RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET POPULAIRE
MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR ET DE LA RECHERCHE SCIENTIFIQUE

ÉCOLE NATIONALE POLYTECHNIQUE

المدرسة الوطنية المتعددة التقنيات
Ecole Nationale Polytechnique

مخبر التحكم في العمليات التصنيعية
Laboratoire de Commande des Processus

Département d'Automatique

Final Year Project
For the obtention of state engineer in Automatics diploma

Learning Algorithms Based State Estimation, Optimization and Control Of Nonlinear Processes

---

ABEDOU Abdelhadi & BENNACER Amine Rami
Under the direction of **Pr. TADJINE Mohamed** ENP

Presented and discussed publicly the (03/07/2024)

Composition of the jury:

President:	Pr. BOUCHERIT Mohamed Seghir	ENP
Promotor:	Pr. TADJINE Mohamed	ENP
Examiner:	Dr. ACHOUR Hakim	ENP
Examiner (Guest):	Pr. ZIOUI Nadjet	UQTA

ENP 2024

Département Automatique ENP, 10 Rue des Frères OUDEK, El Harrach 16200 Alger Algérie.

RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET POPULAIRE
MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR ET DE LA RECHERCHE SCIENTIFIQUE

ÉCOLE NATIONALE POLYTECHNIQUE

المدرسة الوطنية المتعددة التقنيات
Ecole Nationale Polytechnique

مخبر التحكم في العمليات التصنيعية
Laboratoire de Commande des Processus

Département d'Automatique

Final Year Project
For the obtention of state engineer in Automatics diploma

Learning Algorithms Based State Estimation, Optimization and Control Of Nonlinear Processes

---

ABEDOU Abdelhadi & BENNACER Amine Rami
Under the direction of **Pr. TADJINE Mohamed** ENP

Presented and discussed publicly the (03/07/2024)

Composition of the jury:

President:	Pr. BOUCHERIT Mohamed Seghir	ENP
Promotor:	Pr. TADJINE Mohamed	ENP
Examiner:	Dr. ACHOUR Hakim	ENP
Examiner (Guest):	Pr. ZIOUI Nadjet	UQTA

ENP 2024

Département Automatique ENP, 10 Rue des Frères OUDEK, El Harrach 16200 Alger Algérie.

RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET POPULAIRE
MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR ET DE LA RECHERCHE SCIENTIFIQUE

ÉCOLE NATIONALE POLYTECHNIQUE

Département d'Automatique

Mémoire de Projet de Fin d'Etudes

Pour l'obtention du diplôme d'ingénieur d'état en Automatique

---

Estimation d'Etat, Optimisation et Contrôle des Processus Nonlinéaires Basés sur des Algorithmes d'Apprentissage

---

ABEDOU Abdelhadi & BENNACER Amine Rami

Sous la direction de **Pr. TADJINE Mohamed** ENP

Présenté et soutenu publiquement le (03/07/2024)

Composition du jury:

Président:	Pr. BOUCHERIT Mohamed Seghir	ENP
Promoteur:	Pr. TADJINE Mohamed	ENP
Examinateur:	Dr. ACHOUR Hakim	ENP
Examinatrice (Invitée):	Pr. ZIOUI Nadjet	UQTA

ENP 2024

Département d'Automatique ENP, 10 Rue des Frères OUDEK, El Harrach 16200 Alger Algérie.

ملخص

التعليم الآلي (ML)، بما في ذلك التعلم العميق والتعلم التعزيزي، يوفر أدوات قوية لمعالجة المشاكل المعقدة. تستغل هذه المذكرة تعلم الآلة لتعزيز تقدير الحالة، وتحديد النظام، والتحسين في الأنظمة غير الخطية، حيث تفشل الأساليب التقليدية غالبًا. تشمل المجالات الرئيسية التركيز على تحسين الدقة في التقاط ديناميكيات الانظمة المعقدة، واستخراج خصائص النظام مباشرة من البيانات، وحل المشاكل غير المحدبة. تعرض المذكرة هذه الأساليب من خلال تطبيقات في ديناميكيات الطائرات وشبكات المستشعرات الذكية لتقنيات إنترنت الأشياء، مما يبرز إمكانية تعلم الآلة في تحسين الأداء والموثوقية وقابلية التكيف لأنظمة التحكم.

الكلمات المفتاحية: مركبة جوية بدون طيار - التجمد - LMI - الشبكات العصبية - التعريف المتناثر - إنترنت الأشياء - التحسين.

Résumé

L'apprentissage automatique (ML), y compris l'apprentissage profond et l'apprentissage par renforcement, offre des outils puissants pour résoudre des problèmes complexes. Cette thèse exploite le ML pour améliorer l'estimation d'état, l'identification de systèmes et l'optimisation dans les systèmes non linéaires, où les méthodes traditionnelles échouent souvent. Les domaines clés incluent l'amélioration de la précision dans la capture de la dynamique complexe des systèmes, l'extraction des caractéristiques du système directement à partir des données, et la résolution des problèmes non convexes. Cette thèse démontre ces méthodes à travers des applications dans la dynamique des avions et les réseaux de capteurs intelligents pour les technologies IoT, mettant en évidence le potentiel du ML pour améliorer la performance, la fiabilité et l'adaptabilité des systèmes de contrôle.

Mots-clés : Avion sans pilote - Givrage - LMI - Réseaux de Neurones - Identification Parcimonieuse - IoT - Optimisation.

Abstract

Machine learning (ML), including deep learning and reinforcement learning, offers powerful tools for addressing complex problems. This thesis leverages ML to enhance state estimation, system identification, and optimization in non-linear systems, where traditional methods often fall short. Key focus areas include improving accuracy in capturing complex system dynamics, extracting system characteristics directly from data, and solving non-convex problems. The thesis demonstrates these methods through applications in aircraft dynamics and smart sensor networks for IoT technologies, highlighting the potential of ML to enhance the performance, reliability, and adaptability of control systems.

Keywords : Unmanned Aerial Vehicle - Icing - LMI - Neural Networks - Sparse Identification - IoT - Optimization.

Dedication

It is not the destination that matters , what really matters is the journey it took to get there, a journey that is defined not by distance, but by the moments shared along the way with special people. This is a thanks to all those special people, those who made all of this possible, those who stood by us at our highest and lowest, those who saw us for who we are, even when we were not.

No person is whole without his family, so we can only start from there . To my dear mother, thank you for the love and affection, thank you for the words of comfort and wisdom, thank you for the years of endless support, thank you for giving my life purpose, thank you , knowing that words could never describe my love for you, in the hope that you'll always be proud of me, and that one day I might be able able to repay if only a little of everything you gave me.

To my beloved father, thank you for showing me what being a man means, thank you for shielding me when I was weak, for protecting me from this world's ugliness , thank you for lending me strength, for giving me a bed to rest, for being a warrior and a king. Thank you in the hopes that you'll always be by my side, not to depend on you, but to show you that I've become a strong man myself. My parents are the two pillars of my life, and I wouldn't be standing without them.

To my dear brothers Nassim and Fares, it is a blessing to have a big brother to look up to, and a little brother to guide , you two are my arms, growing up I always idolised my big bro, even trying to walk in his footsteps and replicate his success, and the more I go along the more I appreciate the efforts you made by understanding the hardships you took, and to my little bro, you have big potential and I think we hope that you become better than the both of us, be your own man, write your own story.

And finally to my grandma and deceased grandpa, mamas , didis, you are the suns of my life, you are my light and my everything, I wish you a long life mamas, and Allah yarhmek didis.

To my aunts , uncles and cousins, especially Abdou and Yanis thank you for growing up with me, for giving me all those precious memories that I will forever cherish.

Have you ever heard of the term brother from another mother? Yeah this sentence could not even begin to express my gratitude to these next persons, who made me who I am today and are my ride till die. To Aghiles, Abdelhak, Said, Yasser and Zaki, in those 3 short years we've already gone through so much, and I can only hope that we achieve more together, you 5 deserve the world and more, you 5 will achieve great things . Chakawi is not a group, it's an idea, the idea that with hard work and perseverance nothing is impossible, an idea that you may move faster alone, but that you move further with such people beside you.Thank you my brothers, to more success inshallah.

To my precious friends who made my days brighter, to Imed (and lpolo with him), Mehdi, Abdelhak, Mahmoud, Hicham, Yassine and Asma, thank you all for giving me many cheerful memories that I will forever treasure. To my other families, the Automatic family, Vic family , IEEE family, V family, ENSTP family and Blida people, thank you all for accepting me and making me part of a loving community.

And last but not least, I wanna thank me, I wanna thank me for being me, thank me for staying strong, thank me for staying true to me, thank me for supporting me, thank me for reminding me that I am not perfect and that I will never be, which is why I must always strive to better myself everyday. And before anything, alhamudillah, this is all part of Allah's will, the greatest of planners, may Allah guide us and protect all of these precious people.

Amine

Dedication

"To my dear parents,
dear siblings, my family, and my friends, especially Omar"

Abdelhadi

Acknowledgements

First of all, we thank **GOD** Almighty **Allah** for giving us the courage and patience for developing this modest work.

We would like to thank the esteemed members of the jury for accepting to review our project, as their remarks and directions will help us improve and better our current work.

Our deepest thanks go to our supervisor, Professor **Mohamed TADJINE**. His expertise, guidance, and mentorship have been invaluable throughout the entire project. His commitment to excellence and his willingness to share his knowledge has inspired us and shaped our academic growth. We are truly fortunate to have had the opportunity to work under his guidance.

Finally, we would like to thank all the individuals who directly or indirectly contributed to the successful completion of our project, especially our friend MAMECHE Omar. Their contributions, whether big or small, have played a significant role in shaping our understanding and achievements.

BENNACER and ABEDOU.

Contents

List of Tables

List of Figures

List of acronymes

- **Adam** : Adaptive Moment Estimation
- **AI** : Artificial Intelligence
- **CTRNN** : Continuous Time Recurrent Neural Network
- **DNN** : Deep Neural Network
- **DNN** : Deep Neural Network
- **GA** : Genetic Algorithm
- **GD** : Gradient Descent
- **GWO** : Grey Wolf Optimizer
- **IoT** : Internet of Things
- **LASSO** : Least Absolute Shrinkage and Selection Operator
- **LMI** : Linear Matrix Inequality
- **MAE** : Mean Absolute Error
- **MPC** : Model Predictive Control
- **MSE** : Mean Squared Error
- **NN** : Neural Network
- **PO** : Puma Optimizer
- **ReLU** : Rectified Linear Unit
- **RNN** : Recurrent Neural Network
- **SINDy** : Sparse Identification of Nonlinear Dynamics
- **SGD** : Stochastic Gradient Descent
- **SR3** : Sparse Relaxed Regularized Regression
- **STLSQ** : Sequential Threshold Least Squares
- **UAV** : Unmanned Aerial Vehicle
- **WOA** : Whale Optimization Algorithm

General Introduction

Learning methods such as machine learning, deep learning, reinforcement learning and metaheuristics algorithms have been the most exciting subject in the scientific community for a while now, mainly due to it's capability to learn from data, recognise complex patterns within it and make accurate prediction. It's popularity is justified, as it's applicability spans across many field and domains : natural language processing, image and speech recognition, and autonomous systems. Such potential has attracted the interest of control systems specialists, as learning methods offers a promising avenue for developing new and innovative solutions.

In the evolving landscape of control systems and engineering, the increasing complexity and nonlinearity of modern systems pose significant challenges. Traditional methods of state estimation, optimization, and system identification often fall short when dealing with highly nonlinear dynamics. This thesis addresses these challenges by leveraging the powerful capabilities of machine learning(ML) to enhance the accuracy and efficiency of optimization tasks, system identification and state estimation.

This thesis focuses on three core areas:

1. **State Estimation:** Accurate state estimation is crucial for monitoring and controlling dynamic systems. Traditional methods, such as the Kalman filter and its variants, perform well for linear systems but struggle with nonlinear systems. In this work, we propose an LMI-based nonlinear observer to address this challenge and demonstrate its application on aerial systems.

2. **System Identification:** Understanding the underlying dynamics of a system is essential for effective modeling and control. Traditional system identification methods are primarily suited for linear systems and rely heavily on precise mathematical modeling, which can be challenging to apply to nonlinear systems. Advanced learning techniques, such as recurrent neural networks and sparse learning methods, allow for the extraction of system characteristics directly from data. These techniques lead to more accurate and generalizable models, enhancing the ability to manage and control complex nonlinear systems.

3. **Metaheuristics for optimization:** Optimization plays a vital role in control system design and performance enhancement. However, nonlinear problems often lead to nonconvex optimization challenges, making them difficult to solve using conventional techniques. Metaheuristic approaches, which are widely regarded as machine learning methods, such as genetic algorithms and the grey wolf optimizer, offer innovative solutions for optimizing nonlinear problems by efficiently exploring large solution spaces and adapting to changing environments.

In our work, we will demonstrate these methods by applying them to real-world challenges. The thesis is divided into two parts:

1. **LMI Observers, Learning Based Identification: Application to Aircraft Dynamics:** In this part, we will address two key problems. The first is observer design and state estimation using Linear Matrix Inequalities approach, which will be applied to aerial systems with a specific focus on the issue of icing accretion on aircraft. The second problem is the system identification of nonlinear systems, also applied to the nonlinear dynamics of aerial systems. For this task, we will be testing two different identification methods (Continuous-Time Reccurent Neural Networks, Sparse Identification of Nonlinear Dynamics).

2. **Advances Optimization Of Smart Sensor Networks In Internet Of Things Technologies:** Our work involves implementing various metaheuristic algorithms, including the Grey Wolf Optimizer, Whale Optimizer, and Puma Optimizer, to solve a p-median-based modeling problem related to energy consumption in an Internet of Things (IoT) application. This study includes a performance comparison of the mentioned algorithms and demonstrates the efficiency of metaheuristics in this application.

Leveraging the strengths of machine learning tools in the fields of automation and control systems represents a significant paradigm shift in addressing complex systems. We will end each part of this thesis with conclusions that includes perspectives and possible future works.

Part I

LMI Observers, Learning Based Identification: Application to Aircraft Dynamics

Introduction

Aircraft icing has been a topic of research for many years, and that is mainly due to the danger it represents for the safety of aerial vehicles. In fact, National Transportation Safety Board findings showed that between 2008 and 2021 [7], there were at least 52 aircraft accidents and 64 fatalities that were attributed to icing as a cause. Most famously, we can find the Air Algérie Flight 5017 [8] in 2014, West Wind Aviation Flight 282 [9], the Myanmar Air Force Shaanxi Y-8 accident [10], and the Bek Air Flight 2100 in 2019 [11].

Ice accumulates on every exposed frontal surface of an aircraft. This includes wing and tail edges, propellers and spinners, engine intakes, landing gear, windshield, antennas, and other miscellaneous areas. The accumulation of ice on the tail and wings of a fixed UAV can alter its geometry and aerodynamic performance, thus changing the different forces and laws governing its flight, causing reduced lift, increased drag, and a higher risk of stalling, which leads to loss of control and eventually a plane crash.

Aircraft certified for flight in icing conditions are typically equipped with both de-icing systems, which remove accumulated ice, and anti-icing systems, which prevent ice from forming. For manned aircraft, the pilot is responsible for managing icing by monitoring the windscreen, wings, and other visible parts of the aircraft for ice formation. Additionally, most aircraft are not equipped with icing sensors. Various existing anti-icing and de-icing methods are employed for such aircraft, including anti-icing liquids, pneumatic systems, and electric-thermal systems.

However, current challenges require navigation through cloudy and icy weather conditions, such as those encountered during surveillance and reconnaissance operations. Due to their efficiency, UAVs (Unmanned Aerial Vehicles) are often preferred over manned aircraft for these missions. These unique requirements render traditional solutions unsuitable. UAVs typically operate at lower altitudes and speeds, resulting in longer exposure to icing conditions. The sensors needed for icing detection add extra weight and create aerodynamic constraints, and the lack of direct human intervention complicates the situation further, especially in fully autonomous operations.

With all of this, recent focus has been on finding new and reliable ways to detect ice formations in order to be able to deal with it properly and in time. From a control point of view, finding the variables affected by the icing and estimating the resulting changes is the main goal. Many works have been published regarding this aspect, from smart icing systems [12], to neural network-based solutions and model-based observers [13], which mainly focused on observing the changes of drag [14], lift (or other kinds of forces), or even the changes in the temperature gradient.

The main contribution of this part is the study of a new unprecedented variable, which is the aircraft's mass. Typically, in an unmanned aircraft, the mass is considered a constant, but in reality, it varies based on different parameters, mainly the fuel consumption (in the case of a fuel-powered aerial vehicles) and the ice's additional mass, thus making it an indicator of the icing accretion phenomena. Having this as a basis, we suggested a model for the change of

mass and augmented an already existing state-space model of the Jetstream J31 [15]. We have applied a similar approach to electrically-powered UAVs , precisely the AAI Aerosonde, where the primary variation in mass is due to ice accumulation, as these UAVs do not consume fuel.

Our main purpose is to implement learning methods to the aircraft system, taking into account two different aspects of control engineering: **Observer Design** and **System Identification**. The structure of this part will be as follows :

- **Chapter 1** Problem Formulation.
- **Chapter 2** LMI based Nonlinear Observer: Application to the Icing Accretion Problem.
- **Chapter 3** Learning Based Identification of Nonlinear systems: Application to Aircraft dynamics.

Literature Review

Works on ice detection have developed since the late 90's to early 20's. One of the first remarkable contributions to the literature is the work of Briggs et al. [12], in which they proposed a smart icing system (SIS) containing an ice protection system (IPS) and ice management system (IMS) .From then different methods followed, such as H_{inf} based parameter identification [16], ice severity diagnosis methods [17] and analysis of ice-affected variables. Fault tolerant control (FTC) has been wildely investigated in the field, in addition to the use of sliding mode observers, and more recently Super twisting algorithms (STA) were used by L.Chen and James F.Whidborne [14] in order to observe changes in drag caused by icing accumulations.

Machine learning and deep learning methods have been introduced to this problem for a while now, we can mention the work of Fikret Caliskan et al [18] in which a kalman filter was developed to feed a NN that predicted parameters affected by icing, or Yiqun Dong [19] where parameter identification was done using DNN's and compared to standard H_{inf}, identification.

From the observation point of view, recent advances in linear observers include adaptive and robust versions of the Luenberger observer and Kalman filter. These enhancements aim to improve performance under varying operating conditions and in the presence of unmodeled dynamics and noise. Adaptive observers adjust the observer gain in real-time based on the observed data, enhancing their applicability in systems with time-varying parameters. In nonlinear observers, significant progress has been made in developing robust and adaptive high-gain and sliding mode observers. These include: Adaptive High-Gain Observers: These observers adjust the gain dynamically to balance between fast convergence and noise sensitivity, and Integral Sliding Mode Observers: Integrating sliding modes with integral action to reduce steady-state estimation errors. Research also focuses on observer-based fault detection and isolation (FDI) systems, where observers are used to monitor system health and detect anomalies.

In the other hand, Neural networks are being developed for the purpose of system identification, observer design, and feedback control. Recurrent neural networks (RNN) have been utilised for their efficiency with mimicking dynamical systems and for their robustness, and works have shown their utility in approximating nonlinear system behaviour. In [20] [21] [22] RNN's have been trained for the purpose of identifying and observing nonlinear changes in a chemical reaction, as well as using the trained model to implement an LMPC controller. The methods's computational implementation, limits and constraints have also been discussed in these papers.

Another powerful tool is the Sparse Identification of Nonlinear Dynamics (SINDy) algorithm [23][24].This algorithm allows for the reconstruction of the equations governing a dynamical system directly from data. It is based on the premise that many physical systems can be described by a few active terms from a larger set of potential functions. This approach leverages the sparsity of the governing equations, enabling efficient and accurate model discovery even in the presence of complex nonlinear interactions. SINDy has been successfully applied in various fields, including fluid dynamics, biological systems and control theory, demonstrating its effectiveness in uncovering the underlying structure of nonlinear dynamical systems.

Chapter 1

Problem Formulation

In this subsection, we will develop the equations for the two systems: the Jetstream J3 Aircraft and the AAI Aerosonde. The first system is a fuel-powered UAV with a weight of 6073 kg, and the second system is an electrically powered UAV with a weight of 13.5 kg. We will extend these equations to include the additional equations of mass change. We will also highlight all assumptions made to facilitate the work done in the following sections. Finally, we will conclude the subsection by emphasizing the objectives and goals of this chapter.

1.1 System Description

1.1.1 Fuel Powered UAV (Jetstream J3)

When dealing with icing-related research problems, an interesting variable to consider is the change in mass due to the accumulation of ice. In what follows, we describe the aircraft's mathematical model and all its variables.

We begin with drag. Drag is the aerodynamic force that opposes an aircraft's motion through the air. It is generated by the interaction of the aircraft's surfaces with the air. It can be influenced by many factors, including the shape of the aircraft, airspeed, and the accumulation of ice on the aircraft's surfaces.

The expression of drag is as follows:

$$Cd = C_{D0} + k.C_L^2 \tag{1.1}$$

With C_{D0} being the known zero-lift coefficient, C_L represents the lift coefficient and k is the lift-independent drag coefficient factor obtained experimentally from flight tests [15].

Assumption 1:

For simplicity purposes, we consider the flight during steady flight only.

During a steady level flight, the aircraft lift is $L = mg$ and thus C_L is:

$$C_L = \frac{L}{0.5\rho_0.S.V_T^2} = \frac{m.g}{0.5\rho_0.S.V_T^2} \tag{1.2}$$

Where the parameter ρ_0 is the air density of International Standard Atmosphere (ISA) at sea level, S denotes the wing area and V_T represents true airspeed. Therefore:

$$Cd = C_{D0} + k.(\frac{m.g}{0.5\rho_0.S.V_T^2})^2 \tag{1.3}$$

The change of drag of an aircraft is expressed as follows:

$$\delta Cd = Cd_{act} - Cd_{exp} \tag{1.4}$$

Where Cd_{act} is the actual drag coefficient (drag influenced by icing accretion) and Cd_{exp} represents the expected drag coefficient (drag in absence of icing).

In addition, to the change of drag, one of our main goals is to observe the effect of icing on the aircraft's mass, thus we make the following distinction between expected and actual drag:

$$Cd_{exp} = C_{D0} + k.(\frac{m_{exp}.g}{0.5\rho_0.S.V_T^2})^2 \tag{1.5}$$

$$Cd_{act} = C_{D0} + k.(\frac{m_{act}.g}{0.5\rho_0.S.V_T^2})^2 \tag{1.6}$$

Where Cd_{exp} and m_{exp} are the expected drag and mass, while Cd_{act} and m_{act} are the actual drag and mass. For what follows we consider $m_{act} = m$ (Kg).

The crucial longitudinal nonlinear dynamic equation of the fixed wing aircraft during a steady level flight is

$$\dot{V}_T = \frac{F_x.cos\alpha + F_z.sin\alpha}{m} \tag{1.7}$$

$$\dot{\alpha} = \frac{-F_x.sin\alpha + F_z.cos\alpha}{m} + q \tag{1.8}$$

Where F_x, F_z are the forces applied on the x and z axis respectively, while q is the pitch rate.

Substituting the body-axis aerodynamic forces into equations (1.7) and (1.8) yields to [14]:

$$\dot{V}_T = -\frac{\rho_0.V_T^2.S}{2m}.Cd_{act} + \frac{T_{tol}}{m}.cos\alpha + g.sin(\alpha - \theta) \tag{1.9a}$$

$$\dot{\alpha} = -\frac{T_{tol}}{m.V_T}.sin\alpha + \frac{g}{V_T}.(cos(\alpha - \theta) - 1) + q \tag{1.9b}$$

$$\dot{q} = 0 \tag{1.9c}$$

Where V_t is the true airspeed (kts), α is the angle of attack (rad) and T_{tol} is the applied thrust (N). (The rest of parameters are defined in Appendix A (6.6))

Since the model represents the steady flight of an aircraft, the pitch angle remains constant. Consequently, the pitch rate and its derivative are equal to zero, $q = \dot{q} = 0$.

The mathematical model of the steady flight of an aircraft, becomes as follows :

$$\dot{V}_T = -\frac{\rho_0 . V_T^2 . S}{2m} . Cd_{act} + \frac{T_{tol}}{m} . cos\alpha + g . sin(\alpha - \theta) \tag{1.10a}$$

$$\dot{\alpha} = -\frac{T_{tol}}{m . V_T} . sin\alpha + \frac{g}{V_T} . (cos(\alpha - \theta) - 1) \tag{1.10b}$$

Remark 1:

We acknowledge that the aircraft's mass changes over time due to fuel consumption, resulting in a decrease in mass, and due to icing accretion, resulting in an increase in mass.

We propose the following dynamics for the mass, representing the change in mass due to fuel consumption, in absence of icing:

$$\dot{m} = a \cdot V_T \cdot (m - m_f) \tag{1.11}$$

- Here, a is a negative constant that varies with respect to speed values. Increasing V_T leads to greater fuel consumption, while decreasing V_T results in less fuel consumption.
- The value of a can be obtained either from a LookUp Table or modeled using fuzzy logic to ensure continuity, unlike the LookUp Table approach.
- In this application, a was fitted from available data using the convex optimization toolbox in MATLAB.

Finally the final system is described using the following equations:

$$\dot{V}_T = -\frac{\rho_0 . V_T^2 . S}{2m} . Cd_{act} + \frac{T_{tol}}{m} . cos\alpha + g . sin(\alpha - \theta) \tag{1.12a}$$

$$\dot{\alpha} = -\frac{T_{tol}}{m . V_T} . sin\alpha + \frac{g}{V_T} . (cos(\alpha - \theta) - 1) \tag{1.12b}$$

$$\dot{m} = a \cdot V_T \cdot (m - m_f) \tag{1.12c}$$

Where m_f is the final mass (the mass of the zero-fuel and iceless aircraft).

We can re-write the system in the following classical control format for simplicity:

$$\dot{x}_1 = -a_1 x_1^2 x_3 + x_3 cos(x_2) u + a_2 sin(x_2 - b) \tag{1.13a}$$

$$\dot{x}_2 = -\frac{x_3}{x_1} sin(x_2) u + \frac{a_2}{x_1} (cos(x_2 - b) - 1) \tag{1.13b}$$

$$\dot{x}_3 = -a_3 x_1 (x_3 - a_4) \tag{1.13c}$$

Where X=$[V_T, \alpha, m]$=$[x_1, x_2, x_3]$ and $u = T_{tol}$.

1.1.2 Electrically Powered UAV (AAI Aerosonde)

Icing affects all sorts of aircrafts and planes, however those effects are much more noticeable in smaller aerial vehicles, mainly due to their low speed which exposes them longer to icy conditions, and in our particular case, it's size makes the load added by the ice relatively greater thus making the change of mass more impactful.

With this in mind, we chose to work on an aerosonde UAV with a weight of 13.5 Kg's. This aircraft is represented by a size of 12 nonlinear state space model, starting with the equations of position :

$$\dot{p}_n = u(cos(\theta)sin(\psi)) + v(sin(\phi)sin(\theta)cos(\psi) - cos(\phi)sin(\psi)) + w(cos(\phi)sin(\theta)cos(\psi) + sin(\phi)sin(\psi)) \tag{1.14a}$$

$$\dot{p}_e = u(cos(\theta)cos(\psi)) + v(sin(\phi)sin(\theta)sin(\psi) + cos(\phi)cos(\psi)) + w(cos(\phi)sin(\theta)sin(\psi) - sin(\phi)cos(\psi)) \tag{1.14b}$$

$$\dot{p}_d = u(-sin(\theta)) + v(sin(\phi)cos(\theta)) + w(cos(\phi)cos(\theta)) \tag{1.14c}$$

Followed by the equations of velocity :

$$\dot{u} = (r.v - q.w) + \frac{f_x}{mass} \tag{1.15a}$$

$$\dot{v} = (p.w - r.u) + \frac{f_y}{mass} \tag{1.15b}$$

$$\dot{w} = (q.u - p.v) + \frac{f_z}{mass} \tag{1.15c}$$

Then the rotation dynamics :

$$\dot{\phi} = p + q(sin(\phi)tan(\theta) + r(cos(\phi)tan(\theta)) \tag{1.16a}$$

$$\dot{\theta} = q(cos(\phi)) + r(-sin(\phi)) \tag{1.16b}$$

$$\dot{\psi} = q\frac{sin(\phi)}{cos(\theta)} + r\frac{cos(\phi)}{cos(\theta)} \tag{1.16c}$$

And finally the rotation rates dynamics :

$$\dot{p} = (G_1.p.q - G_2.q.r) + (G_3.ell + G_4.n) \tag{1.17a}$$

$$\dot{q} = (G_5.p.r - G_6(p^2 - r^2)) + \frac{m}{J_y} \tag{1.17b}$$

$$\dot{r} = (G_7.p.r - G_1.q.r) + (G_4.ell + G_8.n) \tag{1.17c}$$

The state vector being X=[p_n: **North position** (m), p_e: **East position** (m), p_d: **Down position** (negative altitude) (m), u: **velocity along body x-axis** (m/s), v: **velocity along body y-axis** (m/s), w: **velocity along body z-axis** (m/s), ϕ: **roll angle** (rad), θ: **pitch angle** (rad), ψ: **yaw angle** (rad), p: **roll rate** (rad/s), q: **pitch rate** (rad/s), r: **yaw rate** (rad/s)], and the control inputs u=[f_x: **force along the x-axis** (drag-thrust) (N), f_y: **force along the y-axis** (side forces) (N), f_z: **force along the z-axis** (weight-lift) (N), ell: **roll moment** (N.m), m: **pitch moment** (N.m), n: **yaw moment** (N.m)]. (The rest of the parameters in

Appendix B (6.6))

Change of mass:

Typically, the mass of an aircraft is considered a constant parameter for electrically powered UAVs ($m\dot{a}ss = 0$). However, for small UAVs affected by icing, the change of mass can be used as an indicator of icing and its severity.

Assumption 2:

The mass is only affected by ice accretions, and no other external or internal factors.

In addition, it is important to note that icing can usually affect more than just the mass (drag, aircraft geometry, aerodynamics ...etc), but in the scope of our research we only consider the effect on mass change.

With everything set we can showcase the final augmented state vector and control vector as follows:

$$X = \begin{bmatrix} p_n \\ p_e \\ p_d \\ u \\ v \\ w \\ \theta \\ \phi \\ \psi \\ p \\ q \\ r \\ mass \end{bmatrix} \quad u = \begin{bmatrix} f_x \\ f_y \\ f_z \\ ell \\ m \\ n \end{bmatrix}$$

Where the mass change dynamics, in the absence of icing are defined by :

$$m\dot{a}ss = 0 \tag{1.18}$$

Remark 2:

For the stabilisation of the system we linearize around a suitable operating point, check the commandability of the resulting system, then compute the Linear Quadratic Regulator (LQR) gain for a stabilising state feedback $u = -K_{lqr}\Delta X$

1.2 Objectives

We can divide our problem into two parts :

1. **LMI based Nonlinear Observer Application to the Icing Accretion of Fuel powered and Electrically powered UAV's:** The goal of this section is to design nonlinear observers for the given system. In this context, icing will be treated as a disturbance in the dynamic equations governing the mass.
2. **Learning-Based Identification: Application to Aircraft Dynamics Modeling:** The objective is to design a specialized Recurrent Neural Network architecture capable of approximating the behavior of the given dynamical system.

