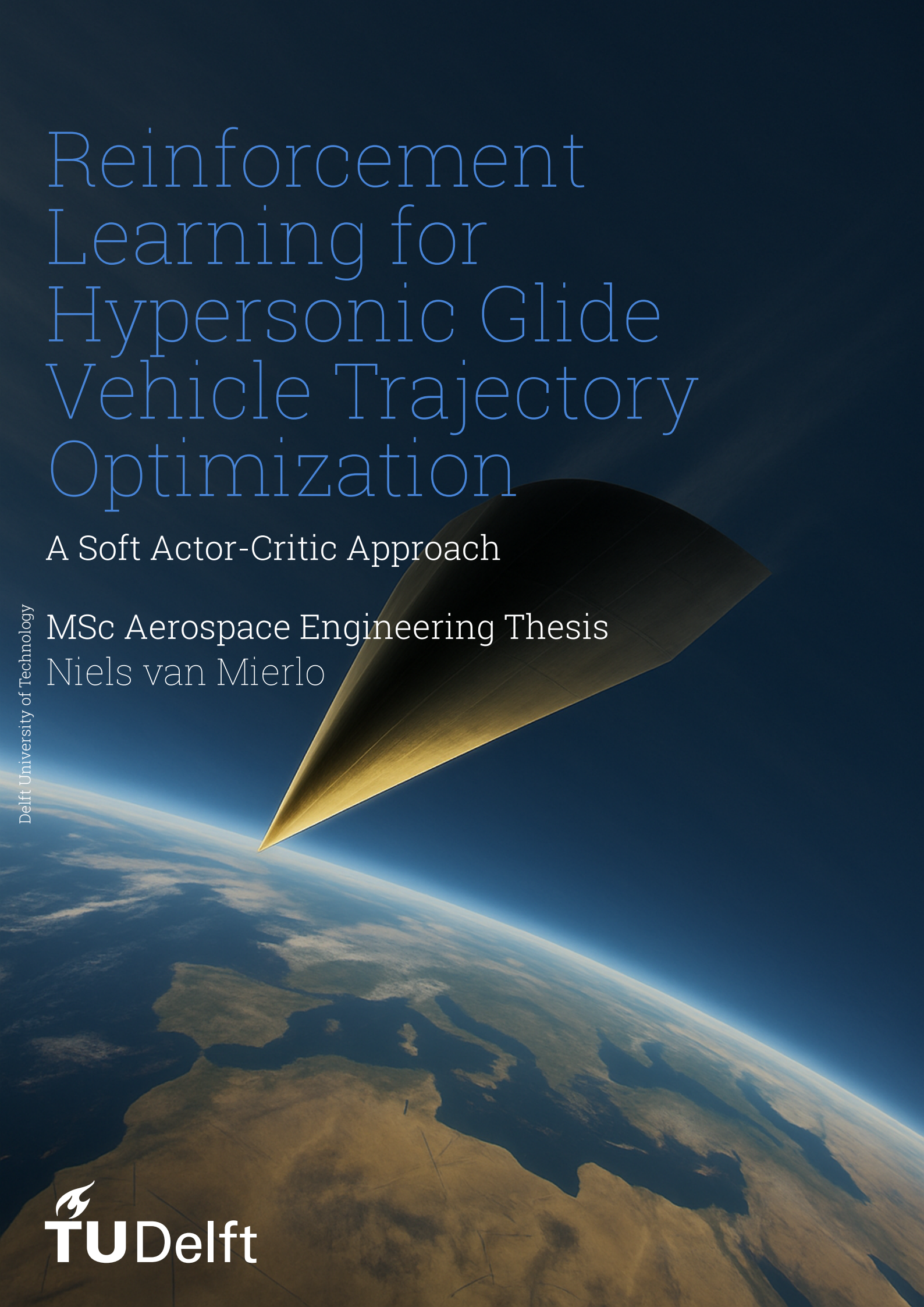
Reinforcement Learning for Hypersonic Glide Vehicle Trajectory Optimization

A Soft Actor-Critic Approach

MSc Aerospace Engineering Thesis

Niels van Mierlo

Delft University of Technology



TU Delft

Reinforcement Learning for Hypersonic Glide Vehicle Trajectory Optimization

A Soft Actor-Critic Approach

by

Niels van Mierlo

To obtain the degree of Master of Science
at Delft University of Technology
to be defended publicly on Tuesday, December 9th, 2025

Supervisor TU Delft:	ir. M.C. Naeije
Supervisor NLR:	ir. M.J. Verveld
External Examiner:	Dr. ir. E. Mooij
Chair:	Dr. ir. W. van der Wal
Project Duration:	March, 2025 – December, 2025
Faculty:	Faculty of Aerospace Engineering, Delft

Cover: Impression of a Hypersonic Glide Vehicle gliding through the atmosphere (Image credit: ChatGPT)

TU Delft

nlr

Preface

I would like to express my sincere gratitude to my supervisor at TU Delft, Marc Naeije, for his supervision throughout this research. I am particularly grateful for the freedom he gave me to explore different approaches and for his continued concern for my well-being in our meetings. I have enjoyed our many meetings and your help throughout this project has been really appreciated.

I am equally grateful to my supervisor at NLR, Mark Verveld, for his supervision and support during this project. I am grateful that he gave me the opportunity to conduct this research at NLR, which I have greatly enjoyed. I am thankful for the weekly meetings we have had to discuss the progress and for the freedom he gave me to explore this research direction. I have really enjoyed working together with you.

I could not have asked for a better team of supervisors.

This thesis marks the end of my 7 years at the Aerospace Engineering faculty at the TU Delft, which has been a journey that started long before I arrived here. This work is dedicated to my father, Emile van Mierlo, who has been fundamental in shaping me into the engineer I am today. From countless hours in high school helping me grasp the fundamentals of mathematics and physics, to supporting me throughout the challenges I faced in my studies at TU Delft, his guidance has always been there. Our many discussions and brainstorming sessions over the years have not only enriched this research but have made this entire journey possible. This thesis would not have been possible without you. I am equally thankful for my mother, Jacqueline van Mierlo, who has had to listen to all these discussions over the years. She has always supported me and I am forever grateful for that.

Niels van Mierlo
Amsterdam, November 2025

Abstract

Hypersonic glide vehicle trajectory optimization requires generating complete reachable footprints for mission planning under strict path constraints. Current methods face limitations including equilibrium glide assumptions, fixed angle of attack profiles and incomplete footprint coverage requiring reoptimization for each target direction. This work presents the first reinforcement learning approach for complete global footprint generation with dual control authority (bank angle and angle of attack) on a rotating spherical Earth model including direct target point guidance and no-fly zone avoidance. The trained policy generates footprints for any location on Earth while incorporating full three-degree-of-freedom dynamics including Coriolis effects and control rate limitations. All trajectories satisfy operational constraints including dynamic pressure, g-load and temperature limits. The policy learns purely from the objective to maximize range in every direction without pre-designed control profiles. Based on the Soft Actor-Critic algorithm, optimal control strategies are learned directly from environment interaction. Large-scale Monte Carlo validation with 100 million randomly sampled trajectories confirms the learned policy discovered the boundaries of the reachable domain, with the reinforcement learning footprint exceeding the Monte Carlo footprint by 5.1% in footprint area. The framework provides precision target guidance to arbitrary global coordinates and allows for no-fly zone avoidance.

Contents

Nomenclature

Abbreviations

ACC	Advanced Carbon/Carbon
AHFV	Airbreathing Hypersonic Flight Vehicle
AoA	Angle of Attack
API	Application Programming Interface
ARD	Atmospheric Reentry Demonstrator
ASA	Adaptive Simulated Annealing
CAV	Common Aero Vehicle
CFD	Computational Fluid Dynamics
CMC	Ceramic Matrix Composite
COESA	Committee on Extension to the Standard Atmosphere
CPU	Central Processing Unit
CSV	Comma-Separated Values
CUDA	Compute Unified Device Architecture
DAE	Differential Algebraic Equations
DARPA	Defense Advanced Research Projects Agency
DCM	Direction Cosine Matrix
DF-ZF	Dong Feng - Zai Fei
DOF	Degree of Freedom
ECEF	Earth-Centered-Earth-Fixed
ESA	European Space Agency
FEI	Flexible External Insulation
FOSTRAD	Free Open Source Tool for Re-entry of Asteroids and Debris
GA	Genetic Algorithm
GPM	Gauss Pseudospectral Method
GPOPS	General Purpose Optimal Control Software
GPU	Graphics Processing Unit
HGV	Hypersonic Glide Vehicle
HTV-2	Hypersonic Technology Vehicle 2
IPOPT	Interior Point OPTimizer
IUGG	International Union of Geodesy and Geophysics
JSON	JavaScript Object Notation
LG	Legendre-Gauss
LLA	Latitude-Longitude-Altitude
MDP	Markov Decision Process
MEX	MATLAB Executable

NASA	National Aeronautics and Space Administration
NED	North-East-Down
NFZ	No-Fly Zone
NLP	Non-Linear Programming
NLR	Netherlands Aerospace Centre
NPSOL	Nonlinear Programming SOLver
ODE	Ordinary Differential Equation
OpenMP	Open Multi-Processing
PICA	Phenolic Impregnated Carbon Ablators
PSO	Particle Swarm Optimization
QEGC	Quasi Equilibrium Glide Condition
RAND	Research and Development Corporation
ReLU	Rectified Linear Unit
RL	Reinforcement Learning
RLV	Reusable Launch Vehicle
SAC	Soft Actor-Critic
SNOPT	Sparse Nonlinear OPTimizer
SOCP	Second-Order Cone Programming
SQP	Sequential Quadratic Programming
TPS	Thermal Protection System
UAV	Unmanned Aerial Vehicle
UHTC	Ultra-High Temperature Ceramics
USSA	US Standard Atmosphere
WGS	World Geodetic System

Symbols

Latin Symbols

a	Intermediate value in haversine formula	$[-]$
a_x, a_y, a_z	Acceleration components in ECEF	[m/s^2]
b_L, b_U	Lower and upper boundary constraints	$[-]$
c	Central angle separating two points	[rad]
c_1	Chapman heat flux coefficient	$[-]$
C_D	Drag coefficient	$[-]$
C_L	Lift coefficient	$[-]$
D	Drag force	[N]
d	Distance	[m]
$d_{\max}$	Maximum possible range	[m]
d_t	Distance to target at time t	[m]
d_{13}	Angular distance between points 1 and 3	[rad]
d_{norm}	Normalized distance to target	$[-]$

dx_t	Cross-track distance	[m]
E	Total specific mechanical energy	[m^2/s^2]
E_h	Energy height	[m]
f	System dynamics function	[$-$]
F	Force vector	[N]
g	Gravitational acceleration	[m/s^2]
g	Path constraint function	[$-$]
g_0	Standard gravity at sea level	[m/s^2]
g_L, g_U	Lower and upper path constraints	[$-$]
$g_{\max}$	Maximum allowable g-load	[$-$]
h	Altitude above Earth's surface	[m]
$h_{\min}$	Minimum altitude	[m]
$h_{\max}$	Maximum expected altitude	[m]
h_{norm}	Normalized altitude	[$-$]
H	Entropy	[$-$]
H_s	Atmospheric scale height	[m]
J	Cost function	[$-$]
k	Episode index	[$-$]
k_{steep}	Steepness parameter (penalty function)	[$-$]
$K_{\max}$	Total number of training episodes	[$-$]
L	Lift force	[N]
L	Running cost	[$-$]
$\mathcal{L}$	Loss function	[$-$]
m	Vehicle mass	[kg]
$\hat{m}$	Normalized margin	[$-$]
m_c	Chapman velocity exponent	[$-$]
m_t	Constraint margin at time t	[$-$]
M	Mach number	[$-$]
n	Chapman density exponent	[$-$]
n	Load factor (g-load)	[$-$]
$n_{\max}$	Maximum allowable load factor	[$-$]
P	Pressure	[Pa]
P	Constraint penalty function	[$-$]
q	Dynamic pressure	[Pa]
$q_{\max}$	Maximum dynamic pressure	[Pa]
$\dot{q}_{\text{conv}}$	Convective heat flux	[W/m^2]
Q	Q-value function	[$-$]
Q_{ϕ_1}, Q_{ϕ_2}	Critic networks with parameters ϕ_1, ϕ_2	[$-$]
r	Radial distance from Earth's center	[m]
r	Reward	[$-$]
r_t	Reward at time t	[$-$]
R	Return (cumulative reward)	[$-$]
R_E	Earth's radius	[m]
R_f	Range	[m]
R_N	Nose radius	[m]

s	Downrange distance	[m]
s	State	[−]
s_t	State at time t	[−]
S_{ref}	Reference area	[m^2]
t	Time	[s]
t_0	Initial time	[s]
t_f	Final time	[s]
T	Temperature	[K]
T	Episode length	[−]
$T_{\max}$	Maximum temperature	[K]
T_{stag}	Stagnation point temperature	[K]
T_w	Wall temperature	[K]
u	Control vector	[−]
v	Velocity vector	[m/s]
v_x, v_y, v_z	Velocity components in ECEF	[m/s]
v_{norm}	Normalized velocity	[−]
V	Velocity magnitude	[m/s]
V_c	Circular orbital velocity	[m/s]
V_{eq}	Equilibrium glide speed	[m/s]
V_{gc}	Maximum load factor speed	[m/s]
V_{qc}	Maximum heat flux speed	[m/s]
W	Weight	[N]
x	State vector	[−]
x_0	Initial state	[−]
x_f	Final state	[−]
x, y, z	Position coordinates in ECEF	[m]
$\mathbf{a}$	Acceleration vector	[m/s^2]
$\mathbf{a}_{\text{aero}}$	Aerodynamic acceleration	[m/s^2]
$\mathbf{a}_{\text{centrifugal}}$	Centrifugal acceleration	[m/s^2]
$\mathbf{a}_{\text{Coriolis}}$	Coriolis acceleration	[m/s^2]
$\mathbf{a}_{\text{grav}}$	Gravitational acceleration	[m/s^2]
$\mathbf{C}$	Direction cosine matrix	[−]
$\mathbf{F}_{\text{aero}}$	Aerodynamic force vector	[N]
$\mathbf{r}$	Position vector	[m]
$\mathbf{v}$	Velocity vector	[m/s]
$\mathbf{V}_{\text{ECEF}}$	Velocity in ECEF frame	[m/s]
$\mathbf{V}_{\text{NED}}$	Velocity in NED frame	[m/s]
$\mathbf{X}_a$	Unit vector along velocity (aerodynamic frame)	[−]
$\mathbf{Y}_a$	Perpendicular vector (aerodynamic frame)	[−]
$\mathbf{Z}_a$	Vertical vector (aerodynamic frame)	[−]
$\mathbf{Z}_e$	Unit vector toward Earth's center	[−]
$\mathbf{y}$	State vector in ECEF coordinates	[−]

Greek Symbols

α	Angle of attack	[°]
α	Temperature parameter	[−]
β	Bearing angle	[rad] or [°]
β_{12}	Bearing from point 1 to point 2	[rad] or [°]
β_{13}	Bearing from point 1 to point 3	[rad] or [°]
γ	Flight path angle	[rad] or [°]
γ	Discount factor	[−]
γ_0	Initial flight path angle	[rad] or [°]
γ_{norm}	Normalized flight path angle	[−]
δ	Angular distance	[rad]
$\Delta\alpha_{\max}$	Maximum change in angle of attack	[°]
$\Delta\sigma_{\max}$	Maximum change in bank angle	[°]
Δt	Time step	[s]
$\Delta\phi$	Latitude difference	[rad] or [°]
$\Delta\lambda$	Longitude difference	[rad] or [°]
ΔE_h	Energy height difference	[m]
$\dot{\alpha}_{\max}$	Maximum rate of change of angle of attack	[°/s]
$\dot{\sigma}_{\max}$	Maximum rate of change of bank angle	[°/s]
ϵ	Random noise (standard normal distribution)	[−]
ϵ	Emissivity of surface	[−]
η	Efficiency ratio ($\Delta R/\Delta E_h$)	[−]
θ	Neural network parameters (actor)	[−]
κ	Difference bearing and heading ($\beta - \psi$)	[rad] or [°]
κ_{norm}	Normalized difference bearing and heading	[−]
λ	Longitude	[rad] or [°]
λ_k	Active penalty factor at episode k	[−]
$\lambda_{\max}$	Maximum penalty magnitude	[−]
λ_c	Circle center longitude	[rad] or [°]
λ_m	Midpoint longitude	[rad] or [°]
μ	Earth's gravitational parameter	[m^3/s^2]
μ_θ	Mean of policy distribution	[−]
π	Policy	[−]
π_θ	Policy parameterized by θ	[−]
ρ	Atmospheric density	[kg/m^3]
ρ_0	Sea-level atmospheric density	[kg/m^3]
σ	Bank angle	[rad] or [°]
σ	Stefan-Boltzmann constant	[$W/(m^2K^4)$]
σ_θ	Standard deviation of policy distribution	[−]
τ	Trajectory	[−]
ϕ	Latitude	[rad] or [°]
ϕ_1, ϕ_2	Critic network parameters	[−]
ϕ_c	Circle center latitude	[rad] or [°]
ϕ_m	Midpoint latitude	[rad] or [°]

Φ	Mayer cost (terminal cost)	$[-]$
ψ	Heading angle	[rad] or [°]
ψ_{ref}	Reference heading	[rad] or [°]
$\omega_{\oplus}$	Earth's rotation rate	[rad/s]
$\boldsymbol{\omega}_E$	Earth's angular velocity vector	[rad/s]

Subscripts and Superscripts

Subscripts

0	Initial condition or sea level value
a	Aerodynamic frame
c	Circular orbital condition or circle center
conv	Convective
D	Down component (NED frame)
e	Earth-centered
E	Earth related
f	Final condition
h	Related to energy height
i	Index variable
k	Episode index
m	Midpoint value
N	North component (NED frame)
R	Reference frame
s	Scale
t	Value at time step t or target
V	Vehicle frame
w	Wall
x,y,z	Cartesian coordinate components
1,2,3	Numbered indices
12, 13	Indices indicating relationship between numbered points
L, U	Lower and upper bounds
$\max$	Maximum value
$\min$	Minimum value
eq	Equilibrium condition
ref	Reference value
gc	Maximum load factor condition
qc	Maximum heat flux condition
aero	Aerodynamic
grav	Gravitational
Coriolis	Coriolis acceleration
centrifugal	Centrifugal acceleration
ECEF	Earth-Centered Earth-Fixed frame
NED	North-East-Down frame

norm	Normalized value
prog	Progress
stag	Stagnation point
steep	Steepness parameter
total	Total value
θ	Actor network parameters
ϕ_1, ϕ_2	Critic network parameters
$\oplus$	Earth symbol

Superscripts

T	Transpose (matrix)
$\top$	Transpose (vector)
2	Squared
$*$	Optimal value
π	Evaluated under policy π

Mathematical Notation

$\dot{(\cdot)}$	Time derivative
$\Delta(\cdot)$	Increment or difference
$\frac{\partial f}{\partial x}$	Partial derivative of f with respect to x
$x(t)$, $u(t)$	State vector and control vector at time t
$\|\mathbf{x}\|$	Euclidean norm of vector $\mathbf{x}$
$\min_x f(x)$	Minimize function $f(x)$ over variable x
$\max_x f(x)$	Maximize function $f(x)$ over variable x
$L(x, u, t)$	Running cost
$f(x, u, t)$	System dynamics function
$g(x, u, t)$	Path constraint function
$b(x_0, t_0, x_f, t_f)$	Boundary condition function
$[\cdot]_L$, $[\cdot]_U$	Lower and upper bounds
s_t, a_t, r_t	State, action, and reward at time step t
s'	Next state after taking action
$\pi(a\|s)$	Policy: probability of action a given state s
$Q^\pi(s, a)$	Q-value: expected return for action a in state s under policy π
$Q_{\phi_i}(s, a)$	Critic network parameterized by ϕ_i
$H(\pi(\cdot\|s))$	Entropy of policy π in state s
$L(\theta)$	Loss function with parameters θ
$\mathbb{E}[X]$	Expected value of random variable X
$\mathbb{E}_{x\sim p}[f(x)]$	Expected value of $f(x)$ when x sampled from distribution p
$x \sim p$	Random variable x sampled from distribution p
$[a, b]$	Closed interval from a to b

$\log(x)$	Natural logarithm of x
$\exp(x)$, e^x	Exponential function
$\sin(\cdot)$, $\cos(\cdot)$, $\tan(\cdot)$	Trigonometric functions
$\arcsin(\cdot)$, $\arccos(\cdot)$, $\arctan(\cdot)$	Inverse trigonometric functions

List of Figures

List of Tables

1 Introduction

1.1. Background

For more than sixty years, hypersonic technologies have been extensively researched with interest on an international scale. Hypersonic Glide Vehicles (HGV) are spaceplanes that travel at speeds more than 5 times the speed of sound. Unlike ballistic missiles that follow predictable parabolic trajectories, HGVs combine hypersonic velocity with lift generation, which gives them the ability to control the flight path during atmospheric flight. These vehicles are usually launched by rockets, after which they glide through the atmosphere while generating lift to extend their range and provide cross-range maneuverability. Entry guidance has been designed for the Space Shuttle since the 1970s and was first pioneered in the Apollo program [1].

Examples of HGV include CAV, HTV, HTV-2, DF-ZF, X-37B, AHW, and YU-71 [2] [3]. Due to the many advantages that HGVs offer, such as high speeds, long glide range, and maneuverability, HGVs have attracted the attention of many countries [4]. Unlike ballistic reentry bodies, HGVs can follow complex trajectories through active flight path control.

The trajectory optimization problem for HGVs is challenging due to the extreme flight environment and the coupled nature of the control problem. During atmospheric flight, these vehicles experience severe heating, structural loads, and dynamic pressures. The flight regime spans altitudes where the HGV goes through many atmospheric layers with varying density, temperature, and pressure. Throughout the whole flight, the vehicle must satisfy three path constraints simultaneously, namely the dynamic pressure, thermal, and load factor constraint. These constraints significantly limit the feasible trajectory space, requiring precise guidance to avoid constraint violations.

HGV trajectory control involves two control variables, namely the angle of attack and the bank angle. By coordinating these two controls, the HGVs can convert altitude into range and maneuver laterally to reach targets, creating a maneuvering space that defines the reachability of the HGVs, which can be visualized as the landing footprint. To calculate a footprint, the boundary points of the landing footprint should be determined. Solving a family of optimal control problems rapidly and reliably is a daunting task, according to Saraf et al.[5].

Traditional approaches to HGV trajectory optimization typically use single-control strategies where either angle of attack is varied while maintaining fixed bank angle, or vice versa [5]. However, simultaneous control of both angle of attack (α) and bank angle (σ) provides significantly enhanced maneuverability and range extension capabilities. The gliding trajectory optimization problem for HGVs is extremely challenging due to strong nonlinear coupling among aerodynamic forces, flight commands, constraints and trajectory states [2].

1.2. Trajectory Optimization Methods

The study of hypersonic trajectory optimization began at RAND corporation in the mid 1960s. Nyland's report analyzed how hypersonic vehicles could achieve lateral range by changing bank angle. Nyland

acknowledged that vehicle control is assumed to be exercised by varying angle of attack and bank angle, however his methodology made a simplification where the control variables are assumed to be constant [6]. This choice was sensible for the time; varying controls continuously throughout the trajectory would have required numerical optimization methods that were beyond the computational resources of the 1960s. By holding both controls constant, Nyland could derive analytical solutions, which simplifies the control structure to gain physical insight.

Building upon these foundations, researchers in the 1970s to 1980s developed a framework of equilibrium glide conditions that would dominate on hypersonic glide vehicle trajectory optimization. The equilibrium glide condition is a condition where the vehicle balances the lift against gravitational and centrifugal forces. This eliminates the possibility of the vehicle to skip through the atmosphere. This simplification of the problem provided a powerful analytical tool for determining angle of attack requirements.

Trajectory optimization methods traditionally fall into two main categories: indirect methods based on analytical optimality conditions, and direct methods that convert the problem into parametric optimization through discretization. The indirect approaches rely on classical variational techniques or Pontryagin's minimum value principle, converting the optimal control problem into a Hamiltonian boundary value problem with two endpoints [7]. This guarantees the optimality of the solution [8] [9] [10]. However, solving two-point boundary value problems is difficult and requires good initial guesses for complex problems [11]. This is also stated by Huang et al. [12], with the exception of simple problems, an accurate solution will be hard to find. The nonlinear system of ordinary differential equations of a hypersonic glide vehicle are complex. Under these conditions, it is much harder to solve the two-point boundary value problem. However, according to Grant et al. [13], an optimization has been performed on a hypersonic glide vehicle using an indirect method.

In contrast, direct methods have become increasingly popular with the advancement of computing technology [2]. These include techniques such as direct shooting, collocation, differential inclusion, and pseudospectral methods, Here the original control problem is converted into a parametric optimization problem and solved through nonlinear programming (NLP) techniques [14], such as sequential quadratic programming [15] [16] [17] [18]. Zhang et al. [19] have proposed a direct approach using non-linear programming and the Gaussian Pseudospectral Method to minimize the terminal control energy. Furthermore, Tu et al. [20] have proposed a direct method using a Non-Linear Programming problem and the direct collocation method. However, the time it takes to perform the direct method is unpredictable, and convergence cannot be guaranteed when applied to hypersonic flight scenarios [1].

Convex optimization techniques are increasingly being used to address guidance, control and trajectory optimization challenges [14]. A subclass of convex optimization is second-order cone programming (SOCP), which was proven to be promising for real time onboard guidance systems. However, the entry trajectory optimization problem does not naturally fit into any convex optimization framework due to the nonlinear dynamics and the inequality constraints are neither second-order cones nor convex. Liu et al. [21] applied the second-order cone programming method by reformulating the equations of motion with respect to energy and the problem was shaped into a suitable form that can be solved by SOCP. Additionally, pseudospectral optimal control and convex optimization were combined to address the optimal drag-energy guidance [22].

Specific methods for footprint generation have evolved over several decades. In 1981 Vinh et al. [23] have produced a method that gives a footprint without vehicle constraints or taking into account the Coriolis effect. This work was continued by Ngo et al. [24] in 2002. They have developed a method that generates nearly optimal landing footprints, but did not take into account the Coriolis effect. Saraf et al. [5] have developed a method that generates a footprint, taking into account the path constraints and the Coriolis effect. However, all of these have not used angle of attack control in combination with bank angle control. This single control fails to exploit the full control authority available through simultaneous modulation of both control inputs.

Most existing methods optimize only bank angle while fixing angle of attack at maximum lift to drag ratio, inherently constraining exploration of the full control space. Reinforcement learning offers a promising alternative for aerospace trajectory optimization problems [25]. In recent years, research has investigated advanced solution approaches that offer reliable convergence, optimal results, and

rapid computation. This effort has been driven by new technologies including artificial intelligence and machine learning [26]. The application of reinforcement learning has accelerated in recent years [27]. Reinforcement learning algorithms have shown promising results for aerospace problems [28] [29]. In contrast to traditional guidance algorithms, reinforcement learning-based guidance algorithms show strong anti-disturbance capabilities and real-time performance [30] [31] [32].

Chai et al. [25] integrated deep neural networks with optimal control for real-time reentry trajectory planning, while Shi et al. [1] developed deep learning frameworks for onboard trajectory generation. However, these approaches rely on supervised learning from pre-computed optimal trajectories. Recent developments in model-free reinforcement learning have explored direct policy training. The Soft Actor-Critic (SAC) algorithm has shown strong sample efficiency in continuous domains [33], enabling direct policy learning without optimal trajectory datasets. Gao et al. [34] applied deep deterministic policy gradient to reentry trajectory optimization, and Su et al. [35] used inverse reinforcement learning for point-to-point entry guidance. Das et al. [36] note that RL offers significant opportunities for HGV trajectory control, though further research is needed.

Most state-of-the-art footprint generation methods rely on several simplifying assumptions that limit the accuracy. Simplifications in research include using equilibrium glide assumptions, heuristics for minimum range boundary determination, neglecting Coriolis forces from Earth rotation, and fixed angle of attack profiles at predetermined values with trajectory shaping through bank angle modulation. These assumptions reduce computational cost, but limit the maneuverability and accuracy of the footprint.

This work presents the first RL-based approach for complete footprint generation with dual control authority (bank angle and angle of attack) on a rotating spherical Earth model, where the RL model can be extended with no-fly zone avoidance and direct target point navigation. The trained policy generates footprints for any global location while incorporating full three-degree-of-freedom dynamics including Coriolis effects and control rate limitations.

1.3. The Challenge of Angle of Attack and Bank Angle Control Together in HGV Trajectories

Coordinating angle of attack and bank angle control simultaneously is one of the most difficult problems in HGV trajectory optimization. Both controls interact with the vehicle's energy and lift distribution, so changes in one directly affect the other. At hypersonic speeds, this coupling becomes highly nonlinear, where small variations produce large shifts in altitude loss, heating, and lateral motion.

Classical optimization methods struggle with this interaction. The standard trajectory method has an angle of attack profile that is designed in advance and the lateral motion depends on the bank angle reversal strategy [37]. This decoupling simplifies the problem, but sacrifices optimality. When the angle of attack is changed, the total lift of the vehicle is changed. When the bank angle is changed, the distribution of the lift between vertical and horizontal is changed. This means adjusting one control parameter requires compensating adjustments in the other.

The primary difficulty with numerical optimization of both controls simultaneously is computational complexity. The coupled problem requires searching over two time-dependent control histories instead of one, which significantly expands the search space. Additionally, hypersonic dynamics are numerically stiff, containing both fast dynamics and slow dynamics, requiring small integration time steps that increase computational cost. The path constraints on g-load, dynamic pressure, and stagnation point temperature create narrow feasible trajectory regions with discontinuous gradients when constraints activate. These factors cause poor convergence in gradient-based methods. The reinforcement learning approach developed in this work addresses the coupled problem by learning a neural network policy that commands both controls simultaneously based on the observed state.

1.4. Thesis Focus and Structure

This research is organized into ten chapters, with four Appendices providing supplementary material. This thesis integrates the literature review and project proposal throughout the document instead of presenting them as a separate standalone chapter, further explained in Section 1.4.1.

Chapter 1 establishes the research context by reviewing the state-of-the-art in hypersonic trajectory optimization. The chapter presents the evolution from equilibrium glide assumptions to modern computational methods, highlighting the limitations of single control approaches. The technical challenge of coupled angle of attack and bank control is examined, which motivates the reinforcement learning approach developed in this work.

Chapter 2 defines the research questions. The chapter gives the main research question regarding dual control implementation and gives five supporting sub-questions. The chapter presents the specific technical objectives that guide the framework development and sub-divides this further into work packages.

Chapter 3 identifies the key stakeholders who would benefit from this research and examines their specific interests and requirements.

Chapter 4 provides a comprehensive overview of trajectory optimization techniques applicable to hypersonic glide vehicles. The chapter examines indirect methods based on Pontryagin's Maximum Principle, direct methods including shooting, collocation, pseudospectral techniques, and heuristic approaches such as genetic algorithms and particle swarm optimization. Software tools including DIDO, GPOPS-II and CasADi are elaborated upon. The chapter concludes with drag-energy optimization formulations and their application to footprint generation problems.

Chapter 5 presents the preliminary development phases that formed the final framework design. The chapter begins with validation of the equations of motion through Monte Carlo simulations. Early optimization attempts using Adaptive Simulated Annealing and DIDO are presented, which revealed the computational challenges of this complex problem. The chapter gives the development of constraint handling and the selection of the Soft Actor-Critic algorithm as the reinforcement learning method. Initial training results demonstrate the feasibility of the RL approach.

Chapter 6 consists of the core technical content of the thesis. The chapter provides detailed documentation of the framework architecture, beginning with the object-oriented design philosophy and module organization. The core physics engine is described, which includes the implementation of atmospheric modeling, aerodynamic coefficient interpolation, and coordinate transformations between spherical and Cartesian reference frames. The equations of motion include Coriolis and centrifugal effects on a rotating Earth. The reinforcement learning implementation is examined in detail, which includes the environment design, observation and action spaces, reward formulation and constraint penalty design. The chapter explains the Soft Actor-Critic algorithm and its application to the trajectory optimization problem, including the training process with randomized initial conditions. The results are presented for two vehicle configurations. Footprint generation results show symmetric patterns consistent with Earth's rotational dynamics. The generalization capability of the framework is demonstrated through footprints generated from arbitrary global locations. Extensions of the model include target point guidance and no-fly zone avoidance, which shows the framework's versatility for operational applications.

Chapter 7 validates the reinforcement learning generated footprint through a large-scale Monte Carlo simulation. 100 million trajectory evaluations with randomly sampled angle of attack and bank angle profiles confirm that the RL policy has discovered the true boundaries of the reachable domain. The chapter describes the spatial thinning algorithm used to manage the massive dataset and provides quantitative comparisons to demonstrate the agreement between the RL and Monte Carlo approaches.

Chapter 8 performs a sensitivity analysis that shows how variations in input parameters affect system performance and tests the learned policy's ability to generalize beyond the training distribution. The chapter examines sensitivity to initial flight conditions including altitude, velocity, and flight path angle, as well as vehicle properties such as mass and reference area. The results show that the learned policy maintains reasonable performance when operating in regions of state space that are not explicitly encountered during training.

Chapter 9 gives the conclusion and recommendations. The chapter summarizes key achievements, including the first RL-based approach to complete generation of HGV footprints with dual control authority. Recommendations address future development directions such as scramjet propulsion integration, waypoint navigation, terminal phase guidance and extension to six-degree-of-freedom dynamics. Finally, the research questions are looked back on and answered.

Chapter 10 presents the journal article that will be submitted to Aerospace Science and Technology. The paper gives a condensed presentation of the framework methodology, key results and validation studies. The chapter demonstrates how the thesis contributions are used to advance the state-of-the-art in hypersonic glide vehicle trajectory optimization.

Appendix A shows comprehensive diagrams documenting the software implementation. The appendix includes class structures for the HGV and environment modules, activity diagrams showing the execution flow of trajectory integration, and sequence diagrams illustrating the interactions between components during footprint generation, training episodes, and trajectory execution.

Appendix B provides additional information on the thermal environment encountered during hypersonic flight and discusses the requirements and material considerations for hypersonic glide vehicles.

Appendix C describes the scramjet integration content, covering the operational envelopes and shock-on-lip conditions.

Appendix D describes the project planning methodology. Here the planning strategies are explained and it is stated that the project has been finished within the nominal time given by the TU Delft.

1.4.1. Literature Review Organization

The literature review is located in Chapter 1 covering the historical development and state-of-the-art in hypersonic trajectory optimization, and in Chapter 4 where a comprehensive review of all relevant optimization techniques, software tools, and applications from literature is described. Additional background and explanatory content related to atmospheric modeling standards, computational implementation techniques, spherical geometry calculation, and constraint formulations are integrated within Chapter 6 where they directly support the methodology and framework design. The appendices also contain literature-based content: Appendix B reviews thermal protection systems and materials for hypersonic flight, and Appendix C explains scramjet integration and propulsion-airframe coupling from existing research. The project proposal is presented in Chapter 2, which defines the research questions. Chapter 3 provides a stakeholder analysis and Appendix D gives the planning of the thesis. This integrated approach provides the literature in the most relevant context.

2

Research Objective

2.1. Introduction & Relevance of Thesis

HGVs represent a critical advancement in aerospace technology, where hypersonic speeds up to Mach 20 are experienced while maneuvering through the atmosphere. Understanding the reachable footprint of these vehicles from any given initial condition is essential for mission planning and threat assessment.

Traditional approaches to HGV trajectory optimization typically focus on single control strategies, where either the angle of attack is varied while maintaining fixed bank angle or the other way around. However, the simultaneous control of both angle of attack and bank angle gives significantly enhanced maneuverability and range extension capabilities. The challenge lies in determining optimal control strategies that maximize the reachable footprint while still satisfying the aerodynamic, thermal and structural constraints throughout the flight.

This thesis addresses the problem of determining what regions an HGV can reach when both control inputs are used simultaneously. Beyond mapping the maximum reachable footprint, the framework enables precision guidance to specific target points within the footprint and can incorporate no-fly zones for realistic operational scenarios.

2.1.1. Project Vision and Scope

The primary objective of this research is to develop a framework for determining the maximum reachable footprint of a HGV using simultaneous angle of attack and bank angle control. The key objectives are the following:

1. Develop a high-fidelity 3-DOF point mass simulation that accurately describes the HGV flight dynamics including atmospheric effects, spherical Earth geometry, Earth rotation, and realistic aerodynamic interpolation from tabulated data.
2. Implement dual control trajectory optimization that gives a guidance strategy for both angle of attack and bank angle simultaneously, rather than keeping one fixed or use pre-determined profiles.
3. Generate footprints that show the complete operational envelope by computing trajectories in multiple radial directions from the initial position.
4. Target guidance to specific points within the footprint, that can be used to check if all points within the footprint can be reached.
5. Incorporate no-fly zone constraints that allow trajectories to avoid specific regions.
6. Satisfy constraints throughout all trajectories, where g-load, dynamic pressure and temperature limits are monitored.
7. Validate the framework through test cases that verify correct implementation of the model.

2.2. Research Questions

This research addresses one main question with four supporting sub-questions that guide the investigation.

Main Research Question:
How can dual control of angle of attack and bank angle be implemented to maximize the reachable footprint of Hypersonic Glide Vehicles under aerodynamic, thermal and structural constraints?

The sub-questions are divided into the following:

- *How can HGV flight dynamics be accurately modeled to support footprint analysis in a high-fidelity model?*
- *How can the coupled nonlinear interaction between angle of attack and bank angle be optimized simultaneously?*
- *What computational approach enables efficient multi-directional trajectory generation to create footprints?*
- *How can trajectories be generated that maintain constraint satisfaction throughout the entire flight?*
- *How can footprint accuracy be validated across different conditions?*

The achievements of the research objectives are given in Section 9.2.

2.2.1. Work Packages

To achieve the research objectives systematically, the work is divided into five work packages. Each package addresses specific challenges and relates this back to the sub-questions. Table 2.1 gives an overview of all work packages and their main deliverables, followed by a more detailed description of each package.

WP	Title	Main Deliverable
WP1	Model development & validation	3-DOF simulation framework
WP2	Dual control optimization	Optimization framework
WP3	Footprint generation and target guidance	Footprint algorithm
WP4	Constraint handling	Constraint monitoring system
WP5	Framework validation	Validation protocol

Table 2.1: Overview of work packages

Work Package 1

Work Package	WP1: Model development and validation
Objective	Develop a high-fidelity 3-DOF point mass simulation that accurately describes HGV flight dynamics
Key Tasks	• Implement spherical Earth equations of motion • Integrate the atmospheric model • Develop aerodynamic interpolation from tabulated data efficiently • Verify equations of motion with equivalent cartesian formulation • Verify North-South symmetry and Coriolis effect
Deliverables	• Validated 3-DOF simulation framework • Verification test cases demonstrating model accuracy
Related Research Question	How can HGV flight dynamics be accurately modeled to support footprint analysis in a high-fidelity model?

Table 2.2: Work Package 1

Work Package 2

Work Package	WP2: Dual Control Optimization
Objective	Implement simultaneous optimization of angle of attack and bank angle for trajectory control
Key Tasks	• Select and implement appropriate optimization algorithms • Define objective functions for range maximization • Represent and structure the control inputs over time, including rate limiting • Analyze interaction effects between control variables
Deliverables	• Dual control optimization framework • Comparison with single control strategies
Related Research Question	How can the coupled nonlinear interaction between angle of attack and bank angle be optimized simultaneously?

Table 2.3: Work Package 2

Work Package 3

Work Package	WP3: Footprint generation and target guidance
Objective	Generate complete reachable footprints and guidance to specific targets
Key Tasks	• Develop multi-directional trajectory computation strategy • Implement radial heading sweeps from initial position • Create footprint boundary determination algorithms • Develop target point guidance capability • Verify reachability of points within footprint using direct guidance
Deliverables	• Footprint generation algorithm • Target guidance framework
Related Research Question	What computational approach enables efficient multi-directional trajectory generation to create footprints?

Table 2.4: Work Package 3

Work Package 4

Work Package	WP4: Constraint Handling
Objective	Ensure all trajectories satisfy operational constraints
Key Tasks	• Implement g-load constraints throughout the whole flight • Implement dynamic pressure constraints throughout the whole flight • Implement temperature constraint throughout the whole flight • Validate constraint satisfaction for all generated trajectories
Deliverables	• Constraint monitoring system • Constraint violation detection system
Related Research Question	How can trajectories be generated that maintain constraint satisfaction throughout the entire flight?

Table 2.5: Work Package 4

Work Package 5

Work Package	WP5: Framework Validation and Verification
Objective	Validate the complete framework through comprehensive tests
Key Tasks	• Verify model implementation through formulation comparison • Validate physical consistency through symmetry tests • Perform Monte Carlo validation • Compare maximum ranges with literature benchmarks • Test framework for multiple vehicles • Perform sensitivity analysis
Deliverables	• Validation test setup • Performance metrics and analysis • Framework limitations
Related Research Question	How can footprint accuracy be validated across different conditions?

Table 2.6: Work Package 5

3

Stakeholder Analysis

3.1. Stakeholders

This thesis is conducted as part of the Master's program Aerospace Engineering at Delft University of Technology and in collaboration with the Netherlands Aerospace Center (NLR) which, in collaboration with the student, provided the assignment and project context.

Primary Stakeholders

- Netherlands Aerospace Center (NLR)
  As the commissioner, NLR is the primary stakeholder. They define the problem scope, provide access to relevant data, tools, or experts, and will evaluate the practical relevance of the results. The research results are expected to contribute directly to the ongoing work of NLR.
- TU Delft - Supervisors and Academic Staff
  The academic supervisors ensure the scientific quality of the thesis and make sure the thesis is according to guidelines. They provide guidance on research design and alignment with academic standards. The thesis must meet the requirements of the TU Delft to be considered for graduation.

Secondary Stakeholders

- Engineers at NLR
  The engineers at NLR are affected by the research outcomes as they can use the findings of the thesis in their according projects.
- Future students or interns
  The thesis may serve as the foundation or reference for future students who are working on similar problems within TU Delft or at NLR.
- European Space Agency (ESA)
  ESA has interest in hypersonic technologies for atmospheric reentry vehicles and potential space plane development. The research findings on HGV trajectory optimization and dual control strategies could inform their programs on reentry guidance.
- NASA
  NASA is actively researching hypersonic flight for atmospheric reentry. The methodology developed in this thesis for handling coupled nonlinear dynamics and constraint satisfaction could be applicable to their trajectory optimization problems.
- Dutch Ministry of Defense
  Due to the strategic implications of hypersonic glide vehicle technology, the Dutch Ministry of Defense has an interest in developments in this field from defense perspectives. Understanding the operational capabilities and limitations of HGVs is relevant for national security considerations.
- NATO
  NATO allies collaborate on defense research, including hypersonic systems. The optimization

methodologies and footprint analysis techniques developed in this research could contribute to a broader understanding of HGV capabilities.

- TNO
  Organizations like TNO and similar defense research institutes have interest in hypersonic technologies for both military applications and civilian technologies. The computational approaches developed could be relevant for their simulation and analysis work.

3.2. NLR Expectations

Together with NLR, the following thesis outline has been thought of, which resulted in an assignment that NLR will benefit from in combination with a suitable graduation project.

This thesis shall develop an optimal guidance strategy for a Hypersonic Glide Vehicle using simultaneous angle of attack and bank angle control to maximize the reachable footprint within flight constraints. The dual control approach is to fully exploit the HGVs maneuverability by coordinating both control inputs throughout the trajectory, instead of using pre-defined angle of attack profiles with bank angle reversal strategies.

The primary challenge lies in the coupled nonlinear interaction between bank angle and angle of attack control. Traditional optimization methods struggle with this dual control problem, especially when generating complete footprints requiring feasible trajectories in all directions. Therefore, a computational approach must be developed that can efficiently handle the coupled dynamics, enforce constraint satisfaction across all trajectories and generalize to arbitrary target directions without requiring individual re-optimization for each direction. This should all be done with a high-fidelity model that will reflect a real-life example.

The primary deliverable is a comprehensive footprint map that visualizes all attainable endpoints from given initial conditions, demonstrating the full operational capability enabled by dual control of angle of attack and bank angle.

4

Optimization Methods

Various optimization methods for solving the trajectory optimization problem are reviewed here. The discussion includes direct, indirect, and heuristic methods, focusing on their applicability, strengths, and limitations in the context of HGV flight.

Hypersonic technology today is studied all over the world. Since the 1950s, a great deal of progress has been made in modeling the dynamics of hypersonic glide vehicles. Especially in optimizing the trajectory a lot of progress has been made.

The exact equations of entry dynamics cannot be solved analytically and, therefore, numerical computer models have to be used to analyze the problem [38]. Both direct and indirect methods aim to find an optimal control strategy that satisfies system dynamics while minimizing an objective function, which could be maximizing range, for example. Direct methods come with two problems: how can the optimal control problem be transformed to a nonlinear programming problem and how to resolve this transformed nonlinear programming problem [12]. Direct methods must solve large-scale nonlinear programming problems, which introduce challenges in convergence and computational cost. The direct method is, in comparison to the indirect method, more widely applied.

4.1. Formulation of Continuous Dynamical Systems

In this section, the general non-linear dynamical system that represents the motion of the HGV is introduced. The state space representation of the system and associated boundary conditions are described, forming the mathematical foundation for trajectory optimization.

Hypersonic glide vehicles follow a carefully designed trajectory to meet mission objectives. The design and optimization of this trajectory ensure compliance with flight constraints while the performance is maximized. Throughout the literature, there is a wide variety of dynamical system models. Although there are many differences, they all share a common mathematical framework. Typically, the state of a space vehicle is described with a dimensional vector $x(t)$, whose time evolution is given by a system of differential equations. The representation is usually as follows [39]:

$$\dot{\mathbf{x}}(t) = f\big(\mathbf{x}(t), \mathbf{u}(t), t\big),$$

where $u(t)$ is the control variable, f is a function with respect to time and $x(t)$ the state. In general, the function f is non-linear. The state vector $x(t)$ starts at a predetermined point called x_0 and at a predetermined time instant t_0. The goal of the system is to reach a predetermined final state x_f at a final time t_f. Therefore, there is an inequality to describe the boundary condition as follows:

$$b_L \leq b(x_0, t_0, x_f, t_f) \leq b_U$$

4.2. Optimization Problem

In trajectory optimization, a feasible flight trajectory is defined as a solution to the vehicle's dynamical system while satisfying the prescribed path constraints. For a given mission profile, there are many feasible trajectories. In order to select an optimal trajectory, a performance index is introduced depending on the mission's objective. This allows for a quantitative measure. The optimization problem is usually formulated using a cost function, which has the following form:

$$J = \Phi(x_0, t_0, x_f, t_f) + \int_{t_0}^{t_f} L(x(t), u(t), t)\, dt \tag{4.1}$$

where Φ is the Mayer cost, which depends on the initial and final conditions, and L is the process cost, which integrates throughout the trajectory. The Mayer cost is evaluated at the endpoints of the trajectory, meaning it does not accumulate during the flight. The process cost is integrated over time domain $[t_0, t_f]$ meaning it accounts for continuous factors during the entire mission [39].

The cost function must be minimized. Here, $x(t)$ is the state vector which represents all the variables that are needed to describe the system at any given time, which usually are position and velocity for an HGV. $u(t)$ is the control vector that represents the variables that can be adjusted to influence the behavior of the system. For an HGV this could be the bank angle and the angle of attack. The state vector describes what the system is doing, while the control vector describes how you influence it. J is the objective function, which is aimed to be minimized. This measures the total cost of the trajectory and control input over time. $\Phi(x_0, t_0, x_f, t_f)$ is the terminal cost. This could be used to minimize the flight time, for example, or to achieve a final state with minimal error. This is applied only at the end of the trajectory. This could be seen as a penalty for arriving late or missing the target destination. $\int_{t_0}^{t_f} L(x(t), u(t), t)\, dt$ is the running cost and represents the cumulative cost that is accumulated during the flight, which is depending on the state, control, and time. This could be used to minimize energy consumption, control effort or deviations from a desired path, for example. The running cost is, for example, fuel consumption throughout a journey. This penalizes excessive energy use or large control input during flight. The cost function J quantifies the performance of the trajectory. Lower values of J mean better performance.

The total formulation of the trajectory optimization can be given as follows [39]:

$$\begin{aligned}
\text{minimize} \quad & J = \Phi\big(x_0, t_0, x_f, t_f\big) + \int_{t_0}^{t_f} L\big(x(t), u(t), t\big)\, dt \\
\text{subject to} \quad & \dot{x}(t) = f\big(x(t), u(t), t\big) \quad \text{(dynamic constraints)} \\
& b_L \;\le\; b\big(x_0, t_0, x_f, t_f\big) \;\le\; b_U \quad \text{(boundary conditions)} \\
& g_L \;\le\; g\big(x(t), u(t), t\big) \;\le\; g_U \quad \text{(path constraints).}
\end{aligned}$$

4.3. Direct methods

Direct methods discretize the trajectory and solve the resulting NLP using numerical solvers. Several variants such as direct shooting, multiple shooting, and collocation methods are presented, with emphasis on their suitability for high-dimensional constrained problems.

Direct methods have been widely used in the field of vehicle trajectory optimization, and indirect methods are also used but to a lesser extent. This has been extensively researched by Betts [40]. Zhang et al. [19] have proposed a direct approach using non-linear programming and the Gaussian Pseudospectral Method to minimize the terminal control energy. Furthermore, Tu et al. [20] have proposed a direct method using a Non-Linear Programming (NLP) problem and the direct collocation method. An indirect method has been used by Adimurthy [41]. It must be noted that this was used in 1976. Afterwards, Adimurthy performed a trajectory optimization in 1987, where a direct method was applied and the robustness of NLP was demonstrated [42]. Moreover, Li et al. [43] in 2009 performed a downrange optimization of a Hypersonic Boosted Glide Vehicle using a direct method in which path constraints were included. By Zhang et al. [44] a direct method has been employed in which the total heat flux has

been minimized. Finally, Prasanna et al. [45] have also used a direct method to maximize the range with a non-linear programming formulation.

4.3.1. Direct Shooting Method

The direct method parameterizes the control inputs and integrates the dynamics forward. The optimizer adjusts the control parameters to minimize the objective. If the final result does not match the target, the starting conditions are adjusted and the process is repeated until the right trajectory has been found. Direct shooting can be accurate but becomes numerically unstable for long-duration or highly sensitive trajectories, making it unstable for HGV problems. If the initial guess is poor, the optimizer could fail to converge, which requires fine-tuning or alternative initialization methods. Additionally, it suffers from numerical instability [46], since small errors in initial conditions grow over time, and direct shooting struggles with long-duration trajectories such as the HGV mission. Since the direct-shooting method optimizes all controls at once, it might find a solution which is not a global optimum but a local minimum.

4.3.2. Multiple Direct Shooting Method

The multiple direct shooting method is an improved version of the direct shooting method, which has been designed to overcome the issues of stability and convergence. This is especially a problem for long-duration or highly sensitive trajectory optimization problems. The multiple direct shooting method first divides the trajectory into segments. Instead of integrating the full trajectory from a single starting point, multiple direct shooting splits the trajectory into smaller segments. Then, the control inputs are optimized for each segment, where the initial conditions for each segment are free variables to be optimized. The entire problem, including the segmented trajectory and boundary constraints, is solved using a numerical optimization method like sequential quadratic programming. This method is more stable than direct shooting since the trajectory is divided, and numerical errors do not accumulate as much. This comes at the price of higher computational costs, and the need for a good initial guess is still an issue [12].

4.3.3. Collocation Method

The collocation method solves the trajectory optimization by breaking the trajectory into discrete points, called collocation points. The systems' behavior is approximated using polynomials. So, instead of solving the entire trajectory as a continuous function, it converts the problem into a set of algebraic equations. The state and control variables are discretized at these specific collocation points. The trajectory between collocation points is approximated using polynomials. The equations of motion are enforced as constraints at each collocation point. The entire trajectory is transformed into a large non-linear programming problem. The solver then finds the best polynomial approximation that satisfies all constraints. This approach is more stable than the shooting methods, since the whole trajectory is optimized at once and no numerical integration errors add up over time. Additionally, this method provides a smoother and more accurate trajectory due to the use of high-order polynomials [12]. The collocation method can be transcribed using pseudospectral methods. A pseudospectral method is part of the direct methods, which have a discretization of the optimal control problem, resulting in a finite non-linear programming problem. Tools that implement pseudospectral methods are DIDO and GPOPS [22], which will be discussed in Section 4.6

Gaussian Pseudospectral Method

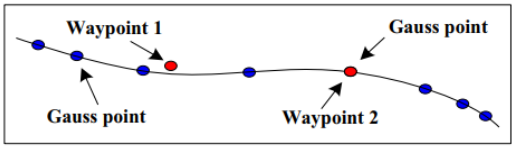
In the past 10 years, the Gauss Pseudospectral Method (GPM) has been used to deal with trajectory optimization problems [47]. GPM can be used for a reentry vehicle to calculate a nominal trajectory prior to launch. It has been demonstrated that, of many numerical methods, GPM is one of the most convenient tools for reentry trajectory optimization problems. This is due to the fact that HGVs are subjected to many constraints such as aerodynamic load, dynamic pressure, and heating rate. Additionally, GPM is crucial for HGV due to the non-linear dynamics of hypersonic glide vehicles. Xiaodang et al. [48], have obtained great results using the Gauss pseudospectral method to optimize the glide trajectory. Additionally, Zhang et al. [19] have used GPM where the cost function was used to reduce terminal control energy and were successful in this.

GPM works with a series of discrete points, which are called Gauss points. These are used to make the optimal control problem from a continuous problem to a discretized problem. Gauss points are specific

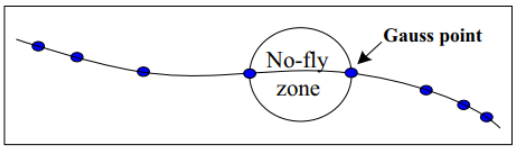
locations that are used in numerical methods to approximate integrals. The final solution meets the requirements of all constraints at each Gauss point. However, in between these points the solution is not verified. Additionally, many Gauss points are needed that can affect the convergence speed and also introduce large approximation errors.

The principle of GPM is that the optimal control problem is discretized by assigning a number of Legendre-Gauss (LG) points. These are the roots of Legendre polynomials, which are solutions to the Legendre differential equation. All these points are within the interval $[-1, 1]$. However, the trajectory optimization is formulated in the time interval $[t_0, t_f]$, so the independent variable is transformed to the interval $[-1, 1]$. After this, the problem is converted into a non-linear programming problem which can afterwards be solved by the Sequential Quadratic Programming (SQP) algorithm.

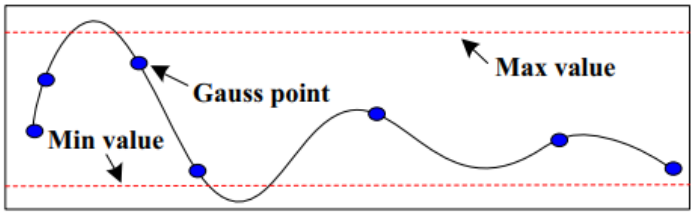
Using this GPM, all constraints can be satisfied with high precision at each Gauss point. However, between Gauss points, constraint satisfaction is not verified. Therefore, this method comes with the problem that the waypoints constraints cannot be satisfied unless one of these intersects with a Gauss point. Additionally, it could be that the conditions for the no-fly zones are not satisfied. Finally, similar to these no-fly zone constraints, the path constraints could also not be satisfied except at the Gauss point positions. These problems are depicted in Figure 4.1.



(a) Limitation of GPM on waypoint constraint.



(b) Limitation of GPM on no-fly zone constraint.



(c) Limitation of GPM on path constraint.

Figure 4.1: Limitations of GPM [47].

These problems are caused by the distribution of these Gauss points. With the improved GPM, it has been suggested that more Gauss points are allocated to no-fly zones and to make sure that the location of waypoints coincide with Gauss points. In order to achieve this, the trajectory will be divided into several segments, and afterwards Gauss points are allocated to each segment, and thus more Gauss points are near the breaks.

4.3.4. Differential Inclusion Method

The differential inclusion method is a way to optimize a trajectory without explicitly defining control inputs [49]. Instead of solving for exact control values at each moment, it defines a range of possible state transitions and will let the optimizer pick the best one. For example, the method might specify ”at time X, the bank angle can be anywhere between 20 degrees and 40 degrees”. This creates a set of possible states rather than a single trajectory. The optimizer then explores all possible state transitions and picks the one that leads to the best trajectory while satisfying the constraints. In general, differential inclusion is computationally faster compared to the collocation method. Since the controls are not directly solved for, reconstructing the exact control strategies is more complex and less precise. Especially when this is compared to the collocation method that gives explicit control profiles at high precision.

4.4. Indirect Methods

Based on Pontryagin's Maximum Principle, indirect methods yield high-accuracy solutions by solving boundary value problems involving costate dynamics. This section outlines their theoretical advantages and practical challenges in application.

Indirect methods are used to solve an optimal control problem by first deriving mathematical equations that describe the best possible solution. These equations are analytical, but are solved numerically. Instead of directly looking for the best trajectory by discretizing the state and control variables, which is the direct method, the indirect method starts by using a tool from optimal control theory that is called *Pontryagin's Maximum Principle*. This principle works using another set of variables, called the costates. These measure how sensitive the performance measure is to changes in the state variables. These are sensitivity indicators, which tell how much a small change in the current state will affect the objective function. The Pontryagin's maximum principle tells that if a trajectory is the best, then it must satisfy certain mathematical conditions. Using the equations of motion along with the extra costate variables, a set of equations is formed. These equations are solved together to find the optimal trajectory. Since the start position and the end positions are known, this is a problem that is fixed at the beginning and at the end, resulting in a two-point boundary value problem. This indirect method can give precise answers in the case that the equations can be solved. However, a good guess for the costate variables is required, and otherwise solving the equations becomes difficult.

According to Huang et al. [12], it is very difficult to solve two-point boundary value problems. With the exception of simple problems, an accurate solution will be hard to find. The nonlinear system of ordinary differential equations of a hypersonic glide vehicle are complex. Under these conditions, it is much harder to solve the two-point boundary value problem.

However, according to Grant et al. [13], an optimization has been performed on a hypersonic glide vehicle using an indirect method. The solution of the indirect method has been compared with a modern trajectory optimization tool, DIDO, which validated the solution of the indirect method. This has been accomplished by first extending the trajectory so that it matches the desired initial and terminal conditions. After that, the constraints are introduced and gradually reduced to the desired value. As a result of this, complex trajectory solutions can be quickly constructed through a sequence of progressively difficult optimization problems. In Figure 4.2 the comparison between the modern optimization tool and the indirect method is shown on the footprint of the HGV.

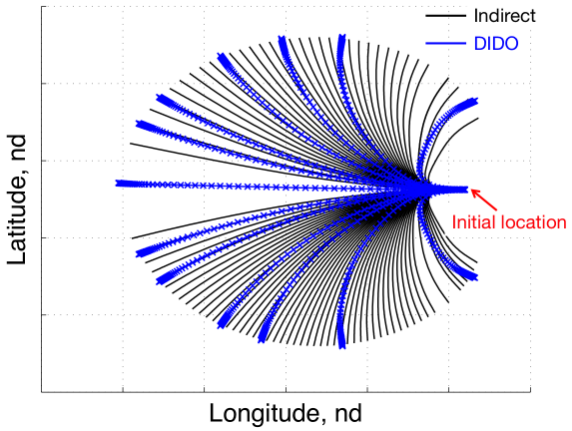


Figure 4.2: Comparison between the indirect method and the modern optimization tool [13].

In summary, the difference between direct methods and indirect methods can be given in the following way. The advantages of the indirect methods are that it gives high accuracy results and the solutions satisfy the optimality conditions. However, necessary optimality conditions must be derived analytically, a good initial guess is needed due to the small convergence space, and also a need for a guess for the

costate functions is needed. The direct method avoids the disadvantages of the indirect methods, but gives less accurate results [50].

4.5. Other Methods

Heuristic and gradient-free methods such as genetic algorithms, particle swarm optimization, and dynamic programming are discussed as alternative approaches. Although generally less accurate, these methods offer robustness and global search capabilities.

4.5.1. Dynamic Programming

Dynamic programming is a mathematical method to solve trajectory optimization problems by breaking the problem into discrete stages and finding the optimal continuous input at each stage by solving smaller subproblems recursively. The advantage of dynamic programming is that it has a simple calculation theory and a relatively high accuracy [12]. However, the largest disadvantage of this method is the large amount of memory space required. Dynamic programming is one of the methods that has been used in the early stages of vehicle trajectory optimization.

4.5.2. Genetic Algorithm

Another possible method to perform an optimization is to use a genetic algorithm, which is based on natural selection of the fittest. Genetic algorithms have been extensively applied in the trajectory optimization of reentry vehicles [51]. Genetic algorithms are robust, are not sensitive to initial values, and have good performance when dealing with large and complex nonlinear systems [12].

As an example, Yokoyama et al. [52] have used an optimization of the reentry trajectory with a nonlinear programming problem formulating using a real-coded genetic algorithm. The genetic algorithm has been shown to perform the global search of the objective function. Moreover, the genetic algorithm has been applied to optimize the reentry trajectory of a space plane, and it was observed that the solution approached the vicinity of an optimal solution.

Furthermore, Gang et al. [53] have proposed a reentry trajectory with minimum control energy and a fixed terminal time using a genetic algorithm. Also here it was shown that the global optimum can be obtained faster and that the approach is not sensitive to the initial values.

A disadvantage of using genetic algorithms is that it often suffers from slow convergence rates [54]. Additionally, genetic algorithms struggle to meet the tight nonlinear constraints [55] that are experienced for this HGV reentry problem.

4.5.3. Particle Swarm Optimization

Particle Swarm Optimization (PSO) is an evolutionary computation and was soon regarded as a promising optimization algorithm. This is due to its simple nature and rapid convergence [12]. Rahimi et al. [56] have successfully used a reentry vehicle trajectory optimization where the objective was to minimize the heat generated. As a conclusion of this study, it was presented as an efficient, reliable, and accurate method to determine trajectory optimization.

However, there are also disadvantages to this method. It is easy to fall into a local optimum in high-dimensional space, like in the HGV problem. Additionally, it has a low convergence rate in the iterative process [57]. The premature convergence is also stated by Liu et al. [58], saying it has a tendency to become trapped in local optima.

4.5.4. Pattern Search

The pattern search method has not been extensively investigated in the field of trajectory optimization. It has the potential to deal with optimization problems that are highly constrained, as it is a direct search-based gradient-free algorithm that does not require the objective function derivative to reach the global minima [51].

For an HGV, there are usually three path constraints. These constraints for gradient-based methods are written as equations and then incorporated into the Hamiltonian function. The optimal direction is then determined by computing the partial derivative of the Hamiltonian with respect to the control

vector. For random search methods such as Genetic Algorithms or Pattern Search, the constraints work differently. There they work as checkpoints, and if the path breaks any rule at any time, the simulation stops immediately and that control set is rejected. Only those control actions that follow all the rules are kept for the next round of testing. To implement the pattern search method, the vehicle trajectory is divided into multiple slots. The number of slots is a trade-off between computation time for fewer slots, whereas the number of slots has to be large enough to capture the complete flight dynamics. The slots are depicted in Figure 4.3.

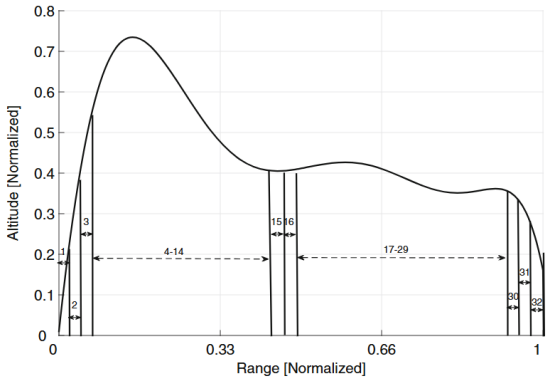


Figure 4.3: Trajectory partitions [51].

4.5.5. Assessment of Numerical Methods

In Table 4.1 the trajectory optimization methods discussed in the previous sections are summarized. This gives a comparative overview of their characteristics, strengths, limitations and suitability for the HGV trajectory optimization.

Table 4.1: Comparison of numerical trajectory optimization methods

Method	Description	Advantages	Disadvantages	Assessment
Direct Shooting Method	Integrates the system forward from initial guess of control inputs	• Simple to apply • Clear structure	• Sensitive to initial value • Numerical instability in long trajectories • Can fall into local minima	Simple but limited accuracy. Struggles with long-duration HGV trajectories
Multiple Direct Shooting Method	Divides trajectory into segments, where each is solved and put together	• Improved version of direct shooting • Better stability than direct shooting • Numerical errors do not accumulate as much	• Higher computational costs • Still requires good initial guess	More stable than direct shooting for long-duration problems

Method	Description	Advantages	Disadvantages	Assessment
Collocation Method	Polynomial interpolation between discretized nodes	• More stable than shooting methods • Whole trajectory optimized at once • Smoother and more accurate trajectory • No numerical integration errors accumulate • Can be transcribed using pseudospectral methods	• Requires advanced solvers	Effective for constrained HGV optimization and widely used in literature
Gaussian Pseudospectral Method	Spectral collocation using Gaussian quadrature	• One of the most convenient tools for HGV reentry • Handles many constraints (aerodynamic load, dynamic pressure, heating rate) • Ideal for nonlinear HGV dynamics	• Complex setup	Demonstrated as a convenient method for HGV trajectory optimization
Differential Inclusion Method	Optimizes over sets of possible state transitions instead of explicit control inputs	• Much faster computation than collocation • Avoids explicit control definition	• Lacks explicit control history • Less precise control reconstruction	Efficient for rapid computation. Less suited for detailed control strategy design
Indirect Method	Derives optimality via Pontryagin's Maximum Principle and costates	• High accuracy results • Theoretically optimal • Solutions satisfy optimality conditions	• Very difficult for complex problems • Requires good initial guess for costates • Small convergence space • Necessary optimality conditions must be derived analytically	Difficult to apply, but Grant et al. successfully demonstrated HGV optimization with validation against modern tools

Method	Description	Advantages	Disadvantages	Assessment
Dynamic Programming	Solves optimization by breaking problem into discrete stages and finding optimal continuous input at each state by solving smaller subproblems recursively	• Simple calculation theory • Relatively high accuracy	• Requires large memory space	Used in early stages of vehicle trajectory optimization but memory requirements limit applicability
Genetic Algorithms	Population-based global search based on natural selection	• Robust • Not sensitive to initial values • Good performance with complex nonlinear systems • Can obtain global optimum faster	• Suffers from slow convergence rates • Struggles to meet tight nonlinear constraints	Extensively applied in reentry trajectory optimization and demonstrated effective global search capabilities
Particle Swarm Optimization	Swarm-based search using social and cognitive update	• Simple nature • Rapid convergence • Efficient, reliable, and accurate (demonstrated in literature)	• Easy to fall in local optima • Low convergence rate	Regarded as a prospective optimization algorithm and successfully applied to minimize heat generation
Pattern Search	A direct search-based and gradient free algorithm	• Does not require derivative information • Has potential for highly constrained problems • Can reach global minima without gradients	• Not extensively investigated for trajectory optimization	Has potential for highly constrained HGV problems but requires further investigation

4.6. Tools To Use

This section presents software tools that implement the optimization techniques described above. The focus is on DIDO, CasADi and GPOPS-II, highlighting their core features, advantages and relevance to HGV trajectory analysis.

4.6.1. DIDO

The DIDO software package emerged in 2001 and is a MATLAB toolbox used for optimal control problems. DIDO is based on pseudospectral optimal control theory and was at first a generic nonlinear programming solver. From then on it has evolved to a new approach that is known as DIDO's algorithm [59]. The optimal control problem can be defined in MATLAB using DIDO's syntax. The formulation includes:

1. State variables $x(t)$
2. Control variables $u(t)$

3. Equations of Motion
4. Constraints
5. Objective function

Instead of solving equations continuously, DIDO uses the pseudospectral methods to pick key points in time and approximate the entire solution. It uses *Legendre-Gauss-Radau collocation* to approximate the states and controls at the specific nodes. The differential equations then become algebraic equations at these nodes. Unlike traditional solvers, DIDO does **not** need an initial guess for the control input. This reduces bias and manual adjustment efforts. DIDO sends the discretized problem to a non-linear programming solver which finds the optimal control trajectory that satisfies the constraints and minimizes the given objective function.

An advantage of the DIDO package is the user-friendly interface for MATLAB, which gives a minimalistic approach [50]. DIDO has been used by NASA to conduct a spin with zero-propellant of the entire International Space Station.

4.6.2. CasADI

CasADi offers a versatile optimization framework created especially for non-linear dynamic systems. This is an open-source software platform for numerical optimization, facilitating the solution of non-linear programming problems [60]. CasADi allows one to define optimization problems symbolically, which means that the equations can be expressed mathematically. Additionally, the software uses algorithmic differentiation that computes exact derivatives of functions without having to manually differentiate or use numerical approximations [61]. After defining the problem, CasADi can compile the symbolic expressions into an efficient numerical code. The solver that CasADi can use is IPOPT. This stands for Interior Point OPTimizer which is designed for solving large-scale non-linear programming problems. CasADi has differentiated the model to generate the information required by IPOPT. Manually deriving and coding complex derivative expressions does not need to be done anymore. Afterwards, IPOPT searches for the optimal solution while ensuring that all the constraints are met.

CasADi has been used in trajectory optimization studies of a hypersonic glide vehicle. According to Chai et al. [39], CASADI was able to provide significantly smoother bank angle profiles compared to their developed approach.

4.6.3. GPOPS II

GPOPS-II is a MATLAB software that is used to solve multi-phase optimal control problems [62]. This means that the entire problem has been split into different phases, each of which has its own equations and goals. For the hypersonic glide vehicle, the trajectory can be split up into the different flight phases and GPOPS II solves these phases together but treats them separately, which ensures a smooth transition between them. The software uses Gaussian quadrature collocation methods. Here, the Legendre-Gauss-Radau quadrature orthogonal collocation method is used where the continuous-time optimal control problem is transcribed into a nonlinear programming problem.

Each of the phases has its own state and control variables, dynamics, constraints, and optimization objectives. However, they are linked together through event constraints; this is where one phase goes over in the other. This is shown visually in Figure 4.4. As stated above, GPOPS-II uses direct collocation, which means that the state and control variables are discretized at selected points in each phase. This can be converted to a non-linear programming problem, which is solved using NLP solvers such as SNOPT and IPOPT. GPOPS-II uses the hp-adaptive Gaussian quadrature collocation methods. In hp-adaptive mesh refinement, the h-method increases the number of mesh intervals, and the p-method increases the degree of polynomials within an interval. The hp method combines both strategies to adaptively refine the mesh based on error estimates. This allows the software to place more points in regions where the solution changes rapidly.

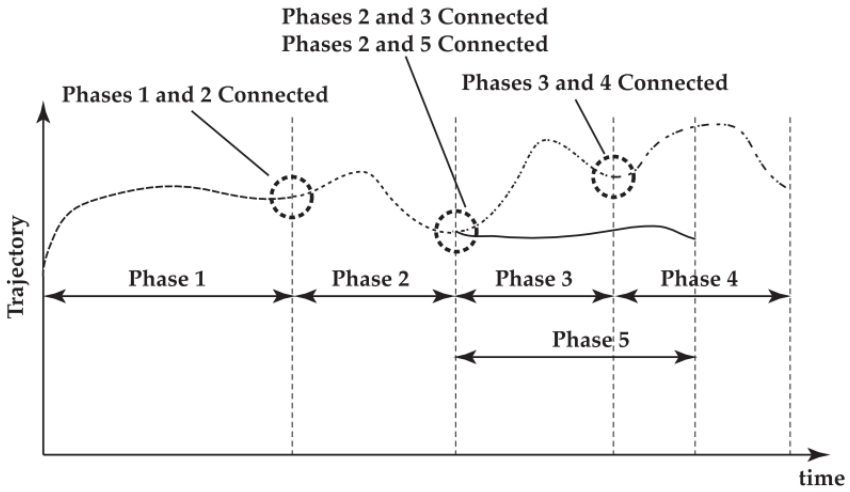


Figure 4.4: Linkages of the multiple phase optimal control problem [62].

GPOPS-II has been applied to the entry of reusable launch vehicles. Here, the goal was to maximize the cross-range during atmospheric entry. This vehicle has three different phases, namely the entry from orbit, hypersonic glide, and final approach. The constraints were dynamic pressure limits, maximum heating rate, and load factor limits. GPOPS-II has been shown to be able to compile an accurate solution using a coarse mesh [62]. The GPOPS II solution has been compared with a solution using the Sparse Optimization Suite (SOS) software, which has been created by Betts [63]. It was found that the two solutions are visually indistinguishable.

GPOPS-II can be divided into four foundation scripts, namely the main script, the DAE script, the connect script, and a cost script [64]. In the main script, the framework of the case is written. This is set into the number of sections, which is based on the number of phases of the problem. Each of these phases has nodes, and the nodes are the points that will be used in the NLP solver. The DAE script stands for Differential Algebraic Equations. This script contains the dynamics of the optimal control problem. These include the Equations of Motion. Each state requires a first-order differential equation to be solved by the NLP solver. The construct script has the final state of the previous phase and the initial state for the phase that follows. From one state to the other state, equations have to be written that perform the transformation of these states. Finally, the cost script will create a cost function for the optimal control problem with the appropriate objective function. It is possible to have a different cost function for each separate phase [64].

Tawfiqur et al. [65] have performed a trajectory optimization using the Gauss pseudospectral method for a hypersonic vehicle trajectory and the application was carried out in the GPOPS-II software program.

4.7. NLP Solvers

There are several non-linear programming solutions used in MATLAB. The three most common are Interior Point OPTimizer (IPOPT), Sparse Nonlinear OPTimizer (SNOPT) and Nonlinear Programming SOLver (NPSOL). Their performance characteristics and compatibility with direct methods are discussed in the context of HGV trajectory optimization problems.

4.7.1. IPOPT

IPOPT is an open source nonlinear solver based on an interior-point method for large-scale nonlinear programming. The interior point method transforms the constrained problem into a series of unconstrained problems. The interior-point then uses a barrier function, which gives more 'resistance' when the optimizer comes close to the limit of the constraint. The interior point method stays inside the allowed values without hitting the constraints abruptly. This makes it faster and more stable for solving large-scale trajectory optimization problems.

This solver is particularly good for problems with a great number of variables and constraints, making it ideal for *direct collocation methods* in trajectory optimization. One of the key strengths is the ability to exploit distributed matrix structures, which significantly increases computational efficiency. Therefore, IPOPT is used for problems where the full state and control trajectories are discretized into a set of collocation points.

4.7.2. SNOPT

SNOPT is an optimization solver that is based on sequential quadratic programming, which is a method that approximates the nonlinear problem through a sequence of quadratic subproblems. SNOPT focuses on active-set methods, which makes it efficient for problems where the constraint set evolves gradually over iterations. Active-set method is a way to solve optimization problems by guessing which constraints are exactly equal at the solution, and then solving the problem as if only those constraints matter. SNOPT is well suited for trajectory optimization problems with sparse Jacobians and Hessians, which is often the case when solving multiphase optimal control problems. The Jacobian represents the first-order partial derivative. A sparse Jacobian means that most of the elements in the Jacobian are zero. The Hessian matrix represents the second-order partial derivatives. It captures how the gradient of the function changes, providing information on the curvature of the cost function. Thus, Jacobians are used to evaluate constraint sensitivities, and Hessians are used to model curvature for faster convergence.

4.7.3. NPSOL

NPSOL is also a solver based on sequential quadratic programming, which is particularly effective for medium-sized problems where constraints and derivatives are well-defined. However, it is less scalable than SNOPT for large, sparse problems. NPSOL uses quasi-Newton methods to approximate the second-order derivative, which reduces computational cost compared to full Hessian evaluations. NPSOL has been used in indirect methods, where the optimal control problem is formulated as a two-point boundary value problem.

4.8. Current Limitations

Current implementations in literature have some limitations that remain areas for this research. Most notable, existing methods such as those by He et al [66], have not fully addressed the inner boundary of the footprint and the optimization of the complete footprint. In addition, many implementations use the simplification of equilibrium glide, which will not capture the full maneuverability of the HGV.

5

Initial Testing and Framework Setup

5.1. Optimization in the 2D Planar field

In order to start this research, small steps have to be taken so that everything can be explained and verified. To do this, the model has first been simplified to a 2D planar field model. This means that the HGV can only glide in the longitudinal direction, and the effect of using the bank angle is simply a reduction in lift. The simplified equations of motion used in the model are the following [38]:

$$\begin{aligned}
\dot{V} &= -\frac{D}{m} - g\sin(\gamma) \\
\dot{\gamma} &= \frac{L}{mV}\cos(\sigma) - \frac{g\cos(\gamma)}{V}\left(1 - \frac{V^2}{V_c^2}\right) \\
\dot{h} &= V\sin(\gamma) \\
\dot{s} &= -\frac{V}{R_E + h}\cos(\gamma)
\end{aligned}$$

where the parameters are given in Table 5.1.

Symbol	Description
V	Velocity
γ	Flight path angle
h	Altitude above Earth's surface
s	Downrange distance
D	Drag force
L	Lift force
m	Vehicle mass
g	Gravitational acceleration
σ	Bank angle
V_c	Circular orbital velocity
R_E	Earth's radius

Table 5.1: Parameter definitions for the equations of motion

In Figure 5.1 a reentry vehicle is shown with all the forces that work on it. The vehicle has a mass m and no thrust. The forces that work on the vehicle are the lift L, the drag D, and the weight $W = mg$ [38]. This is shown in Figure 5.1.

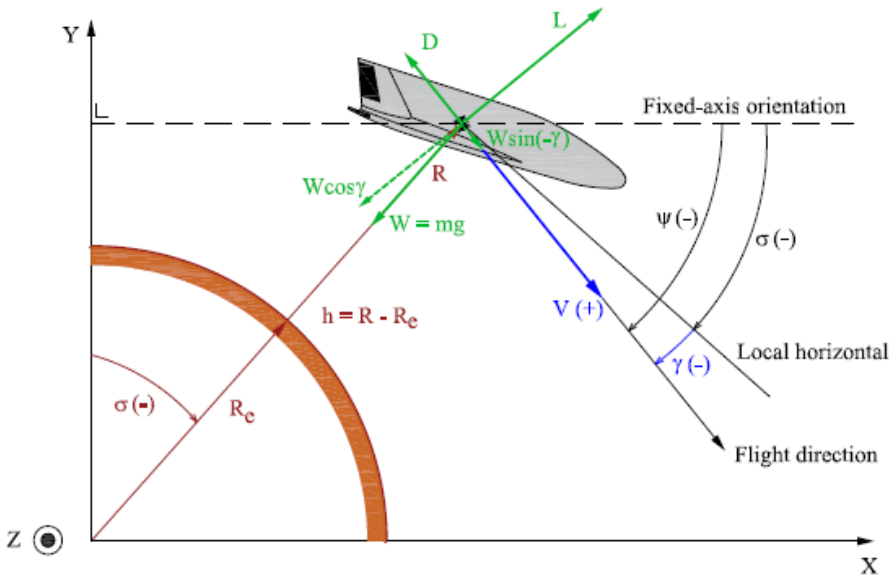


Figure 5.1: Two-dimensional motion of a HGV [38]

For this model, the Coriolis effects have not been taken into account and no aerodynamic data has been provided yet. In this first step, there is a fixed lift coefficient (C_L) and drag coefficient (C_D) and the gravitational acceleration is a fixed constant.

