0 = 4.94010*3600 #km/h
z0 = 3007.49 #km
vz0 = 6.14446*3600 #km/h

# ODE solver parameters
abserr = 1.0e-8
relerr = 1.0e-6
days = 1
stoptime = 24.0
numpoints = 2400

# Create the time samples for the output of the ODE solver.
# I use a large number of points, only because I want to make
# a plot of the solution that looks nice.
t = [stoptime * float(i) / (numpoints - 1) for i in range(numpoints)]

# Pack up the parameters and initial conditions:
p = [mu, B, w, rho]
w0 = [x0, vx0, y0, vy0, z0, vz0]

# Call the ODE solver.
wsol = odeint(vectorfield, w0, t, args=(p,),
             atol=abserr, rtol=relerr)

f = open("3D_AD_real.txt", "w")
for t1, w1 in zip(t, wsol):
    f.write('{} {} {} {} {} {} {}'.format(t1, w1[0], w1[1], w1[2], w1[3],
```

```
        w1[4], w1[5]))
    f.write("\n")

f.close()

from numpy import loadtxt
import pylab
from pylab import figure, plot, xlabel, ylabel, grid, legend, title, savefig
from matplotlib.font_manager import FontProperties

t, x, vx, y, vy, z, vz = loadtxt('3D_AD_real.txt', unpack=True)

figure(1, figsize=(6, 4.5))

xlabel('Time (hours)')
ylabel('Position (km)')
grid(True)
#hold(True)
lw = 1

plot(t, x, 'r', linewidth=lw)
plot(t, y, 'g', linewidth=lw)
plot(t, z, 'b', linewidth=lw)

legend((r'$x$', r'$y$', r'$z$'), prop=FontProperties(size=16))
title('Components of Relative Motion with Atm Drag')
savefig('3D_AD_real.png', dpi=100)
```

Bibliography

AARONSON, S. **Quantum computing since Democritus**. [S.l.]: Cambridge University Press, 2013.

AMARO, D.; MODICA, C.; ROSENKRANZ, M.; FIORENTINI, M.; BENEDETTI, M.; LUBASCH, M. Filtering variational quantum algorithms for combinatorial optimization. **Quantum Science and Technology**, IOP Publishing, v. 7, n. 1, p. 015021, 2022.

AMES, A. E.; MATTUCCI, N.; MACDONALD, S.; SZONYI, G.; HAWKINS, D. M. Quality loss functions for optimization across multiple response surfaces. **Journal of Quality technology**, Taylor & Francis, v. 29, n. 3, p. 339–346, 1997.

ANDERSEN, U. L.; NEERGAARD-NIELSEN, J. S.; LOOCK, P. V.; FURUSAWA, A. Hybrid discrete-and continuous-variable quantum information. **Nature Physics**, Nature Publishing Group UK London, v. 11, n. 9, p. 713–719, 2015.

ATMOSPHERE, U. S. **US standard atmosphere**. [S.l.]: National Oceanic and Atmospheric Administration, 1976.

BARTLETT, S. D.; SANDERS, B. C. Universal continuous-variable quantum computation: Requirement of optical nonlinearity for photon counting. **Physical Review A**, APS, v. 65, n. 4, p. 042304, 2002.

BATTIN, R. H. **An introduction to the mathematics and methods of astrodynamics**. [S.l.]: Aiaa, 1999.

BAYM, G. **Lectures on quantum mechanics**. [S.l.]: CRC Press, 2018.

BENEDETTI, M.; REALPE-GÓMEZ, J.; PERDOMO-ORTIZ, A. Quantum-assisted helmholtz machines: A quantum–classical deep learning framework for industrial datasets in near-term devices. **Quantum Science and Technology**, IOP Publishing, v. 3, n. 3, p. 034007, 2018.

