POLITECNICO DI TORINO

Master's Degree in Computer Engineering

Master's Degree Thesis

A DNN-based algorithm for multi-constraint intelligent reentry guidance techology for hypersonic gliding vehicle

Supervisors

Prof. Diego REGRUTO TOMALINO

Prof. Sophie FOSSON

Prof. Lin CHENG

Candidate

Camilla Zulli

Academic Year 2023/2024
Torino

Abstract

A hypersonic vehicle is an aircraft with a flight speed exceeding 5 times the speed of sound, combining the characteristics of both spacecraft and aircraft, with significant military and economic potential. The thesis focuses on a re-entry hypersonic vehicle, which is a spacecraft that travels through space and re-enters the atmosphere of a planet (e.g., Earth). When landing, a safe re-entry is needed. The entry guidance system is crucial for ensuring a successful atmospheric re-entry, particularly for this class of vehicles which faces multiple constraints during this critical flight phase. Traditional entry guidance approaches, such as reference trajectory-based guidance (RTG) and numerical predictor–corrector guidance (NPCG), have been widely employed in past missions. However, the high non-linearity and non-convex path constraints of hypersonic vehicles demand more advanced solutions.

This thesis introduces an intelligent multi-constraint entry guidance approach that integrates a deep neural network (DNN) with the NPCG algorithm for better real-time performance, and ensures safety during atmospheric re-entry. The DNN is designed to approximate the relationship between flight states and range and flight time, enabling rapid and accurate trajectory predictions. The developed DNN-based predictor significantly improves the NPCG algorithm by replacing traditional propagation-based predictions, offering robust solutions with real-time computational capability.

The structure of the thesis is organised as follows: initially, a global overview about hypersonic vehicles and related works is established. Then, the first section involves the development of the DNN using *Python*, with training, testing, and validation phases. Subsequently, the DNN is integrated into the NPCG algorithm, where the bank angle and the angle of attack of the vehicle are used as control variables. The guidance algorithm consists of longitudinal control, in which the bank angle and angle of attack amplitudes are determined, for each guidance cycle, by constructing a parametrized height profile, in order to satisfy the range constraint, and a velocity profile, for meeting the time constraint. Then, lateral control is elaborated, for the managment of the sign of the bank angle, for each cycle.

Finally, the thesis presents the results of the proposed approach in terms of

trajectory, longitude, and latitude achieved by the vehicle at the end of the controlled flight phase. The analysis demonstrates the advantages of incorporating artificial intelligence into the guidance algorithm, offering improved real-time decision-making capabilities and overall performance.

Acknowledgements

RINGRAZIAMENTI

Ringrazio tutte le persone che mi hanno supportata in questi anni, ognuna a modo proprio.

Per prima, la mia famiglia. Senza di voi, tutto questo non sarebbe stato possibile: mi avete permesso di seguire e portare a termine questo percorso e vivere esperienze incredibili. Mi avete insegnato a puntare in alto e ad inseguire i miei sogni, incoraggiandomi sempre, anche quando alcuni ostacoli sembravano insormontabili. Un enorme grazie a mamma, papà e Filippa per essere stati i miei più grandi sostenitori.

Un ringraziamento speciale a mia nonna, Lucia e ai miei nonni, Raffaele e Rosetta. Grazie, per essere sempre stati la mia forza durante questi anni e per esserlo nella vita, dandomi preziosi consigli e confortandomi nei momenti più difficili.

Grazie anche a mio zio, Barbara e Carlotta. Grazie per aver contribuito al raggiungimento dei miei obbiettivi. Vi voglio bene!

Ringrazio i miei amici di una vita, che mi sono stati accanto in ogni momento di questa avventura, nonostante la distanza fisica dell'ultimo periodo.

A Roberta, da anni ormai la mia certezza in amicizia, ovunque io mi trovi nel mondo. Ci sei sempre, per ascoltarmi, consigliarmi, spronarmi e condividere sia dubbi e paure che momenti felici e spensierati. Mi auguro che il nostro legame duri per tutta la vita, perché non ne potrei fare a meno.

A Federica, un'amicizia che dura da quando eravamo bambine, ben prima che l'università fosse anche solo nei nostri pensieri. Grazie per avermi insegnato cosa significa avere un'amica che mi supporta in ogni scelta, lasciandomi libera di essere me stessa.

Un grande grazie anche a Marianna, dolce, gentile e determinata. La tua presenza è ormai diventata per me un porto sicuro, non importa quanto tempo starò via da casa, perché so che ci sarai sempre al ritorno, per poter costruire altri ricordi indimenticanili insieme.

E ad Angelica, capace di essere te stessa in ogni situazione, senza mai compromessi, con la tua ironia, che ha reso speciali tanti momenti. Sei stata per me un'importante fonte di ispirazione durante questi anni, oltre che un'amica fantastica, e continui ad esserlo.

A Chiara e Asia, siete entrate nella mia vita più tardi, ma i momenti che abbiamo passato insieme sono stati preziosi e fondamentali. Spero di viverne molti altri ancora.

Grazie a Lorenzo, per essere un punto di riferimento ogni volta che torno a casa, e per essere il miglior vicino che potessi desiderare.

Grazie a tutti i miei amici di sempre: Riccardo, Samuele, Glendi, Morgan e Nico, grazie!

Un ringraziamento va anche ai miei colleghi di università. Anche se solo per un anno, avete reso unico questo percorso. Un grazie speciale a Matteo, Francesca e Gaetano per aver reso indimenticabili le giornate di lezione, le sessioni infinite e i momenti di divertimento.

Un grazie di cuore a Franceska, sei arrivata nel momento giusto, quando avevo bisogno di serenità e di sentirmi a casa anche lontano da casa.

Grazie a tutte le persone meravigliose che ho incontrato durante il mio viaggio in Cina. Un grazie particolare a Martina, con cui ho condiviso un'esperienza unica, e senza la quale probabilmente avrei fatto molta più fatica a partire.

Infine, un ringraziamento va ai professori che mi hanno permesso di portare a termine questo lavoro: il Prof. Diego Regruto Tomalino, la Prof.ssa Sophie Fosson, il Prof. Lin Cheng e Jingjing Xu. Grazie tante per il vostro supporto.

Table of Contents

List of Tables

List of Figures

Acronyms

RTG
Reference Trajectory-based Guidance

NPCG
Numerical Predictor-Corrector Guidance

DNN
Deep Neural Network

NASA
National Aeronautics and Space Administration

PRC
People's Republic of China

CAV-L
Common Aereo Vehicle Low Performance

AI
Artificial Intelligence

APDG
Apollo Powered Descent Guidance

ANN
Artificial Neural Network

NN
Neural Network

ReLU

Rectified Linear Unit

ELU

Exponential Linear Unit

FFNN

Feed-Forward Neural Network

CNN

Convolutional Neural Network

RNN

Recurrent Neural Network

LSTM

Long Short-Term Memory

NLP

Natural Language Processing

NTM

Neural Turing Machine

TAEM

Terminal Area Energy Management

MSE

Mean Squared Error

MAE

Mean Absolute Error

EI

Entry Interface

HV

Height-Velocity

QEGC

Quasi-Equilibrium Glide Condition

PI

Proportional-Integral

LOS

Line of Sight

Chapter 1

Introduction

[1] Hypersonic vehicles are a class of aircraft able of traveling at speeds greater than Mach 5. This category includes a range of vehicles, such as airplanes, missiles, and spacecraft, combining the characteristics of both aircraft and space vehicles, thus is mostly involved in military and economic fields. An example of re-entry vehicle for military purporse is shown in Figure 1.1. In this thesis, we focus on hypersonic re-entry vehicles, which are designed to travel through space and safely re-enter a planet's atmosphere, such as Earth's.

The atmospheric entry is a challenging process, due to the extreme conditions which are involved. As the vehicle descends at hypersonic speeds, it encounters intense aerodynamic forces, together with the generation of extremely high temperatures. Engineers must concentrate both on the design of the vehicle and the control techniques of its guidance, ensuring the integrity of the structure and safety of the mission. Historically, missions like NASA's Apollo program (1961-1972) and Space Shuttle have demonstrated the complexity of this task, where precise control over the vehicle trajectory, deceleration, and aerodynamic forces is essential for a successful re-entry. More recently, vehicles like NASA's Orion capsule (Figure 1.2), intended for deep space exploration, are specifically designed to withstand the intense heat generated during re-entry from lunar missions, where the capsule re-enters Earth's atmosphere at speeds of up to 40,000 km/h. In the context of modern research, countries around the world, including the United States, Russia, and China, have made significant progress in hypersonic technologies. The most recent developments have led to a focus on two main categories of these vehicles: powered and unpowered. Re-entry aircraft belong to the latter category, for this reason, they can be also defined as gliding.

The research work for this thesis has been carried out at the research laboratory of the School of Astronautics at Beihang University (laboratory in Figure 1.3) in

Beijing, PRC. This work has been completed also thanks to the support and guidance of a dedicated research group with professors, students, and PhD candidates from the university. The topic has been chosen with the aim of contributing to one of the group's key area of study: the integration of artificial intelligence (AI) into guidance algorithms, with the purpose of enhancing the performance of the re-entry of hypersonic vehicles, with a special attention to the real-time correction of the trajectory, where classical procedures, such as reference trajectory-based guidance (RTG) and numerical predictor–corrector guidance (NPCG), are lacking. Traditional methods, while effective, often face challenges when addressing the complexity of non-linear flight dynamics and non-convex path constraints. The research group at Beihang University has already explored various approaches to improve guidance systems, including those that combine AI technologies with numerical algorithms to optimize the results.

In this thesis, we depart from the intelligent algorithm developed by the research team [2], introducing a different approach. While the main job of the team's algorithm is constructing a parametrized bank angle profile, to follow during the flight of the vehicle, in order to meet the range constraint, our approach focuses on the design of a parameterized height profile, to satisfy the range constraint, and a reference velocity profile, for meeting the time constraint. Both approaches exploit the predictions of a deep neural network (DNN) that approximates flight states with respect to range and flight time. According to the determination of reference height and velocity, the guidance algorithm computes the bank angle and angle of attack amplitudes for the current guidance cycle, for longitudinal control, while lateral control involves determining the sign of the bank angle for each cycle.

The development and results of this intelligent guidance algorithm, as well as the key theoretical foundations behind it, will be further elaborated in the following chapters.

Figure 1.1: Representation of a common aero vehicle - low performance (CAV-L). An unpowered vehicle designed for precision targeting on Earth (image generated with AI), [3]

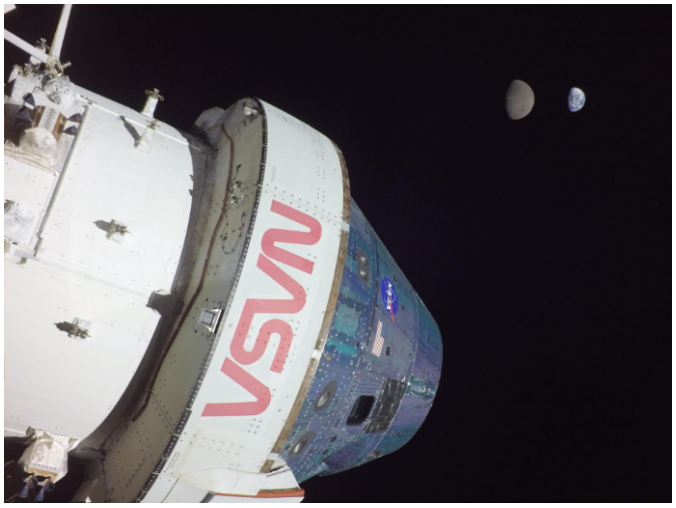

Figure 1.2: NASA Orion Re-entry Vehicle from NASA website[1]

[1]NASA website URL: `https://www.nasa.gov/reference/orion-spacecraft/`

Figure 1.3: Laboratories of the School of Astronautics of Beihang University

1.1 Structure of the document

The document is organized as follows:

- Chapter 2 - A comprehensive review of the related works in the field of hypersonic vehicles guidance and control, and a detailed overview of deep neural networks (DNNs), including fundamental concepts and structures. Then a brief focus on the previous study of the research group at Beihang is provided.
- Chapter 3 - Explanation of the specific DNN architecture developed for this research, along with its role in predicting flight characteristics.
- Chapter 4 - Presentation of the longitudinal guidance strategy, with an emphasis on how the bank angle and angle of attack amplitudes are computed based on the height and velocity profiles, ensuring adherence to range and time constraints.
- Chapter 5 - A thorough description of the lateral guidance method, outlining the process for determining the sign of the bank angle for each guidance cycle.
- Chapter 6 - Discussion of the obtained results from simulations, analyzing the performance of the proposed guidance approach and comparing it with traditional methods.

- Chapter 7 - Conclusion, summarizing the key findings of the thesis, the implications for future research, and the potential applications of this innovative guidance technology.
- Appendix.

Chapter 2

Related works

2.1 Numerical predictor-corrector guidance (NPCG) overview

Numerical predictor-corrector guidance (NPCG) is a foundamental technique used to manage the complex re-entry trajectories of hypersonic vehicles and plays a vital role in this process by predicting and correcting the trajectory of the vehicle to meet the requested constraints.

2.1.1 How NPCG works

[4] The NPCG operates by segmenting the entry trajectory into several phases, each governed by specific set of control variables. A typical entry trajectory, such as the one shown in Figure 2.1, begins at the atmospheric interface and ends at touchdown. The figure emphasizes the descent phase using a logarithmic scale, showcasing the progression of the trajectory through its various segments. Each of these segments requires adjustments to the control vector (e.g., vehicle commands) to meet mission constraints.