Chapter 2

LMI based Nonlinear Observer: Application to the Icing Accretion Problem

State observers are crucial in control systems engineering for estimating the internal state of a dynamic system based on its output measurements. These estimations are particularly important when direct measurement of all state variables is impractical due to physical, technical, or economic limitations. By offering real-time estimates of internal states, state observers facilitate effective monitoring, control, and fault detection across diverse applications.

2.1 Observability of systems

Before discussing observers, it is essential to address a fundamental concept: system observability. Observability measures how well a system's internal states can be reconstructed using its output measurements. This concept is crucial for fault analysis, system control, and observer design.

2.1.1 Observability of linear systems

Consider the following linear system :

$$\begin{aligned} \dot{x} &= Ax + Bu \\ y &= Cx \end{aligned} \tag{2.1}$$

Where :

- x is the state vector of size n.
- u is the command (input) vector of size m.
- y is the measured output vector of size q.
- A, B and C are the system matrices of respective sizes $(n \times n)$, $(n \times m)$ and $(q \times n)$.

The observability of a linear system is analyzed using the observability matrix, which is defined as follows:

$$Obs = \begin{pmatrix} C \\ CA \\ CA^2 \\ . \\ CA^{n-1} \end{pmatrix} \tag{2.2}$$

Theorem
A linear system is completely observable if and only if rank(Obs)=n.

The idea is that if n columns of the observability matrix are linearly independent then each and every single one of our state variables are viewable through a linear combination of the output variables.

2.1.2 Observability of nonlinear systems

Consider the following non-linear system :

$$\begin{aligned} \dot{x} &= f(x, u) \\ y &= h(x) \end{aligned} \tag{2.3}$$

Where $f(.) \in \mathbb{R}^n$ and $h(.) \in \mathbb{R}^q$ are two non-linear vector field functions.

We define the lie derivative of h with respect to f as follows:

$$L_f h(x) = \frac{\partial h}{\partial x} f(x) \tag{2.4}$$

And the consecutive lie derivatives:

$$L_f^k h(x) = L_f L_f^{k-1} h(x) = \frac{\partial}{\partial x}[L_f^{k-1} f(x)] f(x) \tag{2.5}$$

We define the observability matrix as follows:

$$M_o = \frac{\partial}{\partial x} \begin{pmatrix} h(x) \\ L_f h(x) \\ L_f^2 h(x) \\ . \\ L_f^{n-1} h(x) \end{pmatrix} \tag{2.6}$$

The system of equation 2.3 is locally weakly observable if and only if rank(M_o)=size(X)=n.

2.2 Linear State Observers

2.2.1 Luenberger Observer

The Luenberger observer, introduced by David Luenberger in 1966, is designed for linear time-invariant (LTI) systems. It reconstructs the state vector of a system by using the system's output and a model of the system dynamics. The Luenberger observer has the form:

$$\dot{\hat{x}}(t) = A\hat{x}(t) + Bu(t) + L(y(t) - C\hat{x}(t)) \tag{2.7}$$

where:

- $\hat{x}(t)$ is the estimated state vector.
- A, B, C are the system matrices.
- L is the observer gain matrix, designed to ensure the stability of the error dynamics $e(t) = x(t) - \hat{x}(t)$.

The observer gain L is chosen such that the eigenvalues of $(A - LC)$ have negative real parts, ensuring the estimation error converges to zero over time [25].

2.2.2 Continuous-Time Kalman Filter

The continuous-time Kalman filter, introduced by Rudolf E. Kálmán in 1960, is an optimal recursive solution to the linear quadratic estimation problem, providing the best estimate of the state by minimizing the mean of the squared error. It is particularly effective for systems with noisy measurements and disturbances. The Kalman filter operates in two steps: prediction and update.

The continuous-time Kalman filter equations are as follows:

Prediction

$$\dot{\hat{x}}(t) = A\hat{x}(t) + Bu(t) \tag{2.8}$$

$$\dot{P}(t) = AP(t) + P(t)A^T + Q - P(t)C^T R^{-1} CP(t) \tag{2.9}$$

Update

$$\hat{x}(t) = \hat{x}(t) + K(t)(y(t) - C\hat{x}(t)) \tag{2.10}$$

$$K(t) = P(t)C^T R^{-1} \tag{2.11}$$

Where:

- $\hat{x}(t)$ is the state estimate.
- $P(t)$ is the error covariance matrix.
- Q and R are the process and measurement noise covariance matrices, respectively.
- $K(t)$ is the Kalman gain.

The continuous-time Kalman filter optimally combines the model predictions with the measurements to provide accurate state estimates even in the presence of noise [26].

2.3 Nonlinear State Observers

2.3.1 High-Gain Observer

The High-Gain Observer is designed for nonlinear systems with well-defined observable forms. This observer amplifies the correction term to ensure rapid error convergence. Its structure can be represented as:

$$\dot{\hat{x}}(t) = f(\hat{x}(t), u(t)) + H(y(t) - h(\hat{x}(t))) \tag{2.12}$$

Where:

- f and h describe the nonlinear system dynamics and output equations.
- H is the high-gain matrix, typically chosen such that the error dynamics are dominated by fast poles.

The high-gain approach ensures that the estimation error $e(t)$ converges quickly, but it may lead to robustness issues due to sensitivity to noise and model uncertainties [27].

2.3.2 Sliding Mode Observer

The Sliding Mode Observer (SMO) leverages the sliding mode control principle to estimate the states of nonlinear systems robustly against uncertainties and disturbances. The SMO introduces a discontinuous injection term that forces the estimation error to converge to zero.

The observer can be expressed as:

$$\dot{\hat{x}}(t) = f(\hat{x}(t), u(t)) + G(y(t) - h(\hat{x}(t))) + \Phi(\hat{x}(t), y(t)) \tag{2.13}$$

- G is the gain matrix.
- Φ is a discontinuous function designed to drive the error to a sliding surface, ensuring robust convergence.

The key advantage of the SMO is its robustness to model uncertainties and external disturbances, making it suitable for applications requiring high precision under adverse conditions [28, 29].

2.3.3 LMI Observer Design for Nonlinear Systems

Consider a nonlinear system described by the following state-space representation:

$$\dot{x}(t) = f(x(t), u(t)) \tag{2.14}$$

Written in its explicit form:

$$\begin{aligned}
\dot{x}_1(t) &= f_1(x(t), u(t)) \\
\dot{x}_2(t) &= f_2(x(t), u(t)) \\
&\vdots \\
\dot{x}_{n-1}(t) &= f_{n-1}(x(t), u(t)) \\
\dot{x}_n(t) &= f_n(x(t), u(t))
\end{aligned} \tag{2.15}$$

Consider the following transformation:

$$\begin{aligned}
\dot{x}_1(t) &= f_1(x(t), u(t)) \pm x_2 \\
\dot{x}_2(t) &= f_2(x(t), u(t)) \pm x_3 \\
&\vdots \\
\dot{x}_{n-1}(t) &= f_{n-1}(x(t), u(t)) \pm x_n \\
\dot{x}_n(t) &= f_n(x(t), u(t))
\end{aligned} \tag{2.16}$$

Let the following change of variable:

$$g_i(x(t), u(t)) = \begin{cases} f_i(x(t), u(t)) - x_{i+1}, & \text{for } i = 1, 2, \ldots, n-1 \\ f_n(x(t), u(t)), & \text{for } i = n \end{cases}$$

The system can be re-written in the following form:

$$\dot{x}(t) = Ax(t) + g(x(t).u(t)) \tag{2.17}$$

$$\text{A}= \begin{pmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ 0 & 0 & 0 & \cdots & 0 \end{pmatrix}$$

- $x(t) \in \mathbb{R}^n$ is the state vector,
- $u(t) \in \mathbb{R}^m$ is the input vector,
- $A \in \mathbb{R}^{n \times n}$ is a known constant matrix taken in the canonical form,
- $g(x(t), u(t)) \in \mathbb{R}^n$ is a nonlinear function.

Assume that the function $g(x, u)$ is Lipschitz continuous in x with Lipschitz constant $\gamma > 0$, i.e.,

$$\|g(x_1, u) - g(x_2, u)\| \leq \gamma \|x_1 - x_2\| \quad \forall x_1, x_2 \in \mathbb{R}^n, \forall u \in \mathbb{R}^m. \tag{2.18}$$

We aim to design an observer to estimate the state $x(t)$ based on the output $y(t) = Cx(t)$, where $C \in \mathbb{R}^{p \times n}$ is the output matrix.

The observer is formulated as:

$$\dot{\hat{x}}(t) = A\hat{x}(t) + g(\hat{x}(t), u(t)) + L(y(t) - C\hat{x}(t)), \tag{2.19}$$

where:

- $\hat{x}(t)$ is the estimated state,
- $L \in \mathbb{R}^{n \times p}$ is the observer gain matrix to be determined.

Define the estimation error as $e(t) = x(t) - \hat{x}(t)$. The error dynamics are given by:

$$\dot{e}(t) = (A - LC)e(t) + (g(x(t), u(t)) - g(\hat{x}(t), u(t))). \tag{2.20}$$

Using the Lipschitz condition, we assume g is a locally liptchiz function, we have:

$$\|g(x(t), u(t)) - g(\hat{x}(t), u(t))\| \leq \gamma \|e(t)\|. \tag{2.21}$$

To ensure the observer's stability, we consider the Lyapunov function:

$$V(e(t)) = e^T P e, \tag{2.22}$$

Where $P \in \mathbb{R}^{n \times n}$ is a symmetric positive definite matrix, i.e., $P > 0$.

The time derivative of the Lyapunov function along the trajectories of the error dynamics is:

$$\dot{V}(t) = \frac{d}{dt}(e^T P e) = e^T P \dot{e} + \dot{e}^T P e. \tag{2.23}$$

Substituting the error dynamics, we get:

$$\begin{aligned}\dot{V}(t) &= e^T P[(A - LC)e + (g(x,u) - g(\hat{x},u))] + [(A - LC)e + (g(x,u) - g(\hat{x},u))]^T P e \\ &= e^T P(A - LC)e + e^T P(g(x,u) - g(\hat{x},u)) + e^T (A - LC)^T P e + (g(x,u) - g(\hat{x},u))^T P e \\ &= e^T [P(A - LC) + (A - LC)^T P]e + 2e^T P(g(x,u) - g(\hat{x},u)). \end{aligned} \tag{2.24}$$

Using the Lipschitz condition, we have:

$$2e^T P(g(x,u) - g(\hat{x},u)) \leq 2\gamma e^T P e. \tag{2.25}$$

Thus, the derivative of the Lyapunov function can be bounded by:

$$\dot{V}(t) \leq e^T [P(A - LC) + (A - LC)^T P + 2\gamma P]e. \tag{2.26}$$

To ensure that $\dot{V}(t) < 0$, we need:

$$P(A - LC) + (A - LC)^T P + 2\gamma P < 0. \tag{2.27}$$

For exponential stability, let :

$$P(A - LC) + (A - LC)^T P + 2\gamma P < -\lambda P. \tag{2.28}$$

With $\lambda > 0$.

This inequality can be solved using LMI techniques such as YALMIP, LMI Toolbox and other optimization techniques. We seek a symmetric positive definite matrix P and an observer gain matrix L that satisfy the following LMI:

$$(A - LC)^T P + P(A - LC) + 2\gamma P < -\lambda P. \tag{2.29}$$

Using MATLAB or any LMI solver, solve for P and L.

This implies that:

$$\dot{V}(t) \leq -\lambda e^T P e = -\lambda V(t). \tag{2.30}$$

Thus, exponential stability is achieved.

We will be utilizing this LMI-based observer for our two applications: icing detection in fuel-powered UAVs and in electrically powered UAVs.

2.4 Results and Discussion

2.4.1 Fuel Powered UAV (Jetstream J3)

Taking the dynamics equations in (2.32):

$$\begin{aligned}
\dot{V}_T &= -\frac{\rho_0.V_T^2.S}{2m}.Cd_{act} + \frac{T_{tol}}{m}.cos\alpha + g.sin(\alpha - \theta) \\
\dot{\alpha} &= -\frac{T_{tol}}{m.V_T}.sin\alpha + \frac{g}{V_T}.(cos(\alpha - \theta) - 1) \\
\dot{m} &= a \cdot V_T \cdot (m - m_f)
\end{aligned}$$

Since ice accretion is a slowly varying phenomenon, we modeled it as a constant disturbance w_{ice} with $\dot{w}_{ice} = 0$.

This leads to the following equation:

$$\dot{m} = a \cdot V_T \cdot (m - m_f) + w_{ice} \tag{2.31}$$

Where ω_{ice} is a disturbance representing the icing's effect on the aircraft's mass.

We selected a constant perturbation implying a linearly varying mass for our tests. This choice is made based on the understanding that ice accretion is a slowly varying phenomenon. In real-world conditions, ice accretion does not happen instantaneously but rather builds up gradually over time. By modeling the mass change as a linear variation, we can effectively simulate this gradual accumulation of ice. This approach allows us to evaluate the observer's ability to track and estimate mass changes due to ice accumulation over time.

The observer equations are as follows:

$$\dot{\hat{V}}_T = -\frac{\rho_0.\hat{V}_T^2.S}{2\hat{m}}.\hat{Cd}_{act} + \frac{T_{tol}}{\hat{m}}.cos(\hat{\alpha}) + g.sin(\hat{\alpha} - \theta) + L_1.(y - \hat{y}) \tag{2.32a}$$

$$\dot{\hat{\alpha}} = -\frac{T_{tol}}{\hat{m}.\hat{V}_T}.sin(\hat{\alpha}) + \frac{g}{\hat{V}_T}.(cos(\hat{\alpha} - \theta) - 1) + L_2.(y - \hat{y}) \tag{2.32b}$$

$$\dot{\hat{m}} = a \cdot \hat{V}_T \cdot (\hat{m} - m_f) + \hat{w}_{ice} + L_3.(y - \hat{y}) \tag{2.32c}$$

$$\dot{\hat{w}}_{ice} = L_4.(y - \hat{y}) \tag{2.32d}$$

The augmented state space variables are as follows:

$$X = \begin{bmatrix} V_T \\ \alpha \\ m \\ w_{ice} \end{bmatrix}$$

After implementing the given model in Matlab, the system's observability has been studied, we took as measurements the Airspeed V_T and the angle of attack α. We employed the LMI

Observer for Nonlinear Systems with a liptchiz constant $\gamma \approx 65$, The LMI problem was solved using MATLAB's YALMIP toolbox. Simulations were conducted under various initial conditions, yielding the following results for mass estimation:

The observer's initial conditions are set to $\hat{x}_0 = [70(kts); -\pi/100(rad)]$ and $\hat{m}_0 = 6000Kg$.

The Aircraft's real dynamics are simulated under the following initial conditions:

Case 1: $x_0 = [100(kts); \pi/100(rad)]$ and $m_0 = 6071Kg$.

Case 2: $x_0 = [80(kts); \pi/30(rad)]$ and $m_0 = 6100Kg$.

Case 3: $x_0 = [120(kts); -\pi/15(rad)]$ and $m_0 = 6040Kg$.

Case 4: $x_0 = [100(kts); 0(rad)]$ and $m_0 = 6000Kg$.

Case 5: $x_0 = [115(kts); \pi/36(rad)]$ and $m_0 = 6015Kg$.

Case 6: $x_0 = [110(kts); -\pi/25(rad)]$ and $m_0 = 6050Kg$.

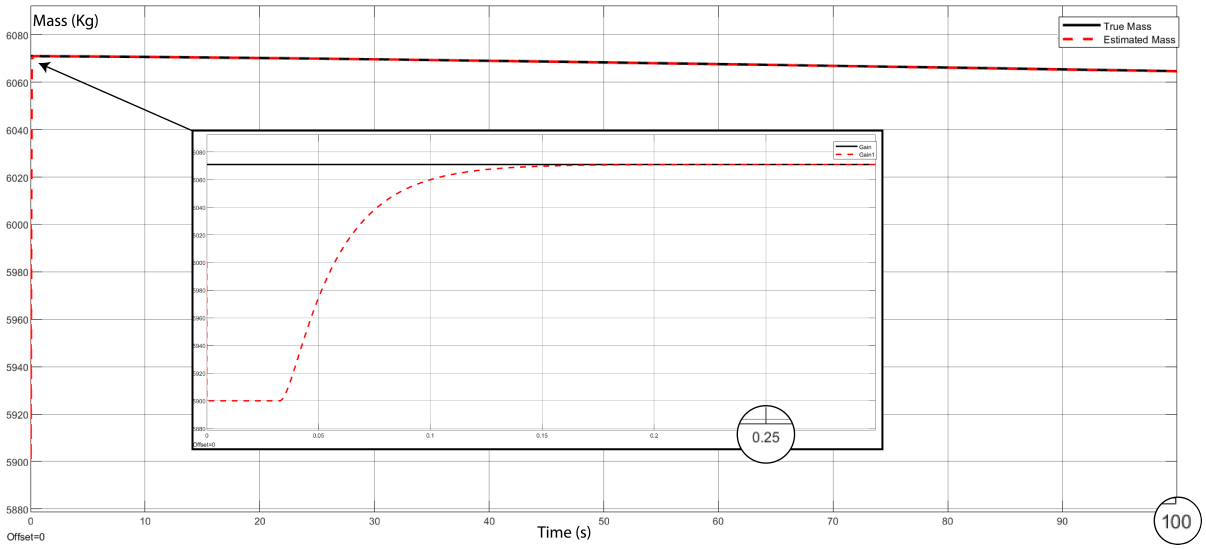

Figure 2.1: Mass icing estimation for fuel powered UAV, **Case 1**.

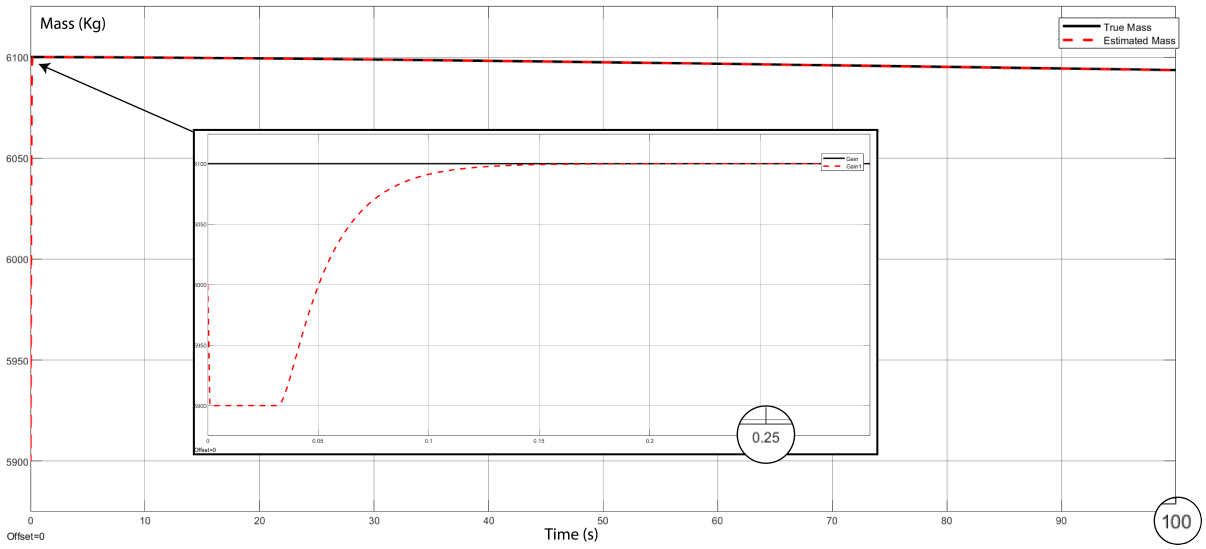

Figure 2.2: Mass icing estimation for fuel powered UAV, **Case 2**.

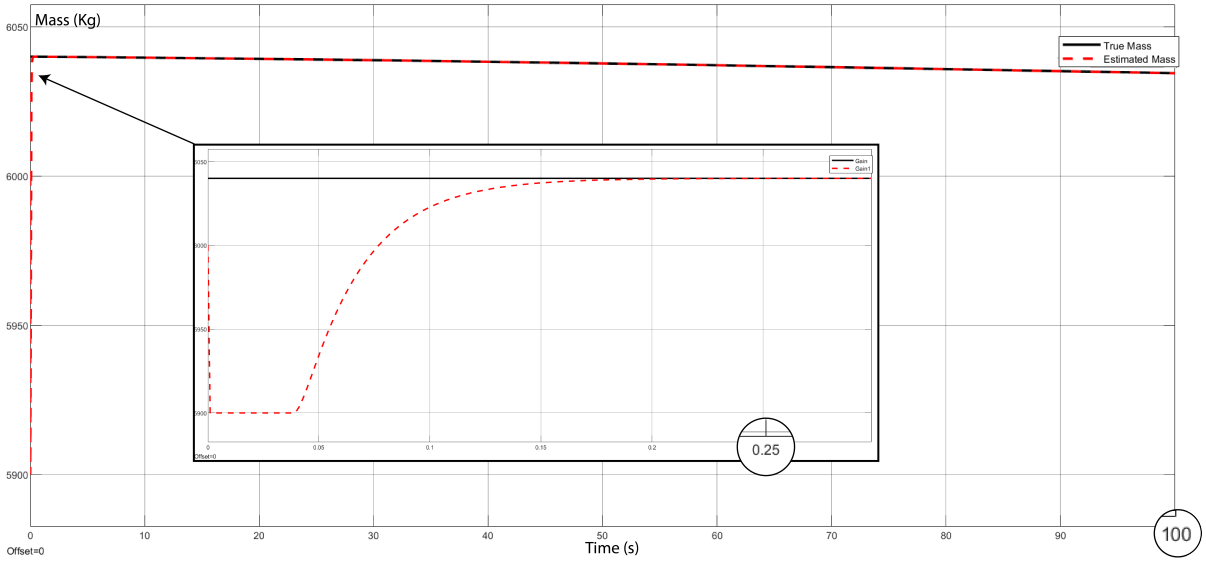

Figure 2.3: Mass icing estimation for fuel powered UAV, **Case 3**.

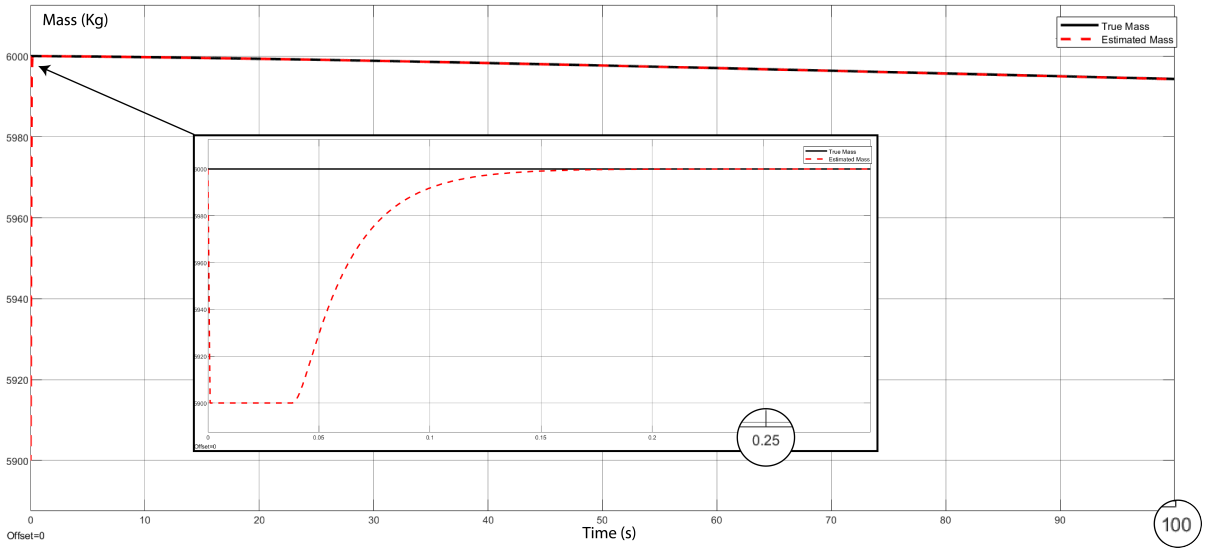

Figure 2.4: Mass icing estimation for fuel powered UAV, **Case 4**.

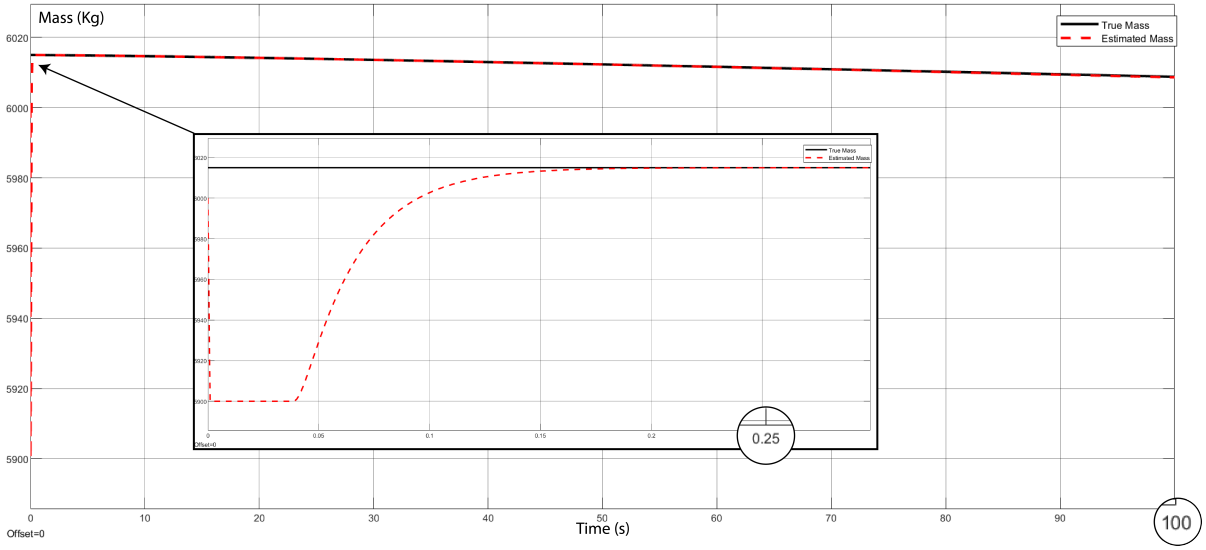

Figure 2.5: Mass icing estimation for fuel powered UAV, **Case 5**.

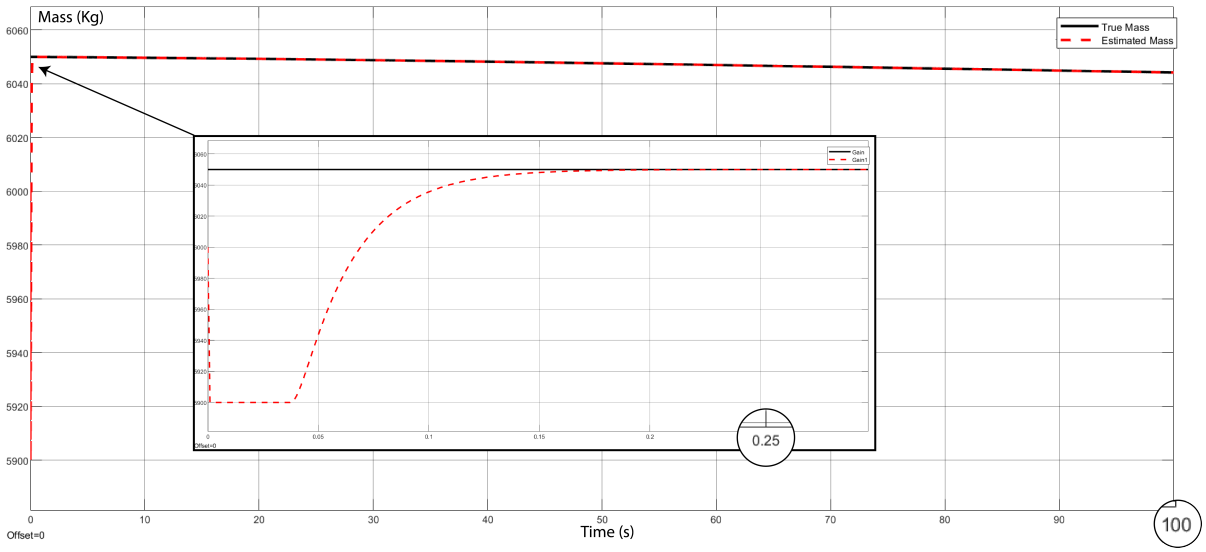

Figure 2.6: Mass icing estimation for fuel powered UAV, **Case 6**.

The estimated accumulated ice mass is determined as $m_{ice} - m_{clean}$, where m_{clean} represents the expected mass of the aircraft in the absence of icing. The value of m_{clean} is typically obtained by simulating the aircraft's dynamic model under icing-free conditions.

The estimated accumulated ice mass is shown in the figure below:

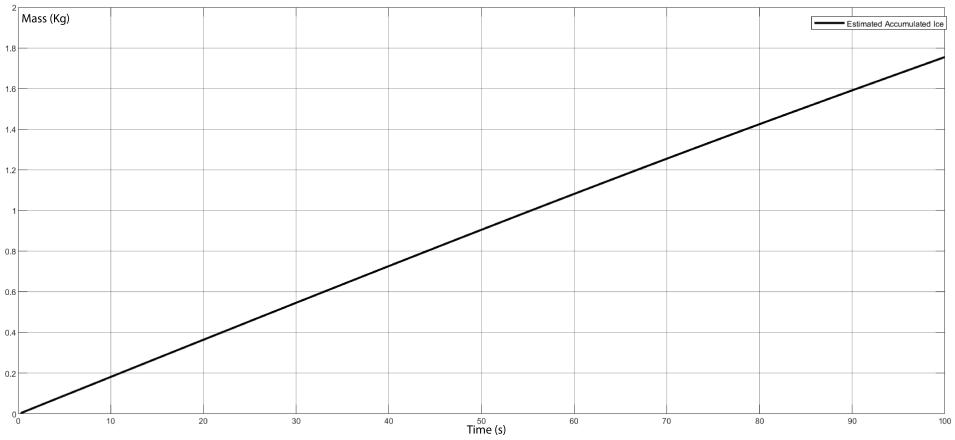

Figure 2.7: Estimated accumulated ice mass for fuel powered UAV.

2.4.2 Discussion

We tested the observer under six different initial conditions, and the results were promising. In all cases, the observed mass followed the behavior of the real mass in a very short time.

We modeled mass change as a linear variation to simulate the gradual accumulation of ice, reflecting the slowly varying nature of ice accretion. This approach helps evaluate the observer's ability to track and estimate mass changes due to ice over time.

The linear variation in mass due to constant perturbation serves as a simplified yet effective model to test the observer's performance. While actual ice accretion can involve more complex dynamics depending on environmental factors such as temperature, humidity, and wind speed, the linear model provides a controlled setting to validate our observer design.