BENGIO, Y.; LAMBLIN, P.; POPOVICI, D.; LAROCHELLE, H. Greedy layer-wise training of deep networks. **Advances in neural information processing systems**, v. 19, 2006.

BENNETT, C. H.; BERNSTEIN, E.; BRASSARD, G.; VAZIRANI, U. Strengths and weaknesses of quantum computing. **SIAM journal on Computing**, SIAM, v. 26, n. 5, p. 1510–1523, 1997.

BERGHOLM, V.; IZAAC, J.; SCHULD, M.; GOGOLIN, C.; AHMED, S.; AJITH, V.; ALAM, M. S.; ALONSO-LINAJE, G.; AKASHNARAYANAN, B.; ASADI, A. *et al.* Pennylane: Automatic differentiation of hybrid quantum-classical computations. **arXiv preprint arXiv:1811.04968**, 2018.

BIAMONTE, J.; WITTEK, P.; PANCOTTI, N.; REBENTROST, P.; WIEBE, N.; LLOYD, S. Quantum machine learning. **Nature**, Nature Publishing Group UK London, v. 549, n. 7671, p. 195–202, 2017.

BOIXO, S.; ISAKOV, S. V.; SMELYANSKIY, V. N.; BABBUSH, R.; DING, N.; JIANG, Z.; BREMNER, M. J.; MARTINIS, J. M.; NEVEN, H. Characterizing quantum supremacy in near-term devices. **Nature Physics**, Nature Publishing Group, v. 14, n. 6, p. 595–600, 2018.

BOTTOU, L. Stochastic gradient descent tricks. **Neural Networks: Tricks of the Trade: Second Edition**, Springer, p. 421–436, 2012.

BOTTOU, L. *et al.* Stochastic gradient learning in neural networks. **Proceedings of Neuro-Nımes**, Nimes, v. 91, n. 8, p. 12, 1991.

CEREZO, M.; ARRASMITH, A.; BABBUSH, R.; BENJAMIN, S. C.; ENDO, S.; FUJII, K.; MCCLEAN, J. R.; MITARAI, K.; YUAN, X.; CINCIO, L. *et al.* Variational quantum algorithms. **Nature Reviews Physics**, Nature Publishing Group, v. 3, n. 9, p. 625–644, 2021.

CEREZO, M.; SONE, A.; VOLKOFF, T.; CINCIO, L.; COLES, P. J. Cost function dependent barren plateaus in shallow parametrized quantum circuits. **Nature communications**, Nature Publishing Group UK London, v. 12, n. 1, p. 1791, 2021.

CHIAVERINI, J.; LEIBFRIED, D.; SCHAETZ, T.; BARRETT, M. D.; BLAKESTAD, R.; BRITTON, J.; ITANO, W. M.; JOST, J. D.; KNILL, E.; LANGER, C. *et al.* Realization of quantum error correction. **Nature**, Nature Publishing Group, v. 432, n. 7017, p. 602–605, 2004.

CHOBOTOV, V. A. **Orbital mechanics**. [S.l.]: Aiaa, 2002.

CLERC, M. **Particle swarm optimization**. [S.l.]: John Wiley & Sons, 2010.

CURTIS, H. **Orbital mechanics for engineering students**. [S.l.]: Butterworth-Heinemann, 2013.

DELGADO, A.; ARRAZOLA, J. M.; JAHANGIRI, S.; NIU, Z.; IZAAC, J.; ROBERTS, C.; KILLORAN, N. Variational quantum algorithm for molecular geometry optimization. **Physical Review A**, APS, v. 104, n. 5, p. 052402, 2021.

DEUTSCH, D.; JOZSA, R. Rapid solution of problems by quantum computation. **Proceedings of the Royal Society of London. Series A: Mathematical and Physical Sciences**, The Royal Society London, v. 439, n. 1907, p. 553–558, 1992.

FEHSE, W. **Automated rendezvous and docking of spacecraft**. [S.l.]: Cambridge university press, 2003.