The NPCG consists of a systematic process that involves both predicting and correcting the flight path of the vehicle. The following lines show how it works:

- **Prediction:** The algorithm generates a predicted trajectory based on the current state and control vector of the vehicle. This is accomplished by integration of the three or six degrees of freedom (3DOF or 6DOF) equations of motion, which model the dynamics during the flight.
- **Correction:** After establishing the predicted trajectory, the algorithm compares it to the specified constraints (such as landing site coordinates or altitude

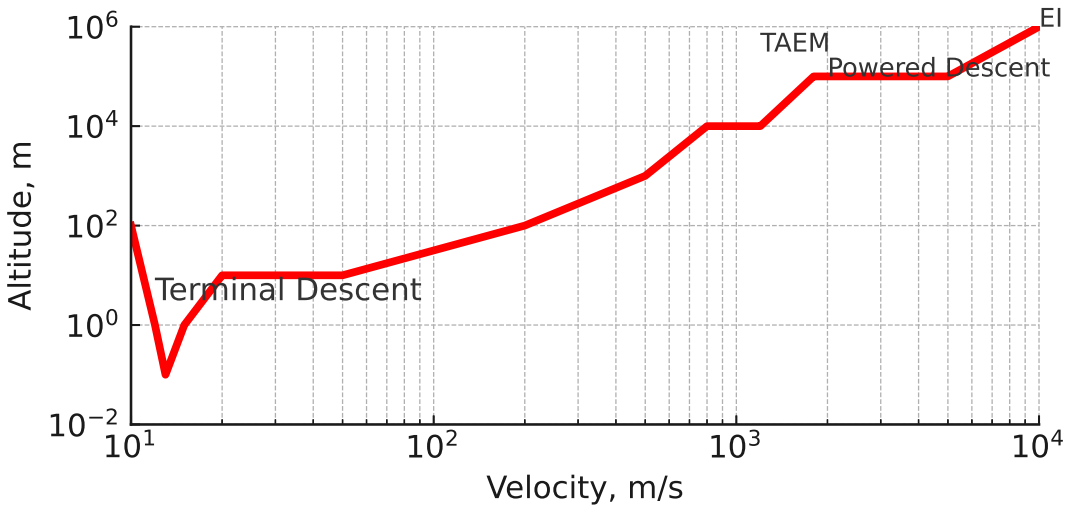

Figure 2.1: Reproduction of a notional entry trajectory with notional guidance segments denoted. Note logarithmic scales on both axes, [4]

profiles). It calculates the errors between the predicted and desired trajectories and determines the necessary adjustments to the control vector to minimize these errors. This correction step uses a gradient-based targeting algorithm to find the "best" control commands for the current guidance segment.

The converged control vector solution is commanded until the NPCG is called again, effectively managing the path in real time. For example, the Apollo powered descent guidance (APDG) generates a desired acceleration vector profile to target the specific landing sites.

A flowchart illustrating the logic flow of the NPCG is shown in Figure 6.1. For each segment of the flight, specific waypoints or targets, such as landing site coordinates, are defined and given as input to the system. Moreover, the algorithm utilizes several inputs, including the current state of the vehicle from the navigation system, accelerations from the inertial measurement unit (IMU), and when available, atmospheric data.

These inputs are used by the NPCG to update internal models related to aerodynamics, gravity, and atmospheric conditions, all of which vary based on the specific requirements of the mission. The updated models allow the NPCG to propagate the trajectory toward a predefined terminal point. The targeting algorithm, working together with the trajectory propagator, then determines the control vector that best satisfies the mission constraints for each segment of the flight.

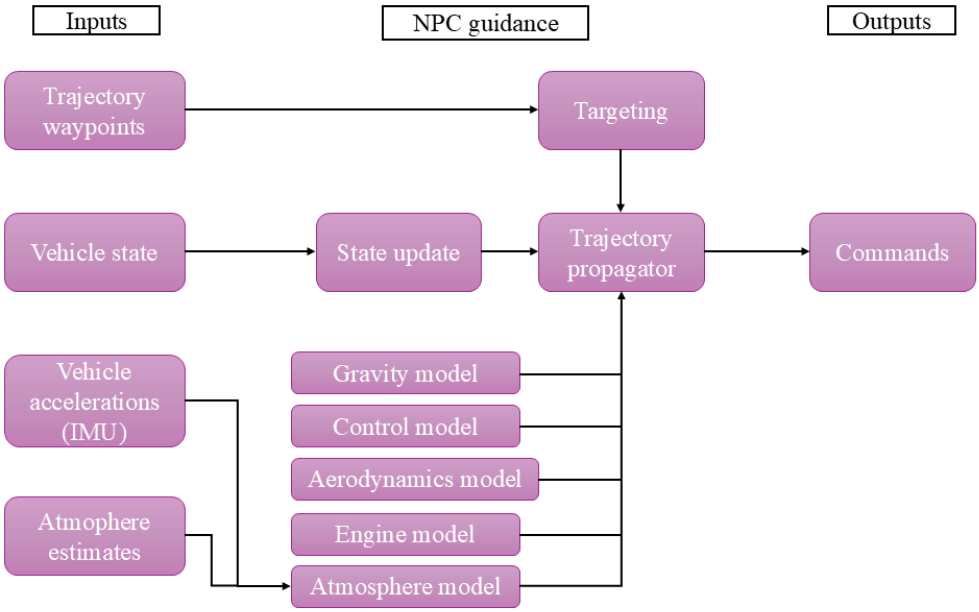

Figure 2.2: Outline of a typical NPC guidance logic, [4]

2.1.2 Advantages and drawbacks of NPCG

On the one hand, NPCG offers several advantages respect to other entry guidance approaches, such as RTG [2]:

- **Accuracy:** NPCG reaches extremely accurate trajectory predictions that can be adjusted in real time, which is essential for meeting strict landing constraints during re-entry.
- **Flexibility:** Unlike RTG methods that rely on offline trajectory planning and online reference trajectory tracking, NPCG does not depend on an optimized reference trajectory. This makes it better suited for handling large trajectory dispersions and the complexities of hypersonic flight.
- **Improved performance:** With advancements in computer technology, NPCG has seen significant developments, allowing it to outperform RTG approaches in certain scenarios, particularly when faced with non-linear flight dynamics.

On the other hand, there are some critical issues associated with this technique:

- **Computational complexity:** The need for real-time trajectory propagation and error correction can require considerable computational resources, making it challenging to implement in time-sensitive situations. This is the reason why the DNN technique is utilized to better enable the real-time performance, as we will see later.
- **Non-linear dynamics:** The non-linear nature of hypersonic flight dynamics can complicate the prediction and correction processes, requiring advanced algorithms and robust models to ensure reliable performance.

2.1.3 Applications of NPCG

NPCG is employed in various aerospace missions where precise trajectory control is necessary. In particular, it is applied in human-scale Mars entry, descent, and landing scenarios [4], where the complexities of atmospheric interaction necessitate reliable guidance systems. This algorithm is also relevant in the development of next-generation space vehicles, demonstrating its versatility and importance in aerospace engineering.

2.2 What is a DNN?

[5] [6] Deep learning is a quickly growing field within AI and machine learning. It builds upon the fundamental principles of artificial neural networks (ANNs), which are computational models inspired by the structure and function of the human brain. ANNs are composed of layers of interconnected nodes (neurons), where each neuron processes input and passes the information forward to the next layer. While ANNs are effective in solving a variety of tasks, their superficial architecture often limits their ability to learn complex patterns in data.

This is where Deep Neural Networks (DNNs) come into play. A DNN is essentially an ANN with multiple hidden layers, allowing it to model complex and hierarchical relationships in data. These deep architectures enable DNNs to learn intricate patterns that are otherwise difficult for shallow networks to capture. This capability has positioned DNNs at the forefront of the research field and real-world applications, from image recognition and natural language processing to self-driving cars and even creative tasks like generating art or music.

In deep learning, the layers of a DNN are trained to automatically extract features from raw data, eliminating the need for manual feature engineering. Each layer in the network progressively transforms the input into higher-level representations, with early layers capturing simple patterns, like edges in an image, and deeper layers capturing more abstract concepts, like objects or scenes. The availability of large datasets, combined with advancements in computational power (especially

graphics processing units or GPUs), has fueled the rise of deep learning in recent years, enabling DNNs to outperform traditional machine learning methods in many domains.

Activation function

An essential component of each layer in a DNN is the **activation function** [7], which introduces non-linearity to the model, simulating biological activation to input stimuli. Without activation functions, the network would simply be a series of linear transformations, limiting its ability to learn complex patterns. Common activation functions include *ReLU* (Rectified Linear Unit), *sigmoid*, *tanh*, and *ELU* (Exponential Linear Unit).

Classification and regression

In this thesis, we develop a DNN specifically for solving a **regression problem**. A *regression* problem involves predicting a continuous output variable based on one or more input features, while a *classification* problem focuses on assigning input data to discrete categories.

Deep learning can be particularly advantageous for regression problems like the one we have to face with. The choice of deep learning over traditional machine learning methods is justified in several ways:

- **Handling complex relationships:** Regression problems often involve complex, non-linear relationships between input features and output variables. While traditional machine learning algorithms, such as linear regression or support vector machines, may struggle to capture these complexities, DNNs can model intricate patterns effectively due to their multiple layers and non-linear activation functions.

- **Large dataset requirements:** The performance of DNNs improves with larger datasets. Traditional machine learning methods may be sufficient for small or moderate datasets but cannot perform as well when the amount of data is extensive. The availability of a consistent number of data points allows the DNN to learn more robust patterns and relationships.

- **Automatic feature extraction:** DNNs can automatically learn features from the raw input data without extensive feature engineering. This contrasts with traditional machine learning, which often requires domain knowledge to manually select relevant features.

Supervised and unsupervised learning

In general, classification or regression can be approached using different learning paradigms: *supervised learning* involves training a model on labeled data, where the input features are paired with the corresponding output labels, whereas *unsupervised learning* deals with unlabeled data, where the model must identify patterns or structures within the data without explicit guidance.

Preparation phase

The setup phase of a DNN typically involves three main steps:

1. **Training:** The model learns from the training dataset by adjusting its weights to minimize the error between the predicted outputs and the actual outputs.
2. **Validation:** The performance of the model is evaluated on a separate validation dataset to tune hyperparameters and prevent overfitting.
3. **Testing:** Finally, the model is assessed on a testing dataset to measure its generalization ability on unseen data.

2.2.1 Types of DNN

Deep learning includes a variety of DNN architectures, each optimized for specific types of data and tasks. Below are some types of DNNs worth mentioning, which have a profound impact on AI research and applications.

1. **Feed forward neural networks (FFNNs)**
   Feed forward neural networks (FFNNs), also known as dense networks, are the simplest type of DNN. In FFNNs, each neuron in one layer is connected to every neuron in the next layer (fully connected), allowing for a comprehensive learning of complex functions. They are typically used for tasks where the input data is structured and does not have any specific spatial or temporal dependencies, such as tabular data or simple classification or regression tasks.

   The components of a FFNN include:

   - **Input layer:** This layer receives the input data.
   - **Hidden layers:** These layers transform the input into higher-level representations using activation functions, allowing the network to learn non-linear relationships.
   - **Output layer:** This layer produces the final output, which could be a class label for classification tasks or a continuous value for regression tasks.

Although FFNNs are less suited for tasks involving spatial or sequential data compared to other architectures, they provide a fundamental understanding of neural networks and serve as the building blocks for more complex DNNs.

2. **Convolutional neural networks (CNNs)**
   Convolutional Neural Networks (CNNs) are one of the most widely used architectures in deep learning, especially for image and visual data. CNNs excel at detecting spatial hierarchies in data, making them particularly suited for image classification, object detection, and segmentation tasks. Unlike traditional fully connected networks, CNNs apply a series of convolutional filters to the input, which helps to capture local patterns (e.g., edges, textures) and reduce the dimensionality of the data.

   The components of a CNN include:

   - **Convolutional layers:** These apply filters to the input image to create feature maps. Each filter detects specific patterns such as edges or textures at different spatial locations.
   - **Pooling layers:** These reduce the dimensionality of the feature maps by selecting representative values (e.g., max pooling), which makes the network more computationally efficient while retaining important information.
   - **Fully connected layers:** The high-level features extracted from the convolutional and pooling layers are passed to fully connected layers (as a FFNN) for final classification.

   CNNs have achieved remarkable success in image classification benchmarks, such as the ImageNet competition, where they significantly reduced error rates. Their hierarchical structure allows them to process data in a way similar to how the human visual cortex processes images, identifying patterns and assembling them into more complex structures.
   In Figure 2.3 the architecture of a CNN is represented.

3. **Recurrent neural networks (RNNs) and long short-term memory networks (LSTMs)**
   Recurrent Neural Networks (RNNs) are designed to handle sequential data (as shown in Figure 2.4), making them ideal for tasks where context and order are crucial, such as speech recognition, time-series prediction, and natural language processing (NLP). Unlike feed forward networks, RNNs have connections that loop back, allowing information to persist across time steps. This structure enables RNNs to "remember" previous inputs, which is essential for processing sequences where the current output depends on previous data points.