2.4.3 Electrically Powered UAV (AAI Aerosonde)

Referring to equations 1.14 through 1.18, We remind the reader that the dynamics of electrically powered UAVs, in the absence of icing, are characterized by $\dot{mass} = 0$. This signifies that the mass of the Aerosonde remains constant due to the absence of both fuel consumption and icing accumulation.

Consequently, Any change in the mass will be considered as ice accretion, and since ice accretion is a slowly varying phenomenon, we model the mass change dynamics in the presence of icing:

$$\dot{mass} = \omega_{ice} \tag{2.33}$$

Where ω_{ice} is a disturbance representing the icing's effect on the aircraft's mass.

Remark 3:

Before designing the state observer, the system's observability has been studied. Thus, we make our final remark:

We admit that the position, velocity, rotation, and rotation rates are all measurable, given the availability of numerous sensors (such as IMUs, GPS, etc.).

After implementing the given model of the Aerosonde aircraft, we stabilized the system around the initial operating point $x_{p0} = [0; 0; -50]$ (m) for positions, $x_{v0} = [50; 0; 0]$ (m/s) for velocities, $x_{r0} = [0; 0; 0]$ (rad) for rotation angles, and $x_{vr0} = [0; 0; 0]$ (rad/s) for rotation rates. We utilized the LMI Observer for Nonlinear Systems, as described in Section 3.4. The LMI problem was solved using MATLAB's YALMIP toolbox to obtain the optimal solution. Simulations were run under different initial conditions, and the following results were obtained for the mass estimation:

The observer's initial conditions are set to $\hat{x}_0 = [50.1; 50.1; -0.1; 100.1; 5.1; 5.1; 5.1; 5.1; 5.1; 5.1; 5.1; 5.1]$ and $\hat{m}_0 = 12.5Kg$.

The UAV's real dynamics are simulated under the following initial conditions:

Case 1: $x_0 = [0.1; 0.1; -50.1; 50.1; 0.1; 0.1; 0.1; 0.1; 0.1; 0.1; 0.1; 0.1]$ and $m_0 = 13.5$.

Case 2: $x_0 = [1.1; 1.1; -40.1; 55.1; 0.6; 0.6; 0.3; 0.3; 0.3; -0.4; -0.4; -0.4]$ and $m_0 = 13.5$.

Case 3: $x_0 = [0.3; 0.3; -48.1; 52.1; 0.3; 0.3; 0.3; 0.3; 0.3; 0.3; 0.3; 0.3]$ and $m_0 = 13.7$.

Case 4: $x_0 = [0.6; 0.6; -45.1; 55.1; 0.6; 0.6; 0.6; 0.6; 0.6; 0.6; 0.6; 0.6]$ and $m_0 = 14.0$.

Case 5: $x_0 = [1.1; 1.1; -40.1; 60.1; 1.1; 1.1; 1.1; 1.1; 1.1; 1.1; 1.1; 1.1]$ and $m_0 = 14.5$.

Case 6: $x_0 = [-0.9; -0.9; -60.1; 60.1; 1.1; 1.1; 1.1; 1.1; 1.1; -0.9; -0.9; -0.9]$ and $m_0 = 14.5$.

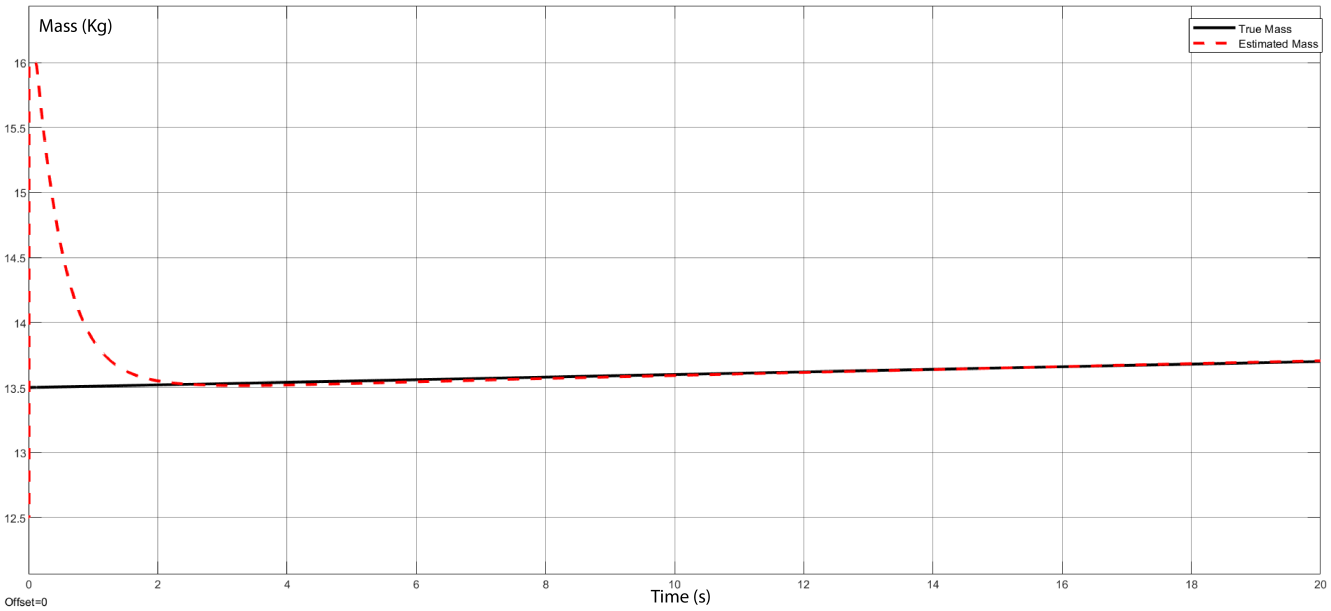

Figure 2.8: Mass icing estimation for electrically powered UAV, **Case 1**.

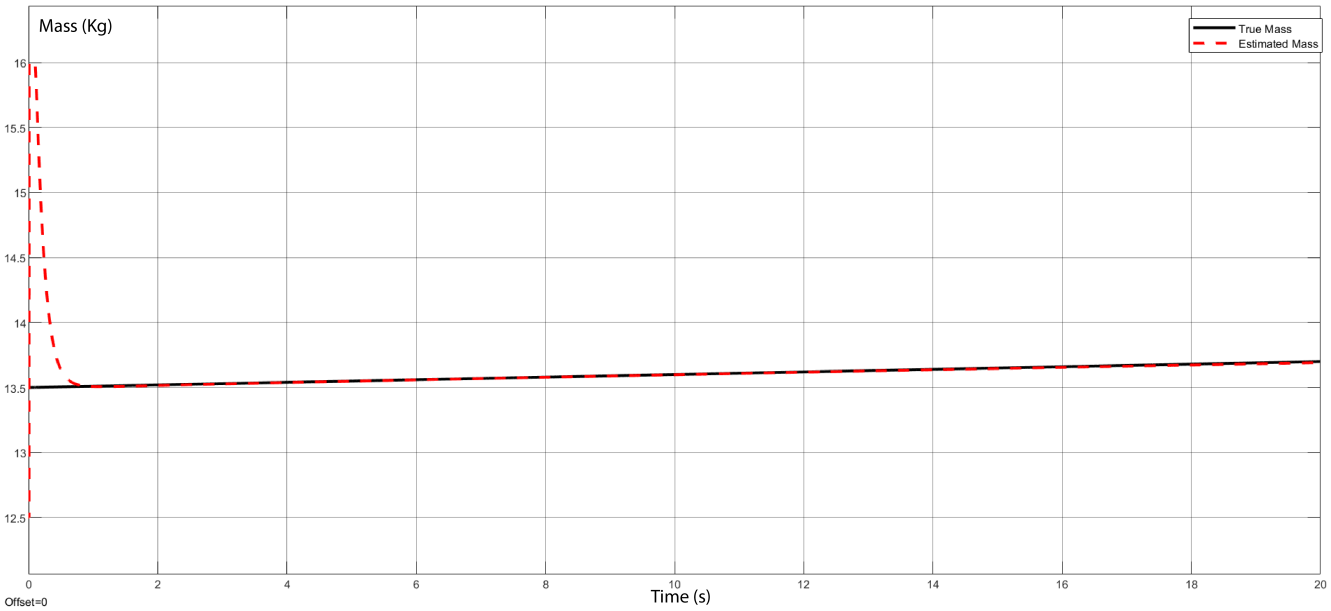

Figure 2.9: Mass icing estimation for electrically powered UAV, **Case 2**.

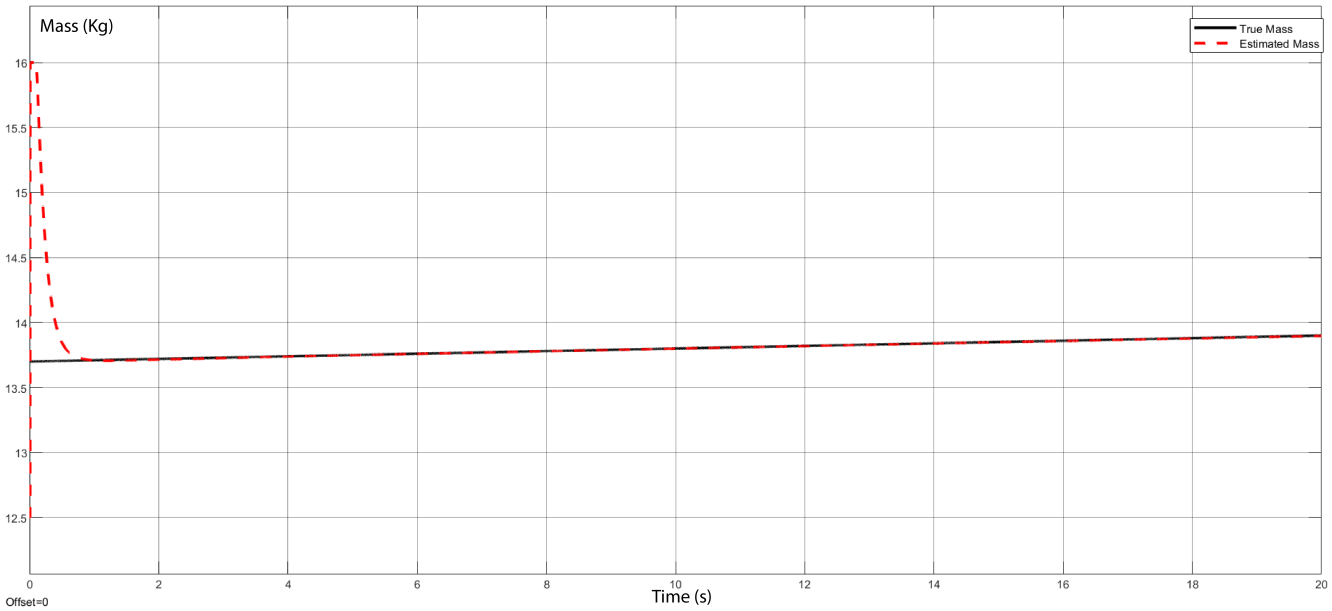

Figure 2.10: Mass icing estimation for electrically powered UAV, **Case 3**.

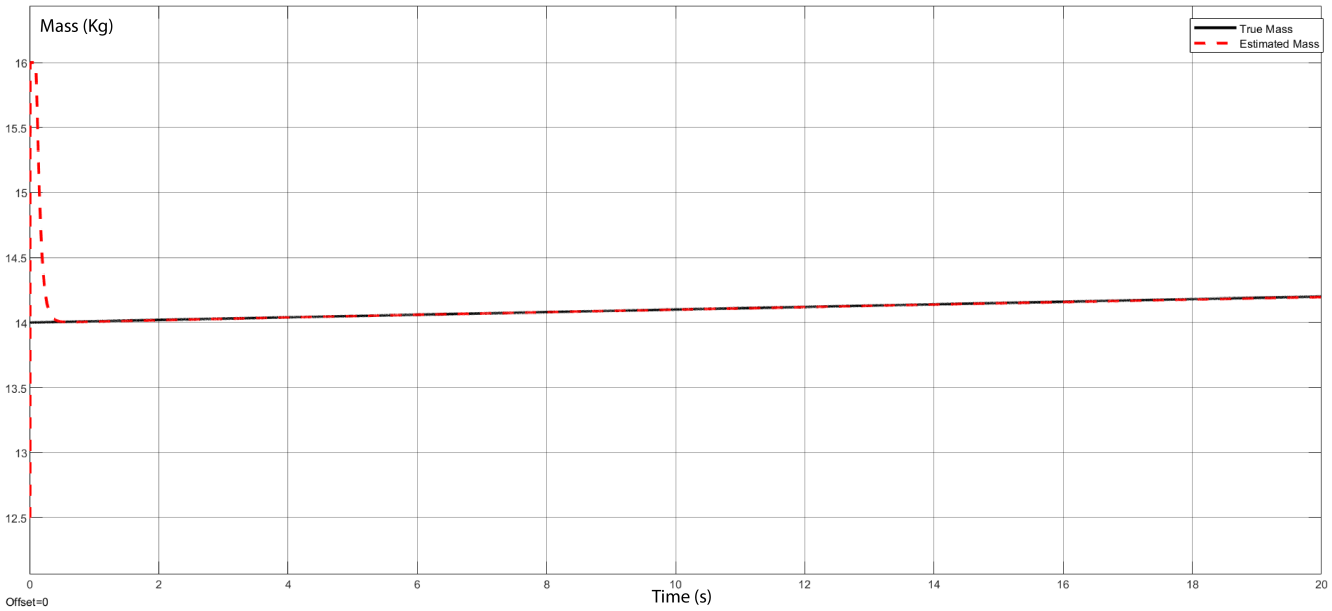

Figure 2.11: Mass icing estimation for electrically powered UAV, **Case 4**.

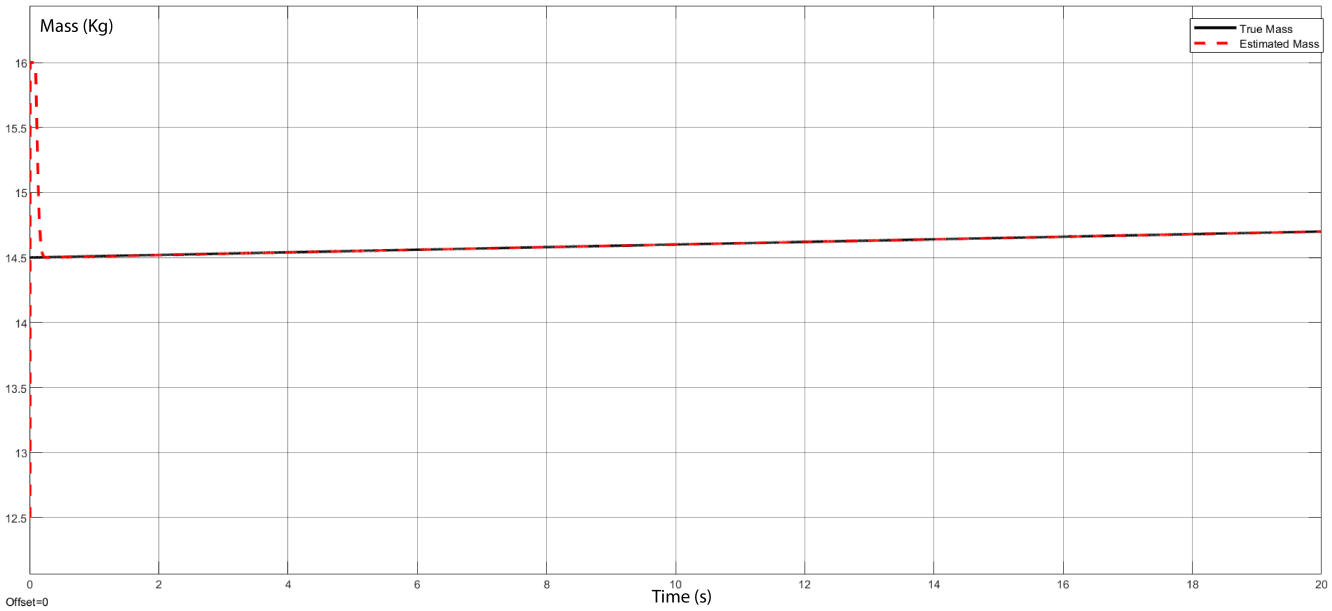

Figure 2.12: Mass icing estimation for electrically powered UAV, **Case 5**.

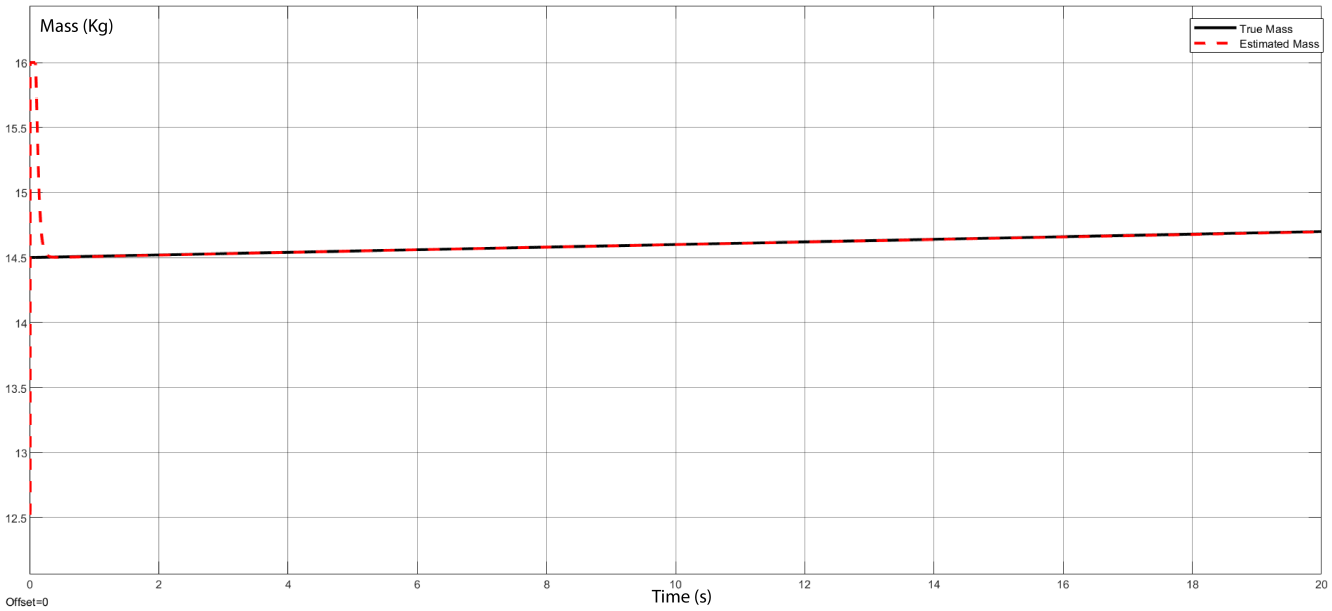

Figure 2.13: Mass icing estimation for electrically powered UAV, **Case 6**.

2.4.4 Discussion

We evaluated the observer under six different initial conditions, and the results were promising: in each instance, the observed mass quickly aligned with the real mass. We simulated mass change using a linear variation to represent the gradual accumulation of ice, reflecting the slow nature of ice accretion. This approach assesses the observer's ability to track and estimate mass changes due to ice over time. Using a linear variation for mass change due to constant perturbation provides a simplified yet effective model to test the observer's performance. Although actual ice accretion can involve more complex dynamics influenced by environmental factors like temperature, humidity, and wind speed, the linear model offers a controlled environment to validate our observer design.

2.4.5 Conclusions

Our contributions to the field of icing accretion detection were achieved through the development and implementation of an LMI based nonlinear observer on fuel powered and electrically powered UAVs. We synthesized our observer under six different initial conditions, and the results were promising. The observer demonstrated a strong ability to accurately track the real system across all tested scenarios.

Looking ahead, there are several avenues for further enhancement. One promising direction involves refining ice accretion models to incorporate nonlinear effects and diverse environmental conditions. Additionally, conducting a comparative study among various observers could provide valuable insights into their effectiveness. Moreover, practical experimentation to validate these findings would be interesting.

By pursuing these avenues, we aim to significantly enhance the effectiveness and applicability of icing accretion detection systems across different UAV platforms.

Chapter 3

Learning Based Identification of Nonlinear systems: Application to Aircraft Dynamics Modeling

System identification is one of the major practices in control engineering, the ability to generate accurate dynamical models from input/output or state measurements data has becomes increasingly important with today's research problems. In this chapter we will delve into learning based identification methods where we will define, explain and implement two learning algorithms, namely the Recurrent Neural Network based method and the Sparse Identification of Nonlinear Dynamics algorithm, which will be both applied to the Jetstream J31 aircraft system.

3.1 RNN based identification

3.1.1 Class of systems

For what follows let's consider the nonlinear systems of the class represented by the following continuous-time state space representation:

$$\dot{x} = f(x) + g(x)u \tag{3.1a}$$
$$y = h(x) \tag{3.1b}$$

Where $x = \begin{pmatrix} x_1 & x_2 & ... & x_n \end{pmatrix}^T \in \mathbb{R}^n$ is the state vector, $u = \begin{pmatrix} u_1 & u_2 & ... & u_m \end{pmatrix}^T \in \mathbb{R}^m$ is the control input signal, and $y = \begin{pmatrix} y_1 & y_2 & ... & y_q \end{pmatrix}^T \in \mathbb{R}^q$ is the output vector.The functions f(x), g(x) and h(x) are nonlinear function matrices of respective sizes of (n × 1), (n × m) and (q × 1).

3.1.2 Neural Networks

An artificial neural network (ANN) is composed of interconnected units or nodes, known as artificial neurons, which mimic the neurons in a human brain. These nodes are linked by edges,

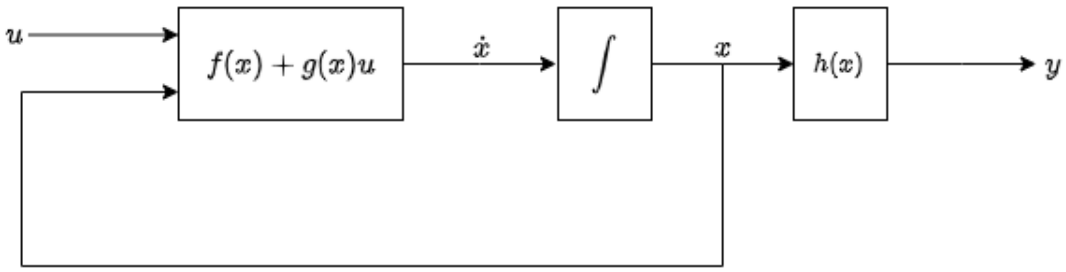

Figure 3.1: Class of system

similar to synapses, which transmit signals between neurons. Each artificial neuron receives inputs from connected neurons, processes them, and then sends an output to other neurons. The signals are represented by real numbers, and each neuron's output is determined by a non-linear function of the weighted sum of its inputs, known as the activation function. The weight of each connection adjusts during the learning process, determining the strength of the transmitted signal.

Typically, neurons are organized into layers. Different layers apply different transformations to their inputs. Signals propagate from the input layer through multiple intermediate layers, known as hidden layers, and finally to the output layer. A network with at least two hidden layers is usually referred to as a deep neural network.

These networks are used in machine learning and deep learning applications to predict future outcomes, classify data, create clusters etc. They have the capability to learn from past data and recognise patterns within it.

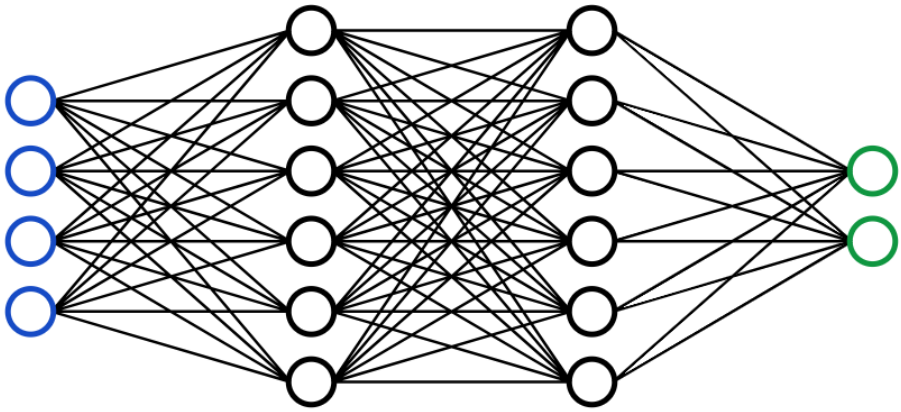

Figure 3.2: Neural Network Architecture [1]

3.1.3 Recurrent Neural Networks

Recurrent Neural Networks, or RNN for short, are a special type of neural networks that include a feedback loop that feeds past hidden states to future ones (figure 3.3), hence the recurrence. This type of neural networks is usually used for time-series prediction, such as weather or financial predictions, but it also attracted the attention of dynamical systems identification and observation specialists due to the possibility of the recurrent term representing the dynamical

connection between past and future states.
A typical recurrent neural network has the following form :

$$\nu_t = W_{in}\mu_t + W_h f_{act}(\nu_{t-1}) \tag{3.2a}$$
$$o_t = W_{out} f_{act}(v_t) \tag{3.2b}$$

Where ν_t is the hidden state vector at time-step t, μ_t is the manipulated input at time-step t and o_t is the network's output at the same time-step. $f_{act}(.)$ is a non-linear activation function such as tanh, sigmoid or ReLu, and the matrices W_{in}, W_h and W_{out} are respectively the trainable input weights, hidden weights and output weights.

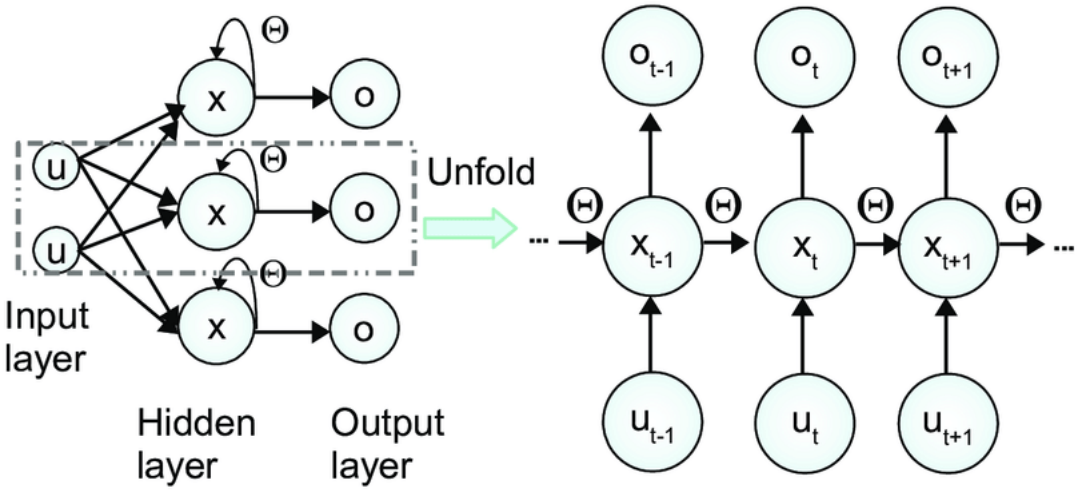

Figure 3.3: Folded/Unfolded RNN Structure [2]

This classical representation is a discrete time representation, considerable work has been done under it ([21][22][20]), however recent developments have led to a more direct approach using continuous-time recurrent neural networks, which will be explained in the following subsection.

Remark 1 :
A certain issue can occur during the training of RNN's caused by the gradient of the network which, with time, will be multiplied by the term w^n (with w one of the weights).If n becomes too large, the gradient can either vanish (vanishing gradient problem for $w<1$) or explode (exploding gradient problem for $w>1$).
To solve this issue many methods have been developed, the two main ones being the use of LSTM networks instead of simple RNN's that introduce a "forget factor" that cuts the influence of past terms on the long term, or the use of back-propagation-through-time (BPTT) truncate, which as the name suggests truncates the past states used to a certain level as to avoid the mentioned issues.

3.1.4 Continuous-Time Recurrent Neural Networks

Continuous-Time Recurrent Neural Networks, or CTRNN for shorts, are the continuous time variant of simple RNN's. The form of a CTRNN slightly differs from it's simple counterpart,

it is as follows :

$$\tau \frac{d\nu}{dt} = -\nu + W_{in}\mu + W_h f_{act}(\nu) \tag{3.3a}$$

$$o = W_{out} f_{act}(\nu) \tag{3.3b}$$

The parameters are practically the same except for the addition of the τ term:
$\nu = \begin{pmatrix} \nu_1 & \nu_2 & ... & \nu_n \end{pmatrix}^T \in \mathbb{R}^n$ is the hidden state vector, $\mu = \begin{pmatrix} \mu_1 & \mu_2 & ... & \mu_m \end{pmatrix}^T \in \mathbb{R}^m$ and $o = \begin{pmatrix} o_1 & o_2 & ... & o_q \end{pmatrix}^T \in \mathbb{R}^q$ is the network's output. The trainable weights W_{in}, W_h and W_{out} are respectively of sizes (n × m), (n × n) and (q × n).

To have a better understanding of the added term's influence let's look at the numerical derivative of the equation 3.2 using Euler's method :

$$dv = v_{t+1} - v_t = \frac{dt}{\tau}(-\nu + W_{in}\mu + W_h f_{act}(\nu))$$

If we put $\frac{dt}{\tau} = \alpha$ we obtain :

$$v_{t+1} = (1-\alpha)\nu + \alpha(W_{in}\mu + W_h f_{act}(\nu)) \tag{3.4}$$

α represents the amount of contribution past states have on the next states, where the closer α gets to 1, the less the contribution.

This method is a continuous-time representation because instead of optimizing an input to output pair, we optimize the dynamics of the desired system. This method allows us to have self recurrent models that can mimic dynamical systems behaviour.

When training a CTRNN on a nonlinear system the goal is to minimize $F(x) - F_{RNN}(x)$ where $F(x)$ and $F_{RNN}(x)$ are respectively the dynamic of the nonlinear system and RNN model. Thus the following theorem:

Theorem : [20] Consider the nonlinear system of Equation 3.1 and the RNN model of Equation 3.3 with the same initial condition $x(0) = \hat{x}(0) = x_0 \in \omega_p$. For any ϵ>0 and T>0, there exists an optimal weight matrix θ^* such that the state $\hat{x}$ of the RNN model of Equation 3.3 with $\theta = \theta^*$ satisfies the following equation:

$$\sup_{t\in[0,T]} |x(t) - \hat{x}| < \epsilon \tag{3.5}$$

3.1.5 Training Procedure

Before detailing the training process for CTRNN's we need to clarify the confusion around the actual equivalence between the models of equations 3.1 and 3.3, so here is the modified CTRNN used to predict nonlinear systems of the defined class (we assume x and u have the sizes defined in subsection 1.5.1) :

$$\tau \frac{d\nu}{dt} = -\nu + W_{in}\mu + W_h f_{act}(\nu) \tag{3.6a}$$

$$\hat{x} = W_{out} f_{act}(\nu) \tag{3.6b}$$

With the hidden state vector $\nu \in \mathbb{R}^{\tilde{n}}$ and $\tilde{n}$ is the hidden layer's size. The manipulated vector $\mu = \begin{pmatrix} x_1 & x_2 & ... & x_n & u_1 & u_2 & ... & u_m \end{pmatrix}^T \in \mathbb{R}^{m+n}$ is the manipulated input vector, and the output is $\hat{x} = \begin{pmatrix} \hat{x}_1 & \hat{x}_1 & ... & \hat{x}_n \end{pmatrix}^T \in \mathbb{R}^n$. Finally the weight matrices are therefore of respective sizes ($\tilde{n}$ × m+n), ($\tilde{n}$ × $\tilde{n}$) and (n × $\tilde{n}$).