FERREIRA, R. P. Quantum algorithm based on the truncated taylor series for solving linearized equations of the relative motion. **Final paper (Undergraduation study)**, Instituto Tecnológico de Aeronáutica, p. 58, 2022.

FEYNMAN, R. P.; LEIGHTON, R. B.; SANDS, M. The feynman lectures on physics; vol. i. **American Journal of Physics**, American Association of Physics Teachers, v. 33, n. 9, p. 750–752, 1965.

FOCK, V. Näherungsmethode zur lösung des quantenmechanischen mehrkörperproblems. **Zeitschrift für Physik**, Springer, v. 61, p. 126–148, 1930.

FORTESCUE, P.; SWINERD, G.; STARK, J. **Spacecraft systems engineering**. [S.l.]: John Wiley & Sons, 2011.

GERJUOY, E. Shor's factoring algorithm and modern cryptography. an illustration of the capabilities inherent in quantum computers. **American journal of physics**, American Association of Physics Teachers, v. 73, n. 6, p. 521–540, 2005.

GROENEWOLD, H. J.; GROENEWOLD, H. J. **On the principles of elementary quantum mechanics**. [S.l.]: Springer, 1946.

GU, M.; WEEDBROOK, C.; MENICUCCI, N. C.; RALPH, T. C.; LOOCK, P. van. Quantum computing with continuous-variable clusters. **Physical Review A**, APS, v. 79, n. 6, p. 062318, 2009.

GUNER, O.; BEKIR, A. Solving nonlinear space-time fractional differential equations via ansatz method. **Computational methods for differential equations**, University of Tabriz, v. 6, n. 1, p. 1–11, 2018.

HAENSCH, W.; GOKMEN, T.; PURI, R. The next generation of deep learning hardware: Analog computing. **Proceedings of the IEEE**, IEEE, v. 107, n. 1, p. 108–122, 2018.

HARROW, A. W.; HASSIDIM, A.; LLOYD, S. Quantum algorithm for linear systems of equations. **Physical review letters**, APS, v. 103, n. 15, p. 150502, 2009.

HEINRICH, B. **Finite difference methods on irregular networks**. [S.l.]: Springer, 1987.

HOCHREITER, S.; BENGIO, Y.; FRASCONI, P.; SCHMIDHUBER, J. *et al.* **Gradient flow in recurrent nets: the difficulty of learning long-term dependencies**. [S.l.]: A field guide to dynamical recurrent neural networks. IEEE Press In, 2001.

JEKELI, C. **The downward continuation to the earth's surface of truncated spherical and ellipsoidal harmonic series of the gravity and height anomalies**. [S.l.]: The Ohio State University, 1981.

JOHNSON, N. L. Medium earth orbits: is there a need for a third protected region? In: **61st International Astronautical Congress**. [S.l.: s.n.], 2010.

JR, L. J. I. **Satellite communications systems engineering: atmospheric effects, satellite link design and system performance**. [S.l.]: John Wiley & Sons, 2017.

KAMATH, U.; LIU, J.; WHITAKER, J. **Deep learning for NLP and speech recognition**. [S.l.]: Springer, 2019.

KASIRAJAN, V. Dirac's bra-ket notation and hermitian operators. In: **Fundamentals of Quantum Computing: Theory and Practice**. [S.l.]: Springer, 2021. p. 35–73.

KENDON, V. M.; NEMOTO, K.; MUNRO, W. J. Quantum analogue computing. **Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences**, The Royal Society Publishing, v. 368, n. 1924, p. 3609–3620, 2010.

KENNEDY, J.; EBERHART, R. Particle swarm optimization. In: IEEE. **Proceedings of ICNN'95-international conference on neural networks**. [S.l.], 1995. v. 4, p. 1942–1948.

KERENIDIS, I.; PRAKASH, A. Quantum recommendation systems. **arXiv preprint arXiv:1603.08675**, 2016.