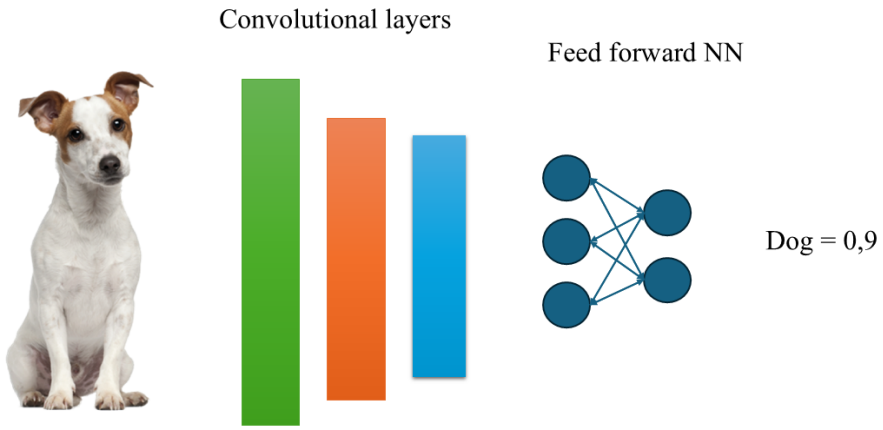

Figure 2.3: CNN architecture, showing convolutional, pooling, and fully connected layers

However, traditional RNNs suffer from issues like the vanishing gradient problem, which makes it difficult for them to learn long-term dependencies. To address this, long short-term memory (LSTM) networks were developed. LSTMs are a type of RNN that includes memory cells capable of maintaining information over long sequences. These cells are regulated by gates that control the flow of information, allowing the network to learn when to remember and when to forget.

LSTMs have been particularly successful in NLP tasks, where understanding the context of words and sentences is crucial. For instance, LSTMs have been used in machine translation systems, enabling accurate translation by maintaining the context of previous words in a sentence.

4. **Autoencoders**
   Autoencoders are a specific type of DNN used for unsupervised learning. They are designed to learn efficient representations (encodings) of input data, typically for tasks like dimensionality reduction, denoising (see Figure 2.5), or anomaly detection. An autoencoder consists of two main parts:

   - **Encoder:** This part compresses the input data into a lower-dimensional representation (the "bottleneck").

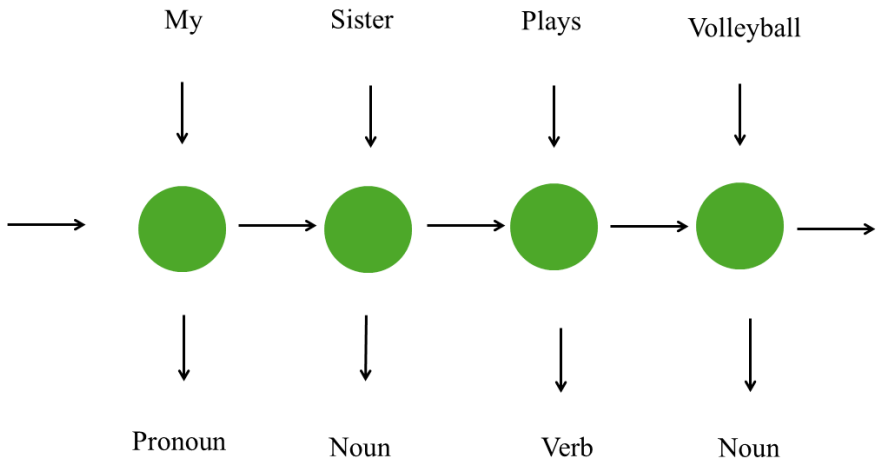

Figure 2.4: RNN architecture

- **Decoder:** The decoder reconstructs the input data from the compressed representation.

The goal of an autoencoder is to minimize the difference between the original input and its reconstruction, forcing the network to learn a compact and efficient representation of the data. Deep autoencoders, with multiple layers of encoders and decoders, are used for tasks like image compression, where they learn to represent complex features in a more abstract form.

5. **Neural turing machines and memory networks**
   Neural Turing Machines (NTMs) and Memory Networks are examples of architectures that extend the capabilities of traditional RNNs by introducing external memory components. These networks are designed to perform tasks that require complex reasoning, such as question-answering or solving problems that involve recalling information from a long sequence.

   - **Neural turing machines (NTMs):** NTMs combine the power of RNNs with an external memory bank, allowing the network to learn how to read from and write to memory. This architecture makes NTMs particularly effective for tasks that involve algorithmic processes, like sorting or following instructions.

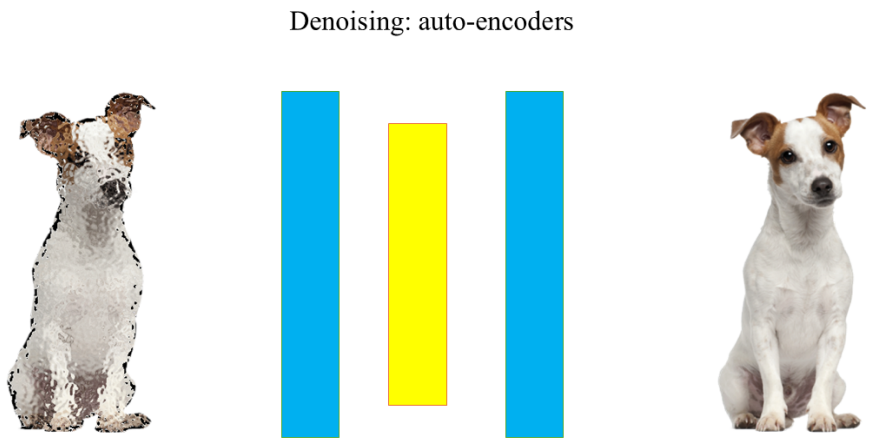

Figure 2.5: Autoencoder architecture

- **Memory networks:** These networks also incorporate an external memory component but are mainly designed for tasks involving reasoning and inference, such as understanding stories and answering questions about them. Memory networks have shown strong performance in question-answering benchmarks by storing relevant information and recalling it when needed.

Both NTMs and memory networks are powerful tools for tasks requiring long-term memory and reasoning, pushing the boundaries of what DNNs can achieve.

2.3 Focus on the work of Lin Cheng et al. [2]: bank angle parameterization and range constraint

The work introduces a novel predictor-corrector entry guidance algorithm aimed at improving real-time control of hypersonic vehicles during atmospheric re-entry. A central element of their approach is the parametrization of the bank angle profile, which is designed to meet both path and terminal constraints while achieving the desired covered distance or range. The total distance constraint or range-to-go s,

can be decomposed into two components:

- **Downrange:** The distance traveled by the vehicle along its trajectory in the horizontal direction, measured from the entry interface or launch point to the target location. It represents the longitudinal distance covered or remaining along the intended flight path.
- **Crossrange:** The lateral distance perpendicular to the vehicle downrange, representing the deviation from the ideal flight path. It is used to assess and correct the lateral position of the vehicle during flight to ensure accurate heading control.

This work involves the parametrization of the bank angle and the use of DNN for range prediction and correction in both the longitudinal and lateral channels.

2.3.1 Bank angle parameterization for range control

The paper addresses the trajectory planning problem by parameterizing the bank angle using a compound bank angle corridor. This approach simplifies the complex entry guidance problem into a univariate root-finding problem, which is solved iteratively to ensure compliance with the range constraint.

The entry flight is divided into three phases:

- **Initial descent phase**: During this phase, the vehicle bank angle is kept constant at a maximum allowable value, denoted as σ_{Imax}, to prevent excessive altitude loss and avoid exceeding heating rate constraints.
- **Quasi-equilibrium glide phase**: In this phase, the bank angle is controlled to maintain the vehicle altitude above the minimum boundary determined by path constraints, such as heating rate and dynamic pressure. The maximum allowable bank angle is denoted as σ_{Emax}.
- **Pre-TAEM phase**: The final phase before the terminal area energy management (TAEM) interface, where the bank angle σ_{TAEM} is controlled to meet the terminal height and velocity constraints.

The bank angle profile is determined by the following weighted sum:

$$|\sigma(v)|_\omega = \omega \cdot \sigma_{min}(v) + (1 - \omega) \cdot \sigma_{max}(v) \tag{2.1}$$

where ω is a weighting coefficient used as parameter to adjust the bank angle based on the desired downrange. The function $\sigma_{min}(v)$ represents the lower boundary of the bank angle corridor, while $\sigma_{max}(v)$ represents the upper boundary. By adjusting ω, the vehicle can achieve different longitudinal downranges, with the relationship between ω and the downrange being monotonically increasing. This ensures that a larger ω leads to a smaller bank angle and thus a larger downrange.

2.3.2 DNN for longitudinal and lateral guidance

The DNN developed by Cheng et al. is trained to predict the downrange and crossrange based on the current flight states and the bank angle profile. This replaces traditional, time-consuming trajectory propagation techniques with a faster prediction method, greatly enhancing real-time performance.

DNN for longitudinal guidance

In the longitudinal channel, the DNN approximates the downrange-to-go based on the current state variables $[h, v, \theta]$ (altitude, velocity, and flight path angle) and the bank angle weighting coefficient ω. The downrange predicted by the DNN is denoted as:

$$\mathrm{Net}_{dpre}(x, \omega) \tag{2.2}$$

where $x = [h, v, \theta]$ is the vector of flight states. The iterative correction of ω is performed using a modified Newton-Raphson method:

$$\omega_{k+1} = \omega_k - \lambda_k \frac{z(\omega_k)}{\frac{\partial \mathrm{Net}_{dpre}(x,\omega_k)}{\partial \omega_k}} \tag{2.3}$$

Here, $z(\omega_k)$ is the difference between the predicted downrange $\mathrm{Net}_{dpre}(x, \omega_k)$ and the required one, and λ_k is the step size. This DNN-based iteration allows for real-time updates of the bank angle without the need for traditional trajectory propagation.

DNN for lateral guidance and bank reversals

In the lateral channel, the DNN output for the crossrange prediction is denoted as:

$$\mathrm{Net}_{cpre}(x, \omega) \tag{2.4}$$

The lateral control is achieved by adjusting the sign of the bank angle to manage heading corrections. The DNN-based algorithm predicts the crossrange and, consequently, if the bank angle is maintained or reversed, following the estimations below:

$$R_c^{Now} = \mathrm{Net}_{cpre}(x, \omega) + \mathrm{Sign}(\sigma) \cdot \mathrm{Net}_{dpre}(x, \omega) \cdot \sin(\psi - \psi_{LOS}) \tag{2.5}$$

$$R_c^{Next} = \mathrm{Net}_{cpre}(x, \omega) - \mathrm{Sign}(\sigma) \cdot \mathrm{Net}_{dpre}(x, \omega) \cdot \sin(\psi - \psi_{LOS}) \tag{2.6}$$

The decision to hold or reverse the bank angle is determined by comparing the estimated crossrange R_c^{Now}, calculated according to the current bank angle sign, with the predicted crossrange if the bank angle reverses R_c^{Next}:

$$\text{Direction} \quad \begin{cases} \text{hold} & \text{if } R_{c,\text{next}} < \frac{R_{c,\text{now}}}{k_L} \\ \text{reverse} & \text{if } R_{c,\text{next}} > \frac{R_{c,\text{now}}}{k_L} \end{cases} \tag{2.7}$$

where $k_L > 1$ is a user-defined coefficient, which determines the magnitude of the crossrange decrement by a single reversal. This predictive lateral guidance algorithm ensures a deterministic number of bank reversals, improving the control over the vehicle heading and crossrange.

Figure 2.6 illustrate the lateral guidance.

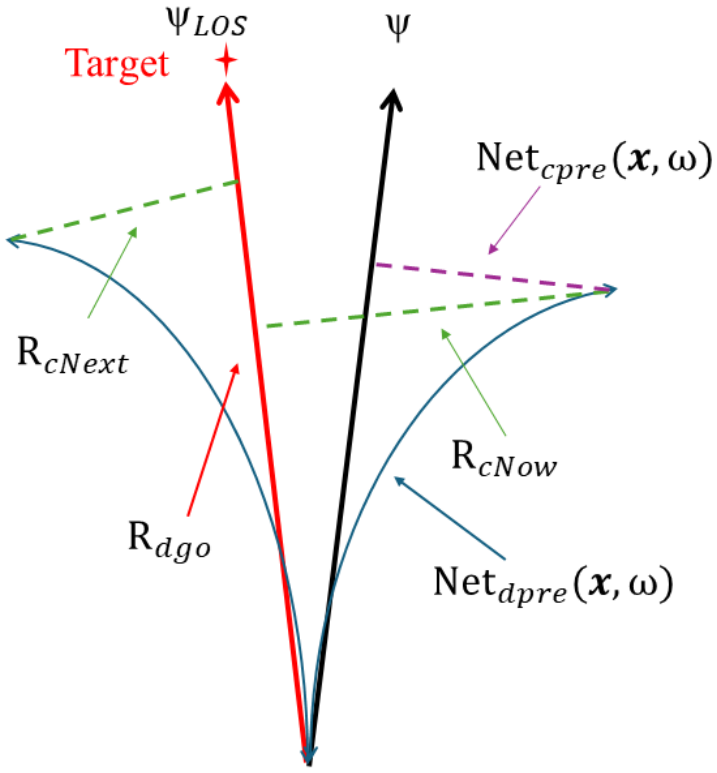

Figure 2.6: Illustration of lateral guidance. ψ_{LOS} and ψ are, respectively, the LOS angle and the heading angle (go to Section 5.1 for definitions), [2]

2.3.3 Real-time performance and constraint satisfaction

By leveraging the DNN for both longitudinal and lateral guidance, the proposed algorithm can achieve real-time performance with a trajectory update frequency of 20 Hz. In summary, the bank angle parameterization, combined with the DNN-based range prediction and correction, allows a precise control over the vehicle's trajectory while satisfying both longitudinal and lateral range constraints in real-time.

Chapter 3

Design and implementation of the DNN

The current chapter is aimed to describe the structure and the implementation of the DNN which will be integrated within the guidance algorithm, subsequently. The network is developed using *PyTorch*.

3.1 Input and output data

In this work, the objective of the DNN is to predict the two key parameters during the re-entry flight: flight time t (measured in seconds [s]) and downrange R_d, measured in radians [rad] (see definition in Section 2.3. Then, these two quantities form the output of our neural network.