As an atmospheric model, the exponential atmosphere is used. Here the density is calculated as follows:

$$\rho = \rho_0 e^{-h/H_s}$$

where H_s is the atmospheric scale height chosen of 6950m and ρ_0 is the density at sea level equal to $1.225 \frac{kg}{m^3}$.

5.1.1. Constraints

For this model, three limiting constraints have been taken into account. These are the load factor limit, the heat flux limit, and the equilibrium glide constraint. The final constraint is not a hard boundary, but a visualization of when the vehicle will start to skip. Above this line (in the Mach-altitude plane) the vehicle will not have sufficient lift for sustained flight and will therefore fall towards the ground. In the event that the vehicle is below this line, it will produce too much lift and result in a skipping flight. The formulas for the velocities that indicate the limits at each altitude are given below and the parameters given in Table 5.2:

$$V_{\text{eq}} = \sqrt{\frac{V_c^2}{\frac{0.5 \cdot \rho \cdot C_L \cdot S_{\text{ref}} \cdot V_c^2}{W} + 1}}$$

$$V_{qc} = V_c \left(\frac{RN^n \cdot q_{\text{max}} \cdot \left(\frac{\rho_0}{\rho} \right)^{1-n}}{c_1} \right)^{\frac{1}{m_c}}$$

$$V_{gc} = \sqrt{\frac{2 \cdot W \cdot g_{\text{max}}}{S_{\text{ref}} \cdot \rho \cdot \sqrt{C_D^2 + C_L^2}}}$$

Table 5.2: Variable definitions used in the equations

Symbol	Description
V_{eq}	Equilibrium glide speed
V_{qc}	Maximum heat flux speed
V_{gc}	Maximum load factor speed
V_c	Circular orbital velocity
ρ	Atmospheric density
C_L	Lift coefficient
C_D	Drag coefficient
S_{ref}	Reference area
W	Weight
RN	Nose radius
n	Chapman density exponent
q_{max}	Maximum dynamic pressure
ρ_0	Sea-level atmospheric density
c_1	Chapman heat flux coefficient
m_c	Chapman velocity exponent
g_{max}	Maximum allowable g-load

5.1.2. fmincon optimization

To perform trajectory optimization in the simplified 2D planar model, the MATLAB function *fmincon* from the optimization toolbox is used. fmincon is a gradient-based solver for non-linear programming problems. It allows minimizing an objective function while satisfying a set of constraints.

For this simulation, the goal is to find the best control strategy, in this case a sequence of bank angles over time, that guides the vehicle through the atmosphere without exceeding constraints while maximizing the downrange distance. Optimization is performed using a blockwise control approach; the trajectory is divided into discrete time segments, and a constant bank angle is assigned to each. The optimization algorithm adjusts these angles to achieve the best possible flight path.

At each iteration of the optimization, the vehicle's motion is simulated using numerical integration of the equations of motion. The aerodynamic forces, gravity, and the vehicle's orientation affect how the velocity, altitude, and position evolve over time. By altering the bank angle profile, the optimizer can influence how the vehicle trades energy between speed, altitude, and distance traveled.

fmincon works by adjusting a set of control variables to improve the objective function value while ensuring that all the constraints are respected. It uses gradient-based methods, which means that it uses derivatives of the objective and constraint functions in order to decide how to change the variables.

Using fmincon, the constraints are implemented using a separate nonlinear constraint function, called nlcon. This function ensures that the trajectory remains within the entry corridor by enforcing the limits. At each iteration, the optimizer evaluates the trajectory that corresponds to a bank angle profile and computes the constraint values. This formulation ensures that there is strict compliance with the path constraints, instead of penalizing violation in cost functions, which will be used in other optimization methods.

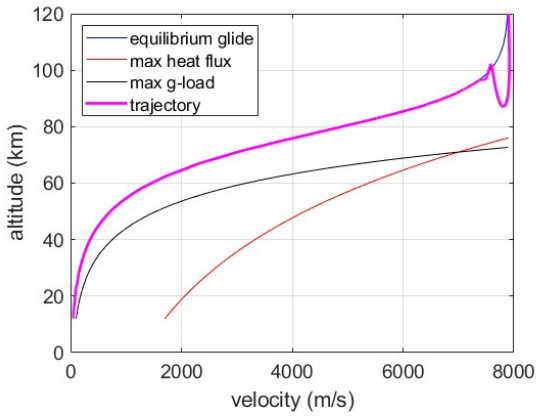
While fmincon is a powerful optimization tool in MATLAB, applying it to trajectory optimization problems is not without difficulties. One of the key challenges is that fmincon is a local optimizer. This means that it relies on gradients to iteratively improve the solution, starting from an initial guess. It does not explore the global landscape of possible solutions, but instead will follow the steepest improvement path from the starting point. As a result, it often converges to local minima, which are solutions that appear optimal in a small neighborhood, but are far from the global best.

For the trajectory optimization of a hypersonic glide vehicle, the cost landscape is highly nonlinear due to the interactions between aerodynamics, gravity, and constraints. This makes it easy for fmincon to choose suboptimal solutions, especially when the initial guess is poorly chosen.

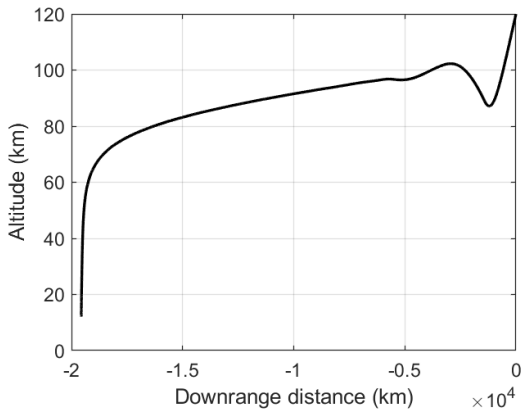
Optimization to Equilibrium Glide line

As a test case, an optimization was performed in which the objective was to steer the vehicle toward the equilibrium glide line. This trajectory represents a flight path where the vertical acceleration is zero, with aerodynamic lift balancing both the component of gravitational force perpendicular to the velocity vector and centrifugal effects. This ensures that the vehicle will not experience a skipping flight and is used as a reference for energy-efficient gliding.

The cost function used in this case did not aim to reach a target point but instead minimized the total deviation from the equilibrium glide velocity at each altitude along the flight path. By simulating the trajectory corresponding to a given bank angle profile, the optimizer could evaluate how closely the resulting velocity followed this ideal line. The optimization is shown in Figure 5.2a and the resulting downrange in 5.2b.



(a) Trajectory optimization to equilibrium glide line.



(b) Downrange distance flown for equilibrium glide condition.

Figure 5.2: Trajectory optimization equilibrium glide condition.

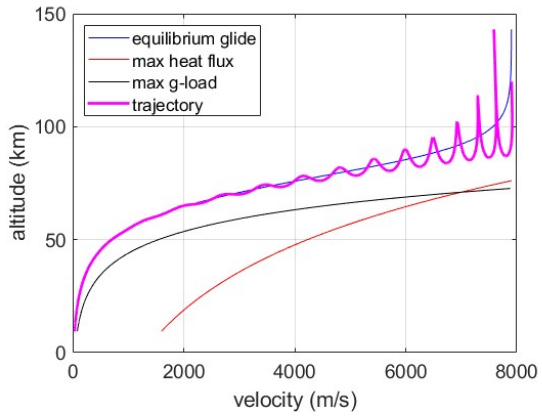
Optimization for maximum range

In a separate test, the optimization objective was to maximize the total downrange distance traveled by the vehicle. Instead of targeting a reference trajectory like the equilibrium glide line, this formulation optimizes the horizontal displacement over the Earth's surface.

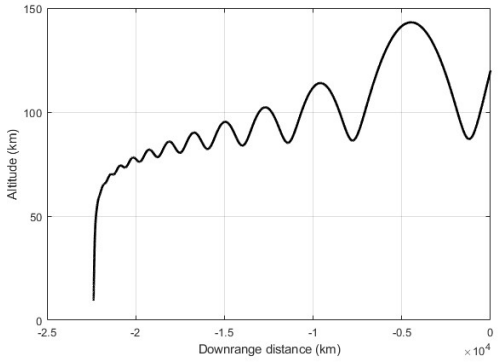
The optimization again adjusted a blockwise bank angle profile, simulating the trajectory for each possible solution. The cost function was defined to minimize the negative downrange distance at the final time, which effectively maximizes it.

An interesting and consistent result was obtained from this formulation. In the simplified 2D planar model, the optimal solution was achieved by maintaining a bank angle close to zero throughout the entire flight. This outcome makes physical sense; with zero bank angle, the vehicle generates maximum lift in the vertical plane, which can be used to sustain long periods of shallow glide or induce skipping flight. Both behaviors allow the vehicle to remain at high altitudes with lower air density, where drag is minimized and range is maximized.

This result gives a good insight into the 2D maximum-range problem: without cross-range maneuvering or heading constraints, the most efficient path involves flying with a bank angle close to zero. The trajectory with the constraints is shown in Figure 5.3a and the downrange distance flown in Figure 5.3b.



(a) Trajectory optimization for maximum range.



(b) Downrange distance flown for maximum range.

Figure 5.3: Trajectory optimization maximum range.

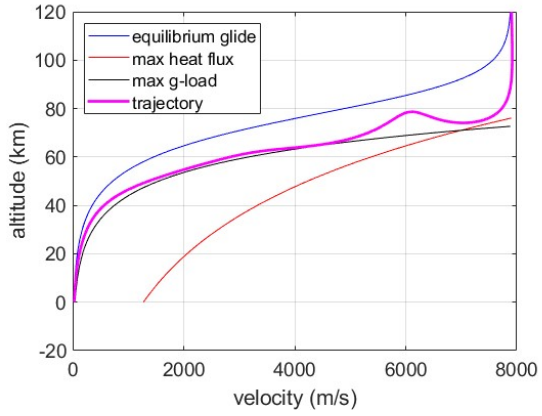
Comparing Figure 5.2b and Figure 5.3b it can be seen that if the vehicle exhibits skipping behavior, it will obtain a further range. Therefore, the equilibrium glide condition can be considered as a simplification of the problem, and allowing the HGV to skip will result in a larger range.

Optimization for minimum range

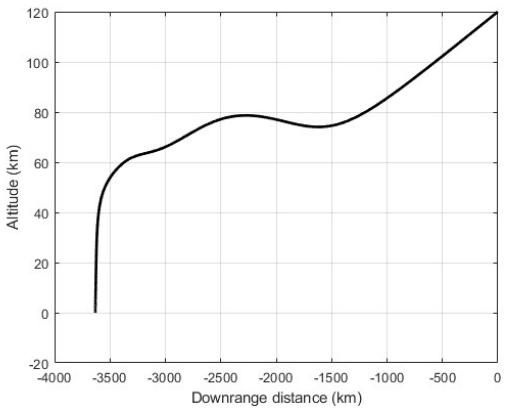
In addition to maximum-range optimization, a trajectory was computed that minimizes the horizontal range traveled. This test simulates an extreme maneuver that is aimed at achieving the steepest and most direct descent possible while still respecting the constraints.

The solution presents a trajectory that consistently applies high bank angles, which greatly reduces the effective lift. This causes the vehicle to drop quickly through the atmosphere, minimizing horizontal motion and maximizing the deceleration.

As shown in Figure 5.4a, the trajectory comes close to the heat flux and load factor lines without violating them, confirming that the optimizer pushed as far as possible towards a vertically steep path. In contrast to maximum-range trajectories, this solution demonstrates how aggressive banking can rapidly dissipate energy which will result in minimal horizontal distance, shown in Figure 5.4b. It must be noted that minimum range is in this case in a straight line. For the footprint for actual minimum range, an optimal turning method must be designed that reverses the flight direction and from that point on maximizes range in that direction.



(a) Trajectory optimization for minimum range.



(b) Downrange distance flown for minimum range.

Figure 5.4: Trajectory optimization minimum range.

5.2. Increasing the Speed of the Simulation in MATLAB

This section presents practical techniques for improving simulation efficiency. It discusses the use of MATLAB Parallel Toolbox, C++ MEX functions, and CUDA GPU acceleration to reduce computational time during optimization.

The trajectory optimization and simulations can be performed in MATLAB. MATLAB is known for its efficient handling of vectorized operations and built-in functions, but it performs poorly when executing explicit loops, especially for large-scale simulations. In order to address this, several alternatives have been explored to improve the execution speed. A test script was written in MATLAB to compute the Euclidean distance between 20,000 3D points, resulting in 400 million pairwise calculations to simulate a computationally expensive task. The plain MATLAB version of the function took **42 seconds** to complete.

5.2.1. MATLAB Parallel Toolbox

The first approach used MATLAB's Parallel Computing Toolbox, specifically the *parfor* construct. This enables automatic parallel execution across CPU cores, integrates easily with existing code, and requires minimal changes. However, it introduces overhead: all variables used inside a *parfor* loop must be copied to each worker (a separate MATLAB instance), and additional time is needed to divide iterations and initialize workers. In this case, the overhead outweighed the benefits, and execution took **51 seconds**, which is slower than a regular for loop, even with 400 million iterations.

5.2.2. MEX Functions with C++ and OpenMP

The next approach used a MEX function: a C++ function compiled into a shared library that MATLAB can call directly. Compiled C++ code runs much faster than interpreted MATLAB code, especially for large computations or tight loops. The MEX interface is very thin and allows direct access to MATLAB's memory through pointers, so even large matrices can be passed without copying between MATLAB and C++. This makes it both efficient and powerful for performance-critical tasks.

Additionally, OpenMP can be added in C++. This allows loop parallelization with minimal code modifications. This gives multi-core CPU parallelism in C++, which is faster and more memory-efficient than MATLAB's *parfor*.

A C++ version of the distance function was written and compiled using MEX, which reduced the computation time to **4.5 seconds**. Adding OpenMP to the C++ code allowed for multi-core parallelism with minimal effort. With OpenMP, the execution time was further reduced to **2.2 seconds**.

5.2.3. GPU Acceleration with CUDA

The function has also been implemented using CUDA to offload computations to the GPU. CUDA enables thousands of lightweight GPU threads to run in parallel, making it ideal for large numbers of independent calculations. This approach introduces overhead due to data transfers between CPU and GPU memory and is typically only faster for large datasets. On an older, low-end GPU, the CUDA version completed the computation in **1.7 seconds**. Performance is expected to improve significantly on newer GPUs. While CUDA adds coding and debugging complexity and is not suitable for all problems, the speedup can be substantial when applicable.

5.3. Simulation Framework MATLAB 3D Equations of Motion

As an initial approach to modeling HGV reentry trajectories, a full physics simulation was developed in MATLAB. This implementation modeled complete reentry scenarios using a constant angle of attack and bank angle. The simulation incorporated a 3-DOF point mass model with COESA 1976 atmospheric modeling, spherical Earth equations of motion accounting for rotation effects, and aerodynamic coefficient interpolation from the HGVs database. While this MATLAB environment successfully demonstrated the physics of HGV reentry, the approach was not pursued for the trajectory optimization. Instead, the simulation framework was reimplemented in Python to enable integration with modern reinforcement learning libraries, as is explained in Chapter 6. The theoretical foundations of everything used in this model are extensively explained in Chapter 6.

The trajectory integration continued from the starting condition of speed 6000 m/s and altitude 160 km until the vehicle descended below 5 km, after which the integration was terminated. The evolution of all state variables, namely velocity, flight path angle, altitude, longitude, latitude, and heading, has been plotted in Figure 5.5. The results show the expected energy dissipation behavior. As the vehicle descends, it loses altitude and speed while undergoing heading change due to the high bank angle.

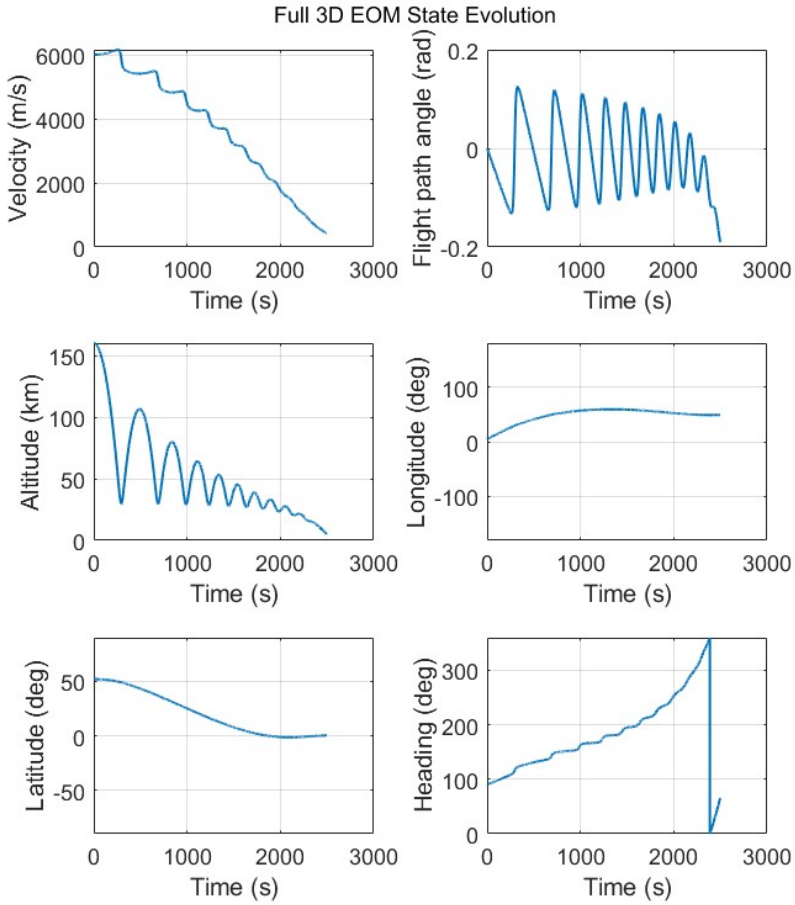


Figure 5.5: Evolution of all state variables for a high bank angle trajectory

This trajectory is also plotted in 3D to see the movement in latitude and longitude, shown in Figure 5.6, and the path over Earth's surface is showin in Figure 5.7.

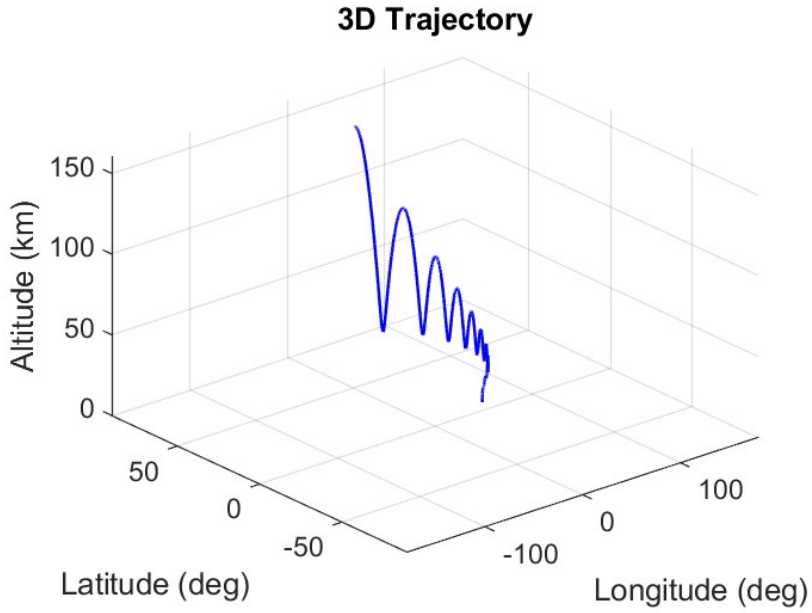


Figure 5.6: Trajectory where the altitude can be seen relative to the latitude and longitude.

Figure 5.7: Trajectory of HGV with bank angle $30°$.

In Figure 5.8 the same trajectory is shown, but from a different perspective. By doing this, the skipping behavior of the HGV can be seen.

Figure 5.8: Skipping behavior of the HGV shown over the Earth's surface.

5.3.1. Early Version Monte Carlo Simulations

Monte Carlo simulation is a method that is used to understand how a system behaves. Many simulations are run, where each time the input is randomly changed. This helps to see the full range of possible outcomes. In this project, Monte Carlo simulations have been used to study how the HGV behaves when the control inputs (bank angle and angle of attack) are changed. By simulating millions of trajectories with different bank angle and angle of attack sequences, the variations in control variables influence on the final footprint can be examined.

It must be noted that this has been an early version of a Monte Carlo simulation, where the actual version including rate limitations and more simulations is given in Chapter 7.

Constant AoA, varying bank angle

Each trajectory begins with the same initial condition, with velocity, flight path angle, altitude, longitude, latitude, and heading specified to represent a reentry scenario. The control input consists of a piecewise constant bank angle profile, where the angle is updated at fixed intervals. For each simulation, the bank angles are randomly sampled from a discrete set ranging between $-90°$ and $+90°$.

Multiple Monte Carlo configurations are considered, each characterized by a different control update rate and number of bank angles. These include coarse control updates (5 bank angles), medium control updates (10 bank angles), and fast control updates (25 bank angles). This variation allows us to see how the control authority changes the reachable footprint.

In this Monte Carlo simulation, an event handling function monitors the violation of the constraint. If any of the constraints are violated in the bank angle sequence, that particular trajectory is ignored and is not plotted on the map. Only trajectories that terminate due to the altitude condition are considered valid. For each of these, the final geodetic coordinates are extracted and stored.

MATLAB's *parfor* construct is used to parallelize the workload of generating the trajectories to decrease the run time. The results are visualized on a world map, which allows for comparison between different control schemes.

The initial sweep of constant bank angles serves as a reference of the complete footprint. Each sweep trajectory uses a fixed bank angle from $-90°$ to $+90°$ in discrete increments and is integrated until one of the constraint thresholds is violated.

Finally, this simulation does not only give insight into the feasible footprint, but also to validate the optimization frameworks. The generated Monte Carlo endpoints can be compared to the results of

trajectory optimization algorithms to assess whether the optimized solution lies within the Monte Carlo envelope or whether optimization explores reachable regions that random control does not.

A Monte Carlo simulation with 3 million runs is shown in Figure 5.9. Here the different colors represent the different control update rate, where red is coarse control update (5 bank angles), green is medium control updates (10 bank angles) and yellow is fast control updates (25 bank angles).

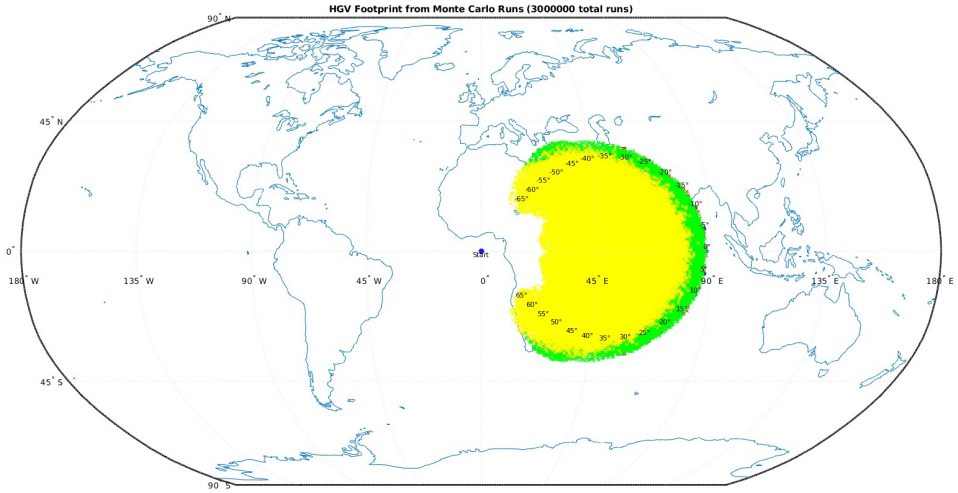


Figure 5.9: Monte Carlo simulation with random bank angle sets and fixed angle of attack.

Interestingly, trajectories with coarse control updates tend to reach significantly farther than those with faster updates. This is because the optimal control for the maximum range is a near-zero bank angle for straight-line flight, which minimizes energy loss. With fewer control updates, each randomly sampled bank angle has a higher influence over the trajectory, and there is a greater chance that the entire profile remains close to zero. In contrast, as the number of updates increases, the likelihood that the bank angle profile deviates from zero grows due to more random changes which results in greater energy dissipation and less range.

To see the influence of Earth's rotation on the footprint, an additional Monte Carlo simulation was performed with an initial heading of $0°$. In this configuration, the Coriolis acceleration causes an asymmetric dispersion of endpoints, with a noticeable bias toward the right-hand side of the nominal heading direction. This is consistent with the expectation that in the northern hemisphere, a rightward deflection occurs because of the Coriolis effect acting perpendicular to the velocity. This simulation is shown in Figure 5.10.

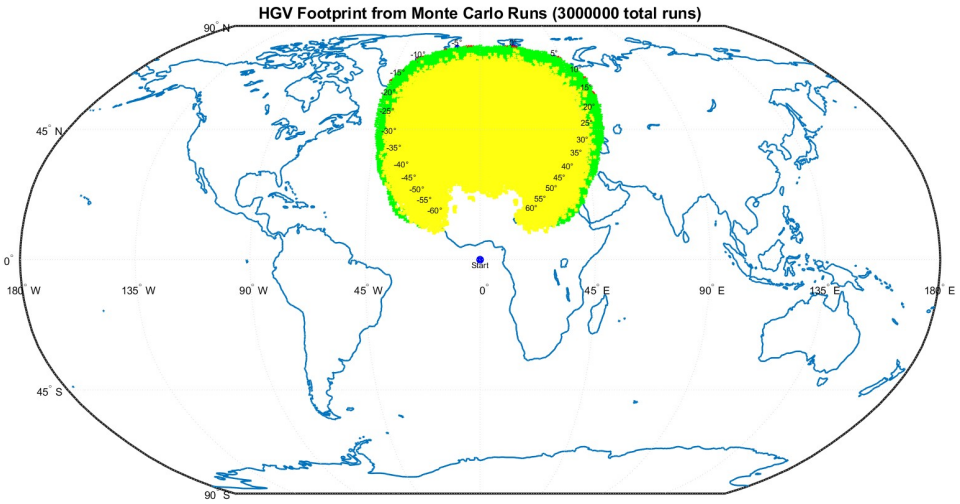


Figure 5.10: Monte Carlo simulation with random bank angle sets with an initial heading of $0°$

Varying AoA and Varying bank angle

In the extended Monte Carlo simulation, where both angle of attack and bank angle vary, each trajectory again starts from identical initial conditions. However, unlike the previous case with fixed AoA, both control inputs are now treated as independent, time-varying parameters. At each control update interval, a new pair of bank angle and AoA values are randomly selected. This introduces significantly more variability in the resulting footprints.

The inclusion of AoA variability leads to a broader and less concentrated footprint, as AoA deviations can cause more severe skipping behavior or energy dissipation. As with the bank only case, an event function monitors for constraint violations, rejecting trajectories that exceed heating, load factor or dynamic pressure thresholds. The footprint with angle of attack and bank angle variability is shown in Figure 5.11.

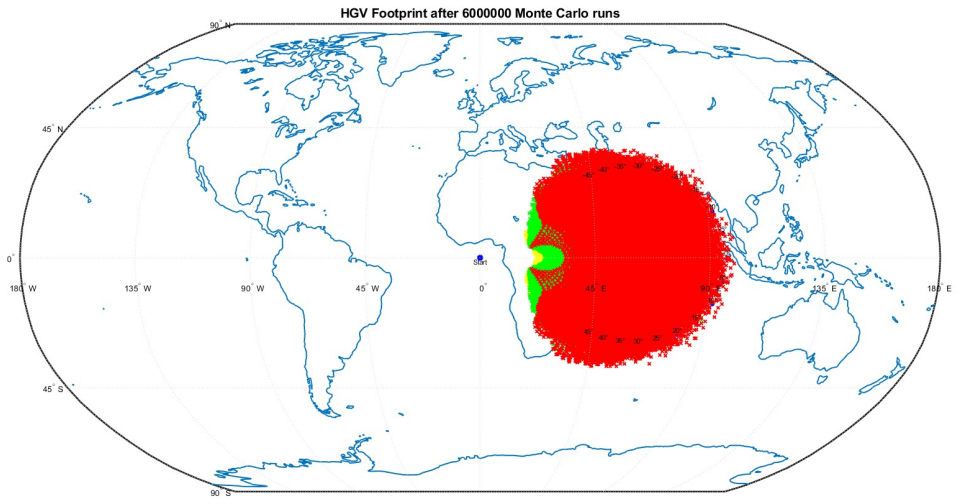


Figure 5.11: Monte Carlo simulation with random bank angle and random angle of attack sets with an initial heading of $90°$.

This extension of the reachable domain indicates that allowing the AoA to vary during flight introduces a favorable variation in the vehicle's lift-to-drag ratio. These variations can locally reduce drag or increase lift, which enables the vehicle to convert energy more efficiently into downrange motion. Therefore, the inclusion of a time-varying AoA not only expands the diversity of trajectory shapes but also demonstrates a positive influence on maximizing range. The footprint with a heading angle of $0°$ is also for this case shown in Figure 5.12, to show the influence of Coriolis.

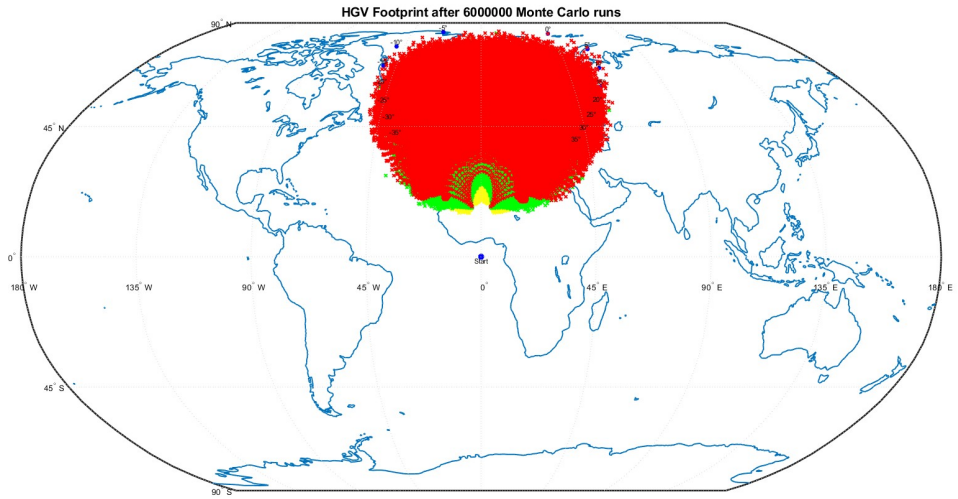


Figure 5.12: Monte Carlo simulation with random bank angle and random angle of attack sets with an initial heading of $0°$ illustrating the Coriolis effect.

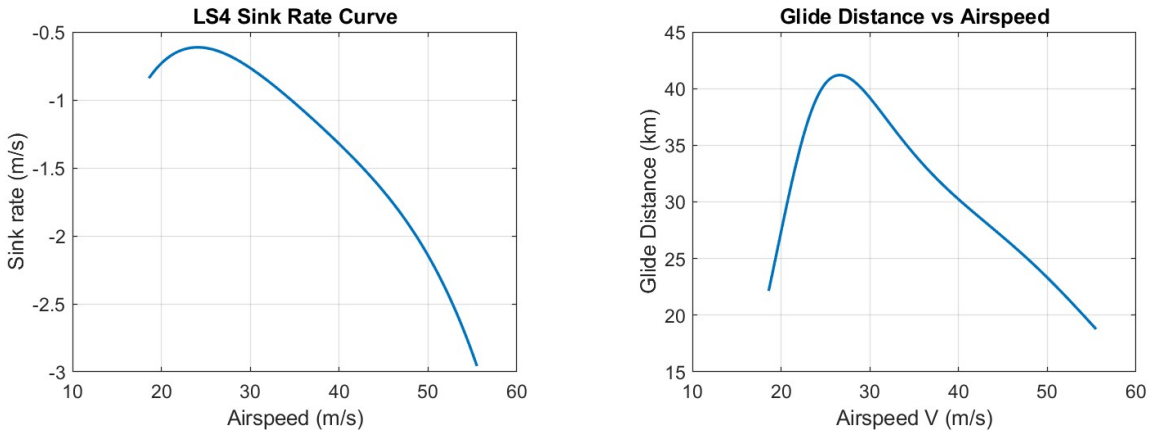
5.4. Optimization Possibilities

In this section the most promising optimization methods will be discussed. Given the complexity of the problem, it is crucial to evaluate multiple approaches, as some may prove unsuitable despite initial promise. After comparative testing, three methods have been selected for further development: Adaptive Simulated Annealing, Gaussian Pseudospectral Method using DIDO, and Reinforcement Learning. The following sections will elaborate on each method in detail.

To fairly evaluate each optimization method, a simple baseline problem has been designed and applied across all approaches. Starting from this controlled scenario, the problem will gradually increase in complexity. This progression allows for precise observation of how the optimizer behaves, and it ensures correct performance at each stage before advancing further.

The baseline problem is defined as follows: a glider starts at an altitude of 1000 m and descends by gliding. For every velocity, a corresponding sink rate is defined from a polar. This polar is shown in Figure 5.13a. The task for the optimizer is to determine the optimal speed profile that will give either the maximum or minimum range. Each of those corresponds to a known solution.

If the optimizer successfully identifies these values, a constraint is introduced: at every point, the speed must be greater than or equal to the altitude divided by 20. This constraint introduces a challenge that is commonly known to cause difficulties for many optimizers, and it can be seen whether the optimizer is able to deal with these difficulties. In Figure 5.13b, the glide distance for every airspeed can be seen. From this it can be seen that the airspeed for maximum distance is approximately 26.5 m/s and the airspeed for minimum distance approximately 55.5 m/s.



(a) Sink rate versus airspeed for LS4 glider.

(b) Glide distance versus speed for the LS4 glider.

Figure 5.13: Sink rate curve and glide distance versus speed for LS4 glider.

This simple optimization problem, including the constraint, will be given to each optimizer and the results will be evaluated in the following sections.

5.4.1. Adaptive Simulated Annealing (ASA) Optimization

Adaptive Simulated Annealing is a global optimization algorithm that has been designed to handle complex problems where traditional optimization methods could fail. It is inspired by the physical process of annealing in materials, where controlled cooling allows a substance to settle into a stable, low-energy crystalline structure. For numerical optimization, this process is mimicked in order to explore the parameter space of a cost function and to locate the global minimum. The cooling process is simulated by the algorithm by gradually lowering the temperature of the system until it converges to a steady state [67].

The core idea of simulated annealing is that it allows for occasional uphill moves. This means that it accepts worse solutions to escape local minima and explore the search space more broadly. At each iteration, a new solution is generated by perturbing the current solution. If the new solution gives a lower cost, it is unconditionally accepted. If it yields a higher cost, it could still be accepted with a probability that decreases as the difference in cost increases, with a scale proportional to the temperature. By accepting these points that increase the cost, the algorithm avoids being trapped in local minima [68].

The likelihood of accepting worse moves will become smaller as the algorithm goes on, just as how a metal cools and the atoms settle into more stable positions. The cooling is controlled by temperature and in the beginning, when the temperature is high, the algorithm is able to explore more freely. Later, when the temperature drops, the algorithm will become more picky. Now it mostly accepts better solutions and will fine-tune the results.

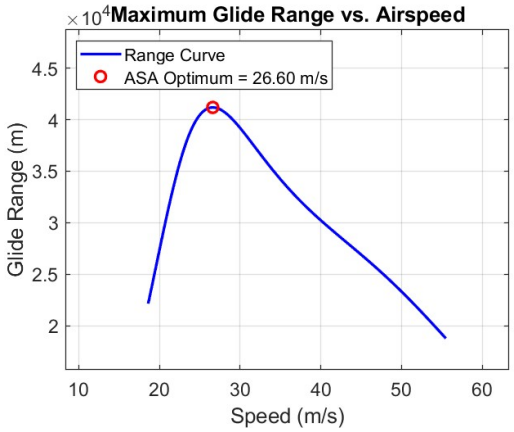
What has been described above is simulated annealing. Adaptive Simulated Annealing (ASA) makes this process smarter. Instead of using one temperature for everything, each parameter has its own temperature assigned. If one variable affects the result a lot, the temperature will cool down faster. This means that the algorithm will become more careful in an earlier stage. For variables that will not affect the outcome a lot, the algorithm is able to explore more loosely.

The fact that the algorithm can learn which variables are sensitive and is able to adapt the search is what makes ASA powerful. In this way, broad exploration and fine-tuned search is balanced and automatically adjusted how it behaves depending on the problem.

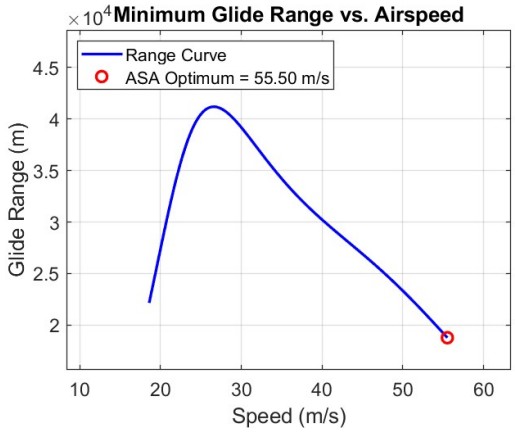
ASA can be used in optimization when the best set of choices is trying to be found, in a complicated search space. For the HGV problem, the bank angle and angle of attack must be chosen at every point in time, where the HGV is subject to constraints. It is not feasible to try every possible combination since there are too many. As seen in previous research, gradient-based methods often fail because the search space has local minima and the optimizer will get stuck in one of these.

Using ASA, the bank angle values at different times are treated as variables. ASA will then randomly try new combinations of angles and will keep the good ones and occasionally try worse solutions to make sure it is not stuck in a local minimum, and will slowly become more selective as it 'cools down'.

Adaptive simulated annealing has been performed on the simple glider problem explained in Section 5.4. The results are plotted in Figure 5.14, where the maximum range and minimum range is shown. As can be clearly seen, the optimizer managed to find the optimal speed for the maximum range and the minimum range.



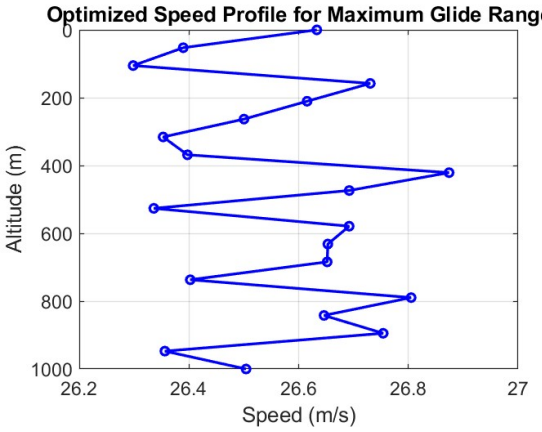
(a) Optimal speed for maximum glide distance found by ASA optimizer.



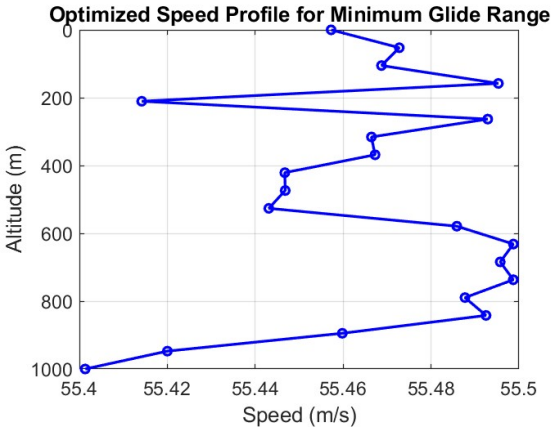
(b) Optimal speed for minimum glide distance found by ASA optimizer.

Figure 5.14: ASA optimization for maximum and minimum range

In Figure 5.15, the speed profile of the speeds for the maximum range and minimum range is also depicted, where the speed oscillates around the optimal speed.



(a) Optimal speed profile for maximum glide distance.



(b) Optimal speed profile for minimum glide distance.

Figure 5.15: Optimized speed profiles for maximum and minimum glide distance.

5.4.2. DIDO Optimization

In this research, one of the possible optimization methods is DIDO. DIDO is a MATLAB-based optimal control solver that applies the Gaussian Pseudospectral Method. The strength of DIDO, and the reason it was considered as one of the options for this thesis, is the ability to handle complex nonlinear dynamics, path constraints and terminal conditions by transforming the continuous problem into a nonlinear programming problem that can be solved with standard optimization techniques. This section provides an overview of how DIDO applies the Gaussian Pseudospectral method and how the

DIDO code works. For a more extensive explanation of the Gaussian Pseudospectral Method, refer to Section 4.3.3, where other applications of GPM in the HGV trajectory optimization are also explained.

As described in Ross [59], setting up a problem in DIDO requires a modular formulation that consists of five key components. These include the dynamics, cost function, event constraints, path constraints and bounds on the search space. Each of these is specified through a separate MATLAB function.

Dynamic Function

The dynamic functions give the equations of motion that describe how the state evolves under the influence of control inputs over time:

$$\dot{x}(t) = f(x(t), u(t), t). \tag{5.1}$$

Cost Function

The cost function defines what has to be minimized during optimization and consists of two parts. First, the Mayer cost, which is a function that is evaluated at the final state and time:

Endpoint (Mayer) cost:

$$E(x_f, t_f) \tag{5.2}$$

The second part is the running cost, called the Lagrange cost. This is a cost that is integrated over the entire time duration:

Running (Lagrange) cost:

$$F(x(t), u(t), t) \tag{5.3}$$

Event Constraints

In the event constraints file, the conditions are specified that must be satisfied at the initial and final times. So these are the initial state conditions, final state conditions and the constraints on initial and final times. These constraints are crucial to ensure that the system starts and ends in the desired configurations. In addition, mission-specific requirements can be included.

Path Constraints

The constraints that must be met along the trajectory are specified in this file. The constraints are evaluated at each collocation point during the optimization. For the goal of this thesis, this will be the maximum load factor, maximum dynamic pressure, and maximum temperature experienced.

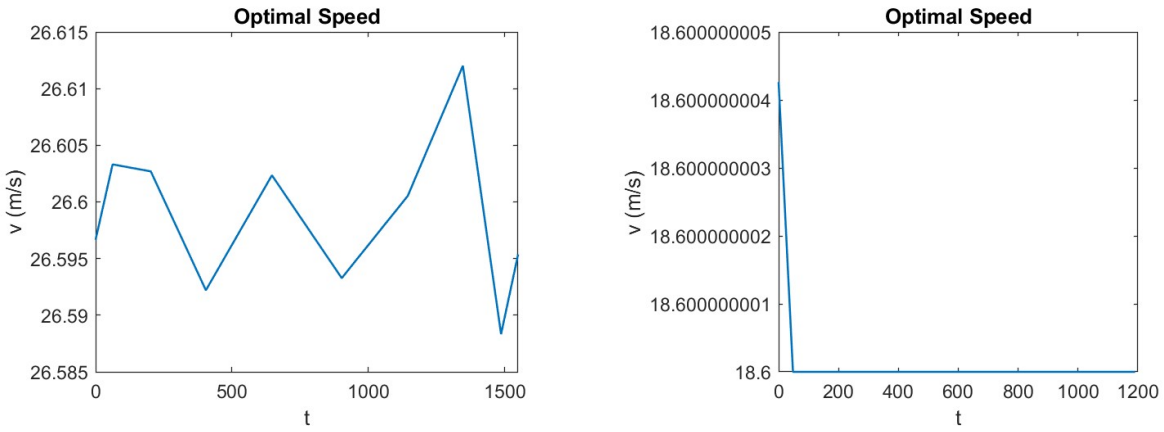
Bounds on Search Space

The final file specifies the limits of the search space. These define the allowable ranges for state variables, control variables, event constraints, and the time. These bounds give DIDO's search a feasible region, which reduces computational load and enforces physical limitations.

5.4.3. Simple Glider Problem Implementation

To validate the DIDO implementation and understand the practical behavior before applying it to the full HGV footprint problem, the simple glider problem is tested. The dynamics, cost function, and boundary conditions were constructed following the DIDO textbook formulations.

DIDO successfully found the optimal speed for the maximum range, as shown in Figure 5.16a. However, for the minimum range, the resulting solution did not yield the expected minimum range, as can be seen in Figure 5.16b. The optimized trajectory converged to a velocity of 18.6 m/s, whereas the actual velocity for minimal range is 55.5 m/s.



(a) Optimal speed profile for maximum glide distance found by DIDO.

(b) Optimal speed profile for minimum glide distance found by DIDO.

Figure 5.16: Optimized speed profiles for maximum and minimum glide distance using DIDO.

This discrepancy indicates that there may be issues either in the transcription of the problem into DIDO's framework, in the scaling, or in the formulation of the cost function in the DIDO structure.

After contacting DIDO support, the cause has not been determined and the possibility of using DIDO has been abandoned.

5.4.4. Reinforcement Learning

Reinforcement learning is a way to teach a system to make decisions by letting it interact with an environment. A full theoretical explanation of how reinforcement learning works is given in Section 6.16. Here the algorithm that is used, Soft Actor-Critic (SAC), is explained as well. In this section, the implementation is elaborated upon on the simple glider control problem.

To apply reinforcement learning on the glider descent optimization, the Soft-Actor algorithm was implemented in Python using the *stable-baselines3* library. The goal of the agent is to control the glider's speed at each time step in order to maximize the horizontal distance while minimizing the vertical loss and avoiding the constraint.

To train the reinforcement learning agent, the glider descent problem must first be formulated as a standardized interface that the learning algorithm can interact with. This is done by creating an environment that consists of a framework that defines how the agent observes the state, what actions it can take, and what rewards it receives for its decision.

In this formulation, the state has a normalized altitude, and the action space allows for continuous speed selection between realistic limits ($18.6 - 55.5$ m/s). As a test of how well the model functions, a rate limiter has been added to prevent abrupt changes in speed. Additionally, a constraint has been applied as explained in Section 5.4.

The reward trains the model on what to do and what not to do. The reward is implemented as follows:

$$\text{reward} = \frac{dx}{|dz|} - 10 \cdot \max\left(0, \frac{z}{20} - V\right) \tag{5.4}$$

This reward function consists of two components that guide the agent's learning. The first term, $\frac{dx}{|dz|}$, represents the glide ratio. This is the horizontal distance gained per unit of altitude lost. By maximizing this ratio, the agent learns to fly efficiently and extend its range. The second term applies a penalty when the constraint is violated. The constraint requires that the velocity V must be greater than or equal to $\frac{z}{20}$ at all times. When this condition is not satisfied, the max function returns the positive difference, which is then multiplied by 10 and subtracted from the reward as a penalty. This penalty discourages

the agent from choosing speeds that violate the constraint while still allowing it to learn the optimal speed profile for maximum range.

The reward in this environment is computed at every simulation step, rather than only at the end of the simulation. This stepwise formulation was essential for stable learning. Initially, experimenting with providing the total glide distance as a terminal reward has been done, but this led to ineffective policies. The agent struggled to associate specific actions with eventual outcomes. By providing immediate incremental rewards based on current glide efficiency and constraint violations, the agent receives direct feedback for each speed decision.

A parallelized environment has been created where the instances can be increased to accelerate learning depending on the computer on which it is run. The advantage of reinforcement learning is that you have many parameters that can be changed to obtain the correct learning model for the problem at hand. The batch size is one of these parameters. The batch size determines how many experiences, thus the state, action, reward, and next state are sampled from the replay buffer at each update. Larger batch sizes can lead to more stable updates due to better gradient averaging but slow down learning by reducing the update frequency. Smaller batch sizes will result in noisier and more frequent updates, which can accelerate learning but can also cause instabilities.

Another parameter that can be tuned is the entropy coefficient. This coefficient controls the trade-off between choosing the best-known action and exploring new actions. Higher entropy encourages the agent to explore more by making the policy more stochastic, and lower entropy makes the policy more deterministic and focused on utilizing the known good actions. With auto-tuning, SAC adjusts the entropy coefficient during training in order to maintain a target policy entropy.

The duration of training can also be adjusted. The time steps are the total number of agent-to-environment interactions used during training. Longer training can allow the agent to converge on more complex policies; however, shorter training can be sufficient for simpler environments or early testing but may not allow full convergence.

During training, the agent is evaluated regularly. After each evaluation, the average performance across 5 simulations is computed. If the new average is better than any previous one, the current model is saved as the best model. Sometimes, the agent gets worse near the end of training due to over exploration caused by high entropy. If only the final model is saved, this could be a suboptimal version. Additionally, the run time can be reduced in the case that the model is not learning anymore. Even if training continues, the best version of the agent that is found is always kept.

The SAC agent successfully learned a descent strategy, in which the maximum velocity change rate had to be satisfied and constraints were not allowed to be violated. This proof-of-concept setup demonstrates the potential of reinforcement learning for continuous trajectory optimization under dynamic constraints. The result of the trajectory, which is fully learned, is shown in Figure 5.17.

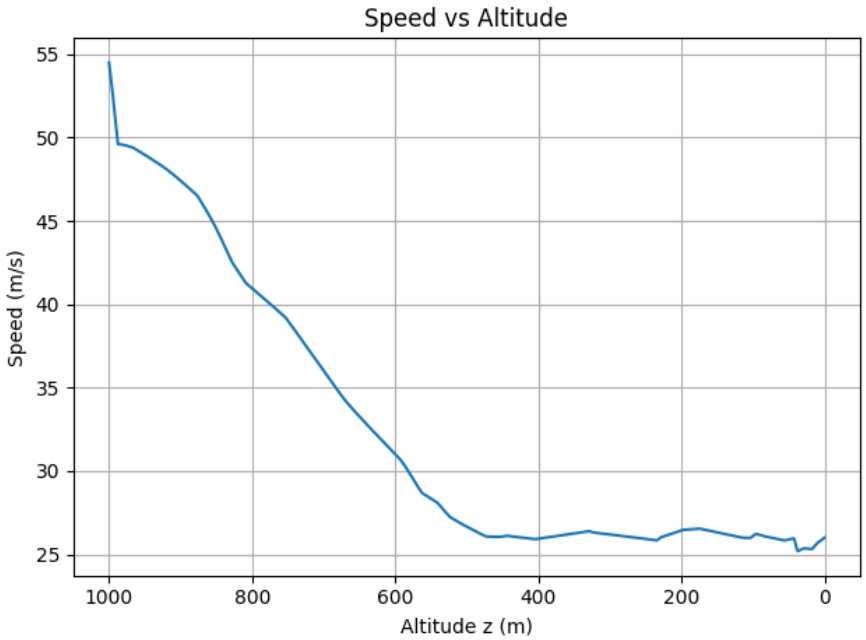


Figure 5.17: Glider descent model made by reinforcement learning

In this model, the start velocity has been given a value of 55 m/s. It can be seen that the model adjusts the speed with the *maximum velocity change rate* to the minimum speed given by the constraint (from 55 m/s to 50 m/s). From this, the model optimizes the speed while adhering to the constraint and finally gets the optimal speed when the constraint is no longer a limiting factor.

Parallelization using GPU

To accelerate the training process, GPU acceleration will be used. This allows for a highly parallelized computation of the model. The CPU handles the trajectory rollouts simulating the environment dynamics and collecting experience data. The GPU's strength is parallel weight calculations. This will be further explained in Section 6.16.5.

6

Discussion & Results

6.1. Introduction

The developed software framework is the computational backbone of the thesis. It provides a complete simulation environment for the trajectory optimization of an HGV, integrating physical modeling, atmospheric conditions, vehicle dynamics, and reinforcement learning control. The code is organized in a modular fashion, where a clear separation is made between the vehicle dynamics model, the reinforcement learning environment, and the scripts for training, evaluation, and footprint generation. This modularity makes sure that the framework is extensible to incorporate new vehicles or new constraint formulations.

The execution flow of the program follows a consistent logic. In the center is the vehicle dynamics model, which contains all the physical properties and equations of motion of the HGV. Surrounding the vehicle model is a reinforcement learning environment, which provides an interface between the physical simulation and the learning algorithms. On top of the model and environment, a set of scripts coordinate the main workflows of the program. These scripts manage the training of the reinforcement learning agents, the execution of direct simulations for validation and the generation of two-and three-dimensional footprints. The next sections describe this architecture in detail beginning with the core physics engine.

This chapter presents a comprehensive description of the framework architecture and the results obtained from the application. The discussion begins with the core physics engine and progresses to the implementation of atmospheric modeling, aerodynamic coefficient interpolation, constraint monitoring, and coordinate transformations. The reinforcement learning implementation is then examined, including the environment design, observation and action spaces, reward formulation, and the Soft Actor-Critic algorithm used for policy learning. The chapter shows the evolution from footprint generation to direct target-point navigation and no-fly zone avoidance. Finally, results are presented including footprint generation for multiple vehicle configuration and trajectory execution examples that show the learned control policies in various scenarios. The chapter establishes both the technical foundations of the approach and the effectiveness in solving the complex trajectory optimization problem.

6.2. Assumptions

This research models the hypersonic glide vehicle as a three-degree-of-freedom point mass moving on a rotating spherical Earth. The Earth is assumed to be a perfect sphere with a constant radius of 6371 km, neglecting oblateness and gravitational anomalies. Gravitational acceleration varies only with altitude according to the inverse square law. The atmospheric properties are obtained from the COESA Standard Atmosphere 1976 model, which provides density, temperature and pressure as functions of altitude. The HGV is assumed to operate under windless conditions in the atmosphere.

The simulation framework has been designed with flexibility in coordinates system selection. The code supports both Earth-Centered-Earth-Fixed (ECEF) coordinates and spherical coordinates, each with

their corresponding equations of motion fully implemented. Users can select their preferred coordinate system depending on the specific requirements of their analysis or computational preferences.

6.3. Module Organization

The code is organized into distinct Python modules, each of which is responsible for specific aspects of the simulation. The core vehicle dynamics and physics simulation are in `hgv.py`, while `coesa-1976.py` handles atmospheric properties. The physical constants are centralized in `constants.py`, spherical geometry calculations for navigation are implemented in `spherical.py` and ECEF coordinate system calculations in `ecef.py`.

The reinforcement learning interface is provided through `hgv_env.py` which wraps the physics simulation in a Gymnasium environment. Gymnasium is a Python library that provides a standardized interface for reinforcement learning environments, defining how agents observe states, take actions, and receive rewards. By implementing the Gymnasium API, the HGV simulation becomes compatible with a wide range of reinforcement learning algorithms, including Stable-Baselines3 used in this work. The `hgv_env.py` module translates the physical HGV simulation into the structure required by reinforcement learning algorithms.

Footprint generation capabilities are split between `footprint.py` for geographic visualization and for three-dimensional trajectory analysis `3d_footprint.py`. An interactive command-line interface in `menu.py` allows the user to configure simulations without modifying the code.

Additional modules support the training and analysis workflow. The `train_hgv.py` module gives multi-process reinforcement learning training using the Soft Actor-Critic algorithm, while `train_hgv_1.py` gives a simpler single-environment variant. Model execution and trajectory visualization are performed by `run_trajectory.py`. Direct physics simulation without reinforcement learning is possible using `simulate.py` which can be used for validation as well.

A fork of this codebase has been developed for no-fly zone avoidance and target point navigation. This fork utilizes the same physics engine from `hgv.py` but requires a different training environment due to modified reward structures and observables.

6.3.1. Program Architecture Documentation

For readers interested in understanding the detailed implementation and execution flow of the framework, complete program architecture diagrams are provided in Appendix A. These diagrams illustrate the complete workflow of the major components of the program using both activity and sequence diagrams.

Appendix A includes visualizations of the footprint generation workflow, showing how the parallel processing architecture distributes 36 trajectory computations across multiple CPU cores. The footprint activity diagram shows the overall execution flow. The footprint orchestration sequence diagram shows the interactions between components during parallel execution, while the footprint simulation sequence shows what occurs inside each worker process during trajectory generation.

The core HGV class architecture shows how the vehicle properties, aerodynamic data, and flight states are organized within the class structure. This is subdivided into the standard core HGV class, the inheritance class that uses C++ acceleration, and the inheritance class that uses ECEF coordinates with or without C++ acceleration. The equations of motion activity diagram shows how the spherical numerical integration advances the simulation and terminates when the vehicle reaches the terminal altitude. The HGV step diagrams illustrate each simulation step, where control inputs are applied, integrating the equations of motion over the time interval, and the checking of violations of constraints.

The reinforcement learning environment is documented through the `HGVEnv` class diagrams and step activity diagrams that show how observations are constructed, the actions are processed with rate limitations, rewards are calculated including constraint penalties, and termination conditions are evaluated. These diagrams show the interface between the physical simulation and the learning algorithms.

Training workflows are shown in activity and sequence diagrams for the parallel training architecture. The `train_hgv` activity diagram shows the complete training setup including environment configuration,

callback configuration for evaluation and early stopping, and the main training loop. The `train_hgv` orchestration sequence diagram shows the coordination between the training process and parallel environments. The `train_hgv` episode sequence diagram gives a view of a single training episode, showing the randomization of initial conditions in training mode, the generation of target points, the step-by-step interaction between the SAC agent and the environment, and the constraint penalty calculation.

Finally, the trajectory execution workflow used for validation and testing is documented in the `run_hgv` activity diagram, which shows the complete execution flow. The `run_hgv` orchestration sequence diagram gives the coordination between the main process and environment setup, while the `run_hgv` simulation sequence diagram illustrates the step-by-step trajectory execution with the deterministic policy.

These diagrams are reference material for understanding the implementation details and can guide future modifications to the framework. They bridge the gap between the conceptual architecture described in this chapter and the actual code implementation, which makes the framework more accessible to other researchers and developers.

6.4. Object-Oriented Design Philosophy

The system is built around a central HGV physics class that contains all vehicle-specific properties, dynamics, and state information. This implements the three-degree-of-freedom point-mass trajectory simulation. This object-oriented approach is different from procedural implementations, where state variables and parameters are typically passed between standalone functions. The object-oriented approach has multiple advantages. Encapsulation makes sure that all vehicle physics, constraints and aerodynamic data come from a single object, which means no global variables or parameter passing between functions is needed. This design choice is also good for modularity, as multiple vehicle configurations can co-exist independently without interference, each maintaining its own aerodynamic database and constraint parameters.

The architecture also guarantees thread and process safety, since independent HGV instances can execute parallel trajectory simulations. This independence is needed for the parallel footprint generation which will later be described. Furthermore, the design is extensible. New vehicle types can be incorporated through class inheritance, which allows to accommodate for future requirements, such as scramjet-equipped variants, without modifying the existing implementation.

6.5. Core Physics Engine

At the foundation of the framework is the HGV physics class, which implements a point-mass model of the vehicle. The class diagram is shown in Appendix A. This is the primary interface for trajectory simulation. The philosophy of the design is that all vehicle specific properties, aerodynamic data and simulation state is contained within a single object. This makes sure no global variables are needed and enables parallel execution, since each HGV instance operates independently without a shared state. Upon instantiation, the class performs several initialization steps:

```
HGV(aero_data, vehicle_constants, alt, v, fpa, lat, lon, hdg, bank, aoa)
```

The class contains the six state variables that represent the HGV's motion and position. These state variables are continuously updated during trajectory integration through the equations of motion. This initialization process begins by preserving the initial state, enabling the simulation to be reset to the same conditions without reconstructing the object. This is essential for reinforcement learning, since thousands of episodes must be run from consistent starting points.

The JSON file of vehicle constants contains parameters that remain fixed throughout a simulation. These include the vehicle mass, the reference area used in aerodynamic calculations, and the operational limits that define the flight envelope, namely the maximum allowable load factor, maximum dynamic pressure, and maximum stagnation point temperature. These values are loaded once during initialization and stored as immutable instance attributes. This ensures consistency throughout the simulation.

Control inputs are managed separately from the state variables. The bank angle represents the roll orientation of the vehicle and controls the lateral component of the lift. Positive bank angles correspond to right wing down orientation. The angle of attack determines the pitch attitude of the vehicle relative to its velocity vector and influences the lift and drag coefficients.

Simulation management is done using internal variables. A Boolean flag is used to indicate whether the simulation has reached the terminal condition, which prevents further integration attempts. Additionally, a flight time counter tracks the elapsed time and the terminal altitude threshold defines when the integration should stop. This represents the altitude at which the HGV would transition to the terminal landing phase. This could also be replaced with a minimum Mach number, which is a design choice.

6.6. Aerodynamic Data

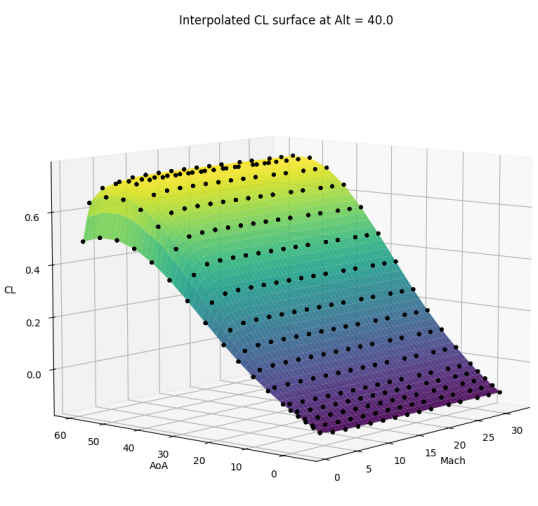
Aerodynamic coefficients for hypersonic vehicles cannot be expressed through analytical formulas. The shock structures are too complex, boundary layer interactions are present and the temperatures are very high in hypersonic flow for a mathematical description. Instead, the aerodynamic data must be generated. The aerodynamic model is therefore constructed from tabulated data, imported from a CSV file, which specifies the lift coefficient C_L, drag coefficient C_D and stagnation point temperature T as functions of altitude, Mach number and angle of attack. As a result, a three-dimensional interpolation is required, over Mach, angle of attack and altitude for both C_L and C_D.

In this work, the aerodynamic coefficients are obtained using FOSTRAD, which stands for Free Open Source Tool for Re-entry of Asteroids and Debris [69]. This is a simulation tool that gives an estimation of aerodynamics and aerothermodynamics of an entry object in rarefied hypersonic flow by using the local panel formulation. The method is based on the division of the vehicle surface into many small panels. For each panel, the flow properties are estimated and then combined to obtain the overall aerodynamic coefficients. In the continuum regime, FOSTRAD uses simple engineering models based on the radius of the surface, whereas in the free-molecular regime it applies flat-plate approximations. To handle the transition between these two extremes, a bridging function is used based on the vehicle's nose radius.

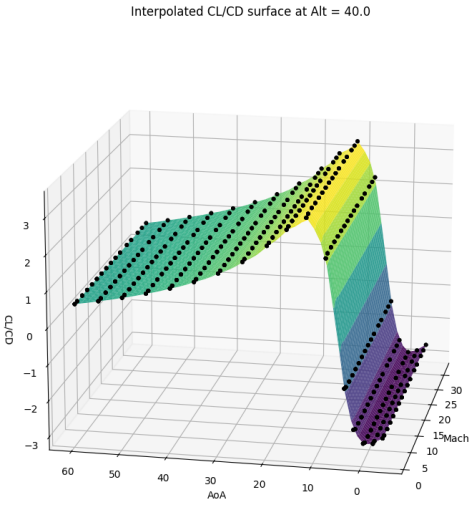
During trajectory simulation, aerodynamic coefficients must be evaluated thousands of times. Each evaluation must be executed quickly to avoid slowing down the overall simulation. At the same time, the interpolation must maintain sufficient accuracy.

6.6.1. Delaunay Triangulation

A scattered interpolant tool is used that constructs an interpolation surface over scattered data using Delaunay triangulation. Delaunay triangulation can be used to construct a surface of scattered data points [70]. The construction divides the data points into non-overlapping tetrahedra. The important property is that no data point lies inside the circumsphere of any tetrahedron [71]. This prevents poorly shaped tetrahedra, which leads to more stable and accurate interpolation within each tetrahedra. The interpolated data has been plotted to see the shape and check if the data have been interpolated in a correct way. In Figure 6.2a the C_L for varying values of Mach number and angle of attack are shown for an altitude of 40 kilometers. In Figure 6.2b the C_L/C_D ratio is shown for varying Mach numbers and angle of attack also at an altitude of 40 kilometers. The black dots are the known values, and the surface was created using the scattered interpolator.



(a) C_L interpolation for varying Mach number and angle of attack at an altitude of 40 km.



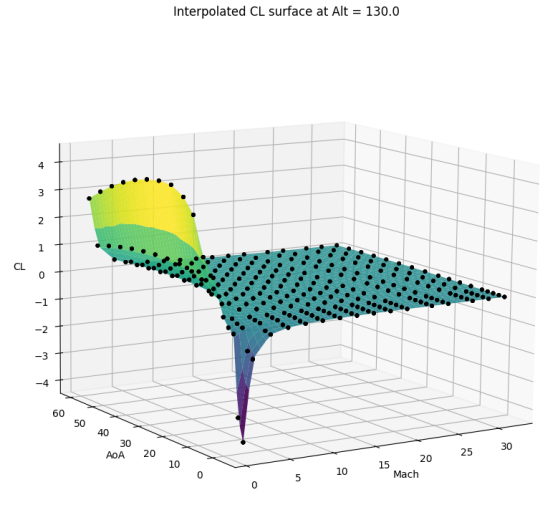
(b) C_L/C_D ratio interpolation for varying Mach number and angle of attack at an altitude of 40 km.

Figure 6.1: Three dimensional interpolation of aerodynamic data.

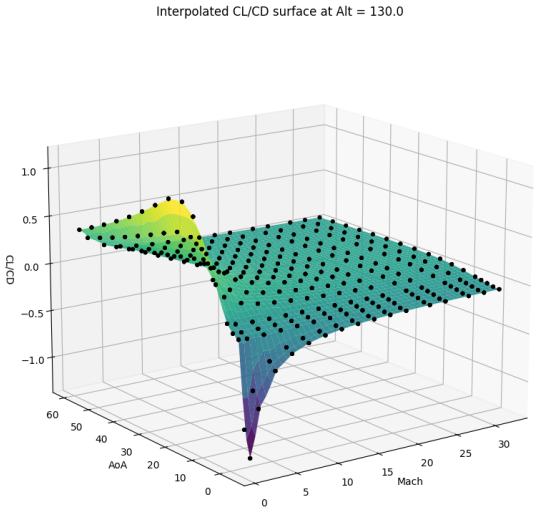
Aerodynamic Data Verification

Before using the aerodynamic coefficients in simulation or optimization, the full dataset was verified. Rather than directly applying the coefficients from the source files, a series of visualizations were generated to check consistency between Mach numbers, angle of attack, and altitude. These plots helped verify the overall behavior of the lift coefficient and the drag coefficient within the flight envelope.

During this verification process, a notable anomaly was observed. At high altitudes and low Mach numbers, the lift coefficient showed unexpectedly large values, which are physically unrealistic. This is shown in Figure 6.2.



(a) C_L interpolation for varying Mach number and angle of attack at an altitude of 130 km.



(b) C_L/C_D ratio interpolation for varying Mach number and angle of attack at an altitude of 130 km.

Figure 6.2: Anomaly with aerodynamic data at high altitudes (130 km) and low Mach numbers

Upon further investigation, this was traced to the use of modified Newtonian theory in the aerodynamic modeling tool FOSTRAD, which was used to generate the dataset. Modified Newtonian theory is only valid in the hypersonic regime and tends to break down at low Mach numbers, particularly in low-density flow conditions which are encountered at high altitudes. As a result, the extrapolated lift values in these regimes are not reliable. However, it must be noted that this is not a practical issue. At these altitudes, low Mach numbers are not reached and, therefore, will not cause a reliability issue for the use of the data.

Bilinear interpolation

In the initial model, where the altitude dependence was not yet included, bilinear interpolation was used. This method is computationally efficient for single-coefficient lookups because it avoids the overhead of triangulation. However, it is limited to datasets defined on a rectangular grid, an assumption that was valid in the early model. Despite working well in that simplified case and while it remains fast for individual lookups, it becomes less efficient when many lookups are required in simulation. Once altitude was introduced as an additional dimension, the data no longer conformed to a rectangular grid, and bilinear interpolation could no longer be applied. In the updated model, the Delaunay triangulation works better as they support scattered data in higher dimensions and are better suited to this application.

6.7. C++ Inheritance Class

The code includes three implementations of the HGV class, each with different characteristics. The base implementation uses SciPy's `LinearNDinterpolator`, which is a pure Python solution that constructs Delaunay triangulations for scattered data. This implementation provides good results with no external dependencies. The triangulation process combines the three independent variables into a single array and then constructs separate triangulations for each variable.

When aerodynamic data are needed for a particular flight condition, a query point array is constructed and passed to the interpolator. The interpolator identifies which tetrahedron in the triangulation contains the query point and returns the corresponding aero- and thermo-dynamic data.

The nearest data point is identified by computing Euclidean distances to all points in the database and selecting the minimum. While this approach increases the computational cost, this cost is acceptable since out-of-hull queries rarely occur during typical trajectories that remain well within the flight envelope.

Python can be extended with C++ libraries through Python bindings. Many Python libraries, such as parts of `SciPy`, are implemented in this way. A custom C++ library has been made, `interpolate.so`, that exposes the `ScatteredInterpolator3D` class to Python using `pybind11`. This implementation uses the `CGAL` library for Delaunay triangulation and is optimized for many individual scalar function calls. In contrast to this, `SciPy LinearNDInterpolator` is implemented in Cython using the `Qhull` library for triangulation which is designed for vectorized batch evaluations. For cases requiring many individual scalar interpolations, calling it repeatedly causes significant overhead from array construction and function call dispatch. Benchmarking shows that the `CGAL`-based implementation is more than 5 times faster for workloads involving many individual scalar interpolation calls. To integrate this performance improvement in the code, a Python class `HGV` was implemented that uses `SciPy LinearNDInterpolator` as the default interpolation. Afterwards, an inheritance was made, `HGVcgal`, that overrides several class functions to use the custom `CGAL` library instead. This design allows `HGVcgal` class to serve as a drop-in replacement for `HGV`, requiring minimal code changes, while it provides significantly improved performance for scalar interpolation workloads.

A critical implementation is made to enable reinforcement learning in a reasonable time. The triangulation structure is only constructed **once** during the class initialization and reused for all subsequent queries. On average, a look up in the triangulation is done 40 times per time step. This equates to tens of million calls to triangulation during the training. Constructing the triangulation once and reusing it for all queries reduces the training time significantly. In parallel training each worker constructs the triangulation once during startup, then processes millions of trajectories using that same structure.

The interpolation performed by C++ on a full trajectory of the HGV has been compared to the interpolation that has been done by Python's `LinearNDinterpolator`. The resulting accumulated error between

the two interpolators of one full trajectory on a total distance of approximately 10000 km was only 200 m. This is a difference of only $1.99 \times 10^{-3}\%$, which can be considered negligible.

6.8. Atmospheric Modeling

The atmospheric modeling implementation gives the US Standard Atmosphere (USSA) 1976 model, which is the international reference for aerospace applications. It covers altitudes from sea level to one thousand kilometers. This implementation supports both geometric altitude, measured as the actual height above mean sea level and the geopotential altitude, which also accounts for the variation of gravity with altitude and is used in some atmospheric tables.

For altitudes from the surface up to 51 kilometers, the model adopts the same values as those in the earlier 1962 edition and relies on traditional definitions. However, the definitions below 20 kilometers do not represent a true statistical mean of all the available measurements, but rather a smoothed and idealized approximation [72]. Between 51 kilometers and approximately 85 kilometers, the model is based on averaged observations of present-day atmospheric conditions, assuming a well-mixed atmosphere.

Above this range, the model is more sophisticated in order to incorporate the effects of dissociation, diffusion, and composition variation with altitude. In the 86-100 kilometer region, the temperature profile is no longer constructed as piecewise linear segments. Instead, it is defined through a series of height-dependent functions, ensuring continuity in both the temperature and the gradient. The expressions are not arbitrary, but are derived from observational data [72].

Altitudes above approximately 110 kilometers show a lot of variation, so data from this region were especially important to define the temperature profile at high altitude. Observations at 110, 120 and 150 km were used to decide where the model should include constant temperature layers and where the temperature should change with height [72].

The main function accepts altitude in meters and returns ten values: pressure, density, temperature, speed of sound, dynamic viscosity, gravitational acceleration, and four thermodynamic constants.

6.8.1. Comparison to Exponential Atmosphere

The exponential atmosphere is a simplified model in which the density decreases as

$$\rho(h) = \rho_0 \, e^{-h/H_s} \tag{6.1}$$

where ρ_0 is the sea level density and H_s the constant scale height. This formulation assumes a constant temperature and a uniform atmospheric composition, which makes this model only valid in the lower atmosphere.

In Figure 6.3, the difference in density and temperature between the exponential atmosphere and the USSA 1976 model is shown.

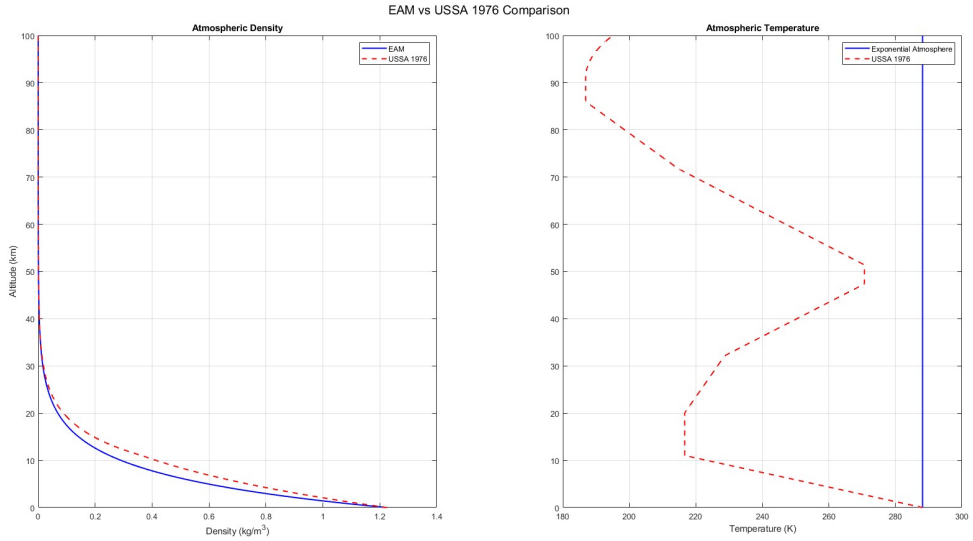


Figure 6.3: Comparison between the exponential atmosphere and the USSA1976 atmosphere, with density difference on the left and temperature difference on the right.

6.9. Physical Constants Module

The constants module defines the physical parameters that are used throughout the simulation framework. These are centralized in a single file, which makes sure there is consistency within the whole program. The constants are grouped into three main categories. Firstly, the Earth model constants. These include the Earth's mean radius, representing a spherical approximation to Earth's actual oblate spheroid shape. This approximation introduces maximum errors of approximately 20 kilometers in radius but simplifies great circle calculations significantly while maintaining sufficient accuracy for trajectory applications spanning thousands of kilometers. Earth's rotation rate corresponds to one complete rotation per sidereal day and is used for the Coriolis and centrifugal acceleration terms in the equations of motion.

The gravitational constants include standard gravity at sea level, which is used in the code as a normalization factor for load factor calculations. The circular orbital velocity at sea level is used for a characteristic velocity scale for hypersonic flight. This velocity represents the minimum speed that is needed for orbital flight and is a unit to normalize the HGV velocity.

Atmospheric constants include the standard sea level density, the ratio of specific heats for air and the specific gas constant for air. These values come from the USSA 1976 model and make sure that there is consistency between the atmospheric property calculations and the constant definitions used in other models. The specific gas constant is equal to the universal gas constant divided by the molecular weight of the air. This assumes a well-mixed air composition.

6.10. Spherical Geometry Calculations

In the `spherical.py` module the functions are provided for navigation calculations on Earth's curved surface. In order to understand these calculations, it must be recognized that Earth is approximately spherical and that the shortest path between two points on a sphere is not a straight line through space, but actually an arc along the surface which is called a great circle. These arcs are the intersection of the sphere with a plane passing through Earth's center and the two points of interest.

All functions in this module use great circle mathematics and have a consistent convention. The latitude and longitude values are in radians, the bearings are measured clockwise from the north with values from 0 to 2π radians, and the cross-track distances use negative values for positions to the right of a path and positive values to the left.

An area on a sphere that is bounded by arcs of great circles is referred to as a spherical polygon or spherical triangle, which is shown in Figure 6.4 [38].

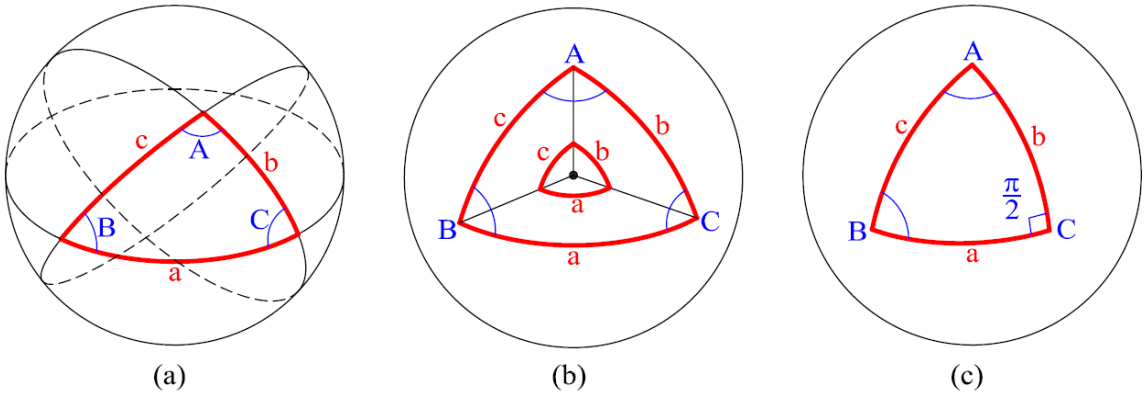


Figure 6.4: Spherical trigonometry [38]

Spherical triangles satisfy several important trigonometric relationships. The spherical law of cosines relates the three sides of a spherical triangle (a, b, c) to one if its angles (C):

$$\cos c = \cos a \cos b + \sin a \sin b \cos C \tag{6.2}$$

The second spherical law of cosines has the opposite form, expressing an angle (C) in terms of the three sides:

$$\cos C = -\cos A \cos B + \sin A \sin B \cos c \tag{6.3}$$

Furthermore, spherical triangles have their own version of the law of sines. Finally, spherical triangles satisfy the half-side formula. These relations can be found in the book: *Re-entry Systems* by Mooij et al. [38] and the following relationships can be derived from these relations.

6.10.1. Distance Calculation

The distance function uses the haversine formula, which computes great-circle distances while maintaining precision even when points are close together. The calculation begins by computing the difference in coordinates between two points. For latitudes ϕ_1 and ϕ_2 the difference is simple: $\Delta\phi = \phi_2 - \phi_1$. Longitude differences are handled differently, since the longitude wraps around at the international date line. The difference $\Delta\lambda = \lambda_2 - \lambda_1$ is adjusted by adding π, taking the result modulo 2π and then subtracting π again. This wrapping makes the difference fall in the range $[-\pi, \pi]$, which gives the shorter of the two possible paths around the globe.