KILLORAN, N.; BROMLEY, T. R.; ARRAZOLA, J. M.; SCHULD, M.; QUESADA, N.; LLOYD, S. Continuous-variable quantum neural networks. **Physical Review Research**, APS, v. 1, n. 3, p. 033063, 2019.

KILLORAN, N.; IZAAC, J.; QUESADA, N.; BERGHOLM, V.; AMY, M.; WEEDBROOK, C. Strawberry fields: A software platform for photonic quantum computing. **Quantum**, Verein zur Förderung des Open Access Publizierens in den Quantenwissenschaften, v. 3, p. 129, 2019.

KINGMA, D. P.; BA, J. Adam: A method for stochastic optimization. **arXiv preprint arXiv:1412.6980**, 2014.

KOCZOR, B.; ENDO, S.; JONES, T.; MATSUZAKI, Y.; BENJAMIN, S. C. Variational-state quantum metrology. **New Journal of Physics**, IOP Publishing, v. 22, n. 8, p. 083038, 2020.

KONAR, D.; SARMA, A. D.; BHANDARY, S.; BHATTACHARYYA, S.; CANGI, A.; AGGARWAL, V. A shallow hybrid classical–quantum spiking feedforward neural network for noise-robust image classification. **Applied Soft Computing**, Elsevier, v. 136, p. 110099, 2023.

KÖNIGSMANN, H. J.; COLLINS, J. T.; DAWSON, S.; WERTZ, J. R. Autonomous orbit maintenance system. **Acta Astronautica**, Elsevier, v. 39, n. 9-12, p. 977–985, 1996.

KWAK, Y.; YUN, W. J.; JUNG, S.; KIM, J.-K.; KIM, J. Introduction to quantum reinforcement learning: Theory and pennylane-based implementation. In: IEEE. **2021 International Conference on Information and Communication Technology Convergence (ICTC)**. [S.l.], 2021. p. 416–420.

KYRIIENKO, O.; PAINE, A. E.; ELFVING, V. E. Solving nonlinear differential equations with differentiable quantum circuits. **Physical Review A**, APS, v. 103, n. 5, p. 052416, 2021.

LARA, M.; SAN-JUAN, J. F.; LÓPEZ, L. M.; CEFOLA, P. J. On the third-body perturbations of high-altitude orbits. **Celestial Mechanics and Dynamical Astronomy**, Springer, v. 113, p. 435–452, 2012.

LARSEN, M. V.; GUO, X.; BREUM, C. R.; NEERGAARD-NIELSEN, J. S.; ANDERSEN, U. L. Deterministic multi-mode gates on a scalable photonic quantum computing platform. **Nature Physics**, Nature Publishing Group UK London, v. 17, n. 9, p. 1018–1023, 2021.

LECUN, Y. 1.1 deep learning hardware: past, present, and future. In: IEEE. **2019 IEEE International Solid-State Circuits Conference-(ISSCC)**. [S.l.], 2019. p. 12–19.

LECUN, Y.; BENGIO, Y.; HINTON, G. Deep learning. **nature**, Nature Publishing Group UK London, v. 521, n. 7553, p. 436–444, 2015.

LINKS, J.; ZHOU, H.-Q.; MCKENZIE, R. H.; GOULD, M. D. Algebraic bethe ansatz method for the exact calculation of energy spectra and form factors: applications to models of bose–einstein condensates and metallic nanograins. **Journal of Physics A: Mathematical and General**, IOP Publishing, v. 36, n. 19, p. R63, 2003.

LIU, J.; NAJAFI, K.; SHARMA, K.; TACCHINO, F.; JIANG, L.; MEZZACAPO, A. Analytic theory for the dynamics of wide quantum neural networks. **Physical Review Letters**, APS, v. 130, n. 15, p. 150601, 2023.