Instead, the input data consists of: velocity v (measured in meters per second $[\frac{m}{s}]$) and weighting coefficient ω. Previously, we have already mentioned ω in Chapter 2, when describing the work of the research group, but, while in their work, ω was used to parameterize the bank angle profile, here it is involved to parameterize the altitude profile. By varying ω, we can effectively control the altitude profile of the vehicle during flight, always satisfying path and terminal constraints.

It is important to note that, in this chapter, the DNN is discussed as a standalone component. Its integration into the overall guidance algorithm will be addressed in subsequent chapters, where it will play a crucial role in the predictor-corrector scheme.

The training data have been provided directly by the research laboratory at Beihang University, which generated it through a detailed procedure. [2] Specifically, feasible entry trajectories with different ranges were generated. The dataset consists of 10,000 different trajectories, with 100 samples extracted from each trajectory,

resulting in a total of 1,000,000 data samples. The dataset was then divided into three subsets: 80% for training, 10% for validation, and 10% for testing, ensuring a robust learning process for the network.

This extensive dataset allows the DNN to learn the complex non-linear relationships between the input variables and the flight outcomes, ensuring accurate predictions in a variety of entry scenarios.

3.2 Network description

The neural network implemented in this work is a *fully connected feed-forward* network. It consists of 4 layers: an input layer, 2 hidden layers, and an output layer. The details of the network structure and its components are given below.

3.2.1 Structure

- **Input Layer (fc1)**: The first layer is a linear transformation that maps the input data with dimension 2 (`input_dim`) to a space with dimension 128 (`hidden_dim`). The `ELU` (Exponential Linear Unit) activation function is applied. It is defined as follows:

$$ELU(x) = \begin{cases} x & \text{if } x \geq 0 \\ \alpha(e^x - 1) & \text{if } x < 0 \end{cases} \tag{3.1}$$

  where α is typically set to 1.

- **Hidden layers (fc2, fc3)**: Two subsequent hidden layers, each containing 128 nodes. Both layers apply a linear transformation followed by the `tanh` activation function. The `tanh` function is defined as:

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{3.2}$$

  The output of $\tanh(x)$ is constrained in the range $[-1, 1]$, providing a strong non-linear mapping that helps with the learning of complex patterns.

- **Output layer (fc4)**: The final layer is a linear transformation that maps the 128-dimensional data from the last hidden layer to the output space with dimension 2. No activation function is applied to the output layer, as is typical for regression tasks where raw values are required.

3.2.2 Key characteristics summary

- **Number of layers**: 4 (including the output layer).
- **Number of nodes per layer**: 128 nodes in the hidden layers.
- **Activation functions (AF)**:
  - *ELU* in the input layer. This function allows positive outputs for $x \geq 0$ and smooth negative outputs for $x < 0$, helping to avoid the vanishing gradient problem.
  - *Tanh* in both hidden layers. This function outputs values in the range $[-1, 1]$, providing a bounded, non-linear mapping.
  - No activation function in the output layer.

Table 3.1: Structure of the DNN

Inputs	v (m/s), ω
Outputs	t (s), R_d (rad)
Network size	2 / 128
AF	Input: ELU; Hidden: Tanh; Output: None

Table 3.1 summarizes the network structure, specifying the inputs, outputs, layer sizes, and AF used in the network.

3.3 DNN accuracy evaluation

The DNN is trained using the normalized dataset provided by the laboratory. The key aspects of the training process are summarized in Table 3.2:

Parameter	**Description**
Epochs	200
Optimizer	Adam
Initial learning rate	0.002
Loss function	Mean squared error (MSE)
Batch size	32
Learning rate adjustment	Decreased by 20% every 2 epochs
Model checkpointing	Every 10 epochs, the model is saved

Table 3.2: Summary of DNN training parameters

The network is trained using mini-batches of 32 samples to stabilize convergence. The Adam optimizer is applied with an initial learning rate of 0.002, which is reduced by 20% every 2 epochs to improve performance as the model is near to the convergence. The mean squared error (MSE) loss function is used to evaluate the difference between predicted and actual outputs. The training process lasts 200 epochs, with model weights saved every 10 epochs.

3.3.1 Statistics

Three key statistical measures: mean absolute error (MAE), standard deviation (σ), and maximum absolute error, are provided in Table 3.3.

Metric	**MAE**	**Standard deviation (σ_s)**	**Maximum absolute error**
Flight time (s)	0.1703	0.1574	1.4905
Downrange (km)	0.7234	0.7035	8.2638

Table 3.3: Statistics of the DNN-based time and range prediction

For the flight time predictions, the MAE is 0.1703 s, σ_s equal to 0.1574 s and the maximum absolute error to 1.4905 s. These values indicate that the network achieved a high level of precision in predicting the flight time, with minimal variations from the actual values, considering that the time of the re-entry flight we are analyzing in this thesis is usually on the order of thousands of seconds.

For the range predictions, the MAE is 0.7234 km, and σ_s is 0.7035 km. The maximum absolute error is found to be 8.2638 km. The mean of the errors is less than 1 km, and the maximum error is less than 10 km. In this case too, since the the large values of the downrange, we can see that the network achieves a high accuracy for range prediction. From Figure 3.1 to 3.4, flight time and downrange comparisons of different speeds and weights are reported.

3.3.2 Partial derivatives

Partial derivatives are essential in the training of neural networks as they measure how changes in the input variables affect the output predictions. In this work, we utilize the `torch.autograd.grad` function from PyTorch to compute these derivatives. This function allows for automatic differentiation, enabling us to efficiently calculate gradients with respect to our model parameters. These gradients are critical for optimizing the network during the training process, as they guide the adjustments made to minimize the loss function.

Table 3.4 summarizes the computed partial derivatives for flight time t and range S:

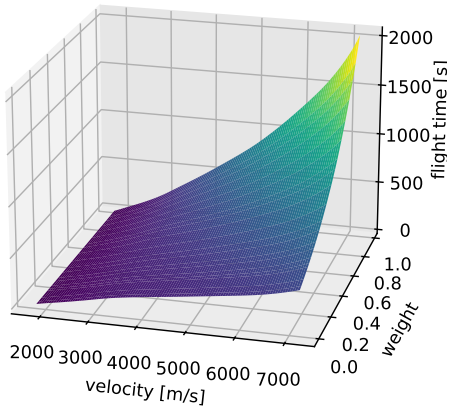

Figure 3.1: Flight time comparison of different speeds and weights

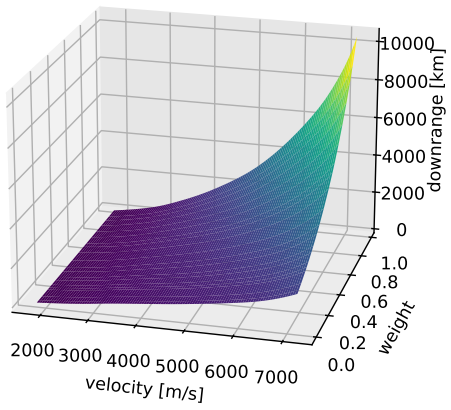

Figure 3.2: Downrange comparison of different speeds and weights

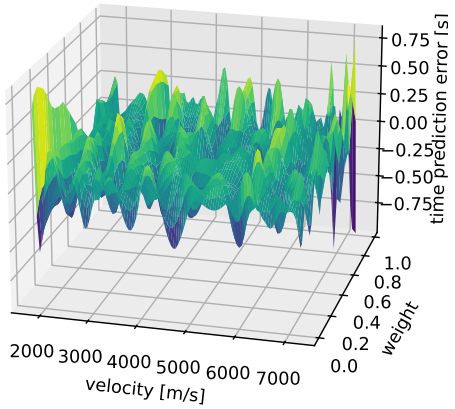

Figure 3.3: Flight time prediction error comparison of different speeds and weights

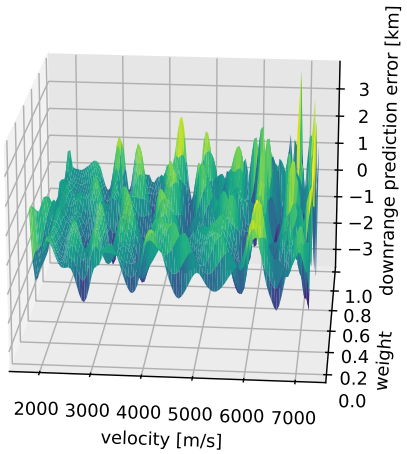

Figure 3.4: Downrange prediction error comparison of different speeds and weights

The following snippet of code illustrates how we compute the partial derivatives using PyTorch [8]:

```
1 def cal_partial(self, x: np.ndarray):
2     [...]
3
```

Variable	Partial derivative	Value
Flight time t	$\frac{dt}{dv}$	$9.24523132 \times 10^{-2}$
	$\frac{dt}{d\omega}$	2.99091414×10^{2}
Downrange R_d	$\frac{dR_d}{dv}$	2.99091414×10^{2}
	$\frac{dR_d}{d\omega}$	$1.47965992 \times 10^{-1}$

Table 3.4: Partial derivatives for flight time t and downrange R_d.

```
    # Normalize the input based on mean and standard deviation
    input_p = (x - self.xmean) / self.xstd
    input_p = torch.FloatTensor(input_p)
    input_p.requires_grad = True  # Enable gradient tracking

    # Forward pass through the neural network
    output = self.net(input_p)

    df_dx = np.empty((x.shape[0], 0, 2))  # Initialize array for partial derivatives
    for i in range(2):
        d = torch.zeros_like(output)  # Create tensor for gradient output
        d[:, i] = torch.ones_like(d[:, i])  # Set one column to compute the derivative

        # Compute the partial derivative using autograd
        df_dx_ = torch.autograd.grad(outputs=output, inputs=input_p, grad_outputs=d, create_graph=True)[0]
        df_dx_ = df_dx_.detach().numpy()  # Detach from the graph and convert to NumPy
        df_dx_ = df_dx_.reshape((x.shape[0], 1, 2))  # Reshape for concatenation
        df_dx = np.concatenate((df_dx, df_dx_), axis=1)  # Concatenate results

    out = np.zeros_like(df_dx)  # Initialize output for adjusted derivatives
    # Scale the derivatives based on standard deviations
    out[:, 0, 0] = self.ystd[0] / self.xstd[0] * df_dx[:, 0, 0]
    out[:, 0, 1] = self.ystd[0] / self.xstd[1] * df_dx[:, 0, 1]
    [...]

    return out  # Return the computed partial derivatives: dt/dv, dt/dw, dRd/dv, dRd/dw
```

Listing 3.1: Partial derivatives calculation in PyTorch

The validity of these derivative values has been thoroughly verified by the

research laboratory at Beihang University, ensuring that the model predictions are reliable and consistent.
Furthermore, the utility of the partial derivatives, specifically their indication of how changes in the input data affect the output, will be shown in Chapter 4.

3.4 Observation on the choice of input and output data

In our guidance algorithm, we must adhere to constraints on both range and flight time, which are known prior to the algorithm initiation. This raises an important question: *why are the inputs and outputs of the neural network not inverted? Specifically, why do we predict not the velocity and weighting coefficient that satisfy the constraints on flight time and range, but rather the opposite?* The answer lies in the fact that there is no one-to-one correspondence between the pairs of time-range and velocity-weighting coefficient. Therefore, if we were to invert the inputs and outputs of the neural network, the learning rate would be significantly low, resulting in poor model performance. As we will discuss in the following chapter, we will need to derive the velocity and the parameter ω from the neural network predictions.

Chapter 4

Longitudinal guidance

In this chapter, we focus on the integration of the DNN within the NPCG algorithm. By using the predictions provided by the DNN, the trajectory of the vehicle is corrected in real-time using two control variables: the bank angle (σ) and the angle of attack (α). The main purpose of the entry guidance system is to steer the vehicle from the entry interface (EI) to a desired terminal area, all while satisfying multiple constraints.

- **Bank angle (σ)**: inclination of wings of an aircraft relative to the horizontal plane (Figure 4.1) [9].
- **Angle of attack (α)**: the angle between a plane wing and the oncoming air (relative wind) (Figure 4.2) [10].

This chapter specifically illustrates longitudinal guidance, which involves determining the magnitude of both bank angle and angle of attack. The determination of the sign of σ will be discussed in the next chapter, while the sign of α is always positive.

4.1 Problem formulation

4.1.1 Entry dynamics

The 3-D and 6DOF equations of motion for a lifting vehicle through the atmosphere over a spherical Earth are described as follows, with effects of the rotation of the Earth neglected.