The haversine formula first computes an intermediate formula [38]:

$$a = \sin^2\left(\tfrac{\Delta\phi}{2}\right) + \cos\phi_1 \cos\phi_2 \, \sin^2\left(\tfrac{\Delta\lambda}{2}\right) \tag{6.4}$$

From this equation, the central angle separating the points is extracted:

$$c = 2 \arctan\left(\frac{\sqrt{a}}{\sqrt{1-a}}\right) \tag{6.5}$$

Finally, the great circle distance is simply the arc length along the Earth's surface:

$$d = R_E \cdot c \tag{6.6}$$

where R_E is Earth's mean radius.

6.10.2. Bearing Calculation

The bearing function computes the initial azimuth angle for traveling from one point to another along the great circle that connects the two points. This is not the angle that would point directly at the destination in a straight line through space, but the direction to start traveling along Earth's surface to follow the shortest path.

Spherical trigonometry gives the following formula:

$$\beta = \arctan\left(\frac{\sin(\lambda_2 - \lambda_1)\cos\phi_2}{\cos\phi_1 \sin\phi_2 - \sin\phi_1 \cos\phi_2 \cos(\lambda_2 - \lambda_1)}\right) \tag{6.7}$$

where β is the bearing direction. To give an intuitive understanding, if you travel from the equator at zero degrees longitude toward the north pole, the bearing would be exactly zero. The formula generalizes this to arbitrary starting and ending positions on the sphere.

6.10.3. Destination Function

The destination function performs the inverse operation. With a given starting position, a bearing direction, and a distance, the end point is computed. The distance is first converted to an angular distance:

$$\delta = \frac{d}{R_E} \tag{6.8}$$

This angular distance is the number of radians of arc along the surface that the path covers. The destination latitude is computed using the spherical law of cosines:

$$\phi_2 = \arcsin(\sin\phi_1 \cos\delta + \cos\phi_1 \sin\delta \cos\beta) \tag{6.9}$$

The longitude change is computed as:

$$\Delta\lambda = \arctan\left(\frac{\sin\beta \sin\delta \cos\phi_1}{\cos\delta - \sin\phi_1 \sin\phi_2}\right) \tag{6.10}$$

The destination longitude is then $\lambda_2 = \lambda_1 + \Delta\lambda$, also wrapped to the standard range $[-\pi, \pi]$.

6.10.4. Cross-Track Distance

The cross-track distance function measures how far a point lies from a great circle path defined by two other points. This is useful to determine how far the HGV can reach in the lateral range. The calculation first determines the angular distance from the path's starting point, point 1, to the measurement point, point 3:

$$d_{13} = \frac{d(1,3)}{R_E} \tag{6.11}$$

where $d(1,3)$ is computed using the haversine formula as was previously described. Then, two bearings are computed. The bearing point from point 1 to point 3, β_{13}, and the bearing from point 1 to point 2, β_{12}. The cross-track distance formula from spherical trigonometry is:

$$dx_t = R_E \cdot \arcsin(\sin(d_{13}) \sin(\beta_{13} - \beta_{12})) \tag{6.12}$$

The sign of the sine gives the left-right distinction. This is negative for right of track and positive for left of track.

6.10.5. Midpoint Calculation

The midpoint function determines the point halfway along a great circle path between two points. This implementation uses the existing destination function. It computes the total distance between the points, computes the bearing from the first to the second point, and then projects halfway along that bearing:

$$(\phi_m, \lambda_m) = \text{destination}\left(\phi_1, \lambda_1, \beta_{12}, \frac{1}{2}d_{12}\right) \tag{6.13}$$

6.11. Equations of Motion and Integration

The HGV is modeled as a 3-DOF point mass on a rotating spherical Earth. The state is:

$$x = [h, \lambda, \phi, V, \gamma, \psi]$$

with altitude h, longitude λ, latitude ϕ, speed V, γ flight path angle relative to the local horizontal, and ψ heading relative to the north.

The control inputs are the angle of attack α, which determines the aerodynamic coefficients $C_L(\alpha, M, h)$ and $C_D(\alpha, M, h)$, and the bank angle σ which orients the lift vector about the velocity axis. These two controls allow management of both energy and trajectory shape during the reentry.

An activity diagram for the equations of motion function is given in Appendix A.

6.11.1. Forces and Gravity

The aerodynamic forces are defined as:

$$L = \tfrac{1}{2}\,\rho(h)\,V^2\,S_{\text{ref}}\,C_L(\alpha, M, h), \qquad D = \tfrac{1}{2}\,\rho(h)\,V^2\,S_{\text{ref}}\,C_D(\alpha, M, h), \tag{6.14}$$

where L and D are the lift and drag forces respectively, $\rho(h)$ is the atmospheric density from the USSA-1976 model and S_{ref} is the surface area of the HGV. M is the Mach number with local speed of sound $a(h)$:

$$M = \frac{V}{a(h)} \tag{6.15}$$

In this work, gravity is modeled using the spherical Earth approximation, which assumes that the Earth is a perfect sphere of radius $R_E = 6371$km, which is defined by the IUGG standard. Under this approximation, the gravitational acceleration varies only with altitude and is as follows:

$$g(h) = \frac{\mu}{(R_E + h)^2} \tag{6.16}$$

where $\mu = 3.986004415 \times 10^{14}$ m^3/s^2 and h is the altitude above the Earth's surface.

6.11.2. Equations of Motion on a Rotating Earth

The motion of the HGV for a rotating spherical is described by the following differential equations [22]:

$$\dot{h} = V \sin\gamma \tag{6.17}$$

$$\dot{\lambda} = \frac{V \cos\gamma \sin\psi}{r \cos\phi} \tag{6.18}$$

$$\dot{\phi} = \frac{V \cos\gamma \cos\psi}{r} \tag{6.19}$$

$$\dot{V} = -\frac{D}{m} - g\sin\gamma + \omega_{\oplus}^2 r\cos\phi\left(\sin\gamma\cos\phi - \cos\gamma\sin\phi\cos\psi\right) \tag{6.20}$$

$$\begin{aligned}\dot{\gamma} = \frac{L\cos\sigma}{mV} + \left(\frac{V}{r} - \frac{g}{V}\right)\cos\gamma + 2\omega_{\oplus}\cos\phi\sin\psi \\ + \frac{\omega_{\oplus}^2 r}{V}\cos\phi\left(\cos\gamma\cos\phi + \sin\gamma\sin\phi\cos\psi\right)\end{aligned} \tag{6.21}$$

$$\begin{aligned}\dot{\psi} = \frac{L\sin\sigma}{mV\cos\gamma} + \frac{V}{r}\cos\gamma\sin\psi\tan\phi + 2\omega_{\oplus}\left(\sin\phi - \cos\phi\tan\gamma\cos\psi\right) \\ + \frac{\omega_{\oplus}^2 r}{V\cos\gamma}\sin\phi\cos\phi\sin\psi\end{aligned} \tag{6.22}$$

where r is the radial distance from Earth's center and $\omega_{\oplus}$ is Earth's rotation rate with a value of 7.2921159×10^{-5} rad/s. These six derivatives are assembled into an array and returned to the integrator.

6.11.3. The Step Method

The step method advances the simulation by a specified time duration; in this case a time duration of 5 seconds was chosen. For the activity and sequence diagram of this function can be referred to Appendix A. During this interval, the control inputs bank angle and angle of attack remain constant at their most recently set values. The method integrates the equations of motion from the current flight time forward using SciPy's `solve_ivp` function with an adaptive 4th/5th-order Runge-Kutta method. This method is the interface between the control system and the physics simulation.

The integration process repeatedly evaluates the Equations of Motion function to compute the state derivatives at intermediate points, using these to propagate the state vector forward. When the method is called, it is first checked whether the simulation has already terminated, returning immediately if the done flag is set. Otherwise, it proceeds with the integration. The adaptive algorithm automatically adjusts the internal step sizes based on error estimates. This means it takes smaller steps during rapid transients and larger steps during flight phases that vary slowly. This provides computational efficiency across the different flight conditions that are encountered.

Event detection monitors the altitude to enable precise termination of the program. The integrate function evaluates the stop-criterion function at each internal step. Once the threshold is crossed, the integrator halts the integration when the threshold is crossed.

This architecture is built in such a way that it aligns with reinforcement learning. The agent observes the state, selects control inputs, and the step method advances the simulation by one time increment to produce the next state. After each time step has passed, the output is transferred back to the agent, which will provide a new bank angle and angle of attack based on this result. Each call represents one decision point, with controls kept constant during integration.

6.12. Verification with Equivalent Cartesian Formulation

An equivalent Cartesian formulation in Earth-Centered-Earth-Fixed (ECEF) coordinates was implemented for verification. ECEF is a Cartesian coordinate system with its origin at Earth's center that rotates with the planet. The x-axis point to the intersection of the equator and prime meridian ($0°$ latitude, $0°$ longitude), the y-axis points to $0°$ latitude and $90°$ longitude, and the z-axis points to the North Pole. This is shown in Figure 6.5.

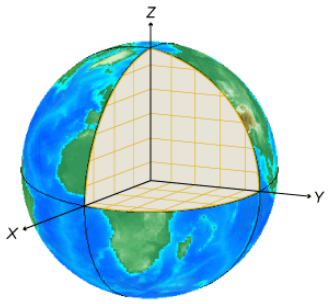


Figure 6.5: ECEF Cartesian coordinate system [73]

6.12.1. Converting Position (Spherical to ECEF)

The vehicle position is defined by latitude ϕ and longitude λ, and altitude h. For this validation study, Earth is modeled as a perfect sphere of radius R_E for consistency with the spherical equations of motion. Note that while ECEF can accommodate ellipsoidal geometry, a spherical Earth model was used here to allow for direct comparison between formulations. The radial distance from Earth's center is $r = R_E + h$. The Cartesian ECEF position is then:

$$\begin{aligned} x &= r\cos(\phi)\cos(\lambda) \\ y &= r\cos(\phi)\sin(\lambda) \\ z &= r\sin(\phi) \end{aligned} \tag{6.23}$$

6.12.2. Converting Velocity (NED to ECEF)

The velocity transformation requires defining the local North-East-Down (NED) frame at the vehicle's position. The NED frame is a local tangent plane where North points along the local meridian, East points along the local parallel, and Down points toward Earth's center. This is shown in Figure 6.6

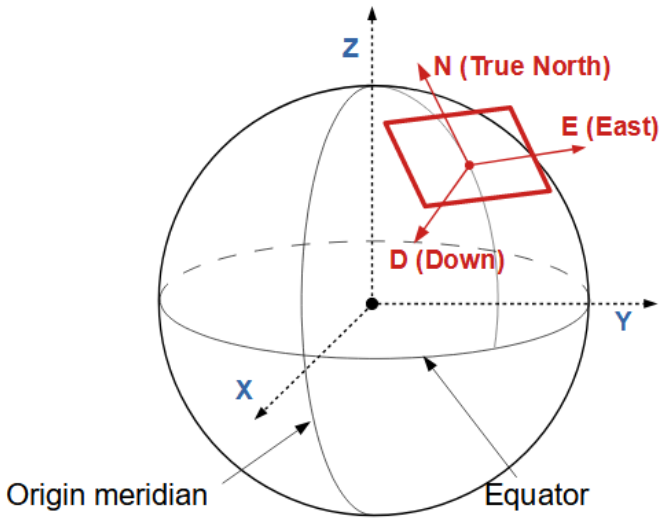


Figure 6.6: Definition of the North-East-Down reference frame [74]

The velocity vector in NED coordinates is as follows:

$$\begin{aligned} V_N &= V\cos(\gamma)\cos(\psi) \\ V_E &= V\cos(\gamma)\sin(\psi) \\ V_D &= -V\sin(\gamma) \end{aligned} \tag{6.24}$$

where V_N is velocity in North component, V_E velocity in East component, V_D velocity in down component, γ flight path angle, and ψ heading.

6.12.3. Constructing the Direction Cosine Matrix

The transformation from NED to ECEF uses a Direction Cosine Matrix (DCM) that depends only on the vehicle latitude ϕ and longitude λ. The DCM is constructed geometrically by expressing each NED axis as a unit vector in ECEF coordinates. Starting from the vehicle's position in ECEF:

$$\mathbf{r} = \begin{bmatrix} r\cos(\phi)\cos(\lambda) \\ r\cos(\phi)\sin(\lambda) \\ r\sin(\phi) \end{bmatrix} \tag{6.25}$$

the three NED axes in ECEF coordinates are

North direction: Found by taking the derivative of position with respect to latitude (moving along the meridian):

$$\frac{\partial \mathbf{r}}{\partial \phi} = \begin{bmatrix} -\sin(\phi)\cos(\lambda) \\ -\sin(\phi)\sin(\lambda) \\ \cos(\phi) \end{bmatrix} \tag{6.26}$$

East direction: Found by taking the derivative of position with respect to longitude (moving along the parallel):

$$\frac{\partial \mathbf{r}}{\partial \lambda} = \begin{bmatrix} -\sin(\lambda) \\ \cos(\lambda) \\ 0 \end{bmatrix} \tag{6.27}$$

Down direction: Points toward Earth's center (opposite of the radial direction):

$$-\frac{\mathbf{r}}{\|\mathbf{r}\|} = \begin{bmatrix} -\cos(\phi)\cos(\lambda) \\ -\cos(\phi)\sin(\lambda) \\ -\sin(\phi) \end{bmatrix} \tag{6.28}$$

These three unit vectors form the columns of the transformation matrix from ECEF to NED, denoted $\mathbf{C}_{V,R}$, where the subscript notation indicates transformation from the ECEF reference frame (R) to the vehicle's local NED frame (V):

$$\mathbf{C}_{V,R} = \begin{bmatrix} -\sin(\phi)\cos(\lambda) & -\sin(\lambda) & -\cos(\phi)\cos(\lambda) \\ -\sin(\phi)\sin(\lambda) & \cos(\lambda) & -\cos(\phi)\sin(\lambda) \\ \cos(\phi) & 0 & -\sin(\phi) \end{bmatrix} \tag{6.29}$$

For the transformation from NED to ECEF, the transpose is used:

$$\mathbf{C}_{R,V} = \mathbf{C}_{V,R}^T \tag{6.30}$$

The ECEF velocity is then:

$$\mathbf{V}_{\text{ECEF}} = \mathbf{C}_{R,V} \cdot \mathbf{V}_{\text{NED}} = \mathbf{C}_{V,R}^T \cdot \mathbf{V}_{\text{NED}} \tag{6.31}$$

Equations of Motion in ECEF coordinates
The state vector in ECEF is defined as follows:

$$\mathbf{y}(t) = \begin{bmatrix} \mathbf{v}(t) \\ \mathbf{r}(t) \end{bmatrix} = \begin{bmatrix} v_x(t) \\ v_y(t) \\ v_z(t) \\ x(t) \\ y(t) \\ z(t) \end{bmatrix} \tag{6.32}$$

where $\mathbf{r} = [x,\, y,\, z]^\top$ is the position vector and $\mathbf{v} = [v_x,\, v_y,\, v_z]^\top$ is the velocity vector. The state derivative is:

$$\frac{d\mathbf{y}}{dt} = \frac{d}{dt} \begin{bmatrix} \mathbf{v}(t) \\ \mathbf{r}(t) \end{bmatrix} = \begin{bmatrix} a_x \\ a_y \\ a_z \\ v_x \\ v_y \\ v_z \end{bmatrix} \tag{6.33}$$

where $\mathbf{a} = [a_x,\, a_y,\, a_z]^\top$ is the acceleration vector in ECEF coordinates, consisting of gravitational, aerodynamic, centrifugal, and Coriolis components.

Gravitational Acceleration
The gravitational acceleration is defined as follows:

$$\mathbf{a}_{\text{grav}} = -g(r) \cdot \frac{\mathbf{r}}{\|\mathbf{r}\|} \tag{6.34}$$

where $g(r)$ is the gravitational acceleration defined in Equation 6.16 and $\mathbf{r}/\|\mathbf{r}\|$ is the unit vector pointing from the Earth's center to the vehicle.

Aerodynamic Acceleration
Aerodynamic forces, consisting of lift and drag, are computed in the aerodynamic frame and transformed directly to ECEF using a geometrically constructed DCM. The aerodynamic frame is defined by three unit vectors in ECEF:

$\mathbf{X}_a$**:** Unit vector along the velocity:

$$\mathbf{X}_a = \frac{\mathbf{v}}{\|\mathbf{v}\|} = \begin{bmatrix} v_x/V \\ v_y/V \\ v_z/V \end{bmatrix} \tag{6.35}$$

$\mathbf{Z}_e$**:** Unit vector toward Earth's center, used as a reference for constructing the frame:

$$\mathbf{Z}_e = -\frac{\mathbf{r}}{\|\mathbf{r}\|} = \begin{bmatrix} -r_x/r \\ -r_y/r \\ -r_z/r \end{bmatrix} \tag{6.36}$$

$\mathbf{Y}_a$**:** Perpendicular to both velocity and vertical:

$$\mathbf{Y}_a = \frac{\mathbf{Z}_e \times \mathbf{X}_a}{\|\mathbf{Z}_e \times \mathbf{X}_a\|} \tag{6.37}$$

$\mathbf{Z}_a$**:** Completes the right-handed frame:

$$\mathbf{Z}_a = \mathbf{X}_a \times \mathbf{Y}_a \tag{6.38}$$

The three axes of the aerodynamic frame ($\mathbf{X}_a$, $\mathbf{Y}_a$, $\mathbf{Z}_a$) form the columns of the DCM from the aerodynamic frame to ECEF, where the subscript notation indicates transformation from the aerodynamic frame to the ECEF frame:

$$\mathbf{C}_{\text{ECEF,Aero}} = \begin{bmatrix} \mathbf{X}_a & \mathbf{Y}_a & \mathbf{Z}_a \end{bmatrix} \tag{6.39}$$

The aerodynamic forces in the aerodynamic frame are:

$$\mathbf{F}_{\text{aero}} = \begin{bmatrix} -D \\ L\sin(\sigma) \\ -L\cos(\sigma) \end{bmatrix} \tag{6.40}$$

where D is the drag force and L the lift force. The transformation to ECEF is then:

$$\mathbf{F}_{\text{ECEF}} = \mathbf{C}_{\text{ECEF,Aero}} \cdot \mathbf{F}_{\text{aero}} \tag{6.41}$$

Centrifugal Acceleration
The centrifugal acceleration acts outward from the Earth's axis of rotation. It is zero in the z-direction and increases with latitude. It is defined as:

$$\mathbf{a}_{\text{centrifugal}} = -\boldsymbol{\omega}_E \times (\boldsymbol{\omega}_E \times \mathbf{r}) \tag{6.42}$$

where $\boldsymbol{\omega}_E = [0,\, 0,\, \omega_\oplus]^\top$ and $\omega_\oplus$ the Earth's rotational rate, this simplifies to:

$$\mathbf{a}_{\text{centrifugal}} = -\boldsymbol{\omega}_E \times (\boldsymbol{\omega}_E \times \mathbf{r}) = \begin{bmatrix} \omega_\oplus^2 x \\ \omega_\oplus^2 y \\ 0 \end{bmatrix}$$

Coriolis Acceleration
The Coriolis acceleration accounts for motion in the rotating reference frame:

$$\mathbf{a}_{\text{Coriolis}} = -2\boldsymbol{\omega}_E \times \mathbf{v} \tag{6.43}$$

For this case, the cross product simplifies to:

$$\mathbf{a}_{\text{Coriolis}} = -2 \begin{bmatrix} 0 \\ 0 \\ \omega_\oplus \end{bmatrix} \times \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix} = \begin{bmatrix} 2\omega_\oplus v_y \\ -2\omega_\oplus v_x \\ 0 \end{bmatrix} \tag{6.44}$$

This means that the Coriolis acceleration always acts perpendicular to the direction of motion. In the northern hemisphere it will deflect motion to the right, and in the southern hemisphere it will deflect motion to the left. The Coriolis term becomes relevant at high speeds and over long distances, which is the case for HGVs.

The Equations of Motion
The full equations of motion in ECEF coordinates is as follows:

$$\dot{\mathbf{v}} = \mathbf{a}_{\text{grav}} + \mathbf{a}_{\text{aero}} + \mathbf{a}_{\text{centrifugal}} + \mathbf{a}_{\text{Coriolis}}, \qquad \dot{\mathbf{r}} = \mathbf{v} \tag{6.45}$$

and more specific:

$$\dot{\mathbf{v}} = -g(r)\frac{\mathbf{r}}{\|\mathbf{r}\|} + \frac{1}{m}\mathbf{F}_{\text{aero}} + \begin{bmatrix} \omega_\oplus^2 x \\ \omega_\oplus^2 y \\ 0 \end{bmatrix} + \begin{bmatrix} 2\omega_\oplus v_y \\ -2\omega_\oplus v_x \\ 0 \end{bmatrix} \tag{6.46}$$

Comparison ECEF to Spherical Coordinates

To verify correctness of the coordinate transformations, trajectories were propagated using both the spherical coordinate formulation and the ECEF Cartesian formulation. Both implementations used identical initial conditions, force models, and time spans. After integration, the ECEF position was converted back to latitude, longitude, and altitude using the inverse transformations. These were compared directly with the output from the spherical integration.

The results show good agreement. 1000 trajectories have been simulated with random initial conditions. The relative difference in final position between the two approaches was only 5.4×10^{-3} km on trajectories spanning thousands of kilometers, which can be attributed to accumulated rounding errors from coordinate transformations and numerical integration tolerances.

6.13. Constraint Monitoring

During hypersonic flight, the vehicle experiences extreme aerodynamic forces and heating that can lead to structural failure or thermal damage if not constrained. The feasible flight envelope for an HGV is bounded by three operational constraints. These constraints include the load factor, the dynamic pressure, and the thermal constraint. All three constraints are necessary because they can be violated independently. The load factor limits structural stress from maneuvering, for example aggressive banking turns. The dynamic pressure limits aerodynamic loads from high-speed flight in dense atmosphere. The thermal constraint limits material temperatures.

The load factor and dynamic pressure are reformulated to determine the critical velocity at which the constraint is violated as a function of altitude. This allows for real-time monitoring of constraint violations during trajectory propagation. The temperature has a fixed maximum stagnation point temperature.

6.13.1. Load Factor Constraint

The load factor is defined as the ratio of aerodynamic force to weight [75]:

$$n = \frac{\sqrt{L^2 + D^2}}{mg} \leq n_{\max} \tag{6.47}$$

Setting this equal to the maximum allowable load factor n_{max} and using the aerodynamic force equations and then solving for velocity, gives:

$$V_{gc} = \sqrt{\frac{2 \cdot m \cdot n_{\max} \cdot g}{S_{\text{ref}} \cdot \rho \cdot \sqrt{C_D^2 + C_L^2}}} \tag{6.48}$$

Excessive loads can cause structural failure due to overstressing of the airframe. This is a structural limit in the HGV design. Due to the lack of publicly available or declassified data for the structural load limits of operational HGV systems, a reasonable estimate is necessary. Based on a comparative analysis with other high-speed vehicles, a maximum allowable load factor of 25 g is chosen. This can easily be changed in the code, where in the `.json` file for the specific HGV the maximum load factor can be set. This is the same for the other constants.

6.13.2. Dynamic Pressure Constraint

The dynamic pressure condition is stated as follows [75]:

$$q = \frac{1}{2}\rho V^2 \leq q_{\max} \tag{6.49}$$

Setting this equal to the maximum allowable dynamic pressure q_{max} and solving for the velocity gives:

$$V_{qc} = \sqrt{\frac{2q_{max}}{\rho}} \tag{6.50}$$

This formulation shows that the allowable velocity decreases with increasing atmospheric density. At low altitudes, where air is more dense, lower velocities generate the same dynamic pressure as higher velocities at altitude.

Dynamic pressure is a critical constraint. Control surfaces experience aerodynamic hinge moments that scale with dynamic pressure. At excessive dynamic pressure levels, these hinge moments can exceed the authority of control actuators which can lead to loss of controllability, or cause structural failure of the control surfaces themselves

In the optimization, an upper bound of 500 kPa is enforced, as has been done for the Common Aero Vehicle according to Yao et al [76]. Limiting dynamic pressure effectively prevents the optimizer from selecting aggressive low-altitude and high-speed trajectories that would violate physical feasibility.

This high dynamic pressure is also checked with another high-speed re-entry vehicle, the RAM C-III. This NASA experiment was created to study the problem of radio frequency blackouts at high entry velocities [77]. The dynamic pressure versus time is shown in Figure 6.7, and it can be seen that this vehicle reaches dynamic pressures of over 600 kPa.

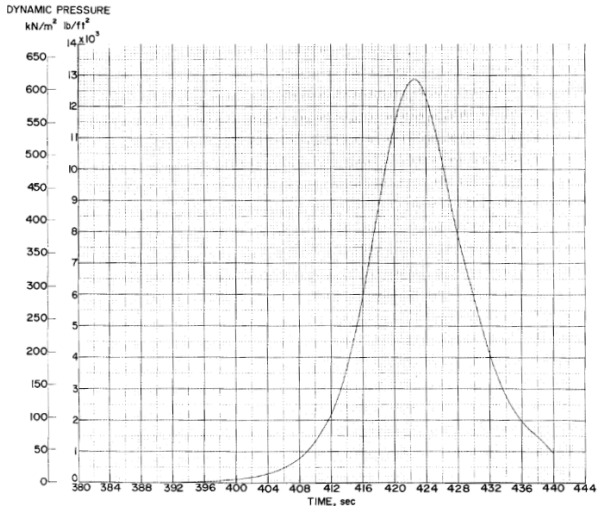


Figure 6.7: Dynamic pressure versus time of the RAM C-III re-entry vehicle [77].

6.13.3. Temperature Constraint

The temperature constraint ensures that the HGV thermal protection system can withstand the aerodynamic heating experienced during hypersonic reentry. An aerodynamic simulation has been performed using the actual geometry of the hypersonic glide vehicle. For each combination of altitude, Mach number and angle of attack the aerodynamic heating at the stagnation point was calculated. This has been interpolated for every altitude, shown in Figure 6.8 for an altitude of 40 km.

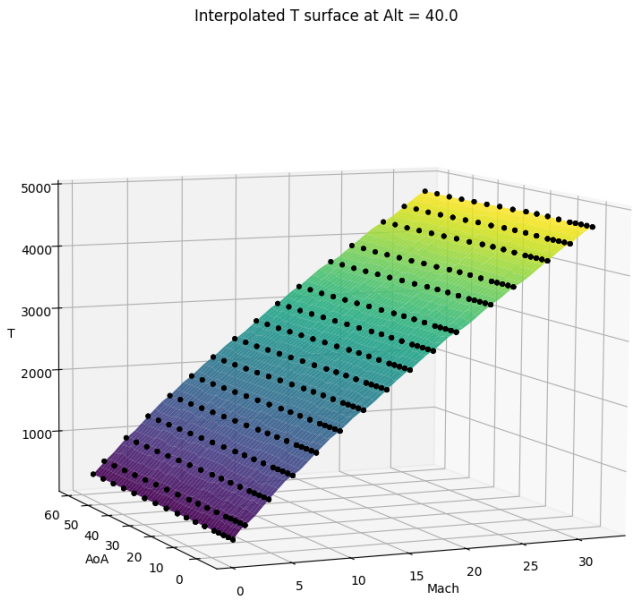


Figure 6.8: Temperature for varying angle of attack and Mach numbers at an altitude of 40 kilometers.

The constraint is then defined as:

$$T_{stag} \leq T_{max} \tag{6.51}$$

where T_{max} is the maximum defined temperature and T_{stag} the stagnation point temperature of the vehicle.

During the analysis of the temperature profile, it was noted that variations in angle of attack had a surprisingly limited effect on the observed temperature of the stagnation point. This is because the stagnation point remains in approximately the same location for changes in angle of attack. Since the peak temperature experienced by the HGV is at the stagnation point and this location does not change significantly, the stagnation temperature remains relatively constant.

For the thermal protection system of the HGV, it can be divided into two types: ablative and non-ablative materials. Ablative materials protect the vehicle by sacrificing material. As the material chars and erodes, the heat is carried away with the ablating mass, which limits the heat that is transferred to the underlying structure. This type of TPS allows for handling high heat fluxes and surface temperatures. However, since the material is consumed during flight, ablative materials cannot be used on aerodynamic surfaces like the wings and control surfaces. This is due to the effect that the ablative TPS will alter the shape of the wings and control surfaces which directly impact the vehicle's aerodynamic characteristics. Therefore, non-ablative materials are typically used on wings and control surfaces to ensure that the aerodynamic shape remains unchanged.

Nonablative materials are based on high temperature resistant materials to withstand the heat experienced. These materials will absorb heat into their mass and radiate it back to the environment while the structure remains intact.

The Stardust was a NASA space mission with the goal of collecting samples from a comet and returning back to Earth [38]. This capsule had an ablative heat shield made of PICA. PICA stands for Phenolic Impregnated Carbon Ablators and is typically used for planetary entry probes [78]. Olynick et al. [79] have done an analysis on the heat shield of Stardust that is made of PICA. A maximum wall temperature of 3735 K is experienced when effects were not modeled. A temperature of 3380 is experienced which

included an ablative analysis, showing that ablative cooling reduced the maximum wall temperature to 3380K. This is shown in Figure 6.9.

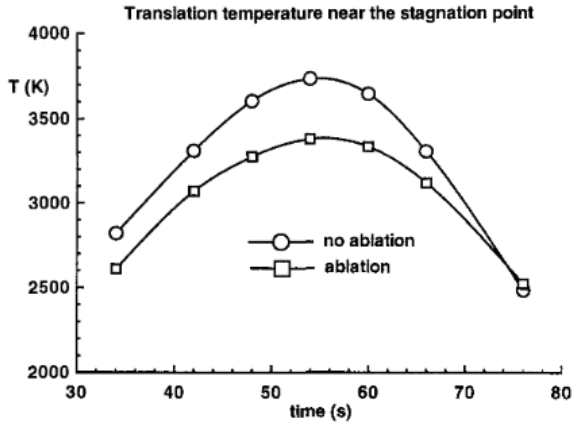


Figure 6.9: Comparison of the nonablating and ablating surface temperatures at the stagnation point [79]

Another possibility is a class of materials called Ultra-High Temperature Ceramic (UHTC) composites, which have been developed to withstand temperatures as high as $3500K$ [80]. Ceramic compounds based on metal borides such as zirconium diboride (ZrB2) and hafnium diboride (HfB2), also UHTCs, have extremely high melting temperatures, which are respectively $3300K$ and $3500K$ [81]. This is proven to be suitable for sharp-shaped leading edges for hypersonic vehicles. For the model used, the maximum temperature that cannot be exceeded has been set to $3500K$, but again this can easily be changed to whatever value is appropriate for the vehicle you want to model.

For more information on possible thermal protection systems of HGVs, refer to Appendix B.

6.13.4. Built-in Constraint Monitoring

The HGV object continuously tracks the three primary flight constraints. Each constraint is checked in real-time at every simulation step. Violations are plotted post-run in the simulation model. This gives immediate feedback on the feasibility of a simulated trajectory with a certain bank angle and angle of attack control. This is shown in Figure 6.10.

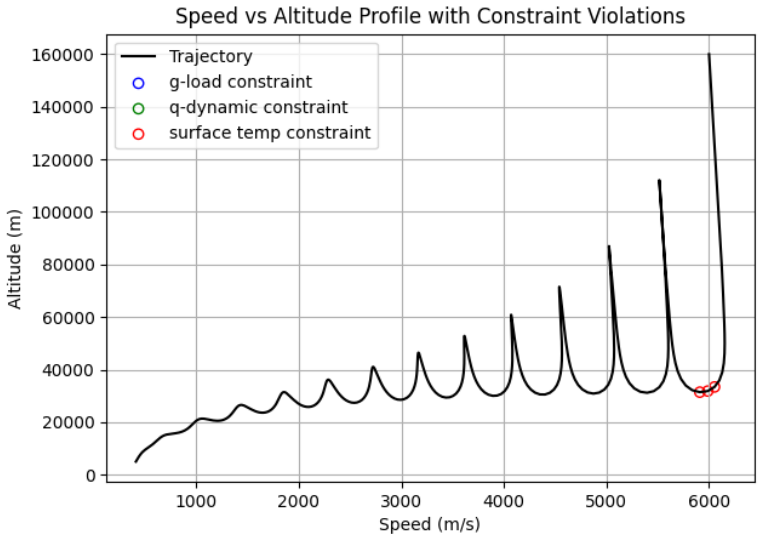


Figure 6.10: Speed versus Altitude Profile with Constraint Violations

The constraint checking is encapsulated within a single method inside the HGV class. This means that adding new constraints, such as for the scramjet integrated flight mode, requires minimal effort and no architectural changes.

6.14. Equilibrium Glide Possibility

A concept in hypersonic glide vehicle dynamics is the equilibrium glide condition. This condition is a theoretical boundary where the lift force exactly balances the net gravitational forces accounting for centrifugal effects due to the vehicle's velocity along the curved Earth:

$$L = mg\left(1 - \frac{V^2}{V_c^2}\right) \tag{6.52}$$

This equilibrium glide condition defines one specific relationship that results in non-skipping flight. However, it represents only a single constraint line within the broader entry corridor defined by the already existing temperature and dynamic pressure limits. There exist many possible combinations of bank angle and angle of attack that can produce non-skipping trajectories while remaining within the entry corridor constraints.

In the reinforcement learning framework, the non-skipping flight condition is an additional constraint boundary. Instead of forcing the vehicle to follow the equilibrium glide line exactly, the RL agent learns to find any combination of bank angle and angle of attack that satisfies the entry corridor constraints. The agent explores the feasible design space, and through the reward function, trajectories that violate these constraints are penalized. This guides the policy toward non-oscillatory gliding paths. This approach allows the RL agent to discover optimal trajectories that uses the full entry corridor, finding the best balance between bank angle and angle of attack to maximize range while ensuring skip-free flight.

6.15. Reinforcement Learning Environment

An environment class has been implemented that serves as the interface between the HGV physics simulation and the reinforcement learning framework. The architecture of the HGVEnv class is presented in Appendix A. It follows the Gymnasium standard designed by *OpenAI*, which enables reinforcement learning agents to interact directly with the trajectory model. The environment allows an agent to control the vehicle by adjusting the angle of attack and bank angle, while it is ensured that the motion, constraints and penalties follow the physical behavior of the HGV. The environment is built around the HGV class, described above, which contains the full 3-DOF dynamics of the vehicle.

The flow of the environment follows the Gymnasium pattern and can be divided into four phases: initialization, reset, step and termination.

6.15.1. Initialization

During initialization, the environment loads the aerodynamic data and vehicle constants, which include the mass, reference area, and the constraints for load factor, dynamic pressure, and stagnation point temperature, which define the physical limits of the simulation. The environment then creates an instance of the HGV physics class, which integrates the equations of motion over time and describes the motion of the HGV. Learning-specific parameters include the target index, which indicates which of the 36 target points should be pursued, and a train mode flag that enables randomization when set. This is done such that a single trajectory can be run using this environment, but also the complete training of the model.

Training mode

In training mode, three parameters are randomized. These are the latitude, heading and target point index. These parameters were selected deliberately, since they together form the complete configuration of a flight scenario on a rotating Earth. Randomizing them ensures that the trained reinforcement learning agent is not biased toward a specific geographical position, heading or target point, which is essential to produce a valid footprint. Flight path angle, speed, and altitude are not varied during training, as these are considered fixed initial conditions from which the HGV starts its trajectory.

The latitude determines the vehicle's initial position relative to the equator, which directly affects the Coriolis forces that are experienced during flight. At higher latitudes, the Coriolis deflection acts differently compared to the equator, where the rotational velocity component is the largest. By sampling across multiple latitudes, the environment exposes the learning agent to a range of Coriolis effects, which ensures that the final policy generalizes to any possible location on Earth.

The heading angle specifies the initial direction of flight relative to true north. Since the HGV dynamics are modeled on a spherical Earth, heading changes the great-circle trajectory, which influences the longitudinal and latitudinal position of the flight path. Different headings represent different heading orientations with respect to the Earth's rotation, which allows the agent to learn how to optimize trajectories regardless of initial flight orientation or intended target. This randomization is important to avoid overfitting to a specific eastward or westward bias that is introduced by Earth's rotation.

The third parameter, the target point index, selects one of the 36 equally spaced bearings around the starting point. Each index defines a great circle arc that extends radially outwards from the launch point, which is the framework for the footprint. By training with randomized target points, the agent learns to handle all possible bearing directions, which generalizes the model to the full domain.

Together, these three randomized variables give the entire space of possible starting and ending configurations on Earth. Randomizing these parameters is sufficient to generate the full reachable footprint because the other state variables, altitude, velocity, and flight path angle determine the flight performance and energy but not global directionality. In other words, altitude and speed control the size of the footprint but the combination of latitude, heading and target point define the global shape and orientation.

Importance of Randomization for Preventing Catastrophic Forgetting

Without randomization of initial latitude, heading, and target direction during the runs, the reinforcement learning agent would be exposed many times to a single, fixed environment. This leads to a

phenomenon that is known as catastrophic forgetting. This is when the agent is learning a new task, the learner can forget the information that was previously learned [82]. For this problem, it would mean that the trained policy becomes specialized for one specific geographic configuration and will fail when applied to different locations or headings on Earth.

Catastrophic forgetting becomes a problem since the policy in the reinforcement learning algorithm continuously updates the parameters based on recent experience. When all training comes from near-identical states, the neural network minimizes the reward function only for that subset of the state space. According to McCloskey et al. [83], behaviors that were once optimal are forgotten since training is overwritten and interfered with previously learned good policies. This significantly degrades performance on previous tasks.

By introducing randomization of the geographic and directional parameters, each training episode represents a unique environment configuration. This forces the neural network to develop a policy that is valid everywhere on Earth rather than memorizing a single trajectory solution. This approach mitigates catastrophic forgetting by ensuring continual generalization. When training is complete, the resulting model can be evaluated in each of the 36 radial directions on any location on Earth, without further retraining.

6.15.2. Reset Phase

The reset phase initializes a new episode. The HGV state is reset to the initial altitude, velocity, and flight path angle. The position of the vehicle is defined by the current latitude and longitude, which determine the location on Earth. From this starting position, the environment computes the set of possible target points that define the boundaries of the reachable footprint. The environment computes the geographic coordinates of the target points using the function that will be explained in Subsection 6.19.5.

6.15.3. Step Phase

Once the environment is reset, the step phase begins. In each time step, the reinforcement learning agent outputs a vector consisting of two control values, the angle of attack and bank angle. The complete step phase workflow is shown in the activity diagram in Appendix A. These control inputs are applied to the HGV model with rate limitations to eliminate unrealistic transitions. The vehicle state is then advanced forward in time by integrating the equations of motion for the fixed interval. After the new state is computed, the environment evaluates the resulting constraints, calculates the progress towards the target and generates a reward. This reward, together with the normalized observation of the new state, is returned to the agent.

6.15.4. Termination Phase

The simulation continues until a termination condition is reached. This occurs when the vehicle descends below a set final altitude. A final altitude of 5 km is set, but this can also be easily changed according to the mission objective.

6.15.5. Action Space

The reinforcement learning agent controls the HGV through two continuous inputs, the angle of attack and bank angle. These two parameters fully define the orientation of the aerodynamic force. Together, they control the longitudinal energy management and the lateral directional control of the HGV. The angle of attack can vary between -4° and $+60^\circ$ and the bank angle is defined between -90° and $+90^\circ$.

Angle of Attack and Bank Angle Rate Limitations

Hypersonic Glide Vehicles are subject to strict actuator and structural limitations for control surfaces that modify the angle of attack and bank angle. At each simulation step, the reinforcement learning agent outputs an action vector:

$$a_t = \begin{bmatrix} \alpha_t \\ \sigma_t \end{bmatrix}$$

Although this is a 3-DOF model, to get realistic results, the reinforcement learning environment applies rate constraints, which makes sure that control inputs cannot change instantaneously between two con-

secutive time steps beyond the rate limit. The rate limits are defined by the maximum time derivatives of the control variables.

$$\dot{\alpha}_{\max} = 1^\circ/\text{s}, \quad \dot{\sigma}_{\max} = 3^\circ/\text{s} \tag{6.53}$$

These rate limitations correspond to how fast the HGV can physically change the orientation through body deflection or roll commands. These constraints make sure that each new control command remains within a reachable range. The agent's action between two consecutive steps is limited as follows for a time step of $\Delta t = 5\text{s}$:

$$\Delta\alpha_{\max} = \dot{\alpha}_{\max}\,\Delta t = 5^\circ, \quad \Delta\sigma_{\max} = \dot{\sigma}_{\max}\,\Delta t = 15^\circ \tag{6.54}$$

which is applied to the physics model of the HGV. The resulting aerodynamic forces from this action vector update the equations of motion, which in turn determine the new position, altitude, and velocity. Together, the angle of attack and bank angle give the agent full control authority. By learning how to coordinate both inputs, the agent discovers strategies to extend range, avoid constraint violations and reach the target point in the most efficient matter.

6.15.6. Observation Space

The observation space defines what the reinforcement learning agent perceives from the environment at every time step. These observations must contain all relevant information needed to make control decisions that influence the system's future state. If important variables are missing or are not well scaled, the learning process becomes unstable or it fails to converge to a physically meaningful behavior. For an HGV, the chosen observables must describe both the flight dynamics and the geometric relation to the target.

In the implemented environment, the state vector consists of five normalized observables: altitude, velocity, flight path angle, difference between bearing to target and the current heading, and the distance to the target point. Together these parameters describe the full motion of the vehicle on a spherical Earth. The state vector is as follows:

$$s_t = [\, h_{\text{norm}},\ v_{\text{norm}},\ \gamma_{\text{norm}},\ \kappa_{\text{norm}},\ d_{\text{norm}} \,]$$

where κ is the bearing to target minus the current heading.

Altitude h

The altitude defines the position of the HGV in the atmosphere and this affects both aerodynamic forces and constraints. To maintain scale consistency, altitude is normalized relative to the maximum expected altitude h_{max} set at 160 km and minimum altitude h_{min} set at 5 km:

$$h_{\text{norm}} = \frac{h - h_{\min}}{h_{\max} - h_{\min}}$$

The reinforcement learning policy interprets the altitude as a dimensionless measure of atmospheric depth, ranging from 0 to 1.

Velocity V

The velocity defines the total kinetic energy of the vehicle. To make this variable dimensionless, it is normalized by the circular orbital velocity at sea level:

$$V_c = \sqrt{g_0 R_E} \tag{6.55}$$

The normalization then becomes:

$$v_{\text{norm}} = \frac{v}{V_c}$$

This normalization makes sure that the velocity input remains in the range between 0 and 1.

Flight Path Angle γ

The flight path angle gives the inclination of the velocity vector relative to the local horizontal. This defines whether the HGV is descending, maintaining level flight or is climbing. It is normalized by its theoretical range of $-\pi/2 \leq \gamma \leq \pi/2$:

$$\gamma_{\text{norm}} = \frac{\gamma}{\pi/2}$$

This normalization gives all the possible values in the interval between -1 and 1. By experimenting with this, it was found that the model works better if in the case of negative values, the normalization is also in the negative range.

Difference Between Target Bearing and Heading κ

The heading error measures how good the vehicle's current direction aligns with the direction to the target. It is defined as the difference between the bearing to the target (β) and the current heading (ψ):

$$\kappa = \beta - \psi \tag{6.56}$$

This variable is of great importance to the lateral guidance of the HGV. If κ equals 0, the HGV is flying directly toward the target point. Positive or negative values indicate that the vehicle is heading to the left or to the right of the target direction.

Since both the heading and bearing to target are angular parameters defined on a sphere, the difference is wrapped to the range $[-\pi, \pi]$. The normalized heading error is as follows:

$$\kappa_{\text{norm}} = \frac{\kappa}{\pi}$$

This maps the value in the range of [-1, 1]. This variable is not a control command, but an observed quantity that gives the reinforcement learning model spatial awareness. During training, the environment randomly assigns the target direction at the beginning of each episode. This means that the policy is exposed to different angular differences between the heading and the target direction. Over many episodes, the model learns that different target points correspond to different values of κ and how these relate to optimal control behavior.

After training, the learned policy does not explicitly compute a corrective bank angle. Instead, it has implicitly learned how to interpret this angle difference in context with other state variables. For example, the same angular deviation at high altitude and low dynamic pressure require a different bank angle than at lower altitude, where aerodynamic forces are stronger.

Therefore, κ is a geometric indicator rather than a direct steering signal. It allows the trained model to generalize directional guidance in all 36 target directions.

Distance to Target d

The final observable is the distance to the target. This gives the remaining great-circle distance between the HGV's current position and the target point. This parameter measures how far the vehicle still needs to travel. To give numerical stability and generalization, this distance is also normalized by the maximum possible range d_{max} and will also be in the range of [0, 1]:

$$d_{\text{norm}} = \frac{d}{d_{\text{max}}}$$

Why These Five Observables

These five observables together form the complete state representation of the HGV used by the reinforcement learning policy. They were selected to provide the minimal amount of information necessary for the agent to understand the vehicle's physical state and make control decisions that are optimal. Each of these observables contributes to a different part of the flight course. The altitude and velocity describe the energy state of the vehicle. This defines how much potential energy remains available and where in the trajectory the HGV is. The flight path tells how steeply the vehicle is moving up or down. This links altitude and velocity and defines how the vehicle trades vertical speed for horizontal speed, or vice versa, along the trajectory. The heading difference measures the angular offset between the current heading and the direction toward the target, allowing the agent to align its trajectory toward the target. Finally, the distance to target measures progress, which allows the policy to evaluate what the control choices do on the progression to the target. Over time, the model will learn that it receives higher rewards when it gets closer to the target.

Role of Observables during Training

During training, the RL agent interacts repeatedly with the simulation environment. At each time step, it receives the observation vector and produces control actions. The environment then simulates the resulting motion using the full HGV dynamics and calculates rewards. Afterwards, it updates the state for the next step.

Important to note is that only these five normalized variables are provided to the policy. The neural network is never given explicit aerodynamic parameters, it must implicitly learn their relationships through the physical evolution of the environment. For example, by observing that certain combinations of altitude, speed and flight path angle will lead to high penalties or low rewards, the network learns the underlying physics without being directly told the equation.

Role of Observables After Training

Once the model has been trained, the learning process is complete. The neural network parameters are fixed. At this stage, the agent relies exclusively on these five observables to make all decisions.

For example, when executing a trajectory, the trained agent observes a certain combination of altitude, velocity, flight path angle, heading difference, and distance to target. Based on patterns it learned during training, it outputs an appropriate change in bank angle and angle of attack that will minimize the range towards that specific target point. Thus, it does not know the exact equations of motion, but it recognizes that similar conditions previously led to better outcomes when turning or pitching a certain way.

6.15.7. Coriolis Forces

A deliberate choice that has been made is that latitude does not appear in the observation space. Despite this, the trained models successfully produce footprints, correctly compensating for Coriolis effects at different latitudes, as demonstrated in Section 6.19.8.

The decision to exclude latitude from the observation space was made after experimentation with both approaches. When latitude was included as an observable, the trained agent showed worse performance for the same training duration. The issue was reduced generalization capability around the globe. By providing latitude explicitly, the agent learned to associate specific latitude values with corresponding Coriolis compensation strategies, instead of learning the more general skill of compensating for disturbances.

The Coriolis term introduces a lateral acceleration that depends on the (unobserved) latitude. This acceleration affects the evolution of the observed states, and therefore appears to the agent as a consistent disturbance in the state dynamics. The RL policy learns to map these disturbed state trajectories to control actions that generate aerodynamic forces counteracting the effect. Because the aerodynamic forces are several orders of magnitude larger than Coriolis accelerations during flight, the agent effectively rejects the disturbance without explicitly observing latitude. This is the agent's disturbance rejection behavior. Disturbance rejection is the ability of a control system to maintain desired performance despite external perturbations. During training, the environment randomly selects from a continuous set of latitudes, which exposes the agent to different Coriolis force magnitudes and

directions. Although the agent cannot directly observe the latitude, Coriolis forces are systematic disturbances in vehicle dynamics. This affects heading rate, required bank angle to maintain course and the relationship between control inputs and trajectory outcomes. Through this implicit learning mechanism, the agent develops a robust policy that generalizes effectively across all latitudes without requiring explicit knowledge of the geographical position.

6.15.8. Constraint Handling and Penalty Function

The difficulty of the trajectory optimization of a hypersonic glide vehicle is partly caused by the limiting constraints. Exceeding the maximum allowable load factor leads to structural failure, surpassing the dynamic pressure bound risks aerodynamic breakup and violating the thermal limit causes material degradation and failure. These constraints define the feasible flight envelope and must be respected in the trajectory optimization framework.

However, the challenge lies in how these constraints are represented in the learning and optimization process. Reinforcement learning relies on feedback to improve the policy. If constraints are enforced too harshly, such as by terminating the simulation immediately upon violation, the algorithm receives little gradient information and cannot learn how to operate near the boundaries of feasibility. However, if the constraints are enforced too softly, the optimizer may converge to trajectories that do violate constraints.

To resolve this, the constraints are encoded through penalty functions. A penalty function transforms the distance between the current state and its limit into a negative contribution to the reward. In this way, constraint violations do not simply end the episode, but instead contribute to a smoothly increasing cost that grows with the severity of the violation. Trajectories that are still moderately far away from the exceed point are penalized lightly, while trajectories that come close to the exceed point are penalized more heavily.

This methodology not only improves the stability of training, but also reflects practices in non-linear trajectory optimization, where soft constraints are frequently used to avoid infeasible solutions while still allowing exploration of the neighborhood around the constraint surface. In the context of the HGV, the design of the penalty function is critical since the most extreme range trajectories often lie very close to the constraint boundaries. There were three training methodologies considered for the penalty, a cut-off penalty, a block penalty and exponential penalty.

Cut-off penalty

The most straightforward option is a cut-off penalty, often referred to as a death penalty in optimization. In this approach, the trajectory is terminated as soon as a constraint is violated, and the roll-out provides no further reward. While this guarantees that all collected trajectories remain strictly feasible, it deprives the learning algorithm of meaningful feedback on how the trajectory was infeasible. As a result, exploration is severely limited and the convergence becomes inefficient.

Block penalty

A second option is the block penalty, where the safety buffer is divided into a number of zones, each associated with a fixed penalty. As the vehicle approaches the constraint boundary, the penalty increases stepwise from zone to zone. This provides more information to the agent than the cutoff method. In principle, this approach works and can guide the agent away from unsafe regions, but the stepwise increase makes it less smooth than an exponential formulation. The exponential penalty is therefore preferred, as it provides a more continuous and stronger learning signal close to the constraint, while the block penalty remains a workable but less effective alternative.

Exponential penalty

The exponential penalty is implemented as a smooth shaping function that activates before the constraint is actually violated. Each constraint has a defined safe margin. When the vehicle is comfortably within this margin, the penalty is zero and no influence is exerted on the reward. As the trajectory approaches the constraint boundary, the penalty begins to grow gradually, ensuring that the agent already receives negative feedback while still inside the feasible domain.

The penalty is computed by first normalizing the margin to a range between 0 (in the safe margin) and 1 (in the constraint boundary). Over this interval an exponential function is applied:

$$P(m) = \text{max\_penalty} \cdot \frac{\exp(k_{\text{steep}}\,\hat{m}) - 1}{\exp(k_{\text{steep}}) - 1} \tag{6.57}$$

where $\hat{m}$ is the normalized margin, k_{steep} is the steepness parameter, and max_penalty is the negative weight assigned to full violation. This formulation starts almost flat, no penalty in the safe zone, increases smoothly as the margin shrinks and rises steeply near the boundary.

An important feature of this formulation is that it can be tuned. The steepness parameter controls how quickly the penalty grows. With a low steepness the penalty increases more gradually, allowing the model to explore closer to the boundary, whereas with a high steepness the penalty becomes sharp and strongly discourages the agent from approaching the limit.

This approach has two benefits. First, the agent does not get cut off without feedback, as in cut-off penalties. Second, the exponential curve provides continuity and smoothness, which is critical for stable reinforcement weight updates. Similar exponential penalties are common in optimal control and RL literature for handling path constraints, since they allow exploration near boundaries (where optimal solutions usually lie) while discouraging violations.

Figure 6.11 shows the exponential penalty function for three different values of the steepness parameter. The horizontal axis represents the constraint margin, with the safe zone at -500 m/s and the violation boundary at 0 m/s, where these margins represent the velocity margin until a constraint is violated. The vertical axis shows the penalty magnitude, which grows smoothly from zero in the safe zone to the maximum negative value at the boundary. A lower steepness value produces a more gradual slope but begins penalizing earlier, resulting in larger cumulative penalties as the agent approaches the constraint. Higher steepness values result in sharper curves that remain near zero longer before dropping steeply near the boundary. The higher steepness allows for more exploration near the constraint boundary. This visualization demonstrates how the steepness parameter controls the aggressiveness of the penalty growth, allowing flexible tuning of the constraint handling strategy.

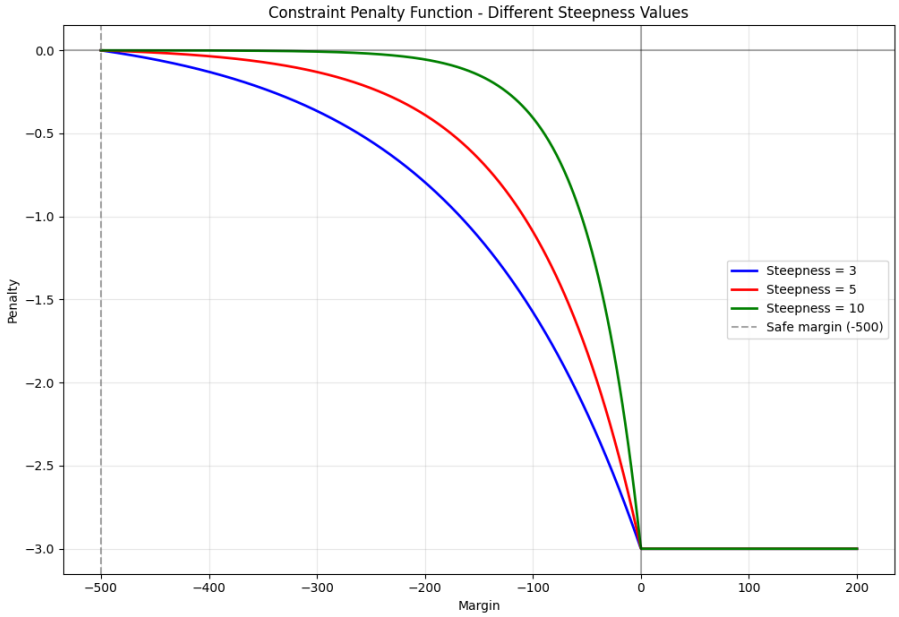


Figure 6.11: Exponential penalty function with varying steepness.

Dynamic Penalty Trial

Originally, a dynamic penalty strategy was explored to use during the learning process. The idea behind a dynamic penalty is to gradually increase the penalty magnitude over time, to balance exploration and

constraint satisfaction. Early on in training, the agent is allowed to explore freely, even into infeasible regions, to better understand the boundary of the feasible domain. As training progresses, the penalty grows stronger, encouraging convergence to feasible and optimal behavior. The penalty factor was increased linearly throughout training according to the following formula:

$$\lambda_k = \lambda_{\text{max}} \cdot \frac{k}{K_{\text{max}}} \tag{6.58}$$

where λ_k is the active penalty factor at episode k, λ_{max} is the full penalty magnitude and K_{max} is the total number of training episodes. This means that the agent receives 0% of the penalty in the first episode and the full penalty in the final episode.

However, in practice, this approach did not give successful results for this application. The agent continued to violate the physical constraints because it did not learn the feasible boundary fast enough before the penalties became dominant. Although this approach has been shown to improve convergence properties of reinforcement learning [84], they were not effective in this specific trajectory optimization problem. Therefore, the approach was abandoned in favor of the static exponential penalty formulation as described above.

6.15.9. Reward Structure

In reinforcement learning, the reward function, or equivalently the cost function, serves as the central element that defines the learning objective of the agent. Unlike traditional optimization methods, where the objective function is explicitly minimized under given constraints, reinforcement learning relies on a reward signal that must provide the desired behavior. Therefore, the design of this reward function is of great importance; not only does it determine the final policy but also influences the stability, convergence speed, and feasibility of the learning process.

In the initial design of the reinforcement learning environment, the reward function was constructed around the ratio of ground distance gained to energy lost, which will be explained in more detail in the following section. The motivation for this choice was to encourage the agent to fly as efficiently as possible, converting the available energy into the maximum horizontal range. By rewarding efficiency at each step, it was expected that the agent would learn to conserve energy on the trajectory and thus maximize its overall performance. This approach worked well for straight-line trajectories, but the agent struggled to learn lateral maneuvers under this reward structure, which is required for footprint generation.

However, during the course of this research, it became clear that this formulation introduced an unnecessary complication. The total episode reward in reinforcement learning is the sum of all step rewards, meaning that what ultimately matters is the final return accumulated over the trajectory. If the objective of the task is to get as close as possible to a target point, then maximizing the cumulative reward is equivalent to maximizing the total reduction in distance to that target. In other words, energy conservation is already implicitly enforced when the agent is trained to minimize the final distance to the target. A trajectory that wastes energy unnecessarily will end further away from the target point and therefore yield a lower return.

This realization led to a reformulation of the reward function, which is a more direct metric of mission success. The agent is now rewarded only on the basis of how much closer it is to the target during each step. This new approach still has the benefits of step-by-step feedback, but ensures that the cumulative reward corresponds to the objective of the mission. An important consequence of this reformulated reward structure is that the agent's behavior is no longer shaped by designer-imposed biases about how the vehicle should fly. The only information provided through the reward is whether the distance to the target has been reduced. As a result, the agent is entirely free to explore and discover whichever trajectory most effectively achieves that goal within the given dynamics and constraints. No preference is encoded for specific control strategies, flight profiles, or energy management techniques. These come naturally if they contribute to reaching the target.

When determining a footprint, a set of unreachable target points is used and the distance is minimized to each of these. Since the targets are positioned beyond maximum range, the agent consistently receives

rewards for progress without ever reaching them, naturally learning to fly boundary trajectories. The footprint emerges from the terminal positions of all trajectories. Footprint generation is further explained in Section 6.19.

Step Reward Definition

The reward that the agent receives at every simulation step is made up of two components: the progress term and the penalty term. The progress term rewards the vehicle for moving closer to the target point, while the penalty term discourages the violation of the constraints.

The progress reward is calculated from the change in great-circle distance between the HGV and the target point. The distance to the target point at the previous step is defined as d_{t-1} and the distance to the target point at the current step is d_t. The difference between these two values gives the instantaneous reward at the step:

$$r_{\text{prog}} = d_{t-1} - d_t$$

The higher the reward, the closer it is to the target. This formulation gives the agent instantaneous feedback on the progress toward the mission objective. Importantly, because the reinforcement learning return is the sum of all step rewards over an episode, the cumulative reward simplifies to the following:

$$R = \sum_{t=1}^{T} (d_{t-1} - d_t) = d_0 - d_T \tag{6.59}$$

where d_0 is the initial distance from the target and d_T the final distance to the target. This formula represents the net reduction in distance between the initial and final states. In this way, the stepwise formulation gives immediate feedback at each action, while the overall return gives the success of the trajectory in approaching the target point.

To make sure that the constraints are satisfied, the penalty term is included, as described in the above section. At each step, the simulation computes three constraint margins corresponding to the load factor, dynamic pressure, and stagnation point temperature. The largest of these margins, m_t, determines how close the vehicle is to violating any constraint. The total step reward combines the progress reward with the penalty reward:

$$R = \sum_{t=1}^{T} r_t = \sum_{t=1}^{T} [(d_{t-1} - d_t) + P(m_t)] \tag{6.60}$$

Here P is the constraint function that exponentially scales with the margin.

6.15.10. Energy Height Reward Function Formulation

An earlier version of the cost function was formulated around the concept of energy height, which expresses the vehicle's total mechanical energy in terms of an equivalent altitude. The motivation for this choice was that the fundamental objective of range maximization can be expressed as an energy management problem. The vehicle starts its descent with a finite energy budget, which consists of potential and kinetic energy. The trajectory that converts this energy most efficiently into horizontal range achieves the maximum glide distance. According to Mooij et al., within the energy-state approximation the maximum range is obtained by maximizing the magnitude of dE_h/dR_f [38].

Incremental Energy Height Loss ΔE_h

The concept of energy height provides a compact way to express the total mechanical energy of a vehicle in terms of a single equivalent altitude. It combines both potential energy and kinetic energy into one scalar measure in units of meters. The total specific mechanical energy of a vehicle is given by:

$$E = \frac{V^2}{2} + gh, \tag{6.61}$$

where the first term is the kinetic energy per unit mass and the second term is the potential energy per unit mass. Here V is the velocity magnitude, h the altitude, and g the gravitational acceleration.

Dividing by g, an equivalent altitude is obtained:

$$E_h = h + \frac{V^2}{2g}. \tag{6.62}$$

This is called the energy height. It expresses the vehicle's altitude, representing stored gravitational potential energy. In other words, if all the kinetic energy of the vehicle was instantly converted into potential energy, E_h would be equal to the altitude the vehicle could reach.

The energy height is used as an independent variable in trajectory studies, since it monotonically decreases during unpowered flight, while altitude and velocity can vary up or down [38]. Moreover, every unit of energy height lost corresponds to a portion of the vehicle's total energy budget being spent. The efficiency of the trajectory can then be judged by how much horizontal distance is gained per unit of E_h dissipated.

The incremental loss of energy height between two steps is therefore defined as:

$$\Delta E_h = E_{h,t-1} - E_{h,t} \tag{6.63}$$

This difference is always positive (or zero in the theoretical limit), reflecting the fact that the vehicle irreversibly loses energy to the atmosphere.

Using ΔEh as denominator of the reward function has a clear physical meaning; it quantifies the 'cost' of making progress. The more energy height is spent per unit distance, the less efficient the trajectory. In contrast to this, if the vehicle covers a large ground distance while losing a small amount of energy, the efficiency is high.

Efficiency Ratio $\frac{\Delta R}{\Delta E_h}$

The efficiency of the glide can be expressed as the ratio between incremental ground distance and energy height lost:

$$\eta = \frac{\Delta R}{\Delta E_h} \tag{6.64}$$

This ratio has a clear interpretation: it measures the number of meters of horizontal range gained per meter of energy height dissipated. A higher value of η indicates a more efficient trajectory, in which the available energy is converted into ground distance with minimal losses. In contrast, a low value of η reflects poor energy management, where energy is dissipated without contributing much to the total range.

This formulation is directly linked to the trajectory optimization principle described by Mooij [38], who emphasizes that within the energy-state approximation the maximum achievable range of a re-entry vehicle can be evaluated in terms of its energy height. The energy-state approximation assumes that kinetic and potential energy can be exchanged back and forth in zero time without loss of total energy. By presenting the reward in terms of $\frac{\Delta R}{\Delta E_h}$, the reinforcement learning agent is motivated to follow the same principle: extend the range by maximizing the distance covered on the ground per unit of energy expended.

Logarithmic Reward Formulation

Although the ratio of horizontal distance increment to energy loss captures the correct physical intuition, using it directly as a reward signal leads to difficulties. In particular, reinforcement learning algorithms are highly sensitive to the scale and numerical behavior of the reward. Very small energy losses can cause the ratio to take on unboundedly large values, producing unstable or misleading learning signals. At the same time, at high altitudes where air density is low, the vehicle can cover extremely large ground distances while wasting only a small amount of energy. If this effect is translated directly into the reward,

the agent would receive disproportionately large positive signals for behavior that is not particularly meaningful for the overall trajectory, since range achieved in the upper atmosphere contributes little to the final outcome. In addition, at lower altitudes, the same measure would assign relatively modest rewards, even though small changes in energy management can have decisive effects on the total achievable range. By applying the logarithmic transformation to the distance to energy ratio, these excessive rewards are naturally compressed. As a result, the reward signal remains balanced across the entire descent profile, giving appropriate weight to both the initial glide and the lower-altitude phases where careful energy management is most decisive for the final range.

The reward function is therefore defined as:

$$r = \log\left(1 + \frac{\Delta R}{\Delta E_h}\right). \tag{6.65}$$

The addition of 1 inside the logarithm ensures nonnegative values even for very small efficiency ratios, while the logarithmic form compresses excessively large ratios into manageable magnitude.

Range Gain per Energy Height Loss

To directly visualize the efficiency measure of the reward function, another Python script was developed. This script computes the ratio between incremental ground distance and incremental energy height loss at each step, and applies the logarithmic transformation used in the cost function.

Figure 6.12 shows the results for several fixed angles of attack. The vertical axis indicates the log-transformed ratio, while the horizontal axis shows the flight time. Each curve represents the instantaneous reward contribution of the trajectory at a constant AoA.

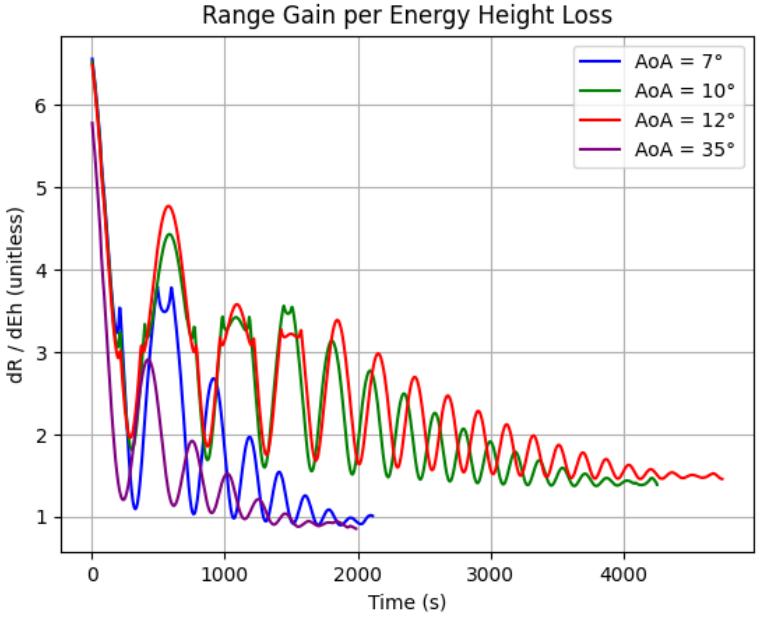


Figure 6.12: Logarithmic Range Gain per Energy Height Loss for various AoA.

The figure clearly illustrates how AoA affects glide efficiency. At very high AoA, the ratio remains low throughout, reflecting that energy is being dissipated rapidly while only modest ground distance is achieved. At the other extreme, very low AoA shows fluctuating values: while periods of seemingly high efficiency occur early in the flight when ΔE_h is small, the ratio eventually collapses as energy is wasted inefficiently. The intermediate angles of attack show consistently higher values over a long time period.

It can also be seen from the figure why the logarithmic transformation in the reward function is necessary. Without the log, short phases with very small ΔE_h, especially at high altitudes, produce extremely large values of $\Delta R/\Delta E_h$. These large spikes are not truly indicative of useful range performance, but would dominate the reward signal if left unmodified. By applying the logarithmic function, these extreme peaks are compressed into moderate values, while still preserving the relative efficiency ordering of different trajectories.

Although this formulation is consistent with the energy-based optimization framework, this formulation was ultimately replaced. During development, it became clear that this energy-based reward introduced complexity and potential biases that complicated the learning process. Instead, the final adopted reward is much simpler and less biased. The agent is rewarded purely based on minimizing the distance to the target point, where all target points are placed out of reach. This approach eliminates biases toward specific flight regimes and allows the agent to discover optimal trajectories without being constrained by assumptions. The final reward formulation was described in Section 6.15.9.

The energy height analysis presented here remains valuable for understanding the physical principles of glide efficiency and was an important intermediate step in the development process, but all results presented in this thesis use the simpler distance-based reward formulation.

6.16. Reinforcement Learning Training

The development of an effective control policy for HGV is a challenging optimization problem that requires to balance multiple competing objectives while still adhering to the constraints. Traditional control approaches, such as trajectory optimization through direct or indirect methods, struggle often with nonlinear dynamics and complex aerodynamic relationships and constraint boundaries that characterize hypersonic flight. This was also found when experimenting with several traditional approaches. To address these challenges, this research uses Soft Actor-Critic (SAC) algorithm, to learn robust control policies that can navigate the complex state action space of HGV trajectory optimization. The Soft Actor-Critic algorithm is used via the *stable-baselines3 (SB3)* Python library, which is based on research project *Baselines* by OpenAI and the SAC algorithm developed by Berkeley University.

Among the available RL frameworks, SB3 was chosen for its proven performance in continuous control tasks. Additionally, this is a production-ready implementation that contains state-of-the-art RL algorithms that have been extensively validated. Within SB3, SAC was chosen over alternatives such as Proximal Policy Optimization (PPO) for several reasons specific to the HGV trajectory optimization problem. PPO is an on-policy algorithm, which means it can only learn from experiences collected by the current policy and must discard these data after each update. In contrast, SAC is off-policy and stores all experiences in a replay buffer, which allows each computationally expensive HGV simulation to be reused multiple times for learning. Additionally, SAC's entropy maximization objective encourages exploration throughout training, which helps the agent discover diverse control strategies without prematurely converging to suboptimal solutions. This is especially valuable for exploring the complex, non-linear relationships in hypersonic flight where the optimal control strategies are not known beforehand.

Reinforcement learning offers several advantages for this application. Unlike traditional optimization methods that require accurate gradients of the objective function with respect to control inputs, RL methods learn directly from experience through interaction with the environment. This approach that is built on experience is useful for systems with complex dynamics such as HGVs, where aerodynamic coefficients vary nonlinearly with Mach, Angle of Attack, and altitude. The following sections lay out the theoretical foundations of the SAC algorithm, the training algorithm architecture that is used, and the specific hyperparameter choices that gave effective learning for HGV control. The overall training workflow is shown in `train_hgv` activity diagram in Appendix A.

6.16.1. Reinforcement Learning Fundamentals

Before going into the specific algorithm used, it is important to understand the basic framework of reinforcement learning. Reinforcement learning is a way to teach a system to make decisions by letting it interact with an environment. Instead of being told what to do, the agent tries things out and sees what works well. Then it gradually improves its behavior. Feedback is provided in the form of a reward.

If the agent makes a good decision, it will receive a high reward. If it makes a poor decision, the reward will be lower. The reward function will define what is objectively good and bad for the agent [85].

The concepts of state and action are commonly used in reinforcement learning. An action is a decision that the agent will need to learn to make, and a state is a factor that the agent can take into consideration when making that decision. The state used to determine an action can include a model of the environment which can also contain past environmental conditions. This model is part of the agent's internal state at the moment it chooses an action. Therefore, reinforcement learning does not limit what a state is. It can be simple or complex with a full memory of past events. Some learning methods, such as basic neural networks, react only to what is happening right now. They do not use memory. However, reinforcement learning can work with both styles [85].

Thus, the agent observes the state and then chooses an action. The environment responds by giving a new state and a reward. The agent then uses this experience to improve its decisions. Over time, the agent learns which actions lead to better results. The goal is not just to get high rewards in this moment in time, but to get the best total reward over many steps into the future. A schematic view of how reinforcement learning works is shown in Figure 6.13.

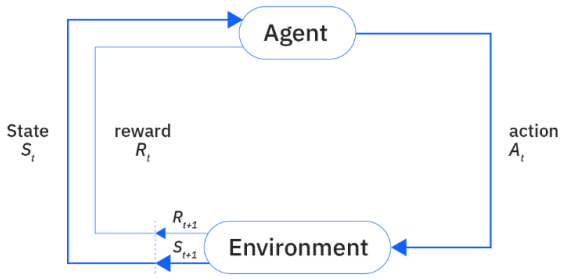


Figure 6.13: Working of Reinforcement Learning [86].

6.16.2. Soft Actor-Critic Algorithm

The Soft Actor-Critic (SAC) algorithm is an off-policy reinforcement learning method. This means that it can learn from experiences collected by any policy, not only the current one. During training, the agent stores all experiences, the state, action, reward, and next state in a replay buffer. When the policy improves and the behavior changes, these old experiences, which were collected when the policy was different, remain useful for learning. The algorithm can sample these past experiences from the buffer and extract valuable information on which actions lead to good or bad outcomes, even though these actions were selected by an earlier and different version of the policy. The off-policy nature of SAC improves the sample efficiency, since each experience can be reused multiple times for learning instead of being thrown away after one use.

SAC is also model-free, which means that it learns entirely from trial-and-error experience without requiring a predictive model of how the environment works. So instead of trying to predict if it takes a certain action what the next state will be, it directly learns if it takes that action what reward will ultimately be received. This model-free approach is especially suitable for complex systems such as HGVs where building predictive models of aerodynamics and trajectory dynamics would be challenging.

SAC is specifically designed for continuous action spaces, where actions are numbers instead of discrete choices. For the HGV problem, the actions are continuous for angle of attack [$-4°$, $60°$] and bank angle [$-90°$, $90°$]. This continuous formulation allows for precise control that would not be possible with discrete actions.

SAC differentiates itself from traditional reinforcement learning algorithms by using entropy maximization as an objective function. Traditional reinforcement learning algorithms have a simple goal, namely to maximize the total reward that is accumulated over an episode. The agent learns to select actions that lead to the highest cumulative reward. SAC adds an additional consideration. It wants the agent

to maximize the reward while also maintaining randomness in the action selection, which is referred to as entropy.

The mathematical objective that SAC maximizes is the following [33]:

$$J(\pi) = \mathbb{E}_{\tau\sim\pi}\left[\sum_{t=0}^{T}\gamma^t(r(s_t,a_t) + \alpha H(\pi(\cdot|s_t)))\right] \tag{6.66}$$

Where τ represents a trajectory, which is a sequence of states and actions from the episode start to termination. The summation $\sum_{t=0}^{T}$ indicates that the reward is summed over all time steps from the episode start to termination. The term $r(s_t, a_t)$ is the standard reward the agent receives for taking action a_t in state s_t. The term α is the temperature parameter that controls how much entropy is valued relative to the reward. This parameter balances between maximizing reward, finding actions that give high returns, and maintaining exploration, keeping the policy random to avoid getting stuck in suboptimal solutions. Higher α means the agent values exploration heavily and will try many actions even if they do not immediately seem optimal. Lower α means that the agent focuses almost entirely on the reward and will quickly converge to what appears to be the best. This term helps prevent early convergence and makes the agent more robust.

The entropy term $H(\pi(\cdot|s_t))$ quantifies the degree of randomness in the policy's action distribution in the state s_t. It is defined as:

$$H(\pi(\cdot|s_t)) = -\mathbb{E}_{a\sim\pi}[\log\pi(a|s_t)] \tag{6.67}$$

$\pi(a|s_t)$ represents the probability of selecting action a given state s_t. The entropy measures how spread out this probability distribution is across the action space. When the policy concentrates most of the probability mass on a single action, the entropy is low, which is deterministic behavior. When probability is distributed more uniformly across multiple actions, the entropy is high, indicating stochastic behavior.

By including this entropy bonus weighted by α in the objective function, SAC encourages the policy to keep exploring different actions during training instead of prematurely committing to a single strategy that might not be the optimal strategy.

The expectation operator $\mathbb{E}_{\tau\sim\pi}$ represents the average of all possible trajectories that could occur when following policy π. This averaging accounts for the randomness in the policy, which samples actions from a probability distribution instead of selecting actions deterministically. Running the same policy multiple times produces different trajectories, each with a different total reward.

In practice, the algorithm cannot compute the expectation analytically. Instead, it estimates the average by running many episodes during training and collecting rewards. Each episode provides one sample trajectory, and averaging over thousands of episodes approximates the true expected performance. This sampling-based estimation is why extensive training with many episodes is required.

Note that the discount factor γ in this formula is not explained here, but will be done later in Chapter 6.16.6.

SAC uses the actor-critic architecture using two separate neural networks, each with a separate role. The actor network represents the policy and handles the selection of actions. Given a state observation, it produces a probability distribution over possible actions. During training, actions are sampled from this distribution to encourage exploration, as explained above, while during evaluation the mean of the distribution gives deterministic behavior.

The actor network is parameterized by weights θ, denoted by π_θ, while the two critic networks are parameterized by weights ϕ_1 and ϕ_2, denoted Q_{ϕ_1} and Q_{ϕ_2} respectively. During training, experiences are stored in a replay buffer denoted D.

Critic Training

The critic network estimates the Q-value. This is the expected cumulative reward for taking a particular action in a given state and then following the policy thereafter. The mathematical formulation for the Q-value is as follows [87]:

$$Q^{\pi}(s,a) = \mathbb{E}_{\tau\sim\pi|s_0=s,a_0=a}\left[\sum_{t=0}^{\infty}\gamma^t r(s_t,a_t)\right] \tag{6.68}$$

In practice, this true Q-value cannot be computed exactly, so SAC uses neural networks to approximate it. SAC uses two critic networks with parameters ϕ_1 and ϕ_2 that learn to estimate $Q^{\pi}(s,a)$, denoted as $Q_{\phi_1}(s,a)$ and $Q_{\phi_2}(s,a)$. Using two critics is done for a problem in Q-learning called overestimation bias. Q-values are important since they provide the basis for deciding which action to take. If the agent is in state s and must choose between action a_1 and action a_2, it can compare $Q(s,a_1)$ versus $Q(s,a_2)$:

$$Q_{\text{min}}(s,a) = \min(Q_{\phi_1}(s,a), Q_{\phi_2}(s,a)) \tag{6.69}$$

The action with the higher Q-value will lead to better total performance over the rest of the trajectory. The actor network learns to output a policy that selects actions with high Q-values.

When estimating Q-values by looking at maximum expected future rewards, noisy estimates systematically overestimate true values. This overestimation becomes greater as the agent uses inflated Q-values to update earlier values, which leads to unstable learning where the agent becomes overconfident about poor actions. By maintaining two independent critics and using the lower prediction of the two, it is less likely to overestimate. If one critic overestimates due to noise, the other likely will not overestimate identically, so the minimum provides a more reliable estimate.

The critic networks learn by minimizing the temporal difference error. This is the gap between their predicted Q values and their target values. When the agent takes action a in state s, receives reward r and transitions to the next state s', the critic compares the prediction $Q(s,a)$ against the target, which is the reward plus the value of the next state s'.

The critic learns by comparing the prediction with what actually happened. For each experience in a batch sampled from the replay buffer, the critic predicted some Q-value. After the action was taken, the actual reward that is received and the next state are known. Then the target value can be computed. The temporal differences error is squared, such that large mistakes are penalized heavily and is averaged over many experiences from the replay buffer. The replay buffer will be further explained in Section 6.16.6.

Actor Training and Q-Value Feedback

The actor network learns to select actions that maximize Q-values while maintaining exploration entropy. The actor and the critic work together in a continuous feedback loop during training.

The actor proposes actions based on the current policy, it looks at the observables, and outputs a probability distribution over possible actions. The critic then evaluates these proposed actions by computing Q-values, which indicate how much total reward each action is expected to achieve. The actor uses these Q-value evaluations as a learning signal. If the critic assigns a high Q-value to an action, the actor increases the probability of selecting that action in similar states.

This Q-value feedback is implemented through the actor loss function [33]:

$$L(\theta) = \mathbb{E}_{s\sim D, a\sim\pi_\theta}\left[\alpha\log\pi_\theta(a|s) - \min_{i=1,2} Q_{\phi_i}(s,a)\right] \tag{6.70}$$

where s is sampled from the replay buffer D and action a is sampled from the current policy π_θ. This loss has two competing components. The term $\min_{i=1,2} Q_{\phi_i}(s,a)$ gives the Q-value evaluation from the critics, where it takes the minimum of both critics' predictions. The minus sign means minimizing this

loss maximizes the Q-values. So the actor learns to select actions that the critics evaluate as having high Q-values. This is how the critic's feedback guides the actor's training.