LUBASCH, M.; JOO, J.; MOINIER, P.; KIFFNER, M.; JAKSCH, D. Variational quantum algorithms for nonlinear problems. **Physical Review A**, APS, v. 101, n. 1, p. 010301, 2020.

LUO, Y.-z.; YANG, Z. A review of uncertainty propagation in orbital mechanics. **Progress in Aerospace Sciences**, Elsevier, v. 89, p. 23–39, 2017.

MACKEY, D. S.; MACKEY, N. **On the determinant of symplectic matrices**. [S.l.]: Manchester Centre for Computational Mathematics Manchester, England, 2003.

MARTIN-LOPEZ, E.; LAING, A.; LAWSON, T.; ALVAREZ, R.; ZHOU, X.-Q.; O'BRIEN, J. L. Experimental realization of shor's quantum factoring algorithm using qubit recycling. **Nature photonics**, Nature Publishing Group UK London, v. 6, n. 11, p. 773–776, 2012.

MCCARTHY, P.; SAYRE, J.; SHAWYER, B. Generalized legendre polynomials. **Journal of mathematical analysis and applications**, Elsevier, v. 177, n. 2, p. 530–537, 1993.

MCCLEAN, J. R.; BOIXO, S.; SMELYANSKIY, V. N.; BABBUSH, R.; NEVEN, H. Barren plateaus in quantum neural network training landscapes. **Nature communications**, Nature Publishing Group UK London, v. 9, n. 1, p. 4812, 2018.

MEEKHOF, D.; MONROE, C.; KING, B.; ITANO, W. M.; WINELAND, D. J. Generation of nonclassical motional states of a trapped atom. **Physical review letters**, APS, v. 76, n. 11, p. 1796, 1996.

MICHAEL, M. H.; SILVERI, M.; BRIERLEY, R.; ALBERT, V. V.; SALMILEHTO, J.; JIANG, L.; GIRVIN, S. M. New class of quantum error-correcting codes for a bosonic mode. **Physical Review X**, APS, v. 6, n. 3, p. 031006, 2016.

MOLL, N.; FUHRER, A.; STAAR, P.; TAVERNELLI, I. Optimizing qubit resources for quantum chemistry simulations in second quantization on a quantum computer. **Journal of Physics A: Mathematical and Theoretical**, IOP Publishing, v. 49, n. 29, p. 295301, 2016.

MONTANARO, A.; PALLISTER, S. Quantum algorithms and the finite element method. **Physical Review A**, APS, v. 93, n. 3, p. 032324, 2016.

MORITA, S.; NISHIMORI, H. Mathematical foundation of quantum annealing. **Journal of Mathematical Physics**, American Institute of Physics, v. 49, n. 12, p. 125210, 2008.

MOUNT, E.; GAULTNEY, D.; VRIJSEN, G.; ADAMS, M.; BAEK, S.-Y.; HUDEK, K.; ISABELLA, L.; CRAIN, S.; RYNBACH, A. van; MAUNZ, P. *et al.* Scalable digital hardware for a trapped ion quantum computer. **Quantum Information Processing**, Springer, v. 15, p. 5281–5298, 2016.

MÜLLER, J.; ZEINHOFER, M. Error estimates for the deep ritz method with boundary penalty. In: PMLR. **Mathematical and Scientific Machine Learning**. [S.l.], 2022. p. 215–230.

NICHOLSON, A.; SLOJKOWSKI, S.; LONG, A.; BECKMAN, M.; LAMB, R. Nasa gsfc lunar reconnaissance orbiter (lro) orbit estimation and prediction. In: **SpaceOps 2010 Conference Delivering on the Dream Hosted by NASA Marshall Space Flight Center and Organized by AIAA**. [S.l.: s.n.], 2010. p. 2328.

NIELSEN, M. A.; CHUANG, I. **Quantum computation and quantum information**. [S.l.]: American Association of Physics Teachers, 2002.