[3]MISB Motion Imagery Standards Board 0601.8, 23 October 2014

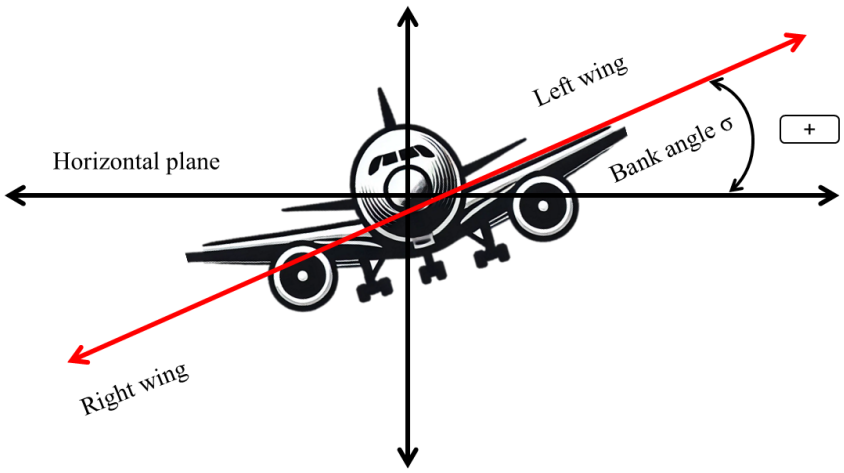

Figure 4.1: Illustration of the bank angle, the angle between the horizontal plane and the wing of the aircraft (left wing in the case of the image), [9]

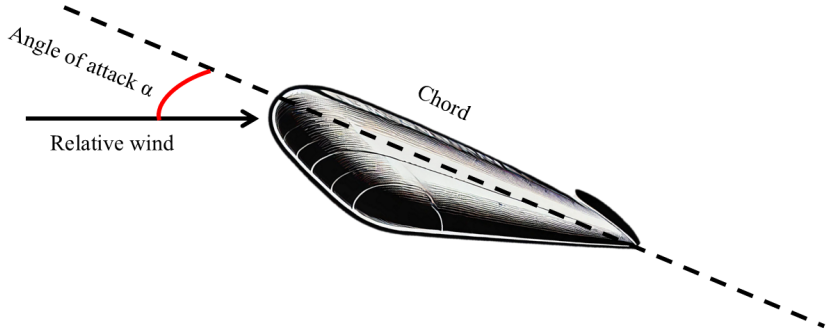

Figure 4.2: Illustration of the angle of attack. The chord is the straight-line distance between the leading edge and the trailing edge of an aircraft airfoil, [10]

$$\dot{r} = v \sin\theta \tag{4.1}$$

$$\dot{\lambda} = \frac{v \cos\theta \sin\psi}{r \cos\phi} \tag{4.2}$$

$$\dot{\phi} = \frac{v \cos\theta \cos\psi}{r} \tag{4.3}$$

$$\dot{v} = -\frac{D}{m} - g \sin\theta \tag{4.4}$$

$$\dot{\theta} = \frac{1}{v}\left(\frac{L\cos\sigma}{m} + \left(\frac{v^2}{r} - g\right)\cos\theta\right) \tag{4.5}$$

$$\dot{\psi} = \frac{1}{v}\left(\frac{L\sin\sigma}{m\cos\theta}\right) \tag{4.6}$$

Above, r is the radial distance from the Earth's center to the vehicle, λ and ϕ represent the Earth-relative longitude and latitude, respectively, and v is the Earth-relative velocity. The variable θ denotes the flight-path angle of the velocity vector, measured upward from the local horizontal direction. More, ψ represents the heading angle of the velocity vector, measured clockwise in the local horizontal plane from the north. The mass of the vehicle m is assumed to be constant, as the entry vehicle is unpowered.

The downrange distance Rd, which is used in trajectory planning, can be calculated as:

$$\dot{R}_d = \frac{v\cos\theta}{r} \tag{4.7}$$

The vehicle is driven by Earth's gravity and aerodynamic forces. The gravitational acceleration g is calculated as:

$$g = g_0 \frac{R_0^2}{r^2} \tag{4.8}$$

where $g_0 = 9.81\,\mathrm{m/s^2}$ denotes the gravitational acceleration at sea level, and $R_0 = 6371\,\mathrm{km}$ is the radius of the Earth.

The aerodynamic lift and drag forces, denoted as L and D, respectively, are expressed as:

$$L = 0.5\,\rho v^2 C_L S_{\mathrm{ref}} \tag{4.9}$$

$$D = 0.5\,\rho v^2 C_D S_{\mathrm{ref}} \tag{4.10}$$

where ρ denotes the atmospheric density, which is calculated using the US Standard Atmosphere 1976 model. S_{ref} represents the aerodynamic reference area, and C_L and C_D denote the lift and drag coefficients, respectively, which are functions of the velocity and the angle of attack α. In this case, the angle of attack α is predetermined as a function of velocity.

4.1.2 Flight constraints

The initial conditions for the hypersonic vehicle entry flight refer to the CAV-L model [3], and are defined as:

$$\begin{aligned} h_0 &= 100 \text{ km}, \\ \lambda_0 &= 0^\circ, \\ \phi_0 &= 0^\circ, \\ v_0 &= 7.200 \text{ km/s}, \\ \theta_0 &= -2^\circ, \\ \psi_0 &= 55^\circ, \\ t_0 &= 0 \text{ s}. \end{aligned} \tag{4.11}$$

where

$$r0 = h0 + R_0 \tag{4.12}$$

To ensure flight safety during the entry phase, several flight path constraints are imposed on the vehicle. These include the heating rate, total g-load (or aerodynamic load), and dynamic pressure, defined as follows:

$$\dot{Q}(t) = k_Q \left(\frac{\rho}{\rho_0}\right)^{0.5} \frac{v}{\sqrt{R_0 g_0}} \tag{4.13}$$

$$\dot{Q}(t) \leq \dot{Q}_{\max} \tag{4.14}$$

where $\dot{Q}(t)$ represents the heating rate at the stagnation point, and k_Q is a vehicle-dependent constant.
Additionally, the total g-load constraint is expressed as:

$$n(t) = \sqrt{\left(\frac{L}{mg_0}\right)^2 + \left(\frac{D}{mg_0}\right)^2} \leq n_{\max} \tag{4.15}$$

Finally, the dynamic pressure constraint is defined as:

$$\bar{q}(t) = 0.5\ \rho v^2 \leq \bar{q}_{\max} \tag{4.16}$$

Moreover, specific terminal conditions must be satisfied at the end of the entry flight. The range-to-go constraint is given by:

$$S = \arccos\left(\sin\lambda \sin\lambda_f + \cos\lambda \cos\lambda_f \cos\left(\phi - \phi_f\right)\right) \tag{4.17}$$

Here, λ and ϕ are the current coordinates, and

$$\lambda_f = 50^\circ, \quad \phi_f = 0^\circ \tag{4.18}$$

represent the terminal longitude and latitude, respectively, indicating the position of the vehicle projected onto the surface of the Earth. Since this distance is measured in radians, it can be converted to meters using R_0, where the total length of the arc, d, in meters is given by:

$$d = R_0 \cdot S \tag{4.19}$$

Equation 4.19 accounts for the curvature of the Earth.
An additional terminal constraint is imposed on the velocity:

$$v(t_f) = v_f = 1.8 \text{ km/s} \tag{4.20}$$

The target velocity v_f represents the stop condition for the guidance algorithm.
Lastly, the control variables σ and α are subject to the following limitations:

$$|\sigma| \in [0°, 90°] \tag{4.21}$$

$$\alpha \in [5°, 25°] \tag{4.22}$$

4.1.3 Initial descent phase

In the initial phase of re-entry, the vehicle is in free fall, with no control on the trajectory due to two main factors: its high speed and inertia, which make it difficult to achieve quick responses, and the low atmospheric density at high altitudes, which limits the generation of aerodynamic forces on control surfaces. As the vehicle descends and slows down, control becomes more effective. The trajectory is simply calculated through the integration of the equations of motion using the Runge-Kutta method until it reaches a certain height threshold.

To ensure flight safety during this phase, the initial amplitude of the bank angle ($|\sigma_0|$) is determined. This is achieved by iterating through potential bank angles and ensuring that the heating rate ($\dot{Q}(t)$) remains within acceptable limits, specifically not exceeding the maximum allowable heating rate ($\dot{Q}_{\max}$).

Finally, we set an initial final time constraint t_f for this phase, its validity will be assessed in Subsection 4.4.3.

4.2 Compound HV corridor

[11] The trajectory of the vehicle must remain within a specific height-velocity (HV) corridor to accomplish all flight constraints mentioned in Subsection 4.1.2. The compound HV corridor, known for its simplicity and clear physical meaning, provides upper and lower bounds for flight trajectory. To ensure smooth transitions

in the height bounds, these are fitted with fifth-order polynomials, as shown by the red lines in Figure 4.3, where the red dashed line (HCUb) represents the upper bound and the red solid line (HCLb) is the lower bound.

The aircraft entire corridor is composed of the conventional HV corridor, HCUb and HCLb, and the terminal flight envelope, HTUb and HTLb, considering the terminal states.

The corridor defines a feasible flight space. Within this space, the vehicle can glide while adhering to all process constraints, since they have been considered in the design of the corridor itself.

Our objective is to ensure that the vehicle remains within this composite HV corridor throughout the flight. The boundaries have been provided by the laboratory, and their determination is not part of the work conducted in this thesis. However, further details on the construction of this feasible space will be included in Section A.1.

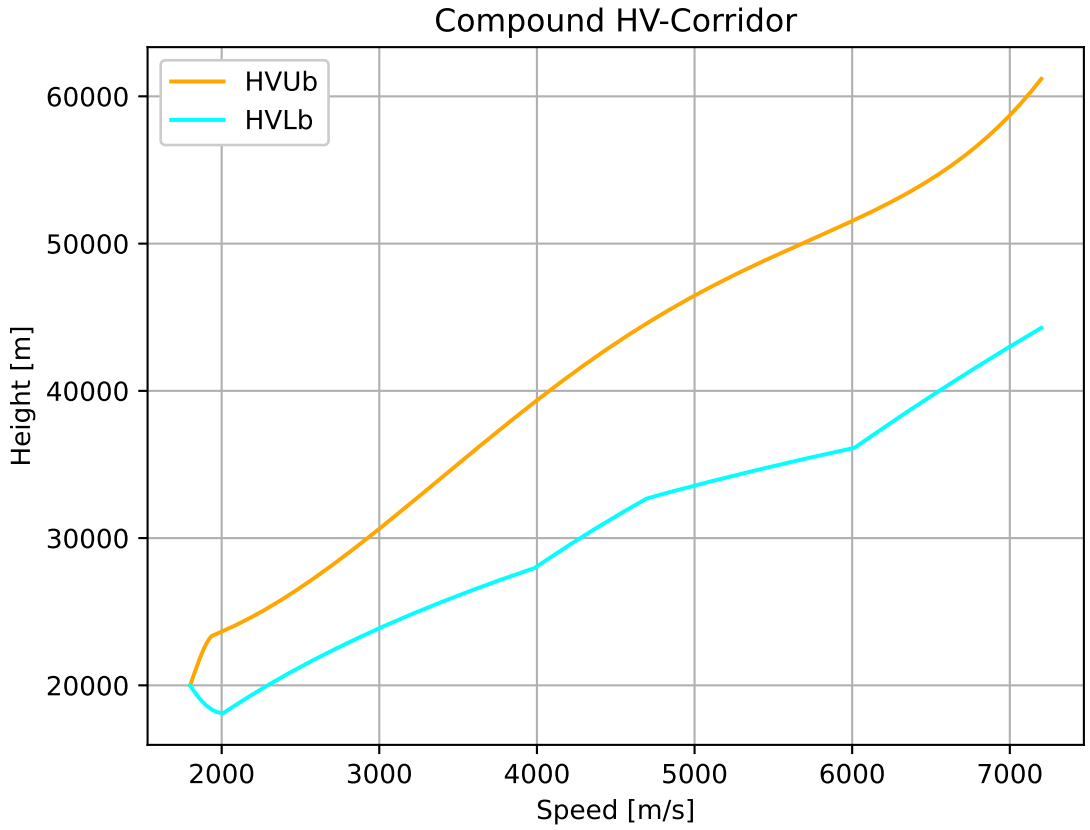

Figure 4.3: Compound HV corridor given by the union of the conventional corridor (HCUb and HCLb) and the terminal flight envelope (HTUb and HTLb), [11]

4.3 Trajectory parametrization and range constraint management

4.3.1 Control parametrization

Relationship between downrange and height

The motion of the aircraft can be described by the following equation of motion, which relates the downrange distance to velocity [2]:

$$\frac{dR_d}{dv} = \frac{\frac{v\cos(\theta)}{r}}{-\frac{D}{m} - g\sin(\theta)} \tag{4.23}$$

For simplification, under the quasi-equilibrium glide condition (QEGC), we assume that $r \approx R_0$ and $\theta = 0$. This allows us to reduce the equation further by incorporating the specific expression for drag (D):

$$\frac{dR_d}{dv} = \frac{m}{-\frac{1}{2}\rho v C_L S_{\text{ref}} R_0} \tag{4.24}$$

The air density ρ is a function of the height, represented by:

$$\rho(h) = \rho_0 e^{-\frac{h}{h_s}} \tag{4.25}$$

where ρ_0 is the air density at sea level, and h_s is the scale height. Since ρ decreases exponentially with altitude, the term $\frac{dR_d}{dv}$ becomes an increasing function of the height.

This implies that, at higher flight altitudes, the vehicle experiences lower drag, enhancing its downrange gliding capability.

Height profile parametrization

A method for generating a feasible height profile is weighting the upper and lower bounds of the composite corridor, as expressed in Equation 4.26:

$$h_{\text{ref}} = \omega_h \cdot H_{\text{Ub}} + (1 - \omega_h) \cdot H_{\text{Lb}} \tag{4.26}$$

From the derivation process of the composite corridor, it is evident that the designed h_{ref} is a function of velocity and consequently, h_{ref} satisfies the flight path constraints. The adjustment of ω_h, that is called weighting coefficient, enables the height profile to rise or fall. By associating this adjustment with the previously discussed relationship between altitude and range, a parameterized scheme for the trajectory profile is established, facilitating the accurate satisfaction of the desired

downrange R_{dgo} through online adjustments of ω_h. Considering the k-th guidance cycle, the Equation 4.26 becomes:

$$h_{\text{ref}}(k+1) = \omega_h(k+1) \cdot H_{\text{Ub}} + (1 - \omega_h(k+1)) \cdot H_{\text{Lb}} \tag{4.27}$$

Determination of ω_h and v_{ref}

An iterative optimization approach is employed to determine the reference velocity v_{ref} and weighting coefficient w_h for trajectory planning. This optimization leverages the deep neural network (DNN) presented in Chapter 3, which predicts the downrange R_d and flight time t based on the inputs: velocity v and weighting coefficient ω_h. The relationship between the changes in the outputs t and R_d and the inputs v and ω_h can be expressed using the *chain rule* as follows:

$$\begin{pmatrix} dt \\ dR_d \end{pmatrix} = \begin{pmatrix} \frac{\partial t}{\partial v} & \frac{\partial t}{\partial \omega_h} \\ \frac{\partial R_d}{\partial v} & \frac{\partial R_d}{\partial \omega_h} \end{pmatrix} \begin{pmatrix} dv \\ d\omega_h \end{pmatrix} \tag{4.28}$$

Given the non-linear nature of the outputs concerning the inputs, a direct determination of v and ω_h is not possible. Instead, an iterative method to refine these parameters through multiple iterations is implemented, analogous to *gradient descent*, where updating the parameters is made in the direction of reducing the error. Appropriate initializations of the variables are made.