The remark is that the hidden states and model state **are not necessarily the same variable**. The output of the neural network is trained to reproduce the states of the nonlinear system, thus the hidden states do not necessarily have the same size as our real system.

Remark 2 :
The model as defined above requires knowledge of all states for training so we assume that all states are measurable during this period.

For the training of our model 3 steps are required : **Data Generation**, **Model Training**, **Model Evaluation**, with an additional implementation step.

3.1.5.1 Data Generation

CTRNN's require large amount of data, these can be obtained from extensive open-loop simulation of the system under different control inputs and initial conditions. This requires two conditions, the first one is that our control signal must be bounded $u_{min} < u < u_{max}$, and the second one is that the system must be at least stable at the sense of Lyapunov, under the applied control input.

Given that the region of closed-loop stability of the nonlinear system of equation 3.1 is Ω_p, the goal is to generate enough information about the system inside a sub-stable region $\Omega_{p'}$ that will become the RNN's characteristic region of prediction (figure 3.4). We do that by generating random control inputs of period Δt, written as : $u(t) = u_k$ for $t \in [t_k, t_k + 1]$, and using a sampling period $h_c < \Delta t$ for a finite simulation time.

After obtaining our data, we first organise it as a set of sequences, then into an input and target variables. If for a single simulation we obtain k data points, our input will be these data points from t=0 to t=k-1, while our target variable will take data from t=1 to t=k. The final form of the data should have a size of (number of sequences, length of sequence, number of features), with length of sequence equal k. and number of features equal m+n for the input variable and equal n for the target variable. (figure 3.5)

Finally we divide our data into training, validation and testing sets.

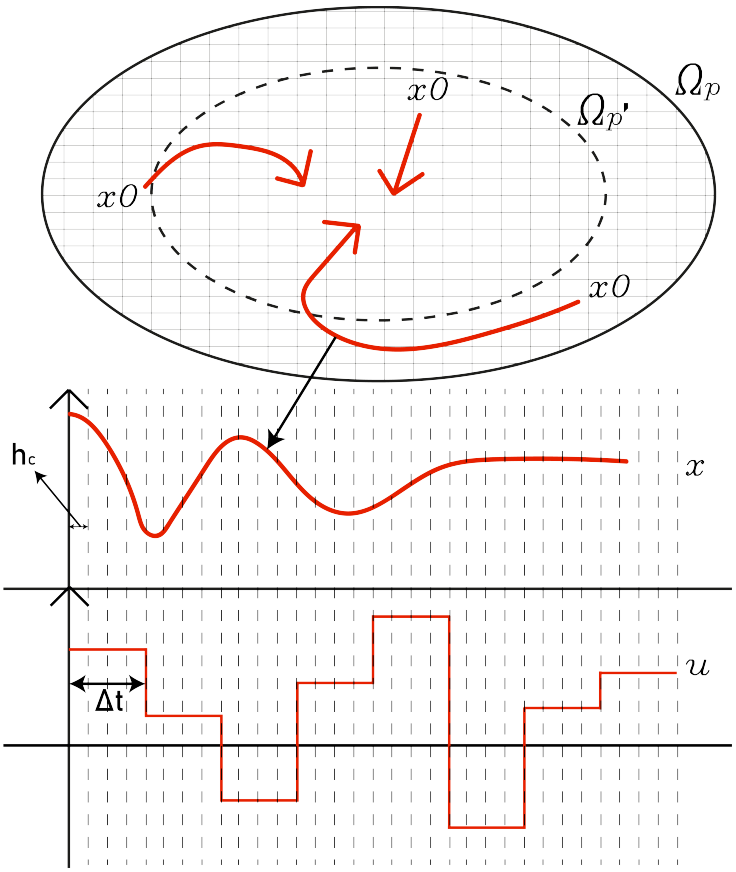

Figure 3.4: Discretisation of the state and input signals inside the region of stability

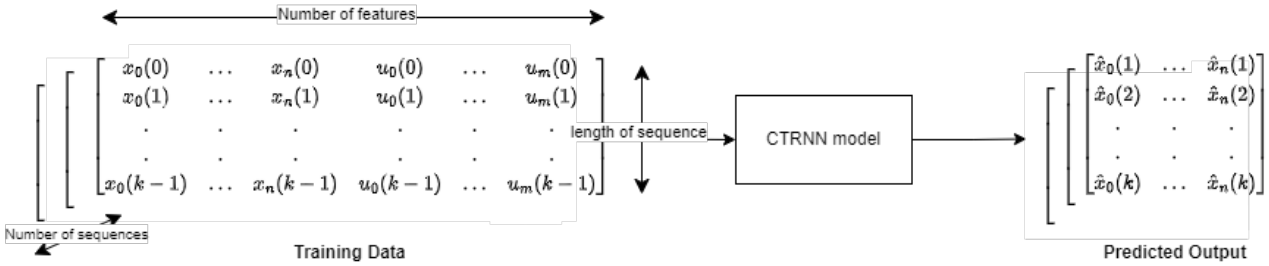

Figure 3.5: RNN Training Procedure

3.1.5.2 Model Training

Training our model consists in finding the optimal weights of equation 3.3 that approximates the nonlinear system with minimal loss criteria. In order to achieve that, we must go through a set of steps that are as follows :

- **Data preparation/Normalisation :** Using the data generated as it is can be troublesome, the scope of the different states may differ creating an imbalance in the model by giving more importance to larger states, that is why normalisation of the data is advised. One of the most Normalisation techniques, and the one that we will be using for our tests, is the Z-score Normalisation (or Standardisation) method, it scales our numerical data to a range of [0;1] as follows :

$$x_{norm} = \frac{x - \mu}{\sigma} \tag{3.7}$$

  Where x is the original feature value, μ is its mean value and σ its standard deviation.

- **Defining a loss function :** The loss function computes a defined difference between the predicted outputs and the actual data, this function will be the base of the optimization process as well as an indicator of the training's precision. The most commonly used loss functions for numerical outputs are **Mean-Squared-Error (MSE) :** $\frac{1}{n}\sum_{i=1}^{n}(x_i - \hat{x}_i)^2$ and **Mean-Absolute-Error (MAE) :** $\frac{1}{n}\sum_{i=1}^{n}|x_i - \hat{x}_i|$.

- **Choosing an optimization algorithm :** Optimization algorithms aim to update the networks parameters in order to minimize the loss function defined previously. Usual optimization algorithms include **Gradient Descent (GD) :** one of the most basic algorithms where the parameters are updated based on the loss function's negative gradient. **Stochastic Gradient Descent (SGD) :** a variant of the classical GD with more frequent updates by varying parameters for each training sample (or part of it), leading to faster convergence time. **Adaptive Moment Estimation (Adam) :** An advanced optimization algorithm that computes adaptive learning rates for each parameter.

- **Choosing an appropriate activation function :** In our application, the choice of activation function plays a major role, as we are attempting to approximate a nonlinear model with a NN that is mostly linear, the non-linearity behaviour comes from the activation function. The use of one or many non-linear activation function requires trial and error, and the most common activation functions are : **Hyperbolic Tangent** $tanh = \frac{e^x - e^{-x}}{e^x + e^{-x}}$, **Sigmoid** $\sigma(x) = \frac{1}{1+e^{-x}}$, **Rectified Linear Unit** $ReLU = \max(0, x)$.

- **Implementing the code :** Turning the mentioned theory into practical code is a tedious task, however using the python torch library and its neural network module, we are able to create custom neural networks structures. In our case we must define the initial parameters, the input layers, the hidden layers, the output layer the activation function, the forward pass, the recurrence and finally the back propagation.

- **Hyper-parameter tuning :** The final step is to tune the parameters of the model, this can be done by trial and error, but a more efficient method would be to use a tuning method such as **grid search** or **random search**.

3.1.5.3 Model Evaluation

After training our model, it is important to analyse and measure its performancec, for this we usually look at these main criteria:

- **Evaluation metrics :** These metrics give us a direct indicator of our model's precision by computing the difference between the predicted and true outputs. As mentioned earlier, the main metrics used are **MSE** and **MAE**.

- **Generalisation :** When training a model the goal is not to memorise results, but to learn patterns within the data, which is why we look at the models ability to generalise on unseen data. In order to achieve this we use validation data, or cross-validation methods so that when updating the loss function, the accuracy is measured on new data that was not part of the training set.

- **Underfitting/Overfitting :** Underfitting is when the loss for both the training and validation sets is high, showing that the model could not learn any pattern in the data and indicating that major changes must be done to our parameters. Meanwhile, overfitting is when the training loss is low, while the validation loss is high meaning that instead of understanding the data, our model is memorising it, in which case the validation method and the data structure should be modified.

3.1.6 Results

For this part we will be considering the first model of the Jetstream J31 of equation 1.13 over the model of AAI Aerosonde, due to the computation power required to generate and learn the patterns from the data.

We implemented the system in Simulink Matlab then run simulations for a fixed sampling period of $h_c = 0.4s$ and simulation time of $500s$. The solver used was the classical Euler method, and the input signal being a uniform random signal of sampling/holding time $\Delta t = 10s$. For each simulation we vary the initial conditions as well as the random signal generator's seed within a predefined stability space $\Omega_{p'} : 50 < x_1 < 150, \pi/100 < x_2 < \pi/50, 1/6100 < x_3 < 1/6000$ with x3 in here being the inverse of mass. And finally we run the process for 1600 iterations saving the data points each time resulting in over 2 million data points. Data was divided into training, validation and testing sets $(70\%, 20\%, 10\%)$ respectively. And python was used for implementation.

Below, The list of parameters and results of simulations are presented:

Parameter	Value
h_c	0.4s
Δt	10s
α	1
$f(.)$	tanh
Optimizer	Adam
Learning rate	0.1
Hidden layers size	3
Epochs	3000
Loss	MAE

Table 3.1: RNN parameters

Training loss	**Validation loss**	**Testing loss**
$3,09.10^{-3}$	$3,08.10^{-3}$	$3,15.10^{-3}$

Table 3.2: RNN Identification results

Weight matrices W_{in},W_h and W_{out} were exported into Matlab to compare the identified model obtained with the actual aircraft system. Tests were conducted under various initial conditions, as outlined below:

Case 1: $x_0 = [100(kts); \pi/100(rad); 1/6071(Kg^{-1})]$. (Figure 3.6)

Case 2: $x_0 = [150(kts); \pi/150(rad); 1/6100(Kg^{-1})]$. (Figure 3.7)

Case 3: $x_0 = [50(kts); \pi/50(rad); 1/6000(Kg^{-1})]$. (Figure 3.8)

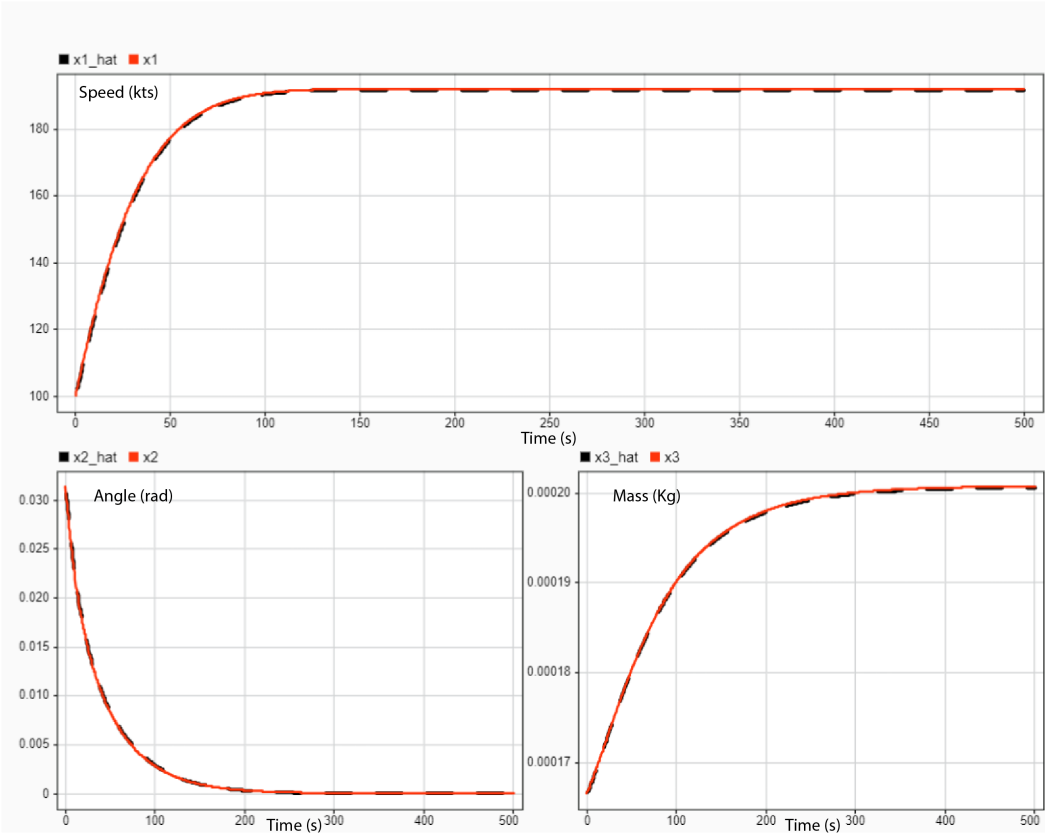

Figure 3.6: True and RNN predicted states **(Case1)**

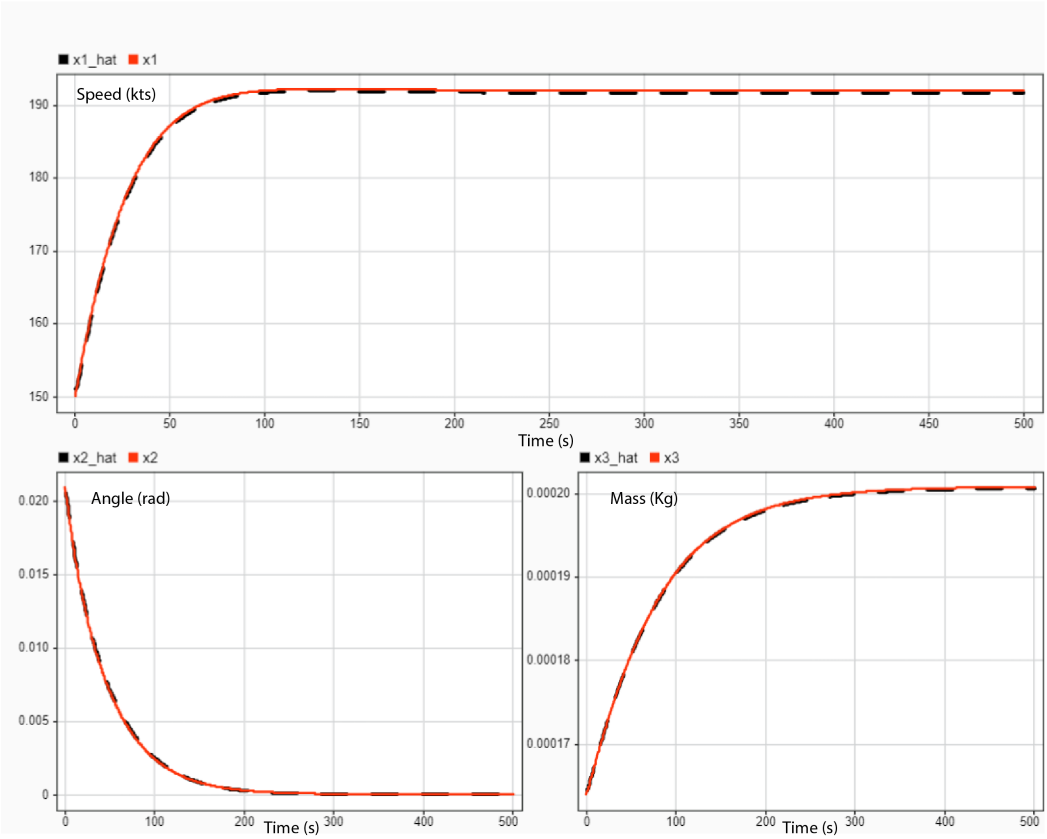

Figure 3.7: True and RNN predicted states **(Case2)**

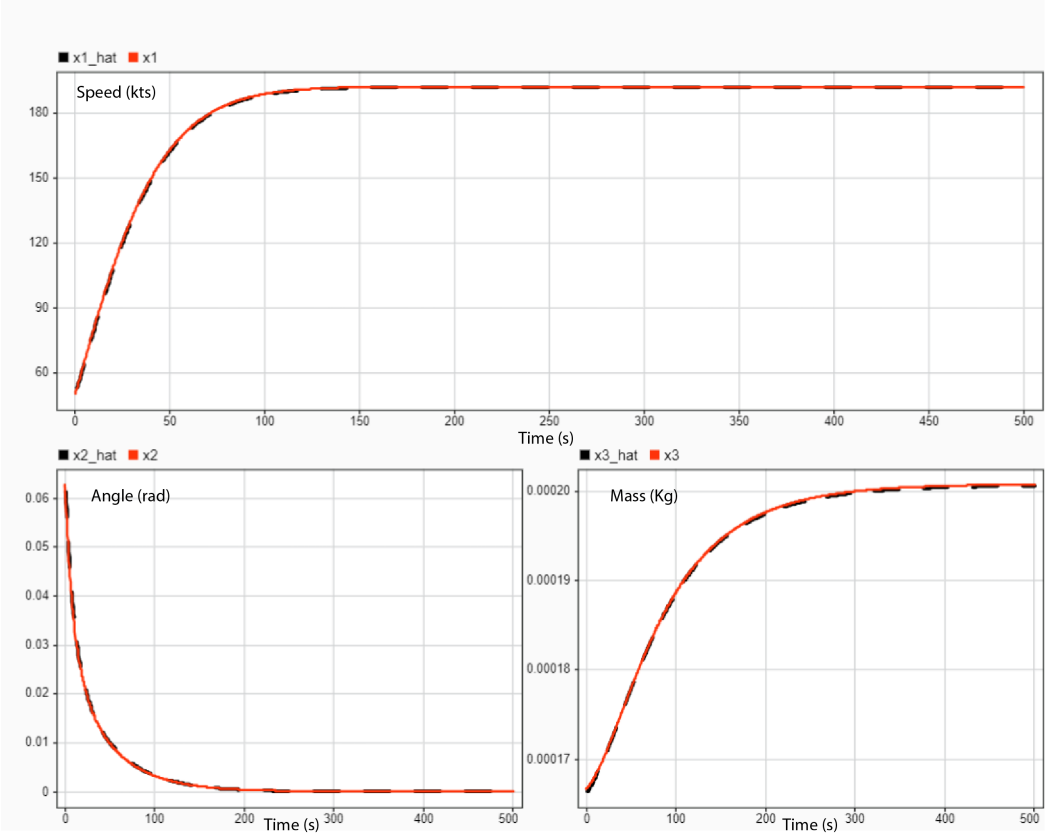

Figure 3.8: True and RNN predicted states **(Case3)**

3.1.7 Discussion

The results of the Continuous-Time Recurrent Neural Network (CTRNN) model were highly promising. The model achieved a very low loss value, particularly on the testing set, demonstrating its strong generalization capability by performing well with new, unseen data. Additionally, the identified model was tested under three different initial conditions, yielding similarly positive outcomes. These results indicate that the trained CTRNN effectively replicates the system's behavior, validating its robustness and reliability.

3.2 Sparse Identification of Nonlinear Dynamics (SINDy)

SINDy is a data-driven nonlinear system identification method that leverages the principle of sparsity. This principle suggests that dynamical model equations can be reconstructed using only a few candidate features. The method aims to extract the exact dynamical equations from time-series data using regression-based techniques.

3.2.1 Mathematical Overview

Consider the system of form :

$$\dot{x}(t) = f(x, u) \tag{3.8}$$

Where $x(t) \in \mathbb{R}^n$ is the state vector, and f(.) is a nonlinear function of the states. $x(t)$ and its derivative $\dot{x}(t)$ are sampled equi-distantly in time with $t = t_1, t_2, ..., t_m$, and are arranged into the following matrices :

$$x(t) = \begin{bmatrix} x^T(t_1) \\ x^T(t_2) \\ \vdots \\ x^T(t_m) \end{bmatrix} = \begin{bmatrix} x_1(t_1) & x_2(t_1) & \dots & x_n(t_1) \\ x_1(t_2) & x_2(t_2) & \dots & x_n(t_2) \\ \vdots & \vdots & \ddots & \vdots \\ x_1(t_m) & x_2(t_m) & \dots & x_n(t_m) \end{bmatrix} \tag{3.9}$$

$$\dot{x}(t) = \begin{bmatrix} \dot{x}^T(t_1) \\ \dot{x}^T(t_2) \\ \vdots \\ \dot{x}^T(t_m) \end{bmatrix} = \begin{bmatrix} \dot{x}_1(t_1) & \dot{x}_2(t_1) & \dots & \dot{x}_n(t_1) \\ \dot{x}_1(t_2) & \dot{x}_2(t_2) & \dots & \dot{x}_n(t_2) \\ \vdots & \vdots & \ddots & \vdots \\ \dot{x}_1(t_m) & \dot{x}_2(t_m) & \dots & \dot{x}_n(t_m) \end{bmatrix} \tag{3.10}$$

A set of functions $\Theta(x)$ called the feature library that contains the nonlinear candidate functions (polynomials, sinusoidals, fractions. etc) was then defined:

$$\Theta(x) = \begin{bmatrix} f_0(x) & f_1(x) & f_2(x) & \dots \end{bmatrix} \tag{3.11}$$

Each function $f_i(x)$ represents a candidate function, and the liberty of choosing these functions varies from application to application, and it may be chosen based on physical properties of the system.

Knowing that only few of these candidates are actually present in the right-hand size of equation 3.8, we can define the sparse weight vectors (coefficients) $\Xi = [\xi_1, \xi_2, ..., \xi_n]$ such that :

$$\dot{x} = \Theta(x)\Xi \tag{3.12}$$

With Ξ containing mostly zeros except for the actual terms present in the real system as highlighted in the figure 3.9. In this figure, both the red and blue columns in the left most vector represent single states through time, while in the right most vector the red and blue dots represent the non-zero parameters in our matrix, which defines the nonlinear candidates chosen to represent our states (red for the first state and blue for the second).

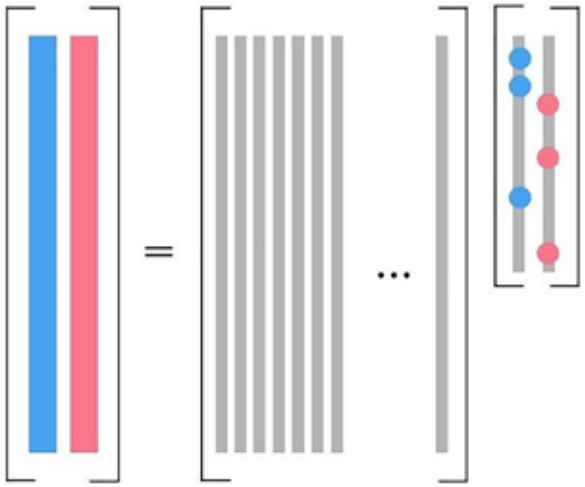

Figure 3.9: SINDy Algorithm mathematical overview [3]

In the SINDy algorithm, the control input u is treated as a candidate variable similar to the states. However, unlike the states, its derivative is not calculated.

This becomes an optimization problem of finding the optimal weights Ξ that best embed $\dot{x}$. For this, sparsity promoting L_1 regularisation is used with least squares errors to find out the parameters:

$$\xi_k = \underset{\xi'_k}{argmin} ||\dot{x_k} - \Theta(x)\xi'_k||_2 + \lambda||\xi'_k||_1 \tag{3.13}$$

3.2.2 Main Challenges

The main challenges encountered with this method can be divided into two separate points :

1. **Choice of coordinates :** Choosing the correct set of variables to measure that represent the system can be difficult, as omitting important variables may lead to incomplete models, while adding irrelevant variables can make it unnecessarily complicated. Typically, an idea of the systems most representative variables must already be acquired by the user by having an understanding of the application itself and its context.
2. **Choice of candidates :** Another crucial choice to make, is the set of candidates to use. The SINDy method assumes dynamics can be represented as a sparse linear combination of functions $(x, x^2, cos(x), sin(x), e^x, etc)$. To overcome this problem, prior knowledge of the system can be used to determine which functions to use, if such information is unavailable we can use general functions such as the polynomial and sinusoidal functions, however extracting the exact dynamics this way cannot be slightly difficult.

3.2.3 Results

The SINDy Algorithm was implemented using python, to identify the J31 Jetstram system 1.13 using the same generated data from the previous CTRNN Algorithm.

After trying many combinations of the candidates, and using a grid search to optimize the parameters of the model, we obtained the following set of hyper-paramters: The chosen algorithm

Parameter	Value
Optimization algorithm	STLSQ
Threshold	10^{-3}
Maximum iterations	20
Normalisation	True
Learning rate	0.05

Table 3.3: SINDy parameters

Sequential Threshold Least Squares (STLSQ) iteratively performs least-squares regression and then thresholds the small coefficients to zero, which promote sparsity in the identified model. The trained model achieved an accuracy of 0.99, we then plotted the actual and predicted states for various initial conditions as shown below:

Case 1: $x_0 = [116; \pi/73; 1/6083]$. (Figure 3.10)

Case 2: $x_0 = [66; \pi/50; 1/6028]$. (Figure 3.11)

Case 3: $x_0 = [99; \pi/97; 1/6066]$. (Figure 3.12)

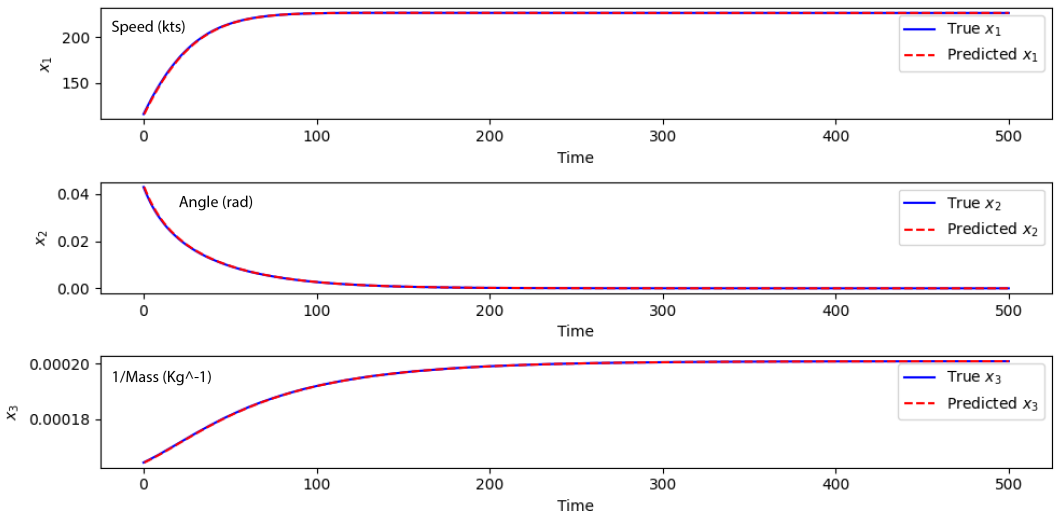

Figure 3.10: Real and estimated SINDy states **(Case 1)**

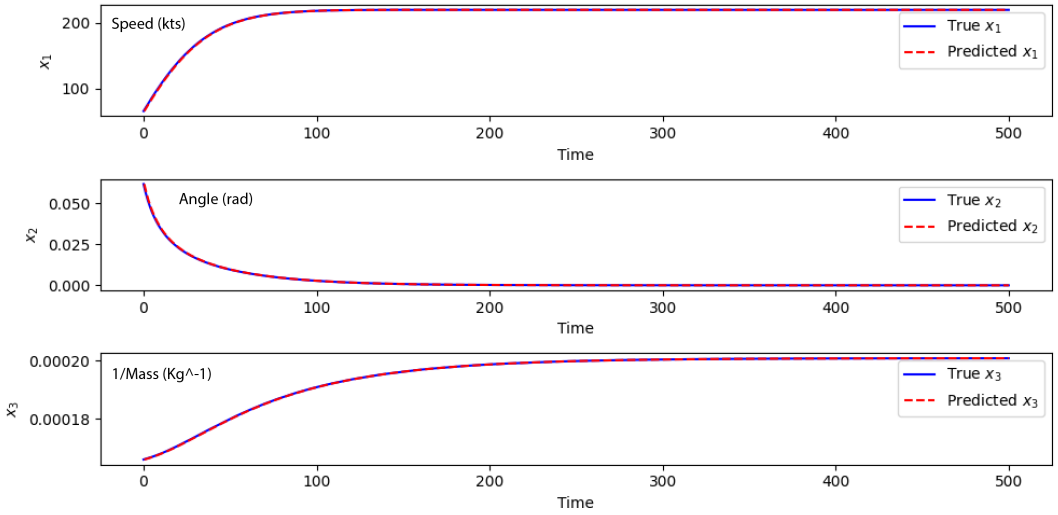

Figure 3.11: Real and estimated SINDy states **(Case 2)**

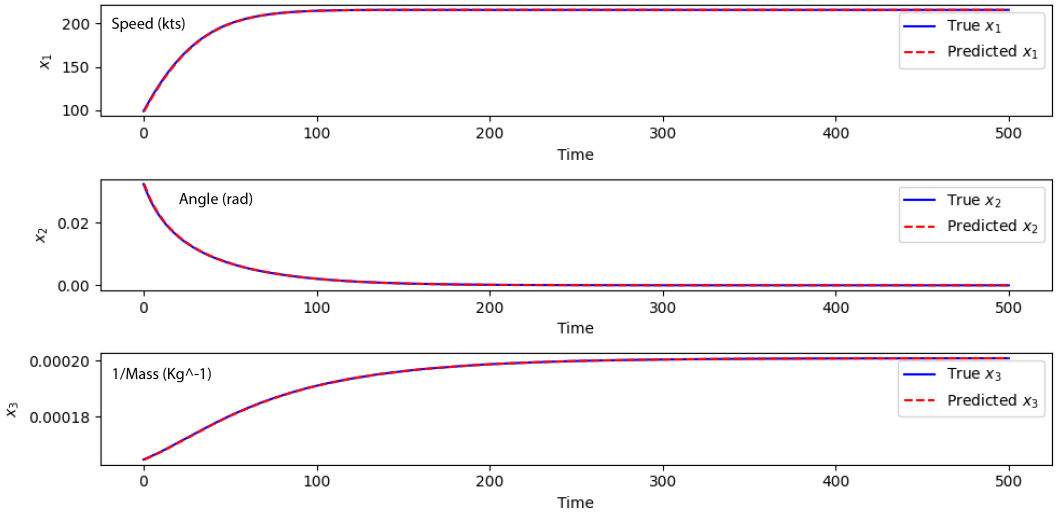

Figure 3.12: Real and estimated SINDy states **(Case 3)**

3.2.4 Discussion

The results of the Sparse Identification of Nonlinear Dynamics (SINDy) model were highly promising. The model accurately reconstructed the functions and coefficients of the real system. Additionally, the identified model was tested under three different initial conditions, yielding similarly positive outcomes. These results indicate that the SINDy algorithm effectively replicates the system's behavior, validating its high performance.