NIU, Z.; ZHONG, G.; YU, H. A review on the attention mechanism of deep learning. **Neurocomputing**, Elsevier, v. 452, p. 48–62, 2021.

PAN, J.; CAO, Y.; YAO, X.; LI, Z.; JU, C.; CHEN, H.; PENG, X.; KAIS, S.; DU, J. Experimental realization of quantum algorithm for solving linear systems of equations. **Physical Review A**, APS, v. 89, n. 2, p. 022313, 2014.

PARRA-RODRIGUEZ, A.; LOUGOVSKI, P.; LAMATA, L.; SOLANO, E.; SANZ, M. Digital-analog quantum computation. **Physical Review A**, APS, v. 101, n. 2, p. 022305, 2020.

PERUZZO, A.; MCCLEAN, J.; SHADBOLT, P.; YUNG, M.-H.; ZHOU, X.-Q.; LOVE, P. J.; ASPURU-GUZIK, A.; O'BRIEN, J. L. A variational eigenvalue solver on a photonic quantum processor. **Nature communications**, Nature Publishing Group UK London, v. 5, n. 1, p. 4213, 2014.

PFISTER, O. Continuous-variable quantum computing in the quantum optical frequency comb. **Journal of Physics B: Atomic, Molecular and Optical Physics**, IOP Publishing, v. 53, n. 1, p. 012001, 2019.

PRESKILL, J. Quantum computing in the nisq era and beyond. **Quantum**, Verein zur Förderung des Open Access Publizierens in den Quantenwissenschaften, v. 2, p. 79, 2018.

PRIETO, D. M.; GRAZIANO, B. P.; ROBERTS, P. C. Spacecraft drag modelling. **Progress in Aerospace Sciences**, Elsevier, v. 64, p. 56–65, 2014.

RADHAKRISHNAN, K.; HINDMARSH, A. C. **Description and use of LSODE, the Livermore solver for ordinary differential equations**. [S.l.], 1993.

REN, S.; HE, K.; GIRSHICK, R.; ZHANG, X.; SUN, J. Object detection networks on convolutional feature maps. **IEEE transactions on pattern analysis and machine intelligence**, IEEE, v. 39, n. 7, p. 1476–1481, 2016.

ROMERO, J.; OLSON, J. P.; ASPURU-GUZIK, A. Quantum autoencoders for efficient compression of quantum data. **Quantum Science and Technology**, IOP Publishing, v. 2, n. 4, p. 045001, 2017.

SAFFMAN, M. Quantum computing with neutral atoms. **National Science Review**, Oxford University Press, v. 6, n. 1, p. 24–25, 2019.

SAINI, S.; RAPPLEYE, J.; CHANG, J.; BARKER, D.; MEHROTRA, P.; BISWAS, R. I/o performance characterization of lustre and nasa applications on pleiades. In: IEEE. **2012 19th International Conference on High Performance Computing**. [S.l.], 2012. p. 1–10.

SANTORO, G. E.; TOSATTI, E. Optimization using quantum mechanics: quantum annealing through adiabatic evolution. **Journal of Physics A: Mathematical and General**, IOP Publishing, v. 39, n. 36, p. R393, 2006.

SESHADREESAN, K. P.; OLSON, J. P.; MOTES, K. R.; ROHDE, P. P.; DOWLING, J. P. Boson sampling with displaced single-photon fock states versus single-photon-added coherent states: The quantum-classical divide and computational-complexity transitions in linear optics. **Physical Review A**, APS, v. 91, n. 2, p. 022334, 2015.

SHANKAR, T.; SHANMUGAVEL, S.; RAJESH, A. Hybrid hsa and pso algorithm for energy efficient cluster head selection in wireless sensor networks. **Swarm and Evolutionary Computation**, Elsevier, v. 30, p. 1–10, 2016.

SHOR, P. W. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. **SIAM review**, SIAM, v. 41, n. 2, p. 303–332, 1999.