The adjustment of the parameters is expressed as:

$$\begin{pmatrix} dv \\ d\omega_h \end{pmatrix} = \begin{pmatrix} \frac{\partial t}{\partial v} & \frac{\partial t}{\partial \omega_h} \\ \frac{\partial R_d}{\partial v} & \frac{\partial R_d}{\partial \omega_h} \end{pmatrix}^{-1} \begin{pmatrix} dt \\ dR_d \end{pmatrix} \tag{4.29}$$

This reflects the principle of *Newton's method*, utilizing the inverse of the Jacobian matrix to update estimates more effectively. The use of the chain rule highlights how small changes in the input parameters v and ω_h affect the outputs t and R_d. Additionally, ω_h is constrained within the range $[0, 1]$.

The iterative process continues until the terminal conditions for both dt (the difference in time) and dR_d (the difference in range) are sufficiently small, indicating convergence towards a solution that satisfies the required conditions for trajectory planning.

Filtering of the reference height

[11] The height profile may change too drastically, causing a large jump in the bank angle tracking instruction. Therefore, the height is filtered before the trajectory tracker, according to a proportional-integral (PI) filtering algorithm, defined as:

$$h_{\text{cmd}}(k) = (1-k_{Pf})h_{\text{cmd}}(k-1)+k_{Pf}h_{\text{ref}}(k)+k_{If}\int(h_{\text{cmd}}(k-1)-h_{\text{ref}}(k-1))dt \quad (4.30)$$

where h^k_{ref} and h^{k-1}_{ref} are the reference heights corresponding to the current and previous sample points, respectively, h^k_{cmd} and h^{k-1}_{cmd} are the height commands after filtering of the current and previous sample points. The parameters k_{Pf} and k_{If} are the proportional and integral filter coefficients, respectively. Their values, generally in the range $[0, 1]$ are determined using a trial and error approach to ensure optimal filtering performance.

The choice of these coefficients is crucial for controlling the responsiveness and stability of the height profile filtering. Higher values of k_{Pf} make the system more reactive to changes in the reference height, while lower values provide a smoother response. Similarly, higher values of k_{If} allow for better error correction over time by integrating past errors, whereas lower values lead to slower adjustments. An appropriate tuning of these parameters is requested.

4.3.2 Range constraint and tracking controller design

[12] The amplitude of the bank angle σ is computed by tracking the height-velocity profile using the *feedback linearization* approach. Since $r = R_0 + h$, it follows that $\dot{r} = \dot{h}$. According to Equation 4.1, we can calculate the second-order derivative of the altitude $\ddot{h}$, and after taking $\cos\sigma$ as the control input u and organizing the equation into linear form, we can approximate $\ddot{h}$ as:

$$\ddot{h} = a + bu \quad (4.31)$$

where

$$a = (\frac{-D}{m} - g\sin\theta)\sin\theta + (\frac{v^2}{r} - g)\cos\theta, \quad b = \frac{L}{m}\cos\theta.$$

A control law is designed to track the altitude h_{cmd}, calculated from the altitude-velocity profile, which is expressed as

$$(\ddot{h} - \ddot{h}_{cmd}) + 2\lambda(\dot{h} - \dot{h}_{cmd}) + \lambda^2(h - h_{cmd}) = 0 \quad (4.32)$$

where λ is a constant coefficient that controls the allowable reaching speed of h_{cmd}. Substituting Equation 4.1 into Equation 4.32, we obtain the amplitude of σ from:

$$\cos\sigma = \frac{1}{b}\left(\ddot{h}_{cmd} - 2\lambda(\dot{h} - \dot{h}_{cmd}) - \lambda^2(h - h_{cmd}) - a\right) \quad (4.33)$$

The right side of the equation is restricted to the domain $[0, 1]$ during the solution process to ensure that the equation is meaningful.

4.4 Time constraint management

4.4.1 Angle of attack as a function of velocity

In Subsection 4.1.1 we presented the angle of attack α as a function of velocity. Below, the specific relationship between the two quantities [11]:

$$\alpha = \begin{cases} \alpha_{\max} & \text{if } v > v_1 \\ \frac{(\alpha_{\max} - \alpha_{\min})(v - v_2)}{(v_1 - v_2)} & \text{if } v_2 \leq v \leq v_1 \\ \alpha_{\min} & \text{if } v < v_2 \end{cases} \tag{4.34}$$

Where:

- $\alpha_{\max} = 20°$ is the maximum angle of attack.
- $\alpha_{\min} = 8.5°$ is the angle of attack for maximum upward resistance: the angle of attack at which an aircraft generates the highest lift-to-drag ratio ($\frac{L}{D}$) during the flight.
- $v_1 = 4700\,\text{m/s}$ and $v_2 = 3100\,\text{m/s}$ are the velocities at the two division points.
- v is the current velocity of the vehicle.

4.4.2 Reference velocity tracking

In the control action for adjusting the angle of attack α, in order to track v_{ref}, a *feedback control* strategy is employed, resembling a PI control approach. The nominal angle of attack $\alpha_{nominal}$ is firstly computed based on the current velocity. Then, the difference between the current velocity and the reference velocity is calculated as follows:

$$\text{error} = v_{\text{now}} - v_{\text{ref}}. \tag{4.35}$$

To account for accumulated errors over time, the integral component is updated, and the adjustment to the angle of attack, denoted as δ_α, is then determined using both the proportional and integral components:

$$\delta_\alpha = k_p \cdot (v_{\text{now}} - v_{\text{ref}}) - k_i \cdot \int (v_{\text{now}} - v_{\text{ref}})\, dt. \tag{4.36}$$

Here, k_p and k_i are the proportional and integral gain coefficients, respectively, again tuning according to a trial and error strategy, and the adjustment is clipped to remain within a specified range of $[-5, 5]$ degrees. Finally, the reference angle of attack is calculated by combining the nominal angle with the correction:

$$\alpha_{\text{ref}} = \alpha_{\text{nominal}} + \delta_{\alpha}. \tag{4.37}$$

This control strategy ensures that the angle of attack is dynamically adjusted based on the difference between the current and reference velocities, taking into account the reference speed, calculated on the basis of predictions made by the DNN, the velocity that the vehicle should have to accomplish the time constraint.

Basically, since there exists a linear relationship between flight time and range, when both the starting point and the end point are set, the variables v and ω_h must be adjusted simultaneously to satisfy the time and range constraints.

4.4.3 Time boundaries

Before the guidance algorithm starts, we need to check if the fixed terminal constraint t_f, is respectable. In particular we must ensure that t_f is within a specific range.

In the optimal scenario, after the initial and terminal points are fixed, the maximum flight time is calculated by determining the necessary velocity, v_{need}, that fulfills the range constraint when $\omega_h = 1$. However, setting $\omega_h = 1$ results in a large change in velocity dv, defined as the difference between v_{ref} (speed that the vehicle should have to meet both the range and time constraints, calculated with the help of the DNN, Equations 4.28 and 4.29) and the current velocity v_{curr} of the vehicle. This great gap dv is really difficult to bridge, regardless of how the angle of attack α is adjusted afterward. Therefore, v_{need} that meets the range constraint is calculated starting from $\omega_h = 1$ and decreasing. Once dv becomes acceptable, the maximum flight time is achieved. In particular, we fix a tolerance of 800.

Conversely, the minimum flight time is determined by incrementing ω_h from 0 upwards.

Chapter 5

Lateral guidance

Lateral guidance is the other component for maintaining the desired trajectory during the atmospheric re-entry phase of hypersonic vehicles. The primary objective is to ensure that the vehicle stays on course by adjusting the lateral control surfaces to minimize deviations from the ideal path. One of the key parameters in lateral guidance is the crossrange (R_c), already defined in Section 2.3.

The rate of change of the crossrange with respect to time, denoted by $\dot{R}_c$, is useful for understanding how quickly the vehicle is deviating laterally.

It is defined by the following equation [2]:

$$\dot{R}_c = \frac{v \cos\theta \sin(\psi - \psi_0)}{r} \tag{5.1}$$

where ψ_0 represents the initial heading angle at the EI.

This equation indicates that the rate of change of the crossrange depends on the heading error, represented by $\sin(\psi - \psi_0)$. If ψ aligns with ψ_0, the rate $\dot{R}_c$ becomes equal to 0, implying that there is no lateral deviation from the initial defined trajectory. Conversely, if there is a significant heading error, the rate will increase, indicating a need for lateral correction.

By continuously monitoring $\dot{R}_c$, the lateral guidance system can dynamically adjust the sign of the bank angle to correct the course of the vehicle and reduce lateral deviations.

In this thesis, the primary focus is the longitudinal guidance, which forms the core of the work. In particular, no dedicated control for managing crossrange or heading angle has been developed. However, a **bank reversal logic** [13] [14] is implemented. This logic is described in detail in the following sections of this chapter.

5.1 Initial sign of the bank angle

The initial sign of the bank angle σ_{flag} is determined by comparing the line of sight (LOS) angle ψ_{LOS} with the terminal heading angle of the vehicle ψ_T. In this context, ψ_{LOS} represents the angular position between the vehicle and the target point (λ_f, ϕ_f), measured from the current position of the vehicle (λ, ϕ):

$$\psi_{LOS} = \arctan\left(\frac{\sin(\lambda_f - \lambda)}{\cos(\phi) \cdot \tan(\phi_f) - \sin(\phi) \cdot \cos(\lambda_f - \lambda)}\right) \tag{5.2}$$

It provides a reference for the direction that the vehicle needs to follow.

This angle is then compared to ψ_T, which is the final heading orientation of the vehicle.

Finally, σ_{flag} is computed based on the difference between these two angles. If ψ_{LOS} is greater than ψ_T, σ_{flag} is set to 1, indicating a positive roll (right bank). Conversely, σ_{flag} is set to -1, indicating a negative roll (left bank). Mathematically, this can be expressed as:

$$\sigma_{flag} = \begin{cases} 1, & \text{if } \psi_{LOS} > \psi_T \\ -1, & \text{otherwise} \end{cases} \tag{5.3}$$

σ_{flag} allows the vehicle to initiate its descent with the correct lateral orientation.

5.2 Bank reversals

In each guidance cycle, σ_{flag} is determined by evaluating the delta angle $\Delta\psi$, which is defined as the difference between the current ψ_{LOS} and ψ. This is accomplished through the function `get_delta_range` (see Subsection A.4.2 for implementation details), which calculates $\Delta\psi$ as:

$$\Delta\psi = \psi_{LOS} - \psi \tag{5.4}$$

After that, the following conditions are evaluated to set σ_{flag}:

$$\sigma_{flag} = \begin{cases} 1, & \text{if } \Delta\psi \geq \frac{1}{57.3} rad \\ -1, & \text{if } \Delta\psi < -\frac{1}{57.3} rad \end{cases} \tag{5.5}$$

This approach allows for adaptive control of the sign of the bank angle, enabling precise adjustments to maintain the desired lateral trajectory throughout each guidance cycle.

Chapter 6

Simulations and results

In this chapter, we present the obtained results from our research, focusing on the achieved trajectory and accuracy in reaching the final target in terms of longitude and latitude. We compare the performance of our algorithm, specifically looking at trajectory update frequency, with data from traditional guidance techniques that do not utilize predictions from a DNN. Furthermore, the discussed results are again based on the CAV-L model, Subsection 4.1.2 and Table A.3, [3], which is an example of re-entry vehicle, employed in the military field.

Figure 6.1 shows the flow chart illustrating the functionality of the algorithm.

6.1 Evaluation of real-time performance and convergence

6.1.1 Entry trajectory

The entry trajectory of the vehicle is shown in Figure 6.2, read from right to left, illustrating the flight within the HV corridor. After the initial descent phase, during which no control actions are applied, the vehicle reaches a height threshold (around 60 km) that marks its entry into the corridor. At this point, the guidance algorithm starts. Throughout the control phase, the vehicle remains well within the corridor limits, except for a brief moment at the beginning of guidance when it exceeds the upper boundary of the corridor (looking at the graph, when the red line crosses the green one, for speeds greater than 6500 m/s). This temporary deviation is acceptable, as the upper limit can be considered a "soft" constraint, derived from approximations in the vehicle dynamics. Conversely, the lower limit serves as a "hard" constraint, because its determination takes path constraints (Subsection 4.1.2) into account and compliance with which is crucial for ensuring the vehicle

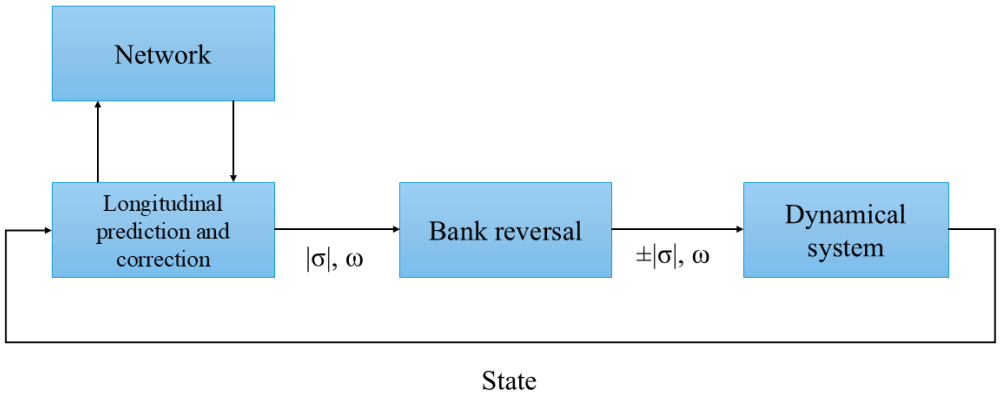

Figure 6.1: Flow chart of the proposed DNN-based entry NPCG algorithm

integrity and safety throughout the mission.