The second component is the entropy term α times $\log \pi_\theta(a|s)$, where $\pi_\theta(a|s)$ is the probability the policy assigns action a in state s. Since logarithmic probabilities are always negative, this entire term is also negative. This term penalizes policies that become too deterministic. If the policy concentrates all the probability on one action, this penalty increases, which pushes the policy to spread the probability across multiple reasonable actions.

During training, the feedback loop operates continuously. The actor proposes actions based on observations, the critic evaluates them and provides Q-value feedback, the actor adjusts the policy to favor higher valued actions and as the actor improves and explores new states, the critic sees new experiences and adjusts the Q-value estimates according to this. The improved critic then provides better feedback to the actor and the cycle continues. The actor acts and the critic criticizes those actions, providing feedback that guides policy improvement.

Reparameterization Trick

During training, the actor network must be updated to improve its action selection. Neural networks learn by adjusting their internal weights through backpropagation, which requires computing gradients. Gradients are derivatives that measure how the loss function changes with respect to each weight parameter. These derivatives indicate the direction and rate of change. If a weight is increased by a small amount, the gradient tells how much the loss will change. By following these gradients, the weights can be adjusted to minimize the loss.

Backpropagation works by tracing backward through all operations in the network. Starting from the loss at the output, moving backwards through each layer calculating how each weight contributed to that loss. However, if one of these operations involves randomness, it can not be traced backward through it. Randomness has no gradient; it is not possible to compute how the output would change if the input is changed when the operation is random. This means that standard backpropagation fails when the network's output involves a random sampling step.

SAC solves this problem by using the reparameterization trick. Instead of the actor network directly producing random actions, it outputs two deterministic values: a mean μ_θ and a standard deviation σ_θ where θ represents the network parameters. The final action is then computed as:

$$a = \mu_\theta(s) + \sigma_\theta(s) \cdot \epsilon \tag{6.71}$$

where ϵ is the random noise sampled from a standard normal distribution. The insight here is that the randomness now comes entirely from ϵ, which is external to the network and independent of its parameters. The network itself only computes μ_θ and σ_θ which are deterministic functions. This separation is important because when computing gradients during backpropagation, ϵ is treated as a fixed constant. Now, it can be computed how changes to the network weights affect the mean and standard deviation and how they affect the action and final loss. Backpropagation can trace backwards through the deterministic path while treating the random noise as a constant. The actor network can therefore learn through standard gradient descent while maintaining the stochastic behavior that is necessary for exploration during training.

6.16.3. Replay Buffer

The replay buffer is a large memory storage that has a default value of one million past experiences. Each experience is a tuple that contains the state, action taken, reward received, and the resulting next state, and whether the episode ended. As the agent interacts with the environment during training, every transition is added to this buffer.

This buffer has two important functions. First, it allows for off-policy learning. SAC can learn from experiences collected by any policy, not just the current one. When the policy improves and behavior changes, old experiences from earlier policy versions remain useful since they still contain information about which actions led to good or bad outcomes in specific states. Second, the buffer allows that the

experience can be reused. Each experience can be sampled multiple times for weight updates rather than being used once and discarded. This significantly improves the sample efficiency.

During each weight update, 512 experiences are sampled at random from the buffer. This random sampling breaks the temporal correlations in the training data. Temporal correlation means that consecutive experiences are similar because they happened close together in time. If the network trained on experiences in the order they occurred, it would mainly see similar states in sequence and overfit these local patterns instead of learning general control principles that work across multiple situations. Random sampling solves this since each training batch contains 512 experiences from completely different episodes, transition phases, and moments in time. The sampled experiences are uncorrelated since they come from different initial conditions, different altitudes, and different distances to the target. This forces the network to learn a more general policy that translates observations into optimal actions that work everywhere. When the buffer reaches capacity, the oldest experiences are overwritten.

6.16.4. Training Scripts Architecture

Two separate scripts have been made that differ in how they collect training data. The first script creates a single HGVEnv class, where the agent interacts with one simulation at a time in sequence. The second script creates 16 parallel HGVEnv instances, each running its own complete HGV simulation on a different CPU core. The HGVEnv class is schematically shown in Appendix A.

In the parallel version, one SAC agent coordinates with all 16 environments. The parallel training architecture is illustrated in the `train_hgv.py` orchestration sequence diagram in Appendix A. The training loop works as follows: the agent's actor network receives 16 observations, one from each environment, and outputs 16 actions. These actions are sent to the 16 sub-processes, where each sub-process receives its corresponding action. The 16 environments then simulate their physics forward by 5 seconds on different CPU cores. The 16 resulting experiences are sent back to the main process and added to the single shared replay buffer. First, the SAC agent samples 512 experiences from the replay buffer. Second, the losses are computed. For the critics, it is calculated how wrong their Q-value predictions were compared to the targets, and for the actor, it is calculated how well it is selecting high-value actions based on the critic's feedback. Third, the gradients are computed using backpropagation, where it is determined how each network weight should be changed to reduce the loss. Fourth, the weights are updated by adjusting all parameters in the actor and critic networks based on these gradients. This is the moment when the networks learn; the weights and biases are improved based on the experiences.

The advantage of parallel environments is the speed of data collection. While the neural networks that run on GPU wait for simulation results, all 16 physics simulations run in parallel instead of sequentially.

The scripts configure the environment with train mode set to True, which randomizes initial conditions for each episode. When an episode ends and the environment resets, the latitude, heading, and target index are randomly chosen. A detailed view of a single training episode is shown in the `train_hgv` episode sequence diagram in Appendix A.

6.16.5. Neural Network Architecture

The actor and critic networks use multilayer perceptron architecture, which means they are fully connected neural networks where every neuron in one layer connects to every neuron in the next layer. The network has three hidden layers with 1024, 512 and 256 neurons.

The input observation vector (5 values for footprint, 7 values for no-fly zones included) enters the first layer with 1024 neurons. Each neuron in this layer computes a weighted sum of the five inputs, adds a bias term, and applies an activation function. With 1024 neurons, this first layer can learn to detect 1024 different patterns in the input data. For example, one neuron can activate strongly when altitude is high and velocity is high, which indicates an early phase in the trajectory, while another neuron might activate when altitude is low and distance to target is small, where it is close to the final approach of the trajectory.

The output of these 1024 neurons become the input to the second layer with 512 neurons. Each of these 512 neurons combines information from all 1024 first-layer neurons, which creates higher-level subjects. The second layer can combine multiple first layer patterns, such as combining high-altitude and far-from-

target signals into a more abstract early trajectory concept. The third layer with 256 neurons continues this process before passing it to the output layer.

Finally, the output layer produces the network result. For the actor network this is 4 values: the mean and standard deviation for angle of attack, and mean and standard deviation for bank angle. During training, actions are sampled from these distributions to enable exploration, but during evaluation the mean values are used directly as deterministic steering commands. For the critic networks, this is a single Q-value. The total number of trainable parameters is approximately 660,000. This large parameter count allows the network to learn nonlinear relationships in HGV control.

Between each layer, a ReLU (Rectified Linear Unit) activation function is applied. Activation functions are used to introduce non-linearity into the neural network model, enabling the network to learn complex patterns beyond linear relationships. ReLU is defined as $\max(0, x)$, which means it passes positive values through unchanged but sets all negative values to zero. Activation functions allow neural networks to learn non-linear relationships, where the output does not vary in a simple straight line fashion with the inputs, allowing the network to capture complex patterns [88]. Each neuron computes a weighted sum of the inputs, multiplying each input by a weight, and adding them together. This weighted sum can be positive or negative, depending on the input values and the learned weights. ReLU acts as a filter; if the weighted sum is positive, the neuron activates and passes the value to the next layer. If negative, the neuron remains silent and outputs zero. Different neurons respond to different input patterns, with some neurons activating for high-altitude conditions and others activating for low-altitude conditions for example.

By combining selective activations where some neurons are on and others off across many neurons and layers, the network can approximate the complex curved relationship instead of being limited to a straight line.

ReLU can also be used to avoid a problem in training deep networks called vanishing gradients. Traditional activation functions, such as sigmoid and tanh, have the problem of flattening out at extreme input values. A sigmoid function outputs values between 0 and 1, but for large positive inputs it outputs values close to 1, and for large negative inputs it outputs values close to 0. In these regions, the function becomes nearly flat, which means that small changes in the input produce almost no changes in the output. Since the gradient measures how much the output changes when the input changes, a flat function has a gradient near zero. When these near-zero gradients are multiplied together during backpropagation through many layers, they become vanishingly small, making it difficult to train deep networks. ReLU avoids this problem because for positive inputs the output equals the input, so the gradient never vanishes.

During training, the network must determine how to adjust each of the weights to reduce loss. For each weight, the question is whether the weight should be increased or decreased and by how much. This is what a gradient answers. It measures the rate of change of the loss with respect to the weight. The magnitude of the gradient indicates how sensitive the loss is to that weight. Large gradients mean that the weight has a strong effect on the loss and should be adjusted significantly.

The challenge is that the weights in the early layers are buried deep in the network. To find the total effect of an early layer weight on the final loss, backpropagation is used which multiplies the individual effects together. At each layer during this backward pass, the gradient signal must pass through the derivative of the activation function. ReLU's derivative is exactly 1 for positive inputs when neurons are active and 0 for negative inputs when neurons were inactive, meaning that gradients flowing through active neurons maintain their magnitude. This allows gradients to reach early layers.

The neural networks run on GPU instead of the CPU. The GPU contains thousands of small processing cores designed for parallel computation. A forward pass through the network requires millions of multiply and add operations, where inputs are multiplied by weights and added together. On a CPU, these operations would mostly execute sequentially. On a GPU, thousands of operations execute simultaneously in parallel.

This GPU acceleration applies only to neural network computations. The actual HGV physics simulation that performs the trajectory propagation runs on CPU cores. Each environment integrates the equations of motion forward in time, simulating the vehicle's trajectory. This ODE integration process is inherently

sequential, since each timestep depends on the result of the previous time step. Therefore, the CPU handles the physics rollouts, while the GPU handles all weight-related calculations for neural networks.

6.16.6. Hyperparameters

Several key parameters control the learning process. The SAC hyperparameters are given below:

```
model = SAC(
    'MlpPolicy',
    train_env,
    policy_kwargs=dict(
        net_arch=[1024, 512, 256],
        activation_fn=nn.ReLU
    ),
    device='cuda',
    gamma=0.99,
    train_freq=1,
    gradient_steps=1,
    batch_size=512,
    ent_coef='auto',
    verbose=1
)
```

`train_freq = 1` means the policy updates after every environment step. With 16 parallel environments, each step collects 16 experiences and then performs one weight update. The number of parallel environments is configured separately during environment initialization. The choice of `train_freq = 1` allows the agent to learn immediately from new experiences instead of waiting to accumulate larger batches.

`gradient_steps = 1` means one weight update step per update. During each weight update, the algorithm samples one batch of 512 experiences, computes the losses and the gradients and updates the network weights once. This can also be increased, such that it performs multiple separate weight updates.

`batch_size = 512` means each weight update samples 512 experiences from the replay buffer. The batch size affects the gradient quality and the computational efficiency. Smaller batches provide faster updates but noisier gradient estimates. Larger batch sizes give very stable gradients by averaging over many experiences but would require more computations per update and slow down learning. The batch size of 512 is a balance. It is large enough to average out noise and produce reliable gradients, but small enough to update efficiently.

`ent_coef = auto` allows automatic adjustment of the entropy coefficient α which controls the exploration exploitation trade-off. SAC does not use a fixed entropy coefficient. Instead, α is learned during training so that the policy maintains a specified target entropy. The algorithm increases α when the policy becomes too deterministic and decreases α when the policy becomes too stochastic. This ensures that the level of exploration is automatically regulated throughout training.

`gamma=0.99` is the discount factor that determines how future rewards are weighted relative to the immediate rewards. A discount factor of 0.99 means that a reward received one time step in the future is worth 99% as much as the same reward received immediately. A reward two time steps away is worth 0.99^2 etc. This creates a slight preference for achieving rewards sooner rather than later. This is especially important when flying towards target points, where the trajectory in the smallest time is preferable.

6.16.7. Hyperparameter Interactions

The hyperparameters `train_freq`, `gradient_steps`, and `batch_size` interact to control the total amount of learning for a fixed number of timesteps. The relationship can be understood through the total number of weight updates that occur during training.

For a training run of N total timesteps with 16 parallel environments, the number of weight updates is determined by `train_freq`. With `train_freq` = 1, there are N weight updates, one after each timestep.

With `train_freq` = 4, there would be N/4 weight updates. Each weight update can then perform multiple gradient steps. The total number of weight updates equals (N / `train_freq`) $\times$ `gradient_steps`.

This means `train_freq` and `gradient_steps` have inverse effects on learning intensity. Doubling `train_freq` (updating half as often) can be compensated by doubling `gradient_steps` (doing twice as many weight updates when there is an update), resulting in the same total number of weight updates. However, these configurations learn differently. A lower `train_freq` with lower `gradient_steps` spreads learning more evenly throughout training, while higher `train_freq` with higher `gradient_steps` performs more intensive learning at each update.

The `batch_size` affects how much data each gradient step uses. Each weight update samples `batch_size` experiences from the replay buffer to compute the gradient. Larger batch sizes provide more stable gradients because they average over more experiences, but each gradient computation takes longer. Smaller batch sizes give noisier gradient estimates but allow faster computation per gradient step. The `batch_size` doesn't change how many weight updates occur, but it changes the quality and computational cost of each update.

The number of parallel environments affects the rate of data collection. With 16 parallel environments and `train_freq` = 1, each environment step collects 16 experiences simultaneously before performing a weight update. Increasing the number of environments accelerates training by collecting more diverse experiences in parallel, filling the replay buffer faster.

Together, these parameters determine the learning profile: `train_freq` controls the update frequency, `gradient_steps` controls the learning intensity per update, `batch_size` controls the stability and computational cost of each weight update, and the number of parallel environments controls data collection speed.

6.16.8. Batch Size Selection

Different batch sizes were tested to determine the optimal configuration. A smaller batch size of 256 resulted in lower performance, with the learned policy achieving smaller maximum distances compared to the 512 batch size. The noisier gradient estimates from fewer samples per batch appeared to degrade the policy's effectiveness.

A larger batch size of 1024 produced similar results to the chosen batch size of 512 in terms of footprint area and maximum distance achieved. However, the larger batch size required approximately 1.5 times as long to complete training due to the increased computational cost of processing twice as many experiences per gradient step. Since the performance gains were negligible while the training time increased significantly, the larger batch size was not justified.

The effectiveness of a larger batch size is also dependent on the number of environments. With a larger number of environments, the replay buffer fills up more quickly and more diverse, making large batch sizes more effective.

The batch size of 512 provided the best balance between gradient stability and computational efficiency for this problem. It is large enough to average out noise and produce reliable gradients for learning effective control polices, but small enough to maintain reasonable training times.

6.16.9. Network Architecture Design Considerations

The choice for three hidden layers with progressively decreasing width (1024, 512, 256 neurons) provides enough complexity to learn the control policy while keeping training computationally manageable. This architecture design involves two key dimensions: network width and network depth.

Network width refers to the number of neurons per layer. Wider layers can learn relationships between multiple input variables simultaneously. The first layer with 1024 neurons can identify many different combinations of the input observations. If the layer was narrower, for example 512 neurons, it would have less capacity to represent all the different flight conditions that can occur. However, making layers too wide increases the computational cost. Each neuron connects to all neurons in the next layer, so doubling a layer's width doubles the number of computations required. Excessively wide networks also risk overfitting, where the network memorizes the training data instead of learning the general control strategy [89]. Network depth refers to the number of layers. Deeper networks enable multi-step

reasoning by learning causal chains. For HGV control, the network must understand that changing bank angle affects heading, which changes the angular difference between the current heading and the bearing to the target, and this has to be done in an energy efficient manner. Deeper networks can learn more complex representations. The first layer detects basic patterns such as high altitude combined with high velocity. The second layer combines these patterns into higher level concepts, such as recognizing the early phase of a trajectory. The third layer can recognize complete control regimes. However, adding more layers has costs. Each additional layer makes training slower because backpropagation must compute gradients through more steps.

Several network architectures were tested during development. Smaller networks with two hidden layers of 512 and 256 neurons showed worse performance, likely due to insufficient capacity to represent the complex state-action mapping. Larger networks were also tested, including a four-layer architecture with an additional 2048 neurons first layer. These larger networks did not improve results compared to the chosen architecture, while requiring approximately four times as many computations per training iteration. The current three-layer configuration with 1024, 512, and 256 neurons provided the best balance between performance and training time.

6.16.10. Evaluation and Early Stopping

Evaluation occurs every 10,000 time steps to assess whether the policy is still making improvements or if learning can be stopped. The EvalCallback manages this process by running five complete episodes in a separate evaluation environment that is not used for training data collection.

During evaluation, the policy uses deterministic action selection, meaning it always selects the mean of the action distribution instead of sampling from it. During training, the policy is stochastic, it samples actions from the probability distribution to encourage exploration. Deterministic evaluation removes this randomness. Given the same initial conditions, the policy will always produce the same actions.

After five evaluation episodes are complete, their total rewards are averaged to produce a mean evaluation reward. If the current evaluation achieves a new best score, the current policy network weights are saved to a model in a directory. This makes sure that even if training later makes the policy worse, which can happen due to continued exploration, the best performing version encountered during the entire training is also preserved.

The StopTrainingOnNoModelImprovement callback implements early stopping to terminate training when the policy has converged and is no longer improving. This sets a counter tracking consecutive evaluations without improvement. Each time an evaluation occurs and the mean reward sets a new highest reward, the counter resets to zero. When the counter reaches 20 consecutive evaluations without improvement, a termination signal is given.

The training loop runs until either the maximum time steps is reached of 10 million or early stopping triggers due to lack of improvement. When the training is finished, two models are saved: the final policy at the end of training that may have slightly degraded, and the model that contains the policy weights that achieved the highest evaluation score during the entire training run.

6.17. Multiple HGV Type Entries

The simulation framework has been designed so that it can be extended to multiple vehicles. This allows to analyze and compare different HGV designs, which will result in a different optimal control strategy. The framework currently supports two representative vehicle types, the HTV-2 and the DF-ZF, each with different characteristics and operational profiles.

6.17.1. Configuration Architecture

The multiple vehicle architecture is implemented such that vehicle-specific parameters are separated from the core simulation. Each type of HGV is defined by three components that completely specify the physical and aerodynamic properties.

Aerodynamic Data Table

A CSV file containing the vehicle's aerodynamic coefficients (C_L, C_D and stagnation point temperature) mapped across a three-dimensional parameter space of altitude, Mach number and angle of attack is

provided. These tables can be generated by any computational program and put in the code as a CSV file.

Vehicle Constants File
A JSON file is included that specifies the vehicle's physical properties, including the mass, surface area, and operational constraints. The constraints include maximum g-load, maximum dynamic pressure, and maximum stagnation point temperature. These parameters define the vehicle's structural and thermal limits that must be respected at any time during the trajectory optimization. This can differ for any type of HGV.

Trained Model File
Finally, a ZIP archive containing the trained SAC neural network weights that encode the optimal control policy for that specific HGV. Each model is trained independently using the vehicle's aerodynamic characteristics and constraints, which results in control policies that are adapted to the vehicle's flight dynamics.

This separation of configuration from code has several advantages. First, adding a new vehicle type requires no changes to the core simulation code, only the three configuration files have to be provided. Second, the same training and evaluation scripts can be used for all vehicle types. Finally, quick changes can be made to vehicle parameters without having to make changes to the code.

6.17.2. HTV-2 Configuration
The HTV-2 represents the high-speed end of the HGV design spectrum and is an HGV developed by DARPA [90]. The default initial conditions are:

- Altitude 100km
- Velocity 6000 m/s
- Flight path angle -3°

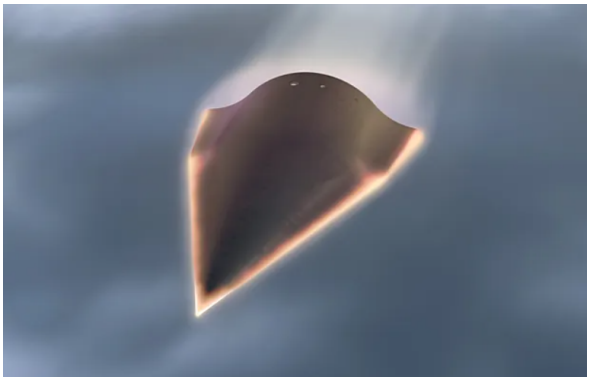
Due to the high velocity, it can be seen that this HGV contains a lot of energy at release and has a lot of potential to distribute this energy along the trajectory. The mass of the HTV-2 is 2183.6 kg.

6.17.3. DF-ZF Configuration
The DF-ZF HGV represents a different point in the HGV design, with a lower initial velocity, this vehicle will have a different operational envelope. The mass of the DF-ZF is 2292.4 kg. The initial conditions are as follows:

- Altitude 100 km
- Velocity 3450 m/s
- Flight path angle 0°

The lower velocity shows that this vehicle is optimized for a different mission than HTV-2. The trained model for DF-ZF has learned control policies adapted to these different characteristics. For example, the policy may use a different angle of attack during descent and use different bank angle strategies.

Artist impressions of the HTV-2 and DF-ZF are shown in Figure 6.14.



(a) HTV-2 Hypersonic Glide Vehicle [91]

(b) DF-ZF Hypersonic Glide Vehicle [92]

Figure 6.14: Artist impressions of the HTV-2 and DF-ZF Hypersonic Glide Vehicles.

6.18. Single Trajectory Execution

Once the training process produced a converged SAC model, the performance was validated. A script was made that is a tool that runs a full HGV simulation under closed-loop control of the trained neural policy. It shows how the learned control behavior translates to an actual flight trajectory, including energy management and adherence to the constraints.

It should be noted that the model is trained in such a way that it works anywhere on Earth. The user can select in the program the start latitude and longitude and choose the target location it should fly towards, and a single trajectory will be made which the agent has trained.

At the beginning of the execution, the user can select the trained model. This could be for any of the trained HGV models. The script loads this model and initializes a simulation environment with the same aerodynamic and physical configuration used during training. Here, the initial conditions can be specified, including the start latitude, longitude, heading, velocity, altitude, and flight path angle. The environment contains the HGV model, which integrates the state vector using the equations of motion. Within each integration step, the trained agent receives the observation vector. Based on this observation, the policy outputs the two control values. These actions are applied to the HGV that determines the trajectory for the next integration step. The target index parameter determines which of the 36 azimuthal directions the trajectory will pursue. By varying this index parameter, it can be examined how the learned policy adapts the control strategy for different target bearings.

The simulation loop continues until the vehicle reaches the terminal condition. At each iteration, the environment checks all constraints. If any constraint is positive, the violation is displayed, which can be used to check whether any of the trajectories has violated the constraints.

This process operates identically to the learning loop used during training. The environment is stepped forward in time, new observations are generated and the agent produces an action at every step. The only difference is that during training these actions were used to update the neural network weights through gradient-based optimization to improve the policy over time. When running a trajectory, the learned weights are fixed. The neural network no longer learns or adapts, instead it applies the control policy. Thus, the same perception to action cycle is preserved but the model now performs deterministic decision making based on the trained knowledge rather than exploration or learning.

After completion, the script outputs the flight duration, great circle distance traveled, and the residual distance to the target. For the unreachable circle, this will never be close to zero, since the agent is trained to fly as close as possible to this target that is out of reach.

6.18.1. Single Trajectory Results and Control behavior Analysis for Maximum Range

In Figure 6.15 a single trajectory is shown targeting a point on the equator that lies beyond the HGV's maximum reachable range, together with the angle of attack and bank angle control chosen by the agent. This unreachable target ensures that the policy maximizes range in the specified direction without prematurely terminating its optimization when the target is reached.

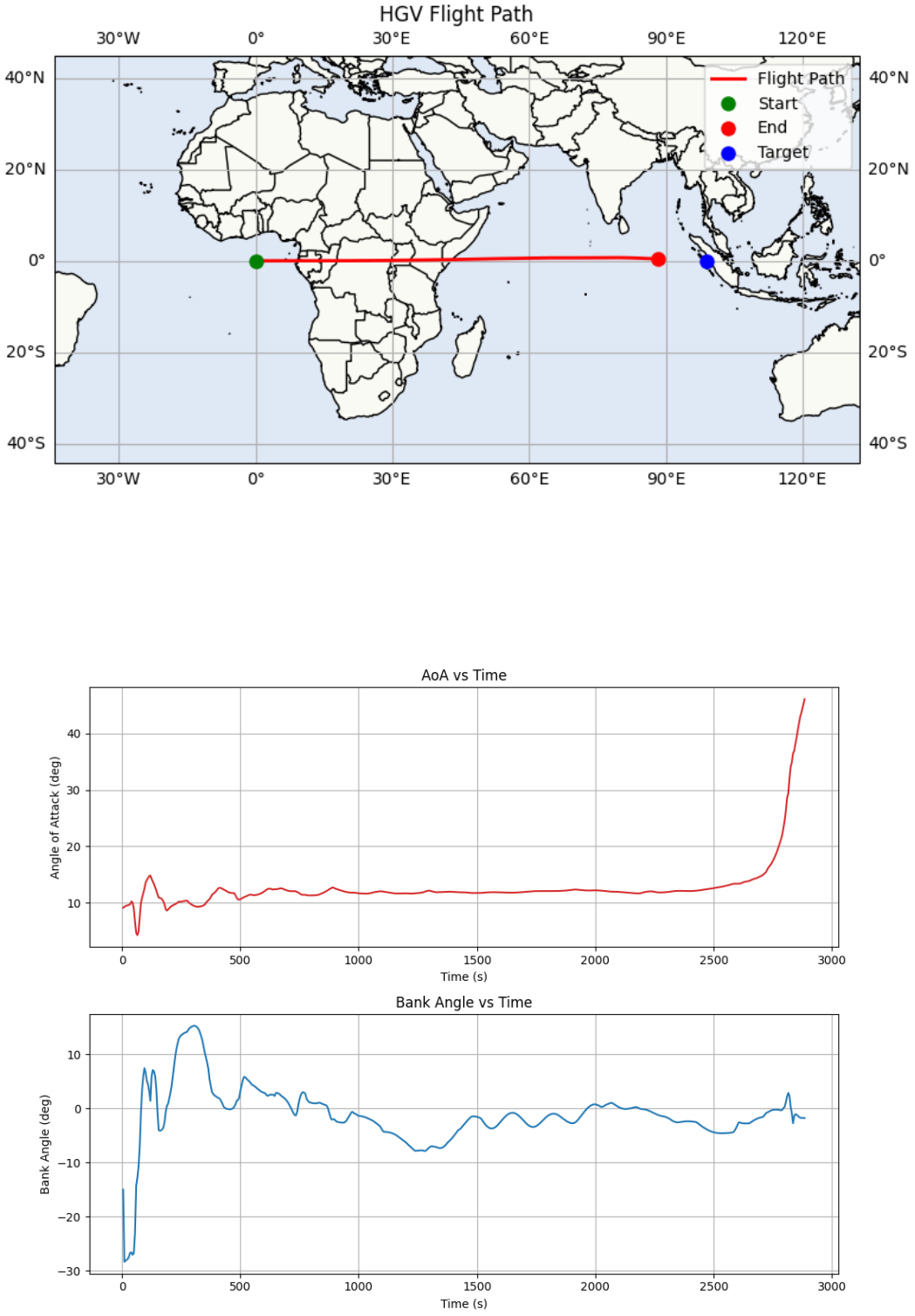


Figure 6.15: Ground trajectory and control variable inputs for maximum range for the HTV-2 HGV.

Angle of Attack Strategy Maximum Range

The angle of attack shows that the learned policy immediately converges to approximately 12 degrees and maintains this value throughout most of the flight. This is a significant result because 12 degrees corresponds to the maximum lift to drag ratio of the HTV-2 vehicle configuration. The policy discovered this optimal operating point purely through reinforcement learning, without being provided with aerodynamic efficiency information.

Flying at maximum L/D minimizes the rate of energy dissipation per unit distance traveled, which will result in great range in unpowered gliding flight. The fact that the policy learned this strategy validates the reward function formulation based on distance to target reduction. By maximizing progress toward an unreachable target, the agent naturally discovered the control law that maximizes range.

The increase in angle of attack that can be seen in the final phase represents a flare maneuver as the vehicle approaches the terminal altitude threshold. This flare converts the remaining kinetic energy into additional horizontal range rather than allowing that energy to be wasted in the final descent. By pulling up sharply, the HGV trades remaining velocity for extra distance traveled before crossing the terminal altitude threshold. This behavior demonstrates energy management. The policy learned that the optimal strategy is not to maintain the cruise angle of attack until the end, but to perform a final range-extending maneuver that extracts maximum range from the residual kinetic energy.

According to Mooij et al. for maximum range glide: ”Toward the conclusion of the flight, enhancing the range can be achieved by executing a final flare, during which kinetic energy is converted into potential energy until reaching stall speed” [38]. Interestingly, in this research a similar behavior was found autonomously. The reinforcement learning agent learned to execute such a terminal flare maneuver on its own.

However, it must be noted that the increase in range of this flare maneuver is minimal, and the reinforcement learning agent will not always perform a flare at the end of the trajectory, since the energy management can also be performed in other stages of the flight and the gain in range is very minimal compared to the total distance flown.

Bank Angle Behavior Maximum Range

The initial bank angle fluctuations reflect the HGV’s flight through the upper atmosphere, where air density is extremely low. At these altitudes, the aerodynamic forces are negligible regardless of the deflections of the control surface. The bank angle commands given by the policy during this phase have essentially no effect on the vehicle’s motion since there is insufficient dynamic pressure to generate lateral forces.

As the vehicle descends into a denser atmosphere, the dynamic pressure increases to levels where the bank angle commands begin to produce aerodynamic forces. The policy responds by changing the bank angle toward zero. This approximately zero-bank attitude is maintained throughout the trajectory. Since the target is on a straight line from the starting position, a pure longitudinal flight will result in maximum range. Any banking redirects lift horizontally, reducing the vertical component and decreasing range. The policy learned to avoid energy wasting lateral movements for targets aligned with the initial trajectory direction.

The nearly straight ground track at the top of Figure 6.15 confirms that minimal lateral maneuvering occurred during flight. The trajectory follows an approximately great-circle path, which is the shortest distance between two points on a sphere.

Altitude versus Speed Profile Maximum Range

The altitude versus speed profile for the maximum range trajectory is shown in Figure 6.16. This shows the characteristic skipping behavior of the HGV for large-range flights. The vehicle alternates between climbing and descending phases in an oscillatory manner as it progresses through the atmosphere.

It is important to note that while skipping behavior could be suppressed through active damping using bank angle and angle of attack modulation, doing so would be counterproductive for range maximization. Banking the vehicle reduces the vertical component of the lift, which would prevent the oscillations. However, this reduction in lift increases energy dissipation, which results in shorter range. The learned

policy therefore permits and exploits the natural skipping dynamics rather than attempting to fully damp them.

The amplitude of these oscillations gradually decreases as the vehicle loses energy, eventually transitioning into a final glide phase at lower altitudes.

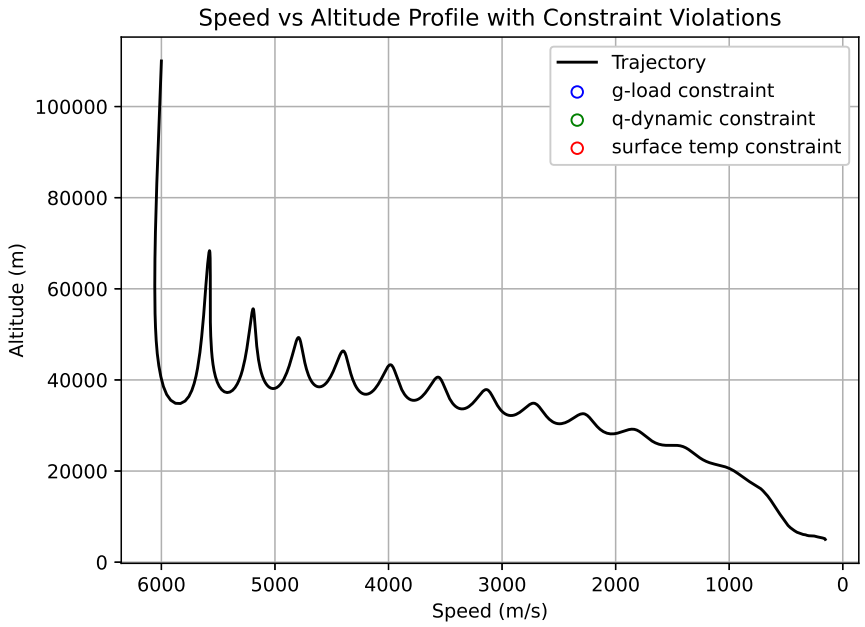


Figure 6.16: Speed versus altitude profile for HTV-2 maximum range trajectory.

6.18.2. Single Trajectory Results and Control behavior Analysis for Minimum Range

The minimum range trajectory gives a different scenario where the target point lies on the latitude line behind the vehicle's initial position, which requires a reversal of direction while still maximizing range in the opposite direction of the launch location. This presents one of the most challenging control problems the policy must solve. It must turn the vehicle 180 degrees while managing energy and making sure no constraints are violated. The trajectory and the bank angle and angle of attack control is shown in Figure 6.17.

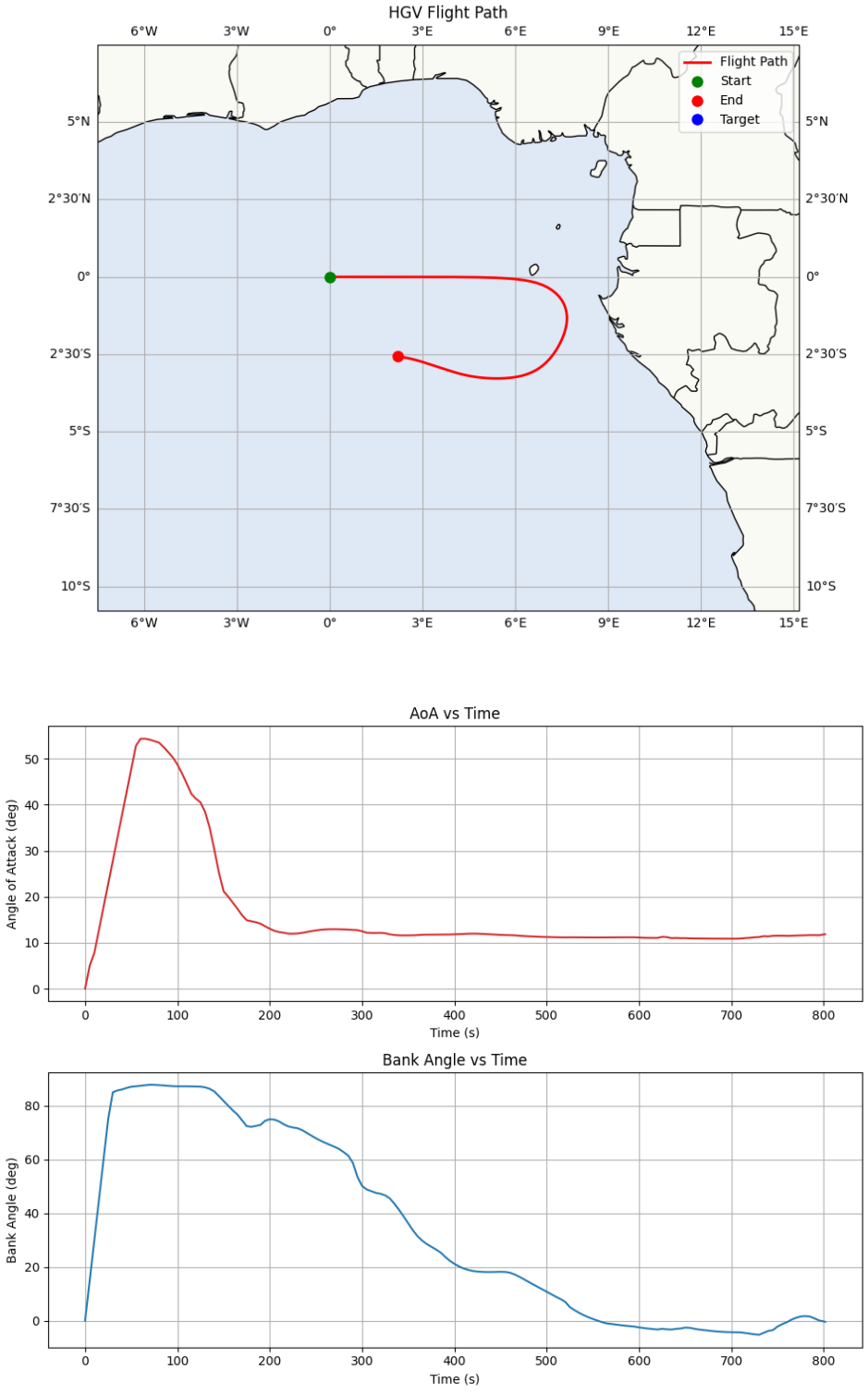


Figure 6.17: Ground trajectory and control variable inputs for minimum range for the HTV-2 HGV.

Angle of Attack Strategy Minimum Range

The angle of attack profile for the minimum range looks significantly different from the maximum range trajectory. Initially, the policy commands an increase to the maximum allowable angle of attack. This

applies the maximum angle of attack rate limit of $1^\circ/\text{s}$. This high angle of attack is necessary for managing the temperature constraint during the turning phase. When the HGV banks steeply to execute the directional reversal, the total lift vector is tilted away from vertical. With this low lift, the descent rate is high.

To increase the range in the reversed direction, it is important for the HGV to perform the direction reversal as quick as possible. The angle of attack is increased to increase the turn rate of the HGV. Since this is a 3-DOF point mass model and the angle of attack is defined in the body frame, increasing the angle of attack when it is performing a high bank angle maneuver will increase the turn rate. This increases the load factor experienced, but this constraint is also monitored in real time and the agent will have received a penalty if this was violated and changed the control action according to this.

After this initial spike, the angle of attack decreases back to around 12 degrees, the maximum lift-to-drag ratio angle of attack. This convergence happens as at this point in the trajectory, the largest deceleration has occurred, and the range has to be increased again now that the heading reversal has happened.

Bank Angle Strategy Minimum Range

The bank angle profile shows the turning strategy that the policy learned for this more complex maneuver. Immediately, the policy commands an aggressive bank maneuver with the maximum bank rate change of $3^\circ/\text{s}$ and sustains high bank angles for a period of time. This sustained extreme banking is done to achieve the heading reversal as quickly as possible, whereafter the range can be maximized in the opposite direction. It can be seen that after this high bank angle, the bank angle is gradually reduced until the desired heading is achieved, where the bank angle goes to zero.

According to Mooij et al., maximum lateral ranges are obtained by applying large bank angles early in the trajectory, which is followed by a reduced one once the desired heading is achieved [38]. That is what can be seen here as well. The trajectory starts off with a high bank angle, which gradually reduces over time so that it achieves the desired final heading. This is also shown in Figure 6.18, where a target point is chosen for maximum cross range. Here, it can clearly be seen that the trajectory starts with a large bank angle early in the trajectory, followed by a reduced one until the desired heading has been achieved.

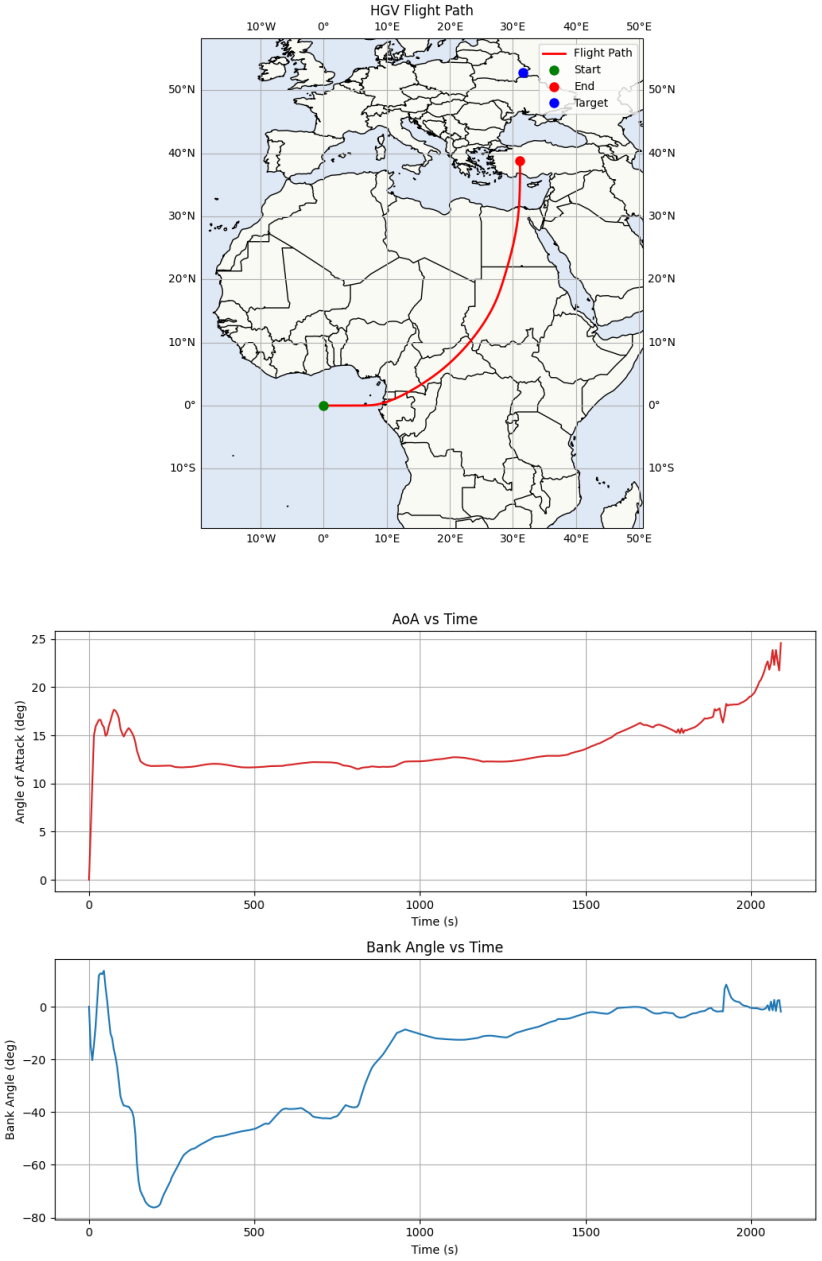


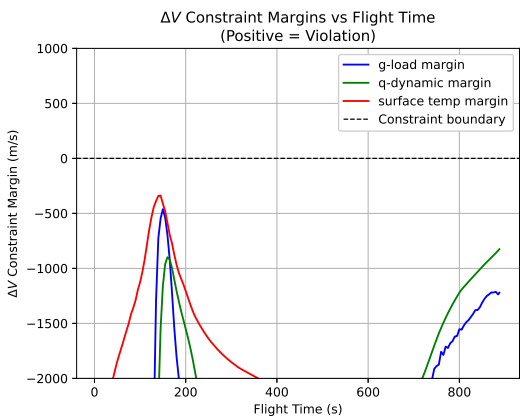
Figure 6.18: Ground trajectory and control variable inputs for maximum cross range for the HTV-2 HGV.

6.18.3. Constraint Margin and Altitude Speed Profile for Minimum Range

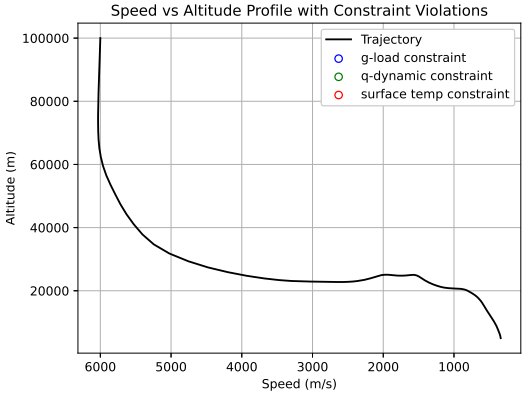
Figure 6.19a shows the velocity margin before a constraint would be violated for this trajectory. It can be seen that the agent's adaptive control strategy has resulted in the vehicle coming close to the constraints to optimize performance, but does not violate any of the constraints for this extreme maneuver.

The corresponding trajectory in the altitude speed profile is shown in Figure 6.19b. This shows different characteristics compared to the maximum range case. The trajectory displays a descent without the oscillatory skipping behavior observed in the maximum-range trajectory. This absence of skipping is intentional and shows the control strategy required for rapid directional change.

The smooth descent pattern is a consequence of the aggressive bank angle maneuver that is performed during the turn. As discussed in Section 6.18.2 the high bank angles required for directional reversal reduce the vertical component of the lift, which prevents the vehicle of skipping behavior.



(a) Velocity constraint margins versus flight time.



(b) Speed versus altitude profile for direction reversal trajectory.

Figure 6.19: Constraint margins and speed versus altitude profile for direction reversal trajectory

6.18.4. Single Trajectory Run DF-ZF HGV

To demonstrate the applicability of the framework to different HGV types, a single run has also been performed for the DF-ZF HGV model. After training with the same methodology, but using the DF-ZF specific aerodynamic data and vehicle constants, a maximum range trajectory was executed. The trajectory and control inputs are shown in Figure 6.20

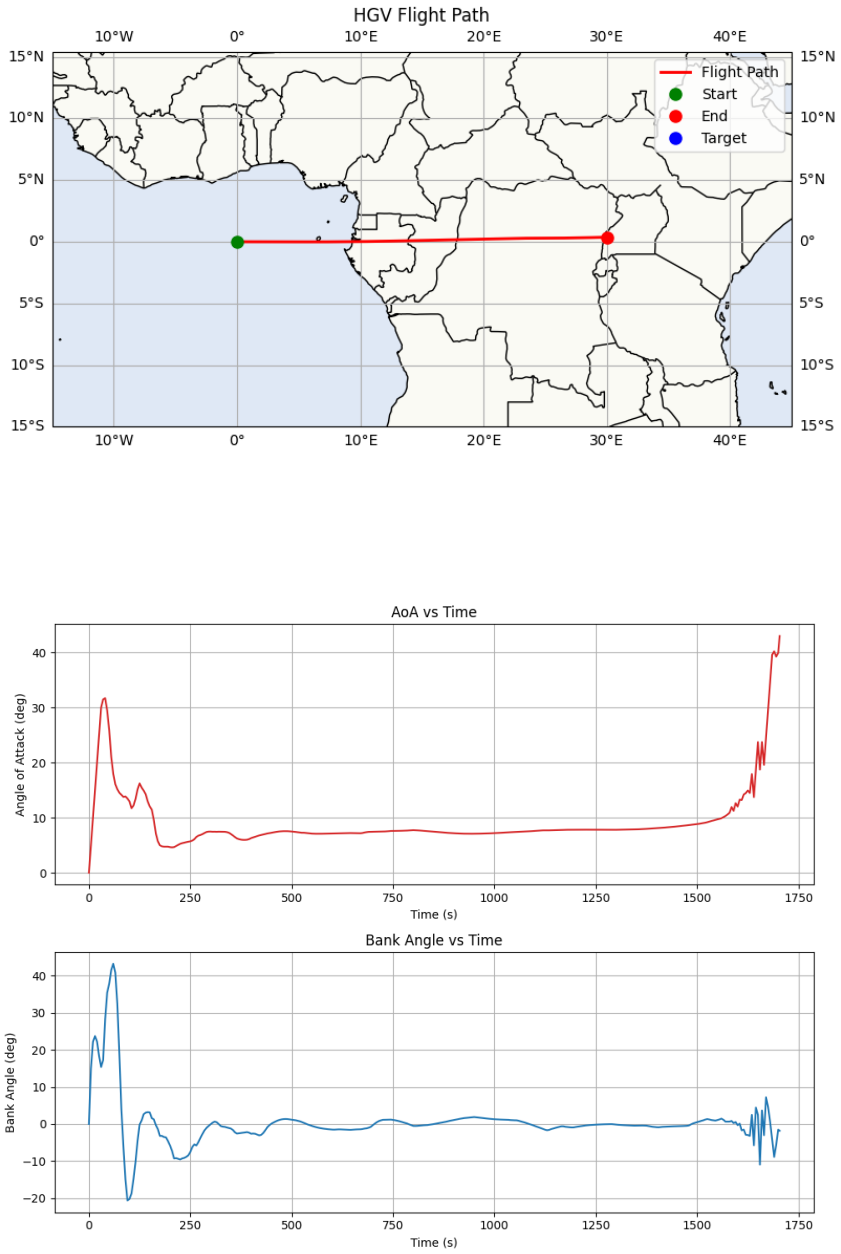


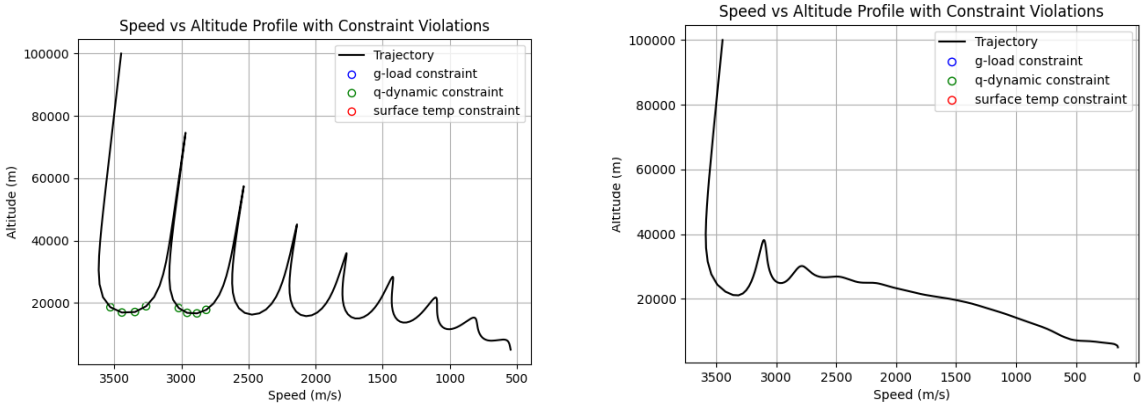
Figure 6.20: Ground trajectory and control variable inputs for maximum range for the DF-ZF HGV.

Angle of Attack Behavior DF-ZF

Unlike the HTV-2 which converged to 12° angle of attack, the DF-ZF trajectory converges to approximately 8 degrees. This value corresponds to the maximum lift-to-drag ratio for the DF-ZF configuration. The fact that the trained policy identifies the optimal angle of attack for this fundamentally different vehicle validates that the reinforcement learning framework successfully learns vehicle-specific optimal control strategies.

However, the initial phase shows a deviation from this optimal cruise value. At the beginning of the trajectory, the angle of attack increases sharply before settling to the $8°$ cruise condition. This initial increase represents a crucial constraint avoidance maneuver rather than inefficient control.

The necessity of this maneuver becomes evident from the constraint violation analysis shown in Figure 6.21a. If the vehicle maintained a constant $8°$ from the start, violations of dynamic pressure constraints would occur during the initial descent through the denser atmosphere.



(a) Speed versus altitude profile showing dynamic pressure constraint violations with constant maximum L/D AoA for DF-ZF vehicle configuration.

(b) Speed versus altitude profile showing the trajectory that the agent learned, without constraint violations for the DF-ZF vehicle configuration.

Figure 6.21: Speed versus altitude profile for constant $8°$ AoA on the left and the RL agent's learned policy on the right.

The learned policy prevents this violation by temporarily increasing the angle of attack during the high dynamic pressure phase. This control action generates a higher drag that decelerates the vehicle more rapidly and reduces the dynamic pressure experienced. As the vehicle descends through the peak dynamic pressure region and enters the less critical flight conditions, the policy immediately reduces the angle of attack back to the optimal lift-to-drag angle of attack of $8°$.

This adaptive constraint management was not explicitly programmed in the policy. Instead, it naturally came from the DF-ZF training process through the formulation of constraint penalties in the reward function. The agent learned to recognize the combination of parameters that lead to violations of the constraints and developed a control response that keeps the trajectory feasible while minimizing the deviation from the optimal cruise conditions.

The effectiveness of this learned strategy is demonstrated in Figure 6.21b. Here, the altitude versus speed profile for the agent-generated trajectory is shown. In contrast to the constant angle of attack case for maximum L/D shown in Figure 6.21a, the adaptive control strategy successfully maintains all constraint margins throughout the flight, with no dynamic pressure violations occurring.

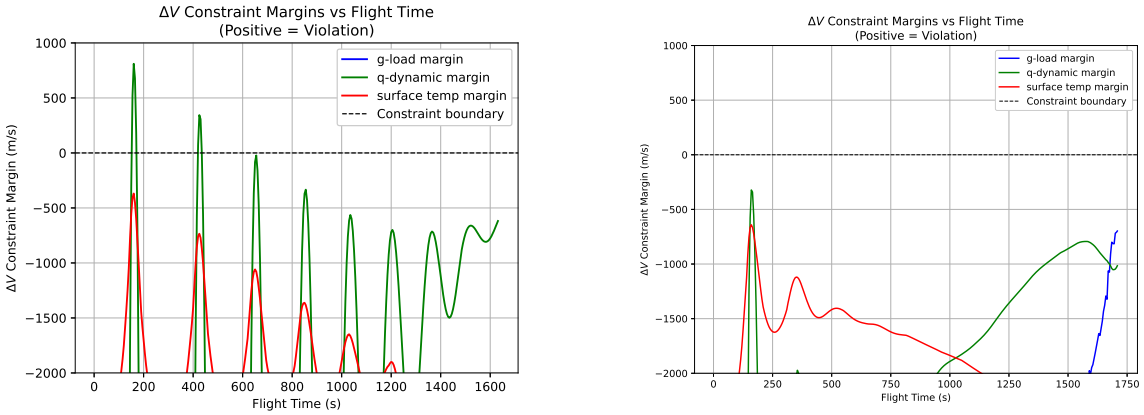
Bank Angle Behavior DF-ZF

The bank angle profile for the DF-ZF gives the same behavior pattern as was observed for the HTV-2. Initial fluctuations occur during the upper atmospheric phase, where aerodynamic forces are negligible. As the vehicle descends into denser atmosphere and control authority becomes effective, the policy converges the bank angle toward zero and maintains this near-zero attitude throughout the trajectory.

ΔV Constraint Margins

The necessity of the angle of attack maneuver becomes evident from the constraint violation analysis. Figure 6.22a shows the velocity constraint margins versus flight time for a trajectory with constant $8°$ angle of attack, where positive values indicate constraint violation. Figure 6.22b shows the velocity constraint margins for the adaptive trajectory of the RL agent. As explained in Section 6.15.8, the

constraint violation is measured in the velocity difference towards the velocity where a violation would occur.



(a) Velocity constraint margins versus flight time for constant $8°$ AoA.

(b) Velocity constraint margins versus flight time for the RL agent's adapted trajectory.

Figure 6.22: Velocity constraint margins versus flight time for constant AoA and RL agent's adapted trajectory

For the constant $8°$ AoA trajectory, violations of the dynamic pressure occur, where the margins are shown in Figure 6.22a and the margins for the altitude speed profile for this trajectory in Figure 6.21a. These violations again demonstrate that flying at constant maximum L/D ratio from the beginning is infeasible for the DF-ZF.

The learned policy prevents these violations by temporarily increasing the angle of attack during the high dynamic pressure phase shown in Figure 6.20. This control action generates higher drag, which decelerates the vehicle more rapidly and reduces the dynamic pressure experienced. As the vehicle descends through the peak dynamic pressure region and enters the less critical flight conditions, the policy reduces the angle of attack back to $8°$. It can be seen in Figure 6.22b that the agent does not violate any of the constraints.

6.18.5. Validation with Maximum L/D Angle of Attack

To validate the performance of the trained reinforcement learning policies, a comparison was made between the trajectories generated by the agent and a heuristic baseline: constant flight mat maximum L/D ratio with zero bank angle.

For the HTV-2, the constant angle of attack corresponds to $12°$ and for the DF-ZF $8°$, which has been checked for every altitude to be consistent. The zero bank angle guarantees that the full magnitude of lift is preserved in the vertical direction since any banking would reduce the vertical lift component and result in greater energy dissipation and reduced range in the initial heading direction. To validate the model's performance, trajectories were computed from multiple launch conditions, including different headings and start positions. The comparison is summarized in Table 6.1, which gives the final range achieved by both constant maximum L/D with zero bank strategy and the learned policy for both HGVs across different starting conditions.

Table 6.1: Range comparison between RL policy and constant max L/D angle of attack for different starting conditions

Vehicle	Start Location (lat, lon)	Heading	RL Policy Range (km)	Constant Max L/D AoA Range (km)	Difference (km)	Difference (%)
HTV-2	$0°, 0°$	$90°$	9923	10041	-118	-1.18
HTV-2	$0°, 0°$	$270°$	7948	8094	-146	-1.80
HTV-2	$0°, 0°$	$0°$	8777	8873	-96	-1.08
HTV-2	$52°, 5°$	$230°$	8357	8514	-157	-1.84
DF-ZF	$0°, 0°$	$90°$	3335	3303	$+32$	$+0.97$
DF-ZF	$0°, 0°$	$270°$	3040	3027	$+13$	$+0.43$
DF-ZF	$0°, 0°$	$0°$	3172	3151	$+21$	$+0.67$
DF-ZF	$52°, 5°$	$230°$	3127	3106	$+21$	$+0.68$

The results demonstrate that the reinforcement learning policies achieve ranges that are comparable to and in some cases even exceed the maximum L/D ratio benchmark for constant AoA. For the DF-ZF vehicle, the RL policy consistently slightly outperforms the constant maximum L/D angle of attack strategy across all tested conditions, with improvements ranging from $+0.43\%$ to $+0.97\%$. The altitude versus speed graph for $0°, 0°$ start location and $90°$ heading for the RL policy is shown in Figure 6.21b and the altitude versus speed graph of the constant $8°$ angle of attack in Figure 6.21a, even though this trajectory does not satisfy the constraints. All the red numbers in the table are trajectories that have violated one or more constraint for their maximum lift to drag constant angle of attack. For the $0°, 0°$ start location and $90°$ heading trajectory to not violate any of the constraints, it would require a constant angle of attack of $11°$ which drastically reduces the range, so for this comparison it is compared with the maximum L/D AoA even though this violates a constraint.

This superior performance is primarily due to better energy management throughout the trajectory, of which the terminal flare maneuver is a notable component. The policy learned to more efficiently manage the vehicle's energy state throughout the flight. By increasing the angle of attack at the end of the trajectory, the policy converts the remaining kinetic energy into potential energy through a terminal flare, extending the range by several tens of kilometers beyond what is achievable with a constant cruise angle of attack. This shows that the learned policy not only discovered the optimal cruise conditions, but also developed an advanced energy management strategy that extracts additional performance.

To verify this interpretation, an independent simulation was performed with the manually designed angle of attack profile shown in Figure 6.23, which introduces a flare maneuver at the end of flight. This test has been performed for start location $0°, 0°$ and heading $90°$ and resulted in a total range of 3334km, almost identical to the 3335km achieved by the RL policy.

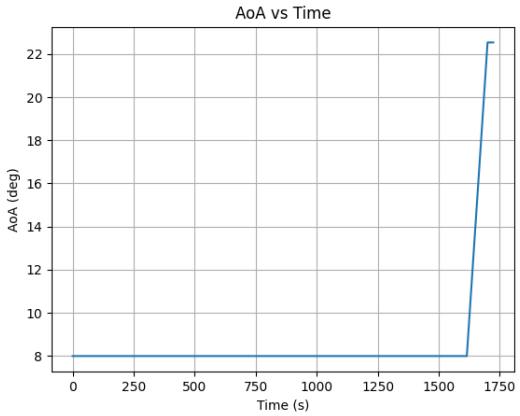


Figure 6.23: Angle of attack versus time with introduced flare maneuver.

For the HTV-2, the RL policy achieves between 98.2% and 98.9% of the theoretical maximum range across all tested conditions, which represents a great result for a RL agent. Traditional gradient-based optimization methods require extensive problem-specific tuning, accurate initial guesses and explicit analytical derivatives of the objective function and constraints. In contrast, the reinforcement learning approach developed in this work discovered the optimal control strategy purely through interaction with the simulation environment. The 1.08% to 1.84% gap relative to the theoretical benchmark is minimal and can be attributed to the neural network policy's behavior.

6.19. Footprint Generation

The footprint generation file arranges the computation of 36 trajectories, which represents flight in all azimuthal directions from a common starting point. The footprint provides a visual and quantitative measure of the operational range of the HGV. It is the final stage in the simulation and optimization framework, combining the aerodynamic model, the atmosphere, control policy, and constraint monitoring into a single operation. The resulting footprint gives the operational envelope of the vehicle, at every possible start location on Earth, without violating any of the constraints.

6.19.1. Conceptual Overview

For each footprint analysis, the vehicle starts from a fixed initial state defined by the altitude, velocity, and the flight path angle. Each target direction corresponds to a different azimuthal heading the vehicle could fly towards. The learned policy from the training phase provides the control strategy that attempts to maximize range towards each target, and the collection of all endpoints gives the boundary of the reachable region. This sampling approach changes the continuous reachability problem into a discrete set of trajectory optimizations that can be computed in parallel to improve the time efficiency of the program.

The implementation uses Python's `ProcessPoolExecutor` to distribute the 36 trajectory computations across multiple CPU cores. The parallel footprint architecture is documented in the activity and sequence diagrams in Appendix A. The footprint activity diagram shows the overall execution flow, the footprint orchestration sequence diagram illustrates how the 36 trajectory computations are distributed across multiple CPU cores during parallel execution, and the footprint simulation sequence diagram shows what occurs inside each worker process during trajectory generation. Each worker process initializes once with its own copy of a trained model and environment and then computes multiple trajectories without reloading these objects. The worker initialization loads the trained SAC model and instantiates an HGV environment with the specified initial conditions for the footprint. These objects continue throughout the worker's lifetime, which becomes available to all trajectory computations within that worker. When a trajectory computation is requested for a specific target index, the worker configures the environment to simulate flying towards that azimuthal direction and executes the policy until the terminal altitude is reached.

6.19.2. Policy Execution and Trajectory Generation

For each target direction, the trajectory is generated by applying the learned policy to each trajectory to obtain control actions. The policy receives the current observation and outputs the commanded angle of attack and bank angle. The deterministic policy evaluation mode disables the exploration noise that would be active during training. This guarantees repeatability; running the same initial conditions twice produces identical trajectories. Each of the 36 trajectories begins from the same initial state but flies to a different target point.

Again, the physics simulation advances in discrete time steps. At each time step, the equations of motion are integrated again to update the vehicle state based on the applied control inputs. Based on the observables that are returned, new control inputs are given at each step. This integration continues until the HGV descends through the terminal altitude, whereafter the trajectory is terminated and the final position is recorded.

Throughout each trajectory, the position is accumulated as a sequence of latitude and longitude pairs. This together forms the complete ground track. Constraint margins are also monitored at each step. If a violation occurs, this indicates that the policy did not correctly learn the avoidance of constraints during training, which is a verification of the satisfaction of the constraints.

The simulation loop is straightforward. The environment is reset to obtain the initial conditions, then enters a loop that calls the policy for an action, applies that action to advance the simulation, records the new position, and finally checks for termination. This continues until the done flag indicates that the terminal altitude has been reached. The resulting position list gives the complete trajectory from the initial conditions until termination.

6.19.3. Footprint Boundary

The 36 trajectories produce terminal positions that gives the footprint boundary. These endpoints define a polygon on the Earth's surface, where the interior represents the reachable region. The footprint area is computed using spherical geometry algorithms that account for the curvature of the Earth, which is essential for accuracy over the thousands of kilometers that the footprint spans. It must be noted that it is assumed that all the points that lie within this perimeter can be reached by the HGV. However, in reality some points, especially close to the starting position, cannot be reached without violating the constraints. For this, target point guidance has been made, where every point within the footprint can be flown and checked to determine whether this is feasible within the constraints. This is explained in Section 6.20.

The implementation uses the `pyproj` library's geodesic polygon area function, which applies spherical geometry to compute the area that is enclosed by a polygon on a sphere. The algorithm accounts for the convergence of longitude lines toward the poles and varying size of degree increments at different latitudes. The same library is used to compute the area of the polygon.

Distance calculations use the haversine formula, as explained in Section 6.10, to compute the great circle distances from starting position to each endpoint. The maximum and minimum distances are tracked during these calculations. These statistics quantify the longitudinal extent of the footprint, showing how far the vehicle can fly in the best direction and how far it could come when it turns around and flies back for the minimum-range case.

Cross-track analysis give the lateral spread of the footprint. The great-circle path from the starting position to the maximum range endpoint is used as the reference trajectory. For each of the other endpoints, the perpendicular distance from this reference path is computed using the cross-track distance function, explained in Section 6.4. This calculation determines the angular distance from the starting point to the end point, computes the bearing to that end point and applies spherical trigonometry to get the perpendicular component. The maximum absolute cross-track distance defines the HGV's lateral maneuverability.

6.19.4. Footprint Visualization

After all trajectories have been simulated and their terminal coordinates have been determined, the footprint is rendered on a stereographic map projection centered on the midpoint of the footprint region. This adaptive centering is essential, since footprints can be generated from any launch location on Earth. The midpoint calculation is done using spherical geometry, also explained in Section 6.10. The enclosed polygon represents the boundary of all terminal points. This projection choice is important for high-latitude footprints, since it handles polar regions without severe distortion that projections like Mercator have. For Mercator projections near the poles, cylindrical projections stretch features, which eventually become singular at 90° latitude.

The stereographic projection avoids the limitation by projecting the sphere on to a tangent plane. For a projection that is centered at a chosen point, the tangent plane touches Earth's surface at that location and points on the sphere are projected by drawing lines from the opposite side of Earth through each surface point to where they intersect the plane. By doing this, angles locally are preserved and great-circles are shown as either straight lines or circular arcs, which is ideal for trajectory visualization.

A visualization of how the stereographic projection works for a projection centered on the North Pole is shown in Figure 6.24.

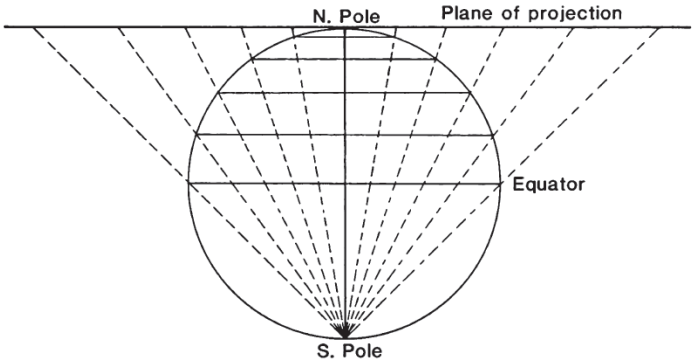


Figure 6.24: Stereographic projection centered on the North Pole [93].

Although the stereographic projection preserves local angles, it introduces scale distortions that increase with distance from the projection center. These distortions are most noticeable near the outer boundary of the footprint, where the distances appear larger than they actually are. However, this does not affect the analysis, since all physical computations are performed on the spherical Earth. The projection is only used for visualization, so any visual distortion has no influence on the underlying trajectory or footprint calculations. Although stereographic projection dramatically distorts areas, there are two reasons why this projection is better than any other [94]. Firstly, all circles on the sphere are transformed into circles on the plane. Secondly, the stereographic projection preserves angles, which makes it a conformal map.

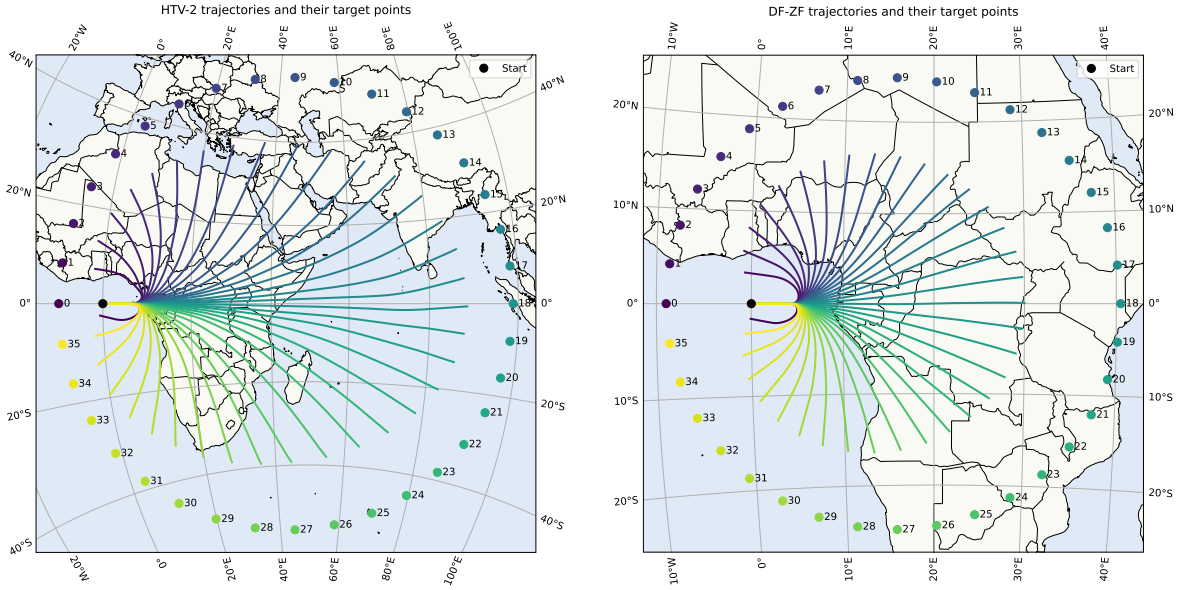
6.19.5. Target Point Construction

In order to define a clear training objective for the reinforcement learning agent, a set of unreachable target points will be generated on the Earth's surface for each starting position of the HGV. These targets serve as reference locations for which the agent is rewarded for making progress. The design of this target set follows two main requirements. First, the target points should be placed far enough away that they cannot be reached. Secondly, they should cover all azimuthal directions for the generation of a footprint.

The circle itself is defined on the spherical Earth. To construct it, a central point (ϕ_c, λ_c) is first identified. This point lies on a great-circle that extends from the initial position in the initial heading direction. The radius of the circle has been chosen so that for the maximum range achievable of the HGV it will not exceed this. From this central point, the full set of target points is then generated. To achieve this, an initial reference azimuth is taken from the center of the circle back to the starting position, and then successive targets are placed by increasing this azimuth in steps of $2\pi/36$ until the entire circle is covered. Equations 6.9 and 6.10 are used to calculate the end point from the given starting position, bearing direction and distance. By evaluating them for the following bearings, the environment generates 36 evenly spaced target points around the circle center.

$$\psi = \psi_{\text{ref}} + i \cdot \frac{2\pi}{36}, \quad i = 0, 1, \ldots, 35 \tag{6.72}$$

The construction guarantees that every azimuthal direction is represented. Figures 6.25a and 6.25b illustrate this target point distribution and the resulting trajectories. The black dot marks the starting position, with trajectories numbered 0 through 35 radiating outward in the evenly spaced azimuthal directions. The color gradient indicates the progression around the full circle. Each numbered trajectory corresponds to one of the 36 target point indices. Comparing the two figures shows that the radius of the unreachable circle is adapted to each HGV's specific range capabilities, with the HTV-2 requiring a significantly larger circle radius due to its higher initial energy state. This adaptation is critical for effective training, since placing target points at excessive distance would degrade the reinforcement learning performance by making it difficult for the agent to distinguish between different radial directions, since the bearing differences become increasingly subtle as target distance increases.



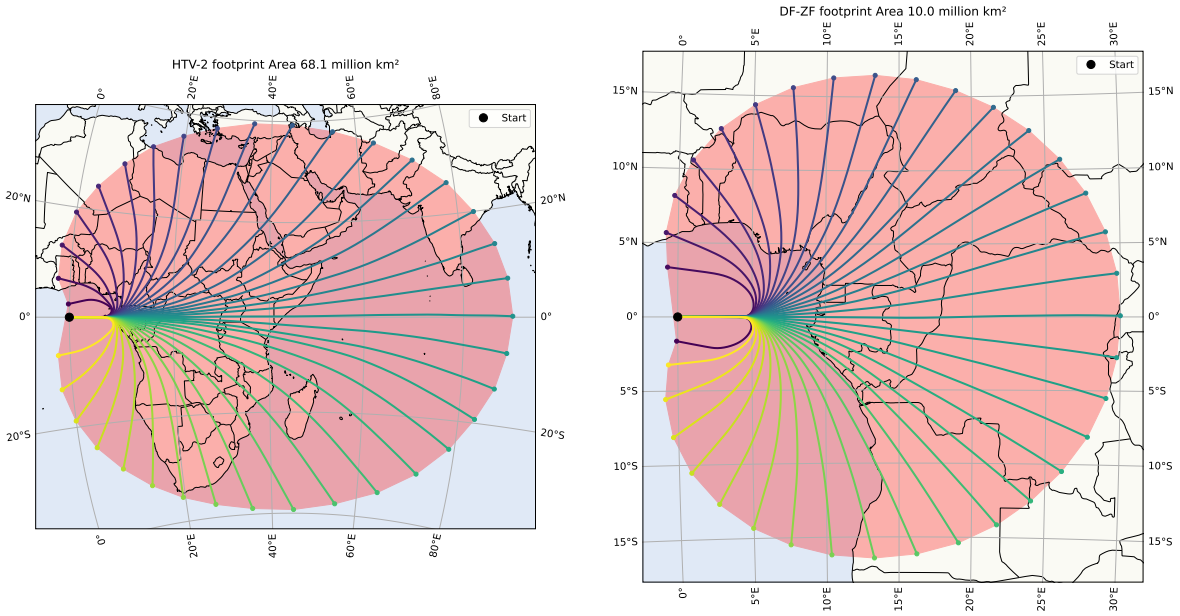
(a) HTV-2 footprint trajectories and target points.

(b) DF-ZF footprint trajectories and target points

Figure 6.25: Footprint trajectories and target points for HTV-2 and DF-ZF, demonstrating the 36-direction azimuthal sampling strategy.

6.19.6. Footprint Analysis

The footprint analysis provides a view of each HGV operational reach by simulating the trajectories in 36 evenly distributed directions. This section presents the comparative results. The footprints for the HTV-2 and DF-ZF are shown in Figure 6.26, for a starting position of latitude $0°$, longitude $0°$ and initial heading $90°$.



(a) Footprint of the HTV-2 HGV.

(b) Footprint of the DF-ZF HGV.

Figure 6.26: Comparison of footprints for the HTV-2 and DF-ZF hypersonic glide vehicles from the initial position across 36 evenly distributed heading directions. The red shaded area represents the operational range envelope, and trajectory colors are used to distinguish individual flight paths.

The two HGV's show different operational capabilities. The initial flight conditions and the resulting footprint area, maximum range, minimum range (measured as great-circle distance to starting position), and maximum cross-track distance are given in Table 6.2. As explained in Section 6.17, these two HGV's are for different operations and therefore have a different range.

Table 6.2: Performance comparison of HTV-2 and DF-ZF footprints.

Parameter	**HTV-2**	**DF-ZF**
Initial Conditions		
Altitude (km)	100	100
Velocity (m/s)	6000	3450
Flight Path Angle (°)	-3.0	0.0
Footprint Metrics		
Footprint Area (million km²)	68.1	10.0
Maximum Range (km)	9923	3335
Minimum Range (km)	290	182
Maximum Cross-Range (km)	4431	1829

Both footprints show clear symmetry about the eastward direction, which is the expected behavior for launches from the equator. This north-south symmetry arises because Coriolis forces at equal latitudes above and below the equator have equal magnitude but opposite signs in the north-south direction. A trajectory heading northeast at $45°$ experiences Coriolis effects that mirror the southeast trajectory at $135°$, resulting in symmetric deflection patterns. The reinforcement learning controllers have learned to compensate for these forces identically in both hemispheres, producing the balanced footprint shapes visible in both figures.

The overall size of the footprint is an indication of the total mechanical energy of the HGV at the beginning. A higher initial speed or altitude increases the available specific energy, which allows the vehicle

to travel further. That is why the HTV-2 footprint is substantially larger than the DF-ZF footprint. The shape similarity between both shows that their control logic is consistent.

The smooth and continuous boundary of the red-shaded region in both two footprints demonstrates successful policy learning across all target directions. Since all 36 trajectories are computed independently using parallel processes, any discontinuities or local asymmetries would indicate numerical instabilities or inconsistencies in the reinforcement learning policy. The fact that the footprint forms a continuous and nearly circular contour without gaps or abrupt curvature changes confirms that the dynamics and control logic behave across all headings. In other words, the geometry of the footprint acts as a visual verification that the integrated simulation framework, which consists of the physics model, control policy and constraint enforcement, is stable and consistent over the entire range of flight directions.

Within the closed contour, it is assumed that every point can be reached through an appropriate combination of bank and angle of attack control. This assumption follows from the fact that the footprint contour is constructed by connecting all end points of the trajectories. Each trajectory corresponds to a distinct heading and all their terminal coordinates define the outer boundary of the footprint. In practice, local constraints or control limitations will result in some interior points being infeasible, especially near the start location of the HGV. To address this, a dedicated target point guidance system has been made, which allows the simulation of individual trajectories toward any location within the footprint. Here it can be checked for every location whether it is feasible and is further explained in Section 6.20.

6.19.7. Cross Track Distance

The cross track distance, measuring the maximum lateral displacement perpendicular to the maximum range direction, reaches 4431 km for the HTV-2 and 1829 km for the DF-ZF. This metric quantifies the HGV's ability to perform lateral maneuvers while still achieving substantial range. The ratio of maximum cross-track distance to maximum longitudinal range is a measure of maneuverability. Higher ratios mean greater lateral reach relative to straight line performance, which reflects the capability for the HGV to reach target points that are not aligned with the initial heading. The cross-track performance comparison is given in Table 6.2.

6.19.8. Footprint Validation

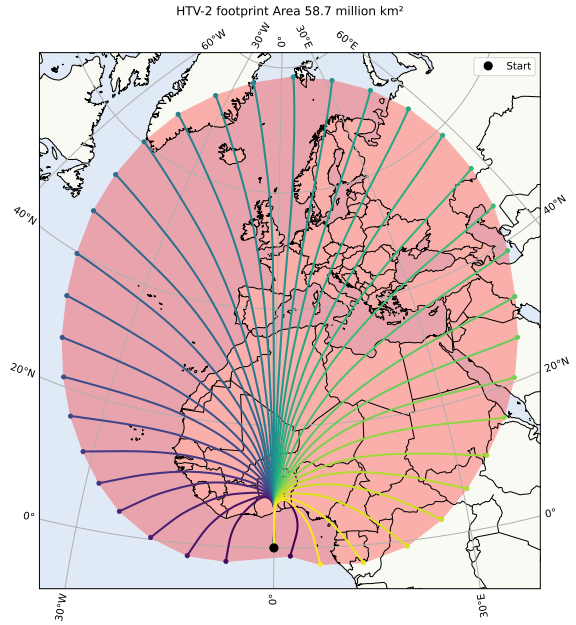
To validate the accuracy of the footprint generation framework, several verification tests were performed. These tests confirm that the simulation physics and reinforcement learning policies behave consistently across different geographic locations and flight conditions.