The dynamical model was reconstructed, and some additional terms with negligible coefficients were present. Due to their negligible coefficients, these terms have very minimal impact on the dynamics. This explains why the states of the identified model perfectly matched those of the real system. With more computational power, these negligible terms can be easily thresholded and removed for a more refined model.

It's important to note that in practice, it's essential to pre-treat the data to accurately calculate derivatives. Filters like the Kalman filter and low-pass filters are commonly used to clean and denoise the signals.

3.3 Comparative study

In this chapter, we implemented two learning-based identification methods: the Continuous-Time Recurrent Neural Network (CTRNN) algorithm and the Sparse Identification of Nonlinear Dynamics (SINDy) algorithm. Both methods performed well, though notable differences exist between them.

Firstly, the primary distinction is that CTRNN approximates nonlinear systems without prior knowledge of the physical properties of the system, making the procedure broadly applicable. However, SINDy aims to find the exact dynamical representation of a system, which make it complicated in identifying suitable candidate functions, which is difficult without prior knowledge of the system's characteristics. Secondly, it is important to note that the CTRNN algorithm requires defining a region of stability or interest. This ensures a bounded small error of identification within that region. On the contrary, the SINDy algorithm attempts to reconstruct the exact model directly, the same functions of $f(x, u)$ and its same coefficients, suggesting a greater generalization ability.

Another point worth mentioning is that both methods require extensive amounts of data. This is reasonable given the complexity of identifying nonlinear differential equations and because the methods are fully data-driven and learning-based.

	RNN	SINDy
Prior Knowledge	No Need	Hard in absence of prior knowledge
Objective	Approximating model	Finding exact model
Limits	Region of stability	No limits

Table 3.4: Summary of the comparaison between RNN and SINDy based identification

3.4 Conclusion

In this chapter, we defined and analyzed two learning-based identification algorithms: the Continuous-Time Recurrent Neural Network (CTRNN) algorithm and the Sparse Identification of Nonlinear Dynamics (SINDy) algorithm. We tested both methods and conducted a detailed comparison of their characteristics. Although these methods are still under development, the results are promising and provide valuable insights for future research.

One point worth mentioning is that the identified models can be used in observer design by incorporating correction terms, such as a linear combination or sliding manifolds of measured errors. This approach is particularly beneficial when mathematical first principles models are not available.

Looking ahead, the primary objective is to reduce the amount of data required for training these algorithms. This reduction will facilitate their application in real-world scenarios without the need for extensive resources. Additionally, less data will alleviate the heavy computational demands currently associated with these methods. Another focus will be on updating or merging the existing algorithms to enhance generalization capabilities and eliminate region-specific limitations. Furthermore, developing new frameworks will be investigated to improve and shorten the training time of these learning methods.

Pursuing these directions will significantly enhance the reliability and effectiveness of learning-based identification algorithms, making them more practical for widespread use.

Part II

Advanced Optimization Of Smart Sensor Networks In Internet Of Things Technologies

Introduction

In Context

The Internet of Things (IoT) is a recent technology that facilitates the interconnection of various types of physical objects with each other and with the internet. This rapidly advancing and remarkable technology has become an integral part of our everyday lives. Indeed, all electronic devices we use are either already intelligent or are on the verge of becoming so. IoT finds extensive applications and plays pivotal roles in numerous domains, including healthcare, urban planning, agriculture, environmental monitoring, energy management, retail, home automation, and education.

The current Internet of Things operates primarily through sensors, actuators, and other connected objects placed within infrastructures or physical objects. These objects collect, transmit, and receive specific data, which is then wirelessly uploaded to IoT platforms. The data is then analyzed and enriched to address user queries and/or enhance the efficiency of devices within the network.

The number of objects connected to IoT networks continues to grow exponentially. Currently, we have millions of connected objects generating a vast amount of data, leading to numerous challenges related to energy conservation, scalability, data security, and quality assurance.

One of the major challenges of IoT is energy conservation. Indeed, sensor nodes are small components that remain active for a limited duration due to the capacity of their energy sources, primarily batteries. In most applications, recharging or replacing the battery is problematic, if not impossible. Consequently, when the sensor node's battery is depleted, it is eventually removed from the network, causing a disruption in its services. The simultaneous disintegration of sensor nodes leads to a degradation in network performance and, over time, renders the network obsolete.

Given the characteristics and limitations of sensor nodes, it is crucial to optimize the utilization of their energy resources to enable data collection and transmission without compromising the overall performance of the network. Numerous research efforts have been conducted in the context of energy conservation with the aim of extending the lifespan of IoT networks. However, few of the proposed techniques are applicable to large-scale networks, and even fewer address the objectives when the base station is located outside the network.

Research Problem

Currently, the number of objects connected to IoT networks amounts to millions, with projections indicating that this figure will reach billions in the years to come. Hence, it's crucial to optimize the energy efficiency of IoT networks to facilitate data collection and transmission without compromising the overall network performance.

The central query explored in this study is: **How might operational research methodologies and the latest advancements in information processing technology be employed to optimize energy consumption within IoT networks?**

In her article titled "Reducing Wireless Sensor Networks Energy Consumption Using P-Median Modelling and Optimization"[30] Mahmoudi Yousra proposed, for the first time, a comprehensive p-median-based mathematical model for the clustering problem in wireless sensor networks and IoT. The model defines a set of decision variables to express the objective of minimizing energy consumption as a mathematical function, considering factors such as node residual energy and other constraints. The proposed clustering will establish a routing strategy that simultaneously addresses multiple objectives.

- Energy efficiency by dynamically selecting cluster heads and assigning member nodes to cluster heads to minimize the distance between member nodes and the base station.
- Data security by employing single-hop transmissions between member nodes and the cluster head, and between the cluster head and the base station.
- Network scalability as clustering facilitates seamless integration of new sensor nodes.

Objectives

In this study, our objective is to address the proposed p-median problem using metaheuristic algorithms. We employ for this three relatively new and previously untilized optimization algorithms. such as Grey Wolf Optimizer (2014)[4], Whale Optimizer (2016)[5], and Puma Optimizer Algorithms (2024)[6] for tackling this exact p-median problem for energy optimization.

Our optimization problem is a 2^n non-convex exponentially increasing problem, in which traditional algorithms may take from months to years to solve the problem, unlike metaheuristics which can provide good results in only few minutes.

Our study focused on non-mobile sensor nodes and a base station with well-known geographical positions. Sensor positions were arranged in both 2D and 3D frameworks. We adopted the same assumptions as Mahmoudi Yousra in her work [30].

- In terms of energy capacities, the base station is powered by a continuous and unlimited energy source, while sensor nodes have restricted, non-expandable, and non-exchangeable energy resources.
- Regarding energy consumption, a sensor node utilizes energy for three main tasks: sensing, processing, and communication.
- In our study, we only consider the energy required for communication since it constitutes the largest portion of energy consumed by a sensor node.

Structure of This Part

This part begins with Chapter Four, delving into the state of the art. Chapter Five follows, presenting definitions of the Internet of Things and its architectures and characteristics, along with an introduction to Genetic Algorithms, Grey Wolf Optimization Algorithm, Whale Optimization Algorithm, and Puma Optimization Algorithm. Moving forward to Chapter Six, we introduce the first mathematical model of energy optimization using the p-median problem, along with the second mathematical model, both proposed by Yousra Mahmoudi. We also present our main optimization results using metaheuristic algorithms for both mobile and non-mobile sensors in both 2D and 3D frameworks. Finally, this part concludes in the last chapter, followed by proposed avenues for future work. The list of references is provided at the end.

Chapter 4

State of The Art

In [31], a brief historical overview of the IoT's evolution and a comprehensive survey of optimization challenges in IoT networks have been provided. Techniques of IoT energy efficiency optimization can be classified as follows:

Radio Optimization Techniques

These techniques influence signal quality, impacting coverage and robustness by dynamically selecting channels and programming cooperative reception/transmission. A novel context-aware approach utilizing a Q-learning algorithm has been published [32], determining the wireless connection type, selecting the data processing unit, and determining the data quantity to be scanned to meet energy consumption, cost, response time, and security objectives. A distributed learning approach to operational control in long-range technology has been outlined. This approach adapts device communication parameters to the environment to minimize energy consumption and data collision over shared channels [33]. An algorithm employing the L system to draw a Hilbert curve has been developed for adaptively and dynamically optimizing a central IoT network with expanded Wi-Fi transmission range. This includes parameter sharing to enhance the energy efficiency of a smart home [34]. Research has investigated the feasibility of applying IoT technology to supervise and enhance the efficiency of energy and spectrum usage in broadcasting networks, particularly utilizing the ultra-high frequency band [35]. This involved a brand new network design with an IoT retroactive circuit and a 2-step long-range/NB-IoT-based algorithm that chooses the best base station set among several possibilities to minimize infrastructure and then optimizes power consumption by linking users to the operational base stations through the best routing pattern that minimizes both data loss and the effective isotropic radiated power. Power consumption by a broadcaster in Havana was thus reduced by 15–16.3% compared to current digital terrestrial multimedia broadcast networks while spectrum usage efficiency was increased by 32–35% and the availability of TVWS channel was increased by 34% in 90% of instances. A small cell allocation plan ensuring the mutual objectives of optimizing energy efficiency and maximizing data rate has been introduced for an IoT application in a smart city [36]. The proposed scheme is obtained based on the formulation of the problem as an integer programming multiobjective optimization problem wich has been resolved by a new algorithm based on the fusion of the branch-and-bound algorithm and the non-dominated sorting genetic metaheuristic (NSGAII) [37].

4.1 Data Reduction Techniques

These methods tackle latency by reducing data transmission and employing aggregation techniques to enhance service quality. Resource allocation in cloud computing is executed through the whale optimization heuristic, drawing inspiration from the collective hunting behavior of humpback whales [38]. Novel examples of environmentally friendly Narrowband Internet of Things (NBIoT) technologies include an energy-efficient approach for optimizing water usage in irrigation and an Energy-Efficient Adaptive Health Monitoring System (E2AHMS) that dynamically distributes transmission power needed for patient monitoring [39]. Additionally, Huang and al in [40] offer a toolbox for modeling and simulating various resource management methods in IoT, edge computing, and fog computing networks.

4.2 Sleep and Wake Techniques

The problem of scheduling and controling productive activity periods in IoT has been addressed using Max-plus dioid algebra with tuning systems that adjust and schedule activity and sleep periods to avoid useless wake-up calls and provide data as required by the application. A timed event graph was employed to model the network, where each vertex signifies a node state, edges denote connections between states, and the weight on each edge indicates transition times between states. This system enables the optimization of energy consumption, extends the lifetime of IoT devices, minimizes unnecessary data generation, storage, and transmission, and ensures data quality preservation [41].

4.3 Energy Efficient Routing

Energy-efficient routing significantly impacts IoT scalability and robustness, achieved either through network clustering with appropriate cluster-head selection or by employing efficient data transmission protocols. To address this, the fitness-averaged rider optimization algorithm was introduced for multi-objective cluster-head selection, considering factors like candidate node suitability, remaining energy, and energy balance level [42]. Comparative analysis with state-of-the-art models such as artificial bee colony, genetic algorithm, particle swarm optimization, gravitational search, moth flame optimization, and others confirmed the superior efficiency of this strategy [43][44][45][46][47][48][49].

Moreover, the development of the multi-objective fractional gravitational search algorithm aimed at providing an energy-efficient transmission protocol in IoT networks [50]. This algorithm, integrating fractional theory and gravitational search, iteratively determines cluster-heads based on multiple objectives like distance, energy, delay, and link lifetime. Another approach proposed by Rashedi, Nezamabadi-Pour & Saryazdi [51] introduced a multi-objective metaheuristic for designing energy-efficient routing protocols in IoT networks, named the multi-objective fractional gravitational search. Additionally, Chu, Horng & Chang [52] presented a data transmission balance model for WSN networks utilizing an enhanced version of the ant colony algorithm to optimize multi-hop communication, reducing energy consumption and enhancing network reliability compared to traditional methods. Abbad and al [53] proposed a weighted Markov chain-based clustering protocol to decrease intra-cluster energy consumption, efficiently managing redundant data transmission in high-density sensor regions and selecting sensors for interrogation in low-density areas, thereby extending sensor lifetime. Furthermore,

Mahmoudi, Zioui & Belbachir [54] introduced a novel quantum-inspired clustering metaheuristic to optimize energy consumption in IoT wireless networks. This approach encodes cluster-head selection solutions using qubits and explores the solution space using firefly motion and PSO motion, leading to significant energy savings and computational speedups leveraging quantum computing principles.

4.4 The Problem of Service Composition

The service composition challenge arises frequently in IoT networks when a device alone cannot fulfill a complex user request, necessitating the creation of a composite service. Multi-objective metaheuristic search algorithms, notably the non-dominated sorting genetic algorithm, have been extensively explored to address this issue, consistently providing optimal solutions to the service composition problem [37]. Furthermore, a bi-objective shortest path selection scheme has been employed for IoT service composition, aiming to maximize quality of service (in terms of runtime, cost, and network latency) while minimizing energy consumption. This scheme utilizes a pulse algorithm with four built-in pruning procedures tailored for this purpose (Mirjalili, 2015). Innovatively, a multi-objective genetic optimization algorithm has been proposed to tackle the resource provision and application placement problem in IoT fog computing. This approach considers conflicting criteria such as energy consumption, execution time, and economic cost, aiming to approximate the Pareto front of optimized application-component placements on available fog devices [5]. Comparative evaluations with state-of-the-art methods [55] have demonstrated its effectiveness in terms of communication data size, component CPU workload, quality, and algorithm scalability.

4.5 In Related Research

A highly efficient multi-objective optimization algorithm based on the chaotic ant swarm was developed by Gupta and al [55]. This algorithm redefines the concepts of "Neighbors" and "Neighbor Selecting" rules, incorporating an archive-based approach to enable rapid convergence to the true Pareto front with a well-distributed set of solutions. When applied to various well-known multi-objective optimization problems, the chaotic ant swarm algorithm outperformed state-of-the-art peer algorithms like particle swarm optimization and the non-dominated sorting genetic algorithm. It demonstrated superiority in terms of generational distance, spacing, and error ratio.

It's crucial to note that within the framework of optimizing energy consumption in IoT-enabled networks utilizing clustering architecture, the majority of these methods were tailored for smaller networks where base stations are situated within the area covered by nodes. However, their performance tends to degrade significantly when applied to networks with peripheral base stations.

Chapter 5

Tools and Materials

Optimization involves choosing the best values for certain parameters in a system to fulfill all design requirements while minimizing costs. This challenge is common across all scientific fields and necessitates the creation of innovative algorithms to handle progressively complex problems. Conventional optimization algorithms have limitations such as focusing on single solutions, converging to local optima, and struggling with unknown search spaces. Over the last few decades, many researchers have developed metaheuristics—algorithms specifically designed to overcome these limitations and address unresolved optimization challenges.

Optimization involves identifying decision variables that yield the best possible outcomes for one or more fitness functions. Prior to the advent of heuristic optimization, analytical methods were the main approach to tackling optimization problems. These methods, especially in the context of first and linear combinations, focused on collecting data related to individual or constraint-penalized fitness function variables, as well as primary information about the fitness function and constraint violations. This made it easier to find the exact optimal solution for linear or convex non-linear optimization problems. However, this approach struggled with more complex optimization problems, often leading to issues such as local optima trapping or premature convergence in stochastic or unknown search spaces.

Metaheuristics represent a prominent category of algorithms specifically created to tackle complex optimization problems. They draw on concepts from evolutionary algorithms (EA), swarm intelligence (SI), physics-based methods (PH), and human-based methods (HU). The simplicity of mathematical models in SI is a key factor contributing to its popularity among all these classes. Metaheuristics have become highly popular in the field of optimization and in various other disciplines. Stochastic optimization algorithms, which are part of this category, are widely used across numerous scientific fields and industries.

As the popularity of metaheuristic algorithms for solving diverse problems continues to rise, various types of metaheuristic algorithms have emerged, each with its own characteristics and approaches to optimization. These algorithms draw inspiration from a range of natural phenomena, including animal and human behavior, principles of physics, and concepts of evolution. Depending on their sources of inspiration, different algorithms have been devised. By mathematically modeling natural behaviors, these algorithms have spurred the development of new methodologies and techniques in optimization. The literature has seen the proposal of numerous optimization approaches, each influenced by unique inspirations. The sources of inspiration for metaheuristic algorithms can be broadly classified into several categories based on natural sources.

The main sources of inspiration for metaheuristic algorithms stem from the realm of animal life and behavior. These algorithms, influenced by the collective behaviors observed in social insects and animals, have significantly advanced the field of metaheuristic algorithms. For instance, the Golden Eagle Optimizer (GEO) [56] is a notable approach inspired by nature, mimicking the hunting intelligence of golden eagles by adjusting its speed during various spiral trajectory phases. Another innovative method is the Whale Optimization Algorithm (WOA) [5], developed based on the social behavior of humpback whales during bubble-net hunting. Similarly, the SailFish Optimizer (SFO) [57], drawing inspiration from the hunting activity of sailfish groups, employs two populations: one for improving search accuracy and another for diversifying search areas. Furthermore, the optimization technique inspired by Coaties' behavior [58] models exploration and exploitation stages, demonstrating superior performance across various objective functions compared to 11 other algorithms. Additionally, the Ant Lion Optimizer (ALO) [59] mirrors the hunting process of antlions through five key prey-hunting steps. The Pathfinder Algorithm (PFA) [60] tackles optimization challenges by emulating the collective dynamics of animal groups and simulating swarm leadership hierarchies to locate optimal resources or prey.

In our research, we aimed to employ modern and efficient algorithms, leading us to select the Grey Wolf Optimization Algorithm[4], initially introduced by Mirjalili et al. in 2014. Additionally, we opted for the Whale Optimization Algorithm[5], first proposed by Mirjalili and Andrew Lewis in 2016. Lastly, we incorporated the Puma Optimization Algorithm[6], which was recently introduced by Benyamin Abdollahzadeh and Nima Khodadadi in 2024.

5.1 Grey Wolf Optimization

For this Algorithm, the inspiration of the proposed method is first discussed. Then, the mathematical model is provided.

5.1.1 Inspiration

The grey wolf, a member of the Canidae family, is regarded as an apex predator, placing it at the top of the food chain. Grey wolves typically live in packs, with an average group size of 5-12. Notably, these packs have a strict social hierarchy, as illustrated in Fig. 1. The leaders, a male and a female known as alphas, are primarily responsible for making key decisions such as hunting, sleeping locations, and wake-up times, which are then followed by the pack. Despite this, some democratic behavior has been observed, where an alpha may follow the lead of other pack members. During gatherings, the pack acknowledges the alpha by lowering their tails. The alpha wolf, also referred to as the dominant wolf, has its directives obeyed by the entire pack and is the only one allowed to mate. Interestingly, the alpha is not always the strongest member but excels in managing the pack, highlighting that organization and discipline are more critical to a pack's success than sheer strength.

The second level in the grey wolf hierarchy is occupied by the beta wolves. These subordinate wolves assist the alpha in decision-making and various pack activities. A beta wolf, which can be either male or female, is typically the most suitable candidate to become an alpha if one of the current alphas dies or becomes too old. While the beta must respect the alpha, it also has authority over the lower-ranking wolves. Acting as an advisor to the alpha, the beta also disciplines the pack, reinforces the alpha's commands, and provides feedback to the alpha.

The lowest-ranking grey wolf is the omega, which serves as the scapegoat of the pack. Omega wolves must submit to all the dominant wolves and are the last to eat. Although the omega might seem unimportant, its presence is crucial for maintaining pack harmony. Without an omega, the pack often experiences internal conflicts and issues, as the omega helps vent the other wolves' aggression and frustration, thus supporting the dominance structure. Additionally, omegas sometimes take on the role of babysitters within the pack.

Wolves that are neither alphas, betas, nor omegas are referred to as subordinates, or deltas in some references. These wolves must submit to the alphas and betas but have dominance over the omega. This group includes scouts, sentinels, elders, hunters, and caretakers. Scouts watch the territory boundaries and alert the pack to any danger. Sentinels ensure the pack's safety. Elders are experienced wolves who were once alphas or betas. Hunters assist the alphas and betas in hunting prey and providing food for the pack. Caretakers look after the weak, ill, and injured wolves within the pack.

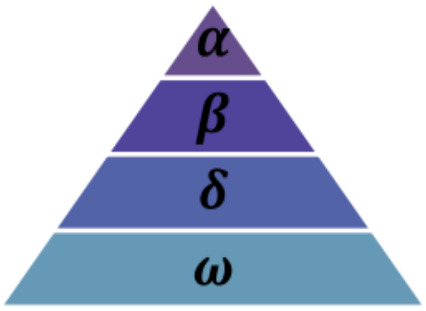

Figure 5.1: Hierarchy of grey wolf[4].

In addition to the social hierarchy, group hunting is another fascinating social behavior of grey wolves. The main phases of grey wolf hunting are as follows:

- Tracking, chasing, and approaching the prey.
- Pursuing, encircling, and harassing the prey until it stops moving.
- Attacking the prey.

In this algorithm, the hunting technique and the social hierarchy of grey wolves are mathematically modeled to design the Grey Wolf Optimizer (GWO) and perform optimization.

5.1.2 Mathematical Model

5.1.2.1 Social hierarchy

To mathematically model the social hierarchy of wolves when designing the Grey Wolf Optimizer (GWO), the authors considered the fittest solution as the alpha (α). Consequently, the second and third best solutions are named beta (β) and delta (δ), respectively. The rest of the candidate solutions are assumed to be omega (ω). In the GWO algorithm, the hunting (optimization) process is guided by α, β, and δ, with the ω wolves following these three leaders.

5.1.2.2 Encircling prey

As mentioned above, grey wolves encircle their prey during the hunt. To mathematically model this encircling behavior, the following equations are proposed: [4]

$$\vec{D} = |\vec{C} \cdot \vec{X}_p(t) - \vec{X}(t)| \tag{5.1}$$

$$\vec{X}(t+1) = \vec{X}_p(t) - \vec{A} \cdot \vec{D} \tag{5.2}$$

Where: $\vec{X}_p(t)$ is the position vector of the prey, $\vec{X}(t)$ is the position vector of a grey wolf, $\vec{A}$ and $\vec{C}$ are coefficient vectors, t denotes the current iteration.

These equations help simulate the grey wolves' encircling mechanism in the optimization process.

The vectors $\vec{A}$ and $\vec{C}$ are calculated as follows: [4]

$$\vec{A} = 2\vec{a} \cdot \vec{r}_1 - \vec{a} \tag{5.3}$$

$$\vec{C} = 2 \cdot \vec{r}_2 \tag{5.4}$$

Where: $\vec{a}$ is a vector that linearly decreases from 2 to 0 over the course of iterations, $\vec{r}_1$ and $\vec{r}_2$ are random vectors in the range [0, 1].

These calculations allow the GWO algorithm to simulate the adaptive and exploratory behavior of grey wolves during the optimization process.

To observe the impact of Equations (5.1) and (5.2), a two-dimensional position vector and some potential neighbors are depicted in Fig. 2(a). As illustrated, a grey wolf located at (X, Y) can update its position based on the prey's position (X*, Y*). By adjusting the values of the $\vec{A}$ and $\vec{C}$ vectors, the wolf can move to various positions around the best agent from its current position. Fig. 2(b) shows the potential updated positions of a grey wolf in 3D space. The random vectors r1 and r2 enable the wolves to reach any position within the range depicted in Fig. 2. Thus, a grey wolf can randomly update its position within the vicinity of the prey using Equations (5.1) and (5.2). This concept can be extended to an n-dimensional search space, where the grey wolves navigate within hyper-cubes (or hyper-spheres) around the best solution found so far.

5.1.2.3 Hunting

Grey wolves possess the ability to detect the location of prey and encircle them, with the hunt typically being led by the alpha. Occasionally, the beta and delta wolves also participate in the hunt. However, in an abstract search space, the location of the optimum (prey) is unknown. To mathematically simulate the hunting behavior of grey wolves, we assume that the alpha (best candidate solution), beta, and delta have better knowledge about the potential location of the prey. Consequently, we save the top three best solutions obtained so far and require the other search agents (including the omegas) to update their positions based on the positions of these top search agents. The following formulas are proposed for this purpose: [4]

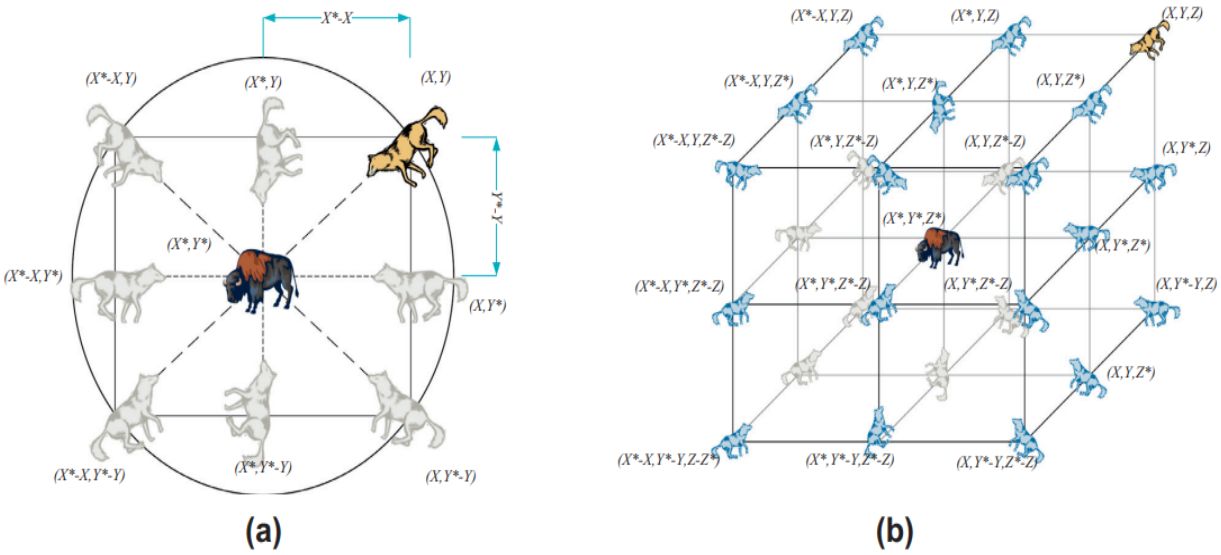

Figure 5.2: 2D and 3D position vectors and their possible next locations[4].

$$\vec{D}_\alpha = |\vec{C}_1 \cdot \vec{X}_\alpha - \vec{X}| \tag{5.5}$$

$$\vec{D}_\beta = |\vec{C}_2 \cdot \vec{X}_\beta - \vec{X}| \tag{5.6}$$

$$\vec{D}_\delta = |\vec{C}_3 \cdot \vec{X}_\delta - \vec{X}| \tag{5.7}$$

$$\vec{X}_1 = \vec{X}_\alpha - \vec{A}_1 \cdot \vec{D}_\alpha \tag{5.8}$$

$$\vec{X}_2 = \vec{X}_\beta - \vec{A}_2 \cdot \vec{D}_\beta \tag{5.9}$$

$$\vec{X}_3 = \vec{X}_\delta - \vec{A}_3 \cdot \vec{D}_\delta \tag{5.10}$$

$$\vec{X}(t+1) = \frac{\vec{X}_1 + \vec{X}_2 + \vec{X}_3}{3} \tag{5.11}$$

Where $\vec{X}_\alpha$, $\vec{X}_\beta$, and $\vec{X}_\delta$ are the positions of the alpha, beta, and delta wolves, respectively. $\vec{C}_1$, $\vec{C}_2$, and $\vec{C}_3$ are random coefficient vectors. $\vec{A}_1$, $\vec{A}_2$, and $\vec{A}_3$ are calculated using the vector $\vec{a}$ and random vectors $\vec{r}_1$, $\vec{r}_2$, as defined earlier.

These formulas guide the other wolves to move towards the alpha, beta, and delta, effectively simulating the hunting and encircling behavior in the optimization process.

Fig. 3 illustrates how a search agent updates its position relative to the alpha, beta, and delta in a 2D search space. As shown, the final position of the agent will be at a random point within a circle defined by the positions of the alpha, beta, and delta. Essentially, the alpha, beta, and delta estimate the prey's location, and the other wolves update their positions randomly around this estimated position.

5.1.2.4 Attacking prey (Exploitation)

As previously mentioned, grey wolves complete the hunt by attacking the prey when it stops moving. To mathematically model this approach, we decrease the value of $\vec{a}$. Consequently, the fluctuation range of $\vec{A}$ also decreases with $\vec{a}$. Specifically, $\vec{A}$ is a random value within the interval [-2a, 2a], where a decreases from 2 to 0 over the course of iterations. When the random values of $\vec{A}$ are within [-1, 1], the next position of a search agent can be anywhere between its current position and the prey's position.