6.1.2 Convergence accuracy

The convergence accuracy of the algorithm is evaluated by comparing the target point with the terminal actual point in terms of longitude and latitude. As shown in Table 6.1, the absolute error between the target and real values is on the order of 10^{-4} for longitude and 10^{-7} for latitude. This indicates that the algorithm achieves high precision in positioning. The extreme accuracy can also be observed in Figures 6.3 and 6.4.

Parameter	Real Value (rad)	Goal (rad)	Absolute Error (rad)
Terminal Longitude	0.8724	0.8727	0.0003
Terminal Latitude	4.3567e-07	0.0	4.3567e-07

Table 6.1: Comparison between real values and goals in radians

6.1.3 Real-time performance

Relying on our results, the DNN-based approach achieved the performance parameters resumed in Table 6.2.

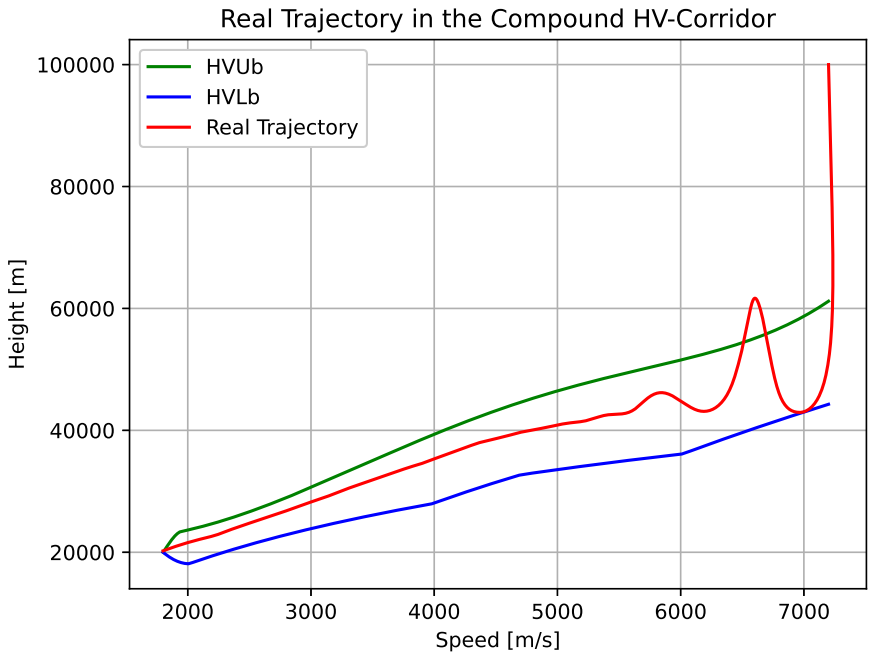

Figure 6.2: Entry trajectory under specific initial conditions based on the CAV-L model

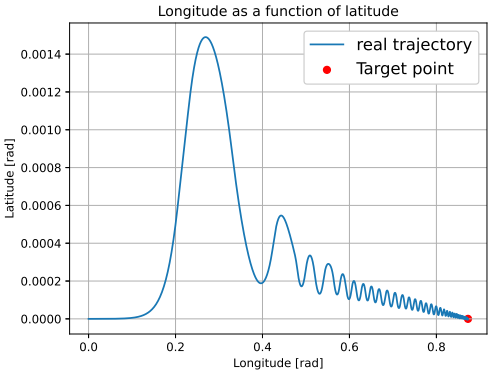

Figure 6.3: Longitude as a function of latitude

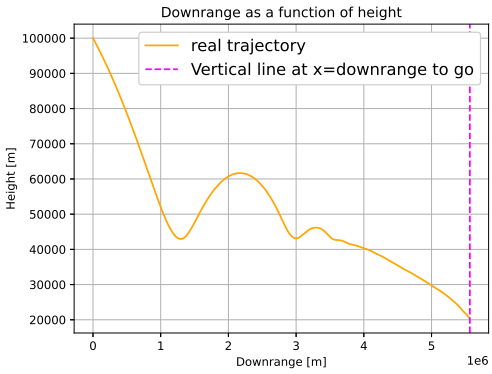

Figure 6.4: Downrange as a function of height

In general, studies demonstrate that the entry trajectory must be corrected/updated every 2–5 seconds, due to rapidly changing flight conditions [2] [4]. This corresponds to a trajectory update frequency of approximately 0.2–0.5 Hz, which traditional trajectory propagation approach can satisfy. However, hypersonic vehicles need a higher update frequency. This necessitates faster trajectory prediction

Parameter	Value
Number of iterations	9,153
Average total execution time	$\sim$ 30.549 seconds
Average time per iteration	$\sim$ 0.003 seconds
Average frequency per iteration	$\sim$ 280 Hz

Table 6.2: Execution statistics of the DNN-based algorithm

algorithms. The frequency of work of our DNN-based approach is significantly larger than the ones cited before, fully meeting the request.

Our DNN-based algorithm achieves a substantial improvement in overall computational speed, providing a more efficient solution for real-time re-entry guidance.

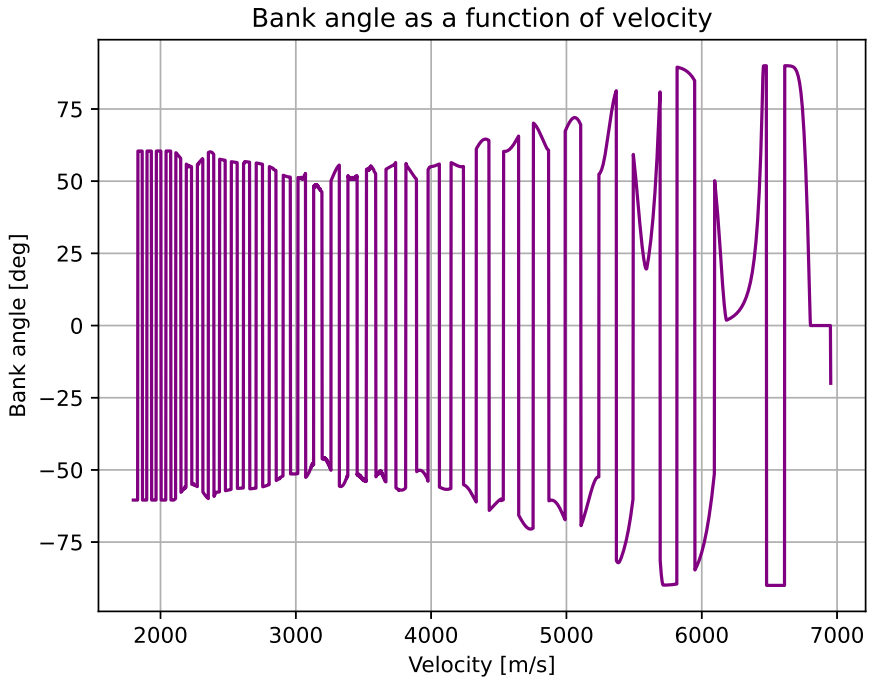

Figure 6.5: Bank reversal history of the considered entry scenario

6.2 Terminal flight phase management

During the terminal phase of flight, achieving the desired terminal constraints becomes increasingly challenging. This difficulty arises because the HV corridor narrows significantly as the vehicle approaches re-entry. The reduction in corridor

width complicates the precision of the DNN to make accurate predictions. In particular, for velocities below 2100 m/s, we chose to cease iterating on velocity (v_{ref}) and weight (ω_h) values that would satisfy the constraints, allowing the vehicle to fall freely. Simulation results indicate that this strategy effectively meets the time and range requirements while preventing abrupt changes in the terminal HV profile. Figures 6.6 and 6.7 demonstrate that the trajectory is closer to the desired path when the vehicle is allowed to be in free-fall during the final phase, as opposed to when control is continuously applied.

The results presented in Table 6.2, specifically the averages of time and frequency per iteration, refer to the flight phase in which the DNN is employed.

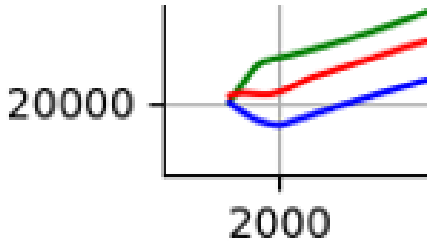

Figure 6.6: Terminal trajectory phase with DNN based control

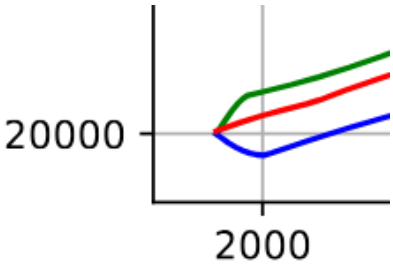

Figure 6.7: Terminal trajectory phase without control

Chapter 7

Conclusion

In this thesis, an intelligent re-entry guidance algorithm is developed, combining a deep neural network (DNN) with a Numerical Predictor-Corrector Guidance (NPCG) approach. The DNN is designed to predict key flight parameters such as flight time and downrange, effectively supporting the longitudinal and lateral trajectory correction. This integration allows the algorithm to meet the demanding multi-constraint requirements of the hypersonic vehicle re-entry problem, including path and terminal constraints, while ensuring real-time performance. The conducted simulations demonstrate the accuracy and efficiency of the proposed algorithm, confirming its potential to improve the safety and precision of atmospheric re-entry trajectories.

The presented research addresses an increasingly relevant topic in today's technological landscape, where artificial intelligence (AI) is rapidly advancing across many fields. By incorporating AI techniques such as deep learning into the guidance algorithm, we contribute to the growing body of knowledge exploring new methods to improve aerospace technologies. The success of this approach shows that deep learning can be a powerful tool in handling complex, non-linear problems that traditional algorithms often struggle to manage.

Looking ahead, the use of DNNs in guidance algorithms opens exciting possibilities for future research and applications. As AI continues to evolve, further advancements in DNN architectures and training methodologies may lead to even more robust and adaptive guidance systems, capable of managing increasingly complex mission scenarios in hypersonic and space vehicles. Additionally, integrating AI with other modern technologies such as edge computing or distributed systems could allow for even faster and more efficient real-time solutions.

Beyond the academic contributions, conducting this research at Beihang University has been a valuable personal and cultural experience. Working in a laboratory in a country different from my own provided the opportunity to get in touch with various academic perspectives and observe different approaches to research and problem-solving. This experience has not only enriched my technical knowledge but has also broadened my understanding of international collaboration in scientific fields.

Appendix A

Appendix A

A.1 Construction of the conventional HV corridor [11]

The HV corridor is a commonly used concept in atmospheric guidance for re-entry. It offers the advantage of being simple to calculate and based on clear physical principles, with straightforward practical applications.

The resolution for determining the HV corridor is based on the following equations:

$$\rho(h) = \rho_0 e^{-h/h_s}. \tag{A.1}$$

This is the equation for air density at a given height h, where ρ_0 is the density at sea level and h_s is a parameter related to the atmospheric scale height.

The equations for lift (L) and drag (D) are given by:

$$L = 0.5\rho(h)v^2 C_L S_{ref} \tag{A.2}$$

$$D = 0.5\rho(h)v^2 C_D S_{ref} \tag{A.3}$$

where v is the speed of the vehicle, S_{ref} is the aerodynamic reference area, and C_L and C_D are the lift and drag coefficients.

A.1.1 Lower boundary

Substituting A.1-A.3 into 4.13-4.16, we get the height constraints at a specific velocity v:

$$h > \frac{2}{h_s} \ln \left(\frac{k_Q}{\dot{Q}_{\max}} \left(\frac{v}{\sqrt{R_0 g_0}} \right)^{3.15} \right) = H_{\dot{Q},\max}(v) \tag{A.4}$$

$$h > \frac{1}{h_s} \ln \left(\frac{\rho_0 v^2 S \sqrt{C_D^2 + C_L^2}}{2 n_{\max} m_{g0}} \right) = H_{n,\max}(v) \tag{A.5}$$

$$h > \frac{1}{h_s} \ln \left(\frac{\rho_0 v^2}{2 q_{\max}} \right) = H_{q,\max}(v). \tag{A.6}$$

These equations determine the lower altitude limit for a given speed, considering the path constraints.

A.1.2 Upper boundary

The upper bound is determined by the quasi-equilibrium glide condition (QEGC), which assumes that the flight path angle θ and its rate of change $\dot{\theta}$ are approximately zero. This simplifies the flight dynamics by ignoring variations in the flight path angle. Under these assumptions, and neglecting the rotation of the Earth, we can write:

$$\frac{L \cos \sigma}{m} + \left(\frac{v^2}{r} - g \right) = 0 \tag{A.7}$$

where σ is the bank angle, m is the mass of the vehicle, r is the radial distance from the center of the Earth, and g is the gravitational acceleration.

Substituting A.1 and A.2 into A.7, we obtain the upper bound equation:

$$h \leq -h_s \ln \left(\frac{2m(g - \frac{v^2}{r})}{\rho_0 v^2 S_{ref} C_L \cos \sigma} \right) \tag{A.8}$$

A.1.3 Final conventional corridor

Among them, $H_{\dot{Q},\max}(v)$, $H_{n,\max}(v)$, $H_{q,\max}(v)$, and $H_{QEGC}(v)$ correspond to the thermal peak, normal overload, dynamic pressure, and quasi-equilibrium gliding altitude boundaries, respectively. The $H_{QEGC}(v)$ has no analytical solution and needs to be solved iteratively using the secant method, for example.