SIMON, D. R. On the power of quantum computation. **SIAM journal on computing**, SIAM, v. 26, n. 5, p. 1474–1483, 1997.

SLUSSARENKO, S.; PRYDE, G. J. Photonic quantum information processing: A concise review. **Applied Physics Reviews**, AIP Publishing LLC, v. 6, n. 4, p. 041303, 2019.

SRINIVAS, M.; PATNAIK, L. M. Adaptive probabilities of crossover and mutation in genetic algorithms. **IEEE Transactions on Systems, Man, and Cybernetics**, IEEE, v. 24, n. 4, p. 656–667, 1994.

SVOZIL, D.; KVASNICKA, V.; POSPICHAL, J. Introduction to multi-layer feed-forward neural networks. **Chemometrics and intelligent laboratory systems**, Elsevier, v. 39, n. 1, p. 43–62, 1997.

TAKEDA, S.; FURUSAWA, A. Toward large-scale fault-tolerant universal photonic quantum computing. **APL Photonics**, AIP Publishing LLCAIP Publishing, v. 4, n. 6, p. 060902, 2019.

TILLY, J.; CHEN, H.; CAO, S.; PICOZZI, D.; SETIA, K.; LI, Y.; GRANT, E.; WOSSNIG, L.; RUNGGER, I.; BOOTH, G. H. *et al.* The variational quantum eigensolver: a review of methods and best practices. **Physics Reports**, Elsevier, v. 986, p. 1–128, 2022.

VALLADO, D.; MCCLAIN, W. Fundamentals of astrodynamics and applications, 2004. **Space Technology Library, Microcosm Press & Kluwer Academic Publishers**, p. 792, 2007.

VIRTANEN, P.; GOMMERS, R.; OLIPHANT, T. E.; HABERLAND, M.; REDDY, T.; COURNAPEAU, D.; BUROVSKI, E.; PETERSON, P.; WECKESSER, W.; BRIGHT, J. *et al.* Scipy 1.0: fundamental algorithms for scientific computing in python. **Nature methods**, Nature Publishing Group, v. 17, n. 3, p. 261–272, 2020.

WANG, D.; HIGGOTT, O.; BRIERLEY, S. Accelerated variational quantum eigensolver. **Physical review letters**, APS, v. 122, n. 14, p. 140504, 2019.

WANG, Q.; MA, Y.; ZHAO, K.; TIAN, Y. A comprehensive survey of loss functions in machine learning. **Annals of Data Science**, Springer, p. 1–26, 2020.

WANG, S.; FONTANA, E.; CEREZO, M.; SHARMA, K.; SONE, A.; CINCIO, L.; COLES, P. J. Noise-induced barren plateaus in variational quantum algorithms. **Nature communications**, Nature Publishing Group UK London, v. 12, n. 1, p. 6961, 2021.

WERTZ, J. R.; EVERETT, D. F.; PUSCHELL, J. J. Space mission engineering: the new smad. **(No Title)**, 2011.

WIEBE, N.; BRAUN, D.; LLOYD, S. Quantum algorithm for data fitting. **Physical review letters**, APS, v. 109, n. 5, p. 050505, 2012.

WIEBE, N.; GRANADE, C. Can small quantum systems learn? **arXiv preprint arXiv:1512.03145**, 2015.

WRIGHT, A. H. Genetic algorithms for real parameter optimization. In: **Foundations of genetic algorithms**. [S.l.]: Elsevier, 1991. v. 1, p. 205–218.

WU, B.-H.; ALEXANDER, R. N.; LIU, S.; ZHANG, Z. Quantum computing with multidimensional continuous-variable cluster states in a scalable photonic platform. **Physical Review Research**, APS, v. 2, n. 2, p. 023138, 2020.