North-South Symmetry and Coriolis Effect Verification

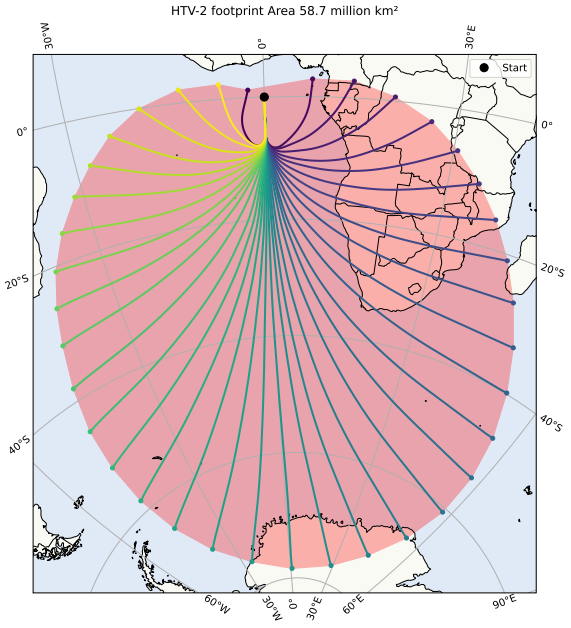
A test has been performed that examines the symmetry properties of footprints generated from identical initial conditions in the northern and southern hemispheres. This validation exploits a property of Earth's rotational dynamics. The Coriolis acceleration has equal magnitude but opposite direction at symmetric latitudes above and below the equator.

The Coriolis acceleration arises from Earth's rotation and deflects moving objects perpendicular to their velocity. In the northern hemisphere this deflection is to the right of the motion direction, while in the southern hemisphere it deflects to the left. For a vehicle flying north from a location in the northern hemisphere, the rightward Coriolis deflection biases the trajectory eastward. The mirror scenario, where the HGV flies south from the equivalent latitude in the southern hemisphere, experiences a leftward deflection, which for a southward moving vehicle also results in an eastward bias. Both trajectories are deflected eastward, but from the perspective of the HGV one deflection is to the right and the other to the left, which should create mirror-symmetric footprint patterns.

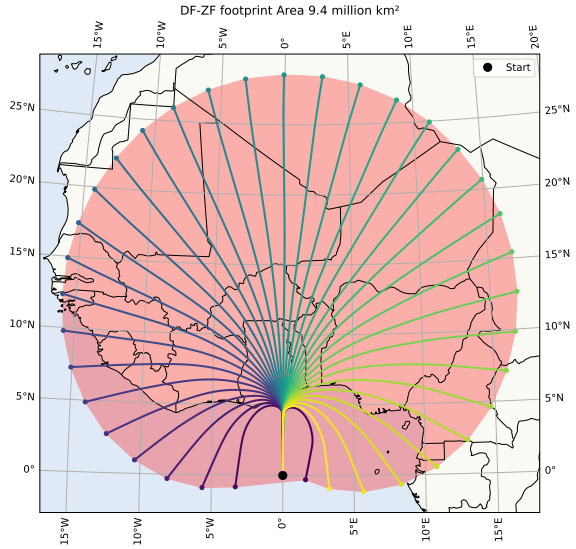
To verify this behavior, footprints were generated for both the HTV-2 and DF-ZF HGVs from two pairs of symmetric launch conditions. These configurations both launch from the equator but fly in opposite directions, shown in Figure 6.27 for the HTV-2 and DF-ZF. Although the Coriolis force is zero at the equator itself, as the vehicle moves away from the equator to higher latitudes, it experiences increased Coriolis acceleration.



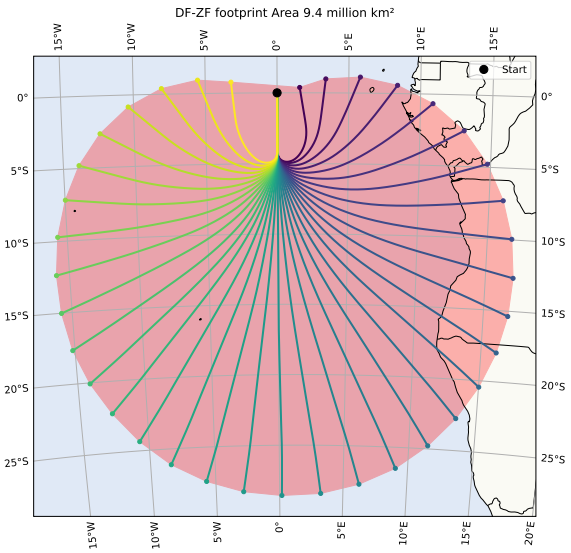
(a) Footprint of the HTV-2 launched northward from the equator ($\psi_0 = 0°$), showing eastward deflection due to Coriolis effect.



(b) Footprint of the HTV-2 launched southward from the equator ($\psi_0 = 180°$), showing eastward deflection due to Coriolis effect.



(c) Footprint of the DF-ZF launched northward from the equator ($\psi_0 = 0°$), showing eastward deflection due to Coriolis effect.



(d) Footprint of the DF-ZF launched southward from the equator ($\psi_0 = 180°$), showing eastward deflection due to Coriolis effect.

Figure 6.27: Footprints for opposite launch headings for both HTV-2 and DF-ZF to verify the North-South Symmetry and Coriolis Effect.

The quantitative metrics for the footprint comparisons is summarized in Table 6.3.

Table 6.3: North-South symmetry validation for Coriolis effect

Vehicle & Launch Config	Footprint Area (million km²)	Max Range (km)	Min Range (km)	Max Cross-Track (km)
HTV-2 (0°, 0°, 0°)	58.7	8747.6	313.9	4346.1
HTV-2 (0°, 0°, 180°)	58.7	8758.5	304.4	4327.5
HTV-2 Difference	**0.0%**	**+0.12%**	**−3.0%**	**−0.43%**
DF-ZF (0°, 0°, 0°)	9.4	3172.3	180.4	1861.5
DF-ZF (0°, 0°, 180°)	9.4	3175.0	180.3	1868.0
DF-ZF Difference	**0.0%**	**+0.08%**	**−0.06%**	**+0.35%**

Area Symmetry Validation

The identical footprint areas between northbound and southbound configurations for each HGV provides a validation of the simulation's correctness. For the HTV-2, both 0° and 180° heading configurations produce exactly 58.7 million km^2 footprints. Similarly for DF-ZF, areas of 9.4 million km^2 were produced in both cases. This is not coincidental, but a consequence of the symmetry in Earth's rotational dynamics.

The reasoning is fairly straightforward. Both configurations start with identical initial energy states and are subject to identical magnitudes of Coriolis acceleration, gravity, and aerodynamic forces, only the geometric orientation differs. Symmetric trajectories must dissipate energy at the same rate and therefore cover equal areas. Any discrepancy in footprint areas would indicate incorrect implementation of the Coriolis force terms, asymmetric behavior in the RL training policy, or numerical instabilities in the trajectory training. The zero difference observed in areas demonstrates that none of these errors are present.

Eastward Bias Confirmation

Examining Figure 6.27 shows a consistent pattern across all four footprints. The eastern side of each footprint extends further than the western side. This eastward bias confirms the correct implementation of the Coriolis force.

For the northbound trajectories, the Coriolis acceleration deflects the vehicle rightward as it enters the northern hemisphere. For the southbound trajectories, the Coriolis deflection is leftward relative to the HGV's motion, but as the vehicle faces south this is also eastward. Thus, both configurations experience eastward drift, which creates the observed asymmetric footprint shapes. The consistent eastward bias across the four footprints validates that the RL agents naturally learned to work with rotational dynamics and emerged naturally from the learned control policies.

Analysis of Metric Variations

For both vehicles, the maximum range shows a relative difference of 0.12% for the HTV-2 and 0.08% for the DF-ZF. The maximum cross-track distances show similarly small variations: -0.43% for HTV-2 and 0.35% for DF-ZF. Cross-track differences depend on sustained lateral maneuvering throughout the trajectory, requiring the policy to continuously compensate for Coriolis deflections while banking to change heading. This sub-0.5% difference indicates that the policy executes these complex maneuvers with consistent precision regardless of the hemisphere.

In contrast, the minimum range shows larger relative differences: 3.0% for HTV-2 and 0.06% for DF-ZF. However, this percentage is misleading. The absolute differences are only 9.5 km for HTV-2 and 0.1 km for DF-ZF, both negligible compared to the total trajectory lengths. At these short distances, even minor variations in the turn initiation point or bank angle profile produce larger percentage changes.

It is important to note that the Coriolis acceleration is relatively weak compared to the dominant aerodynamic and gravitational forces acting on the vehicle. The Coriolis acceleration scales with $2\omega_\oplus V$, where $\omega_\oplus$ is Earth's rotation rate of about $7.29 \cdot 10^{-5}$ rad/s and V is the vehicle velocity. Even at hypersonic speeds of 6000 m/s, the Coriolis acceleration is approximately 0.87 m/s^2, which is less than 10% of the gravitational acceleration and orders of magnitude smaller than the aerodynamic forces during flight. This relatively small magnitude means that the Coriolis effects will show as a gradual drift over

long flight durations and increase in range in eastward direction, and a decrease in range in westward direction.

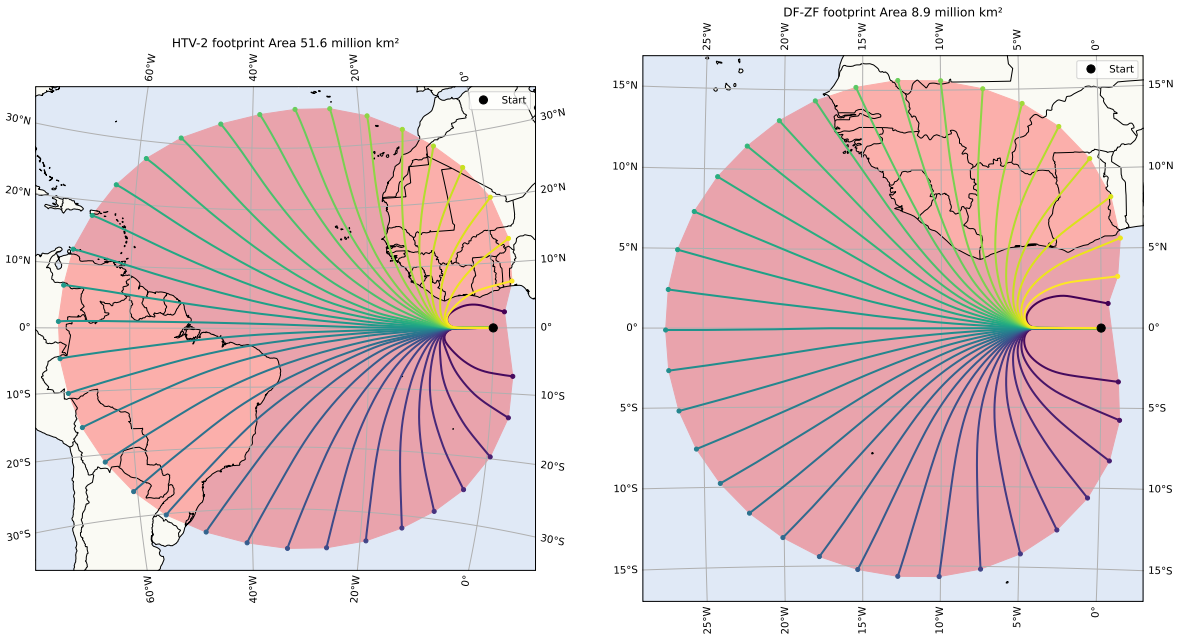
All these metric variations fall well within the expected bounds for a complex nonlinear simulation involving a neural network policy evaluation. This shows that the neural network performs consistently and validates the numerical stability of the integrated framework.

Policy Generalization

During training, the agents were exposed to randomized initial conditions containing different latitudes and headings across the entire globe, but it was never told that northern and southern hemisphere flights should behave symmetrically. The neural network policies received no information on the hemisphere in which it is operating when running the file, as explained in Section 6.15.7. The fact that the policies produce symmetric results without being explicitly trained for symmetry demonstrates effective generalization.

East-West Asymmetry and Rotational Effects

To further validate the model, an additional comparison was performed between eastward and westward launches from identical initial positions. Footprints were generated for both HGVs launching with heading $90°$, shown in Figure 6.26, and $270°$ shown in Figure 6.28.



(a) Footprint of the HTV-2 launched westward from the equator ($\psi_0 = 270°$), showing reduced range.

(b) Footprint of the DF-ZF launched westward from the equator ($\psi_0 = 270°$), showing reduced range.

Figure 6.28: Footprints for westward launches showing reduced range compared to eastward trajectories due to Earth's rotation effects.

The results are presented in Table 6.4.

Table 6.4: East-West footprint comparison

Vehicle & Heading	Footprint Area (million km²)	Max Range (km)	Min Range (km)	Max Cross-Track (km)
HTV-2 (90° East)	68.1	9923	290	4432
HTV-2 (270° West)	51.6	7926	334	4085
HTV-2 Difference	**−24.2%**	**−20.1%**	**+15.2%**	**−7.8%**
DF-ZF (90° East)	10.0	3335	182	1829
DF-ZF (270° West)	8.9	3040	178	1742
DF-ZF Difference	**−11.0%**	**−8.8%**	**−2.2%**	**−4.8%**

The results show substantial differences between eastward and westward flight performance, where eastward launches consistently achieve larger footprint areas and longer maximum ranges. For the HTV-2, the eastward footprint covers 68.1 million km^2 compared to 51.6 million km^2 for westward flight, which translates to a 24.2% reduction when flying west. The maximum range decreases from 9923 km eastward to 7926 km westward, translating to a 20.1% performance reduction when flying west. The DF-ZF shows similar trends with a 11.0% area reduction and a 8.8% maximum range reduction when flying westward.

6.19.9. Physical Explanation

These asymmetries come from the velocity-dependent terms in the Coriolis acceleration within Earth's rotating reference frame. The Coriolis acceleration is given by:

$$\mathbf{a}_{\text{Coriolis}} = -2\boldsymbol{\omega}_E \times \mathbf{v} = \begin{bmatrix} 2\omega_\oplus v_y \\ -2\omega_\oplus v_x \\ 0 \end{bmatrix} \tag{6.73}$$

where $\boldsymbol{\omega}_E = [0,\, 0,\, \omega_\oplus]^\top$ is Earth's rotation vector and $\mathbf{v} = [v_x,\, v_y,\, v_z]^\top$ is the vehicle velocity in the ECEF frame.

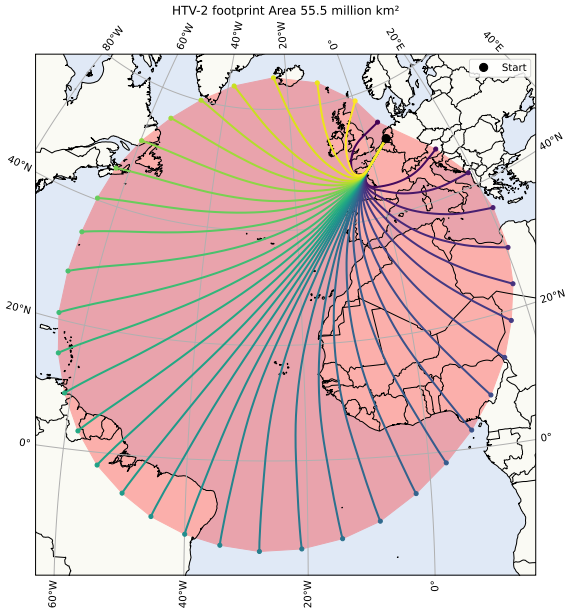
The key insight is that these Coriolis terms ($2\omega_\oplus v_y$ and $-2\omega_\oplus v_x$) have components in the x and y directions. When decomposed at the vehicle's position, these accelerations have radial components in addition to tangential components. For eastward flight, the vehicle moves in the same direction as Earth's rotation. The resulting Coriolis acceleration, when resolved into radial components at each point along the trajectory, produces a net outward effect that partially counteracts gravity. This deduces the effective gravitational acceleration, which allows the vehicle to maintain altitude with less lift and dissipate less energy, thereby extending the range. For westward flight, the vehicle moves opposite to Earth's rotation. The Coriolis acceleration produces radial components that increase the effective gravitational acceleration, requiring more lift to maintain altitude and causing faster energy dissipation, which reduces range.

The observed 20.1% range difference for the HTV-2 and 8.8% difference for the DF-ZF validate the correct implementation of these velocity-dependent Coriolis terms. The smaller percentage for the DF-ZF reflects its lower initial velocity, which reduces the magnitude of velocity-dependent Coriolis effects.

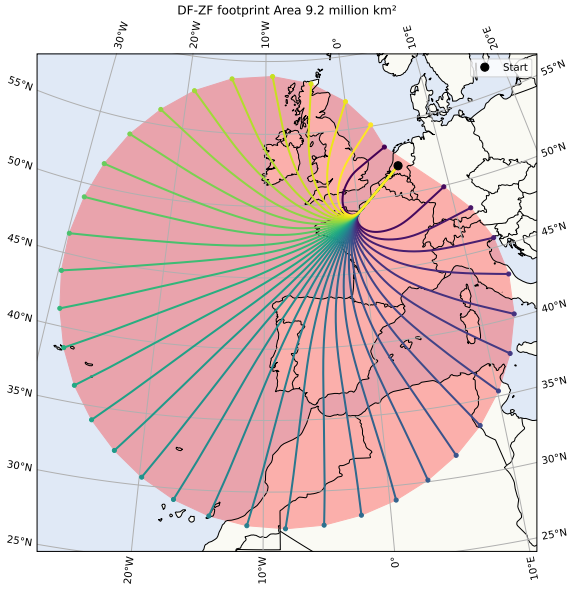
This velocity-dependent radial component of the Coriolis acceleration is different from the position-dependent centrifugal acceleration $\omega_\oplus^2 r$, which affects all trajectories equally regardless of flight direction.

6.19.10. Global Applicability and Geographic Independence

To demonstrate that the trained reinforcement learning policies generalize to arbitrary launch locations worldwide, footprints were generated from various geographic positions. Figure 6.29 shows examples for the HTV-2 and DF-ZF launching from $52°$ latitude, $5°$ longitude with a heading of $230°$.



(a) Footprint of the HTV-2 launched from 52°N, 5°E with heading $\psi_0 = 230$°, demonstrating policy generalization to a different location.



(b) Footprint of the DF-ZF launched from 52°N, 5°E with heading $\psi_0 = 230$°, demonstrating policy generalization to a different location.

Figure 6.29: Comparison of two footprints of different HGVs on arbitrary location on Earth.

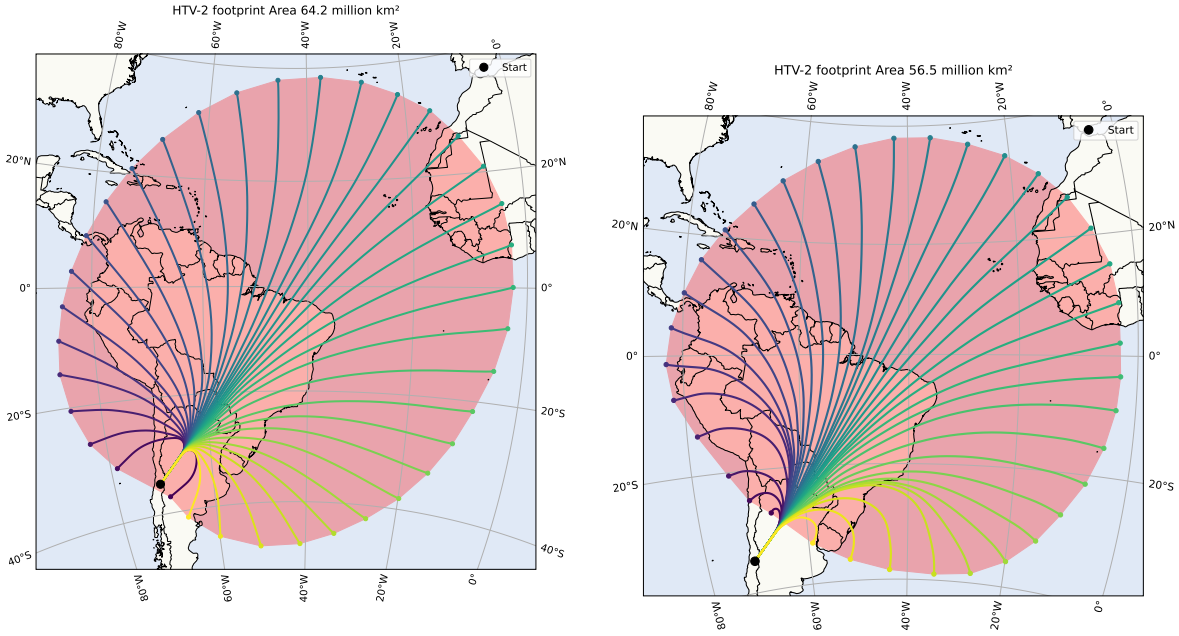
The successful generation of complete, continuous footprints from any geographic position validates an important aspect of the training methodology. The policies operate correctly anywhere on Earth, and the target point generation system adapts automatically to any launch location.

During training, the reinforcement learning environment randomly assigned the initial latitude for every episode, which exposes the agent to launch conditions from the southern to northern hemispheres. In contrast, the longitude was not randomized in any way. Longitude has no physical significance for vehicle dynamics on a uniformly rotating spherical Earth. The Coriolis acceleration, gravitational field and atmospheric properties depend only on latitude and altitude. Therefore, the framework automatically handles longitude without additional training.

The footprint generation system also adapts target points placement to any launch location. The 36 radial target directions are computed relative to the initial position and heading. Whether the start position is on the equator, mid-latitudes or polar regions, the system generates the target points and the policy adapts the control strategy according to the local environment. This geographic independence confirms that a single trained policy can be used globally without having to retrain the whole model for every location.

6.19.11. Bank Angle Limit Impact on Footprint

To evaluate the effect of control authority on the reachable footprint area, two versions of the HTV-2 policy were trained with different bank angle limits. A constrained version limited to ± 60° and a full-authority version allowing ± 90°. The only difference between both policies is the action space bounds for bank angle commands. Footprints were generated from the same arbitrary launch location (35°S, 71°W, 45° heading) to see a direct comparison. Figure 6.30 shows the resulting footprints for the 60° and 90° bank angle limits.



(a) Footprint of the HTV-2 HGV with full control authority (bank angle $\pm 90°$).

(b) Footprint of the HTV-2 HGV with limited control authority (bank angle $\pm 60°$).

Figure 6.30: Comparison of the two footprints of the HTV-2 with different control authorities.

The quantitative comparison is given in Table 6.5

Table 6.5: Bank angle limit comparison

Bank Angle Limit	Footprint Area (million km²)	Max Range (km)	Min Range (km)	Max Cross-Track (km)
±60°	56.5	9371	930	4335
±90°	64.2	9396	299	4361
Difference	**+13.6%**	**+0.3%**	**−67.8%**	**+0.6%**

The full authority bank angle policy achieves a 13.6% larger footprint area compared to the limited bank angle policy version, which shows that the increased control authority directly relates to the operational capability. The maximum range shows almost no difference between the two policies ($+0.3\%$), which is a logical outcome, since for the maximum range, minimal bank angles are required. The same logic applies to the maximum cross-track, where the difference is only 0.6%.

The most dramatic difference appears in the minimum range, which decreases by 67.8% when using $\pm 90°$ bank angles. This demonstrates the ability of the HGV to execute tight direction reversals. To achieve minimum range, as was shown in Figure 6.20, the HGV must perform an aggressive turn to reverse course. With only $\pm 60°$ bank available, the turn radius is larger, which requires the vehicle to travel further before completing the reversal maneuver. The $\pm 90°$ can execute much sharper turns with higher lateral accelerations. By reducing the minimum range, meaning the HGV gets closer to the initial position, the footprint area will also increase, in this case with 13.6%.

6.19.12. 3D Footprint Generation

The 2D footprint shows the horizontal reach but discards altitude information. The 3D visualization incorporates altitude profiles, which shows the HGVs energy management strategies throughout the flight.

The 3D footprint uses the same parallel computation framework as the 2D analysis. The implementation distributes 36 trajectory computations across multiple CPU cores. The 3D footprint of the HTV-2 is shown in Figure 6.31.

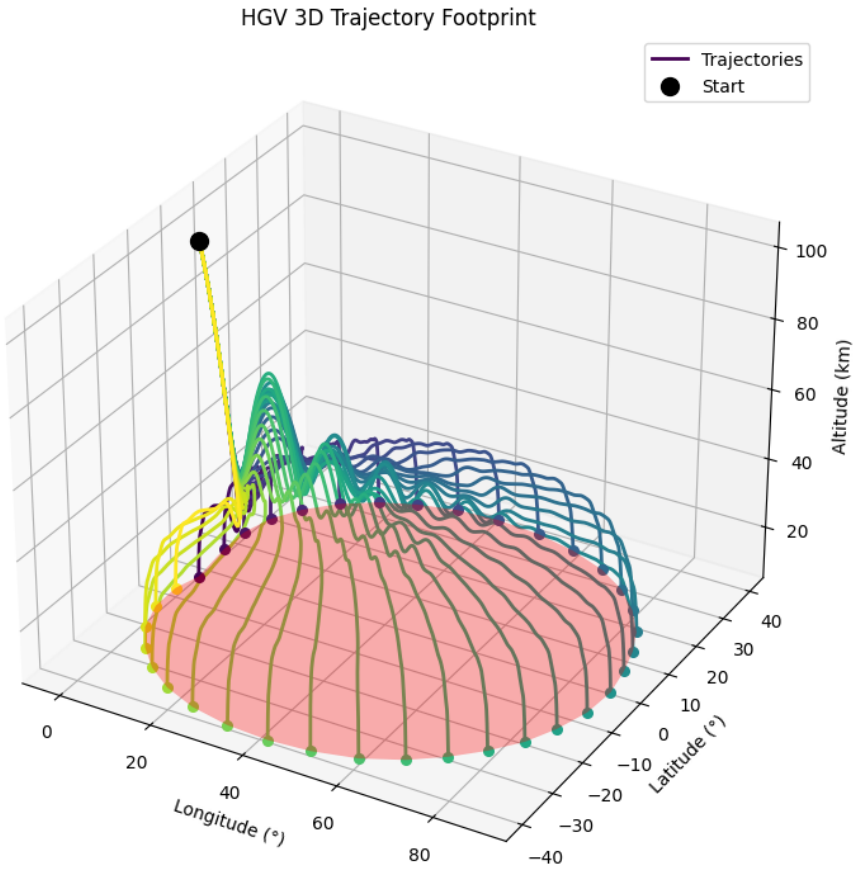


Figure 6.31: 3D footprint of the HTV-2 showing skip-glide trajectories.

Skipping Behavior

In Figure 6.31 the 3D footprint of HTV-2 is shown starting from 100 km altitude at 6000 m/s with an initial flight path angle of $-3°$. This 3D visualization clearly shows the skipping behavior for maximum range flights. In contrast to this, minimum range trajectories show completely suppressed skipping behavior, since the goal is to make a direction reversal as soon as possible and will do this by banking the vehicle, reducing the lift and the oscillatory behavior.

Comparison to DF-ZF 3D Footprint

Figure 6.32 shows the DF-ZF footprint from 100 km at 3450 m/s with an initial flight path angle of $0°$. The lower initial energy state is clearly visible in both the reduced horizontal range and the suppressed altitude dynamics. The footprint geometry is more compact, which also reflects the limited energy of the vehicle compared to the initial conditions of the HTV-2.

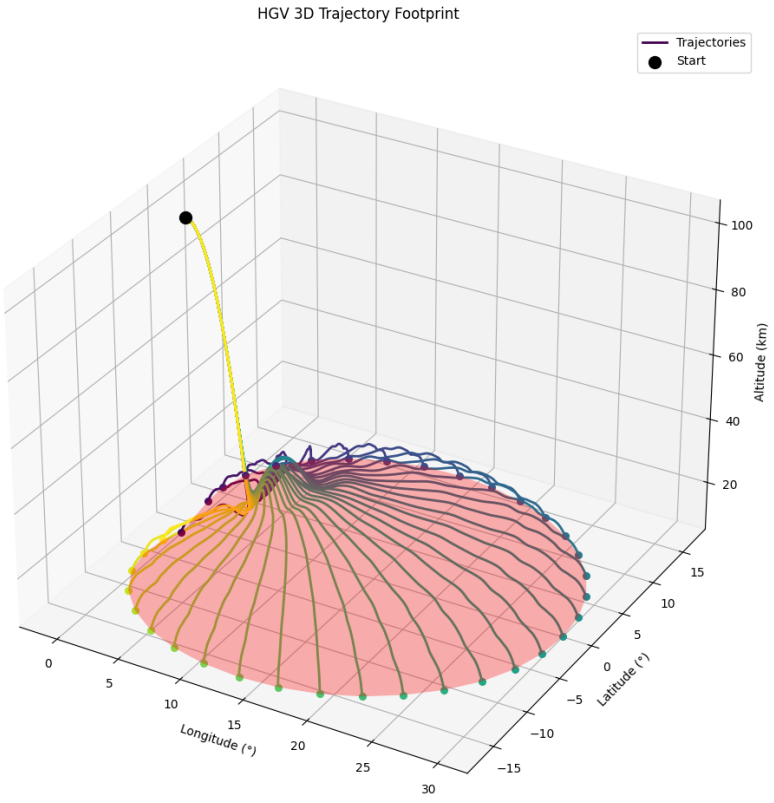


Figure 6.32: 3D footprint of the DF-ZF showing lower altitude skip trajectories due to reduced energy state compared to the HTV-2.

6.20. Extension to Direct Target Point Guidance

The footprint generation framework that was presented in previous sections successfully shows the operational footprint of the HGV by simulating trajectories in 36 evenly spaced radial directions from the initial position. Although this approach effectively maps the boundary of reachable territory, it limits the system to discrete target selection. In practical operational scenarios, mission planners could require the capability to specify an arbitrary target within the footprint. This limitation motivated the development of an extended guidance framework that accepts continuous geographic coordinates as target specifications. Rather than selecting from 36 discrete options, the system now supports navigation to any latitude-longitude coordinate pair within the operational footprint.

6.20.1. Uniform Target Point Distribution

The most significant modification involves how target points are generated during training episodes. Instead of randomly selecting one of 36 fixed indices with evenly spaced target points in the footprint, the extended approach generates random points throughout the reachable footprint.

Generating random latitude-longitude pairs directly would create non-uniform distribution due to Earth's spherical geometry. To achieve uniform spatial distribution, a two-step process is used. First, a bearing angle is uniformly sampled between $0°$ and $360°$ to determine the direction from the initial position. Second, the radial distance uses the square root of a uniform random variable. The square root transformation is crucial for uniform area coverage. Without it, points would accumulate near the center of the reachable circle. On a spherical surface, the area of concentric rings grows proportionally to the radius. Therefore, to achieve equal probability per unit area, the probability density must increase linearly with radius, which is exactly what the square root transformation gives.

The target points are distributed within a circle where the center is displaced from the initial position by half the maximum range in the initial heading direction, and the radius of the circle is half the maximum

range plus an additional padding, also used for the target points to ensure that all possible locations are included. An example of a random uniform target point distribution is shown in Figure 6.33.

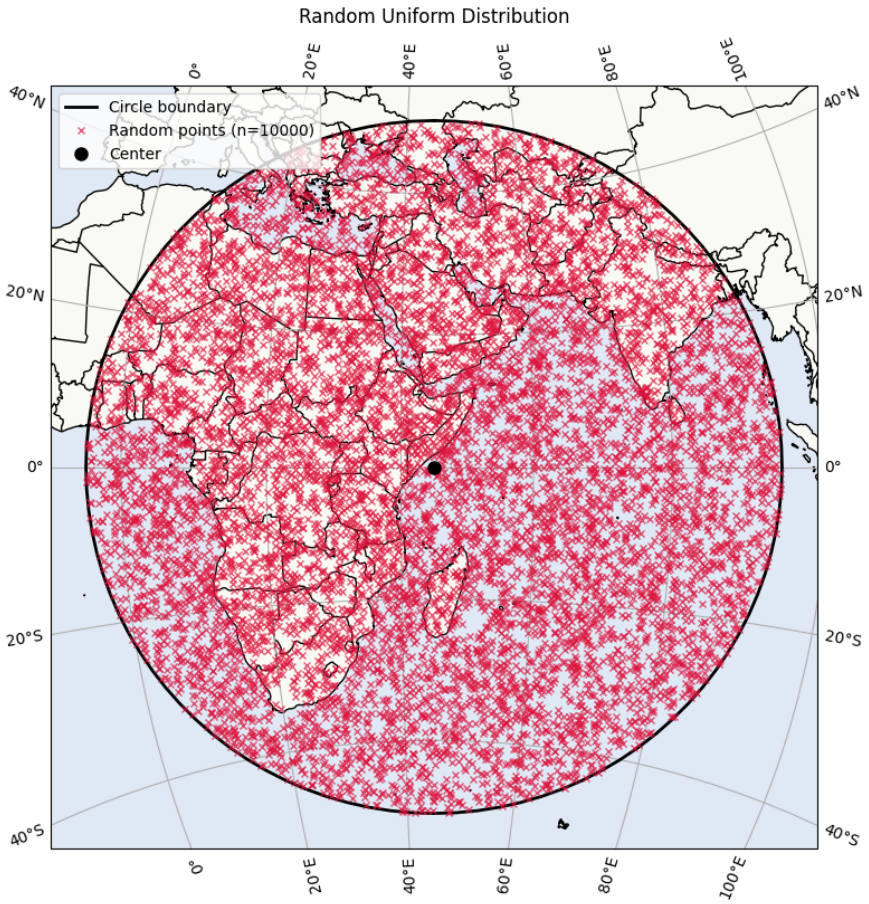


Figure 6.33: Uniform spatial distribution of randomly generated target points, showing correct area-uniform sampling.

6.20.2. Multi-Dimensional Randomization

Training randomizes three parameters simultaneously. These include the initial latitude, initial heading and the target location. This guarantees that the agent encounters diverse configurations, including scenarios that require small course corrections, large heading changes, and varying distances from minimum to maximum range.

6.20.3. Observation Space

The observation space remains five dimensional: altitude, velocity, flight path angle, heading and bearing difference to target point and distance to target point. The difference from the index-based approach is that these measurements are now computed for arbitrary target coordinates rather than predetermined points. The agent must learn to navigate across the full range of possible distances and bearings, instead of only the maximum range boundary trajectories.

6.20.4. Reward Function

The reward structure is the same as for the footprint generation; it combines progress toward the target with constraint penalties. Progress is measured as the great-circle distance to the target between time steps. Constraint penalties apply exponentially as the vehicle approaches the limits, as explained in Section 6.15.8.

The continuous formulation changes the agent's navigation task. With targets distributed throughout the footprint rather than fixed at the boundary, the agent must now learn distance-dependent flight strategies. For targets near the maximum range boundary, the optimal trajectory must maximize the horizontal distance before descent. However, for targets at intermediate distances within the footprint, continuing to maximize range would result in overshooting the target location. The agent must therefore learn to recognize when sufficient progress has been made and transition to a descent phase at the appropriate point such that it ends at that specific point. The reward function remains identical, but the diversity of target distances that have been encountered during the training teaches the agent to adapt the energy management strategy based on the target point specific range requirements, instead of always maximizing distance.

6.20.5. Generalization Capability

The continuous approach allows effective generalization to specific target coordinates that have not been seen in training. Because training exposes the agent to targets distributed throughout the entire footprint at varying distances and bearings, the agent encounters situations similar to any potential test scenario. When a new target coordinate is presented during the evaluation, the agent has experienced nearby targets during training with comparable relationships. The neural network interpolates between these training experiences, applying the learned navigation principles to the specific target. This interpolation-based generalization is different from the discrete approach, where performance between the 36 trained directions remains uncertain as the agent never experienced intermediate geometries during training.

6.20.6. Validation

Validation testing evaluates the direct target point guidance capability using fixed target coordinates from various initial conditions. These tests demonstrate the system's ability to navigate accurately to arbitrary locations while maintaining constraint satisfaction.

The first target point considered involves navigation from the Netherlands (52° N, 5° E) with an initial southwest direction of 230° to the location 39° N 15° E. The vehicle begins at 100 km altitude with 6000 m/s velocity and -3° flight path angle. The trajectory and the appropriate control authority are shown in Figure 6.34.

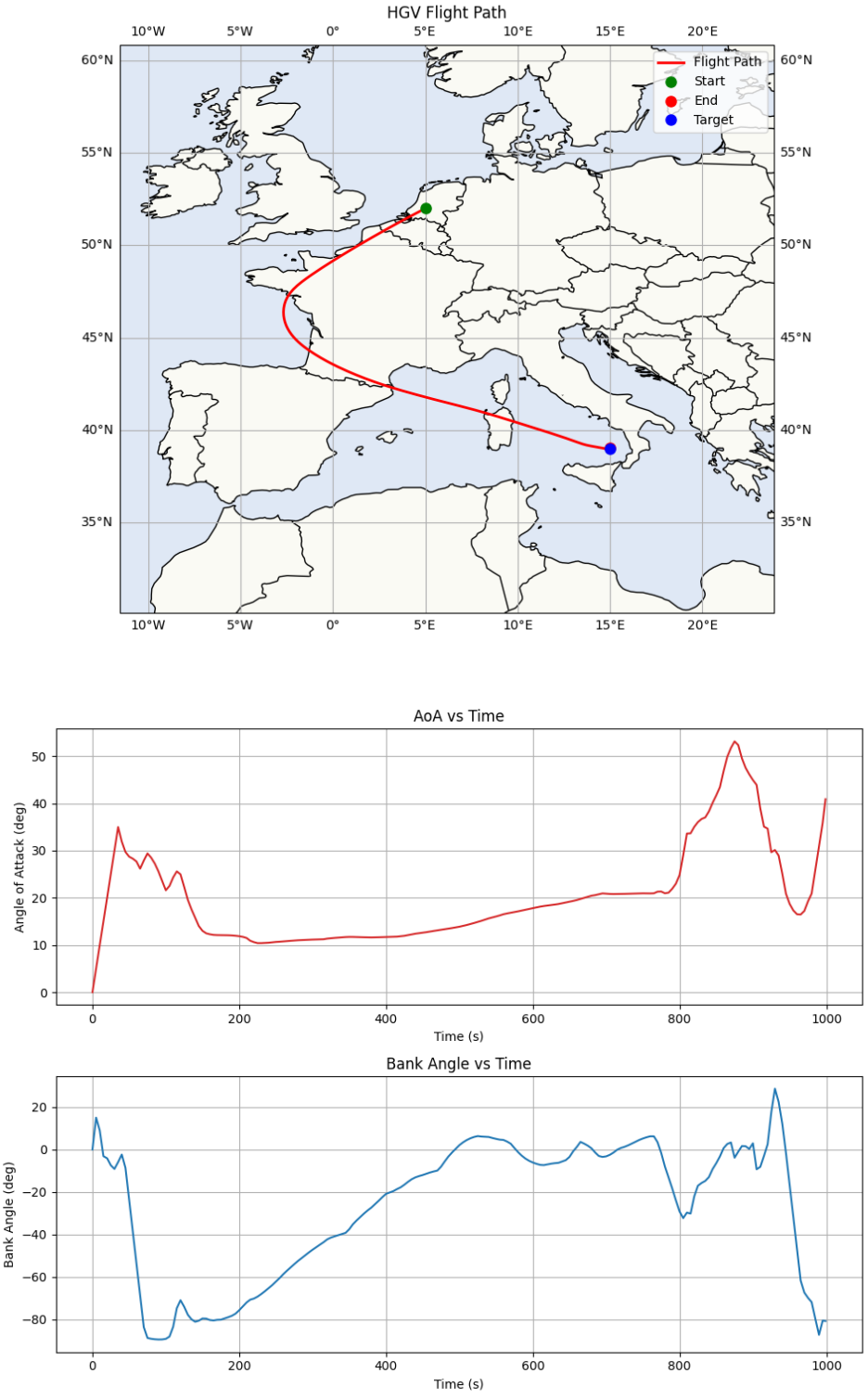


Figure 6.34: Ground trajectory and control variable inputs to navigate towards arbitrary target point.

The trajectory successfully navigates the 1636 km great-circle distance, arriving at the target with a terminal positioning error of only 2.93 km. The trajectory shown in Figure 6.34 shows the geographic flight path, which shows a smooth trajectory. The vehicle executes an initial large bank angle maneuver

to align with the target bearing, which is followed by gradual course corrections as it approaches the destination.

The control history shows the learned navigation strategy. The angle of attack shows an initial increase to approximately $35°$, while banking steeply for the initial turn when it experiences highest dynamic pressure. Afterwards, the angle of attack stabilizes to efficient glide conditions. As the vehicle approaches the target region, the angle of attack increases significantly together with the bank angle to minimize the positioning error and get as close to the target point as possible.

Throughout the whole flight, all path constraints remain satisfied. All constraints are continuously monitored during the trajectory and for every run of the program, it is checked at every step if a violation occurs and will be reported if this is the case.

A second test case is shown in Figure 6.35. This is a short-range navigation starting from $0°$ N, $0°$ E to the target point located at $0°$ N, $35°$ E. Figure 6.15 in Section 6.18.1 shows that the HTV-2 can reach locations further than $80°E$. This relatively short distance requires the vehicle to dissipate substantial energy to avoid overshooting.

The HTV-2 successfully reached the target with a terminal positioning error 3.38 km on a total distance of 3893 km. The trajectory shows the flight path maintaining nearly constant latitude along the equator.

The control profiles show that energy management strategies are fundamentally different from maximum range scenarios. The angle of attack profile shows that as the flight progresses, it progressively increases to high values, generating significant drag to dissipate the excess energy to arrive at the target. The sustained high angles of attack through the mid-trajectory phase shows the agent's learned strategy for range reduction towards a specific goal.

The bank angle shows the lateral maneuvering with multiple systematic reversals. It can be seen that the bank angle oscillates between more positive and more negative bank angles. This high bank angle helps the HGV to lose altitude and energy more quickly by tilting the lift vector from the vertical. The bank reversals are performed such that the HGV is able to lose this energy while still remaining on course for the target that is placed in a straight line from the initial position. In other methods in literature, such as the one used by Mooij et al. for a guidance system of HORUS [38], four bank reversals are employed to get the HORUS to the terminal condition.

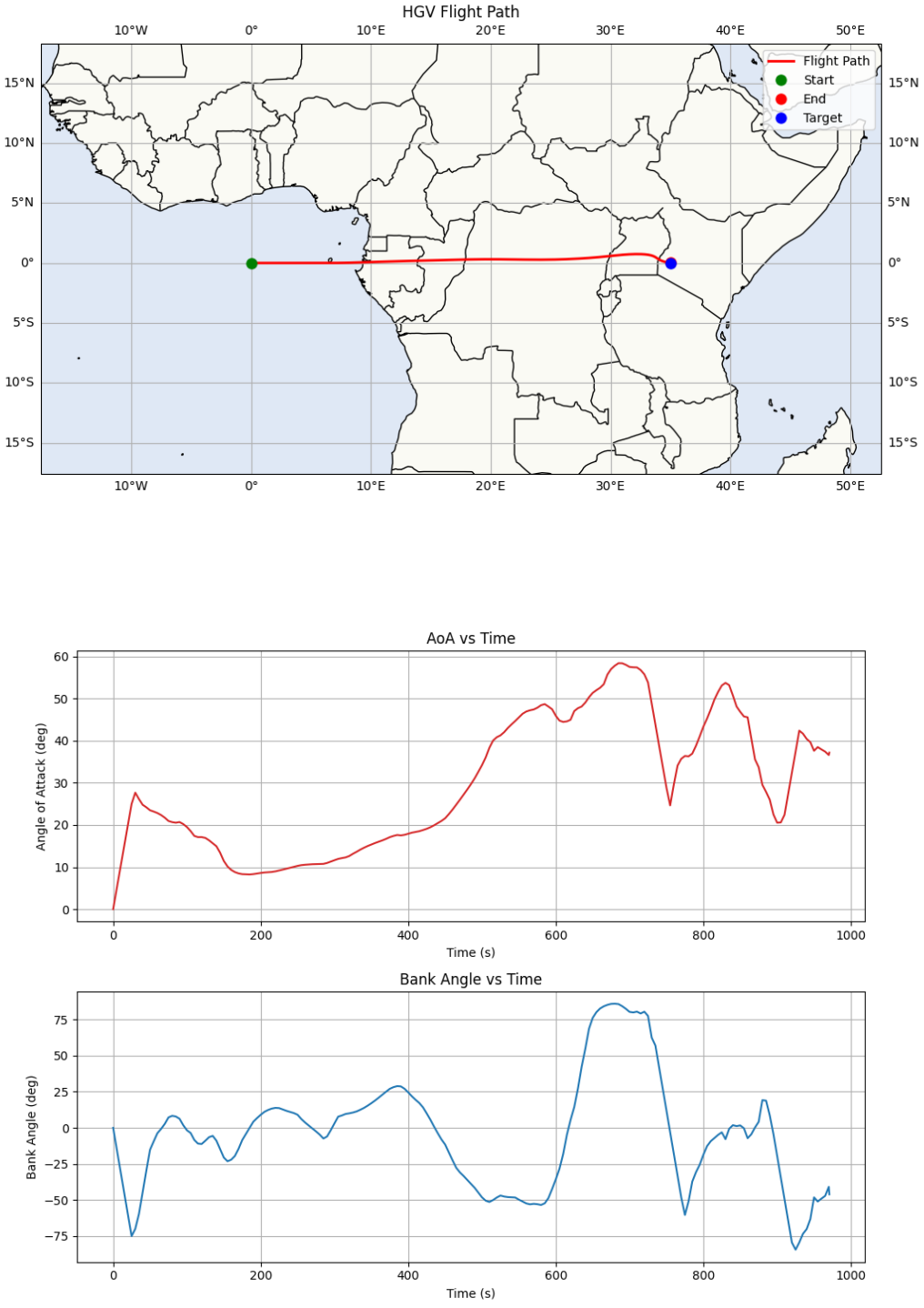


Figure 6.35: Ground trajectory and control variable inputs to navigate towards target point close to the initial position, demonstrating learned bank reversals.

6.20.7. Discount Factor Selection for Target Point Navigation

For target point navigation, the discount factor γ of 0.99 is important to ensure that the agent learns time efficient trajectories. The cumulative discounted reward is calculated as:

$$R_{\text{total}} = \sum_{t=0}^{T} \gamma^t r_t = r_0 + \gamma r_1 + \gamma^2 r_2 + \ldots + \gamma^T r_T \tag{6.74}$$

where R_{total} is the total discounted reward over the trajectory, γ the discount factor, t the time step index, T the total number of steps and r_t the immediate reward received at time step t.

When $\gamma = 1.0$, all future rewards are valued equally regardless of when they occur, which means the agent has no incentive to reach the target quickly. A trajectory taking 100 steps and one taking 200 steps would receive identical total rewards if both eventually reach the target, even though the shorter trajectory is preferable.

By setting $\gamma = 0.99$, future rewards are discounted by 1%, which introduces a mild preference to receive rewards sooner. This creates a subtle but crucial structure, namely that trajectories that reach the target in fewer steps accumulate to higher total discounted rewards.

The choice of 0.99 is an optimal balance. Lower values like 0.97 or 0.95 would make the agent overly short sighted, which could sacrifice the ability to reach distant targets in favor of immediate rewards. The value of 0.99 is strong enough to prefer shorter trajectories while maintaining the long-term planning capabilities to achieve the maximum range.

6.20.8. Footprint Generation with Enhanced Resolution

The direct target point guidance model demonstrates an additional capability beyond navigation to arbitrary coordinates. It can generate operational footprints with enhanced resolution. The continuous target point model allows for footprint visualization with any desired number of trajectories. The index-based footprint generation is also capable of generating these enhanced resolution due to the generalization between the learned target points.

Figure 6.36 shows the HTV-2 footprint generated using 360 trajectories. Each trajectory represents the maximum-range flight toward a distinct bearing from the initial position. The ability to generate this higher resolution footprint comes from the model's training on uniformly distributed target points. Because the agent has learned to navigate to arbitrary coordinates, this includes the target points outside the reachable range of the HGV.

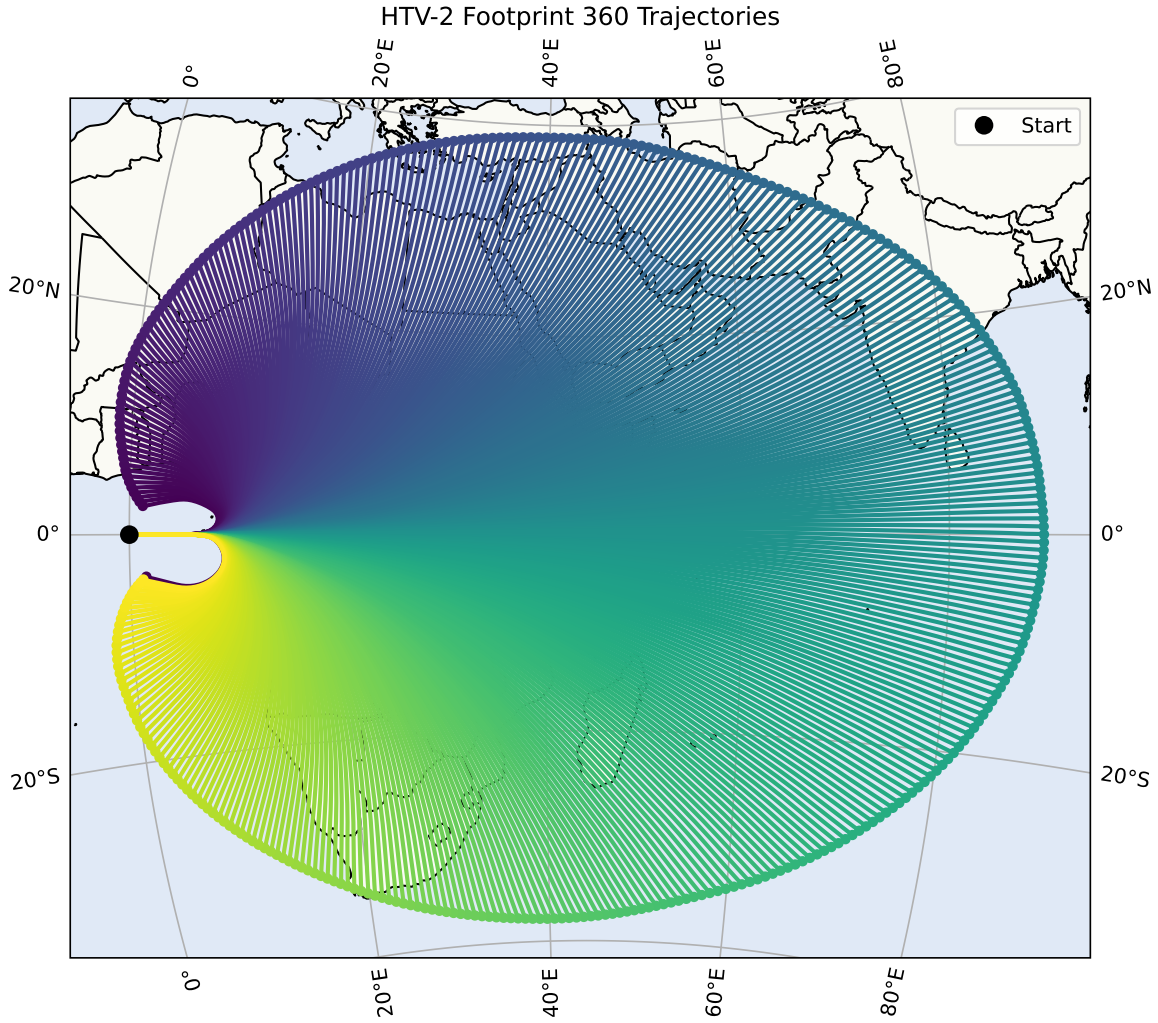


Figure 6.36: HTV-2 footprint generated with 360 maximum-range trajectories in evenly spaced directions from initial position.

6.21. No-Fly Zone Avoidance

No-fly zone (NFZ) avoidance is a challenge for classical trajectory optimization methods. NFZ constraints create exclusion regions in the search space, making gradient based optimizers prone to local minima and requiring manual reformulation for each new configuration. Classical methods also struggle with ensuring continuous constraint satisfaction. Reinforcement learning naturally addresses these limitations through learned avoidance strategies that generalize across all NFZ positions and target

point locations on Earth. The continuous reward-based penalty formulation steers the agent away from violation.

To extend the HGV trajectory optimization framework beyond point-to-point navigation, a no-fly zone (NFZ) avoidance capability was implemented. The trained agent can navigate toward a target point while actively avoiding prohibited cylindrical regions in the airspace. The NFZ implementation is an advancement towards a more realistic operation constraint, where HGVs must respect territorial boundaries or restricted airspace.

The implementation builds upon the existing target point navigation framework by introducing awareness of no-fly zones. During training, both the target point and the NFZ center are randomly generated within reachable range, forcing the agent to learn generalized avoidance behavior.

6.21.1. Implementation Changes

Observation Space

The observation space was expanded from 5 to 7 dimensions by adding two NFZ related observables. These are the normalized bearing minus the heading error and the normalized distance to the NFZ boundary edge. This edge based formulation gives the agent direct information about the proximity of constraint violation and allows to extend the program to variable radii no-fly zones, where the only change is that the radius has to be randomized during training.

NFZ Constraint Penalty

A fixed penalty of -2.0 is applied whenever the vehicle enters the NFZ radius, which gives the agent direct negative feedback for boundary violations. The value of -2.0 was chosen through iterative tuning experiments to be in proportion to the rest of the reward formulation. This penalty works independently from and in addition to the existing constraint penalties and distance-based progress reward. The penalty has a fixed magnitude regardless of how deep into the NFZ it flies, which encourages complete avoidance rather than minimal intrusion.

Randomized Training Scenarios

During training mode, both target point and NFZ position are randomly generated within the reachable range using the uniform distribution over circular areas. The NFZ radius is now fixed at 500 km. Experiments with a varying radius were performed, which also gave promising results.

Limitations

In the case that the NFZ is positioned too close to either the starting position or the target point, successful avoidance becomes physically infeasible for the HGV. The HGV's limited maneuverability prevents sufficiently aggressive maneuvers needed to navigate around obstacles in close proximity. If the NFZ is placed too close, the HGV will violate the NFZ or the aerodynamic constraints.

6.21.2. Discussion of No-Fly Zone Avoidance Results

This section presents trajectory results from the trained NFZ avoidance model, which shows the agent's capability to navigate toward target points while respecting the NFZ. The following cases show performance across different geometric configurations.

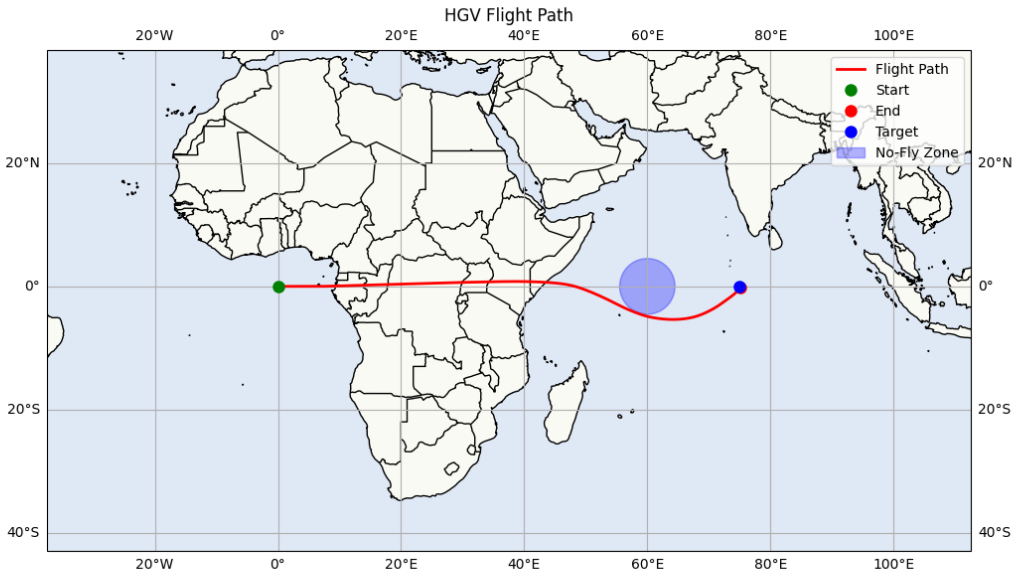


Figure 6.37: NFZ avoidance trajectory with efficient detour routing around NFZ positioned between starting position and target point.

Figure 6.37 shows a successful avoidance of the no-fly zone where the NFZ is placed between the starting point and the target location. The trajectory shows the vehicle making a smooth detour that maintains clearance from the NFZ radius while avoiding excessive deviation to still make it to the target. The agent initiates its avoidance maneuver well before reaching the zone perimeter. After clearing the NFZ, the trajectory naturally re-converges towards the target, which shows that the target point guidance is maintained throughout the avoidance maneuver.

Figure 6.38 shows the model's ability to generate trajectories with NFZ's across different geographic locations and NFZ configurations within a single trained policy. It can be seen that the trajectory shows successful avoidance and target convergence, also with different NFZ placements and target point locations relative to the NFZ.

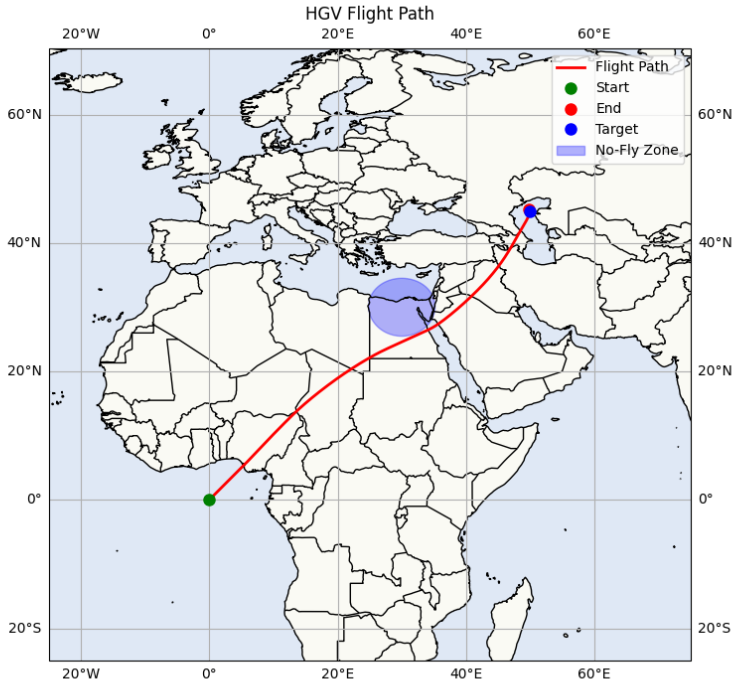


Figure 6.38: HGV trajectory showing NFZ avoidance at different geographic location and different position of NFZ relative to origin and target.

Figure 6.39 shows a scenario in which the target point is positioned adjacent to the NFZ boundary. This configuration creates a challenging situation since the vehicle must navigate around the NFZ while still trying to reach a target located directly behind with minimal clearance.

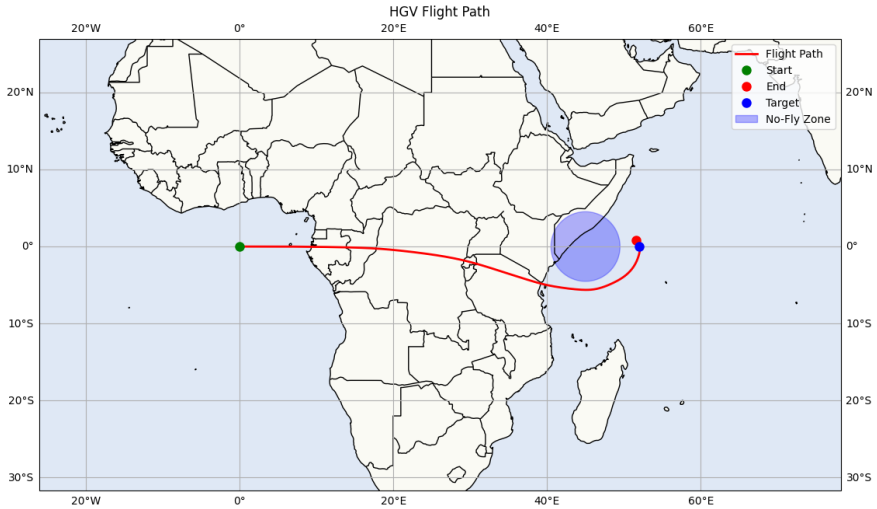


Figure 6.39: Trajectory of the HGV with target point positioned adjacent to NFZ boundary

The trajectory shows that the agent attempts to fly around the NFZ perimeter to approach the target. The flight path shows the vehicle curving around the obstacle and converging toward the target point in the available space between the zone edge and destination. This behavior validates that the policy prioritizes reaching the target, even in tight configurations.

However, this scenario also shows the physical limits of avoidance capability. With the target positioned so close to the NFZ boundary, the required final approach geometry becomes extremely constrained. The HGV's limited turning radius and energy state may not allow achieving both complete NFZ clearance and target point precision. In Figure 6.40 an example is shown of an NFZ zone that is located too close to the initial position of the HGV, where the target point is positioned at $\phi = 0°, \lambda = 60°$. What can be seen here is that the vehicle is not able to make a turn quick enough since it is still in the outer layers of the atmosphere, so it has no choice but to violate the constraint. The target point is located along the equator, but can not be reached since the HGV has lost a lot of energy executing the turn maneuver to avoid the NFZ as much as possible. This is a physically infeasible location of an NFZ.

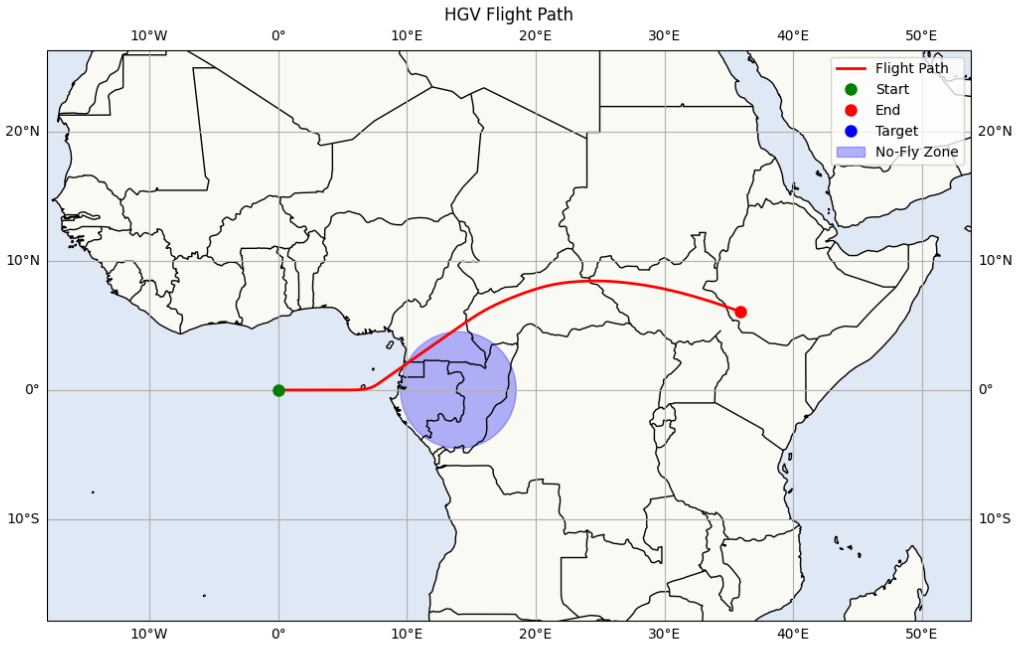


Figure 6.40: NFZ located too close to the initial position which will result in a violation, where the target point is positioned at $\phi = 0°, \lambda = 60°$.

6.22. Limitations and Assumptions

The framework shows effective trajectory optimization and footprint generation, but several modeling assumptions are made. This section examines these limitations and their implications on the accuracy of the model. These assumptions have been explained throughout the document, but will be summarized here again.

6.22.1. 3-DOF Simplifications

The three-degree-of-freedom point-mass formulation neglects rotational dynamics and assumes instantaneous control authority. Vehicle orientation is implicitly defined by the velocity vector and bank angle, but attitude dynamics and stabilization are not represented. While rate limiters restrict changes of $1^\circ/\text{s}$ for angle of attack and $3^\circ/\text{s}$ for bank angle, the model assumes instantaneous control authority. Control surface deflections and associated moments are not modeled.

6.22.2. Environmental Model Limitations

The spherical Earth approximation with constant radius of 6371 km is a simplification. Earth's oblateness causes gravitational acceleration to vary with latitude, which is not governed in the inverse square law formulation used in this work.

All distance computations use great-circle distances derived from spherical trigonometry, which provides the shortest path between two points on a sphere. This neglects Earth's oblate spheroid shape.

The USSA 1976 model assumes standard conditions throughout the flight. Real atmospheric density variations at different altitudes affect the aerodynamic forces and heating rates. The policy has not been exposed to atmospheric uncertainties during training. The model assumes windless conditions.

6.22.3. Aerodynamic and Thermal Assumptions

The aerodynamic database uses interpolation between discrete data points to estimate lift and drag coefficients across all flight conditions. This interpolation approach simplifies the actual aerodynamic behavior due to limited available data for this work. The baseline configuration relies on results at specific altitude, Mach number, and angle of attack combinations, with intermediate values obtained through Delaunay triangulations. This may not capture rapid changes in aerodynamic characteristics caused by shock-boundary layer interactions or flow separation during aggressive maneuvers for example. However, rate limitations are applied which mitigate this risk.

The temperature constraint is only applied at the stagnation point instead of a distribution across the vehicle surface.

6.22.4. Numerical Integration

Trajectory integration uses adaptive Runge-Kutta with a maximum step size of 5 seconds. Control decisions update at fixed 5 second intervals during training, with angle of attack and bank angle held constant between update points while the trajectory state evolves continuously.

6.22.5. Constraint Formulation Simplifications

The load factor and dynamic pressure constraints use simplified mathematical formulations instead of models of the real systems. The load factor constraint uses the total aerodynamic force magnitude divided by the weight, but does not take the structural load distribution throughout the vehicle into account. Real structures have different strength limits for different stresses that can occur at specific locations instead of uniformly across the HGV.

6.22.6. Constraint Checking

Path constraints are evaluated at 5-second intervals corresponding to the control update frequency. This discrete checking introduces a theoretical possibility that a constraint could be violated and afterwards satisfied within a single integration step without detection. However, the physical dynamics of hypersonic flight make such violations highly unlikely in practice.

6.22.7. Practical Value

Despite these simplifications, the framework remains highly effective for trajectory optimization and footprint analysis. These assumptions are standard practices in trajectory modeling and conceptual design studies rather than fundamental limitations that compromise the results.

6.23. Training Time and Learning Outcomes

The training process for the HGV policy provides information on the convergence behavior and the computational requirements of the SAC algorithm. The training metrics, collected throughout the learning process, show the efficiency of the approach and the characteristics of the learned policy.

6.23.1. Training Duration and Computational Cost

The complete training run executed for 7,360,000 timesteps, which is approximately 3 hours of wall-clock time. This relatively short training duration was achieved through the parallelized training architecture that utilizes 16 simultaneous environments on the CPU, with neural network updates performed on the GPU. The parallel environment configuration allowed for efficient data collection, since each environment step provides 16 separate experiences to the replay buffer.

Early stopping was triggered after 20 consecutive evaluations without improvement in mean episode reward, which prevents unnecessary computational time beyond the point where the policy has converged. The best performing model was saved at timestep 7,360,000 indicating that the policy achieved the peak performance near the end of the allocated training time of 10,000,000 timesteps, before early stopping criteria were met.

6.23.2. Episode Performance Evaluation

The rollout mean reward tracks the performance during actual training interactions with stochastic action sampling and shows substantial improvement during early training followed by high variability throughout the remaining timesteps, as shown in Figure 6.41.

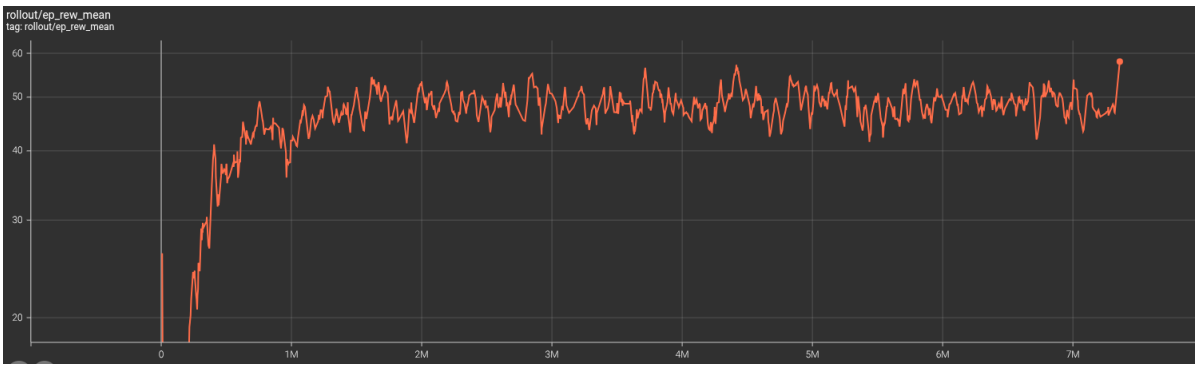


Figure 6.41: Mean rollout reward, showing quick early learning followed by high variability throughout training.

Initial performance at the start of training gave mean rollout rewards in the low 20s. During the first million timesteps, rapid learning occurred as the agent discovered navigation strategies and constraint avoidance. The reward increased during this period, climbing from the low 20s to approximately 45-50 by 1,000,000 timesteps.

From 1,000,000 timesteps onward to the end of training at 7,360,000 timesteps, the rewards entered a plateau phase with variability, oscillating between 45 and 55. The variance did not decrease in later training stages. The final smoothed rollout reward at termination, which is a moving average that filters out noise to show the general trend, was 51.67.

The high variance throughout training show the difficulty of the HGV trajectory optimization problem. This variance is due to several factors. The training mode randomizes initial conditions, including initial latitude, initial heading, and target point selection. Some randomized scenarios are easier for the agent to learn, such as when the target is in line with the initial heading. However, other scenarios are more

challenging, such as when the target lies behind the initial heading, which requires aggressive banking maneuvers and careful energy management.

Importantly, SAC's entropy maximization objective intentionally maintains this stochastic exploration throughout the entire training process. Unlike algorithms that converge to fully deterministic policies, SAC continues to sample actions from a probability distribution even after discovering effective strategies. The automatic entropy coefficient tuning adjusts the exploration to exploitation trade-off to maintain a level of policy randomness. This means that even in later training stages, when the agent has learned good baseline strategies, it continues to explore alternative action sequences. Some of the exploratory actions lead to suboptimal outcomes, which shows in the reward variance, but this exploration is essential to discover potentially better strategies and to prevent premature convergence to local optima.

6.23.3. Episode Length

Mean episode length during rollout, which represents the number of simulation steps from initial conditions at 100 km altitude to terminal altitude of 5 km, increased substantially during early training before stabilizing, as can be seen in Figure 6.42.

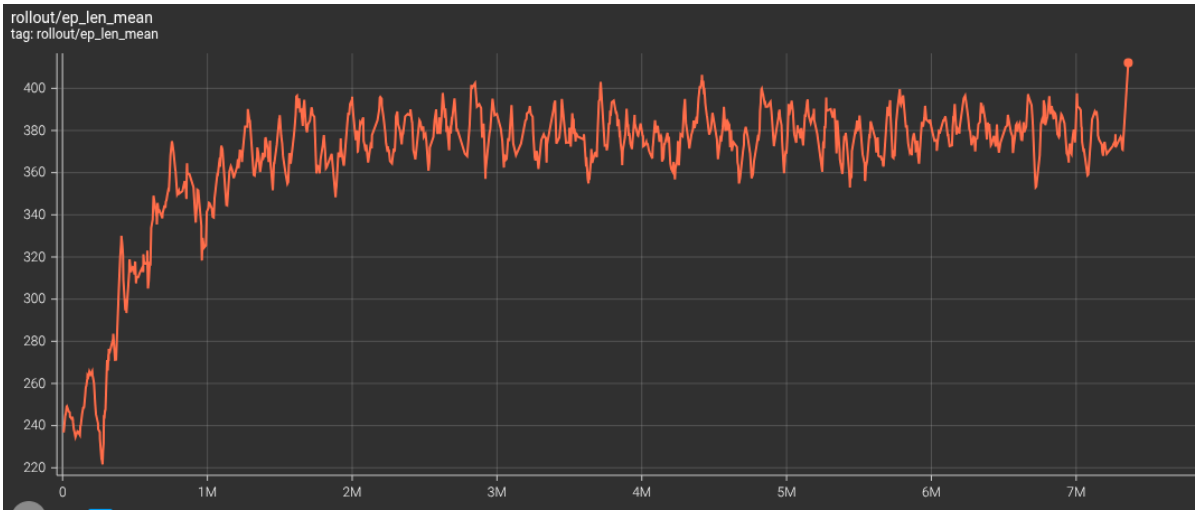


Figure 6.42: Mean episode length during training rollouts, which shows rapid initial increase followed by stable plateau.

Early in training, the episodes terminated on average after 250 steps since the untrained policy did not manage to find energy efficient trajectories that will extend the length of the trajectory. With a 5 second timestep, this corresponds to an average flight time of 1250 seconds. During the first million steps, the episode lengths rapidly increased to around 400 steps, indicating that the agent learned to sustain flight longer by better energy management.

From approximately 1,000,000 timesteps onward, mean episode length stabilized around 380 to 400. The final smoothed episode timesteps at termination was 389 steps. Given the 5 second timestep used in the simulation, this corresponds to 1945 seconds of simulated flight time per episode. The stable mean suggests that the agent is no longer learning fundamentally different approaches that would shift this distribution significantly. Training continued through this plateau phase since the early stopping mechanism evaluates the policy every 10,000 timesteps by running 5 deterministic evaluation episodes, and only terminates training after 20 consecutive evaluations show no improvement in mean reward. This allowed the training to continue searching for potential improvements even after episode lengths had stabilized.

6.23.4. Loss Function Convergence

In SAC, the agent consists of two neural networks that work together: the actor decides which actions to take, while the critic predicts how good those actions will be in terms of future rewards. The actor loss

measures how well the actor is learning to choose high reward actions based on the critic's guidance, while the critic loss measures how accurately the critic can predict future rewards. This was explained in more detail in Section 6.16.2. Both losses should decrease and stabilize as the learning progresses, indicating that the networks have learned their roles effectively. The actor loss is shown in Figure 6.43.

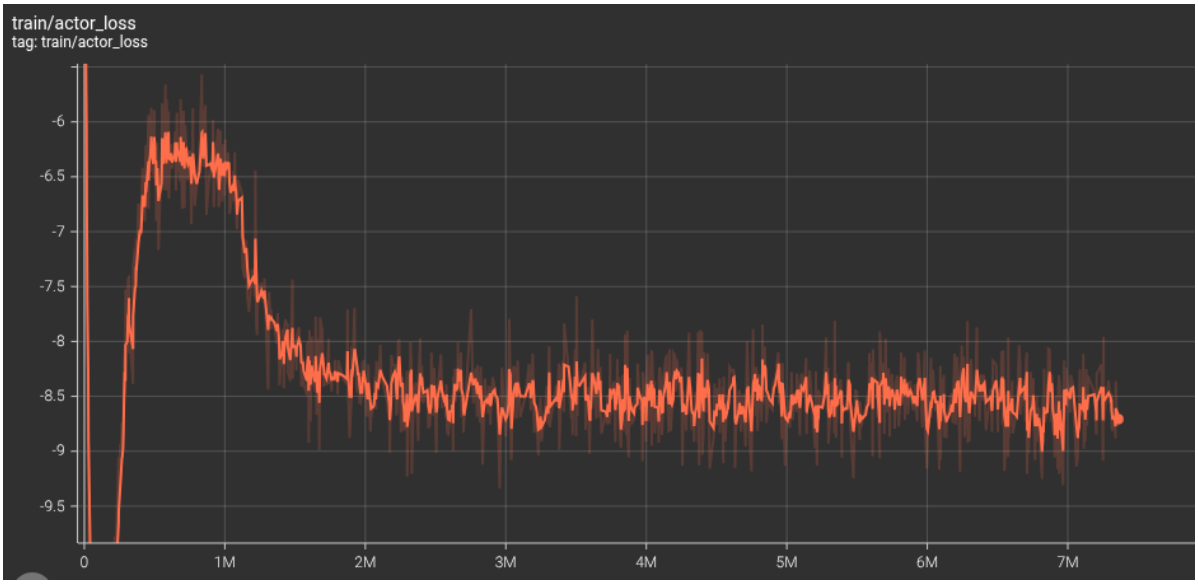


Figure 6.43: Actor loss during training, showing quick initial learning followed by stable convergence.

It can be seen that the actor loss has a rapid initial descent from approximately -6.2 to approximately -8.5 in the first two million time steps. The more negative the actor loss becomes, the better it is performing. This decrease indicates that the actor has discovered increasingly better actions. It learned to select bank angle and angle of attack combinations that the critic predicts will lead to higher cumulative rewards. After this initial learning phase, the actor loss stabilizes around -8.5 to -8.7. This indicates that the policy network parameters are in a stable configuration that represents a general purpose control strategy. The reward variance comes from two factors. There is a diversity of scenarios that are encountered and the SAC's entropy driven exploration, which intentionally maintains the stochastic action sampling, which leads to suboptimal actions from time to time even after discovering effective strategies.

The critic loss is shown in Figure 6.44. This measures the temporal difference error, which is the difference between what the critic predicted would happen and what actually happened.

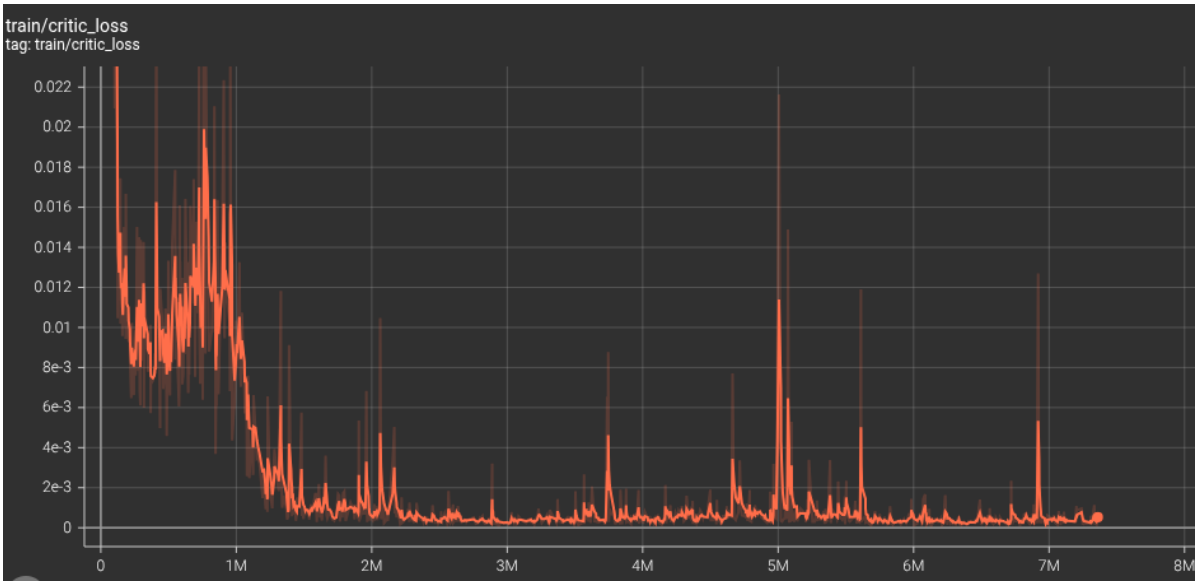


Figure 6.44: Critic loss during training, showing reduction from initial spikes and convergence to low value.