Finally, we can derive the form of the HV corridor:

$$H_{\text{up}}(v) = H_{QEGC}(v) \tag{A.9}$$

$$H_{\text{down}}(v) = \max\left(H_{\dot{Q},\max}(v), H_{n,\max}(v), H_{q,\max}(v)\right) \tag{A.10}$$

where $H_{\text{up}}(v)$ represents the upper boundary of the corridor, and $H_{\text{down}}(v)$ corresponds to the lower boundary of the corridor.

A.2 CAV-L aerodynamic data [11]

For the aerodynamic data, in this thesis, we use the CAV-L configuration, designed by Boeing in 1998, which is characterized by relatively low lift-to-drag ratios. The aerodynamic data for the lift coefficient (C_L) and drag coefficient (C_D) are shown in Tables A.1 and A.2.

Table A.1: CAV lift coefficient data

Alpha	**Mach 3.5**	**Mach 5**	**Mach 8**	**Mach 15**	**Mach 20**	**Mach 23**
10°	0.3401	0.3264	0.3108	0.2856	1.2760	0.2739
15°	0.5786	0.5358	0.4883	0.4491	0.4349	0.4319
20°	0.7975	0.7291	0.6731	0.6137	0.5975	0.5966

Table A.2: CAV drag coefficient data

Alpha	**Mach 3.5**	**Mach 5**	**Mach 8**	**Mach 15**	**Mach 20**	**Mach 23**
10°	0.1838	0.1483	0.1295	0.1226	0.1210	0.1217
15°	0.2691	0.2505	0.2308	0.2178	0.2150	0.2159
20°	0.4197	0.3861	0.3599	0.3388	0.3370	0.3409

It can be seen that the aerodynamic parameters are two-dimensional functions of both the angle of attack (α) and the Mach number (M. However, during actual computations, performing two-dimensional interpolation is computationally slow. Therefore, in this thesis, we fit the aerodynamic parameters into a functional form dependent on α and M, expressed as:

$$\begin{aligned} C_L(\alpha, M) = {} & C_{L00} + C_{L10}\alpha + C_{L01}M + C_{L20}\alpha^2 + C_{L11}\alpha M + C_{L02}M^2 \\ & + C_{L21}\alpha^2 M + C_{L12}\alpha M^2 + C_{L03}M^3 + C_{L22}\alpha^2 M^2 \\ & + C_{L13}\alpha M^3 + C_{L04}M^4 \end{aligned} \tag{A.11}$$

$$C_D(\alpha, M) = C_{D00} + C_{D10}\alpha + C_{D01}M + C_{D20}\alpha^2 + C_{D11}\alpha M + C_{D02}M^2 + C_{D21}\alpha^2 M + C_{D12}\alpha M^2 + C_{D03}M^3 + C_{D22}\alpha^2 M^2 + C_{D13}\alpha M^3 + C_{D04}M^4 \quad \text{(A.12)}$$

where C_{Lij} and C_{Dij} are the fitted coefficients of the functions (referring to A.1 and A.2.

A.3 CAV-L overall parameters [11]

Table A.3: CAV-L aircraft overall parameters and constraints

Symbol	Value
m_0	907 (kg)
S_{ref}	0.35 (m^2)
C_1 (heat flow coefficient)	11030
R_d (stationary point curvature)	0.1
$\dot{Q}_{max}$	1200 (kW/m^2)
n_{max}	4 (g_0)
q_{max}	200 (kPa)

With $k_Q = \frac{C1}{\sqrt{R_d}}$ as vehicle-dependent constant.

A.4 A look into the code

The lines of code below, are snippet of the entire program used for the simulations and represent portions of the algorithm worth mentioning.

A.4.1 Tracking controller design and amplitude of the bank angle

The following script calculates the derivatives of the height command h_{cmd}, employed in the tracking controller design, as reported in Subsection 4.3.2, Equation 4.32.

```
1 def get_h_differential(self, h_ref, h_ref_old, h_cmd, delta_t,
    v_curr):
2
```

```
    kp = 0.01 # proportional coefficient
    ki = 1e-4 # integral coefficient

    h_cmd[0] = h_cmd[1]
    h_cmd[1] = h_cmd[2]

    self.h_sum += -ki * (h_cmd[1] - h_ref_old) # integral term
    h_cmd[2] = self.h_sum + (1 - kp) * h_cmd[1] + kp * h_ref # filtering of h_ref

    h_cmd_dot = (h_cmd[2] - h_cmd[1]) / delta_t # first derivative of h_cmd
    h_cmd_dot2 = (h_cmd[2] + h_cmd[0] - 2 * h_cmd[1]) / (delta_t ** 2) # second derivative of h_cmd

    h_cmd_diff = [h_cmd[2], h_cmd_dot, h_cmd_dot2]
    return h_cmd, h_cmd_diff
```

Listing A.1: Calculation of the derivatives of h_{cmd}

The function `get_tht` determines the amplitude of the bank angle σ according to Equation 4.33.

```
def get_tht(self, state_3, h_cmd_diff):

    lamda = 0.1 # constant coefficient
    r = state_3[0] # states extraction
    v = state_3[3]
    theta = state_3[4]
    chi = state_3[5]
    downrange = state_3[6]
    t = state_3[7]

    h = r - 6378000
    alpha = self.Cav.v2alpha(v)
    rho = self.Earth.h2rho(h)
    g = self.Earth.h2g(h)
    cl = self.AeroForce.get_cy(v, alpha)
    cd = self.AeroForce.get_cx(v, alpha, h)
    L = 0.5 * rho * v ** 2 * cl * self.Cav.S
    D = 0.5 * rho * v ** 2 * cd * self.Cav.S

    h_dot = v * math.sin(theta)
    v_dot = -D / self.Cav.m0 - g * np.sin(theta)

    delta_h = h - h_cmd_diff[0]
    delta_h_dot = h_dot - h_cmd_diff[1]

```

```
26      h_dot2 = h_cmd_diff[2] - 2 * lamda * delta_h_dot - lamda ** 2
    * delta_h
27
28      costht = ((h_dot2 - v_dot * math.sin(theta)) / math.cos(theta)
        + (g - v ** 2 / r) * math.cos(theta)) * self.Cav.m0 / L
29
30      costht = np.clip(costht, 0, 1)
31      tht = math.acos(costht)
32
33      return tht
```

Listing A.2: Amplitude of the bank angle calculation

A.4.2 Downrange and LOS angle

The script below compute the downrange and the LOS angle, given the current longitude and latitude, and terminal ones.

```
1
2       def get_range_angle(self, lamda, phi, lamda_f, phi_f):
3
4           # normalization of the coordinate (initial and final
    longitude) between [-pi, pi]
5           while True:
6               if lamda > math.pi:
7                   lamda = lamda - 2 * math.pi
8               elif lamda < -math.pi:
9                   lamda = lamda + 2 * math.pi
10              else:
11                  break
12
13          while True:
14              if lamda_f > math.pi:
15                  lamda_f = gamma_c - 2 * math.pi
16              elif lamda_f < -math.pi:
17                  lamda_f = lamda_f + 2 * math.pi
18              else:
19                  break
20
21          # Inverse Haversine Formula (from the coordinates of two
    points to the distance of them)
22          ad = 2 * np.cos(phi) * np.sin((lamda_f - lamda) / 2)
23          ec = 2 * np.cos(phi_f) * np.sin((lamda_f - lamda) / 2)
24          cd = 2 * np.sin((phi_f - phi) / 2)
25          ac = np.sqrt(ad * ec + cd ** 2)
26          rrange = 2 * np.arcsin(ac / 2) * self.Earth.R0
27
```

```
        # angle (for the moment I am not interested in the angle)
        if (lamda_f - lamda) > math.pi:
            lamda_f = lamda_f - 2 * math.pi
        if (lamda_f - lamda) < -math.pi:
            lamda_f = lamda_f + 2 * math.pi
        angle = np.arctan(np.sin(lamda_f - lamda) / (
                np.cos(phi) * np.tan(phi_f) - np.sin(phi) * np.cos
    (lamda_f - lamda)))
        if angle == 0:
            if phi_f < phi:
                angle = math.pi
        if (angle < 0) and (lamda_f > lamda):
            angle = angle + np.pi

        if (angle > 0) and (lamda_f < lamda):
            angle = angle - np.pi

        return rrange, angle
```

Listing A.3: Downrange and LOS angle calculation

Instead, the following function determine the difference between the LOS angle and the current heading angle for bank reversal.

```

        def get_delta_range(self, state):

        lamda_curr = state[1]
        phi_curr = state[2]
        chi = state[5]

        lamda_f = self.Cav.lamdaT # terminal longitude
        phi_f = self.Cav.phiT # terminal latitude

        # angle (chi_LOS)
        angle = np.arctan(np.sin(lamda_f - lamda_curr) / (
                    np.cos(phi_curr) * np.tan(phi_f) - np.sin(
    phi_curr) * np.cos(lamda_f - lamda_curr)))

        if (angle < 0) and (lamda_f > lamda_curr):
            angle = angle + np.pi

        if (angle > 0) and (lamda_f < lamda_curr):
            angle = angle - np.pi

        if chi > np.pi/2:
            if angle < -np.pi/2:
                angle = angle + 2*np.pi/2

```

```
25          delta_angle = angle - chi
26
27          return delta_angle
```

Listing A.4: Difference beetween the LOS angle and the current heading angle determination

Bibliography

[1] Yibo DING, Xiaokui YUE, Guangshan CHEN, and Jiashun SI. «Review of control and guidance technology on hypersonic vehicle». In: *Chinese Journal of Aeronautics* 35.7 (2022), pp. 1–18. ISSN: 1000-9361. DOI: https://doi.org/10.1016/j.cja.2021.10.037. URL: https://www.sciencedirect.com/science/article/pii/S1000936121004167 (cit. on p. 1).

[2] Lin Cheng, Fanghua Jiang, Zhenbo Wang, and Junfeng Li. «Multiconstrained Real-Time Entry Guidance Using Deep Neural Networks». In: *IEEE Transactions on Aerospace and Electronic Systems* 57.1 (2021), pp. 325–340. DOI: 10.1109/TAES.2020.3015321 (cit. on pp. 2, 8, 15, 18, 20, 33, 38, 42).

[3] Terry H. Phillips. «A Common Aero Vehicle (CAV) Model, Description, and Employment Guide». In: (Jan. 2003) (cit. on pp. 3, 30, 40).

[4] Rafael Lugo, Richard Powell, and Alicia Dwyer-Cianciolo. «Overview of a Generalized Numerical Predictor-Corrector Targeting Guidance with Application to Human-Scale Mars Entry, Descent, and Landing». In: Jan. 2020. DOI: 10.2514/6.2020-0846 (cit. on pp. 6–9, 42).

[5] Yann LeCun, Y. Bengio, and Geoffrey Hinton. «Deep Learning». In: *Nature* 521 (May 2015), pp. 436–44. DOI: 10.1038/nature14539 (cit. on p. 9).

[6] Jürgen Schmidhuber. «Deep learning in neural networks: An overview». In: *Neural Networks* 61 (2015), pp. 85–117. ISSN: 0893-6080. DOI: https://doi.org/10.1016/j.neunet.2014.09.003. URL: https://www.sciencedirect.com/science/article/pii/S0893608014002135 (cit. on p. 9).

[7] Shiv Ram Dubey, Satish Kumar Singh, and Bidyut Baran Chaudhuri. «Activation functions in deep learning: A comprehensive survey and benchmark». In: *Neurocomputing* 503 (2022), pp. 92–108. ISSN: 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2022.06.111. URL: https://www.sciencedirect.com/science/article/pii/S0925231222008426 (cit. on p. 10).

[8] *PyTorch documentation*. URL: https://pytorch.org/docs/stable/index.html (cit. on p. 24).

[9] aviationfile. *What is Bank Angle?* 2023. URL: https://www.aviationfile.com/what-is-bank-angle/ (cit. on pp. 27, 28).

[10] SKYbrary. *Angle of Attack (AOA)*. URL: https://skybrary.aero/articles/angle-attack-aoa (cit. on pp. 27, 28).

[11] Lin Cheng. «Study on Real-Time Optimal Closed-Loop Reentry Guidance Technology for Hypersonic Vehicles». Supervisor: Prof. Bai Chenggang. PhD dissertation. Beijing, China: Beihang University, 2023 (cit. on pp. 31, 32, 34, 36, 47, 49, 50).

[12] Haoning Wang, Jie Guo, Xiao Wang, Xiang Li, and Shengjing Tang. «Time-coordination entry guidance using a range-determined strategy». In: *Aerospace Science and Technology* 129 (2022), p. 107842. ISSN: 1270-9638. DOI: https://doi.org/10.1016/j.ast.2022.107842. URL: https://www.sciencedirect.com/science/article/pii/S1270963822005168 (cit. on p. 35).

[13] Peng Shi, Jingjing Xu, Lin Cheng, Changhong Dong, and Xu Huang. «Real-Time Lateral Predictor-Corrector Entry Guidance With Terminal Heading Angle Constraint». In: *IEEE Transactions on Aerospace and Electronic Systems* PP (Jan. 2024), pp. 1–13. DOI: 10.1109/TAES.2024.3466125 (cit. on p. 38).

[14] Kelly M. Smith. «Predictive Lateral Logic for Numerical Entry Guidance Algorithms». In: (Feb. 2016). URL: https://ntrs.nasa.gov/api/citations/20160001182/downloads/20160001182.pdf (cit. on p. 38).