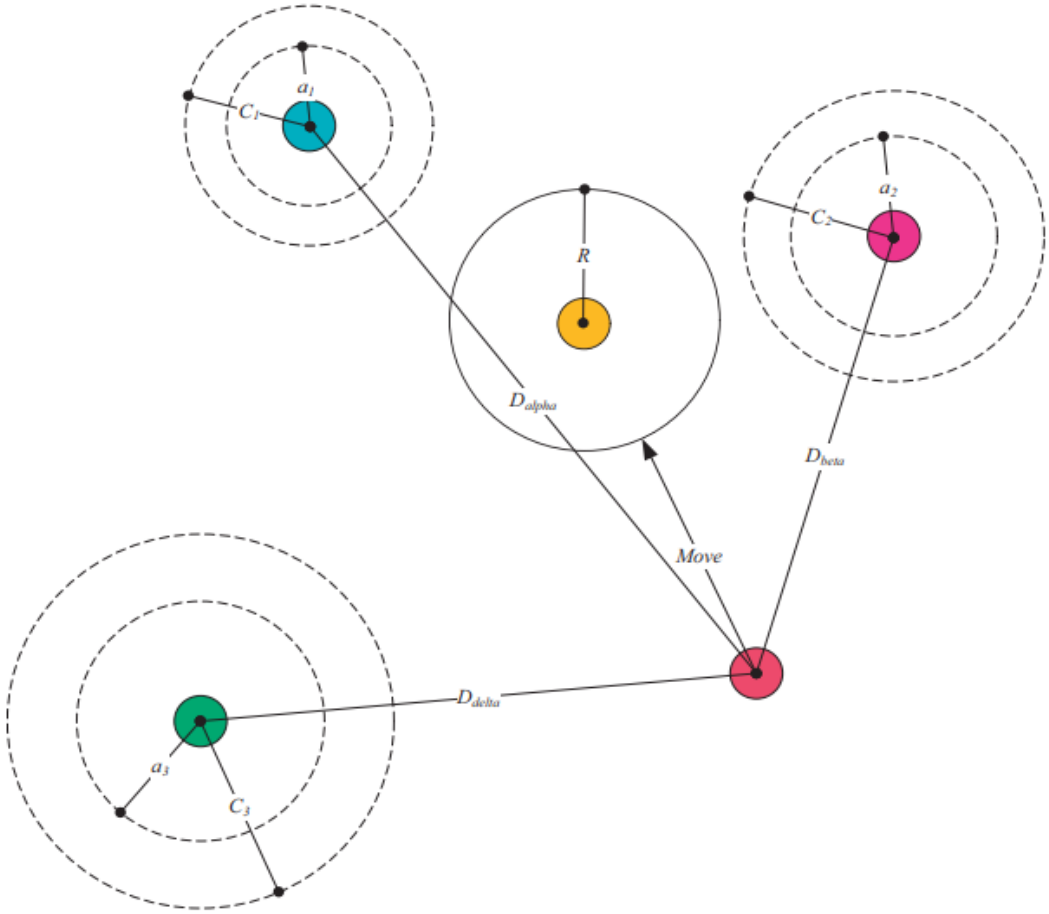

Figure 5.3: Position updating in GWO[4].

Fig. 4(a) illustrates that when $|\vec{A}| < 1$, the wolves are compelled to attack towards the prey. With the operators introduced thus far, the GWO algorithm allows its search agents to update their positions based on the alpha, beta, and delta, and to attack the prey. However, the GWO algorithm may still face stagnation in local solutions with these operators alone. While the proposed encircling mechanism provides some exploration, additional operators are necessary to enhance the exploration capabilities of the GWO.

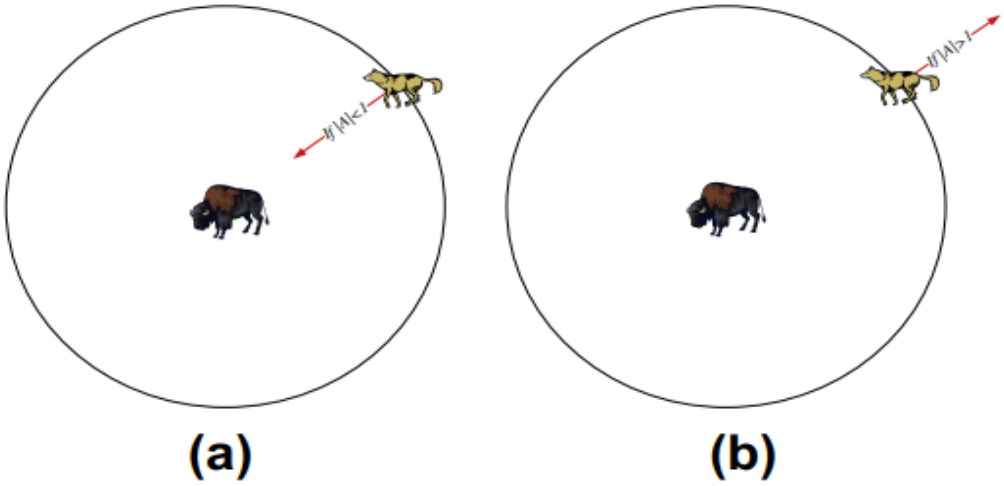

Figure 5.4: Position updating in GWO[4].

5.1.2.5 Search for prey (Exploration)

Grey wolves primarily search based on the positions of the alpha, beta, and delta. They diverge from each other to locate prey and converge to attack it. To mathematically model this divergence, the authors proposed to use $\vec{A}$ with random values greater than 1 or less than -1, which compels the search agents to move away from the prey. This approach emphasizes exploration and enables the GWO algorithm to perform a global search. Fig. 4(b) demonstrates that when $|\vec{A}| > 1$, the grey wolves are forced to diverge from the prey, increasing the chances of finding a more optimal solution.

Another component of the GWO algorithm that enhances exploration is the $\vec{C}$ vector. As seen in Eq. (5.15), the $\vec{C}$ vector includes random values within the interval [0, 2]. This vector introduces random weights to the prey, thereby stochastically emphasizing ($\vec{C} > 1$) or deemphasizing ($\vec{C} < 1$) the influence of the prey in determining the distance in Eq. (5.12). This assists the GWO algorithm in exhibiting more random behavior throughout the optimization process, favoring exploration and helping to avoid local optima. Notably, unlike $\vec{A}$, the $\vec{C}$ vector is not linearly decreased. We intentionally maintain $\vec{C}$ to provide random values at all times to emphasize exploration, not only in the initial iterations but also in the final ones. This component is particularly useful for overcoming local optima stagnation, especially during the later stages of the optimization process.

The $\vec{C}$ vector can also be considered analogous to obstacles that wolves encounter while approaching prey in nature. Typically, obstacles in nature impede wolves from quickly and easily reaching their prey. This is precisely the role of the $\vec{C}$ vector. Depending on a wolf's position, the $\vec{C}$ vector can randomly assign a weight to the prey, making it either harder and farther to reach or easier and closer to reach for the wolves.

To sum up, the search process in the GWO algorithm begins by creating a random population of grey wolves (candidate solutions). Throughout the iterations, the alpha, beta, and delta wolves estimate the likely position of the prey, and each candidate solution updates its distance from the prey accordingly. The parameter $\vec{a}$ decreases from 2 to 0, emphasizing exploration initially and exploitation later on. Candidate solutions diverge from the prey when $|\vec{A}| > 1$ and converge towards the prey when $|\vec{A}| < 1$. The GWO algorithm concludes when a predefined end criterion is met.

5.1.2.6 Remark of the authors

«There are possibilities to integrate mutation and other evolutionary operators to mimic the whole life cycle of grey wolves, However, we have kept the GWO algorithm as simple as possible with the fewest operators to be adjusted.»

So we, decided in this work, to add selection, mutation, crossover and other evolutionary operators to enhance and produce better results.

5.2 Whale Optimization Algorithm

Same as with the grey wolf Algorithm, the inspiration of the proposed method is first discussed. Then, the mathematical model is provided.

5.2.1 Inspiration

Whales are fancy creatures, recognized as the largest mammals on Earth. A fully grown whale can reach lengths of up to 30 meters and weigh as much as 180 tons. Among the seven primary species of these majestic mammals are the killer whale, Minke whale, Sei whale, humpback whale, right whale, finback whale, and blue whale. Predominantly seen as predators, whales have a unique trait: they never fully sleep, as they must regularly surface to breathe. Remarkably, only half of their brain rests at a time. What's fascinating about whales is their widely acknowledged intelligence and emotional depth.

Whales possess specialized cells within certain regions of their brains, akin to those found in humans, known as spindle cells. These cells play a crucial role in judgment, emotions, and social behaviors, setting humans apart from other creatures. Whales have double the number of these cells compared to adult humans, contributing significantly to their intelligence. Studies have demonstrated that whales can engage in complex cognitive processes, learning, judgment, communication, and even emotional responses, albeit at a lower level than humans. Notably, certain whale species, particularly killer whales, have been observed developing distinct dialects.

Another intriguing aspect of whale behavior is their social structure. While they can be solitary, they are often observed in groups, with some species, like killer whales, forming lifelong family units. Among the largest baleen whales is the humpback whale (Megaptera novaeangliae), nearly the size of a school bus when fully grown. These whales primarily feed on krill and small fish.

What makes humpback whales particularly fascinating is their unique hunting technique known as bubble-net feeding. This foraging behavior involves creating a circular or '9'-shaped pattern

of bubbles to corral prey close to the surface. Prior to 2011, this behavior was primarily studied through surface observations.

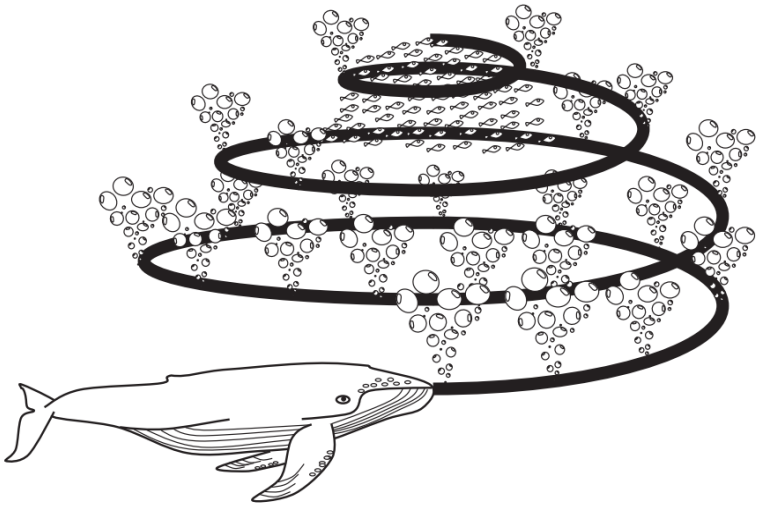

Figure 5.5: Bubble-net feeding behavior of humpback whales[5].

In the upward-spiral maneuver, humpback whales dive around 12 meters down, create a spiral of bubbles around the prey, and then ascend towards the surface. The double-loop maneuver consists of three stages: coral loop, lobtail, and capture loop. It is worth mentioning here that bubble-net feeding is a unique behavior that can only be observed in humpback whales. This behavior is shown in Fig. 5.

In this Algorithm, the spiral bubble-net feeding maneuver is mathematically modeled in order to perform optimization.

5.2.2 Mathematical Model

5.2.2.1 Encircling prey

Humpback whales can recognize the location of prey and encircle them. Since the position of the optimal design in the search space is not known a priori, the WOA algorithm assumes that the current best candidate solution is the target prey or is close to the optimum. After the best search agent is defined, the other search agents will hence try to update their positions towards the best search agent. This behavior is represented by the following equations: [5]

$$\vec{D} = |\vec{C} \cdot \vec{X}^* - \vec{X}(t)| \tag{5.12}$$

$$\vec{X}(t+1) = \vec{X}^* - \vec{A} \cdot \vec{D} \tag{5.13}$$

Where: $\vec{X}^*$ is the position vector of the best solution obtained so far, $\vec{X}(t)$ is the position vector, $\vec{A}$ and $\vec{C}$ are coefficient vectors, t denotes the current iteration, $|\ |$ is the absolute value, and $\cdot$ is an element-by-element multiplication. It is worth mentioning here that $\vec{X}^*$ should be updated in each iteration if there is a better solution.

These equations help simulate the humpback whales' encircling mechanism in the optimization process.

The vectors $\vec{A}$ and $\vec{C}$ are calculated as follows: [5]

$$\vec{A} = 2\vec{a} \cdot \vec{r}_1 - \vec{a} \tag{5.14}$$

$$\vec{C} = 2 \cdot \vec{r}_2 \tag{5.15}$$

Where: $\vec{a}$ is a vector that linearly decreases from 2 to 0 over the course of iterations (in both exploration and exploitation phases), $\vec{r}_1$ and $\vec{r}_2$ are random vectors in the range [0, 1].

These calculations allow the WOA algorithm to simulate the adaptive and exploratory behavior of humpback whales during the optimization process.

Figure. 6(a) provides insight into the principle described by Eq. (5.13) within a 2D context. The coordinates (X,Y) of a search agent can be adjusted relative to the position of the current optimal solution (X^*,Y^*). Manipulating the $\vec{A}$ and $\vec{C}$ vectors enables exploration of various locations surrounding the top-performing agent. Extending this concept to 3D space, Figure. 6(b) visualizes potential adjustments in the agent's position. Introducing the random vector $\vec{r}$ facilitates navigation across the search space, encompassing regions between the depicted key-points. Hence, Eq. (5.13) empowers each search agent to refine its position near the current best solution, and simulates encircling prey for optimization.

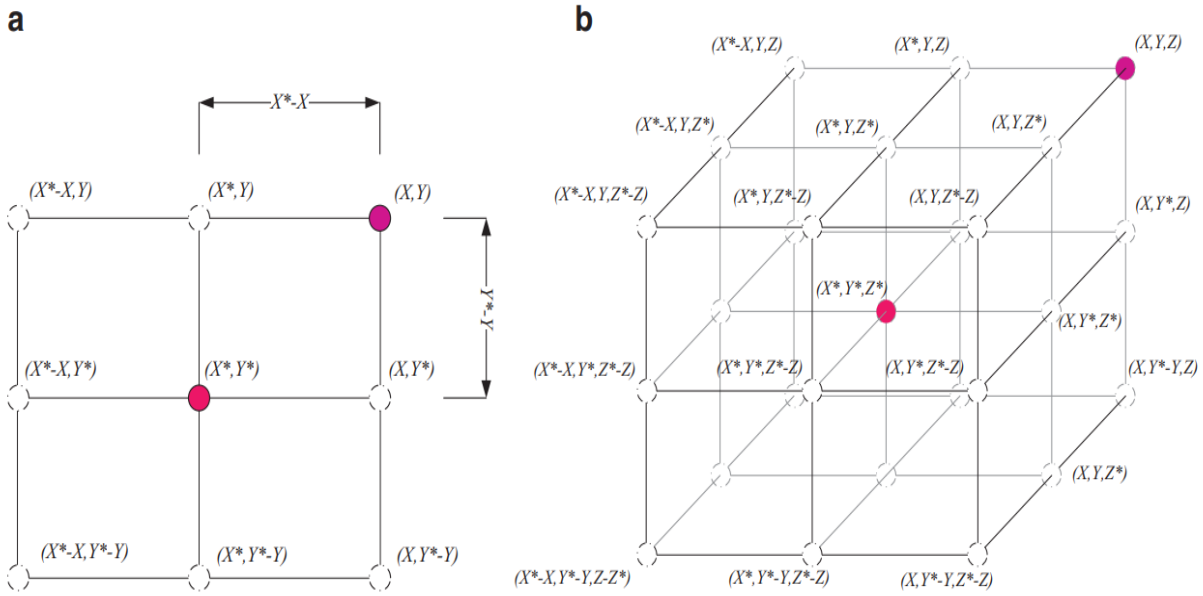

Figure 5.6: 2D and 3D position vectors and their possible next locations.[5]

The concept can be extrapolated to a search space comprising n dimensions, where the search agents navigate within hyper-cubes centered around the most optimal solution achieved thus far. In alignment with the earlier discussion, humpback whales employ a similar tactic when hunting prey through the bubble-net strategy. This strategy can be expressed mathematically as follows:

5.2.2.2 Bubble-net attacking method (Exploitation)

To mathematically represent the bubble-net behavior of humpback whales, two distinct approaches have been devised as outlined below:

1. Shrinking encircling mechanism:

This behavior is accomplished by reducing the magnitude of $\vec{a}$ in Equation (5.14). It's important to note that this reduction also narrows the range of fluctuation for $\vec{A}$, effectively limiting it within a smaller interval. Specifically, $\vec{A}$ is constrained to random values within the range $[a, a]$, with the value of a diminishing gradually from 2 to 0 throughout the iterations. By assigning random values to $\vec{A}$ within the interval $[1, 1]$, the updated position of a search agent can then be determined anywhere between its original position and the position of the current top-performing agent.

2. Spiral updating position:

As depicted in Figure 7, this method begins by computing the distance between the whale positioned at (X,Y) and the prey situated at (X^*,Y^*). Subsequently, a spiral equation is formulated to emulate the helix-like motion observed in humpback whales. This equation is crafted based on the relative positions of the whale and the prey, thereby capturing the characteristic spiral movement exhibited during hunting. [5]

$$\vec{X}(t+1) = \vec{D'} \cdot e^{bl} \cdot cos(2\pi l) + \vec{X}^*(t) \tag{5.16}$$

Where $\vec{D'} = |\vec{X}^* - \vec{X}|$ and indicates the distance of the ith whale to the prey (best solution obtained so far), b is a constant for defining the shape of the logarithmic spiral, l is a random number in [1,1], and . is an element-by-element multiplication.

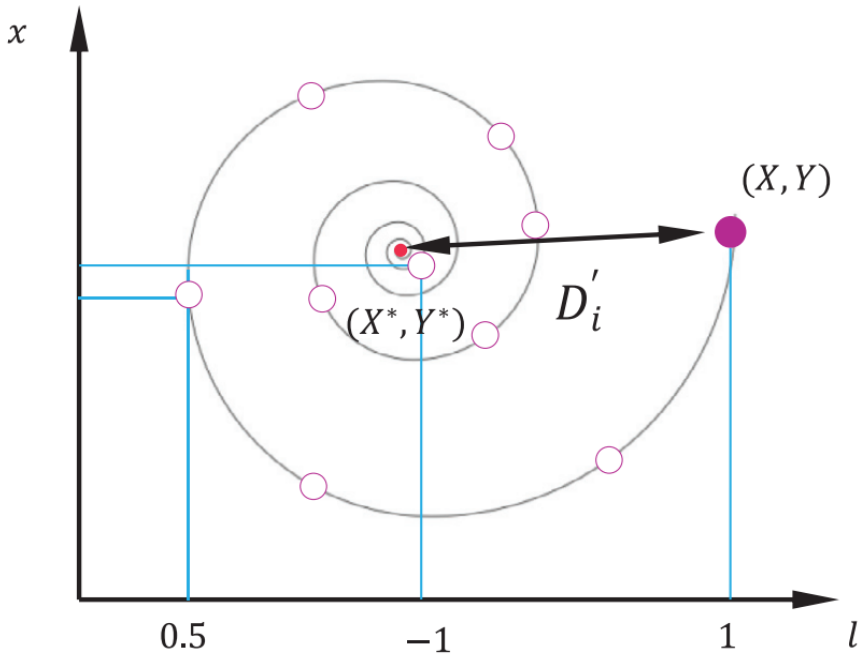

Figure 5.7: Spiral updating position[5].

Note that humpback whales swim around the prey within a shrinking circle while simultaneously following a spiral-shaped path. To model this behavior, we assume a 50% probability for choosing either the shrinking encircling mechanism or the spiral model to update the position of the whales during optimization. The mathematical model is as follows: [5]

$$\vec{X}(t+1) = \begin{cases} \vec{X}^*(t) - \vec{A} \cdot \vec{D}, & \text{if } p < 0.5 \\ \vec{D} \cdot e^{bl} \cdot \cos(2\pi l) + \vec{X}^*(t) & \text{if } p \geq 0.5 \end{cases}$$

Where p is a random number in [0,1].

In addition to the bubble-net method, the humpback whales search for prey randomly. The mathematical model of the search is as follows.

5.2.2.3 Search for prey (Exploration Phase)

The same approach, based on the variation of the $\vec{A}$ vector, can be used for prey search (exploration). Humpback whales search randomly in relation to each other's positions. Consequently, we use $\vec{A}$ with random values greater than 1 or less than -1 to encourage the search agent to move far from a reference whale. Unlike the exploitation phase, the exploration phase updates the position of a search agent based on a randomly selected search agent instead of the best one found so far. This mechanism, combined with the
$|\vec{A}| > 1$, emphasizes exploration and enables the WOA algorithm to conduct a global search. The mathematical model is as follows: [5]

$$\vec{D} = |\vec{C} \cdot \vec{X}_{rand} - \vec{X}(t)| \tag{7}$$

$$\vec{X}(t+1) = \vec{X}_{rand} - \vec{A} \cdot \vec{D} \tag{8}$$

Where $\vec{X}_{rand}$ represents a random position vector (a random whale) chosen from the current population.

To sum up, the WOA algorithm starts with a set of random solutions. At each iteration, search agents update their positions based on either a randomly chosen search agent or the best solution found so far. The parameter a is decreased from 2 to 0 to balance exploration and exploitation. When $|\vec{A}| > 1$, a random search agent is chosen, whereas when $|\vec{A}| < 1$, the best solution is selected for updating the positions of the search agents. Depending on the value of p, the WOA algorithm can switch between spiral and circular movements. The algorithm terminates when a predefined termination criterion is met.

5.2.2.4 Remark of the authors

«Although mutation and other evolutionary operations might have been included in the WOA formulation to fully reproduce the behavior of humpback whales, we decided to minimize the amount of heuristics and the number of internal parameters, thus implementing a very basic version of the WOA algorithm.»

So we, decided in this work, to add selection, mutation, crossover and other evolutionary operators to enhance and produce better results.

5.3 Puma Optimizer

Same as with the grey wolf Algorithm, and the whale optimization Algorithm, the inspiration of the proposed method is first discussed. Then, the mathematical model is provided.

5.3.1 Inspiration

The puma, also known as the cougar or mountain lion, is a large cat species from the small cat subfamily, native to the American continent. Its habitat ranges from the Andes Mountains in South America to the Yukon in Canada. After the jaguar, the mountain lion is the largest feline in the Americas. Pumas are highly adaptable and can live in various habitats, feeding on a diverse array of prey. Though primarily nocturnal, they can sometimes be seen during the day.

As an ambush predator, the puma primarily preys on ungulates, including deer, but also consumes smaller animals like rodents and insects and occasionally targets domestic livestock. It favors dense scrub and rocky areas for ambushing but also inhabits open plains. Like other cats, pumas are territorial, with vast territories and low population densities. They often compete with other predators such as jaguars, wolves, American black bears, and American alligators in certain areas.

The arrival of Europeans in America led to extensive hunting and the development of human settlements, endangering mountain lions in many regions. For example, the eastern puma subspecies became extinct in eastern North America, except for a small population in Florida. However, in recent years, the puma's range has expanded from west to east, even reappearing in Connecticut on the Atlantic coast.

Pumas are slender and agile, ranking as the fourth-largest feline overall. Adult pumas stand 60 to 90 cm tall at the shoulder. Male pumas measure about 2.4 meters from nose to tail, while females are about 1.5 meters long, with the overall length ranging from 1.50 to 2 meters. Their tails alone measure between 63 and 95 cm. Male pumas weigh between 55 and 100 kg, averaging 64 kg, while females weigh between 29 and 64 kg, averaging around 55 kg. Pumas are generally smaller in tropical regions and larger in polar regions, with the largest recorded puma, weighing 105.2 kg, hunted in 1901. While pumas can vary in size, they are typically smaller and less muscular than jaguars. Unlike big cats, pumas cannot roar due to the lack of a specialized larynx and laminar bone.

Pumas are capable of sprinting in pursuit of prey, but they typically ambush. They stalk through bushes and trees, using powerful leaps to catch their prey, which they suffocate with their fangs. Pumas drag their prey to a preferred spot, cover it with bushes, and return to feed over several days. Solitary by nature, pumas usually only form groups of mothers and cubs, with mature individuals rarely interacting. However, within a dominant male's territory, cougars may share prey and form small communities.

Research indicates that the smallest territory for a puma is about 25 square kilometers (9.7 square miles), while the largest can reach up to 1,300 square kilometers (500 square miles) for males. Male territories include overlapping female territories, and males continually mark their boundaries. The size of a puma's territory depends on factors such as population density, vegetation, and prey availability. Large male territories range from 150 to 1,000 square kilometers (58 to 386 square miles), with female territories being roughly half that size.

5.3.2 Mathematical Model

Most meta-heuristic optimizers perform optimization by generating a random solution and then modifying exploration factors using mechanisms specific to each algorithm. The PO optimizer

algorithm introduces a novel and intentional mechanism for shifting between exploration and exploitation phases. Additionally, the PO algorithm employs two distinct mechanisms for optimization operations in both the exploitation and exploration phases.

In the PO algorithm, the best solution is metaphorically considered the male puma, while the entire optimization space represents the puma's territory. Other solutions (X_i) are regarded as female pumas. Each iteration of the algorithm involves all solutions entering either the exploitation or exploration phase through the phase change mechanism, with the selection of these phases being both purposeful and intelligent.

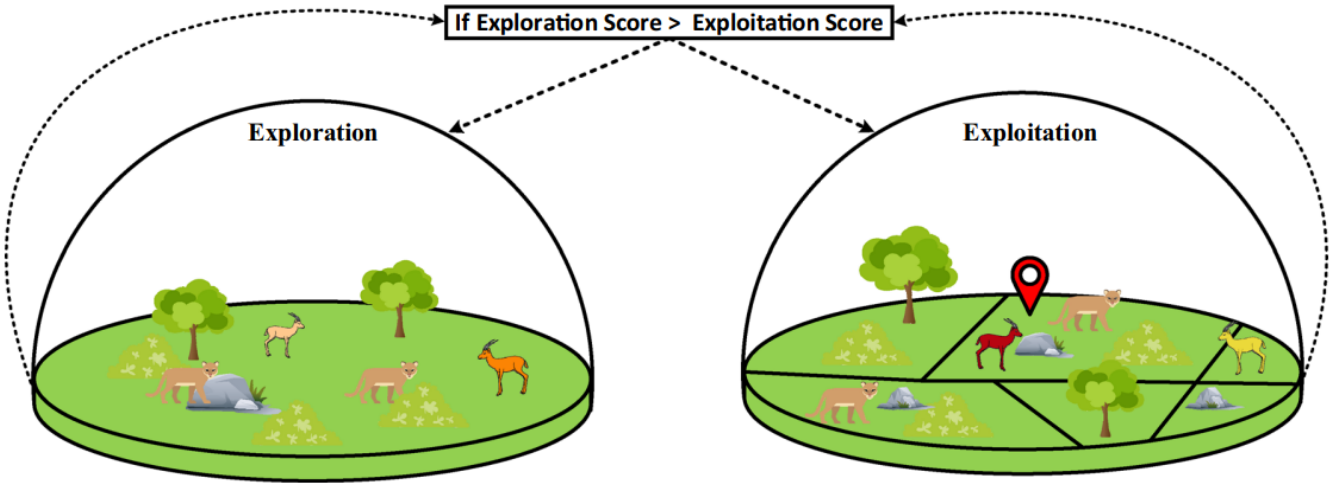

Figure 5.8: PO optimization procedure[6].

In each stage of exploration, various optimization methods have been employed. In every stage, two distinct mechanisms, inspired by the natural behavior of pumas, have been utilized.

Following this, the mathematical formulations and expressions are provided to explain the optimization processes carried out by the Puma Optimizer:

5.3.2.1 Puma Intelligence (phase change mechanism)

Pumas are highly intelligent animals with excellent memories. When hunting, they often revisit locations where they have had previous success, relying on their past experiences. These hunting trips may take them to areas where they have previously hunted and hidden prey or to new places they have not yet explored. In this algorithm, the exploitation phase corresponds to pumas returning to previously promising locations, while the exploration phase involves traveling to new areas. The transition between these phases draws inspiration from the intelligence and memory of pumas. A novel and sophisticated mechanism, considered an advanced hyper-heuristic algorithm, has been fully explained below.

The phase change mechanism in the PO algorithm is a type of heuristic selection algorithm that uses two components: diversity and intensification, to perform reward and penalty operations for scoring. This phase change section is inspired by the intelligence of pumas and has two approaches. The first approach applies to pumas with limited experience and energy, who simultaneously explore unfamiliar territory and ambush in promising areas. This scenario is discussed in the context of first-generation pumas, who lack sufficient experience.

1. Unexperienced phase:

In its early life, a puma lacks experience and often engages in exploration due to unfamiliarity with its environment and uncertainty about the locations of prey within its territory. Simultaneously, it searches for prey in promising areas. In the Puma algorithm, during the first three iterations, both exploration and exploitation operations are conducted simultaneously until the initialization phase is complete. In this initial phase, as both exploitation and exploration phases are selected in each iteration, only two functions (f1 and f2) are used, which are calculated using Eqs. (5.17-5.20). [6]

$$f_{1Explore} = PF_1 \cdot \left(\frac{Seq_{1CostExplore}}{Seq_{1Time}} \right) \tag{5.17}$$

$$f_{1Exploit} = PF_1 \cdot \left(\frac{Seq_{1CostExploit}}{Seq_{1Time}} \right) \tag{5.18}$$

$$f_{2Explore} = PF_2 \cdot \left(\frac{Seq_{1CostExplore} + Seq_{2CostExplore} + Seq_{3CostExplore}}{Seq_{1Time} + Seq_{2Time} + Seq_{3Time}} \right) \tag{5.19}$$

$$f_{2Exploit} = PF_2 \cdot \left(\frac{Seq_{1CostExploit} + Seq_{2CostExploit} + Seq_{3CostExploit}}{Seq_{1Time} + Seq_{2Time} + Seq_{3Time}} \right) \tag{5.20}$$

The values of Seq_{Cost}, associated with the exploitation and exploration phases, are determined using Eqs. (5.17-5.20). Seq_{Time}, another variable, is assigned a constant value of 1. $PF1$ and $PF2$ are fixed parameters that must be set prior to the optimization process. These parameters are used to prioritize the $f1$ and $f2$ functions, respectively. [6]

$$Seq^1_{CostExplore} = \left| Cost^{Initial}_{Best} - Cost^1_{Explore} \right| \tag{5.21}$$

$$Seq^2_{CostExplore} = \left| Cost^2_{Explore} - Cost^1_{Explore} \right| \tag{5.22}$$

$$Seq^3_{CostExplore} = \left| Cost^3_{Explore} - Cost^2_{Explore} \right| \tag{5.23}$$

$$Seq^1_{CostExploit} = \left| Cost^{Initial}_{Best} - Cost^1_{Exploit} \right| \tag{5.24}$$

$$Seq^2_{CostExploit} = \left| Cost^2_{Exploit} - Cost^1_{Exploit} \right| \tag{5.25}$$

$$Seq^3_{CostExploit} = \left| Cost^3_{Exploit} - Cost^2_{Exploit} \right| \tag{5.26}$$

In Eqs. (5.21) and (5.24), $Cost^{Initial}_{Best}$ represents the cost of the optimal solution produced during the initialization phase. Additionally, six variables, namely $Cost^1_{Explore}$, $Cost^2_{Explore}$, $Cost^3_{Explore}$, $Cost^1_{Exploit}$, $Cost^2_{Exploit}$, and $Cost^3_{Exploit}$, represent the cost of the best solution obtained from each of the exploration and exploitation phases across repetitions 1, 2, and 3. After computing

the functions f1 and f2 at the end of the third iteration, only one phase—either exploration or exploitation—will be selected from this point onward. This selection is based on the positive experiences of other Pumas, and the decision between the two phases is made using the values obtained from Eqs. (5.27) and (5.28). [6]

$$Score_{Explore} = (PF_1 \cdot f1_{Explore}) + (PF_2 \cdot f2_{Explore}) \tag{5.27}$$

$$Score_{Exploit} = (PF_1 \cdot f1_{Exploit}) + (PF_2 \cdot f2_{Exploit}) \tag{5.28}$$

After calculating $Score_{Explore}$ and $Score_{Exploit}$ using Eqs. (5.27) and (5.28) to determine which phase to enter, the algorithm proceeds as follows: if $Score_{Exploit} \geq Score_{Explore}$, it enters the exploitation phase; otherwise, it enters the exploration phase. A critical point is that by the end of the third iteration, each phase independently produces solutions, resulting in more solutions than the total population. To address this, the total cost of the solutions produced in both phases is calculated at the end of the third iteration. Only the best solutions, equal to the total population size, are selected to replace the current solutions.