The critic loss shows initial spikes during the first one million timesteps, before rapidly declining. These early spikes show the critic network's early struggle to learn accurate Q-value predictions for a rapidly changing policy. Initially, the critic's predictions are inaccurate because it has limited experience with the consequences of different actions. As training progresses and the replay buffer accumulated more diverse experiences, the critic loss decreases to small values. This low critic loss indicates that the value function approximation became accurate, providing reliable guidance for policy weight updates.

The convergence of both actor and critic losses to stable, low values confirms that the SAC algorithm successfully learned a control policy. The stable losses indicate that the neural network training has converged to a consistent control strategy.

6.23.5. Learning Dynamics Interpretation

The training results confirm that reinforcement learning, specifically the SAC algorithm, can successfully learn complex hypersonic control policies within reasonable computational efforts. The training time of approximately 3 hours, combined with stable network convergence, demonstrates the practical feasibility of this approach for HGV trajectory optimization applications.

6.23.6. Computational Hardware

The training was performed on a high-performance workstation provided by NLR. The workstation has the following specifications:

- **CPU:** Intel Xeon Gold 6348 @ 2.60 GHz, 56 cores
- **RAM:** 503 GB
- **GPU:** NVIDIA A100-PCIE-40GB
- **Operating System:** Red Hat Enterprise Linux 8.10
- **Software:** Python 3.12, CUDA 12.8, stable-baselines3, PyTorch 2.8

The use of GPU acceleration for neural network updates combined with CPU-based parallel environment simulation across 16 processes enabled efficient training, completing 7.36 million timesteps in approximately 3 hours.

7

Monte Carlo Simulation

To validate the reinforcement learning model, a large-scale Monte Carlo simulation was performed with 100 million trajectory evaluations. This validation confirms that the RL-generated footprint represents the true reachable domain by comparing it with a random exploration of the control space.

Unlike the earlier Monte Carlo simulations presented in Section 5.3.1, which used discrete control updates at fixed intervals, this validation uses a continuous random sampling approach throughout the control space. Both the angle of attack and the bank angle are randomly sampled, exploring the full range of control strategies without bias toward any particular flight pattern. This approach ensures that if reachable regions exist beyond the RL footprint, they will be discovered with high probability given sufficient sampling.

The Monte Carlo methodology differs from the RL approach. While the RL agent learns optimal control policies through reward maximization, the Monte Carlo method simply generates random control sequences and records which terminal positions are achievable while satisfying all constraints. This independence between methods strengthens the validation, since agreement between them demonstrates that the RL policy has genuinely discovered the boundaries of the feasible domain rather than converging to a local optimum within a larger reachable set.

7.1. Implementation

7.1.1. Random Control Strategy

The Monte Carlo simulation uses a random flight controller that periodically samples new control inputs from uniform distributions. The angle of attack is sampled from the range $[-4^\circ, 60^\circ]$, while bank angle is sampled from $[-90^\circ, 90^\circ]$, just as the action space in the RL model. Control updates are also randomized, from a value of 5 seconds up to 1200 seconds, to allow the Monte Carlo simulation to reach points closer to maximum range and minimum range. If a time step of 5 seconds is used, just as in the RL model, the probability of high-range trajectories becomes too small since for the maximum range a bank angle of 0° and angle of attack of 12° is desired.

To prevent unrealistic control behavior, rate limiters restrict the maximum change between consecutive control values. The angle of attack rate is limited to $\pm 1^\circ/s$, while the bank angle rate is limited to $\pm 3^\circ/s$. These rate limits ensure that randomly sampled control sequences remain within physical capabilities and have the same values as those used in the RL model.

The simulation continues until the vehicle descends below 5 km altitude, at which point the terminal latitude and longitude are recorded. Throughout the whole flight, constraint margins for load factor, dynamic pressure, and stagnation point temperature are continuously monitored. Any trajectory that violates any of these constraints is immediately terminated and discarded, which means that only endpoints that satisfy the constraints contribute to the footprint.

7.1.2. Spatial Thinning Algorithm

Direct storage of all 100 million terminal positions would require excessive memory and computational resources, since many landing positions cluster very close together, providing redundant information about the footprint boundary. Therefore, a spatial thinning algorithm is used that filters results in real-time during simulation.

Hexagonal Grid Partitioning

The algorithm uses H3, which is a discrete global grid system developed by Uber Technologies which divides the Earth's surface into a hierarchical hexagonal grid [95]. H3 supports sixteen resolutions, with each finer resolution subdividing cells into areas one-seventh of the size of the previous resolution [96]. Resolution 5 was selected for this application, which divides the globe into approximately 2,016,842 hexagonal cells, each with an average edge length of 8.5 km and an average area of 252 km^2. A representation of the H3 grid on Earth is shown in Figure 7.1

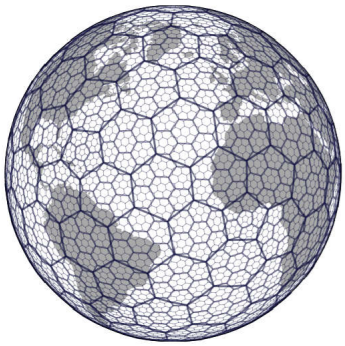


Figure 7.1: Representation of H3 grid on Earth [97].

.

The hexagonal structure was chosen over square grids for two reasons. First, hexagons have all six neighbors at an equal distance from the center, while square grids have four neighbors at one distance and four diagonal neighbors at a greater distance. The uniform spacing is such that the 8.5 km distance is applied consistently in all directions. Second, hexagons maintain more uniform cell areas across different latitudes compared to rectangular latitude-longitude grids. These become distorted near the poles.

A mathematical constraint is present when tiling a sphere. It is impossible to cover a spherical surface using only hexagons. The H3 system addresses this by including exactly 12 pentagonal cells at fixed locations distributed around the globe [96]. This is according to Euler characteristics. This pattern can also be observed on a football, which has 12 pentagonal patches surrounded by hexagons.

Algorithm Implementation

The filtering algorithm operates throughout the simulation time. When a trajectory ends, the terminal latitude and longitude are converted to an H3 cell identifier using a function from H3, which returns a unique 64-bit hexadecimal that represents the hexagonal cell containing the position. A hash set data structure tracks which cell identifiers have been encountered in previous trajectories. If the computed cell identifier is new, the terminal position is stored and the cell is marked as occupied. If the cell identifier already exists, the terminal position is discarded since a point is already in that $252km^2$ region.

This algorithm has several advantages. First, it eliminates the need to store all 100 million positions in memory that will not contribute to the footprint area. Second, it is computationally efficient, since only hash set lookups are required which are computationally fast. The result is a compact representation of the reachable area, while reducing data volume by more than 99%.

The simulation uses multiprocessing parallelization to reduce computational time. Each worker process has its own instance of the HGV dynamics model and random number generator.

7.2. Timestep Variation

This Monte Carlo simulation varies the timestep across different runs to maximize coverage of the search space. The control update time steps range from 5 to 1200 seconds, with values distributed throughout this interval. The different time steps affect how often the control inputs can change during flight. With a short timestep, for example 5 seconds, allows the HGV to change direction very frequently and explore aggressive maneuvering. With a long timestep, controls only change every once in a while, forcing the vehicle to maintain the same control settings for much longer periods which is required for long-range trajectories. By testing both extremes and everything in between, the simulation ensures both rapidly changing flight paths and more smooth gliding flights. By exploring the full spectrum, the simulation avoids biasing results toward any particular control frequency characteristic.

7.3. Results and Comparison

The Monte Carlo simulation produced 100 million trajectory evaluations taking 4000 CPU hours divided over 48 working cores. The distribution is shown in Figure 7.2, and looks visually identical to the footprint generated by the RL model in Figure 6.26.

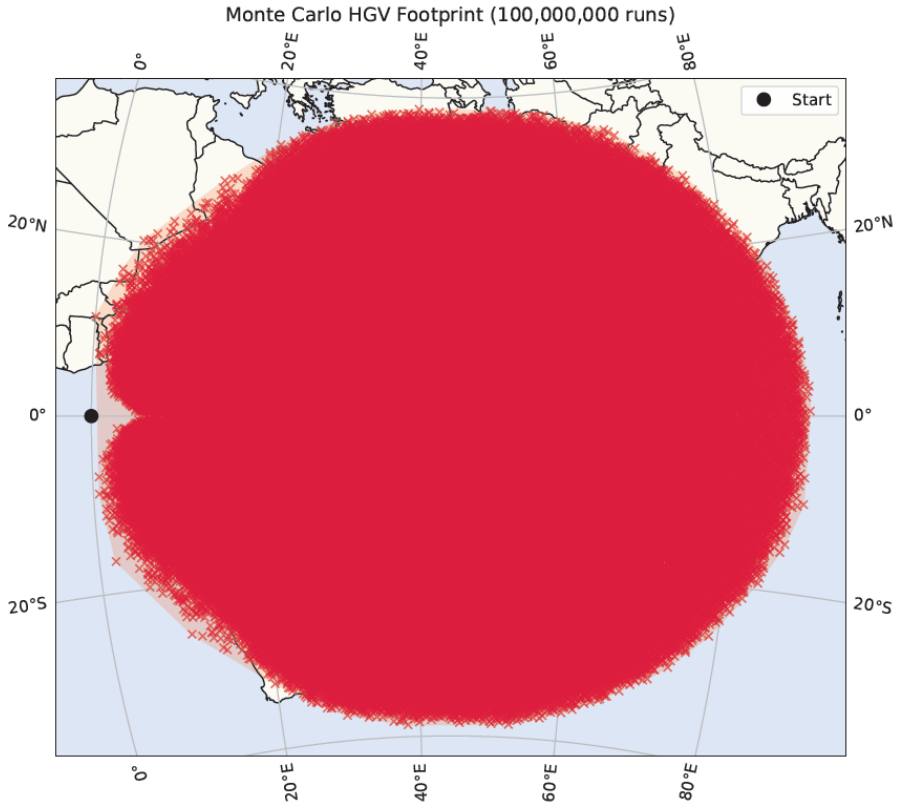


Figure 7.2: Monte Carlo Simulation with 100 million trajectories.

The quantitative metrics are given in Table 7.1. The comparison shows excellent agreement between the two methods, with the RL footprint slightly exceeding the Monte Carlo footprint in total area by 5.1%. This difference is consistent with the expected behavior. The RL agent is trained specifically to maximize range and explore all target directions systematically and discovers slightly more efficient trajectories than random control sampling. The maximum range value differs by 0.38%, where the RL has found a slightly further range. The maximum cross-track shows the RL policy exceeding the Monte Carlo result by 2.3%, differing approximately 98 km only. The optimized RL policy has found a slightly better cross-range control than the random Monte Carlo with 100 million runs. The RL agent has learned to maintain optimal bank angle profiles that maximize lateral displacement while still preserving energy efficiency and respecting constraints.

Table 7.1: Comparison of Monte Carlo and reinforcement learning footprint metrics

Metric	Monte Carlo	RL Policy	Difference
Footprint Area (million km^2)	64.8	68.1	+5.1%
Maximum Range (km)	9885.0	9923.0	+0.38%
Maximum Cross-Track (km)	4332.3	4430.6	+2.3%

7.4. Validation Interpretation

The close agreement between the Monte Carlo and RL footprints provides validation of the reinforcement learning approach. If the RL policy had converged to a suboptimal local solution, the Monte Carlo footprint would extend substantially beyond the RL boundary, since random sampling would eventually discover superior trajectories, especially with 100 million runs. The fact that the RL footprint matches or even slightly exceeds the Monte Carlo footprint in all metrics demonstrates that the learned policy discovered the boundaries of the feasible domain.

The 5.1% larger RL footprint area can be understood with the exploration versus exploitation trade-off. The Monte Carlo approach performs pure exploration, sampling the control space uniformly without learning from past trajectories. This guarantees coverage, but does not concentrate sampling in promising regions. The RL approach balances exploration and exploitation, using experience to identify high performing control strategies and refine them. The result is a policy that consistently achieves performance near the physical limits of the HGV.

The comparison of computational efficiency also favors the RL approach. The 100 million Monte Carlo trajectories in only one direction required several days of computation to achieve sufficient coverage for footprint validation. In contrast, once trained, the RL agent generates a complete footprint in seconds, anywhere on Earth.

The Monte Carlo simulation confirms, just as the specific target point guidance, that locations within the footprint borders are reachable without constraint violations.

8

Sensitivity Analysis

8.1. Introduction

A sensitivity analysis quantifies how variations in input parameters affect the system performance. Here it is the goal to identify which parameters most significantly influence mission outcomes.

In the context of reinforcement learning-based guidance, sensitivity analysis reveals whether the learned policy can generalize beyond the training distribution. During training, the policy experienced various combinations in latitude, heading and target direction, but was trained with initial conditions of 100 km altitude, 6000 m/s velocity and flight path angle of -3°. The sensitivity analysis therefore tests the policy's ability to maintain performance when these previously fixed parameters deviate from their nominal values. A robust RL policy should maintain reasonable performance instead of completely failing when operating in regions of state space it has not explicitly encountered. It should be noted that if a specific range of parameter values is operationally required, the model can be re-trained with those parameters included in the training distribution, which would give accurate performance. The current analysis shows an extrapolation to see the robustness of the learned policy.

This analysis examines two categories of parameters. The first consists of the initial flight conditions, namely the altitude, velocity, and flight path angle. Here the effect on the footprint size is compared and the effect on the constraints during flight. The trajectory characteristics are generated by the learned policy responding to the perturbed initial conditions, demonstrating how the policy adapts its controls strategy. For comparison, a baseline trajectory is included in the trajectory characteristic plots. This baseline uses fixed control inputs with an optimal L/D angle of attack of 12° and a bank angle of 0°, representing a non-adaptive control strategy. By comparing the learned policy trajectories against this baseline, the effectiveness of the adaptive control approach can be seen.

The second category includes the vehicle properties, namely the mass and surface area, which can vary. Again, it must be noted that for this model, it is easy to add a varying mass or surface area to the training, which will give accurate results.

8.2. Varying Initial Flight Conditions

8.2.1. Flight Path Angle Sensitivity

In Figure 8.1 the footprint is shown for varying the initial flight path angle with $\pm 0.5^\circ$, resulting in initial flight path angles of $\gamma_0 = -3.5^\circ, -3.0^\circ$ and -2.5°. It can be seen that the difference between the three footprints is minimal, where separation can be seen in the maximum and minimum ranges.

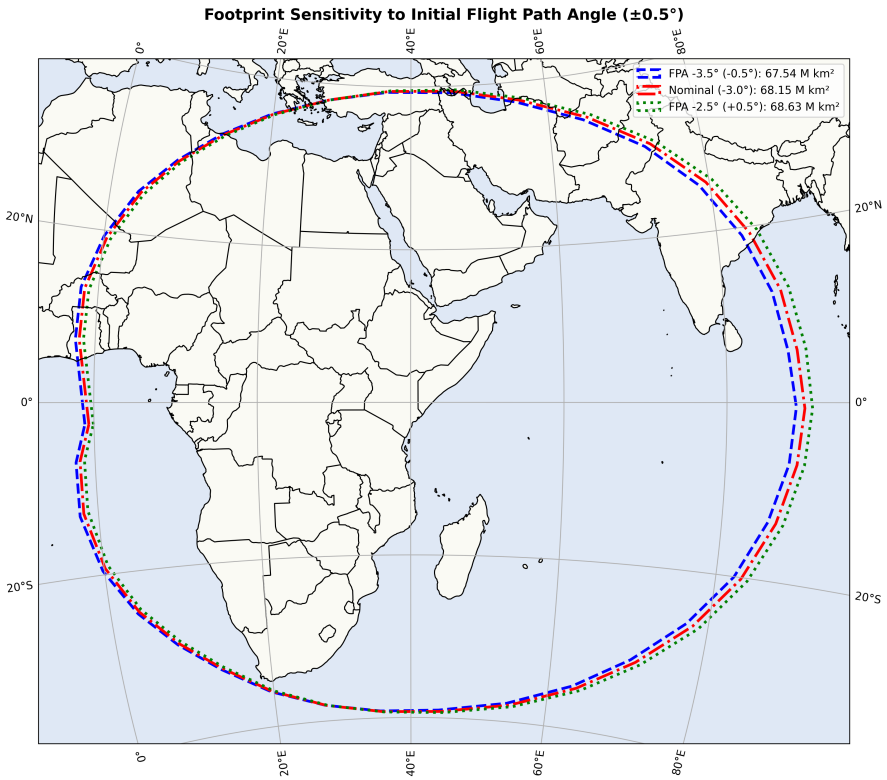


Figure 8.1: Footprint sensitivity to initial flight path angle ($\pm 0.5°$).

By examining the footprint shapes in Figure 8.1 the overall circular shape is preserved across all flight path angle variations. The three cases maintain similar radial extents in all directions, and there is no obvious asymmetry or distortion. It can be seen that the range is slightly increased for shallower initial flight path angles, but for achieving minimum range, greater flight path angles result in more range in reversed direction from the initial heading. The policy appears to successfully complete trajectories in all directions for different flight path angles. This can also be seen from the quantitative data in Table 8.1, where the footprint size is almost similar. The initial flight path angle does not have a significant influence on the total flight envelope of the HGV.

Table 8.1: Flight Path Angle Sensitivity Results

Flight Path Angle	Footprint Area	Absolute Change	Relative Change
$-3.5°$ ($-0.5°$)	$67.54 \cdot 10^6$ km^2	$-0.61 \cdot 10^6$ km^2	-0.9%
$-3.0°$ (nominal)	$68.15 \cdot 10^6$ km^2	-	-
$-2.5°$ ($+0.5°$)	$68.63 \cdot 10^6$ km^2	$+0.48 \cdot 10^6$ km^2	$+0.7\%$

Figure 8.2 shows the trajectory characteristics generated by the learned policy for different initial flight path angles, with a baseline trajectory using fixed controls ($\alpha = 12°$, $\sigma = 0°$) included for comparison. All three learned policy cases follow similar skip glide patterns with synchronized altitude oscillations, showing that the policy maintains consistent control behavior across these variations. It can be seen that the loads are all the highest for the steeper initial flight path angle and the lowest for the shallowest flight path angle.

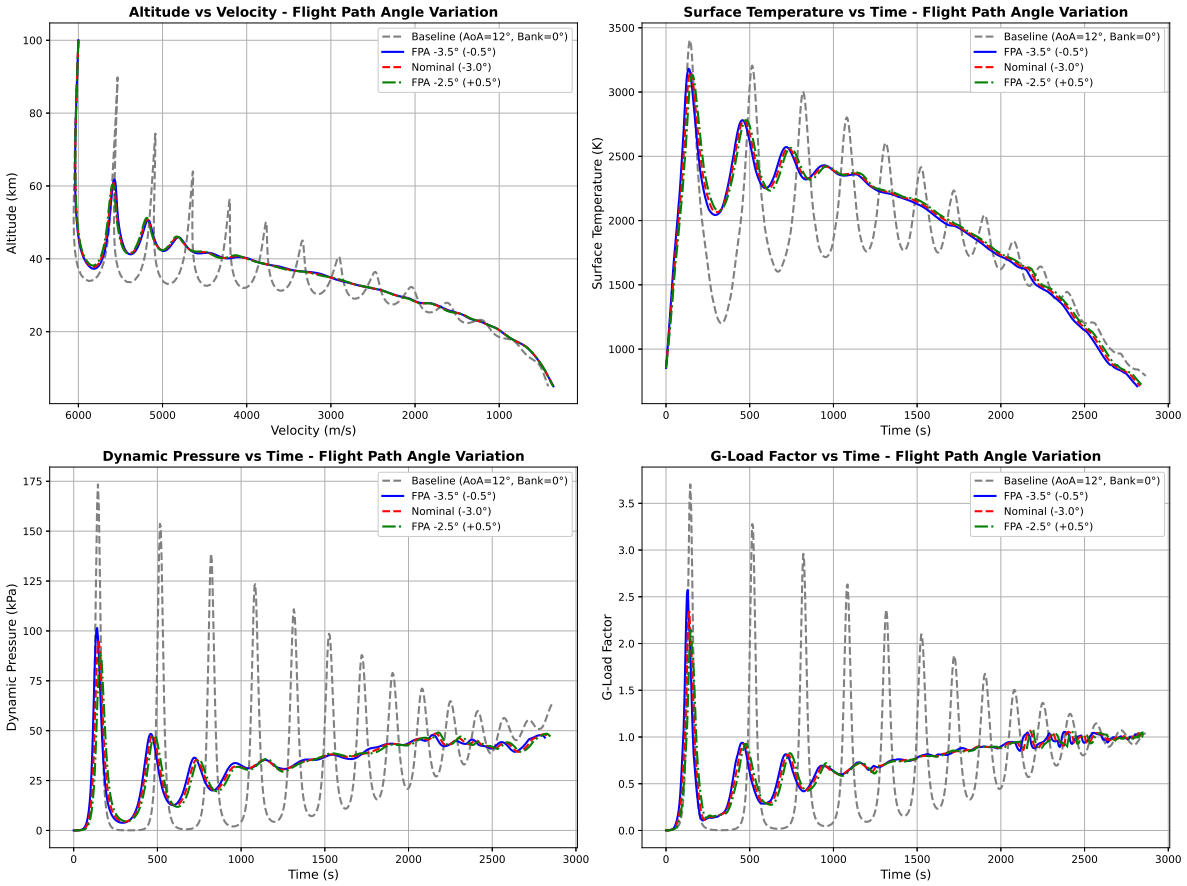


Figure 8.2: Trajectory characteristics for initial flight path angle variations showing altitude vs velocity, dynamic pressure vs time, surface temperature vs time and g load factor vs time for $\gamma_0 = -3.5°, -3.0°, -2.5°$ and baseline trajectory.

8.2.2. Velocity Sensitivity

Velocity perturbations produced the strongest impact on the footprint area, as can be seen in Table 8.2. The reduced velocity case at 5900 m/s gave a -8.8% reduction in footprint area, while the increased velocity case at 6100 m/s gave a footprint area increase of 9.0%.

Table 8.2: Velocity Sensitivity Results

Initial Velocity	Footprint Area	Absolute Change	Relative Change
5900 m/s (-100)	$62.16 \cdot 10^6$km^2	$-5.99 \cdot 10^6$ km^2	-8.8%
6000 m/s (nominal)	$68.15 \cdot 10^6$ km^2	-	-
6100 m/s ($+100$)	$74.27 \cdot 10^6$ km^2	$+6.12 \cdot 10^6$ km^2	$+9.0\%$

Figure 8.3 shows clear separation between the three footprint boundaries, with proportional scaling that maintains the same footprint shape. Despite the substantial area differences, all three cases produce symmetric footprints without directional bias. It can be seen that the higher initial velocity, which is equivalent to a higher energy state, will result in a larger footprint area compared to the case with a lower initial velocity.

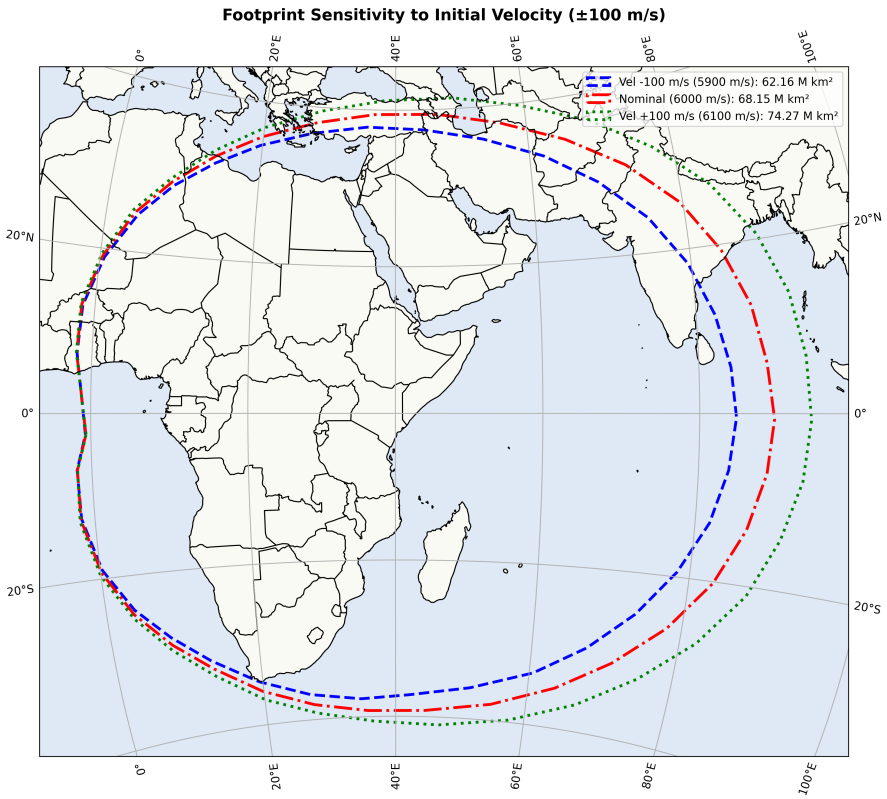


Figure 8.3: Footprint sensitivity to initial velocity ($\pm 100 \frac{m}{s}$).

Figure 8.4 shows the trajectory characteristics generated by the learned policy for different initial velocities, with the baseline fixed-control trajectory included for comparison. The analysis reveals how initial energy state significantly influences the flight behavior and constraint loading.

A counterintuitive but physically important result is observed. The lower initial velocity case (5900 m/s) experiences the highest peak dynamic pressure and g-load, while the higher velocity case (6100 m/s) shows lower mechanical loads despite having more kinetic energy. This lower velocity trajectory cannot generate enough lift at high altitude, so it sinks until density increases enough to restore the required lift. The vehicle with lower velocity must penetrate deeper into the atmosphere during each skip phase to generate sufficient lift for trajectory control. Since atmospheric density increases exponentially with decreasing altitude, and dynamic pressure is defined as $q = \frac{1}{2}\rho V^2$, the combination of higher density at lower altitudes dominates over the quadratic velocity term in this case. Similarly, g-load, which results from aerodynamic forces, increases with dynamic pressure and therefore shows the same result.

The strong velocity dependence reflects the amount of energy that is given. The kinetic energy is defined as $\frac{1}{2}mV^2$, where a small increase in velocity will result in a larger increase in energy. Since achievable range depends on the energy, velocity directly controls footprint area through the quadratic energy relationship.

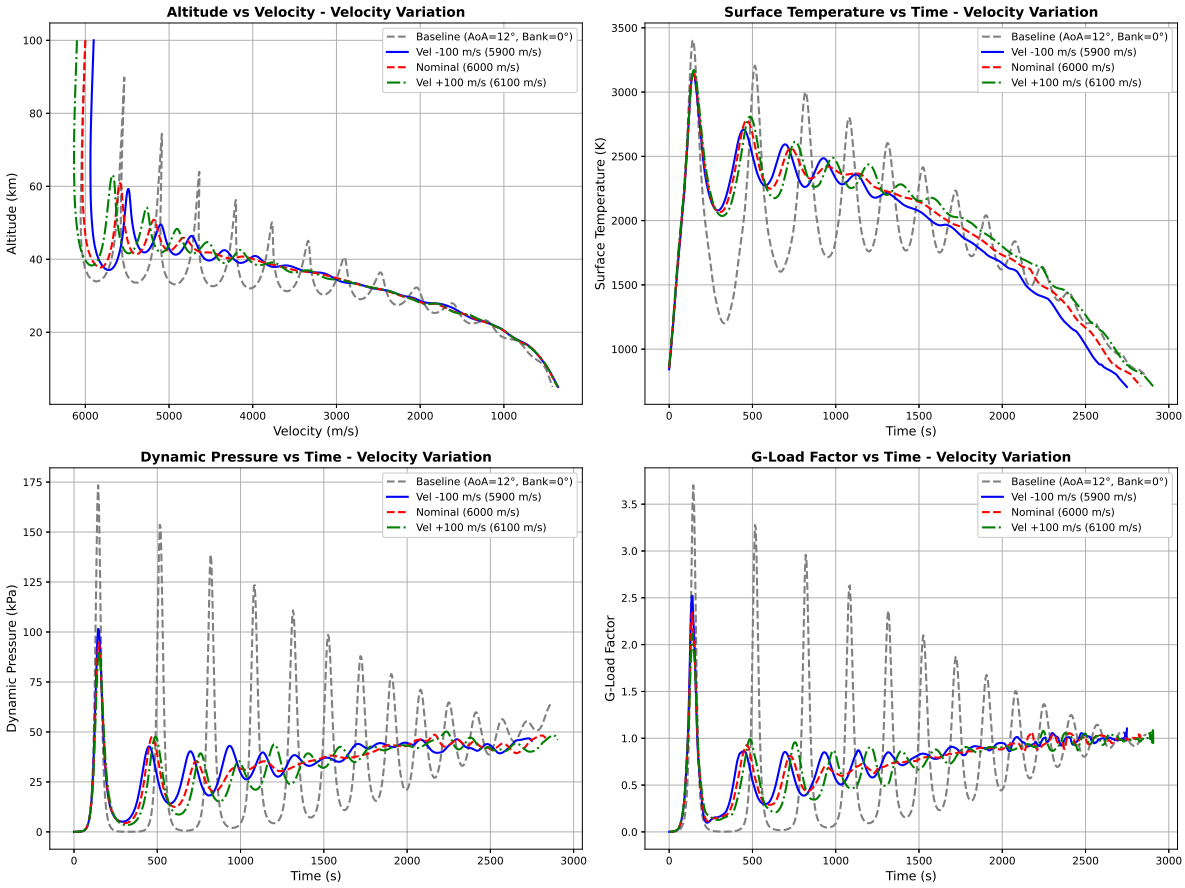


Figure 8.4: Trajectory characteristics for initial velocity variations showing altitude vs velocity, dynamic pressure vs time, surface temperature vs time and g load factor vs time for $V_0 = 6100, 6000, 5900$ m/s and baseline trajectory.

8.2.3. Altitude Sensitivity

Altitude variations did not produce significant changes in the size of the footprint, as shown in Table 8.3, where the relative change is lower than a percent.

Table 8.3: Altitude Sensitivity Results

Initial Altitude	Footprint Area	Absolute Change	Relative Change
90 km (-10 km)	$68.16 \cdot 10^6$ km^2	$+0.01 \cdot 10^6$ km^2	$+0.01\%$
100 km (nominal)	$68.15 \cdot 10^6$ km^2	-	-
110 km ($+10$ km)	$68.11 \cdot 10^6$ km^2	$-0.04 \cdot 10^6$ km^2	-0.06%

Figure 8.5 shows visually that the footprints are very close together, with subtle differences. The higher altitude case achieves slightly greater range in the flight direction, where it benefits from the additional potential energy that converts to kinetic energy during descent. However, when examining the range in reverse direction, the lower starting altitude gives a larger range in this direction.

The footprint area of the three cases is approximately the same. The slight advantage in backward reach for a lower altitude gives a marginally larger total reachable area. However, these differences remain below 0.1% and will not have practical operational significance.

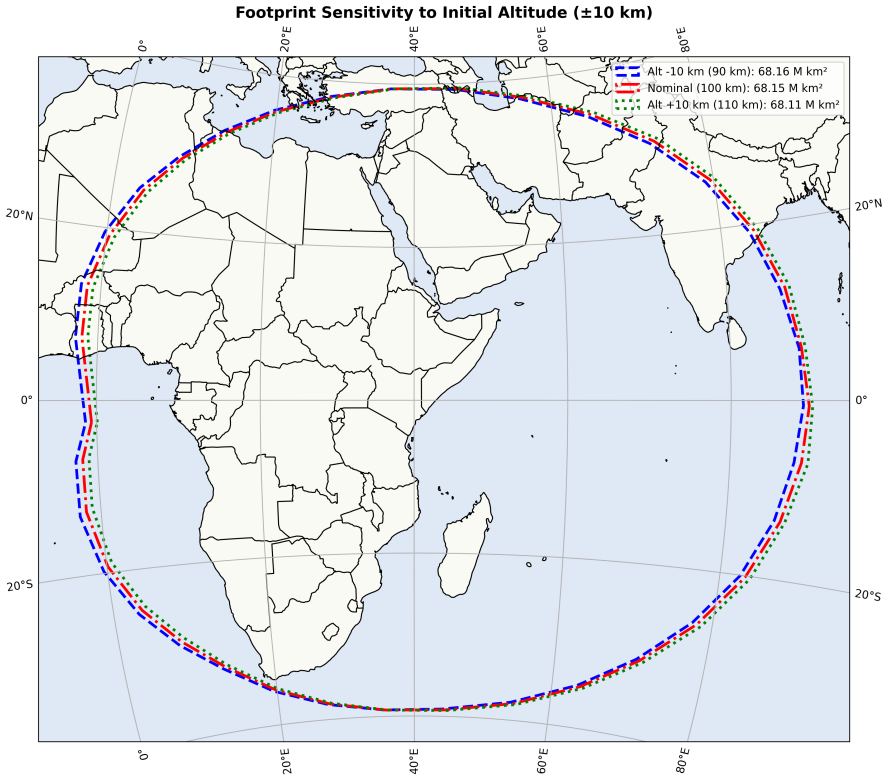


Figure 8.5: Footprint sensitivity to initial altitude ($\pm 10km$).

Figure 8.6 shows the trajectory characteristics generated by the learned policy for different initial altitudes, with the baseline fixed-control trajectory included for comparison. An interesting result is observed: the higher altitude case (110 km) experiences slightly higher g-load and dynamic pressure peaks compared to the lower altitude case (90 km). This is opposite to the velocity sensitivity results where higher energy produced lower loads.

This occurs because, although all trajectories begin with the same initial flight-path angle, a higher starting altitude changes the early descent dynamics. At 110 km the vehicle spends more time in near-vacuum, where lift and drag are negligible and gravity is the dominant force. During this ballistic fall, the vertical velocity component increases while the horizontal component remains almost unchanged, which causes the flight-path angle to become slightly more negative before the atmosphere becomes dense enough to generate lift. In contrast to this, the 90 km case encounters the atmospheric density earlier, which allows aerodynamic forces to counteract gravity sooner and prevent the additional steepening. As a result, the higher-altitude trajectory dips deeper during the first skip, encountering higher density, and therefore slightly higher dynamic pressure and g-load, although the differences remain small.

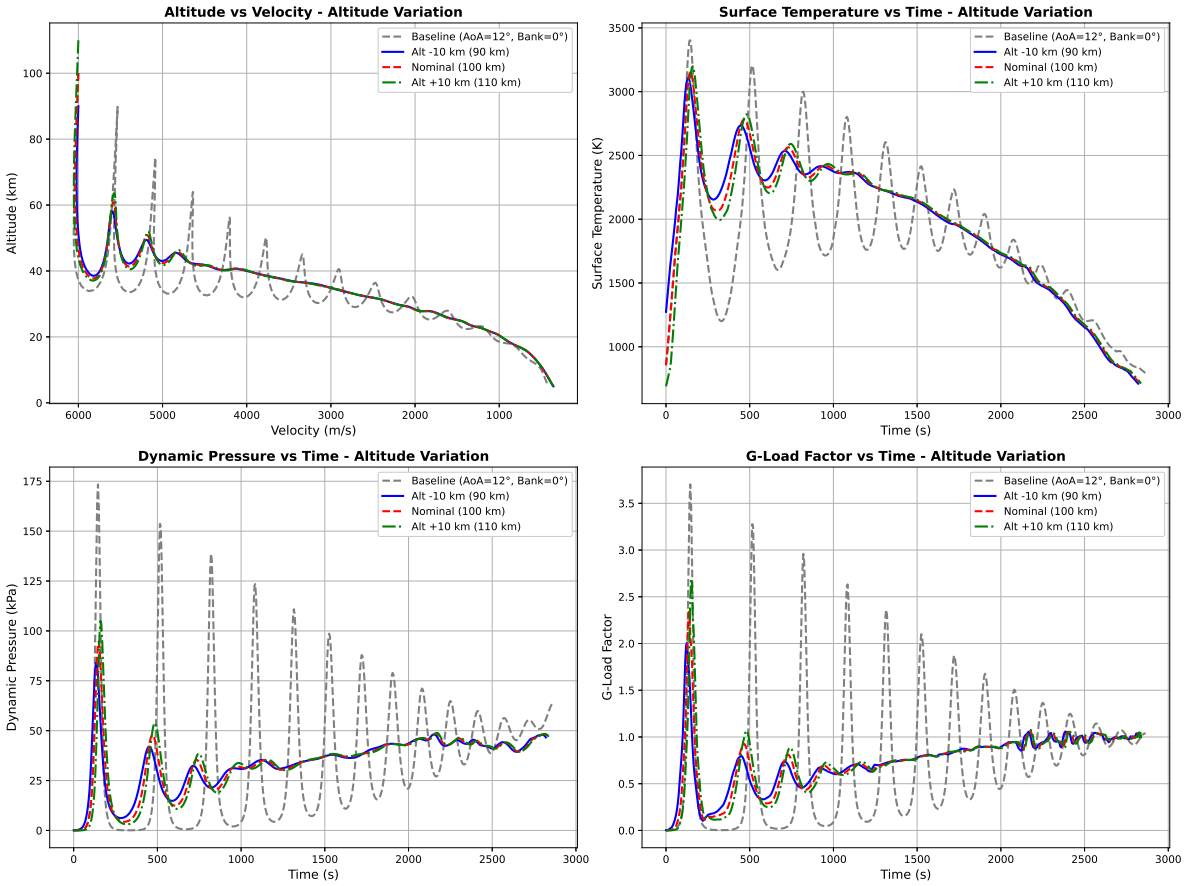


Figure 8.6: Trajectory characteristics for initial altitude variations showing altitude vs velocity, dynamic pressure vs time, surface temperature vs time and g load factor vs time for $h_0 = 110, 100, 90$ km and baseline trajectory.

8.3. Varying Vehicle Parameters

The second category of sensitivity parameters examines the vehicle properties, specifically the mass and surface area. This analysis is to assess the robustness of the learned RL policy when vehicle parameters deviate from the nominal values.

It should be noted that there is a limitation to this sensitivity analysis. The aerodynamic database used for this analysis was generated for the baseline vehicle configuration, with a nominal mass of 2184 kg and a surface area of 4.4 m^2. When the mass or surface area are varied while using the same aerodynamic coefficients, the aerodynamic interpolation is inconsistent. Despite this limitation, the analysis provides valuable insight into the sensitivity of the guidance policy to parameter variations. For operational applications requiring accurate performance with varied mass or surface area, the aerodynamic database would need to be generated with extra interpolations over mass and surface area.

8.3.1. Mass Sensitivity

Vehicle mass was varied by $\pm 10\%$ from the nominal value of 2184 kg, resulting in test cases of 1965 kg, 2184 kg and 2402 kg. Figure 8.7 shows the footprint boundaries for these three mass configurations.

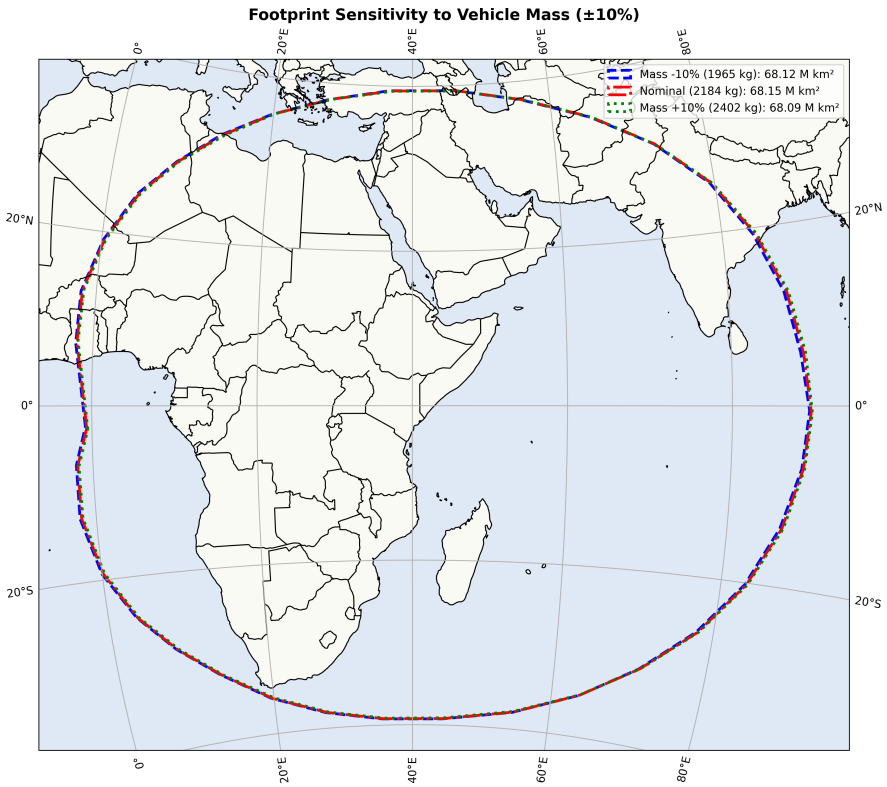


Figure 8.7: Footprint sensitivity to vehicle mass ($\pm 10\%$).

The results show that a small mass difference of 10% will not have a significant impact on the area of the footprint. In Table 8.4 the quantitative data is given, where it can be seen that a maximum deviation of only 0.09% is found from the nominal value, indicating that the mass variations of $\pm 10\%$ have negligible impact on the achievable footprint area.

Table 8.4: Vehicle Mass Sensitivity Results

Vehicle Mass	Footprint Area	Absolute Change	Relative Change
1965 kg (-10%)	$68.12 \cdot 10^6$ km^2	$-0.03 \cdot 10^6$ km^2	-0.04%
2184 kg (nominal)	$68.15 \cdot 10^6$ km^2	-	-
2402 kg $(+10\%)$	$68.09 \cdot 10^6$ km^2	$-0.06 \cdot 10^6$ km^2	-0.09%

Although the footprint remains almost constant, careful examination of Figure 8.7 show subtle asymmetries that are in line with the underlying physics. The higher mass case achieves slightly greater maximum range in line with the initial heading. This occurs since higher mass corresponds to a higher total energy state for the same initial velocity, providing more energy for a longer flight path. In contrast to this, the lower mass case shows better performance in the reversed direction. Lower mass vehicles can initiate the reversal turn earlier in the trajectory due to the lower energy.

Figure 8.8 shows the trajectory characteristics for the mass variations. All three cases show similar skip glide patterns with synchronized altitude oscillations. The higher mass case experiences slightly higher loads throughout the trajectory, while the lower mass case shows marginally lower loads. This is also in line with the higher energy state of the higher mass vehicle.

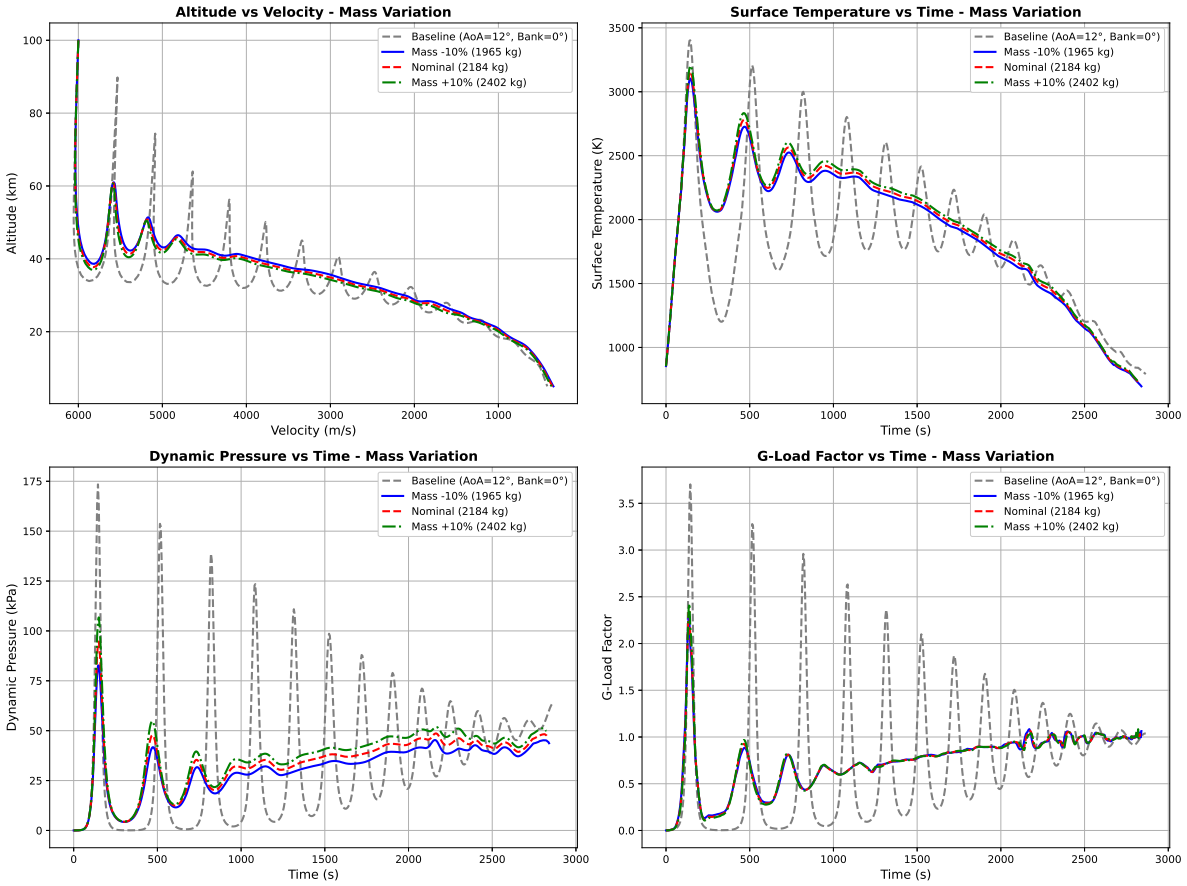


Figure 8.8: Trajectory characteristics for mass variations showing altitude vs velocity, dynamic pressure vs time, surface temperature vs time and g load factor vs time for $m = 1965, 2184, 2402$ kg and baseline trajectory.

8.3.2. Surface Area Sensitivity

The surface area was also varied by $\pm 10\%$ from the nominal value of 4.4 m^2 resulting in the test cases of 3.96 m^2, 4.4 m^2 and 4.84 m^2. Similar to the mass variations, the surface area changes produced minimal impact on the size of the footprint. Figure 8.9 shows the three footprint boundaries with a maximum deviation of 0.10% from nominal. Table 8.5 summarizes the results.

Table 8.5: Reference Area Sensitivity Results

Reference Area	Footprint Area	Absolute Change	Relative Change
3.96 m^2 (-10%)	$68.08 \cdot 10^6$ km^2	$-0.07 \cdot 10^6$ km^2	-0.10%
4.4 m^2 (nominal)	$68.15 \cdot 10^6$ km^2	-	-
4.84 m^2 $(+10\%)$	$68.12 \cdot 10^6$ km^2	$-0.03 \cdot 10^6$ km^2	-0.04%

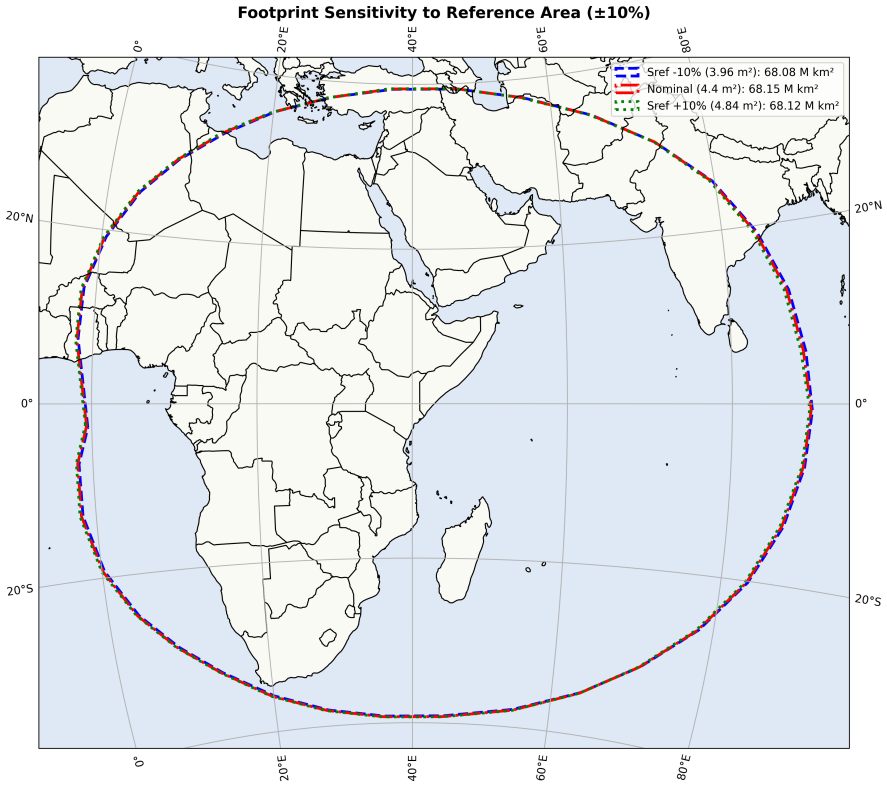


Figure 8.9: Footprint sensitivity to surface area

Figure 8.10 shows the trajectory differences for the surface area variations. What stands out is that the reduced reference area case experiences higher dynamic pressure peaks throughout the skip glide trajectory. This occurs because reduced reference area increases the descent rate since the vertical lift vector is decreased. This causes the vehicle to skip at a lower altitude, which can be seen in the altitude velocity plot. At these altitudes, the atmosphere is more dense and the dynamic pressure will be higher. For the same reason, the increased reference area case shows reduced dynamic pressure peaks, since the larger surface area provides more aerodynamic force at lower velocities.

The surface temperature data, like all aerodynamic coefficients, came from the baseline aerodynamic database generated for $S_{ref} = 4.4$ m^2. The temperature differences observed between the different surface areas do not come from the model accounting for changed area effects on heating, but from the altered trajectory profiles. The reduced surface area case maintains higher velocities for longer periods, which creates more heat. The actual thermal environment for an HGV with altered geometry would differ from these predictions as the real heating rate depends on the surface area.

Despite these trajectory differences in constraint loading and the limitations in accuracy, the final footprint areas remain nearly constant, where no asymmetries or inconsistencies occur when vehicle parameters are slightly changed.

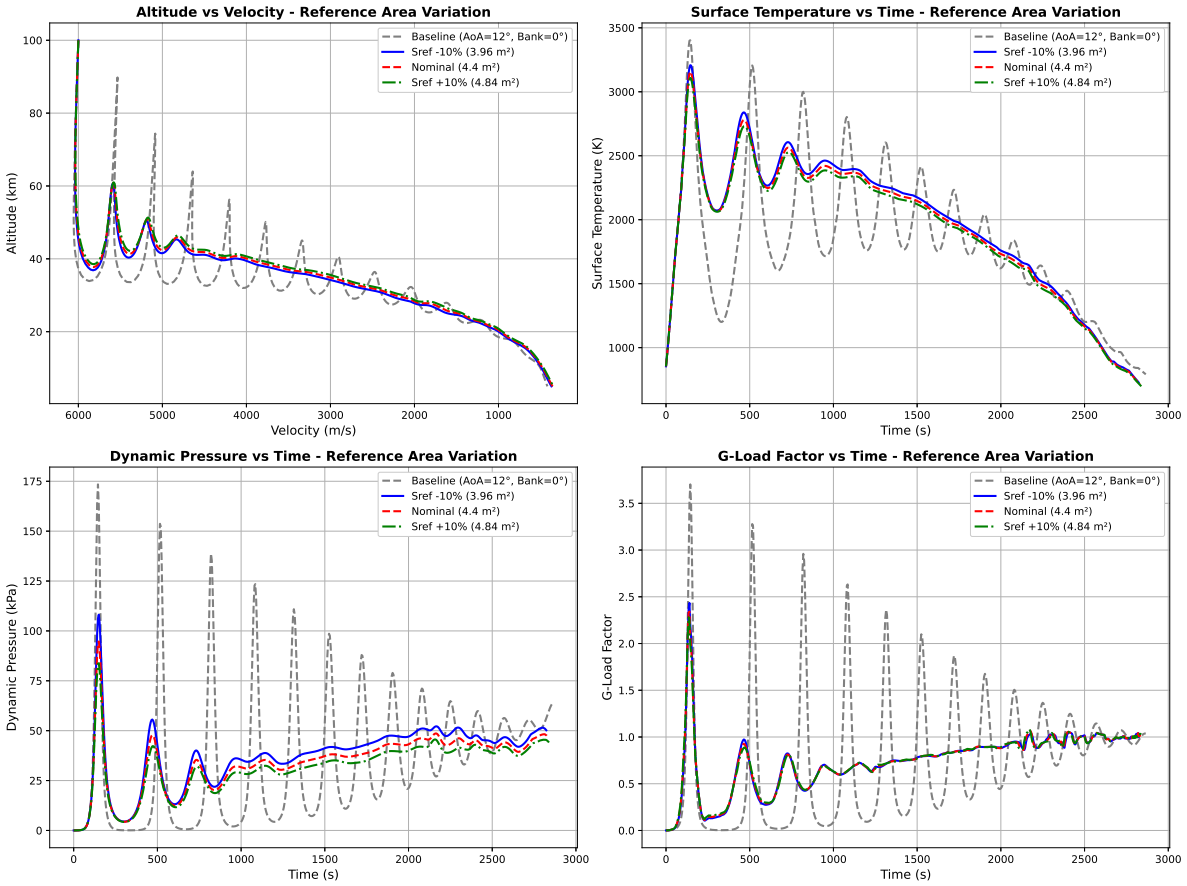


Figure 8.10: Trajectory characteristics for surface area variations showing altitude vs velocity, dynamic pressure vs time, surface temperature vs time and g-load factor vs time for $S_{ref} = 3.96, 4.4, 4.84$ m^2 and baseline trajectory.

8.4. Interpretation and Applicability

The sensitivity analysis examined two distinct parameter categories, namely the initial flight conditions and vehicle properties. The results showed significant differences in sensitivity magnitudes and gave insight into both the RL policy's generalization capabilities and the operational boundaries for mission planning.

8.4.1. Relative Parameter Sensitivities

The analysis identified that the initial velocity is the dominant sensitivity parameter. The strong sensitivity shows the quadratic relationship between velocity and kinetic energy, where small velocity changes create significant energy variations that determine the achievable range. In contrast, the other parameters tested produced footprint area changes of less than 1%, which shows the robustness of the RL policy to deal with variations in these parameters.

The policy's successful performance with varied altitude, flight path angle, velocity, mass and surface area, despite never encountering these variations during training, gives evidence of meaningful generalization beyond the training distribution.

All parameter variations produced physically consistent trajectory behavior with no unexpected characteristics. The skip glide patterns maintained their characteristic oscillatory behavior in all test cases, and the constraint profiles varied according to the expected aerodynamic and thermodynamic principles. The footprint shape was preserved in all configurations, which indicates that the RL policy maintains multi-directional capabilities regardless of parameter changes.

9

Conclusion & Recommendations

This chapter presents the conclusion, achievement of research objectives, and recommendations for extending the HGV trajectory optimization framework.

9.1. Conclusion

This thesis presented the first reinforcement learning approach for complete hypersonic glide vehicle footprint generation on any place on Earth with dual control authority over both bank angle and angle of attack, including direct target point navigation and no-fly zone avoidance. The framework uses the Soft Actor-Critic algorithm to learn optimal control policies directly from environment interaction without relying on heuristics, pre-designed control profiles, equilibrium glide assumptions, or other analytical simplifications.

The developed framework incorporates full three-degree-of-freedom dynamics on a rotating spherical Earth model, including Coriolis effects and control rate limitations. All generated trajectories satisfy operational path constraints including dynamic pressure, g-load, and temperature limits through real-time constraint monitoring. The trained policy generates complete reachable footprints in seconds for any starting location on Earth. Large-scale Monte Carlo validation with 100 million randomly sampled trajectories confirms the learned policy discovered the boundaries of the reachable domain, with the reinforcement learning footprint exceeding the Monte Carlo footprint by 5.1% in footprint area. Beyond footprint generation, the framework provides precision target point guidance to arbitrary global coordinates in less than a tenth of a second and incorporates no-fly zone avoidance integrated with target navigation.

The approach addresses fundamental limitations of existing methods, which typically rely on equilibrium glide assumptions, use fixed angle of attack profiles, or require re-optimization for each target direction. The simultaneous control of angle of attack and bank angle allows the policy to explore the full maneuverability envelope instead of being constrained by decoupled control strategies.

The current implementation uses a three-degree-of-freedom point-mass model and assumes a spherical Earth with standard atmospheric conditions. Future work should extend the framework to six-degree-of-freedom dynamics to capture rotational effects and moments and incorporate atmospheric uncertainties and wind profiles. The framework's modular architecture supports evaluation of multiple vehicle configurations, which can be adapted with minimal modifications for precision target point navigation and no-fly zone avoidance applications. This work demonstrates that reinforcement learning provides an effective approach for complete hypersonic vehicle footprint generation.

9.2. Achievement of Research Objectives

This section addresses the main research question and the supporting sub-questions, to show how each objective was achieved throughout this research.

9.2.1. Main Research Question

How can dual control of angle of attack and bank angle be implemented to maximize the reachable footprint of Hypersonic Glide Vehicles under aerodynamic, thermal and structural constraints?

Achievement:
This research successfully developed the first reinforcement learning-based framework for complete HGV footprint generation with simultaneous control of both angle of attack and bank angle including direct target point guidance and no-fly zone avoidance. The Soft Actor-Critic algorithm was implemented to learn optimal control policies directly from environment interaction, without relying on heuristics or other assumptions.

The framework generates complete reachable footprints in seconds for any starting location on Earth. The learned policy footprint outperforms the Monte Carlo simulation footprint area by 5.1%. All generated trajectories satisfy operational path constraints including dynamic pressure limits, g-load limits, and temperature limits through real-time constraint monitoring. The simultaneous control of both inputs allows the policy to explore the full maneuverability footprint.

9.2.2. Sub-Question 1: Flight Dynamics Modeling

How can HGV flight dynamics be accurately modeled to support footprint analysis in a high-fidelity model?

Achievement:
A 3-DOF point mass model was developed with the following high-fidelity components:

- Implementation of the USSA 1976 atmospheric model, providing altitude-dependent density calculations from sea level to above 100 km. This replaced the simplified exponential atmosphere model used in the initial development phases.
- Full incorporation of Earth rotation effects including Coriolis acceleration and centrifugal acceleration in the equations of motion. The spherical Earth approximation uses the standard radius of 6371 km with altitude dependent gravity.
- Realistic aerodynamic coefficient interpolation from tabulated data as functions of Mach number, altitude, and angle of attack.
- Implementation and validation of both spherical coordinates and ECEF Cartesian coordinates, with verified transformations between reference frames.
- Control rate limitations restricting angle of attack changes to $1^\circ/\text{s}$ and bank angle changes to $3^\circ/\text{s}$ to achieve realistic control behavior.

The model has been extensively validated through comparison between spherical and Cartesian implementations, verification against literature benchmarks, and early Monte Carlo simulations demonstrating the consistent behavior of skip-glide patterns and Coriolis induced asymmetries in the shape of the footprint.

9.2.3. Sub-Question 2: Coupled Nonlinear Optimization

How can the coupled nonlinear interaction between angle of attack and bank angle be optimized simultaneously?

Achievement:
The Soft Actor-Critic algorithm was selected and implemented to handle the simultaneous optimization of both control inputs. This choice was made after extensive experimentation with traditional optimization methods showed computational challenges with complex coupled dynamics. The reinforcement learning has several advantages for this application:

- SAC learns directly from trial and error experience without requiring predictive models for the aerodynamic relationships.
- A continuous action space that allows for continuous angle of attack in the range $[-4^\circ, 60^\circ]$ and bank angle in $[-90^\circ, 90^\circ]$.

- The entropy term in SAC's objective function encourages exploration while maximizing reward, which prevents convergence to suboptimal local solutions.
- Efficient sample reuse through the replay buffer, which allows the agent to learn from past experiences as well.

The trained policy learned to coordinate both angle of attack and bank angle together and automatically adjusts the energy management strategy depending on how far the target is.

9.2.4. Sub-Question 3: Multi-Directional Trajectory Generation

What computational approach enables efficient multi-directional trajectory generation to create footprints?

Achievement:
An efficient footprint generation methodology was developed that generates trajectories in 36 evenly spaced directions around the initial position. The approach has several innovations:

- Instead of optimizing the maximum range only in predefined directions, the framework uses target points positioned beyond the expected maximum range in each direction. This formulation allows the same learned policy to be applied in all directions without direction-specific training or re-optimization.
- The reward function encourages minimizing the great-circle distance to the target point plus constraint penalties. This simple formulation leads naturally to footprint boundary discovery.
- Once trained, the policy generates trajectories in real-time. Complete footprints are generated in seconds, compared to several days it takes for Monte Carlo validation.
- Spherical trigonometry is used to compute bearings, distances, and cross-track deviations on Earth's spherical surface.
- Parallelized training using GPU for neural network weight updates reduces training time.

The computational efficiency allows for footprint generation on any location on Earth in seconds, which allows for real-time operational scenario evaluation.

9.2.5. Sub-Question 4: Constraint Satisfaction

How can trajectories be generated that maintain constraint satisfaction throughout the entire flight?
Achievement:
A comprehensive constraint handling framework was developed and integrated into the reinforcement learning training process.

- Three critical constraints are continuously evaluated at every integration step: dynamic pressure, load factor, and stagnation point temperature.
- Constraint violations during training result in negative rewards that is scaled by how severe the violation is, where the penalty coefficient was tuned through experimentation.
- Instead of terminating episodes when a constraint is violated, the framework gives an exponential penalty that grows smoothly as the vehicle approaches the constraint boundaries. This gives the agent information from near-boundary experiences. Episodes end only when the vehicle reaches the final altitude.

9.2.6. Validation Across Different Conditions

How can footprint accuracy be validated across different conditions?

Achievement:
Validating the reinforcement learning based trajectory optimization framework required proving that the model was implemented correctly and that the learned policy discovered the true physical boundaries instead of a suboptimal solution. A comprehensive validation strategy was performed:

- A test problem with known optimal solutions was used to evaluate different optimization methods. The RL agent successfully learned the correct descent strategy while satisfying constraints, demonstrating the potential of the approach.

- The equations of motion were implemented in both spherical coordinates and Cartesian ECEF coordinates. After propagating identical trajectories in both systems and converting between reference frames, the results agreed up to seven significant figures. This verified the correctness of the coordinate transformations and confirmed that both implementations correctly accounted for Earth rotation effects.
- The custom C++ `CGAL` interpolation was compared against Python's `LinearNDInterpolator` over a complete trajectory. The difference was only 0.00199%, validating the interpolation accuracy while providing significant computational performance improvements.
- Footprints generated from symmetric latitudes in Northern and Southern hemispheres showed identical areas and symmetric patterns, confirming correct implementation of Coriolis forces. The consistent eastward bias further validated the rotational dynamics of the Earth.
- A large-scale Monte Carlo simulation with 100 million trajectory evaluations using randomly sampled control sequences provided independent verification of the reachable domain. The close agreement of the footprint area demonstrates that the RL policy discovered the true boundaries of the reachable domain. The slightly larger RL footprint indicates that the learned policy found more efficient trajectories than pure random sampling. The Monte Carlo simulation also gives a validation that points within the border of the footprint can be reached.
- The continuous circular footprint without gaps or abrupt curvature changes confirmed that the dynamics and control logic behaved consistently across all flight directions. This is a visual verification that the integrated framework was stable.
- The framework has been extended to two fundamentally different HGV configurations and were trained independently. Both vehicles generated continuous footprints despite their different aerodynamic characteristics, initial conditions and operational envelopes. The DF-ZF converged to the optimal L/D ratio of 8°, while the HTV2 converged to 12°, the optimal L/D ratio for this vehicle. This demonstrates that the reinforcement learning framework learns vehicle-specific optimal control strategies. The consistent footprint symmetry across both vehicles validated that the framework correctly handles different vehicle configurations without requiring modifications to the training methodology.
- Footprints were successfully generated from various global locations. All locations produced consistent results without requiring location specific retraining.
- Sensitivity Analysis has been performed to quantify the policy's robustness. This has been done for the initial velocity, altitude, flight path angle, vehicle mass, and surface area. The policy maintained reasonable performance while operating in regions of state space not explicitly encountered during training, which shows good generalization of the model.
- Target point navigation successfully demonstrated the precision guidance to arbitrary global coordinates. This also included avoiding no-fly zones while maintaining target point convergence across different geometric configurations. This shows the framework's ability to deal with operational applications such as airspace restrictions.

9.3. Recommendations

9.3.1. Scramjet Propulsion Integration

As described in Appendix C, integrating scramjet propulsion offers operational advantages. The range can be extended through powered flight. Integration requires modifications to the equations of motion where the effect of the thrust force needs to be implemented. Time-varying vehicle mass must be implemented to account for the fuel consumption. The action space of the reinforcement learning model should be extended from angle of attack and bank angle, to angle of attack, bank angle, and thrust, while the observation speeds need expansion to include remaining fuel fraction and margins to ignition envelope boundaries. Scramjet-specific constraints must be added, including angle of attack limits for shock on lip condition and ignition envelope boundaries that depend on the velocity and altitude of the HGV, which is further explained in Appendix C. The existing HGV class and modular constraint system can accommodate these changes without requiring new class structures.

9.3.2. Waypoint Navigation

The current framework can navigate to a target point while avoiding a no-fly zone. This model can be extended by passing through multiple intermediate waypoints before reaching the final destination. To implement this capability, the framework needs a few modifications. First, the agent must be able to track multiple waypoints simultaneously. The observables should be extended with the angle and distances to all these points. The training scenarios should include different geometric configurations. This can be done by randomizing the locations of the waypoints, just as has been done for the single target points. This diversity ensures that the agent learns to handle different sequences of target points.

9.3.3. Terminal Phase Guidance

The current framework optimizes trajectories until the HGV reaches an altitude of 5 km and successfully demonstrates guidance to target points at this altitude. However, operational scenarios require the vehicle to continue from 5 km to a precise location on the ground. To extend this capability, an additional terminal phase guidance framework should be developed that takes over from the current at 5 km altitude. This terminal framework would control the final descent phase and steer the HGV to a precise ground coordinate.

9.3.4. Extension to 6 degree-of-freedom model

The current framework uses a 3-DOF point-mass model on a spherical Earth. A 6-DOF model would incorporate three additional rotational degrees of freedom, which allows simulation of attitude dynamics and coupling effects between translational and rotational motion. The implementation of a 6-DOF framework within the RL architecture would require expanding the state space to include rotational states and modifying the action space and observation space accordingly.

For footprint generation specifically, the advantages of changing to a 6-DOF model are limited. Although the current 3-DOF model assumes instantaneous angle of attack and bank angle changes subject to rate limits, footprint trajectories do not require frequent control changes, making this simplification acceptable for boundary determination. A 6-DOF model would provide more physical realism by explicitly modeling the rotational dynamics, but this comes at a significant computational cost due to the larger network required to handle the expanded state space. The primary advantages of a 6-DOF model framework would occur when using the trained model for actual vehicle control applications, where explicit representation of attitude dynamics and moment generation becomes essential for implementation on real flight control systems.

10

Submitted Paper to Aerospace Science & Technology

The paper presented in this chapter has been submitted to the journal Aerospace Science & Technology.

Reinforcement Learning for Hypersonic Glide Vehicle Trajectory Optimization: A Soft Actor-Critic Approach

Niels van Mierlo, Marc Naeije, Mark Verveld

Abstract

Hypersonic glide vehicle trajectory optimization requires generating complete reachable footprints for mission planning under strict path constraints. Current methods face limitations including equilibrium glide assumptions, fixed angle of attack profiles and incomplete footprint coverage requiring reoptimization for each target direction. This work presents the first reinforcement learning approach for complete global footprint generation with dual control authority (bank angle and angle of attack) on a rotating spherical Earth model including direct target point guidance and no-fly zone avoidance. The trained policy generates footprints for any location on Earth while incorporating full three-degree-of-freedom dynamics including Coriolis effects and control rate limitations. All trajectories satisfy operational constraints including dynamic pressure, g-load and temperature limits. The policy learns purely from the objective to maximize range in every direction without pre-designed control profiles. Based on the Soft Actor-Critic algorithm, optimal control strategies are learned directly from environment interaction. Large-scale Monte Carlo validation with 100 million randomly sampled trajectories confirms the learned policy discovered the boundaries of the reachable domain, with the reinforcement learning footprint exceeding the Monte Carlo footprint by 5.1% in footprint area. The framework provides precision target guidance to arbitrary global coordinates and allows for no-fly zone avoidance.

Keywords: hypersonic glide vehicle, reinforcement learning, trajectory optimization, footprint generation, soft actor-critic, no-fly zone avoidance, direct target point guidance

1. Introduction

Hypersonic Glide Vehicles (HGV) are spaceplanes that travel at speeds exceeding Mach 5. Unlike ballistic missiles that follow predictable parabolic trajectories, HGVs combine hypersonic velocity with lift generation, which gives them the ability to control the flight path during atmospheric flight. These vehicles are usually launched by rockets, after which they glide through the atmosphere while generating lift to extend their range and provide cross-range maneuverability. Entry guidance has been designed for the Space Shuttle since the 1970s and was first pioneered in the Apollo program [1]. Examples of HGV include CAV, HTV, HTV-2, DF-ZF, X-37B, AHW, and YU-71 [2, 3].

The trajectory optimization problem for HGVs is challenging due to the extreme flight environment and the coupled nature of the control problem. During atmospheric flight, these vehicles experience severe heating, structural loads, and dynamic pressures. The flight regime spans altitudes where the HGV goes through many atmospheric layers with varying density, temperature, and pressure. Throughout the whole flight, the vehicle must satisfy three path constraints simultaneously, namely the dynamic pressure, thermal, and load factor constraint. These constraints significantly limit the feasible trajectory space, requiring precise guidance to avoid constraint violations.

HGV trajectory control involves two control variables, namely the angle of attack and the bank angle. By coordinating these two controls, the HGVs can convert altitude into range and maneuver laterally to reach targets, creating a maneuvering space that defines the reachability of the HGVs, which can be visualized as the landing footprint. To calculate a footprint, the boundary points of the landing footprint should be determined. Solving a family of optimal control problems rapidly and reliably is a daunting task according to Saraf et al. [4].

Traditional optimization approaches can be categorized into indirect and direct methods [5]. Indirect approaches rely on Pontryagin's minimum principle, converting the optimal control problem into a two-point

boundary value problem [6, 7, 8, 9]. While this guarantees optimality, solving two-point boundary value problems is difficult and requires good initial guesses for complex problems [10, 11]. The nonlinear HGV dynamics make this particularly challenging. In contrast, direct methods have become increasingly popular with the advancement of computing technology [2]. These include direct shooting, collocation, and pseudospectral methods, which convert the control problem into a parametric optimization problem solved through nonlinear programming techniques such as sequential quadratic programming [12, 13, 14, 15, 16]. Zhang et al. [17] have proposed a direct approach using nonlinear programming and the Gaussian Pseudospectral Method to minimize the terminal control energy. Furthermore, Tu et al. [18] have proposed a direct method using a nonlinear programming problem and the direct collocation method. However, the time it takes to perform the direct method is unpredictable, and convergence cannot be guaranteed when applied to hypersonic flight scenarios [1].

Convex optimization techniques, particularly second-order cone programming (SOCP), have also been applied [12], though HGV trajectory problems do not naturally fit convex frameworks. Liu et al. [19] reformulated the equations of motion with respect to energy to enable SOCP solution, while pseudospectral optimal control and convex optimization were combined for drag-energy guidance by Sagliano et al. [20].

Early footprint generation work by Vinh et al. [21] produced footprints without vehicle constraints or Coriolis effects. This work was continued by Ngo et al. [22], who generated nearly optimal footprints but also neglected Coriolis effects. Saraf et al. [4] incorporated path constraints and Coriolis effects. However, none of these methods used angle of attack control in combination with bank angle control. Most existing methods optimize only bank angle while using a fixed angle of attack profile, where the full control authority is not explored.

In recent years, research has investigated advanced solution approaches that offer reliable convergence, optimal results, and rapid computation. This effort has been driven by new technologies including artificial intelligence and machine learning [23]. The application of reinforcement learning (RL) has accelerated in recent years [24]. Reinforcement learning algorithms have shown promising results for aerospace problems [25, 26]. In contrast to traditional guidance algorithms, reinforcement learning-based guidance algorithms show strong anti-disturbance capabilities and real-time performance [27, 28, 29]. Chai et al. [30] integrated deep neural networks with optimal control for real-time reentry trajectory planning, while Shi et al. [1] developed deep learning frameworks for onboard trajectory generation. However, these approaches rely on supervised learning from pre-computed optimal trajectories. Recent developments in model-free reinforcement learning have explored direct policy training. The Soft Actor-Critic (SAC) algorithm has shown strong sample efficiency in continuous domains [31], enabling direct policy learning without optimal trajectory datasets. Gao et al. [32] applied deep deterministic policy gradient to reentry trajectory optimization, and Su et al. [33] used inverse reinforcement learning for point-to-point entry guidance. Das et al. [34] note that RL offers significant opportunities for HGV trajectory control, though further research is needed.

Most state-of-the-art footprint generation methods rely on several simplifying assumptions that limit the accuracy. Simplifications in research include employing equilibrium glide assumptions, using heuristics for minimum range boundary determination, neglecting Coriolis forces from Earth rotation, and fixed angle of attack profiles at predetermined values with trajectory shaping through bank angle modulation. These assumptions reduce computational cost, but limit the maneuverability and accuracy of the footprint.

This work presents the first RL-based approach for complete footprint generation with dual control authority (bank angle and angle of attack) on a rotating spherical Earth model, where the RL model can be extended with no-fly zone avoidance and direct target point navigation. The trained policy generates footprints for any global location while incorporating full three-degree-of-freedom dynamics including Coriolis effects and control rate limitations.

The remainder of this paper is organized as follows. Section 2 formulates the HGV trajectory optimization problem including three-degree-of-freedom dynamics, aerodynamic forces, and operational constraints. Section 3 presents the reinforcement learning framework including Markov Decision Process (MDP) formulation, SAC algorithm implementation, and training architecture. Section 4 demonstrates footprint generation, direct target point guidance, no-fly zone avoidance, Monte Carlo validation, and sensitivity analysis. Section 5 summarizes key contributions and outlines future work.

2. Problem Formulation

This section presents the mathematical formulation of the HGV trajectory optimization problem including the

vehicle dynamics model, control inputs, and operational constraints.

2.1. Vehicle Dynamics Model

The HGV is modeled as a three-degree-of-freedom point mass on a rotating spherical Earth. The state vector is defined as:

$$\mathbf{x} = \begin{bmatrix} h & \lambda & \phi & V & \gamma & \psi \end{bmatrix}^{\mathsf{T}} \tag{1}$$

where h is altitude above sea level, λ is longitude, ϕ latitude, V ground-relative velocity, γ is flight path angle relative to the local horizontal, and ψ is heading angle measured clockwise from north.

2.1.1. Aerodynamic Forces

The aerodynamic forces that act on the vehicle are:

$$L = \tfrac{1}{2}\,\rho(h)\,V^2\,S_{\text{ref}}\,C_L(\alpha, M, h) \tag{2}$$

$$D = \tfrac{1}{2}\,\rho(h)\,V^2\,S_{\text{ref}}\,C_D(\alpha, M, h) \tag{3}$$

where L and D are lift and drag forces respectively, $\rho(h)$ is atmospheric density from the COESA 1976 model, S_{ref} is the reference area, and C_L and C_D are the lift and drag coefficients respectively obtained from aerodynamic tables as functions of angle of attack α, Mach number M, and altitude h.

2.1.2. Gravitational Model

Gravitational acceleration is computed using the spherical Earth approximation:

$$g(h) = \frac{\mu}{(R_E + h)^2} \tag{4}$$

where $\mu = 3.986 \times 10^{14}$ m^3/s^2 is Earth's gravitational parameter, $R_E = 6371$ km is Earth's mean radius, and $r = R_E + h$ is the radial distance from Earth's center.

2.1.3. Equations of Motion

The motion of the HGV on a rotating spherical Earth is described by the following differential equations [20]:

$$\dot{h} = V \sin\gamma \tag{5}$$

$$\dot{\lambda} = \frac{V \cos\gamma \sin\psi}{r \cos\phi} \tag{6}$$

$$\dot{\phi} = \frac{V \cos\gamma \cos\psi}{r} \tag{7}$$

$$\begin{aligned} \dot{V} = -\frac{D}{m} - g\sin\gamma + \omega_\oplus^2 r \cos\phi\,(\sin\gamma\cos\phi \\ - \cos\gamma\sin\phi\cos\psi) \end{aligned} \tag{8}$$

$$\begin{aligned} \dot{\gamma} =& \frac{L\cos\sigma}{mV} + \left(\frac{V}{r} - \frac{g}{V}\right)\cos\gamma + 2\omega_\oplus \cos\phi \sin\psi \\ &+ \frac{\omega_\oplus^2 r}{V}\cos\phi\,(\cos\gamma\cos\phi + \sin\gamma\sin\phi\cos\psi) \end{aligned} \tag{9}$$

$$\begin{aligned} \dot{\psi} =& \frac{L\sin\sigma}{mV\cos\gamma} + \frac{V}{r}\cos\gamma\sin\psi\tan\phi + 2\omega_\oplus\,(\sin\phi \\ &- \cos\phi\tan\gamma\cos\psi) + \frac{\omega_\oplus^2 r}{V\cos\gamma}\sin\phi\cos\phi\sin\psi \end{aligned} \tag{10}$$

where $\omega_\oplus = 7.2921 \times 10^{-5}$ rad/s is Earth's rotation rate, m is the vehicle mass, and σ is the bank angle. These equations incorporate Coriolis effects and centrifugal accelerations caused by Earth's rotation.

2.2. Control Inputs

The HGV trajectory is controlled through two independent control variables. The bank angle σ controls the lateral component of lift by rotating the lift vector around the velocity axis. The angle of attack α determines the magnitude of aerodynamic forces through the influence on C_L and C_D. Both controls are subject to the following constraints:

$$\sigma_{\min} \le \sigma \le \sigma_{\max} \tag{11}$$

$$\alpha_{\min} \le \alpha \le \alpha_{\max} \tag{12}$$

$$|\dot{\sigma}| \le \dot{\sigma}_{\max} \tag{13}$$

$$|\dot{\alpha}| \le \dot{\alpha}_{\max} \tag{14}$$

2.3. Path Constraints

Throughout the flight, the vehicle must satisfy three critical operational constraints simultaneously. These include the load factor, dynamic pressure and thermal constraint.

The load factor represents the ratio of total aerodynamic force to weight:

$$n = \frac{\sqrt{L^2 + D^2}}{mg} \le 25 \tag{15}$$

The dynamic pressure constraint limits aerodynamic loads:

$$q = \frac{1}{2}\rho V^2 \leq 500 \text{ kPa} \tag{16}$$

The thermal constraint ensures the thermal protection system can withstand aerodynamic heating:

$$T_{\text{stag}}(\alpha, M, h) \leq 3500 \text{ K} \tag{17}$$

where T_{stag} is the stagnation point temperature obtained from aerodynamic heating calculations and $T_{\max} = 3500$ K is the maximum allowable stagnation temperature.

2.4. Footprint Generation Problem

The footprint represents the complete set of reachable landing points from a given initial state. Each point on the footprint boundary corresponds to a trajectory that maximizes range in a specific azimuthal direction.