XIN, T.; WEI, S.; CUI, J.; XIAO, J.; ARRAZOLA, I.; LAMATA, L.; KONG, X.; LU, D.; SOLANO, E.; LONG, G. Quantum algorithm for solving linear differential equations: Theory and experiment. **Physical Review A**, APS, v. 101, n. 3, p. 032307, 2020.

XU, X.; BENJAMIN, S. C.; YUAN, X. Variational circuit compiler for quantum error correction. **Physical Review Applied**, APS, v. 15, n. 3, p. 034068, 2021.

YAO, Y.-X.; GOMES, N.; ZHANG, F.; WANG, C.-Z.; HO, K.-M.; IADECOLA, T.; ORTH, P. P. Adaptive variational quantum dynamics simulations. **PRX Quantum**, APS, v. 2, n. 3, p. 030307, 2021.

YARKONI, S.; RAPONI, E.; BÄCK, T.; SCHMITT, S. Quantum annealing for industry applications: Introduction and review. **Reports on Progress in Physics**, IOP Publishing, 2022.

ZHANG, Z. Improved adam optimizer for deep neural networks. In: IEEE. **2018 IEEE/ACM 26th international symposium on quality of service (IWQoS)**. [S.l.], 2018. p. 1–2.

ZHAO, Z.-Q.; ZHENG, P.; XU, S.-t.; WU, X. Object detection with deep learning: A review. **IEEE transactions on neural networks and learning systems**, IEEE, v. 30, n. 11, p. 3212–3232, 2019.

FOLHA DE REGISTRO DO DOCUMENTO

1. CLASSIFICAÇÃO/TIPO	2. DATA	3. DOCUMENTO N	4. N DE PÁGINAS
DM	24 de julho de 2023	DCTA/ITA/DM-045/2023	101

5. TÍTULO E SUBTÍTULO:

Variational Quantum Algorithm for Orbit Propagation with Gravitational Perturbations and Atmospheric Drag

6. AUTOR(ES):

Rodrigo Pires Ferreira

7. INSTITUIÇÃO(ÕES)/ÓRGAO(S) INTERNO(S)/DIVISÃO(ÕES):

Instituto Tecnológico de Aeronáutica – ITA

8. PALAVRAS-CHAVE SUGERIDAS PELO AUTOR:

Quantum Computing; Orbital Mechanics; Quantum Algorithms; Differential Equations

9. PALAVRAS-CHAVE RESULTANTES DE INDEXAÇÃO:

Computação quântica; Mecânica orbital; Equações diferenciais; Algoritmos; Mecânica quântica; Perturbação orbital; Astronomia.

10. APRESENTAÇÃO: (**X**) **Nacional** ( ) **Internacional**

ITA, São José dos Campos. Programa de Pós-Graduação em Física. Área de Física Atômica e Molecular. Orientador: Prof. Dr. André Jorge Carvalho Chaves. Co-orientador: Prof. Dr. Willer Gomes dos Santos. Defesa em 12/07/2023. Publicado em 2023.

11. RESUMO:

Variational quantum algorithms (VQAs) are hybrid methods that use classical optimizers and quantum circuits to solve different minimization problems. Due to similarities with neural networks, these circuits might be called quantum neural networks, as they are composed of many layers of quantum gates whose parameters must be optimized to minimize a given loss function. This work's objective is to use a VQA for solving systems of coupled differential equations that describe the satellite's motion in the presence of orbit perturbations. We implement the VQA for propagating an orbit with atmospheric drag and gravitational perturbation considering the effects of the J_2 term – Earth's second zonal harmonic that's used to describe the variation of the gravitational potential due to the planet's oblate geometry. After solving a simplified two-dimensional version of the problem, we propagate the actual 3D orbit and compare the VQA solution to the numerical one. Finally, we discuss the existing challenges and opportunities of this method and how to improve it in future works.

12. GRAU DE SIGILO:

(**X**) **OSTENSIVO** ( ) **RESERVADO** ( ) **SECRETO**