2. Experienced phase:

After three generations, the Pumas have gained sufficient experience to decide whether to change phases. In subsequent iterations, they select only one phase for the optimization operation. Three different functions, f1, f2, and f3, are used for scoring in this phase. The first function, f1, emphasizes the escalation component and gives priority to the phase (either exploration or exploitation) that has performed better. This function places more emphasis on the exploration phase and is calculated using Eqs. (5.29) and (5.3.2.1). [6]

$$f_{1t}^{exploit} = PF_1 \cdot \left| \frac{Cost_{old}^{exploit} - Cost_{new}^{exploit}}{T_t^{exploit}} \right| \tag{5.29}$$

$$f_{1t}^{explore} = PF_1 \cdot \left| \frac{Cost_{old}^{explore} - Cost_{new}^{explore}}{T_t^{explore}} \right| \tag{5.30}$$

In Eqs. (5.29) and (), $f_{1t}^{exploit}$ and $f_{1t}^{explore}$ represent the value of the first function based on the exploitation phase or the exploration phase, where t denotes the current iteration number. $Cost_{old}^{exploit}$ and $Cost_{old}^{explore}$ are the costs of the best solution before improvement in the current phase. Conversely, $Cost_{new}^{exploit}$ and $Cost_{new}^{explore}$ are the costs of the best solution obtained after improving the current selection. $T_t^{explore}$ and $T_t^{exploit}$ are the numbers of iterations that were unselected from the previous selection to the current selection. $PF1$ is a user-adjustable parameter that must be set to a value between 0 and 1 before the optimization operation. This parameter determines the importance of the first function, with its priority increasing as the function's value increases and decreasing as its priority decreases.

The second function emphasizes the resonance component and gives priority to the phase that performs better than the other. Good performances are checked and measured sequentially, allowing this function to aid in the selection of the exploitation phase. The second function is calculated using Eqs. (5.31) and (5.32). [6]

$$f_{2t}^{exploit} = PF_2 \cdot \left| \frac{\left(Cost_{Old,1}^{exploit} - Cost_{New,1}^{exploit}\right) + \left(Cost_{Old,2}^{exploit} - Cost_{New,2}^{exploit}\right) + \left(Cost_{Old,3}^{exploit} - Cost_{New,3}^{exploit}\right)}{T_{t,1}^{exploit} + T_{t,2}^{exploit} + T_{t,3}^{exploit}} \right| \tag{5.31}$$

$$f_{2t}^{explor} = PF_2 \cdot \left| \frac{\left(Cost_{Old,1}^{explor} - Cost_{New,1}^{explor}\right) + \left(Cost_{Old,2}^{explor} - Cost_{New,2}^{explor}\right) + \left(Cost_{Old,3}^{explor} - Cost_{New,3}^{explor}\right)}{T_{t,1}^{explor} + T_{t,2}^{explor} + T_{t,3}^{explor}} \right| \tag{5.32}$$

In Eqs. (5.31) and (5.32), $f_{2t}^{exploit}$ and $f_{2t}^{explore}$ represent the second function related to the exploitation and exploration phases, respectively, with t indicating the current iteration number. $Cost_{Old,1}^{explore}$ and $Cost_{Old,1}^{exploit}$ are the costs of the best solution before improvement in the current selection for the exploration and exploitation phases. $Cost_{Old,2}^{explore}$ and $Cost_{Old,2}^{exploit}$ are the costs of the best solution before improvement in the previous selection. $Cost_{Old,3}^{explore}$ and $Cost_{Old,3}^{exploit}$ are the costs of the best solution before improvement in the two selections prior. For costs after improvement: $Cost_{New,1}^{explore}$ and $Cost_{New,1}^{exploit}$ are the costs of the best solution obtained after improvement in the current selection. $Cost_{New,2}^{explore}$ and $Cost_{New,2}^{exploit}$ are the costs of the best solution obtained after improvement in the previous selection. $Cost_{New,3}^{explore}$ and $Cost_{New,3}^{exploit}$ are the costs of the best solution obtained after improvement in the two selections prior.

For the number of unselected iterations: $T_{t,1}^{explore}$ and $T_{t,1}^{exploit}$ represent the numbers of unselected iterations from the previous selection to the current selection for exploration and exploitation. $T_{t,2}^{explore}$ and $T_{t,2}^{exploit}$ represent the numbers of unselected iterations from two selections prior to the previous selection. $T_{t,3}^{explore}$ and $T_{t,3}^{exploit}$ represent the numbers of unselected iterations from three selections prior to two selections prior.

PF_2 is a parameter that must be set to a value between 0 and 1 before the optimization operation, determining the extent to which the second function is emphasized. It is prioritized as the value of this function increases and decreases as its priority decreases.

The third function in the selection mechanism emphasizes the diversity component. This function ensures that the phase not frequently selected in many repetitions still has a chance to be chosen, preventing the algorithm from collapsing into a local optimum trap. This function is represented in Eqs. (5.33) and (5.34). [6]

$$f_{3t}^{exploit} = \begin{cases} 0 & if selected, \\ f_{3t}^{exploit} + PF_3 & otherwise. \end{cases} \tag{5.33}$$

$$f_{3t}^{explore} = \begin{cases} 0 & if selected, \\ f_{3t}^{explore} + PF_3 & otherwise. \end{cases} \tag{5.34}$$

In Eqs. (5.33) and (5.34), $f_{3t}^{exploit}$ and $f_{3t}^{explore}$ represent the third function related to the exploitation and exploration stages, respectively, t denotes the current iteration number. According to Eq. (5.19), the value of the third function for each phase—exploitation and exploration-will increase by the parameter PF_3 in each iteration if the phase is not selected; otherwise, it will be set to zero. PF_3 is a user-adjustable parameter that must be set to a value between 0 and 1 before the optimization operation. The closer the PF_3 parameter is to 1, the higher the chance

that a phase with a low score will be selected, and the chance of selection decreases as the value of PF_3 decreases. Using Eqs. (5.35) and (5.36), the cost of the phase change function is calculated.
[6]

$$F_t^{exploit} = (\alpha_t^{exploit}.(f_{1_t^{exploit}})) + (\alpha_t^{exploit}.(f_{2_t^{exploit}})) + (\delta_t^{exploit}.(lc.f_{3_t^{exploit}})) \tag{5.35}$$

$$F_t^{explore} = (\alpha_t^{explore}.(f_{1_t^{explore}})) + (\alpha_t^{explore}.(f_{2_t^{explore}})) + (\delta_t^{explore}.(lc.f_{3_t^{explore}})) \tag{5.36}$$

$$lc = \{\{|Cost_{old} - Cost_{New}|\}^{exploitation}, \{|Cost_{old} - Cost_{New}|\}^{exploration}\}, 0 \notin lc \tag{5.37}$$

$$\alpha_t^{explore,exploit} = \begin{cases} \text{if } F^{exploit} > F^{explore}, \alpha^{exploit} = 0.99, \alpha^{explore} = \alpha^{explore} - \lceil 0.01, 0.01 \rfloor \\ otherwise, \alpha^{explore} = 0.99, \alpha^{exploit} = \alpha^{exploit} - \lceil 0.01, 0.01 \rfloor \end{cases} \tag{5.38}$$

$$\delta_t^{exploit} = 1 - \alpha_t exploit \tag{5.39}$$

$$\delta_t^{explore} = 1 - \alpha_t explore \tag{5.40}$$

Using Eqs. (5.35) and (5.36), the final cost of each exploitation and exploration phase is calculated. Parameters a and d for both the exploration and exploitation phases are variable during the search operation based on the results obtained from each phase. If diversity is prioritized, the diversity component is emphasized; otherwise, its priority is reduced. When the value of parameter a is close to 1, the resonance component is prioritized.

According to Eq. (5.38), if the cost of the exploration phase function is larger than that of the exploitation phase, the value of parameter a for the exploitation function will be penalized linearly by a value of 0.01. Conversely, the value of parameter a for the exploitation phase will be allowed to reach a maximum value close to one. If the exploitation phase function costs more than the exploration phase, the procedure will be reversed.

The term lc refers to a set of calculation cost differences derived from the improvements achieved in the exploitation and exploration phases, encompassing a range of non-zero values.

5.3.2.2 Exploration

Pumas often travel extensive distances within their territory to locate and hunt for food. This search may involve revisiting areas where they previously had successful hunts or exploring new parts of their territory where they haven't hunted before. In our exploration approach, we draw inspiration from these puma behaviors. During this phase, pumas perform a random search within their territory to find food, or they may approach other pumas to utilize their prey. Thus, the puma either randomly leaps into the search space or forages in the space between pumas. Initially, the entire population is arranged in ascending order, and then the puma enhances its solutions during the exploration phase using Eq. (5.40). [6]

$$\text{If } rand_1 > 0.5, Z_{i,G} = R_{Dim} * (Ub - Lb) + Lb \tag{5.41}$$

$$\begin{aligned}\text{Otherwise, } Z_{i,G} = X_{a,G} + G.(X_{a,G} - X_{b,G}) + G.(((X_{a,G} - X_{b,G}) - (X_{c,G} - X_{d,G})) \\ + ((X_{c,G} - X_{d,G}) - (X_{e,G} - X_{f,G})))\end{aligned} \tag{5.42}$$

In Eq. (5.41), Ub and Lb represent the upper and lower bounds of the problem, respectively. R_{Dim} consists of randomly generated numbers between 0 and 1, matching the dimensions of the problem. Additionally, $rand1$ is a randomly generated number between 0 and 1. Solutions $X_a^G, X_b^G, X_c^G, X_d^G, X_e^G$, and X_f^G are randomly selected from the entire population. G is determined using Eq. (5.42), where $rand2$ is another randomly generated number from a uniform distribution between 0 and 1. Based on Eq. (5.41), one of two equations is chosen to generate a new solution depending on the current condition. This new solution is then used to improve the current solution. [6]

$$X_{new} = \begin{cases} Z_{i,G}, \text{if } j = j_{rand} \ or \ rand_3 \leq U \\ X_{a,G}, \ otherwise \end{cases} \tag{5.43}$$

$$NC = 1 - U \tag{5.44}$$

$$p = \frac{NC}{Npop} \tag{5.45}$$

$$if \ CostX_{New} < CostX_i, U = U + p \tag{5.46}$$

In Eq. (5.43), Z_i^G is a solution generated using Eq. (5.41). j_{rand} is a randomly generated integer within the range of the problem's dimensions, and $rand3$ is a randomly produced number from a uniform distribution between 0 and 1. NC is calculated using Eq. (5.44). U is a parameter set before the optimization process begins, with a value between 0 and 1. During each iteration, according to the condition in Eq. (5.46), the number of dimensions replaced by new solutions increases, following the guidelines in Eqs. (5.44–5.46). In Eq. (5.45), N_{pop} represents the total number of pumas.

The solution improvement process adheres to the condition in Eq. (5.46), updating only the solution dimensions that meet this condition. This method helps avoid local optima, ensuring the solutions maintain good diversity. Additionally, during the exploration stage, the search agents are sorted in ascending order based on their costs at the start of each iteration, with high-quality solutions prioritized. According to Eqs. (5.44–5.46), initially, quality solutions undergo minimal changes due to the small value of the U parameter. However, as this parameter increases, higher-cost solutions undergo more significant changes. This approach aims to explore less optimal areas of the problem space to discover better optimal points.

A critical point is that if the generated solutions are not better than the current ones, Eq. (5.46) will not be applied, as there is no need for redundant exploration if improvement is achieved. High-quality solutions undergo minimal changes, primarily to avoid local optima. Finally, the newly generated solutions replace the current ones using Eq. (5.47). [6]

$$X_{a,G} = X_{new}, if \ X_{i,new} < X_{a,G} \tag{5.47}$$

According to Eq. (5.47), the newly generated solution replaces the current solution if it has a better cost than the current solution.

5.3.2.3 Exploitation

In the exploitation stage, the PO algorithm employs two different operators to enhance the solutions, drawing on two distinct behaviors of pumas: ambushing and sprinting. In nature, pumas often ambush their prey from hiding spots such as bushes, trees, or rocks. Occasionally, they also chase their prey. These behaviors are simulated in the algorithm, with ambushing represented by one operator and sprinting by another, the latter being modeled using Eq. (5.48). [6]

$$X_{new} = \begin{cases} if\ rand_4 \geq 0.5, X_{new} = \frac{\frac{mean(Sol_{total})}{Npop}.X_1^r - (-1)^\beta \times X_i}{1+\alpha.rand_5} \\ otherwise,\ if\ rand_6 \geq L, X_{new} = Puma_{male} + (2.rand_\gamma).exp(randn_1).X_2^r - X_i \\ otherwise,\ X_{new} = (2 \times rand_8) \times \frac{F_1.R.X(i)+F2.(1-R).Puma_{male}}{(2.rand_9-1+randn_2)} - Puma_{male} \end{cases} \tag{5.48}$$

Equation (5.48) illustrates the two strategies utilized in the PO algorithm: running and ambushing. In the first mode, during the exploitation phase, these strategies are used to simulate pumas' behavior of fast running and ambushing prey. If $rand5$, a randomly generated number from a uniform distribution between 0 and 1, is greater than 0.5, the fast-running strategy is executed, simulating pumas' rapid pursuit of prey. Otherwise, the ambush strategy is chosen, which involves two different operations. The first operation simulates pumas making short jumps towards other pumas' hunts, while the second operation simulates long jumps towards the best puma's hunt.

In Eq. (5.48), the *mean* function represents the average, Sol_{total} denotes the sum of all solutions, and N_{pop} is the total population involved in the optimization procedure. X_{r1} is a randomly selected solution from the entire population, and b is a binary variable (0 or 1) generated randomly. X_i represents the current solution in the current iteration, and a and L are static parameters that need to be tuned before the optimization process. Additionally, $Puma_{male}$ is the best solution in the population, and $rand4$, $rand5$, $rand6$, $rand7$, $rand8$, and $rand9$ are random numbers generated between 0 and 1. The exp function represents the exponential function. $randn1$ and $randn2$ are randomly generated numbers from a normal distribution with the same dimensions as the problem, and X_{r2} is another randomly selected solution, chosen based on Eq. (5.49). [6]

$$round(1 + (Npop - 1).rand_{10}) \tag{5.49}$$

In Eq. (5.49), each element of X is rounded to the nearest integer. $rand10$ is a randomly generated number between 0 and 1, and N_{pop} represents the total number of pumas in the population.

Finally, R, $F1$, and $F2$ are calculated using Eqs. (5.50), (5.51), and (5.52), respectively.

$$R = 2.rand_{11} - 1 \tag{5.50}$$

$$F_1 = randn_3.exp(2 - Iter.(\frac{2}{MaxIter})) \tag{5.51}$$

In Eq. (5.51), $randn2$ is a randomly generated number with the same dimensions as the problem, following a normal distribution. $Iter$ represents the current iteration number, and $MaxIter$ denotes the total number of iterations for the optimization process. The exp function signifies the exponential function. [6]

$$F_2 = w \times (v)^2.cos((2 \times rand_{12}).w) \tag{5.52}$$

$$w = randn_4 \tag{5.53}$$

$$v = randn_5 \tag{5.54}$$

In Eqs. (5.52–5.54), $randn4$ and $randn5$ are both randomly generated numbers with the same dimensions as the problem, following a normal distribution. The cos function represents the cosine function, and $rand12$ is a randomly generated number between 0 and 1. Finally, at the end of this phase, newly produced solutions are replaced if they have a lower cost than the current solution.

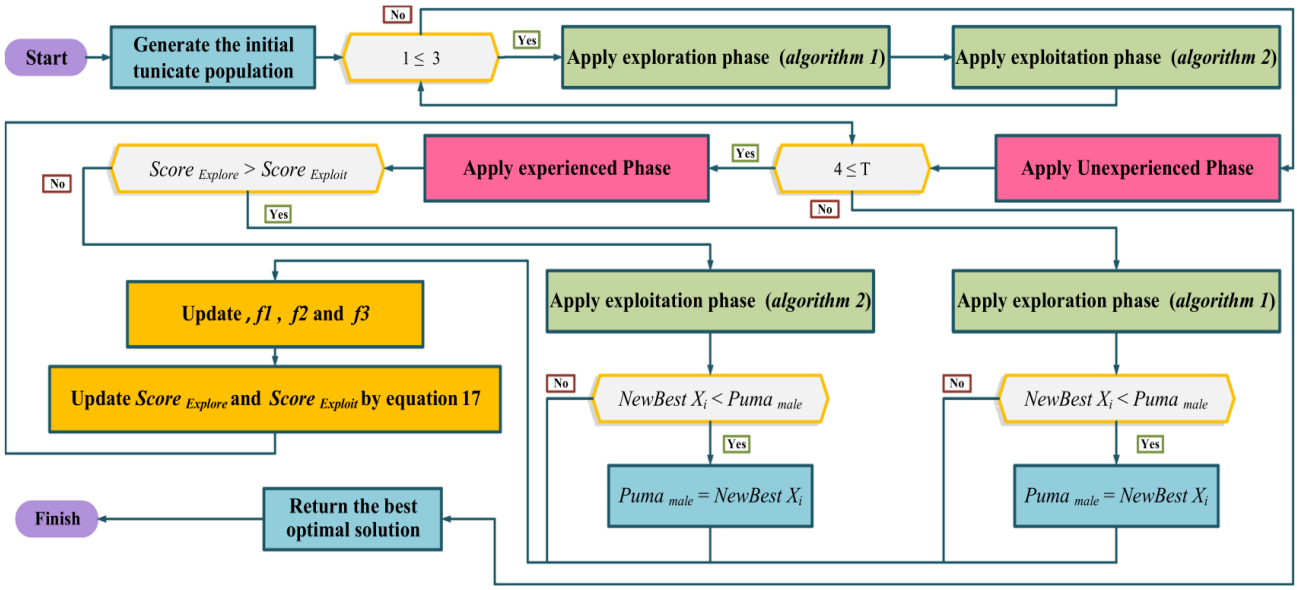

Figure 5.9: PO optimization procedure[6].

5.3.2.4 Remark of the authors

«Even though the PO algorithm can be challenging to implement, it generally maintains low computational complexity, which varies depending on the phase being executed. Each stage has its own computational complexity. Overall, the computational complexity is $O((2T+1)\cdot N\cdot D)$. During the exploration phase, sorting is required, which has a worst-case computational complexity of $O(N^2)$. Therefore, the overall computational complexity is $O((2T+1)\cdot N^2\cdot N\cdot D)$, where T is the maximum number of iterations, D is the dimension of the problem, and N is the total population size.»

Chapter 6

Problem Formulation

Clustering in IoT networks is a widely adopted topology for optimization, as it enables better energy consumption distribution and can support a large number of sensor nodes. When data routing is direct, sensors located farther away deplete their energy more quickly. In multi-hop routing, intermediate nodes must handle additional transmissions by relaying messages from other nodes, which drains their power supplies. Clustering addresses these issues by organizing data collection into clusters, thereby reducing the distance between sensor nodes and the base station, minimizing interference, and enhancing the quality of radio links. This approach decreases the number of retransmissions and lowers energy consumption. Moreover, by frequently rotating the cluster heads, the battery life of sensor nodes can be prolonged and balanced more evenly across all nodes.

6.1 Energy Consumption Assessment

The energy consumption of sensor nodes can be measured using a radio model that depends on the distance between the transmitting and receiving nodes. The value of this distance determines whether the free space model or the multi-path model is used.

The energy consumed by a sensor node when sending a message of length L over a distance d can be formulated as follows 6.1 [54] :

$$E_{TX}(L,d) = \begin{cases} E_{elec} \cdot L + E_{AD} \cdot L + \epsilon_{fs} \cdot L \cdot d^2 & \text{if } d \leq d_0 \\ E_{elec} \cdot L + E_{AD} \cdot L + \epsilon_{amp} \cdot L \cdot d^4 & \text{if } d > d_0 \end{cases} \tag{6.1}$$

Where E_{elec} is the energy consumed per bit sent or received, E_{AD} is the energy consumed for data aggregation, ϵ_{fs} and ϵ_{amp} represent the energy consumption of free space propagation and multipath propagation, respectively $d_0 = \sqrt{\frac{\epsilon_{fs}}{\epsilon_{amp}}}$ is the distance threshold between the sending node and the receiving node.

The energy consumed by a node receiving a message of length L bits is calculated by:

$$E_{RX}(L) = E_{elec} \cdot L \tag{6.2}$$

Given that the energy cost of communication between two sensor nodes depends on the distance between them, minimizing the energy consumption of an IoT network involves minimizing the total distance between the nodes and the cluster heads, as well as between the cluster heads and the base station.

6.2 Mathematical Model

The p-median problem is a special localization problem of combinatorial optimization. Given a graph $G = (V, E)$ consisting of a set V of n vertices connected by the edges in set E, with d_{ij} representing the distance between two vertices i and j in V, the p-median problem involves selecting p vertices from V to serve as "medians". The remaining $n - p$ vertices in V will be "clients" assigned to the medians. The objective is to minimize the total distance between the medians and the clients.

In the clustering problem, an IoT network can be represented as a graph G defined by the set V of sensor nodes and the set E comprising all possible connections between the sensor nodes in V as well as all connections between the sensor nodes and the base station. Therefore, finding a clustering that minimizes energy consumption in an IoT network equates to solving the p-median problem on graph G. The selected medians correspond to the cluster heads, while the assignment of clients to medians represents the formation of clusters, i.e., determining the member nodes of each cluster.

Such a clustering results in a routing scheme that minimizes energy consumption, reduces interference, and improves the quality of radio links by minimizing the total distance traversed by all packets. It also helps to minimize data loss by enabling one-hop transmission of data packets between nodes and cluster heads, as well as between cluster heads and the base station (without intermediate nodes).

Without loss of generality, we consider a network that contains a single base station with unlimited power supply. The sensor nodes and the base station are stationary, and their geographical positions are well known. Additionally, when a sensor node transmits data, the data packet cannot be fragmented and must be transmitted in its entirety. Consequently, the following decision variables are defined [54]:

$$y_j = \begin{cases} 1 & \text{If } \textit{node } j \textit{ is selected cluster head}, \\ 0 & \text{If } \textit{not}. \end{cases} \tag{6.3}$$

$$x_{i,j} = \begin{cases} 1 & \text{If } \textit{node } i \textit{ is associated with cluster head } j, \\ 0 & \text{If } \textit{not}. \end{cases} \tag{6.4}$$

It is clear that a node i can only be associated with a node that has already been selected as a cluster head, hence:

$$x_{i,j} \leq y_j \tag{6.5}$$

Additionally, each node i must be associated with a single cluster head. Thus:

$$\sum_{j=1}^{n} x_{i,j} = 1 \tag{6.6}$$

On the other hand, each cluster head j must have sufficient energy to receive and transmit its own data as well as the data from all the nodes associated with it. Therefore [54]:

$$\sum_{i=1}^{n} x_{i,j} \left(E_{RX}(L_i) + E_{TX}(L_i, d_j)\right) \leq y_j \left(E_j - E_{TX}(L_j, d_j)\right) \tag{6.7}$$

Where E_j is the available energy in node j, and d_j is the distance between node j and the base station.

Finally, the number of cluster heads to be selected must be equal to p.

$$\sum_{j=1}^{n} y_j = p \tag{6.8}$$

The objective is to minimize the energy consumed between nodes and the energy consumed between head of clusters and the base station, it can therefore be formulated as follows [54]:

$$Min \sum_{i=1}^{n} \sum_{j=1}^{n} (E_{TX}(d_{ij}, L) + E_{RX}(L) + E_{TX}(d_j, L)) \cdot x_{i,j} + \sum_{j=1}^{n} E_{TX}(d_j, L) \cdot y_j \tag{6.9}$$

With d_{ij} being the distance between nodes i and j, and d_j being the distance between head of cluster j and the base station.

6.3 Resolution

Several exact methods have been proposed to find optimal solutions, primarily using branch-and-bound, branch-and-cut, and branch-and-price algorithms. Since the p-median problem is NP-hard on a general network, the aforementioned methods cannot solve the p-median problem within a reasonable time for large networks and/or when p is arbitrary. Consequently, several heuristic and metaheuristic methods have been developed to generate high-quality solutions, which are slightly below optimal, in significantly less computation time compared to exact methods.

In this work, we perform a comprehensive comparison between the branch-and-bound algorithm, which provides the exact solution, and various metaheuristic approaches. Specifically, we examine the performance of the hybrid versions of the Grey Wolf Optimizer and Whale Optimizer, as well as the Puma Optimizer. Our study focuses on several key aspects, including the computational time required to reach a solution and the quality of the solutions generated. By analyzing these factors, we aim to determine the effectiveness and efficiency of these metaheuristic methods in comparison to the exact branch-and-bound algorithm, particularly in terms of their ability to provide high-quality solutions within a reasonable timeframe.

6.4 Simulations and Results

We conducted performance tests on wireless IoT networks consisting of a base station and 100 sensor nodes distributed randomly over an area of 10,000 m^2. Each sensor node has an initial energy of 2J. The base station can be located either at the center of the network or outside the network area. The simulation parameters are summarized in the following table.

Parameter	**Value (unit)**
Area	10,000 (m^2) 2D / 400.000 (m^3) 3D
Network size	100
Base station position	(50, 120) 2D / (50,120,40) 3D
Data packet size (L)	500 (bits)
Initial energy (E_0)	2 (J)
Energy consumed in electronics (E_{elec})	50 (nJ/bit)
Free space amplifier energy dissipation (E_{fs})	10 (pJ/bit/m^2)
Multi-path amplifier energy dissipation (E_{amp})	0.0013 (pJ/bit/m^4)
Energy consumed for data aggregation (E_{DA})	5 (nJ/bit)
Distance threshold (d_0)	87 (m)

Table 6.1: Parameters and values used in the model.

In the first series of tests, we consider a variant where the number p of cluster heads has been fixed at 5, as in most previous works. In the second series of tests, we consider another variant where the number p is treated as an integer decision variable, and its value is to be determined during the problem resolution. For both series of tests, we used distributions of sensors in two-dimensional (2D) and three-dimensional (3D) spaces to evaluate the performance of the algorithms under different spatial configurations.

6.4.1 Comparison of Results

The tables below summarize the performance of the different algorithms in terms of simulation time, energy optimization, and the number of clusters found.

After extensive testing, we selected 1,000 search agents and 500 iterations for the algorithms. This configuration balances optimization time and practical applicability for Internet of Things (IoT) networks. It's important to mention that each algorithm was tested 3 times in each case, since the metaheuristic algorithms are of stochastic nature, and chose the one which performed better in each case.

		GWO	WOA	PO
Fix number of clusters	Time(s)	80.2634	78.4127	258.5863
	Energy(mJ)	5.4816	5.4816	5.4690
	Number of clusters	5	5	5
Variable number of clusters	Time(s)	83.3149	79.0144	218.6492
	Energy(mJ)	5.4133	5.4166	5.3803
	Number of clusters	10	11	12

Table 6.2: Table of results for 2D data

		GWO	WOA	PO
Fix number of clusters	Time(s)	83.0889	83.8679	233.2434
	Energy(mJ)	5.7829	5.7829	5.7829
	Number of clusters	5	5	5
Variable number of clusters	Time(s)	83.6148	81.9620	254.2327
	Energy(mJ)	5.5715	5.5778	5.5859
	Number of clusters	14	13	14

Table 6.3: Table of results for 3D data

Below are figures from the simulations of 2D and 3D configurations using the various meta-heuristic optimization algorithms discussed earlier. These simulations were conducted for both fixed and variable numbers of clusters. In the figures, red dots represent cluster heads, black dots represent other sensors, and blue points represent the base. Blue dashed lines indicate the communication links between the base and the cluster heads.

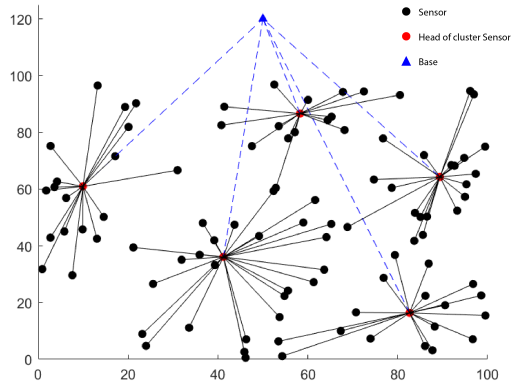

Figure 6.1: GWO, Fixed Clusters number (2D).

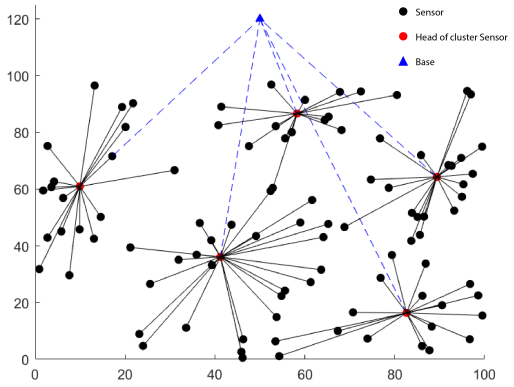

Figure 6.2: WOA, Fixed Clusters number (2D).

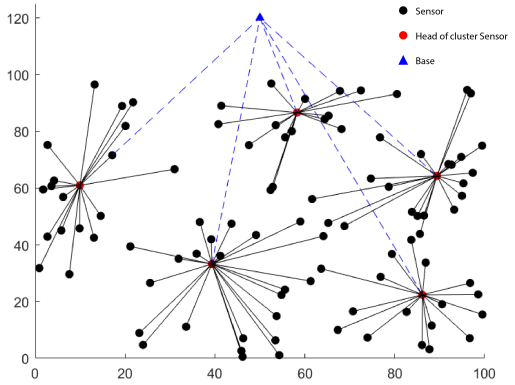

Figure 6.3: PO, Fixed Clusters number (2D).

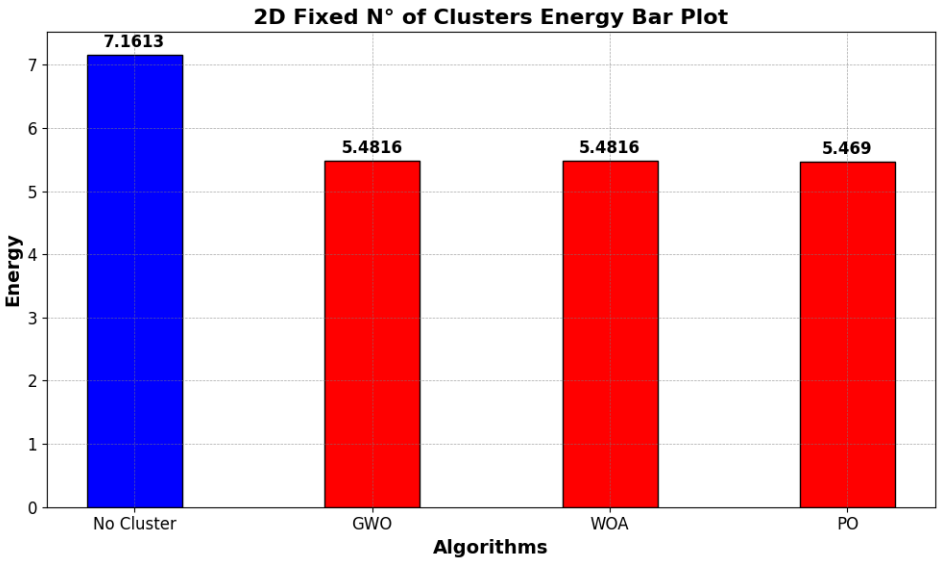

Figure 6.4: (2D) Fixed Clusters number.