For each initial heading angle $\psi_0 \in [0, 2\pi)$, the optimization problem is formulated as:

$$\begin{aligned}
\max_{\alpha(t),\sigma(t)} \quad & d(\mathbf{x}_0, \mathbf{x}_f) \\
\text{subject to} \quad & \dot{\mathbf{x}}(t) = f(\mathbf{x}(t), \alpha(t), \sigma(t)), \\
& n(t) \leq n_{\max}, \\
& q(t) \leq q_{\max}, \\
& T_{\text{stag}}(t) \leq T_{\max}, \\
& \sigma_{\min} \leq \sigma(t) \leq \sigma_{\max}, \\
& \alpha_{\min} \leq \alpha(t) \leq \alpha_{\max}, \\
& |\dot{\sigma}(t)| \leq \dot{\sigma}_{\max}, \\
& |\dot{\alpha}(t)| \leq \dot{\alpha}_{\max}, \\
& \mathbf{x}(0) = \begin{bmatrix} h_0 & \lambda_0 & \phi_0 & V_0 & \gamma_0 & \psi_0 \end{bmatrix}^{\mathsf{T}}, \\
& h(t_f) = h_{\text{terminal}}
\end{aligned} \tag{18}$$

where $d(\mathbf{x}_0, \mathbf{x}_f)$ is the great-circle distance computed using standard spherical trigonometry. The dynamics $f(\mathbf{x}, \alpha, \sigma)$ are defined by Equations (5)–(10), and the path constraints by Equations (15)–(17). The final time t_f is free and determined by the terminal altitude condition, set to an altitude of 5 km in this study.

For a complete footprint with $N = 36$ directions, this problem is solved for headings $\psi_0^{(i)} = i \cdot \pi/18$ rad, $i = 0, \ldots, 35$. The reinforcement learning approach explained in Section 3 trains a single policy that generates near-optimal trajectories for arbitrary headings anywhere on Earth without iterative optimization.

3. Methodology

This section presents the reinforcement learning framework for HGV trajectory optimization. The methodology consists of two phases: a training phase where the SAC algorithm learns optimal control policies through interaction with the simulation environment and a deployment phase where trained policies generate footprints through systematic target sweeps. Figure 1 provides an overview of the complete framework architecture.

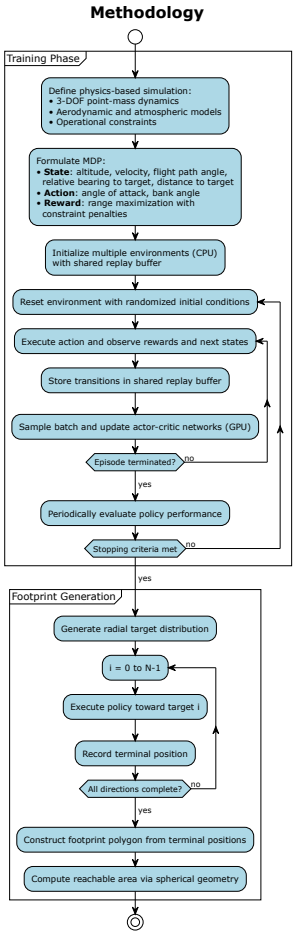


Figure 1: Methodology framework overview showing training phase and footprint generation phase.

3.1. Reinforcement Learning Formulation

The trajectory optimization problem defined in Section 2.4 is reformulated as a Markov Decision Process (MDP), allowing the use of model-free policy learning algorithms. The MDP is defined by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, r, \gamma)$, where $\mathcal{S}$ represents the state space, $\mathcal{A}$ the action space, $\mathcal{P}$ the transition dynamics governed

by Equations (5)–(10), r the reward function, and γ the discount factor. The Soft Actor-Critic algorithm is used to learn a stochastic policy $\pi_\phi : \mathcal{S} \to \mathcal{A}$ that maximizes the entropy-penalized objective [31]:

$$J(\pi) = \sum_{t=0}^{T} \mathbb{E}_{(s_t,a_t)\sim\rho_\pi} \left[r(s_t, a_t) + \alpha H(\pi(\cdot|s_t)) \right] \tag{19}$$

where ρ_π is the state-action distribution induced by the policy π, $r(s_t, a_t)$ is the immediate reward at the time step t, α is the temperature parameter that controls the exploration-exploitation trade-off and $H(\pi(\cdot|s_)) = -\mathbb{E}_{a\sim\pi}[\log \pi(a|s_t)]$ is the policy entropy. The entropy term encourages exploration throughout training by rewarding policies that maintain stochastic action distributions, which prevents premature convergence to suboptimal deterministic strategies. This is particularly important for this problem, in order to discover diverse control strategies in the complex aerodynamic regime of hypersonic flight where multiple local optima exist. SAC was selected over on-policy algorithms like PPO for superior sample efficiency through off-policy learning, and over TD3 for its entropy regularization.

3.1.1. State Space Design

The observation vector provided to the agent at each decision point is defined as:

$$\mathbf{s} = \begin{bmatrix} \bar{h} & \bar{V} & \bar{\gamma} & \bar{\kappa} & \bar{d} \end{bmatrix}^\mathsf{T} \in \mathbb{R}^5 \tag{20}$$

where the normalized components are:

$$\bar{h} = \frac{h - h_{\min}}{h_{\max} - h_{\min}}, \qquad \bar{h} \in [0, 1] \tag{21a}$$

$$\bar{V} = \frac{V}{V_c}, \qquad \bar{V} \in [0, 1] \tag{21b}$$

$$\bar{\gamma} = \frac{\gamma}{\pi/2}, \qquad \bar{\gamma} \in [-1, 1] \tag{21c}$$

$$\bar{\kappa} = \frac{\kappa}{\pi}, \qquad \bar{\kappa} \in [-1, 1] \tag{21d}$$

$$\bar{d} = \frac{d}{d_{\max}}, \qquad \bar{d} \in [0, 1] \tag{21e}$$

Here $h_{\min}$ and $h_{\max}$ define the altitude bounds of 5 km and 160 km respectively, V is the velocity, $V_c = \sqrt{\mu/R_E}$ is the circular orbital velocity, γ is the flight path angle, $\kappa = \beta - \psi$ is the heading difference between the bearing to target β and current heading ψ, and d is the great-circle distance to target normalized by expected maximum range $d_{\max}$, which differs for the type of HGV that is simulated. All state components are bounded within $[-1, 1]$ or $[0, 1]$ to facilitate neural network training and ensure numerical stability during gradient computation. This relative state representation, using heading difference κ and distance d instead of absolute coordinates, allows the policy to generalize across arbitrary geographic locations without retraining.

3.1.2. Action Space Definition

The action space $\mathcal{A}$ consists of continuous aerodynamic control inputs:

$$\mathcal{A} = [\alpha_L, \alpha_U] \times [\sigma_L, \sigma_U] \subseteq \mathbb{R}^2 \tag{22}$$

where the control vector at time step t is:

$$\mathbf{a}_t = \begin{bmatrix} \alpha_t \\ \sigma_t \end{bmatrix} \in \mathcal{A} \tag{23}$$

with:

- $\alpha_t \in [\alpha_L, \alpha_U]$ is the angle of attack measured in the vehicle's longitudinal plane.
- $\sigma_t \in [\sigma_L, \sigma_U]$ is the bank angle measured around the velocity vector.

The specific bounds used in this study are $\alpha_L = -4°$, $\alpha_U = 60°$, $\sigma_L = -90°$, and $\sigma_U = 90°$, selected based on aerodynamic performance characteristics of the HGV configuration. To ensure physically achievable control behavior, first-order rate limits are imposed on the control variables:

$$|\dot{\alpha}_t| \leq \dot{\alpha}_{\max} = 1°/\text{s}, \quad |\dot{\sigma}_t| \leq \dot{\sigma}_{\max} = 3°/\text{s} \tag{24}$$

3.1.3. Reward Function Formulation

The reward function $r(s_t, a_t)$ is designed to incorporate both the objective of maximizing range in each radial direction and constraint satisfaction. The immediate reward at time step t decomposes additively into progress and constraint penalty components:

$$r_t = r_t^{\text{progress}} + r_t^{\text{constraint}} \tag{25}$$

The progress term rewards the vehicle for reducing its distance to the target:

$$r_t^{\text{progress}} = \frac{\Delta d_t}{\ell_{\text{scale}}} \tag{26}$$

where $\Delta d_t = d_t - d_{t+1}$ represents the distance reduction over the control interval and $\ell_{\text{scale}} = 1 \times 10^5$ is a dimensional scaling factor selected to make the reward magnitudes manageable for the neural network. The cumulative progress reward over a complete episode is:

$$\sum_{t=0}^{T-1} r_t^{\text{progress}} = \frac{d_0 - d_T}{\ell_{\text{scale}}} \tag{27}$$

representing the net distance reduction from the initial to final states, thereby naturally aligning the stepwise feedback with the terminal mission objective. This formulation inherently guides the policy toward footprint boundaries when targets are positioned beyond maximum achievable range.

The constraint penalty enforces satisfaction of operational limits through a smooth, exponentially increasing penalty as the vehicle state approaches constraint boundaries. The load factor and dynamic pressure constraints are reformulated as velocity margins, while the thermal constraint uses temperature margin directly. The constraint margin vector is:

$$\mathbf{m}_t = \begin{bmatrix} m_{n,t} \\ m_{q,t} \\ m_{T,t} \end{bmatrix} = \begin{bmatrix} V_t - V_{\text{gc}}(h_t, \alpha_t) \\ V_t - V_{\text{qc}}(h_t) \\ T_{\text{stag}}(\alpha_t, M_t, h_t) - T_{\max} \end{bmatrix} \tag{28}$$

where $V_{\text{gc}}(h_t, \alpha_t)$ and $V_{\text{qc}}(h_t)$ are the critical velocities for load factor and dynamic pressure respectively expressed in m/s, while the thermal margin is a temperature difference in K. Negative margins indicate feasible operation below constraint limits, zero represents the constraint boundary, and positive margins indicate violations. The penalty function operates on the maximum, and therefore most critical margin:

$$m_t^* = \max(\mathbf{m}_t) = \max(m_{n,t}, m_{q,t}, m_{T,t}) \tag{29}$$

The constraint penalty is defined as:

$$r_t^{\text{constraint}} = \begin{cases} 0 & \text{if } m_t^* < \delta \\ r_{\max} \cdot \dfrac{e^{\zeta \bar{m}_t} - 1}{e^{\zeta} - 1} & \text{if } \delta \le m_t^* < 0 \\ r_{\max} & \text{if } m_t^* \ge 0 \end{cases} \tag{30}$$

where the normalized margin is:

$$\bar{m}_t = \frac{m_t^* - \delta}{-\delta} \tag{31}$$

and the parameters are:

- $\delta = -500$ defines the safe margin threshold representing -500 m/s or -500 K,
- $\zeta = 10$ controls the exponential steepness, and
- $r_{\max} = -3$ sets the maximum penalty magnitude.

These parameters were tuned to balance constraint satisfaction with range maximization objectives. The three regions provide distinct behavior. The safe region ($m_t^* < -500$) applies zero penalty enabling unconstrained exploration, the warning region ($-500 \le m_t^* < 0$) applies exponentially increasing penalties providing strong learning signals near boundaries, and the violation region ($m_t^* \ge 0$) applies constant maximum penalty discouraging violations. The total reward becomes:

$$r_t^{\text{total}} = r_t^{\text{progress}} + r_t^{\text{constraint}} \tag{32}$$

3.2. Framework Architecture

The trajectory optimization framework consists of three interconnected components arranged in a hierarchical architecture. At the foundation is the HGV physics class, which implements the three-degree-of-freedom point-mass dynamics model containing the vehicle state $\mathbf{x} = \begin{bmatrix} h & \lambda & \phi & V & \gamma & \psi \end{bmatrix}^\mathsf{T}$, aerodynamic data $\{C_L(\alpha, M, h), C_D(\alpha, M, h), T_{\text{stag}}(\alpha, M, h)\}$, and constraint limits. The physics core integrates the Equations of Motion (Equations 5–10) forward in time through operator $\mathbf{x}_{t+\Delta t} = \mathcal{I}(\mathbf{x}_t, \mathbf{u}_t, \Delta t)$, where $\Delta t = 5$ s and $\mathbf{u}_t = \begin{bmatrix} \alpha_t & \sigma_t \end{bmatrix}^\mathsf{T}$. The integration uses a fourth-order Runge-Kutta method.

The physics core is wrapped by a simulation environment that converts continuous dynamics into discrete time steps for reinforcement learning. Each training episode begins by sampling randomized initial conditions:

$$\phi_0 \sim \mathcal{U}(-\pi/3, \pi/3), \quad \psi_0 \sim \mathcal{U}(0, 2\pi), \quad i_{\text{target}} \sim \text{Discrete}\{0, \ldots, 35\} \tag{33}$$

These three randomizations: initial latitude, initial heading, and target direction, enable a single trained policy to generate footprints globally without retraining for different locations or orientations. By training on diverse latitudes and headings, the policy learns to implicitly account for latitude-dependent Coriolis forces through the relationship between control actions and resulting trajectory curvature.

At each time step, the simulation environment advances by $\Delta t = 5$ s. First, the commanded actions $\mathbf{a}_t$ are rate-limited to satisfy Equation 24. The dynamics (Equations 5–10) are then integrated to obtain $\mathbf{x}_{t+1}$. The reward r_t is computed from the distance reduction and constraint violations (Equations 25–30). The normalized observation vector $\mathbf{s}_{t+1}$ is constructed from the updated state. The method returns the observation

$\mathbf{s}_{t+1}$, reward r_t, and a termination flag, terminating when $h_{t+1} < 5$ km.

3.3. Neural Network Architecture

Both actor and critic networks use three fully-connected hidden layers with [1024, 512, 256] neurons and ReLU activations. This deep architecture was selected to capture the complex nonlinear aerodynamic relationships across the wide flight envelope. The actor network outputs Gaussian distributions with mean and standard deviation parameters for both controls, allowing stochastic exploration during training and deterministic control during footprint generation. Twin critics with independent parameters reduce value overestimation, improving training stability.

3.4. Training Procedure

Training uses 16 parallel environment instances executing on separate CPU processes, while neural network computations occur on the GPU. Each parallel step generates 16 experience tuples stored in the replay buffer. The algorithm performs gradient updates after each environment step by sampling batches of 512 transitions uniformly from the buffer.

The critic networks minimize the temporal difference error using target networks updated with exponential moving averages. The Soft Actor-Critic algorithm [31] is adopted using the Stable-Baselines3 implementation [35]. SAC was selected for its sample efficiency in continuous control domains and stable off-policy learning with automatic entropy tuning. Key training parameters are summarized in Table 1.

Table 1: SAC training hyperparameters

Parameter	Value
Discount factor γ	0.99
Network architecture	[1024, 512, 256]
Activation function	ReLU
Batch size	512
Replay buffer size	10^6
Entropy coefficient	Auto-tuned
Training frequency	1
Gradient steps	1
Total time steps	10^7

Early stopping is implemented through periodic evaluation every 10^4 time steps. A separate evaluation environment runs 5 episodes with deterministic actions, computing the mean episode return. Training terminates if no improvement occurs over 20 consecutive evaluations after a minimum of 50 evaluations.

3.5. Direct Target Point Guidance

The framework provides precision guidance to arbitrary geographic coordinates $(\phi_{\text{target}}, \lambda_{\text{target}})$ using a single trained policy. During training, the target coordinates are uniformly sampled from a circular region centered at half the approximate maximum range $d_{\max}$ in the initial heading direction. The radial distance from this center point is sampled as $r = R\sqrt{U}$ where $U \sim \mathcal{U}(0,1)$ and $R = 0.5d_{\max} + 1000$ km, ensuring uniform area density and preventing target clustering near the origin. The angular position on this circular region is determined by $\theta \sim \mathcal{U}(0, 2\pi)$. These polar coordinates (r, θ) are then converted to geographic coordinates $(\phi_{\text{target}}, \lambda_{\text{target}})$. The complete set of randomized variables is as follows:

$$\begin{aligned} &\phi_0 \sim \mathcal{U}(-\pi/3, \pi/3), \quad \psi_0 \sim \mathcal{U}(0, 2\pi), \\ &U_r \sim \mathcal{U}(0,1), \quad \theta \sim \mathcal{U}(0, 2\pi) \end{aligned} \tag{34}$$

These four randomizations, initial latitude (ϕ_0), initial heading (ψ_0), target radial distance (U_r), and target bearing (θ), allow the policy to provide precision guidance to arbitrary geographic coordinates without retraining for different locations.

The reward function from Equations (25)–(30) naturally accommodates both operational modes. For unreachable targets beyond the maximum range, the policy maximizes cumulative distance reduction from Equation (27), generating footprint boundary trajectories. For reachable targets within the footprint, the identical distance-reduction signal guides convergence to target coordinates, with discount factor of 0.99 favoring trajectories that reach the target in fewer steps. The policy learns adaptive control strategies purely from the distance-reduction reward without explicit mode switching.

3.5.1. No-Fly Zone Avoidance

For scenarios requiring the avoidance of prohibited airspace regions, the state space is expanded to seven dimensions $\mathcal{S}_{\text{NFZ}} \subseteq \mathbb{R}^7$ by appending the normalized difference between heading and bearing toward the NFZ center $(\bar{\kappa}_{\text{NFZ}})$ and the normalized distance to the NFZ boundary edge $(\bar{d}_{\text{NFZ}})$, providing the agent with spatial awareness of prohibited regions:

$$\mathbf{s}_t^{\text{NFZ}} = \begin{bmatrix} \bar{h}_t & \bar{V}_t & \bar{\gamma}_t & \bar{\kappa}_t & \bar{d}_t & \bar{\kappa}_{\text{NFZ},t} & \bar{d}_{\text{NFZ},t} \end{bmatrix}^{\mathsf{T}} \in \mathbb{R}^7 \tag{35}$$

Each NFZ is represented as a circular exclusion region defined by center coordinates $(\phi_{\text{NFZ}}, \lambda_{\text{NFZ}})$ and radius $r_{\text{NFZ}} = 500$ km. The NFZ-relative observables

are $\bar{\kappa}_{\text{NFZ}}$ and $\bar{d}_{\text{NFZ}}$, computed using the same spherical trigonometry operations as the measurements relative to the target. These observables provide the agent with spatial awareness of the location and proximity of the prohibited region.

An additional binary penalty applies when the vehicle enters prohibited regions, where the penalty magnitude was determined through iterative tuning to balance avoidance with range objectives:

$$r_t^{\text{NFZ}} = \begin{cases} -2.0 & \text{if } d_{\text{NFZ},t} < 0 \\ 0 & \text{otherwise} \end{cases} \tag{36}$$

where $d_{\text{NFZ},t}$ is the distance to the NFZ boundary edge. The total reward becomes:

$$r_t^{\text{total}} = r_t^{\text{progress}} + r_t^{\text{constraint}} + r_t^{\text{NFZ}} \tag{37}$$

During training, both target coordinates and NFZ centers are sampled using the circular region approach described in Section 3.5. The randomized variables are

$$\begin{aligned} \phi_0 &\sim \mathcal{U}(-\pi/3, \pi/3), \quad \psi_0 \sim \mathcal{U}(0, 2\pi) \\ U_r^{\text{target}} &\sim \mathcal{U}(0,1), \quad \theta^{\text{target}} \sim \mathcal{U}(0, 2\pi) \\ U_r^{\text{NFZ}} &\sim \mathcal{U}(0,1), \quad \theta^{\text{NFZ}} \sim \mathcal{U}(0, 2\pi) \end{aligned} \tag{38}$$

These six randomizations: initial latitude (ϕ_0), initial heading (ψ_0), target radial distance (U_r^{target}), target bearing (θ^{target}), NFZ radial distance (U_r^{NFZ}), and NFZ bearing (θ^{NFZ}), allow the policy to learn generalized avoidance strategies and direct target point guidance across arbitrary geometric configurations without retraining.

4. Results and Discussion

This section presents the performance of trained reinforcement learning policies for the HTV-2 [36] and DF-ZF [37] configurations. The analysis demonstrates footprint generation, direct target point guidance, no-fly zone avoidance, Monte Carlo validation, and sensitivity analysis results.

4.1. Footprint Generation

Complete footprints were generated by executing trained policies for 36 uniformly distributed unreachable target directions from latitude 0°, longitude 0°, and initial heading 90°. Figure 2 shows the resulting HTV-2 footprint with a stereographic projection.

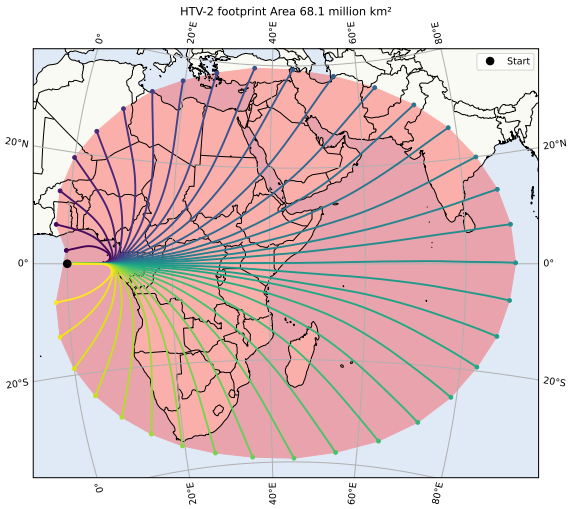


Figure 2: Stereographic projection map showing HTV-2 footprint with 36 trajectories and boundary outlined.

The DF-ZF footprint was generated using the same methodology with 36 uniformly distributed unreachable target directions from latitude 0°, longitude 0°, and initial heading 90°. Figure 3 shows the resulting footprint with a stereographic projection.

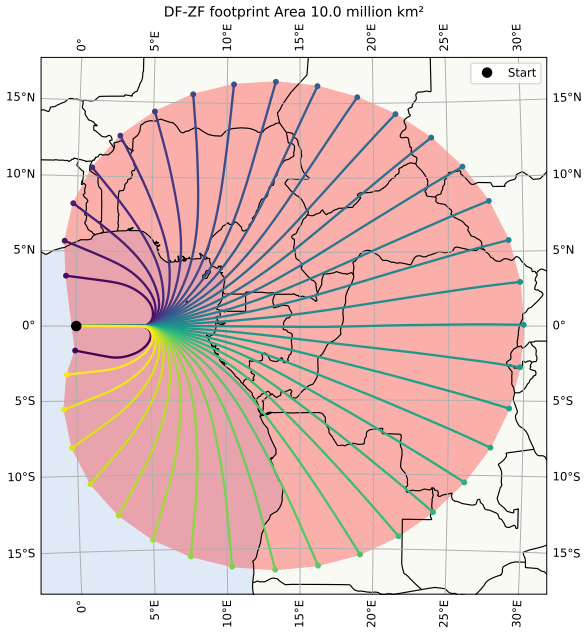


Figure 3: Stereographic projection map showing DF-ZF footprint with 36 trajectories and boundary outlined.

Figure 4 shows the three-dimensional trajectory

structure for the HTV-2 configuration with the same initial and terminal conditions. The altitude profiles clearly reveal the skip glide behavior characteristic of hypersonic atmospheric flight, where the vehicle exchanges kinetic energy with potential energy throughout the glide phase.

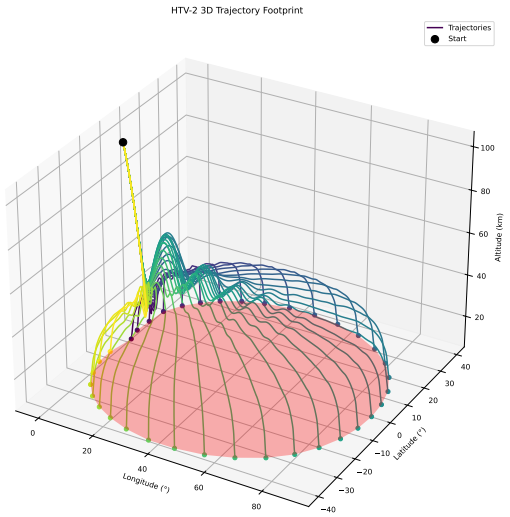


Figure 4: Three-dimensional visualization of HTV-2 footprint showing skip glide behavior across 36 trajectories.

The initial conditions and footprint metrics for both vehicles are given in Table 2.

Table 2: Footprint performance metrics for HTV-2 and DF-ZF configurations

Parameter	**HTV-2**	**DF-ZF**
Initial Conditions		
Altitude (km)	100	100
Velocity (m/s)	6000	3450
Flight path angle (°)	−3	0
Footprint Metrics		
Footprint area (million km^2)	68.15	10.04
Maximum range (km)	9923	3335
Maximum cross-track (km)	4431	1829

The HTV-2 footprint area of 68.15 million km^2 is approximately seven times larger compared to the DF-ZF footprint area of 10.04 million km^2, directly reflecting the initial energy difference. Both footprints exhibit north-south symmetry, consistent with Coriolis compensation at equatorial launch. Trajectories heading northeast and southeast produce symmetric endpoints despite opposite Coriolis deflections, demonstrating successful compensation without geographic coordinates in the observation space. Maximum cross-track distances of 4431 km (HTV-2) and 1829 km (DF-ZF) demonstrate substantial lateral maneuverability. The cross-track-to-range ratios of 0.447 (HTV-2) and 0.548 (DF-ZF) indicate that both configurations can reach targets displaced up to 44 − 55% of maximum range perpendicular to the initial heading. The footprint boundary represents the maximum reachable envelope, though a small region near the initial position remains unreachable due to minimum range constraints. Consequently, the actual reachable area is slightly smaller than the values reported in Table 2. Monte Carlo validation and direct target point guidance attempts confirmed this unreachable zone. All 36 trajectories for both vehicles terminated at $h = 5$ km without constraint violations, confirming that the exponential penalty function successfully enforced load factor, dynamic pressure and thermal limits throughout the flight envelope. Footprints can be generated within seconds,

4.2. Direct Target Point Guidance

The trained policy provides precision guidance to arbitrary geographic coordinates within the operational footprint. Validation demonstrates the navigation of the HTV-2 HGV from $(\phi_0, \lambda_0) = (52°N, 5°E)$ with initial heading $\psi_0 = 230°$ to target coordinates $(\phi_t, \lambda_t) = (40°N, 15°E)$. The initial conditions are $h_0 = 100$ km, $V_0 = 6000$ m/s, $\gamma_0 = -3°$. The great-circle distance is $d = 1558$ km. Figure 5 shows the ground track.

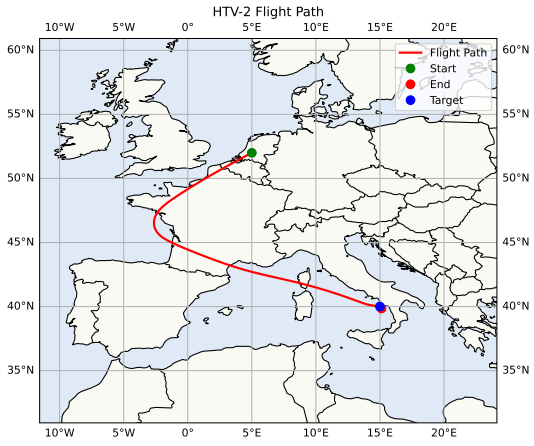


Figure 5: Direct target point guidance trajectory from $(52°N, 5°E)$ to $(40°N, 15°E)$.

Terminal positioning accuracy is $\epsilon_t = 19.4$ km, corresponding to 1.2% of total distance. This accuracy is sufficient for mid-course guidance, with terminal guidance systems handling the final approach to the tar-

get. All path constraints remain satisfied throughout the flight duration. Complete trajectory generation requires $t_{\text{comp}} < 0.1$ s.

4.3. No-Fly Zone Avoidance

The trained policy was evaluated for target point navigation with circular no-fly zones of radius $r_{\text{NFZ}} = 500$ km. Figure 6 shows navigation from $(\phi_0, \lambda_0) = (0°, 0°)$ with initial heading $\psi_0 = 90°$ to target $(\phi_t, \lambda_t) = (0°, 52°E)$ with NFZ centered at $(0°, 30°E)$. The initial conditions are $h_0 = 100$ km, $V_0 = 6000$ m/s, $\gamma_0 = -3°$.

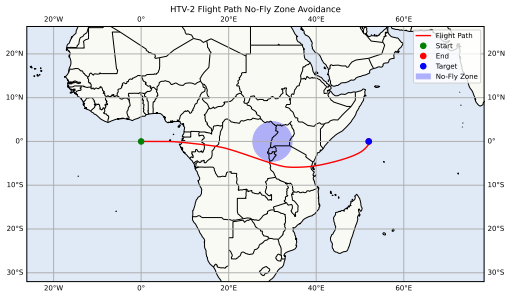


Figure 6: No-fly zone avoidance trajectory showing detour routing around 500 km radius exclusion zone.

The great-circle distance is $d = 5781$ km with terminal positioning accuracy $\epsilon_t = 1.83$ km (0.03% of distance). The vehicle maintains clearance from the NFZ boundary while converging to the target coordinates. All path constraints remain satisfied throughout the trajectory. Complete trajectory generation requires $t_{\text{comp}} < 0.1$ s. NFZ avoidance is subject to physical feasibility constraints, where NFZ positions too close to the initial state or target may be unavoidable.

4.4. Monte Carlo Validation

To validate that the RL policy discovered the true footprint boundaries instead of converging to a local optimum, a Monte Carlo simulation with 100 million random trajectory evaluations was performed. Control inputs were uniformly sampled from the operational ranges $\alpha \in [-4°, 60°]$ and $\sigma \in [-90°, 90°]$ with update intervals randomly distributed between 5 and 1200 seconds. Rate limiters enforced physical constraints $|\dot{\alpha}| \leq 1°/\text{s}$ and $|\dot{\sigma}| \leq 3°/\text{s}$. Trajectories violating any of the constraints were immediately terminated and discarded. Figure 7 shows the resulting Monte Carlo footprint.

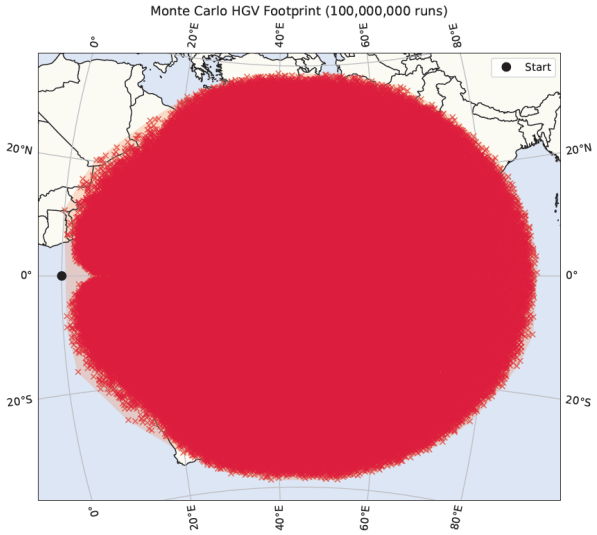


Figure 7: Monte Carlo footprint generated from 100 million random trajectories.

Table 3 compares Monte Carlo and RL footprint metrics for the HTV-2 configuration. The RL policy exceeds Monte Carlo performance in all metrics, with footprint area larger by 5.1%, maximum range by 0.38%, and maximum cross-track by 2.3%. The RL policy achieving superior performance demonstrates successful discovery of optimal control strategies rather than convergence to local optima.

Table 3: Monte Carlo validation comparing 100 million random trajectories with RL policy footprints

Metric	Monte Carlo	RL Policy	Difference
Footprint area (million km^2)	64.83	68.15	+5.1%
Maximum range (km)	9885	9923	+0.38%
Maximum cross-track (km)	4332	4431	+2.3%

4.5. Sensitivity Analysis

The robustness of the trained policy to parameter variations was quantified through systematic perturbation analysis. Two parameter categories were examined: initial flight conditions (altitude, velocity, flight path angle) and vehicle properties (mass, reference area). During training, the policy experienced fixed initial conditions. The sensitivity analysis evaluates generalization beyond the training distribution. Table 4 summarizes the variations in the footprint area for representative perturbations in each parameter.

Initial velocity exhibits dominant sensitivity with ±100 m/s perturbations producing −8.8% to +9.0% area variations, reflecting the quadratic kinetic energy relationship. Flight path angle variations of ±0.5° reduced area by 0.9% for negative perturbations, while positive

Table 4: Sensitivity analysis showing footprint area response to parameter variations for HTV-2 configuration.

Parameter	Variation	Footprint Area (million km^2)	Change
	Initial Flight Conditions		
Altitude	−10 km	68.16	+0.01%
	Nominal (100 km)	68.15	-
	+10 km	68.11	−0.06%
Velocity	−100 m/s	62.16	−8.8%
	Nominal (6000 m/s)	68.15	-
	+100 m/s	74.27	+9.0%
Flight path angle	−0.5°	67.54	−0.9%
	Nominal (−3°)	68.15	-
	+0.5°	68.63	+0.7%
	Vehicle Properties		
Mass	−10%	68.12	−0.04%
	Nominal (2184 kg)	68.15	-
	+10%	68.09	−0.09%
Reference area	−10%	68.08	−0.10%
	Nominal (4.4 m^2)	68.15	-
	+10%	68.12	−0.04%

perturbations increased area by 0.7%. Altitude variations of ±10 km produced negligible effects (< 0.1%). Vehicle property variations (mass ±10%, reference area ±10%) similarly produced changes below 0.1%.

Figure 8 shows clear separation between footprint boundaries for the three velocity cases, with proportional scaling maintaining symmetric shapes without directional bias.

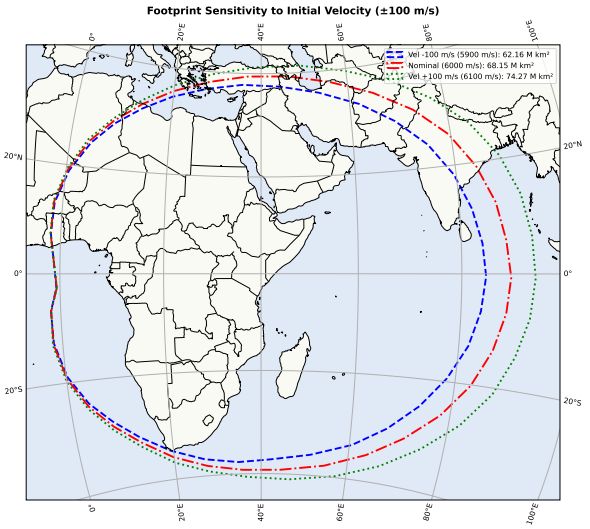


Figure 8: Footprint boundaries for initial velocity variations showing proportional area scaling.

Figure 9 compares trajectory characteristics for velocity perturbations against a fixed baseline trajectory operating at maximum lift-to-drag ratio ($\alpha = 12°, \sigma = 0°$). The baseline shows sustained skip-glide oscillations throughout the trajectory, resulting in elevated thermal and mechanical loads. The trained policy converges to quasi-steady flight, demonstrating reduced g-load factors, surface temperatures, and dynamic pressure variations compared to the baseline across all velocity cases.

5. Conclusion

This work presented the first reinforcement learning approach for hypersonic glide vehicle footprint generation combining dual control authority over bank angle and angle of attack with integrated no-fly zone avoidance and direct target point guidance. The Soft Actor-Critic algorithm successfully learned optimal control policies that generate complete reachable footprints in seconds while satisfying all operational constraints including load factor, dynamic pressure, and thermal limits. Monte Carlo validation with 100 million random trajectories confirmed the RL policy discovered true footprint boundaries, exceeding random sampling performance by 5.1% in area, maximum range by 0.38%, and maximum cross-track by 2.3%.

Key contributions include elimination of equilibrium glide assumptions, fixed angle of attack profiles, and heuristic boundary determination methods that limit existing approaches. The trained policy generates complete footprints in seconds for the HTV-2 and DF-ZF HGV configurations. The framework provides precision target guidance and no-fly zone avoidance with computation times below 0.1 second. Sensitivity analysis revealed initial velocity as the dominant parameter, with ±100 m/s perturbations producing −8.8% to 9.0% footprint area variations. All other parameters including altitude (±10 km), flight path angle (±0.5°), mass (±10%), and reference area (±10%) produced footprint area changes below 1%. The current implementation

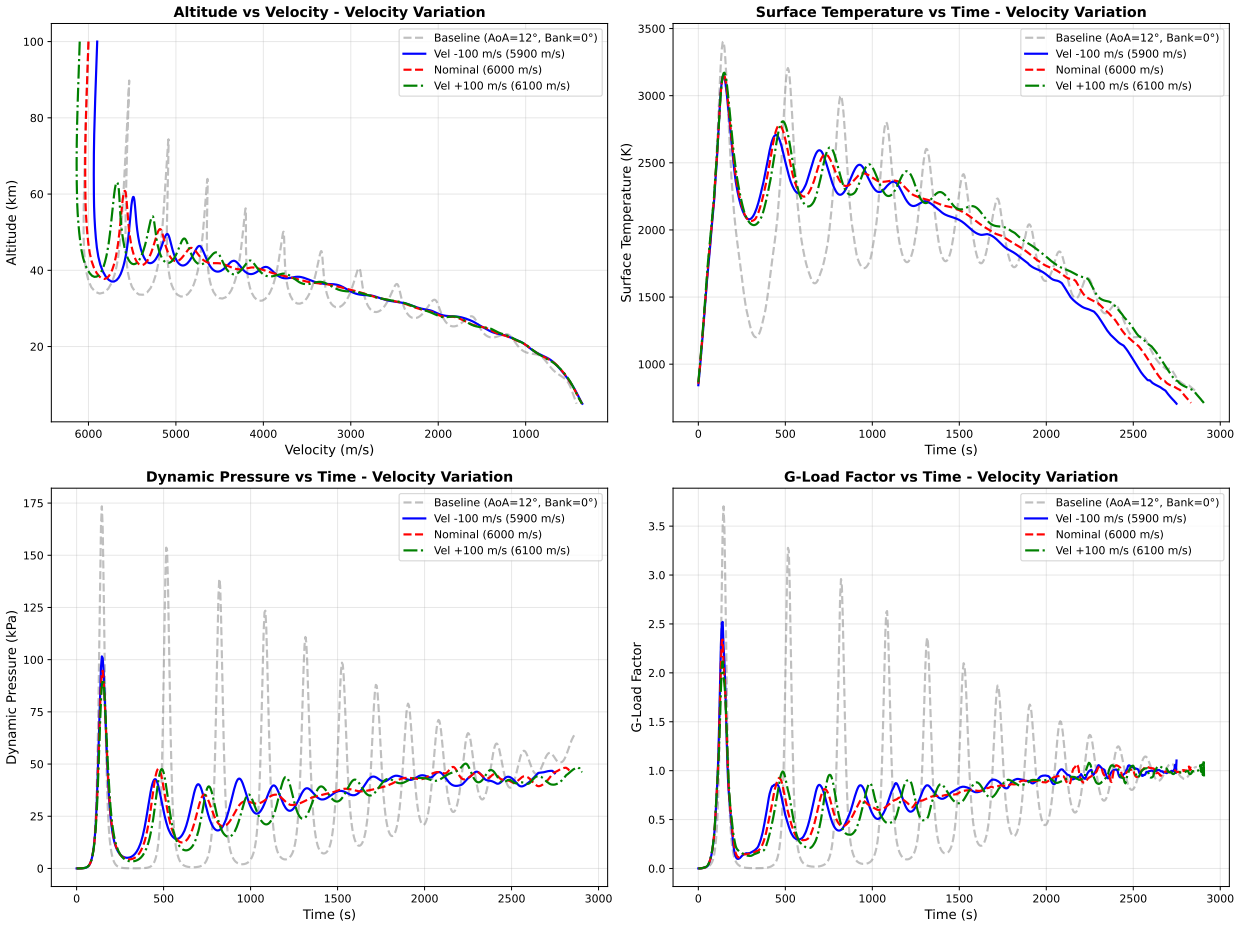


Figure 9: Sensitivity analysis of trajectory characteristics for initial velocity variations (±100 m/s) comparing trained policy performance against fixed maximum L/D baseline.

uses a three-degree-of-freedom point-mass model and assumes a spherical Earth with standard atmospheric conditions. Future work should extend the framework to six-degree-of-freedom dynamics to capture rotational effects and moments and incorporate atmospheric uncertainties and wind profiles. The framework's modular architecture enables future extensions including scramjet propulsion for boost-glide trajectories and multi-waypoint constrained navigation. This work demonstrates that reinforcement learning provides an effective approach for complete hypersonic glide vehicle footprint generation.

Data Availability

The data that support the findings of this study are not publicly available due to institutional policy. Requests for access should be directed to the corresponding author.

References

[1] Y. Shi, Z. Wang, Onboard generation of optimal trajectories for hypersonic vehicles using deep learning, Journal of Spacecraft and Rockets 58 (2) (2021) 400–414.

[2] K. An, Z. Guo, X. Xu, W. Huang, A framework of trajectory design and optimization for the hypersonic gliding vehicle, Aerospace Science and Technology 106 (2020) 106110.

[3] A. Abbas, Hypersonic weapons and the future of strategic stability between the nuclear rivals, Journal of Security and Strategic Analyses 10 (2) (2024) 59–78.

[4] A. Saraf, J. Leavitt, M. Ferch, K. Mease, Landing footprint computation for entry vehicles, in: AIAA Guidance, Navigation, and Control Conference and Exhibit, 2004, p. 4774.

[5] R. Chai, A. Savvaris, A. Tsourdos, S. Chai, Y. Xia, A review of optimization techniques in spacecraft flight trajectory design, Progress in aerospace sciences 109 (2019) 100543.

[6] Z. Wang, M. J. Grant, Hypersonic trajectory optimization by sequential semidefinite programming, in: AIAA Atmospheric Flight Mechanics Conference, 2017, p. 0248.

[7] M. J. Grant, R. D. Braun, Rapid indirect trajectory optimization for conceptual design of hypersonic missions, Journal of Spacecraft and Rockets 52 (1) (2015) 177–182.

[8] M. J. Grant, M. A. Bolender, Rapid, robust trajectory design using indirect optimization methods, in: AIAA Atmospheric Flight Mechanics Conference, 2015, p. 2401.

[9] Y. Zheng, H. Cui, Y. Ai, Indirect trajectory optimization for mars entry with maximum terminal altitude, Journal of Spacecraft and Rockets 54 (5) (2017) 1068–1080.

[10] J. Z. Ben-Asher, Optimal control theory with aerospace applications, American Institute of Aeronautics and Astronautics, 2010.

[11] G. Huang, Y. Lu, Y. Nan, A survey of numerical algorithms for trajectory optimization of flight vehicles, Science China Technological Sciences 55 (2012) 2538–2560.

[12] Z. Wang, Y. Lu, Improved sequential convex programming algorithms for entry trajectory optimization, Journal of Spacecraft and Rockets 57 (6) (2020) 1373–1386.

[13] P. Lu, Entry guidance and trajectory control for reusable launch vehicle, Journal of Guidance, Control, and Dynamics 20 (1) (1997) 143–149.

[14] F. Fahroo, I. M. Ross, Direct trajectory optimization by a chebyshev pseudospectral method, Journal of Guidance, Control, and Dynamics 25 (1) (2002) 160–166.

[15] J. Schierman, J. Hull, In-flight entry trajectory optimization for reusable launch vehicles, in: AIAA Guidance, Navigation, and Control Conference and Exhibit, 2005, p. 6434.

[16] T. R. Jorris, R. G. Cobb, Three-dimensional trajectory optimization satisfying waypoint and no-fly zone constraints, Journal of Guidance, Control, and Dynamics 32 (2) (2009) 551–572.

[17] L. Zhang, M. Sun, Z. Chen, Z. Wang, Y. Wang, Receding horizon trajectory optimization with terminal impact specifications, Mathematical Problems in Engineering 2014 (1) (2014) 604705.

[18] L. Tu, J. Yuan, Reentry trajectory optimization using direct collocation method and nonlinear programming, in: 57th International Astronautical Congress, 2006, pp. C1–4.

[19] X. Liu, Z. Shen, P. Lu, Entry trajectory optimization by second-order cone programming, Journal of Guidance, Control, and Dynamics 39 (2) (2016) 227–241.

[20] M. Sagliano, E. Mooij, Optimal drag-energy entry guidance via pseudospectral convex optimization, Aerospace Science and Technology 117 (2021) 106946.

[21] N. X. Vinh, Optimal trajectories in atmospheric flight, in: Space: Mankind's Fourth Environment, Elsevier, 1982, pp. 449–468.

[22] A. Ngo, D. Doman, Footprint determination for reusable launch vehicles experiencing control effector failures, in: AIAA Guidance, Navigation, and Control Conference and Exhibit, 2002, p. 4775.

[23] D. Izzo, M. Märtens, B. Pan, A survey on artificial intelligence trends in spacecraft guidance dynamics and control, Astrodynamics 3 (4) (2019) 287–299.

[24] P. Razzaghi, A. Tabrizian, W. Guo, S. Chen, A. Taye, E. Thompson, P. Wei, A survey on reinforcement learning in aviation applications. arxiv 2022, arXiv preprint arXiv:2211.02147 (2023).

[25] D. Tianbo, H. Huang, F. Yangwang, Y. Jie, H. Cheng, Reinforcement learning-based missile terminal guidance of maneuvering targets with decoys, Chinese Journal of Aeronautics 36 (12) (2023) 309–324.

[26] W. Haijiao, Y. Zhen, Z. Wugen, L. Dalin, Online scheduling of image satellites based on neural networks and deep reinforcement learning, Chinese Journal of Aeronautics 32 (4) (2019) 1011–1019.

[27] S. Li, Y. Yan, H. Qiao, X. Guan, X. Li, Reinforcement learning for computational guidance of launch vehicle upper stage, International Journal of Aerospace Engineering 2022 (1) (2022) 2935929.

[28] X. Xu, Y. Chen, C. Bai, Deep reinforcement learning-based accurate control of planetary soft landing, Sensors 21 (23) (2021) 8161.

[29] B. Gaudet, R. Linares, R. Furfaro, Deep reinforcement learning for six degree-of-freedom planetary landing, Advances in Space Research 65 (7) (2020) 1723–1741.

[30] R. Chai, A. Tsourdos, A. Savvaris, Y. Xia, S. Chai, Real-time reentry trajectory planning of hypersonic vehicles: A two-step strategy incorporating fuzzy multiobjective transcription and deep neural network, IEEE Transactions on Industrial Electronics 67 (8) (2019) 6904–6915.

[31] T. Haarnoja, A. Zhou, P. Abbeel, S. Levine, Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor, in: International conference on machine learning, Pmlr, 2018, pp. 1861–1870.

[32] J. Gao, X. Shi, Z. Cheng, J. Xiong, L. Liu, Y. Wang, Y. Yang, Reentry trajectory optimization based on deep reinforcement learning, in: 2019 Chinese Control And Decision Conference (CCDC), IEEE, 2019, pp. 2588–2592.

[33] L. Su, J. Wang, H. Chen, A real-time and optimal hypersonic entry guidance method using inverse reinforcement learning, Aerospace 10 (11) (2023) 948.

[34] P. P. Das, W. Pei, C. Niu, Reentry trajectory design of a hypersonic vehicle based on reinforcement learning, in: Journal of Physics: Conference Series, Vol. 2633, IOP Publishing, 2023, p. 012005.

[35] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, N. Dormann, Stable-baselines3: Reliable reinforcement learning implementations, Journal of machine learning research 22 (268) (2021) 1–8.

[36] S. Walker, J. Sherk, D. Shell, R. Schena, J. Bergmann, J. Gladbach, The darpa/af falcon program: The hypersonic technology vehicle# 2 (htv-2) flight demonstration phase, in: 15th AIAA International Space Planes and Hypersonic Systems and Technologies Conference, 2008, p. 2539.

[37] A. Pratap Singh, V. Choudhary, D. Sharma, Design and development of hypersonic aerial vehicles, in: Advances in Interdisciplinary Engineering: Select Proceedings of FLAME 2020, Springer, 2021, pp. 457–467.

References

[1] Y. Shi and Z. Wang, “Onboard generation of optimal trajectories for hypersonic vehicles using deep learning,” *Journal of Spacecraft and Rockets*, vol. 58, no. 2, pp. 400–414, 2021.

[2] K. An, Z. Guo, X. Xu, and W. Huang, “A framework of trajectory design and optimization for the hypersonic gliding vehicle,” *Aerospace Science and Technology*, vol. 106, p. 106 110, 2020.

[3] A. Abbas, “Hypersonic weapons and the future of strategic stability between the nuclear rivals,” *Journal of Security and Strategic Analyses*, vol. 10, no. 2, pp. 59–78, 2024.

[4] G. Li, H. Zhang, and G. Tang, “Flight-corridor analysis for hypersonic glide vehicles,” *Journal of Aerospace Engineering*, vol. 30, no. 1, p. 06 016 005, 2017.

[5] A. Saraf, J. Leavitt, M. Ferch, and K. Mease, “Landing footprint computation for entry vehicles,” in *AIAA Guidance, Navigation, and Control Conference and Exhibit*, 2004, p. 4774.

[6] F. Nyland, *Hypersonic turning with constant bank angle control, the rand corporation*, 1965.

[7] Z. Wang and M. J. Grant, “Hypersonic trajectory optimization by sequential semidefinite programming,” in *AIAA Atmospheric Flight Mechanics Conference*, 2017, p. 0248.

[8] M. J. Grant and R. D. Braun, “Rapid indirect trajectory optimization for conceptual design of hypersonic missions,” *Journal of Spacecraft and Rockets*, vol. 52, no. 1, pp. 177–182, 2015.

[9] M. J. Grant and M. A. Bolender, “Rapid, robust trajectory design using indirect optimization methods,” in *AIAA Atmospheric Flight Mechanics Conference*, 2015, p. 2401.

[10] Y. Zheng, H. Cui, and Y. Ai, “Indirect trajectory optimization for mars entry with maximum terminal altitude,” *Journal of Spacecraft and Rockets*, vol. 54, no. 5, pp. 1068–1080, 2017.

[11] J. Z. Ben-Asher, *Optimal control theory with aerospace applications*. American Institute of Aeronautics and Astronautics, 2010.

[12] G. Huang, Y. Lu, and Y. Nan, “A survey of numerical algorithms for trajectory optimization of flight vehicles,” *Science China Technological Sciences*, vol. 55, pp. 2538–2560, 2012.

[13] M. J. Grant and M. A. Bolender, “Minimum terminal energy optimizations of hypersonic vehicles using indirect methods,” in *AIAA Atmospheric Flight Mechanics Conference*, 2015, p. 2402.

[14] Z. Wang and Y. Lu, “Improved sequential convex programming algorithms for entry trajectory optimization,” *Journal of Spacecraft and Rockets*, vol. 57, no. 6, pp. 1373–1386, 2020.

[15] P. Lu, “Entry guidance and trajectory control for reusable launch vehicle,” *Journal of Guidance, Control, and Dynamics*, vol. 20, no. 1, pp. 143–149, 1997.

[16] F. Fahroo and I. M. Ross, “Direct trajectory optimization by a chebyshev pseudospectral method,” *Journal of Guidance, Control, and Dynamics*, vol. 25, no. 1, pp. 160–166, 2002.

[17] J. Schierman and J. Hull, “In-flight entry trajectory optimization for reusable launch vehicles,” in *AIAA Guidance, Navigation, and Control Conference and Exhibit*, 2005, p. 6434.

[18] T. R. Jorris and R. G. Cobb, “Three-dimensional trajectory optimization satisfying waypoint and no-fly zone constraints,” *Journal of Guidance, Control, and Dynamics*, vol. 32, no. 2, pp. 551–572, 2009.

[19] L. Zhang, M. Sun, Z. Chen, Z. Wang, and Y. Wang, “Receding horizon trajectory optimization with terminal impact specifications,” *Mathematical Problems in Engineering*, vol. 2014, no. 1, p. 604 705, 2014.

[20] L. Tu and J. Yuan, “Reentry trajectory optimization using direct collocation method and nonlinear programming,” in *57th International Astronautical Congress*, 2006, pp. C1–4.

[21] X. Liu, Z. Shen, and P. Lu, “Entry trajectory optimization by second-order cone programming,” *Journal of Guidance, Control, and Dynamics*, vol. 39, no. 2, pp. 227–241, 2016.

[22] M. Sagliano and E. Mooij, “Optimal drag-energy entry guidance via pseudospectral convex optimization,” *Aerospace Science and Technology*, vol. 117, p. 106 946, 2021.

[23] N. X. Vinh, “Optimal trajectories in atmospheric flight,” in *Space: Mankind’s Fourth Environment*, Elsevier, 1982, pp. 449–468.

[24] A. Ngo and D. Doman, “Footprint determination for reusable launch vehicles experiencing control effector failures,” in *AIAA Guidance, Navigation, and Control Conference and Exhibit*, 2002, p. 4775.

[25] R. Chai, A. Tsourdos, A. Savvaris, Y. Xia, and S. Chai, “Real-time reentry trajectory planning of hypersonic vehicles: A two-step strategy incorporating fuzzy multiobjective transcription and deep neural network,” *IEEE Transactions on Industrial Electronics*, vol. 67, no. 8, pp. 6904–6915, 2019.

[26] D. Izzo, M. Märtens, and B. Pan, “A survey on artificial intelligence trends in spacecraft guidance dynamics and control,” *Astrodynamics*, vol. 3, no. 4, pp. 287–299, 2019.

[27] P. Razzaghi *et al.*, “A survey on reinforcement learning in aviation applications. arxiv 2022,” *arXiv preprint arXiv:2211.02147*, 2023.

[28] D. Tianbo, H. Huang, F. Yangwang, Y. Jie, and H. Cheng, “Reinforcement learning-based missile terminal guidance of maneuvering targets with decoys,” *Chinese Journal of Aeronautics*, vol. 36, no. 12, pp. 309–324, 2023.

[29] W. Haijiao, Y. Zhen, Z. Wugen, and L. Dalin, “Online scheduling of image satellites based on neural networks and deep reinforcement learning,” *Chinese Journal of Aeronautics*, vol. 32, no. 4, pp. 1011–1019, 2019.

[30] S. Li, Y. Yan, H. Qiao, X. Guan, and X. Li, “Reinforcement learning for computational guidance of launch vehicle upper stage,” *International Journal of Aerospace Engineering*, vol. 2022, no. 1, p. 2 935 929, 2022.

[31] X. Xu, Y. Chen, and C. Bai, “Deep reinforcement learning-based accurate control of planetary soft landing,” *Sensors*, vol. 21, no. 23, p. 8161, 2021.

[32] B. Gaudet, R. Linares, and R. Furfaro, “Deep reinforcement learning for six degree-of-freedom planetary landing,” *Advances in Space Research*, vol. 65, no. 7, pp. 1723–1741, 2020.

[33] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, “Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor,” in *International conference on machine learning*, Pmlr, 2018, pp. 1861–1870.

[34] J. Gao *et al.*, “Reentry trajectory optimization based on deep reinforcement learning,” in *2019 Chinese Control And Decision Conference (CCDC)*, IEEE, 2019, pp. 2588–2592.

[35] L. Su, J. Wang, and H. Chen, “A real-time and optimal hypersonic entry guidance method using inverse reinforcement learning,” *Aerospace*, vol. 10, no. 11, p. 948, 2023.

[36] P. P. Das, W. Pei, and C. Niu, “Reentry trajectory design of a hypersonic vehicle based on reinforcement learning,” in *Journal of Physics: Conference Series*, IOP Publishing, vol. 2633, 2023, p. 012 005.

[37] H. Xu, G. Cai, C. Mu, and X. Li, “Analytical reentry guidance framework based on swarm intelligence optimization and altitude-energy profile,” *Chinese Journal of Aeronautics*, vol. 36, no. 12, pp. 336–348, 2023.

[38] E. Mooij, *Re-entry Systems*. Springer Nature, 2024.

[39] R. Chai, K. Chen, L. Cui, S. Chai, G. Inalhan, and A. Tsourdos, *Advanced Trajectory Optimization, Guidance and Control Strategies for Aerospace Vehicles*. Springer, 2023.

[40] J. T. Betts, “Survey of numerical methods for trajectory optimization,” *Journal of Guidance, Control, and Dynamics*, vol. 21, no. 2, pp. 193–207, 1998.

[41] V. Adimurthy, “Launch vehicle trajectory optimization including rotational dynamics,” *Journal of Spacecraft and Rockets*, vol. 13, no. 1, pp. 59–61, 1976.

[42] V. Adimurthy, “Launch vehicle trajectory optimization,” *Acta Astronautica*, vol. 15, no. 11, pp. 845–850, 1987.

[43] Y. Li, N. Cui, and S. Rong, “Trajectory optimization for hypersonic boost-glide missile considering aeroheating,” *Aircraft Engineering and Aerospace Technology*, vol. 81, no. 1, pp. 3–13, 2009.

[44] M. Zhang, Y. Sun, G. Duan, and G. Wang, “Reentry trajectory optimization of hypersonic vehicle with minimum heat,” in *8th World Congress on Intelligent Control and Automation*, IEEE, 2010, pp. 1764–1769.

[45] H. M. Prasanna, D. Ghose, M. S. Bhat, C. Bhattacharyya, and J. Umakant, “Ascent phase trajectory optimization for a hypersonic vehicle using nonlinear programming,” in *International Conference on Computational Science and Its Applications*, Springer, 2005, pp. 548–557.

[46] C. Mastalli *et al.*, “A feasibility-driven approach to control-limited ddp,” *Autonomous Robots*, vol. 46, no. 8, pp. 985–1005, 2022.

[47] Y. Mao, D. Zhang, and L. Wang, “Reentry trajectory optimization for hypersonic vehicle based on improved gauss pseudospectral method,” *Soft Computing*, vol. 21, pp. 4583–4592, 2017.

[48] X. Yan, S. Lyu, and S. Tang, “Analysis of optimal initial glide conditions for hypersonic glide vehicles,” *Chinese Journal of Aeronautics*, vol. 27, no. 2, pp. 217–225, 2014.

[49] H. Seywald, “Trajectory optimization based on differential inclusion,” Tech. Rep., 1993.

[50] S. Peña Luque, *Optimal guidance for space applications*, Unpublished manuscript, 2009.

[51] G. Naresh Kumar, M. Ikram, A. K. Sarkar, and S. E. Talole, “Hypersonic flight vehicle trajectory optimization using pattern search algorithm,” *Optimization and Engineering*, vol. 19, pp. 125–161, 2018.

[52] N. Yokoyama and S. Suzuki, “Trajectory optimization via modified genetic algorithm,” in *AIAA Guidance, Navigation, and Control Conference and Exhibit*, 2003, p. 5493.

[53] G. Chen, M. Xu, Z. Wan, and S. Chen, “RLV reentry trajectory multi-objective optimization design based on NSGA-II algorithm,” in *AIAA Atmospheric Flight Mechanics Conference and Exhibit*, 2005, p. 6131.

[54] P. Aivaliotis-Apostolopoulos and D. Loukidis, “Swarming genetic algorithm: A nested fully coupled hybrid of genetic algorithm and particle swarm optimization,” *PLoS One*, vol. 17, no. 9, e0275094, 2022.

[55] M. Vavrina and K. Howell, “Global low-thrust trajectory optimization through hybridization of a genetic algorithm and a direct method,” in *AIAA/AAS Astrodynamics Specialist Conference and Exhibit*, 2008, p. 6614.

[56] A. Rahimi, K. Dev Kumar, and H. Alighanbari, “Particle swarm optimization applied to spacecraft reentry trajectory,” *Journal of Guidance, Control, and Dynamics*, vol. 36, no. 1, pp. 307–310, 2013.

[57] M. Li, W. Du, and F. Nian, “An adaptive particle swarm optimization algorithm based on directed weighted complex network,” *Mathematical problems in engineering*, vol. 2014, no. 1, p. 434 972, 2014.

[58] Y. Liu, H. Zhang, H. Zheng, Q. Li, and Q. Tian, “A spherical vector-based adaptive evolutionary particle swarm optimization for uav path planning under threat conditions,” *Scientific Reports*, vol. 15, no. 1, p. 2116, 2025.

[59] I. M. Ross, “Enhancements to the DIDO optimal control toolbox,” *arXiv preprint arXiv:2004.13112*, 2020.

[60] M. Elhesasy *et al.*, “Non-linear model predictive control using CasADi package for trajectory tracking of quadrotor,” *Energies*, vol. 16, no. 5, p. 2143, 2023.

[61] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, “CasADi: A software framework for nonlinear optimization and optimal control,” *Mathematical Programming Computation*, vol. 11, pp. 1–36, 2019.

[62] M. A. Patterson and A. V. Rao, “GPOPS-II: A MATLAB software for solving multiple-phase optimal control problems using hp-adaptive gaussian quadrature collocation methods and sparse nonlinear programming,” *ACM Transactions on Mathematical Software (TOMS)*, vol. 41, no. 1, pp. 1–37, 2014.

[63] J. T. Betts, *Practical Methods for Optimal Control and Estimation Using Nonlinear Programming*. SIAM, 2010.

[64] D. E. Yaple, *Simulation and application of GPOPS for trajectory optimization and mission planning tool*, Unpublished report, 2010.

[65] T. Rahman, H. Zhou, Y. Z. Sheng, Y. M. Younis, and K. N. Zhang, "Trajectory optimization of hypersonic vehicle using gauss pseudospectral method," *Applied Mechanics and Materials*, vol. 110, pp. 5232–5239, 2012.

[66] R. He, L. Liu, G. Tang, and W. Bao, "Footprint determination for entry vehicles based on three-dimensional acceleration profile," in *36th Chinese Control Conference (CCC)*, IEEE, 2017, pp. 5754–5759.

[67] M. Stephane, "Implementation of a simulated annealing algorithm for MATLAB," *Technical Report*, vol. 1, pp. 1–34, 2002.

[68] MathWorks, *What is simulated annealing?* Accessed 2025-06-17, 2025.

[69] G. Falchi, C. Colombo, and H. G. Lewis, "FOSTRAD: An advanced open source tool for re-entry analysis," in *7th European Conference on Space Debris*, ESA, Darmstadt, Germany, 2017.

[70] N. Awang, R. W. Rahmat, P. S. Sulaiman, and A. Jaafar, "Delaunay triangulation of missing points," *Journal of Advanced Science and Engineering*, vol. 7, no. 1, pp. 58–69, 2017.

[71] P. Maur, "Delaunay triangulation in 3d," Dept. of Computer Science and Engineering, Tech. Rep., 2002.

[72] NOAA, NASA, and U.S. Air Force, "U.S. standard atmosphere, 1976," U.S. Government Printing Office, Tech. Rep. NASA-TM-X-74335; NOAA-S/T 76-1562, 1976.

[73] MathWorks, *Comparison of 3d coordinate systems*, Accessed 2025-06-11, 2025.

[74] SBG Systems Support Center, *Reference coordinate frames*, Accessed 2025-06-11, 2025.

[75] Y. Xie, L. Liu, G. Tang, and W. Zheng, "Highly constrained entry trajectory generation," *Acta Astronautica*, vol. 88, pp. 44–60, 2013.

[76] D. Yao, Y. Hu, D. Liu, and Q. Xia, *Sliding mode formation control for multiple hypersonic glide vehicles*, Preprint, 2023.

[77] W. L. Weaver and J. T. Bowen, "Entry trajectory, entry environment, and analysis of spacecraft motion for the RAM C-3 flight experiment," NASA, Tech. Rep., 1972.

[78] H. Tran, C. Johnson, D. Rasky, F. Hui, M.-T. Hsu, and Y. Chen, "Phenolic impregnated carbon ablators (PICA) for discovery class missions," in *31st Thermophysics Conference*, 1996, p. 1911.

[79] D. Olynick, Y.-K. Chen, and M. Tauber, "Forebody TPS sizing with radiation and ablation for the Stardust sample return capsule," in *32nd Thermophysics Conference*, 1997, p. 2474.

[80] A. J. Lofthouse, "Nonequilibrium hypersonic aerothermodynamics using the direct simulation monte carlo and navier–stokes models," Ph.D. dissertation, University of Michigan, 2008.

[81] L. Scatteia, R. Borrelli, G. Cosentino, E. Beche, J.-L. Sans, and M. Balat-Pichelin, "Catalytic and radiative behaviors of ZrB_2-SiC ultrahigh temperature ceramic composites," *Journal of Spacecraft and Rockets*, vol. 43, no. 5, pp. 1004–1012, 2006.

[82] V. Liu, H. Yao, and M. White, "Toward understanding catastrophic interference in value-based reinforcement learning," in *Optimization Foundations for Reinforcement Learning Workshop at NeurIPS*, 2019.

[83] M. McCloskey and N. J. Cohen, "Catastrophic interference in connectionist networks: The sequential learning problem," in *Psychology of Learning and Motivation*, vol. 24, Elsevier, 1989, pp. 109–165.

[84] H. Yoo, V. M. Zavala, and J. H. Lee, "A dynamic penalty function approach for constraint-handling in reinforcement learning," *IFAC-PapersOnLine*, vol. 54, no. 3, pp. 487–491, 2021.

[85] R. S. Sutton, A. G. Barto, *et al.*, "Reinforcement learning," *Journal of Cognitive Neuroscience*, vol. 11, no. 1, pp. 126–134, 1999.

[86] IBM, *What is reinforcement learning?* Accessed 2025-06-17, 2023.

[87] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd. MIT Press, 2018.

[88] C. Banerjee, T. Mukherjee, and E. Pasiliao Jr, "An empirical study on generalizations of the relu activation function," in *Proceedings of the 2019 ACM southeast conference*, 2019, pp. 164–167.

[89] C. Isaac and K. Zareinia, "Effect of excessive neural network layers on overfitting," 2022.

[90] J. J. Spravka and T. R. Jorris, "Current hypersonic and space vehicle flight test instrumentation challenges," in *AIAA Flight Testing Conference*, 2015, p. 3224.

[91] SpaceRef, "Engineering review board concludes review of htv-2 second test flight," *SpaceNews*, Apr. 2012.

[92] M. D. A. Alliance, "Df-zf hypersonic glide vehicle," *Missile Threat and Proliferation – China*, 2023, Published January 13, 2023.

[93] J. P. Snyder, *Azimuthal projections — section 21*, 2025.

[94] G. van Brummelen, *Heavenly Mathematics: The Forgotten Art of Spherical Trigonometry*. Princeton University Press, 2012.

[95] Uber Technologies, *H3: A Hexagonal Hierarchical Geospatial Indexing System*, 2024.

[96] I. Brodsky, *H3: Uber's Hexagonal Hierarchical Spatial Index*, Uber Engineering Blog, Jun. 2018.

[97] A. Beresnev, A. Semenov, and E. Panidi, "Hexagonal grids applied to clustering locations in web maps," *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 43, pp. 435–440, 2022.

[98] C. Jänsch and A. Markl, "Trajectory optimization and guidance for a hermes-type reentry vehicle," in *Navigation and Control Conference*, 1991, p. 2659.

[99] D. Isakeit, A. Wilson, P. Watillon, T. Leveugle, C. Cazaux, and G. Bréard, "The atmospheric reentry demonstrator," ESA, Tech. Rep., 1998.

[100] A. Thirkettle, M. Steinkopf, and E. Joseph-Gabriel, "The mission and post-flight analysis of the atmospheric re-entry demonstrator (ARD)," *ESA Bulletin*, vol. 109, pp. 56–63, 2002.

[101] E. Wuchina, E. Opila, M. Opeka, B. Fahrenholtz, and I. Talmy, "UHTCs: Ultra-high temperature ceramic materials for extreme environment applications," *The Electrochemical Society Interface*, vol. 16, no. 4, p. 30, 2007.

[102] D. Glass, "Ceramic matrix composite (CMC) thermal protection systems (TPS) and hot structures for hypersonic vehicles," in *15th AIAA International Space Planes and Hypersonic Systems and Technologies Conference*, 2008, p. 2682.

[103] Space.com Staff, *Superfast military aircraft hit Mach 20 before ocean crash, DARPA says*, Aug. 2011.

[104] Y. Ding, X. Yue, G. Chen, and J. Si, "Review of control and guidance technology on hypersonic vehicle," *Chinese Journal of Aeronautics*, vol. 35, no. 7, pp. 1–18, 2022.

[105] K. H. Javaid and V. C. Serghides, "Airframe-propulsion integration methodology for waverider-derived hypersonic cruise aircraft design concepts," *Journal of Spacecraft and Rockets*, vol. 42, no. 4, pp. 663–671, 2005.

[106] D. McRuer, "Design and modeling issues for integrated airframe/propulsion control of hypersonic flight vehicles," in *1991 American Control Conference*, IEEE, 1991, pp. 729–734.

[107] NASA, *NASA's X-43a scramjet breaks speed record*, Dec. 2022.

[108] G. Pezzella, M. Marini, M. Cicala, A. Vitale, T. Langener, and J. Steelant, "Aerodynamic characterization of HEXAFLY scramjet propelled hypersonic vehicle," in *32nd AIAA Applied Aerodynamics Conference*, 2014, p. 2844.

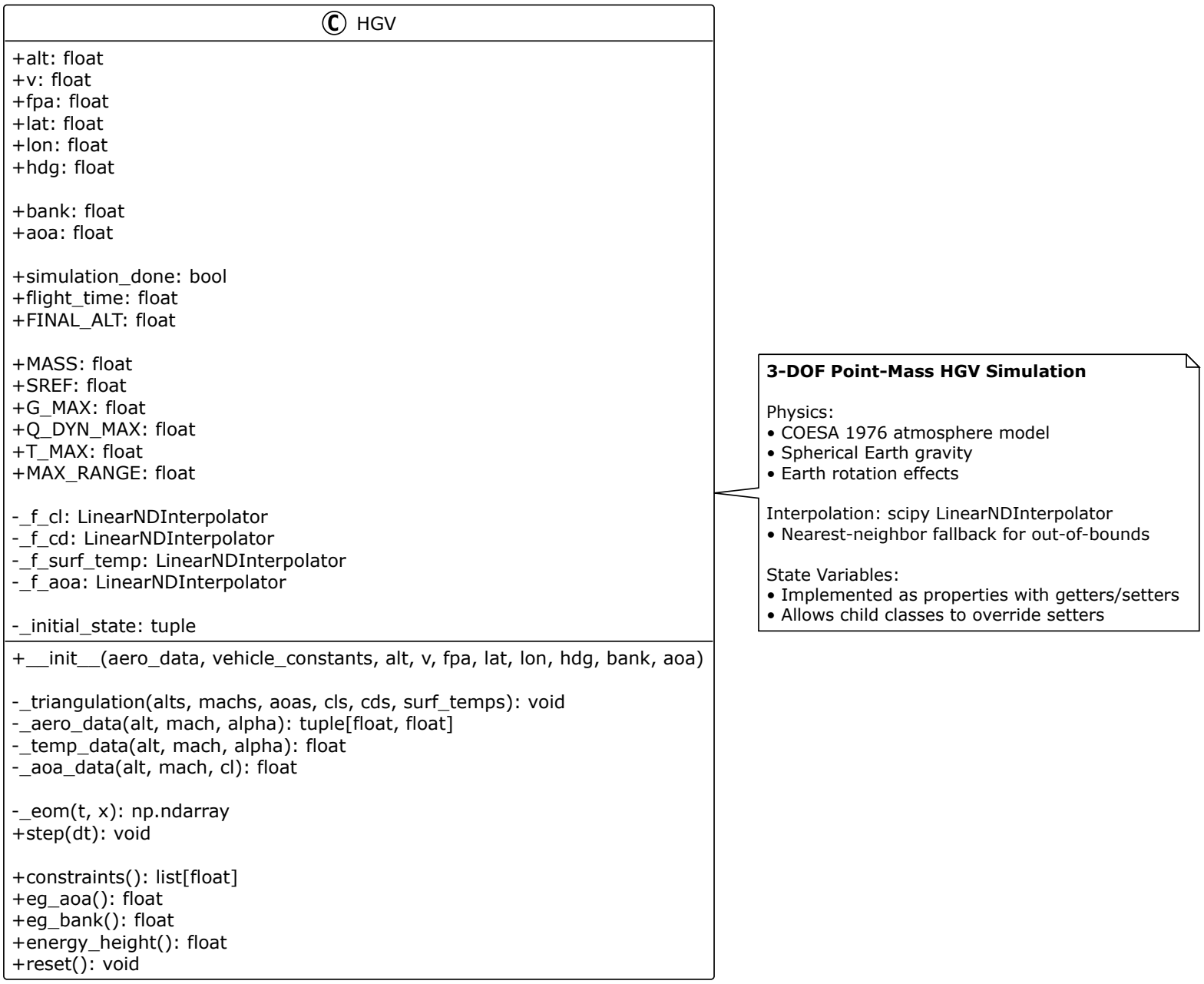
[109] M. Mirmirani, C. Wu, A. Clark, S. B. Choi, and R. Colgren, "Modeling for control of a generic airbreathing hypersonic vehicle," in *AIAA Guidance, Navigation, and Control Conference and Exhibit*, 2005, p. 6256.

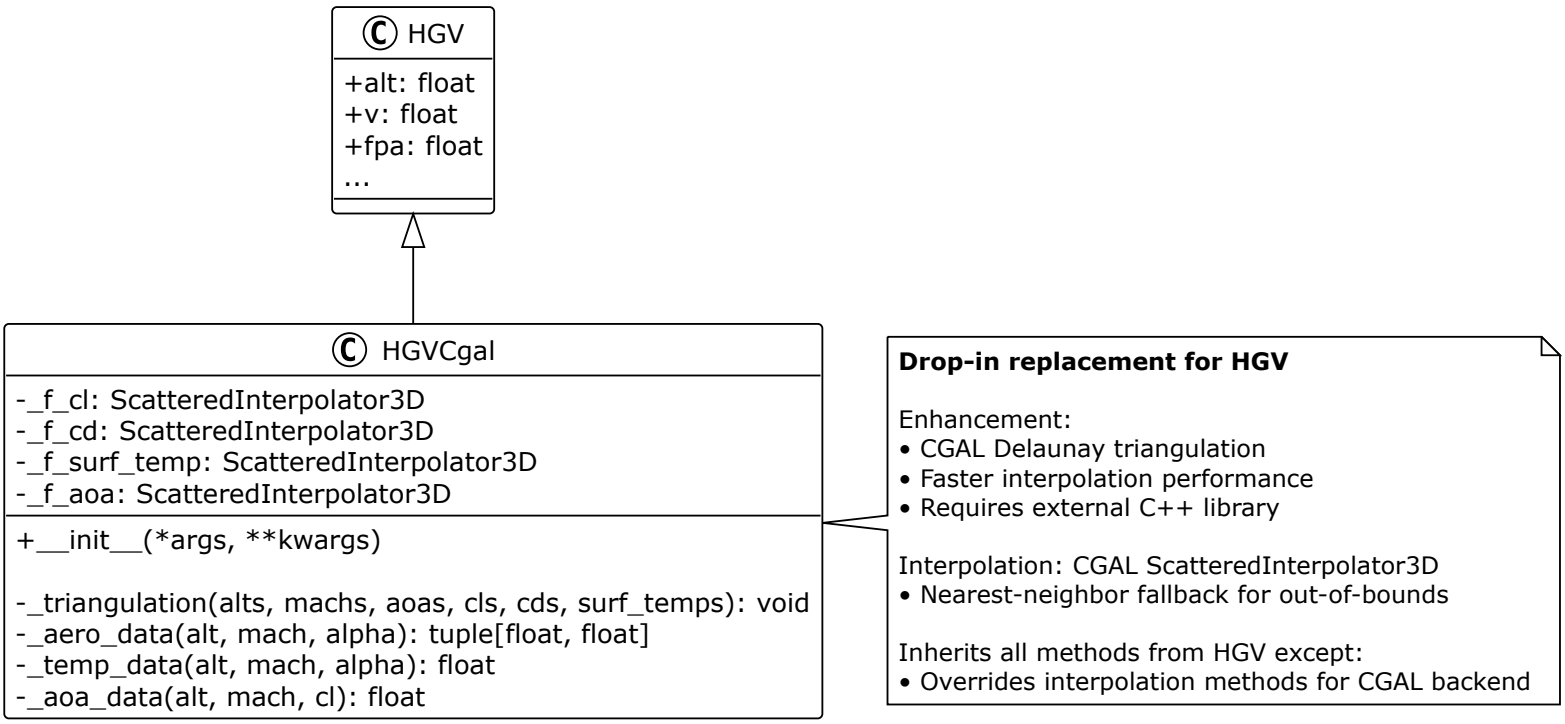
[110] S. O. Macheret, M. N. Shneider, and R. B. Miles, "Scramjet inlet control by off-body energy addition: A virtual cowl," *AIAA Journal*, vol. 42, no. 11, pp. 2294–2302, 2004.

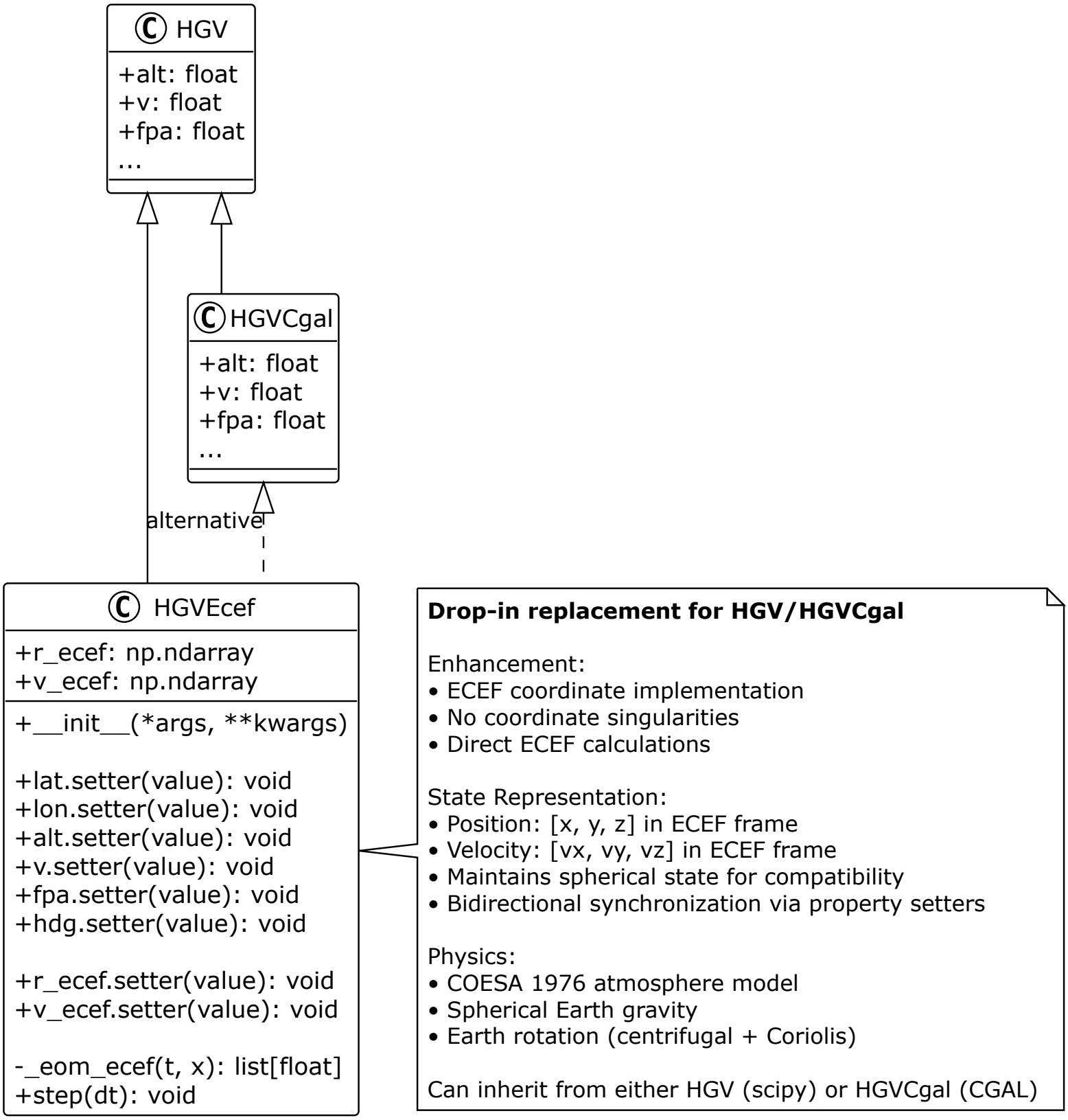
[111] D. Chai, Y.-W. Fang, Y. Wu, and S. Xu, “Boost-skipping trajectory optimization for air-breathing hypersonic missile,” *Aerospace Science and Technology*, vol. 46, pp. 506–513, 2015.