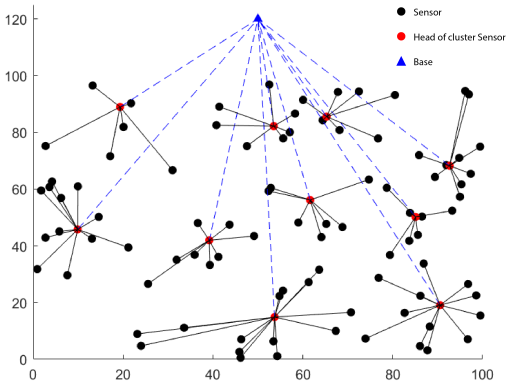

Figure 6.5: GWO, Variable Clusters number (2D).

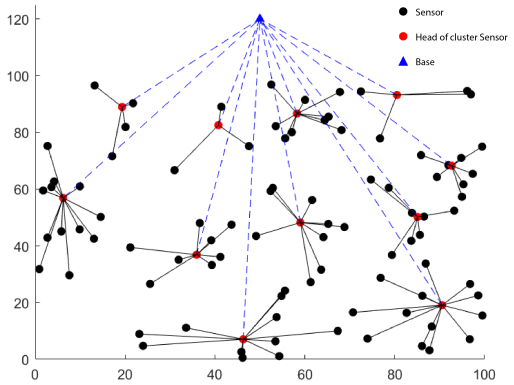

Figure 6.6: WOA, Variable Clusters number (2D).

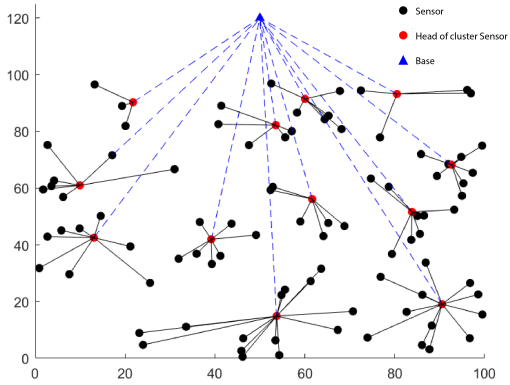

Figure 6.7: PO, Variable Clusters number (2D).

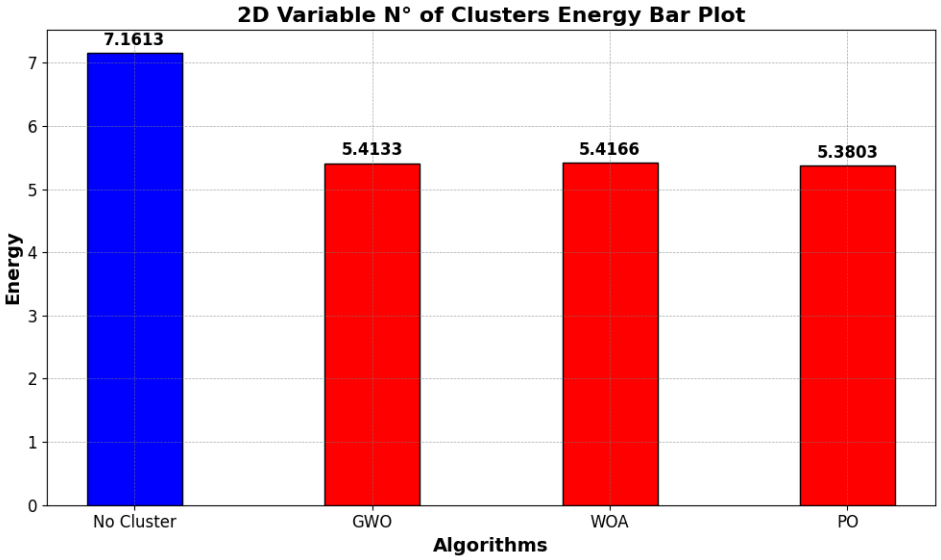

Figure 6.8: (2D) Variable Clusters number.

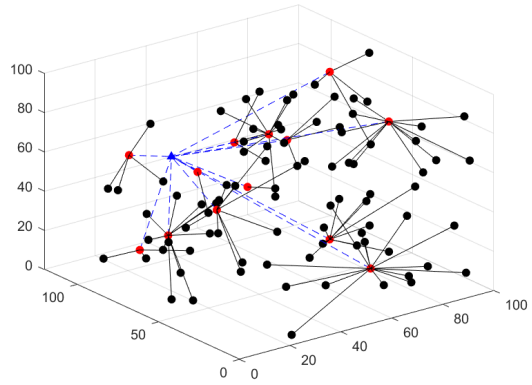

Figure 6.9: GWO, Fixed Clusters number (3D).

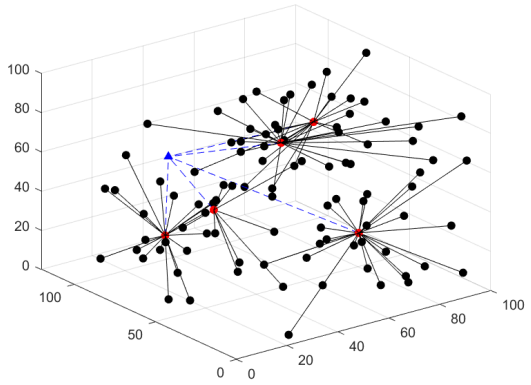

Figure 6.10: WOA, Fixed Clusters number (3D).

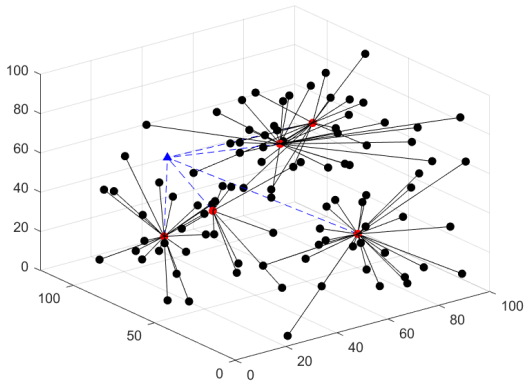

Figure 6.11: PO, Fixed Clusters number (3D).

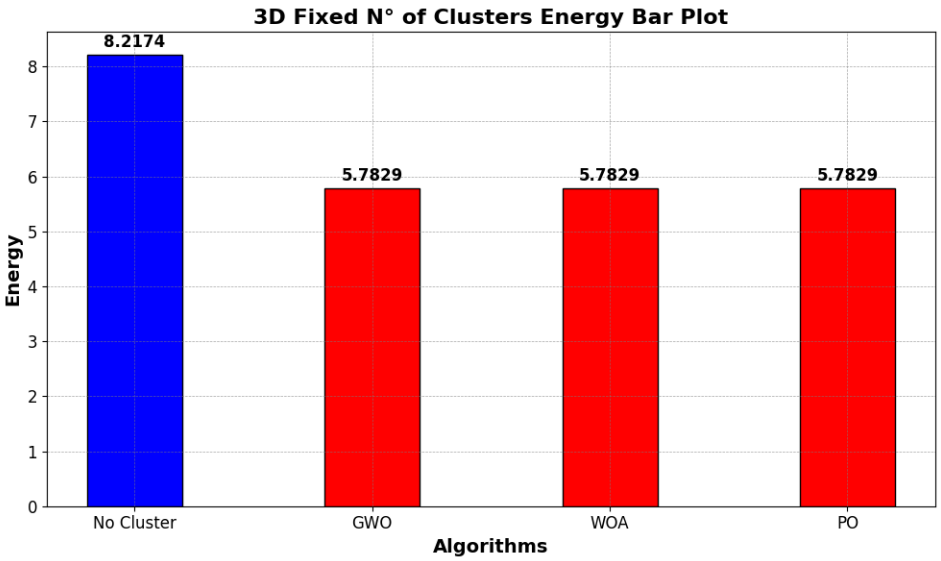

Figure 6.12: (3D) Fixed Clusters number.

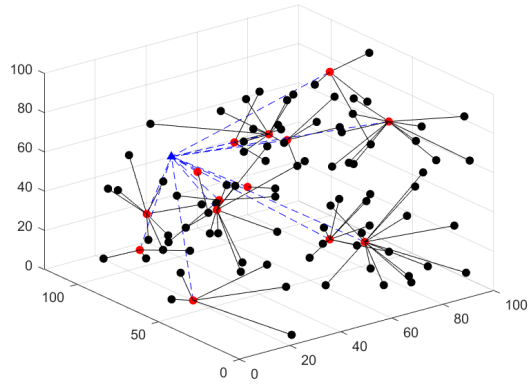

Figure 6.13: GWO, Variable Clusters number (3D).

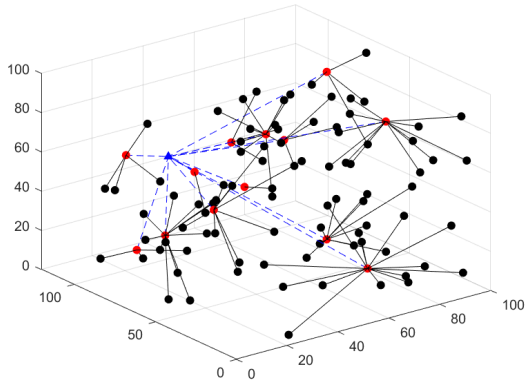

Figure 6.14: WOA, Variable Clusters number (3D).

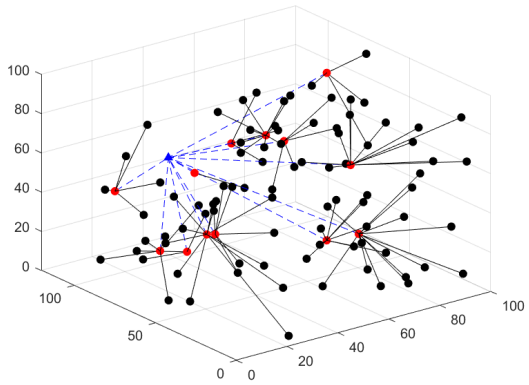

Figure 6.15: PO, Variable Clusters number (3D).

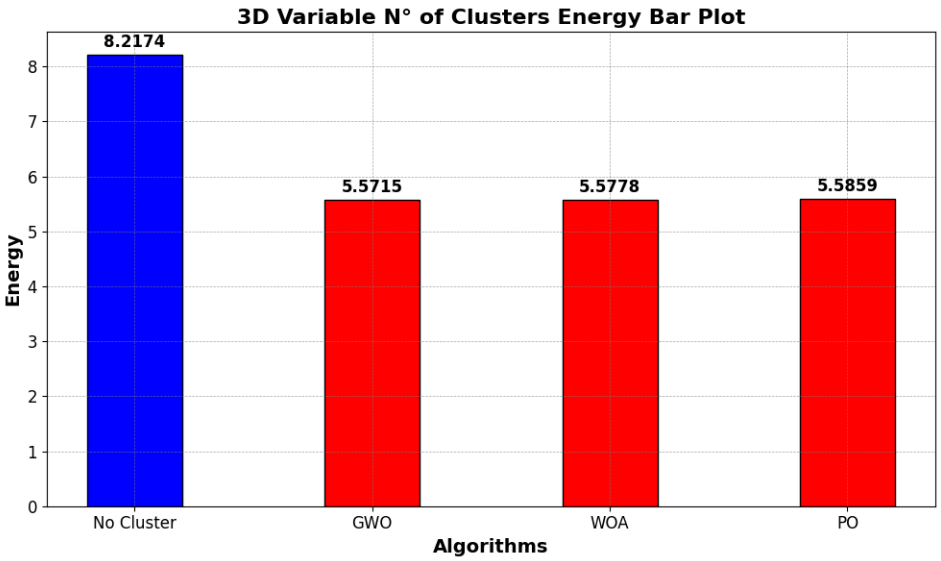

Figure 6.16: (3D) Variable Clusters number.

6.5 Discussion

From the previous tables and histograms, it is evident that the clustering approach effectively minimizes energy consumption. In 2D framework simulations, energy consumption improved by approximately 24.5%, and in 3D framework simulations, it improved by approximately 32%. Distinguishing between algorithms was challenging because energy optimization results were very close. Regarding execution time, the GWO and WOA algorithms produced similar results, whereas the PO algorithm exhibited significantly longer execution times due to its complexity.

The additional dimension did not in any way increase complexity, as the implementation remained largely unchanged except for adjustments in distance calculation formulas. Similarly, the positioning of the base relative to sensor areas do not complicate the problem, as the method used to solve the research problem remained consistent. In addition, Both 2D and 3D frameworks show that selecting varying numbers of clusters optimizes energy consumption better, attributed to the additional degree of freedom in the problem.

Another important point to mention is the application of metaheuristics for optimization tasks, particularly in clustering, are highly effective because they can quickly produce solutions that are close to optimal, even when faced with complex problems where the search space grows exponentially, such as our 2^n problem. Unlike traditional methods that search for exact solutions, which can take days, months, or even years to converge, metaheuristics offer efficient algorithms that significantly reduce computation time. This efficiency makes them particularly suitable for addressing our challenging optimization needs.

6.6 Conclusion

In conclusion, the evaluated clustering approaches have shown considerable advantages in reducing energy consumption in both 2D and 3D frameworks. Their effectiveness in reducing computational time underscores their suitability for addressing challenging optimization needs in various problem domains.

Future work involves an exploration of up to 10 to 20 different optimization algorithms to thoroughly compare their performance against the current results. This comparative analysis will provide deeper insights into the strengths and weaknesses of each algorithm, enabling better informed decision-making in algorithm selection for specific applications. Furthermore, It would be interesting to deploy this clustering algorithms in real-world scenarios. This approach will offer a practical validation of the algorithms' effectiveness, showcasing how they perform under varying conditions such as different data distributions, network topologies, and operational constraints. By doing so, it aims to bridge the gap between theoretical performance metrics and practical utility in diverse application contexts.

In addition to performance evaluation, another critical aspect of future research involves conducting robustness analysis. This analysis will assess how clustering algorithms withstand challenges posed by noisy or incomplete information about sensor placements, akin to real-world scenarios where data may be corrupted or sensors may fail unexpectedly. Understanding the algorithms' robustness will be pivotal in enhancing their reliability and applicability in dynamic and unpredictable environments. By pursuing these avenues of research, the aim is to advance the field's understanding of clustering algorithms' capabilities and limitations, ultimately contributing to more resilient and efficient solutions for real-world sensor network deployments.

General Conclusion

Artificial Intelligence (AI) has become an increasingly vital tool in control engineering, offering sophisticated techniques for modeling, estimation, and optimization of complex systems. In this thesis, we explored several AI-driven approaches to enhance the control and estimation of nonlinear processes in aircraft and IoT sensor networks.

Firstly, we modeled mass variations in both fuel and electrically powered aircrafts. We then designed an observer to estimate icing accumulations. This observer was rigorously tested under different initial conditions and demonstrated a robust capability to accurately track mass changes due to ice accretion, providing a reliable method for real-time monitoring and safety assurance in aviation.

Secondly, we investigated two learning-based identification and modeling techniques for nonlinear systems: Recurrent Neural Network (RNN) based identification and the Sparse Identification of Nonlinear Dynamics (SINDy) algorithm. Both approaches were applied to an aircraft system, showing promising results in accurately modeling the system's dynamics. The comparative analysis highlighted the strengths and potential applications of each method, providing valuable insights into their suitability for different types of nonlinear system identification tasks.

Lastly, we compared the performance of three optimization algorithms applied to a sensor network in an IoT research problem. The study was conducted in both 2D and 3D planes, with variable and fixed numbers of clusters. The results demonstrated the efficiency of introducing a variable number of clusters, adding a degree of freedom that significantly enhanced the optimization process. This finding underscores the importance of adaptable clustering strategies in optimizing sensor networks for various IoT applications.

Future work will focus on the real-time implementation of these methods to further validate their practical applicability. Additionally, efforts will be made to reduce the amount of data required for training, making the models more efficient and scalable. An investigation into a broader range of learning algorithms will also be pursued to identify the most effective approaches for different control and estimation tasks.

In summary, this thesis demonstrates the powerful capabilities of AI in advancing the field of control engineering, offering innovative solutions for complex modeling, estimation, and optimization challenges. The promising results pave the way for future research and practical applications, ultimately contributing to the development of smarter, more efficient control systems.

Appendices

Appendix A

Symbol	Parameter	Value
C_{D0}	Zero-lift coefficient	0.0375
k	Lift-independant drag coefficient	0.0588
ρ_0	Air density	$1.225Kg/m^2$
m_f	Final mass (zero load)	$4980Kg$

Table A1 : Table of parameters (J31 Jetstream)

Appendix B

Coefficient	Expression
G	$J_x J_z - J_{xz}^2$
G_1	$(J_{xz}(J_x - J_y + J_z))/G$
G_2	$(J_z(J_z - J_y) + J_{xz}^2)/G$
G_3	$(J_z)/G$
G_4	$(J_{xz})/G$
G_5	$(J_z - J_x)/J_y$
G_6	$(J_{xz})/J_y$
G_7	$(J_{xz})/J_y$
G_8	$(J_x)/G$

Table B1 : Table of coefficients (Aerosonde)

Symbol	Parameter	Value
M	initial mass	13.5Kg
J_x	moment of inertia around roll-axis	$0.824Kgm^2$
J_y	moment of inertia around pitch-axis	$1.135Kg.m^2$
J_z	moment of inertia around yaw-axis	$1.759Kg.m^2$
J_{xz}	product of inertia in the xz-plane	$0.12Kg.m^2$

Table B2 : Table of parameters (Aerosonde)

Bibliography

[1] Neural networks from scratch. https://victorzhou.com/series/neural-networks-from-scratch/.

[2] Zhe wu and Panagiotis Christofides. Economic machine-learning-based predictive control of nonlinear systems. *Mathematics*, 7:494, 06 2019.

[3] Kai Fukami, Takaaki Murata, Kai Zhang, and Koji Fukagata. Sparse identification of nonlinear dynamics with low-dimensionalized flow representations. *Journal of Fluid Mechanics*, 926:A10, 2021.

[4] Seyedali Mirjalili, Seyed Mirjalili, and Andrew Lewis. Grey wolf optimizer. *Advances in Engineering Software*, 69:46–61, 03 2014.

[5] Seyedali Mirjalili and Andrew Lewis. The whale optimization algorithm. *Advances in Engineering Software*, 95:51–67, 2016.

[6] Benyamin Abdollahzadeh, Nima Khodadadi, Saeid Barshandeh, Pavel Trojovsky, Farhad Soleimanian Gharehchopogh, El-Sayed El-kenawy, Laith Abualigah, and Seyedali Mirjalili. Puma optimizer (po): a novel metaheuristic optimization algorithm and its application in machine learning. *Cluster Computing*, pages 1–49, 01 2024.

[7] In-flight icing (2024). https://www.faa.gov/nextgen/programs/weather/awrp/ifi.

[8] Air algérie flight 5017 (2014). https://en.wikipedia.org/wiki/Air_Alg%C3%A9rie_Flight_5017.

[9] West wind aviation flight 282 (2017). https://en.wikipedia.org/wiki/West_Wind_Aviation_Flight_282.

[10] Myanmar plane: Bad weather blamed for andaman sea crash (2017). https://www.bbc.com/news/world-asia-40653524.

[11] Bek air flight 2100 (2019). https://en.wikipedia.org/wiki/Bek_Air_Flight_2100.

[12] Michael Bragg, Tamer Başar, William Perkins, Michael Selig, Petros Voulgaris, James Melody, and Nadine Sarter. Smart icing systems for aircraft icing safety. 01 2002.

[13] Fikret Caliskan, Rahmi Aykan, and Chingiz Hajiyev. Aircraft icing detection, identification, and reconfigurable control based on kalman filtering and neural networks. *Journal of Aerospace Engineering - J AEROSP ENG*, 21, 04 2008.

[14] Lejun Chen and James Whidborne. Flight data validation of an icing accretion estimation scheme using super-twisting observers. *Automatica*, 155, 09 2023.

[15] N.J. Lawson, H. Jacques, J.E. Gautrey, A.K. Cooke, J.C. Holt, and K.P. Garry. Jetstream 31 national flying laboratory: Lift and drag measurement and modelling. *Aerospace Science and Technology*, 60:84–95, 2017.

[16] Yiqun Dong and Jianliang Ai. Inflight parameter identification and icing location detection of the aircraft: The time-varying case. *Journal of Control Science and Engineering*, 2014, 07 2014.

[17] Sibo Li and Roberto Paoli. Aircraft icing severity evaluation. *Encyclopedia*, 2(1):56–69, 2022.

[18] Fikret Caliskan, Rahmi Aykan, and Chingiz Hajiyev. Aircraft icing detection, identification, and reconfigurable control based on kalman filtering and neural networks. *Journal of Aerospace Engineering - J AEROSP ENG*, 21, 04 2008.

[19] Yiqun Dong. An application of deep neural networks to the in-flight parameter identification for detection and characterization of aircraft icing. *Aerospace Science and Technology*, 77:34–49, 2018.

[20] Mohammed S. Alhajeri, Zhe Wu, David Rincon, Fahad Albalawi, and Panagiotis D. Christofides. Machine-learning-based state estimation and predictive control of nonlinear processes. *Chemical Engineering Research and Design*, 167:268–280, 2021.

[21] Zhe wu, Anh Tran, David Rincón, and Panagiotis Christofides. Machine learning-based predictive control of nonlinear processes. part i: Theory. *AIChE Journal*, 65, 08 2019.

[22] Zhe Wu, Anh Tran, David Rincon, and Panagiotis D. Christofides. Machine-learning-based predictive control of nonlinear processes. part ii: Computational implementation. *AIChE Journal*, 65(11):e16734, 2019.

[23] Steven L. Brunton, Joshua L. Proctor, and J. Nathan Kutz. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences*, 113(15):3932–3937, 2016.

[24] Steven L. Brunton, Joshua L. Proctor, and J. Nathan Kutz. Sparse identification of nonlinear dynamics with control (sindyc). *IFAC-PapersOnLine*, 49(18):710–715, 2016. 10th IFAC Symposium on Nonlinear Control Systems NOLCOS 2016.

[25] David Luenberger. Observing the state of a linear system. *Military Electronics, IEEE Transactions on*, MIL8:74 – 80, 05 1964.

[26] R. E. Kalman. A New Approach to Linear Filtering and Prediction Problems. *Journal of Basic Engineering*, 82(1):35–45, 03 1960.

[27] ANTONIO TORNAMBÉ. High-gain observers for non-linear systems. *International Journal of Systems Science*, 23(9):1475–1489, 1992.

[28] Christopher Edwards and Sarah K. Spurgeon. Sliding mode control : theory and applications. 1998.

[29] A.Jafari Koshkouei and Alan Zinober. Sliding mode controller-observer design for multivariable linear systems with unmatched uncertainty. *Kybernetika*, 1, 01 2000.

[30] Yousra Mahmoudi, Nadjet Zioui, Hacène Belbachir, hanane dagdougui, and Said Bentouba. Reducing wireless sensors networks energy consumption using p-median modelling and optimization. 6:180–196, 07 2023.

[31] José Leite, Paulo Martins, and Edson Ursini. *Internet of Things: An Overview of Architecture, Models, Technologies, Protocols and Applications: Emerging Trends and Challenges in Technology*, pages 75–85. 01 2019.

[32] Martinez Alonso, Rodney and Plets, David and Fontes Pupo, Ernesto and Deruyck, Margot and Martens, Luc and Guillen Nieto, Glauco and Joseph, Wout. IoT-based management platform for real-time spectrum and energy optimization of broadcasting networks. *WIRELESS COMMUNICATIONS MOBILE COMPUTING*, page 14, 2018.

[33] Arun Kumar, Ali Asghar Rahmani Hosseinabadi, Morteza Babazadeh Shareh, Atekeh Zolfagharian, Naveen Chilamkurti, and Seyed Yaser Bozorgi Rad. Iot resource allocation and optimization based on heuristic algorithm. *Sensors*, 20:539, 01 2020.

[34] Sakshi Popli, Rakesh Kumar Jha, and Sanjeev Jain. A survey on energy efficient narrowband internet of things (nbiot): Architecture, application and challenges. *IEEE Access*, 7:16739–16776, 2019.

[35] Ugur Tekin, Jaafar Gaber, Maxime Wack, and Julien Bourgeois. Iot activities tuning for energy consumption optimization. *Procedia Computer Science*, 175:566–571, 2020. The 17th International Conference on Mobile Systems and Pervasive Computing (MobiSPC),The 15th International Conference on Future Networks and Communications (FNC),The 10th International Conference on Sustainable Energy Information Technology.

[36] Neeti Kashyap, A. Charan Kumari, and Rita Chhikara. Multi-objective optimization using nsga ii for service composition in iot. *Procedia Computer Science*, 167:1928–1933, 2020. International Conference on Computational Intelligence and Data Science.

[37] Mamoun Alazab, Kuruva Lakshmanna, Thippa Reddy G, Quoc-Viet Pham, and Praveen Kumar Reddy Maddikunta. Multi-objective cluster head selection using fitness averaged rider optimization algorithm for iot networks in smart cities. *Sustainable Energy Technologies and Assessments*, 43:100973, 2021.

[38] Hao Ran Chi and Ayman Radwan. Multi-objective optimization of green small cell allocation for iot applications in smart city. *IEEE Access*, 8:101903–101914, 2020.

[39] Osama Alsaryrah, Ibrahim Mashal, and Tein-Yaw Chung. Bi-objective optimization for energy aware internet of things service composition. *IEEE Access*, 6:26809–26819, 2018.

[40] Jun Huang, Liqian Xu, Cong-cong Xing, and Qiang Duan. A novel bioinspired multiobjective optimization algorithm for designing wireless sensor networks in the internet of things. *Journal of Sensors*, 2015:1–16, 08 2015.

[41] Narges Mehran and Dragi Kimovski. Mapo: A multi-objective model for iot application placement in a fog environment, 10 2019.

[42] Praveen Reddy and M. Babu. Energy efficient cluster head selection for internet of things. *New Review of Information Networking*, 22:54–70, 01 2017.

[43] Xiang-Yang Li, Yu Wang, Haiming Chen, Xiaowen Chu, Yanwei Wu, and Yong Qi. Reliable and energy-efficient routing for static wireless ad hoc networks with unreliable links. *IEEE Transactions on Parallel and Distributed Systems*, 20(10):1408–1421, 2009.

[44] Zhixin Liu, Lili Dai, Liang Xue, Xinping Guan, and Changchun Hua. Reliability considered routing protocol in wireless sensor networks. In *Proceedings of the 30th Chinese Control Conference*, pages 5011–5016, 2011.

[45] Habib Mostafaei. Energy-efficient algorithm for reliable routing of wireless sensor networks. *IEEE Transactions on Industrial Electronics*, 66(7):5567–5575, 2019.

[46] Aamir Mahmood, M. M. Aftab Hossain, Cicek Cavdar, and Mikael Gidlund. Energy-reliability aware link optimization for battery-powered iot devices with nonideal power amplifiers. *IEEE Internet of Things Journal*, 6(3):5058–5067, 2019.

[47] D. Binu and B. S Kariyappa. Ridenn: A new rider optimization algorithm-based neural network for fault diagnosis in analog circuits. *IEEE Transactions on Instrumentation and Measurement*, 68(1):2–26, 2019.

[48] Yunfeng Xu, Ping Fan, and Ling Yuan. A simple and efficient artificial bee colony algorithm. *Mathematical Problems in Engineering*, 2013, 01 2013.

[49] John McCall. Genetic algorithms for modelling and optimisation. *Journal of Computational and Applied Mathematics*, 184(1):205–222, 2005. Special Issue on Mathematics Applied to Immunology.

[50] M.E.H. Pedersen and A.J. Chipperfield. Simplifying particle swarm optimization. *Applied Soft Computing*, 10(2):618–628, 2010.

[51] Esmat Rashedi, Hossein Nezamabadi-pour, and Saeid Saryazdi. Gsa: A gravitational search algorithm. *Information Sciences*, 179(13):2232–2248, 2009. Special Section on High Order Fuzzy Sets.

[52] Kai-Chun Chu, Der-Juinn Horng, and Kuo-Chi Chang. Numerical optimization of the energy consumption for wireless sensor networks based on an improved ant colony algorithm. *IEEE Access*, 7:105562–105571, 2019.

[53] Leila Abbad, Azzedine Nacer, Houda Abbad, Mohammed Taieb Brahim, and Nadjet Zioui. A weighted markov-clustering routing protocol for optimizing energy use in wireless sensor networks. *Egyptian Informatics Journal*, 23(3):483–497, 2022.

[54] Yousra Mahmoudi, Nadjet Zioui, and Hacene Belbachir. A new quantum-inspired clustering method for reducing energy consumption in iot networks. *Internet of Things*, 20:100622, 2022.

[55] Harshit Gupta, Amir Dastjerdi, Soumya Ghosh, and Rajkumar Buyya. ifogsim: A toolkit for modeling and simulation of resource management techniques in internet of things, edge and fog computing environments. *Software: Practice and Experience*, 47, 06 2016.

[56] Abdolkarim Mohammadi-Balani, Mahmoud Nayeri, Adel Azar, and Mohammadreza Taghizadeh-Yazdi. Golden eagle optimizer: A nature-inspired metaheuristic algorithm. *Computers Industrial Engineering*, 152:107050, 12 2020.

[57] Soudeh Shadravan, Hamid Naji, and Vahid Bardsiri. The sailfish optimizer: A novel nature-inspired metaheuristic algorithm for solving constrained engineering optimization problems. *Engineering Applications of Artificial Intelligence*, 80, 02 2019.

[58] Mohammad Dehghani, Zeinab Montazeri, Eva Trojovska, and Pavel Trojovsky. Coati optimization algorithm: A new bio-inspired metaheuristic algorithm for solving optimization problems. *Knowledge-Based Systems*, 259:110011, 01 2023.

[59] Laith Abualigah, Mohammad Shehab, Mohammad Al Shinwan, Seyedali Mirjalili, and Mohamed Elsayed Abd Elaziz. Ant lion optimizer: A comprehensive survey of its variants and applications. *Archives of Computational Methods in Engineering*, 28, 04 2020.

[60] Xiao Cui and Hao Shi. A*-based pathfinding in modern computer games. 11, 11 2010.

[61] Tamer Başar, E. Schuchard, J. Melody, W. Perkins, and P. Voulgaris. Detection and classification of aircraft icing using neural networks. 02 2000.

[62] Developing and understanding continuous-time rnns (ctrnns) for neuroscience applications. https://medium.com/@wangzhz1027/build-a-continuous-time-rnn-for-neuroscience-tasks-8de9b0313275.

[63] R. Hermann and A. Krener. Nonlinear controllability and observability. *IEEE Transactions on Automatic Control*, 22(5):728–740, 1977.