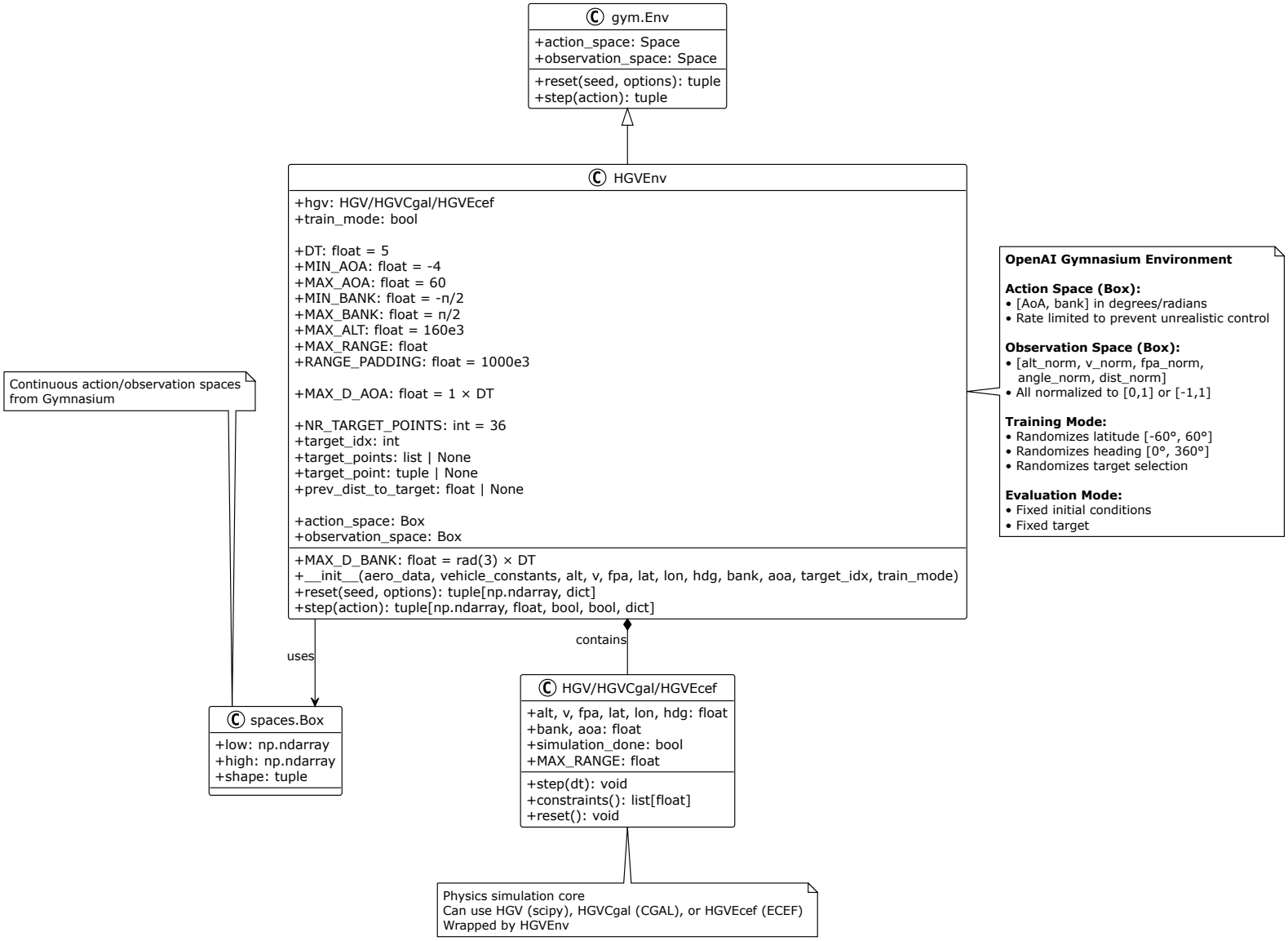
A

Program Architecture Diagrams

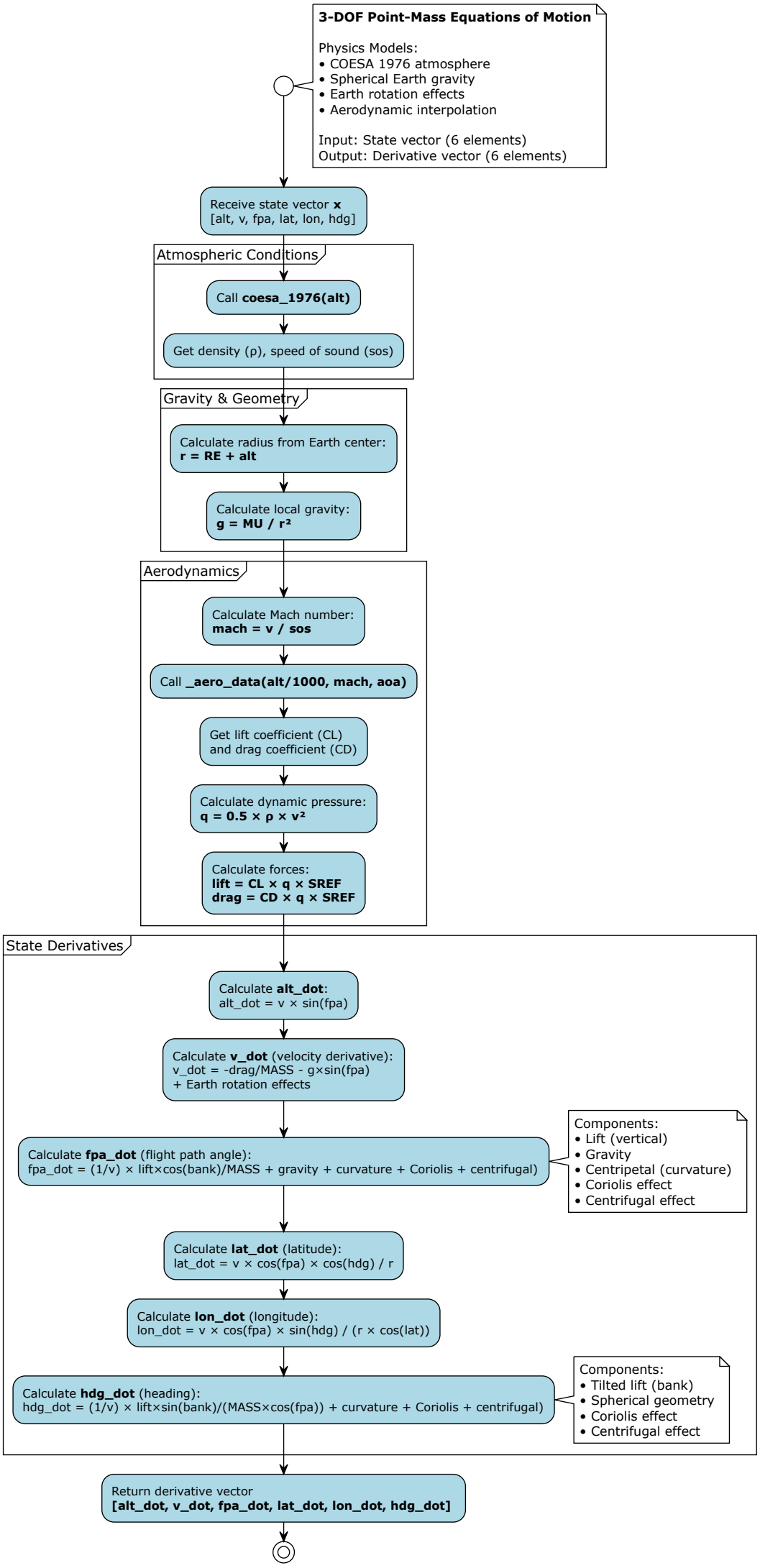

HGV
+alt: float
+v: float
+fpa: float
+lat: float
+lon: float
+hdg: float
+bank: float
+aoa: float
+simulation_done: bool
+flight_time: float
+FINAL_ALT: float
+MASS: float
+SREF: float
+G_MAX: float
+Q_DYN_MAX: float
+T_MAX: float
+MAX_RANGE: float
-_f_cl: LinearNDInterpolator
-_f_cd: LinearNDInterpolator
-_f_surf_temp: LinearNDInterpolator
-_f_aoa: LinearNDInterpolator
-_initial_state: tuple
+__init__(aero_data, vehicle_constants, alt, v, fpa, lat, lon, hdg, bank, aoa)
-_triangulation(alts, machs, aoas, cls, cds, surf_temps): void
-_aero_data(alt, mach, alpha): tuple[float, float]
-_temp_data(alt, mach, alpha): float
-_aoa_data(alt, mach, cl): float
-_eom(t, x): np.ndarray
+step(dt): void
+constraints(): list[float]
+eg_aoa(): float
+eg_bank(): float
+energy_height(): float
+reset(): void
3-DOF Point-Mass HGV Simulation
Physics:
• COESA 1976 atmosphere model
• Spherical Earth gravity
• Earth rotation effects
Interpolation: scipy LinearNDInterpolator
• Nearest-neighbor fallback for out-of-bounds
State Variables:
• Implemented as properties with getters/setters
• Allows child classes to override setters


HGV
+alt: float
+v: float
+fpa: float
...
HGVCgal
-_f_cl: ScatteredInterpolator3D
-_f_cd: ScatteredInterpolator3D
-_f_surf_temp: ScatteredInterpolator3D
-_f_aoa: ScatteredInterpolator3D
+__init__(*args, **kwargs)
-_triangulation(alts, machs, aoas, cls, cds, surf_temps): void
-_aero_data(alt, mach, alpha): tuple[float, float]
-_temp_data(alt, mach, alpha): float
-_aoa_data(alt, mach, cl): float
Drop-in replacement for HGV
Enhancement:
• CGAL Delaunay triangulation
• Faster interpolation performance
• Requires external C++ library
Interpolation: CGAL ScatteredInterpolator3D
• Nearest-neighbor fallback for out-of-bounds
Inherits all methods from HGV except:
• Overrides interpolation methods for CGAL backend

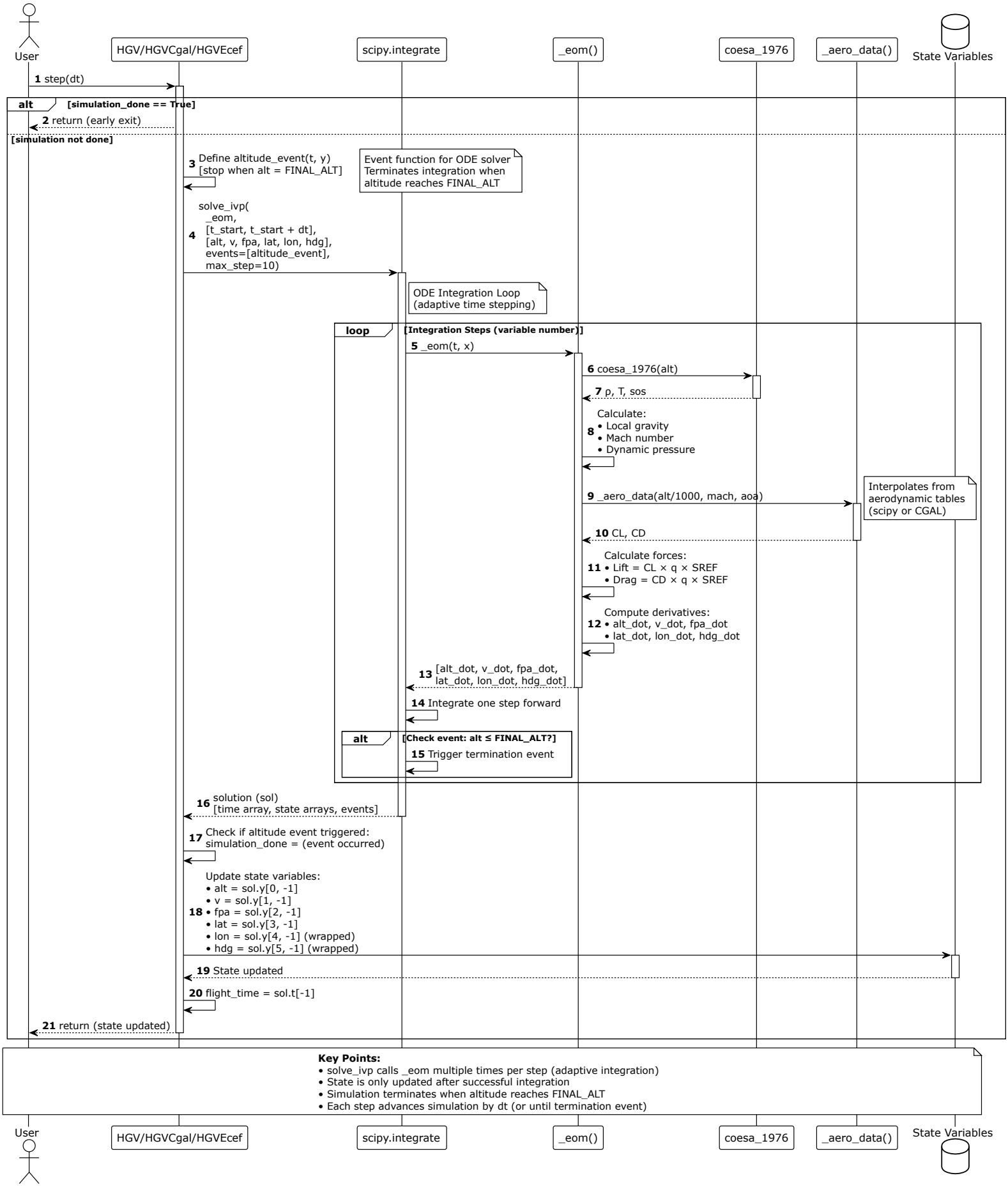

HGV
+alt: float
+v: float
+fpa: float
...
HGVCgal
+alt: float
+v: float
+fpa: float
...
alternative
HGVEcef
+r_ecef: np.ndarray
+v_ecef: np.ndarray
+__init__(*args, **kwargs)
+lat.setter(value): void
+lon.setter(value): void
+alt.setter(value): void
+v.setter(value): void
+fpa.setter(value): void
+hdg.setter(value): void
+r_ecef.setter(value): void
+v_ecef.setter(value): void
-_eom_ecef(t, x): list[float]
+step(dt): void
Drop-in replacement for HGV/HGVCgal
Enhancement:
• ECEF coordinate implementation
• No coordinate singularities
• Direct ECEF calculations
State Representation:
• Position: [x, y, z] in ECEF frame
• Velocity: [vx, vy, vz] in ECEF frame
• Maintains spherical state for compatibility
• Bidirectional synchronization via property setters
Physics:
• COESA 1976 atmosphere model
• Spherical Earth gravity
• Earth rotation (centrifugal + Coriolis)
Can inherit from either HGV (scipy) or HGVCgal (CGAL)


gym.Env
+action_space: Space
+observation_space: Space
+reset(seed, options): tuple
+step(action): tuple
HGVEnv
+hgv: HGV/HGVCgal/HGVEcef
+train_mode: bool
+DT: float = 5
+MIN_AOA: float = -4
+MAX_AOA: float = 60
+MIN_BANK: float = -π/2
+MAX_BANK: float = π/2
+MAX_ALT: float = 160e3
+MAX_RANGE: float
+RANGE_PADDING: float = 1000e3
+MAX_D_AOA: float = 1 × DT
+NR_TARGET_POINTS: int = 36
+target_idx: int
+target_points: list | None
+target_point: tuple | None
+prev_dist_to_target: float | None
+action_space: Box
+observation_space: Box
+MAX_D_BANK: float = rad(3) × DT
+__init__(aero_data, vehicle_constants, alt, v, fpa, lat, lon, hdg, bank, aoa, target_idx, train_mode)
+reset(seed, options): tuple[np.ndarray, dict]
+step(action): tuple[np.ndarray, float, bool, bool, dict]
OpenAI Gymnasium Environment
Action Space (Box):
• [AoA, bank] in degrees/radians
• Rate limited to prevent unrealistic control
Observation Space (Box):
• [alt_norm, v_norm, fpa_norm, angle_norm, dist_norm]
• All normalized to [0,1] or [-1,1]
Training Mode:
• Randomizes latitude [-60°, 60°]
• Randomizes heading [0°, 360°]
• Randomizes target selection
Evaluation Mode:
• Fixed initial conditions
• Fixed target
Continuous action/observation spaces from Gymnasium
uses
contains
spaces.Box
+low: np.ndarray
+high: np.ndarray
+shape: tuple
HGV/HGVCgal/HGVEcef
+alt, v, fpa, lat, lon, hdg: float
+bank, aoa: float
+simulation_done: bool
+MAX_RANGE: float
+step(dt): void
+constraints(): list[float]
+reset(): void
Physics simulation core
Can use HGV (scipy), HGVCgal (CGAL), or HGVEcef (ECEF)
Wrapped by HGVEnv

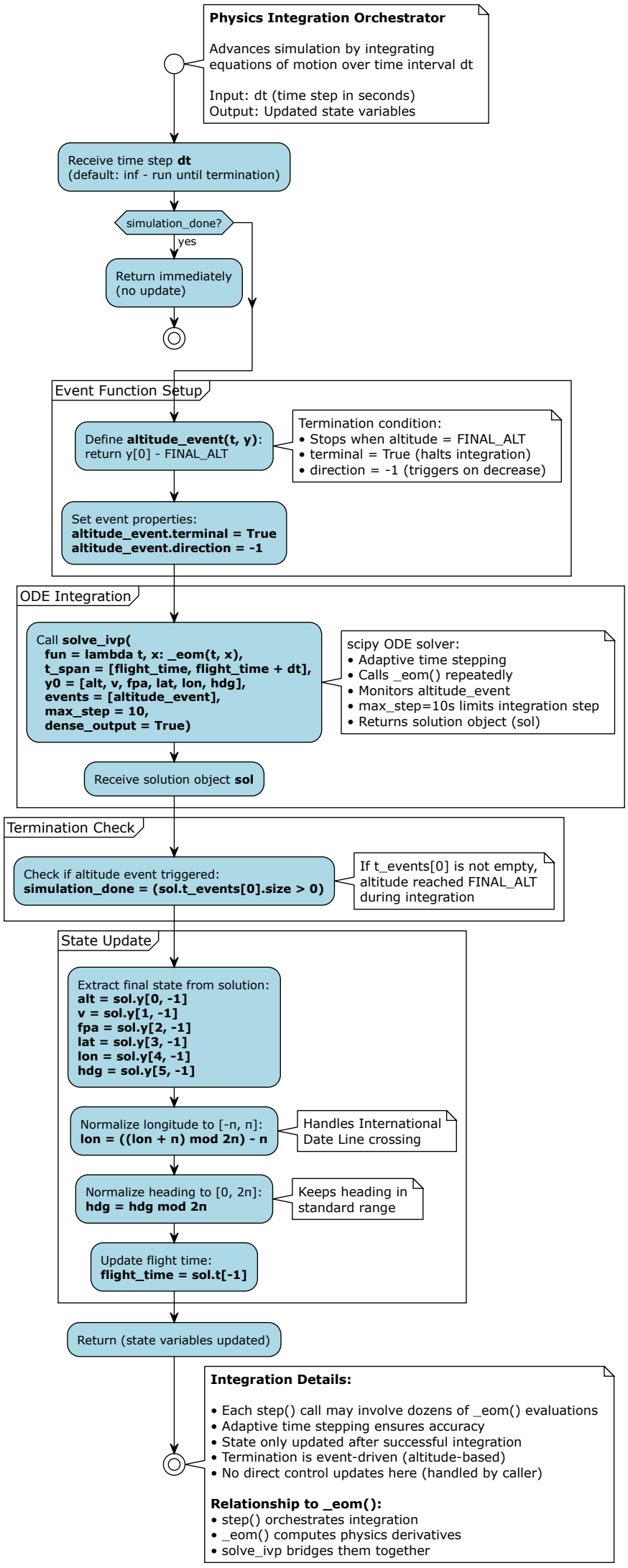
HGV.eom Activity Diagram



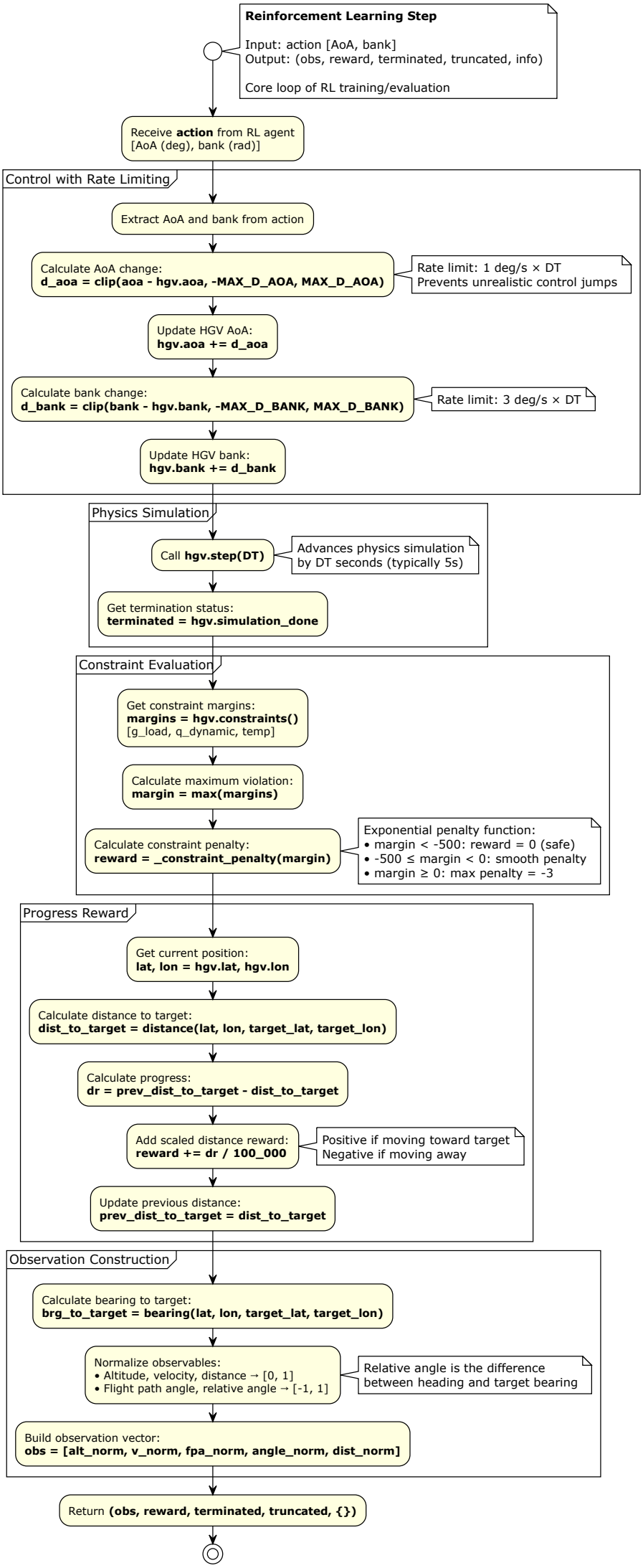
HGV.step Sequence



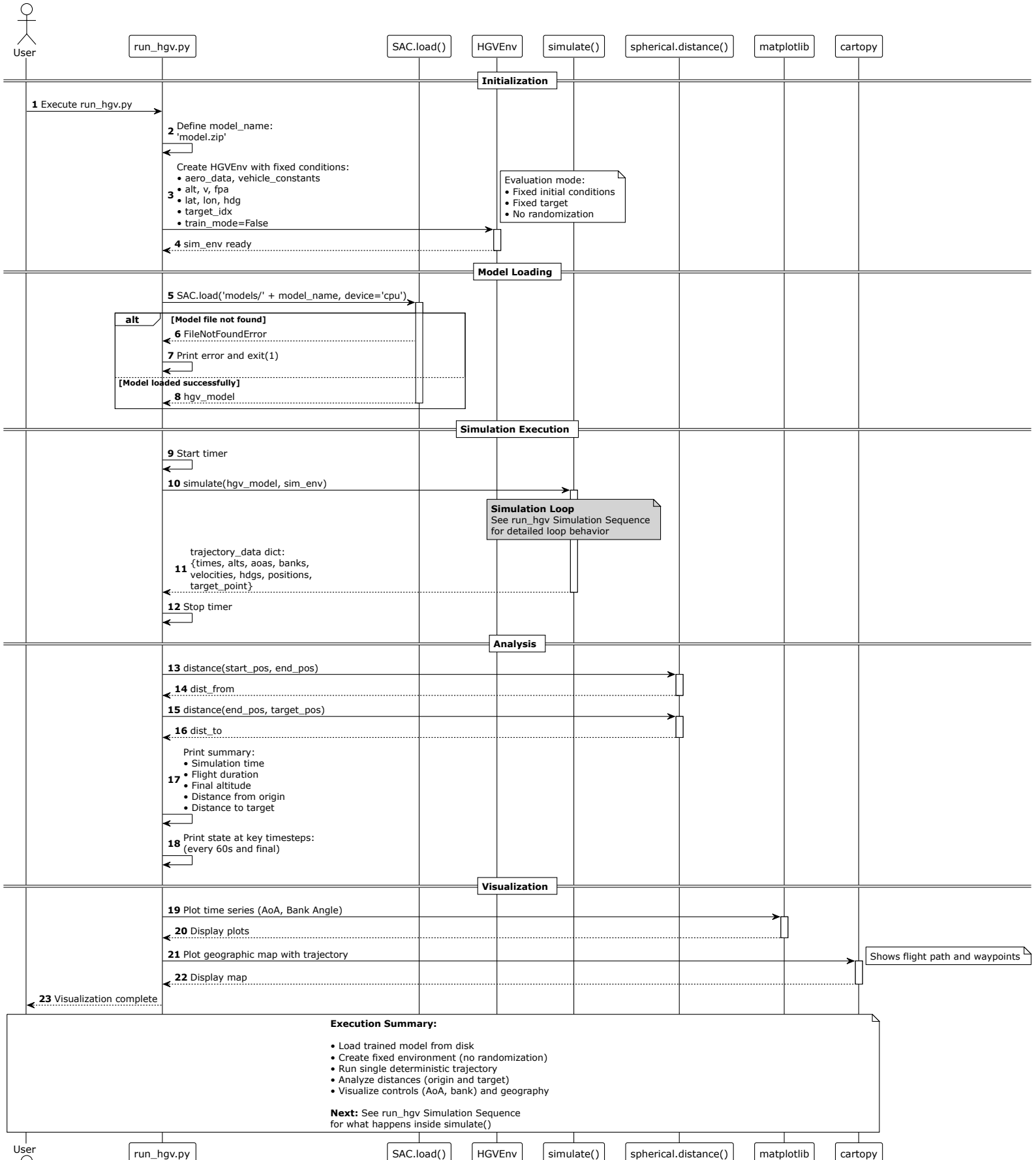
HGV.step Activity Diagram



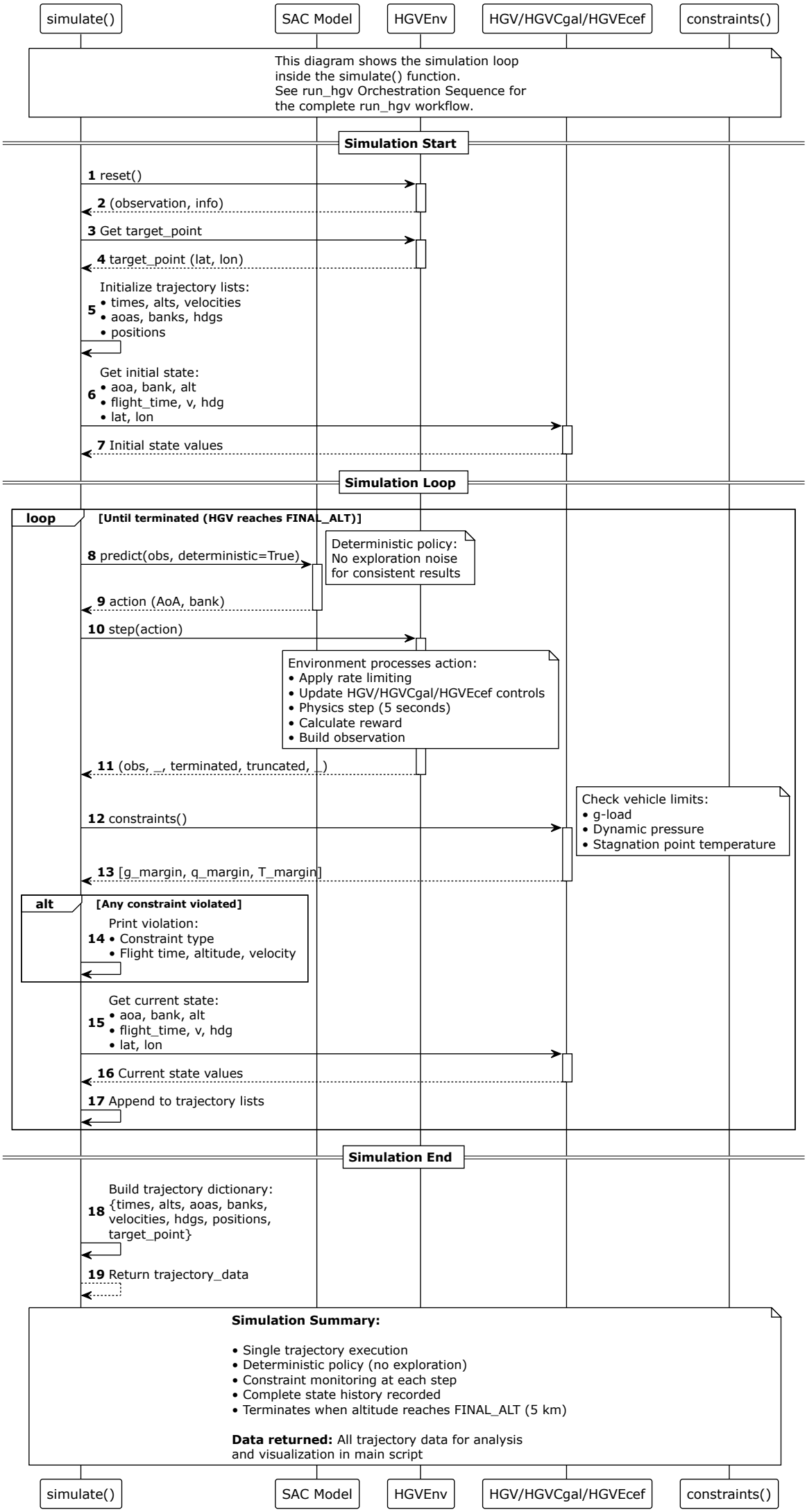
HGVEnv.step Activity Diagram



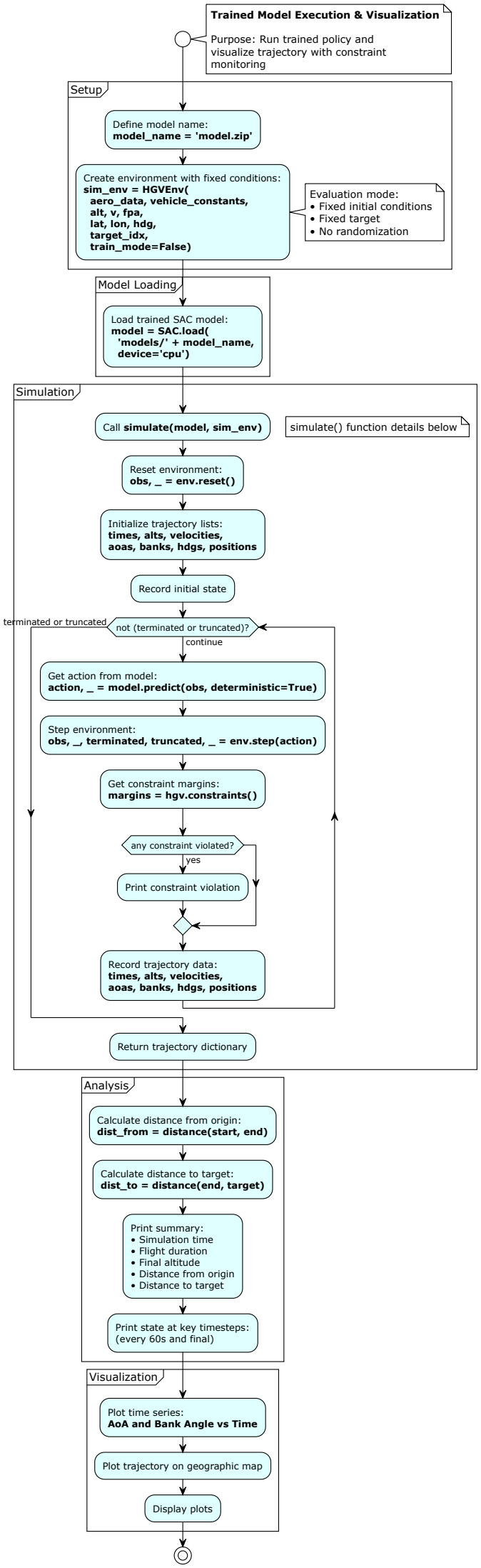
run_hgv Orchestration Sequence



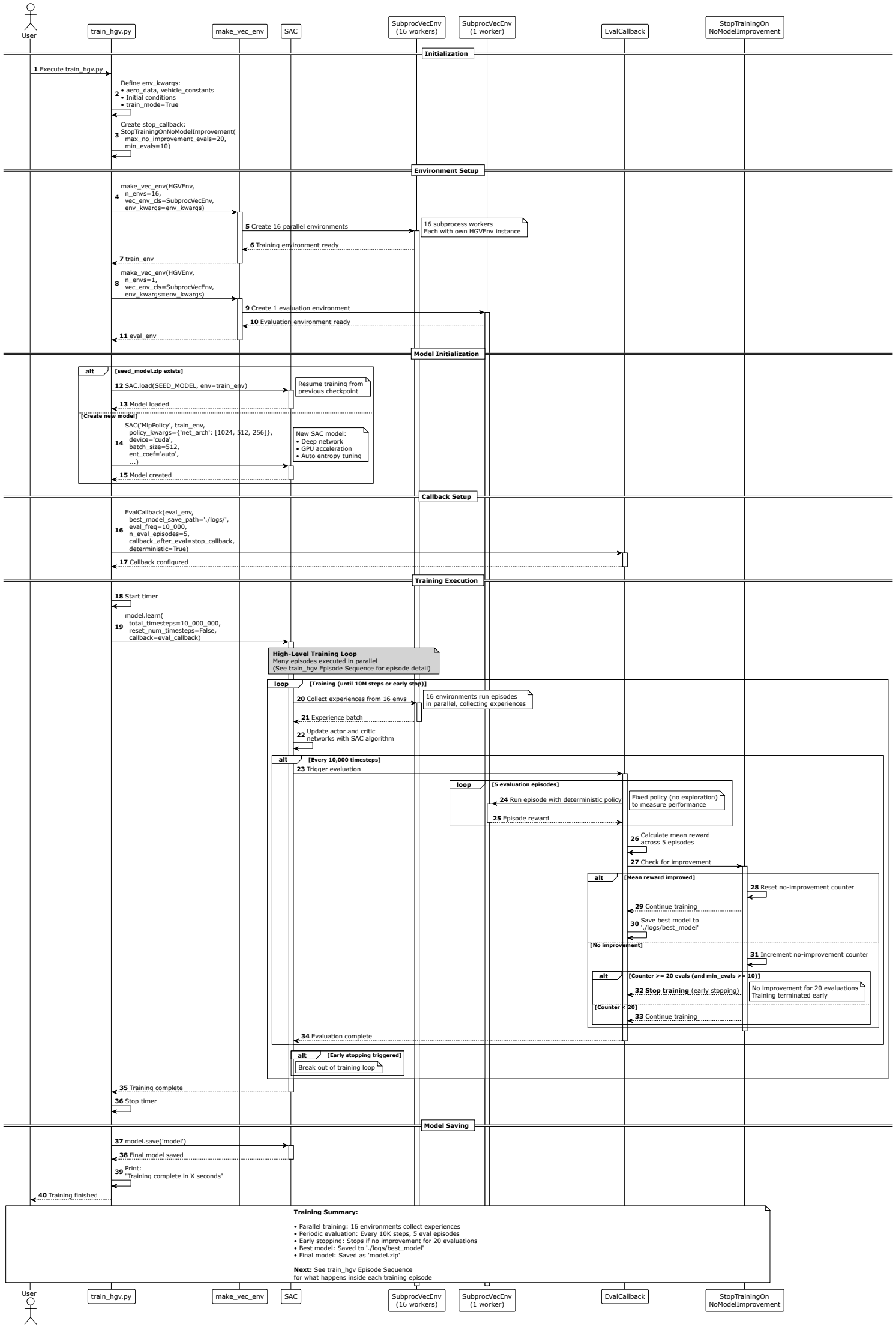
run_hgv Simulation Sequence



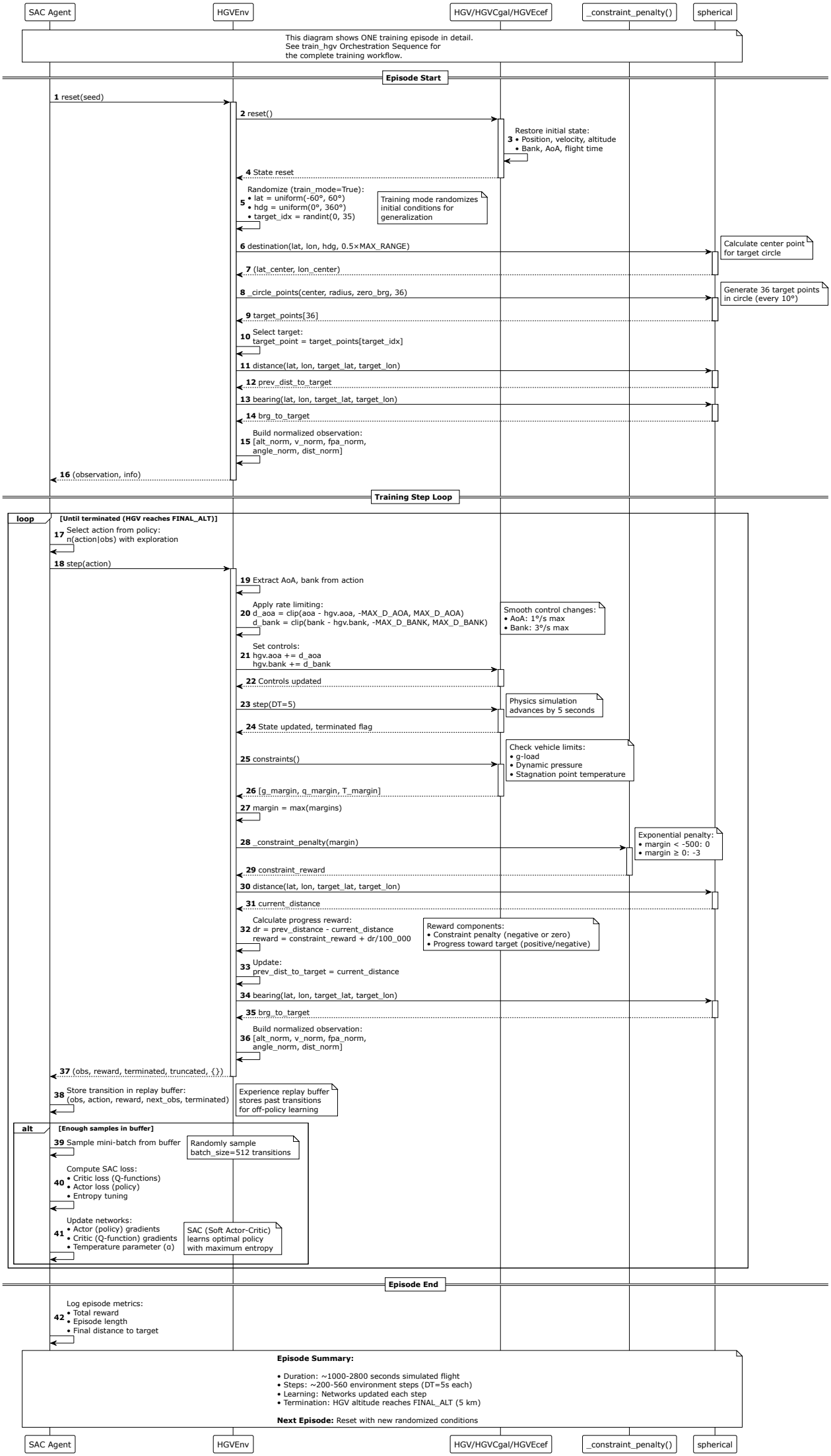
run_hgv Activity



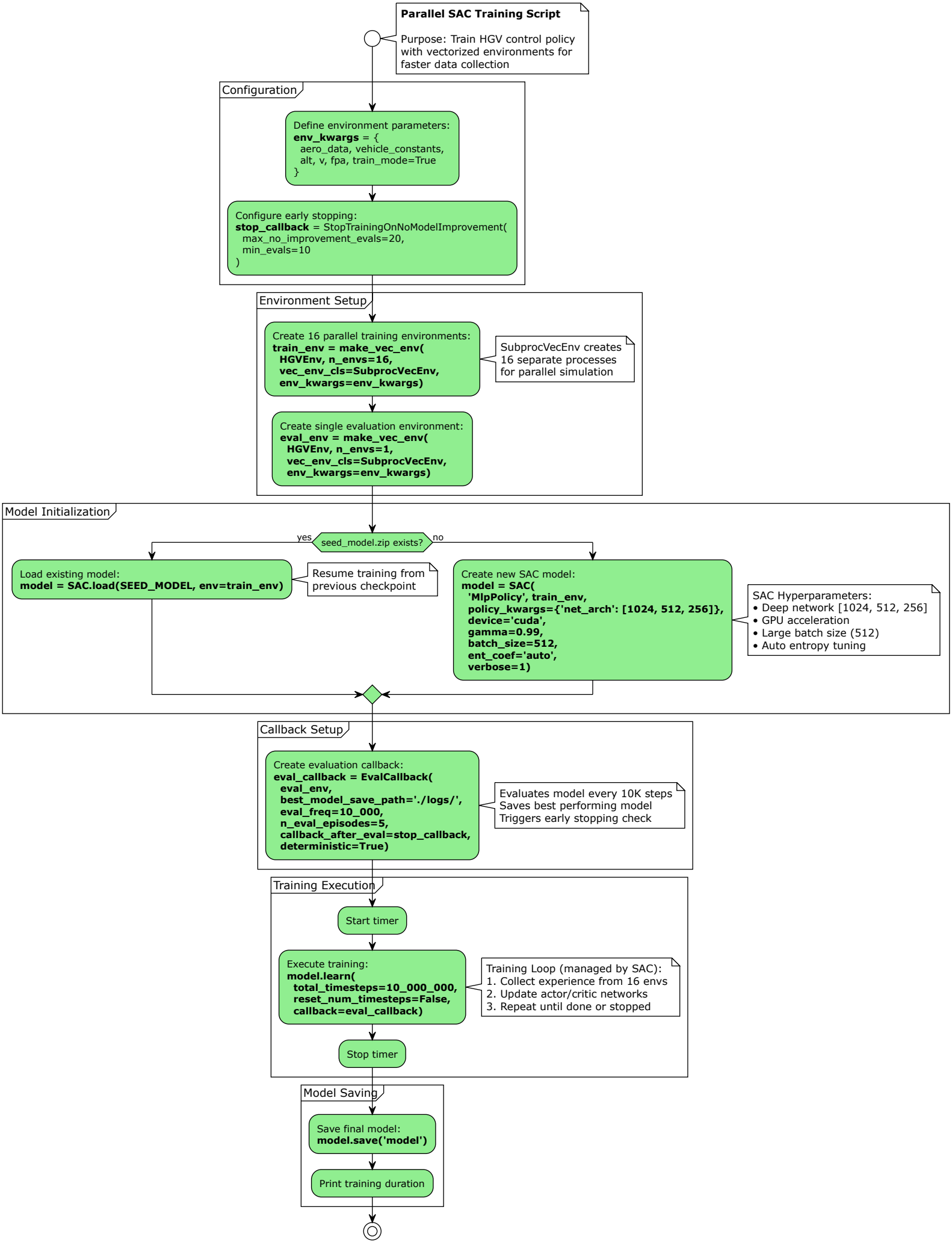
train_hgv Orchestration Sequence



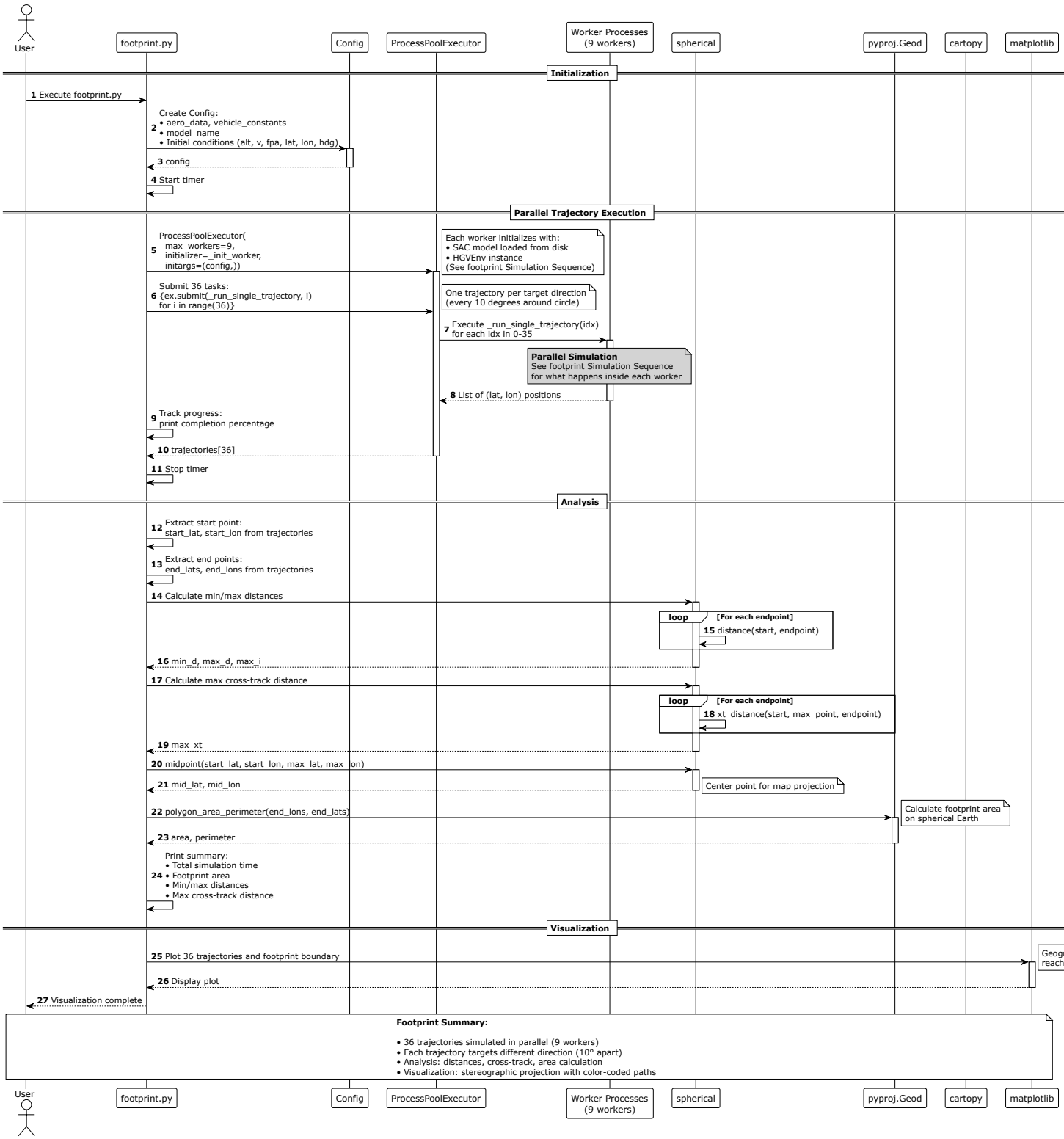
train_hgv Episode Sequence



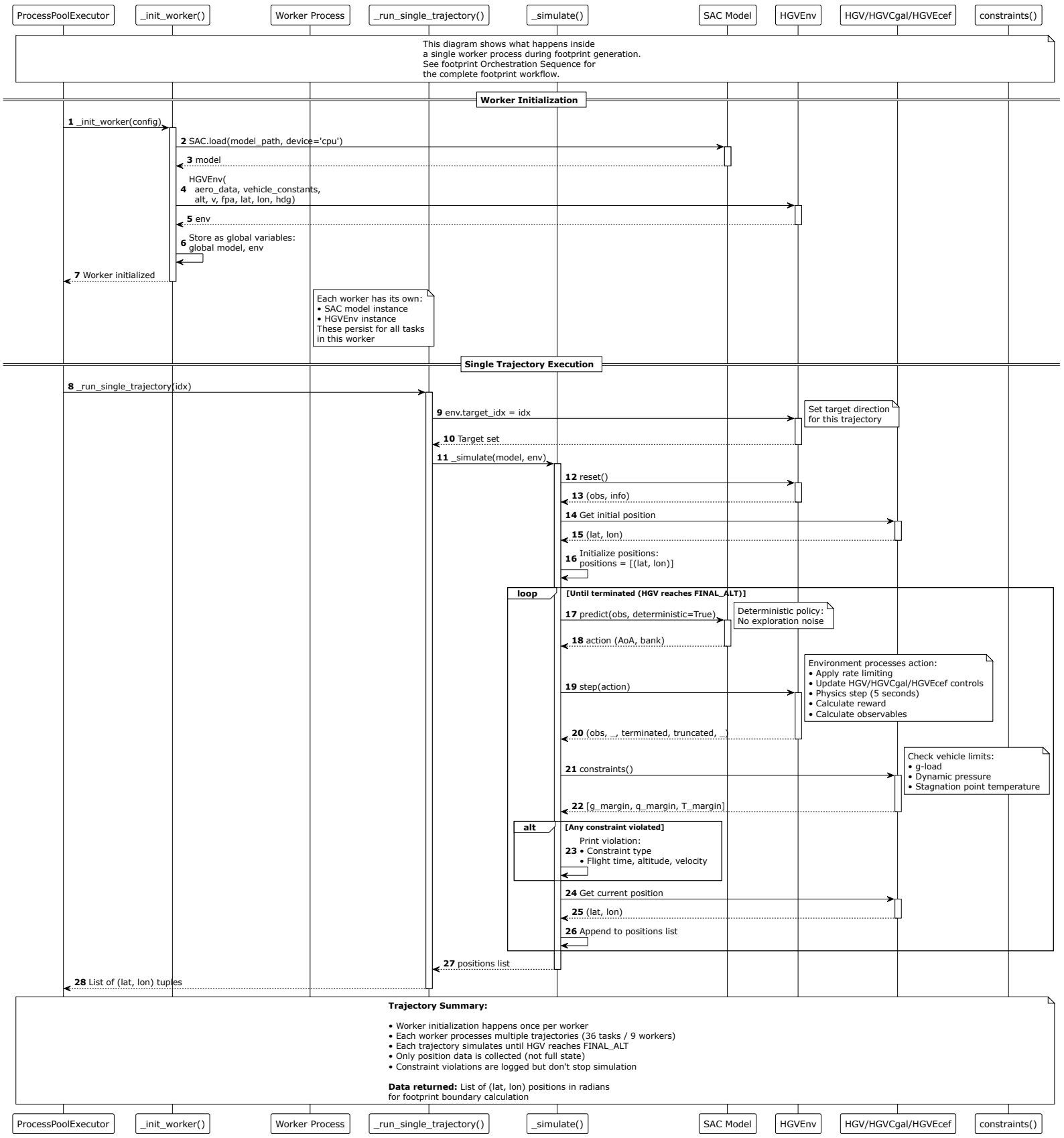
train_hgv Activity Diagram



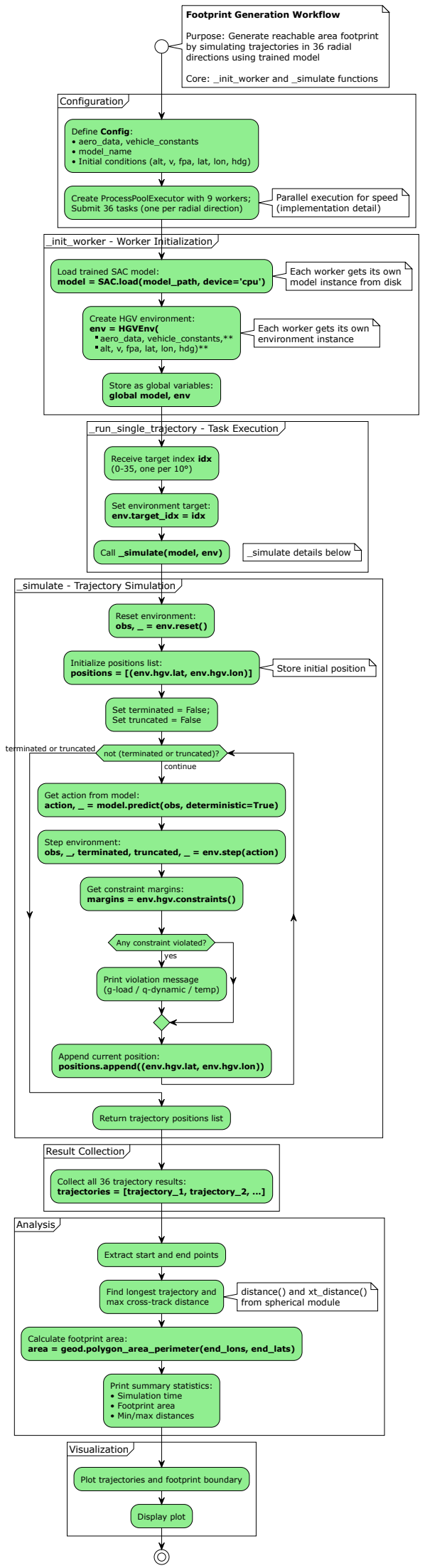
footprint Orchestration Sequence



footprint Simulation Sequence



footprint Activity



B

Thermal Protection System

B.1. Thermal Protection System

The design of a Thermal Protection System (TPS) is essential to ensure vehicle survivability. For the thermal protection system an approximation can be made that the TPS wall temperature can be computed from the convective and radiative heat flux equilibrium. Here, during atmospheric entry, the heat absorbed from aerodynamic heating is balanced by the heat radiated away from its surface resulting in an equilibrium temperature. This can be calculated with the following formula:

$$T_w = \left(\frac{\dot{q}_{conv}}{\epsilon \sigma} \right)^{1/4}$$

where T_w is the wall temperature at radiative equilibrium, $\dot{q}_{conv}$ represents the convective heat flux, ϵ is the emissivity of the surface and σ is the Stefan-Boltzmann constant. This temperature equilibrium is also depicted in Figure B.1.

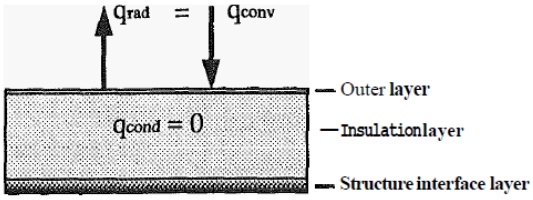


Figure B.1: Steady-state TPS heating model [98]

A material for the outer surface of the hypersonic glide vehicle has to be chosen such that it can withstand the high temperatures that occur when hypersonic velocities are reached. ESA has designed an Atmospheric Reentry Demonstrator (ARD) which is a capsule that follows a suborbital path and reaches velocities in the atmosphere of 27000 km/h [99]. The ARD heatshield will be exposed to temperatures up to 2000°C with a heat flux of around $90 - 125$ kW/m^2.

The thermal protection system that has been used on ARD consists of several different types. First, aleastrasil is used on the main heat shield, which is a compound containing randomly oriented silica fibers that are impregnated with phenolic resin. Furthermore, Norcoat is used for the cone section, which is composed of cork powder and phenolic resin. Finally, samples of Flexible External Insulation (FEI) and Ceramic Matrix Composite (CMC) have been used [100]. However, it is critical to note that such ablative systems are generally unsuitable for HGVs. Due to the extended flight durations and the need for aerodynamic stability, HGVs require shape-preserving, non-ablative materials. These typically

include advanced carbon-carbon composites, Ultra-High Temperature Ceramics and reinforced CMCs, which offer high thermal resistance without altering the vehicle's geometry during flight.

Ultra-High Temperature Ceramics (UHTC) can be used to protect the hypersonic glide vehicle against these high temperatures. These are a type of ceramic that can withstand high temperatures without degradation, often above $2000°$C [101]. These materials are also highly resistant to thermal shock, which means that extreme changes in temperature can be withstood in a short amount of time. UHTCs, particularly Hafnium and Zirconium-based diborides, are being developed to withstand the extreme forces and temperatures that are encountered at the leading edges of vehicles during atmospheric reentry and sustained hypersonic flight. These surfaces can face temperatures that exceed $2500°$C. One of the key challenges facing HGV development lies in designing materials that are able to withstand prolonged thermal loads at extreme speeds. Unlike traditional blunt reentry capsules, which benefit from a bow shock that forms a dense, cooler plasma layer insulating the surface, HGVs are designed with sharper, more aerodynamic profiles to enable sustained lift and maneuverability. These slender geometries reduce the protective standoff distance of the shock layer, resulting in significantly higher localized surface heating. As a result of this, HGVs require more advanced, non-ablative thermal protection systems capable of maintaining their aerodynamic shape under high thermal and mechanical stress.

DARPA's HTV-2 uses an advanced carbon-carbon aeroshell as its thermal protection system [102]. The second test flight of HTV-2 planned a 30-minute Mach 20 glide flight. The glider surface reached $1930°$C, which caused part of the skin to peel away from the aerostructure [103]. The flight took a total of 9 minutes before contact was lost with the vehicle.

It should be noted that the relation between the radius of curvature and the temperature at a leading edge is inversely proportional. Thus, vehicles with sharp leading edges experience higher temperatures, but also benefit from a higher lift-to-drag ratio.

The Space Shuttle is a prime example of a vehicle that has to deal with high temperatures. For the leading edges and the nose cap, where temperatures are experienced that are greater than $1260°C$, reinforced carbon/carbon is used. The TPS on the X-33 was similar to that on the Space Shuttle, where carbon-carbon leading edges were used.

The required material properties of hypersonic air-breathing vehicles are that they can withstand high temperatures, high strength at those temperatures, light weight, and high toughness. In Figure B.2, an overview is given of the different materials and their corresponding strength at different temperatures. From this figure, the type of material that has to be used can be deduced. Ceramic Matrix Composites (CMC), the C/SiC material, Advanced Carbon/Carbon (ACC) and SiC/SiC provide high strength at elevated temperatures [102].

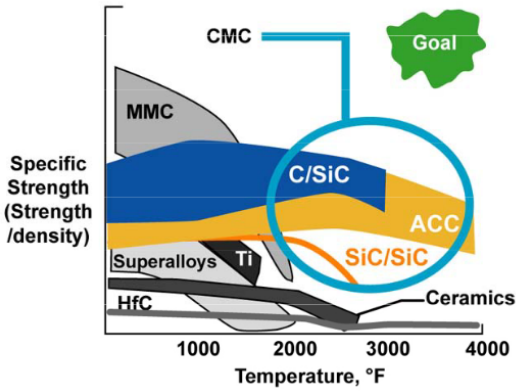


Figure B.2: Specific strength versus temperature for different materials.

C

Scramjet Integration

C.1. Scramjet Integration

This section explores the modeling and integration of scramjet propulsion into HGV trajectories. The chapter discusses aerodynamic-propulsion coupling and design criteria for ensuring effective propulsion within the limits of the atmosphere.

In the current stage, the development of unpowered hypersonic vehicles is relatively advanced. However, air-breathing powered aircraft include the newest technologies in the aerospace sector. These include hypersonic aerodynamic technology, fuselage engine integrated design technology, thermal protection on hypersonic speeds, hypersonic scramjet technology, and hypersonic flight control technology. Compared to more traditional aircraft, the integrated design of the engine into the fuselage of an Air-breathing Hypersonic Flight Vehicle (AHFV) results in a strong coupling between the aerodynamic system and propulsion system. Additionally, due to the strong non-linear behavior, static instability, and uncertain aerodynamic parameters, significant challenges are faced in the control system design [104].

A viable solution for hypersonic glide vehicles is to add an engine to the design. This engine can be used during a skip maneuver to increase its range, and this propulsion system can be integrated within the lower part of the vehicle airframe. However, the free-stream capture area must be large for the engine to generate sufficient thrust at high altitudes, and the nozzle exit area should also be large enough so that it can provide sufficient expansion [105]. This has been demonstrated by NASA's X-43A experimental research vehicle. Here, the forebody of the aircraft has been used as the compression surface for efficient operation at hypersonic speeds. The afterbody functioned as part of the propulsion system, which gives importance to the interdependence of aerodynamic and propulsion components in hypersonic flight.

Vehicles flying with hypersonic speeds will experience different flows compared to vehicles flying in the subsonic or supersonic domain, due to the high stagnation temperatures and enthalpies that are experienced. The efficiency of the aerodynamics and propulsion systems is critical at the cruise Mach numbers, which leads to a slender aerodynamic shape that has the potential of also incorporating the propulsion system.

The interactions of the propulsion system, the airframe, and the controls may define regions where the vehicle cannot fly [38]. In the case of the boosted HGV, this should be taken into account because the propulsion system will only work in certain areas of the atmosphere. As an example, Figure C.1 gives the limitations on the glide corridor of the HGV in the event that it has an air breathing propulsion system on board.

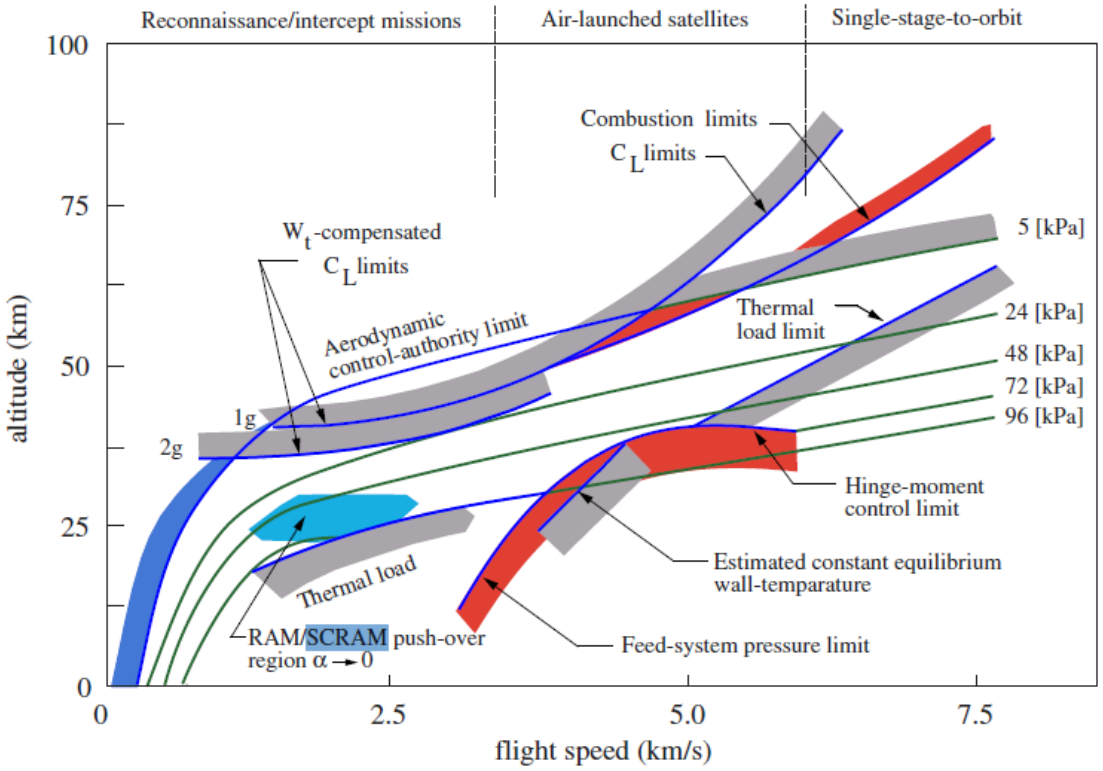


Figure C.1: The attainable envelope of an air-breathing vehicle in space [106], adapted by Mooij et al. [38].

To date, the maximum speed attained by a scramjet engine is Mach 9.8. [107]. The European Space Agency (ESA) has conducted a study on a scramjet propelled hypersonic vehicle and has investigated flight control activities and their impacts on vehicle layout and aerodynamic performance [108]. This study was designed to provide a flyable, trim-able, and statically and dynamically stable vehicle configuration. However, for this thesis, a study focused more on the operational limits of the scramjet in a hypersonic flight is required. The study by Mirmirani et al. [109] was more focused on this goal. In the 1970s, NASA focused on a rectangular airframe-integrated engine configuration. Here, the inlet sidewalls are used in order to produce extra horizontal compression in addition to the vertical forebody compression. This can be seen in Figure C.2, where the flow characteristics of an AHFV are shown.

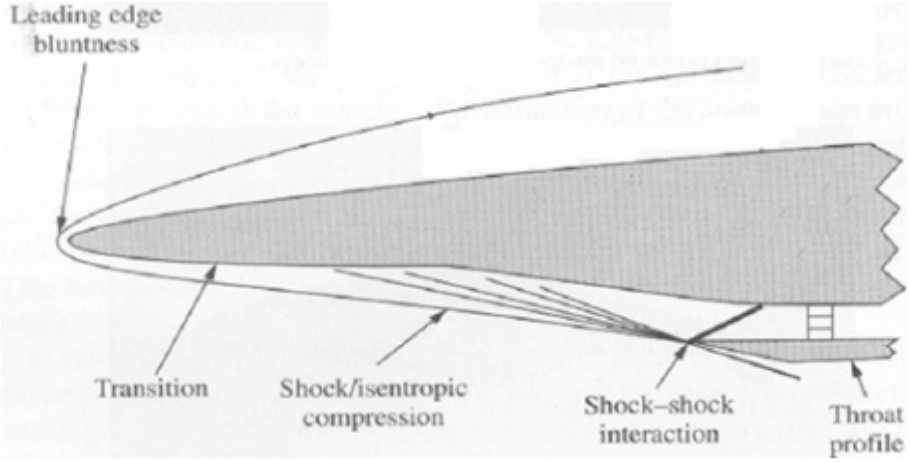


Figure C.2: The flow features on an AHFV [109].

Several tests of AHFV have already been performed. The X-43A is a scaled AHFV from a 200-ft operational concept. This vehicle has successfully demonstrated powered flight tests at Mach 7 and Mach 10, where hypersonic aerodynamic effects are especially challenging.

Optimal operating conditions will be based on the shock wave that is formed on the leading edge to be captured by the inlet lip. This can also be seen in Figure C.2. In addition, vibrations of the aircraft structure and aero-elastic effects result in mass-flow spillage. This is because the bow shock changes as the structure deforms, resulting in the engine operating under off-design conditions. Therefore, these vehicles require a variable geometry inlet to operate at optimal aerodynamic and propulsion conditions at any time [109].

At hypersonic speeds, the air no longer behaves as an ideal gas, leading to significant changes in aerodynamics. The temperature behind the shock waves, the forces in the aircraft and the pitching moment all increase as the speed increases. The extreme heat generated at these speeds causes the air layer around the aircraft to thicken, altering the effective shape of the wings and control surfaces.

Scramjet engines work by burning fuel in supersonic airflow that is compressed by the aircraft's speed. Since fuel and air mix and burn quickly under supersonic conditions, the combustion chamber must be longer to allow enough time for proper combustion. To generate enough thrust for hypersonic speeds, the engine must capture as much incoming air as possible. This is achieved by designing the engine as part of the aircraft's body, with the air intake merging smoothly with the underside of the vehicle. The sides of the engine inlet also help compress the air from the sides, adding to the compression that already happens from the front of the aircraft [109].

A 2D configuration of the hypersonic vehicle is based on inviscid compressible flow theory of a perfect gas. It must be noted that this article makes simplifications compared to the real world. The upper surface is modeled as a flat surface and is kept at an angle of attack of zero. The lower side contains a frontal wedged surface, a trailing wedged surface, and a scramjet engine that has a constant cross-sectional area. This has been done in such a way that the frontal wedged surface acts as a diffuser for the flow that enters the scramjet and the trailing surface as a propulsive surface. This design is shown in Figure C.3 and is considered a generic configuration of hypersonic vehicles [109].

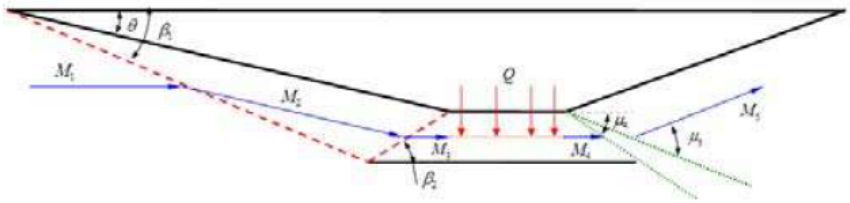
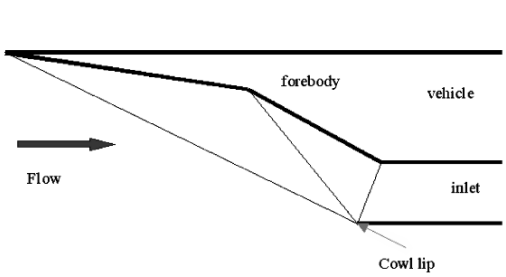


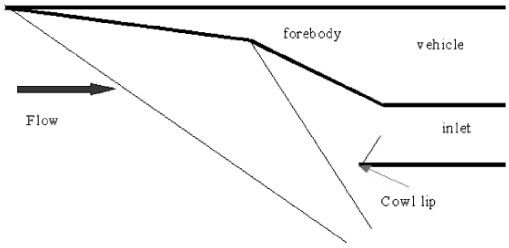
Figure C.3: The shock and expansion waves of a hypersonic vehicle [109].

The optimum geometry of a scramjet is the shock-on-lip condition. This means that the compression ramp shocks converge on the cowl lip and the reflected shock hits the shoulder of the inlet. This shock-on-lip condition depends on the Mach number, since the shock is dependent on the Mach number as well. Therefore, the shock-on-lip condition cannot be met at Mach numbers that are higher or lower than the design Mach number. The shock-on-lip condition is shown in Figure C.4a [110]. In the case that the HGV flies at Mach numbers that are lower than the design value, there is flow spillage which creates drag and the air mass capture decreases. This is shown in Figure C.4b [110].

A possibility of reducing the performance penalty on the non-design Mach numbers is to use a variable geometry according to Macheret et al. [110].



(a) Shock-on-lip condition for design Mach number.



(b) Flow spillage due to Mach number lower than the design value.

Figure C.4: Shock-on-lip condition for different Mach numbers.

Study performed on boost-skipping trajectory

A boost-skipping trajectory for an air-breathing hypersonic vehicle with the scramjet ignited in break-cycle mode has been proposed by Chai et al. [111]. For this model, the Gauss Pseudospectral Method (GPM) has been used, which was described in Section 4.3.3. The difference from normal reentry is that the air-breathing hypersonic vehicle is still in powered flight after booster separation. It should be noted that the angle of attack is strictly restricted by the operating conditions of a scramjet during the optimization process. Additionally, the thrust of the scramjet varies with time due to the change of angle of attack, which will result in more collocation points to satisfy the convergence condition. The normal trajectory of an air-breathing hypersonic vehicle is shown in Figure C.5.

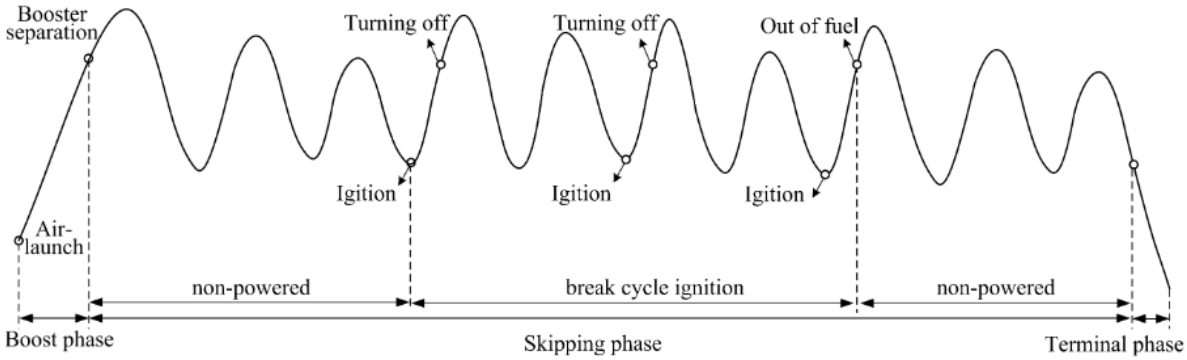


Figure C.5: The trajectory of an air-breathing hypersonic vehicle [111].

The flight scheme of such a scramjet-integrated hypersonic vehicle is as follows: after the booster has been separated, the vehicle gets through a skipping stage without power. When the moment occurs that the speed and altitude of the vehicle drop to the lowest ignition conditions, the scramjet ignites. The scramjet works in the so-called break cycle ignition mode, which can also be seen in Figure C.5.

D

Project Planning

This research followed an adaptive planning approach in which the project timeline changed throughout the thesis period. The nature of trajectory optimization research, particularly when designing something that has not been done before, required continuous reassessment and adjustment of the work plan. Different optimization methods were investigated and evaluated for their applicability to the HGV footprint generation problem.

When the project started, initial planning anticipated a more linear progression through established optimization techniques. However, the research process showed that several traditional methods encountered fundamental challenges when applied to the dual control problem with the full complexity of hypersonic flight dynamics.

D.1. Long-Term Planning

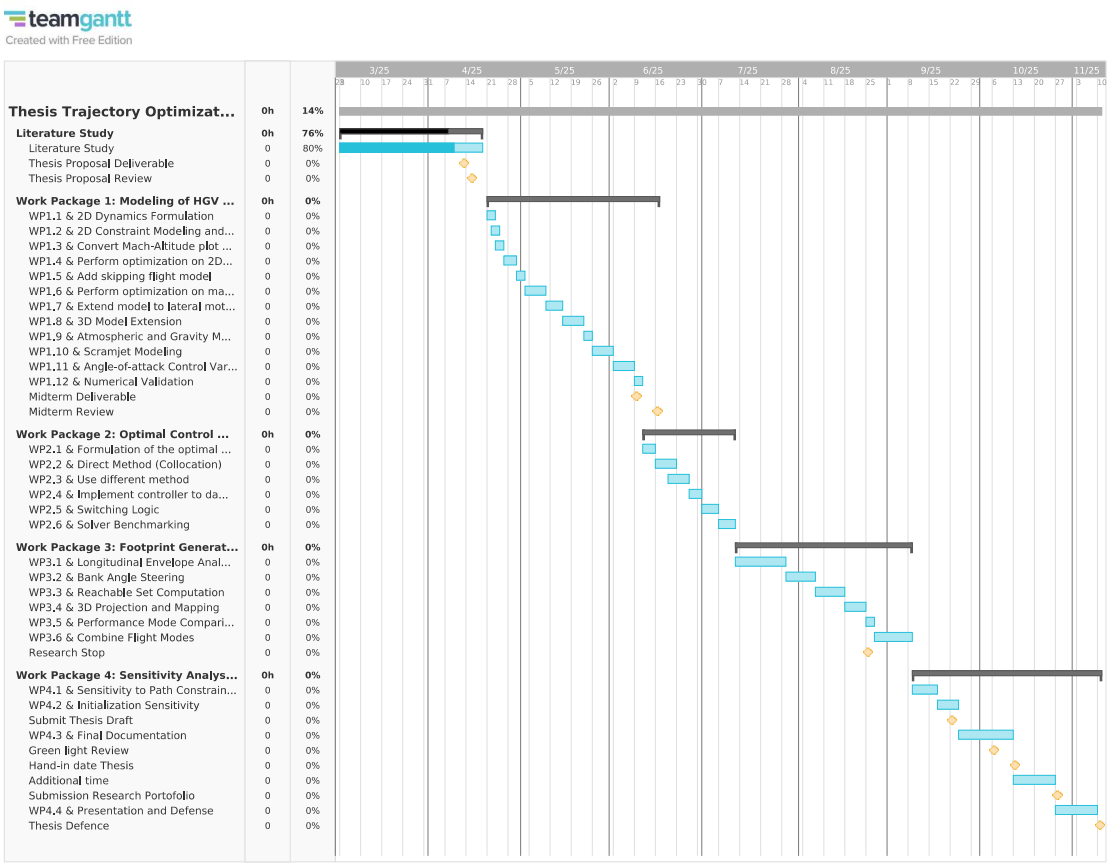
At the start of this research, a comprehensive long-term planning structure was developed to guide the execution of the thesis and ensure alignment with the requirements of both TU Delft and NLR. This initial planning framework consisted of the following deliverables:

1. **Literature review and Research Questions**
   (a) An extensive literature review in which the state-of-the-art in hypersonic trajectory optimization, footprint generation, and control strategies was presented.
   (b) Research gaps in current approaches.
   (c) Formulation of one main research question supported by multiple sub-questions to guide the investigation throughout the project.
2. **Work Package Subdivision**
   (a) Subdivide the research proposal into multiple work packages.
   (b) Each work package represents a specific component of the framework development, including deliverables.
   (c) Structured breakdown of technical objectives.
3. **Gantt Chart**
   (a) Development of a detailed Gantt chart that shows the planning for the thesis project. This Gantt chart has been updated multiple times throughout the project and has been discussed with the supervisors.
   (b) Allocation of time resources across the different work packages.
   (c) Identification of milestones and expected completion dates for the main deliverables.

The Gantt chart shown below shows the initial project planning developed at the beginning of the research phase. This early timeline reflected an optimistic assessment of how quickly various optimization

methods could be implemented and evaluated. As documented in Chapter 5, the research included extensive experimentation with multiple approaches before arriving at the reinforcement learning framework that ultimately proved to be successful. Throughout the project, the Gantt chart was continuously updated to keep up with the evolving research strategy. This iterative planning process ultimately led to a better final framework than would have been achieved by adhering to the initial timeline. The flexibility to pursue the most promising technical approaches was essential to achieve the research objectives.

The thesis has been completed within the nominal assigned time frame of *32 - 37 weeks*.



D.2. Short-Term Planning

In addition to the long-term planning, the project was managed through weekly meetings with the supervisors from both TU Delft and NLR. At each meeting, progress on previously assigned objectives was presented and discussed. Based on this review, new goals for the next week were established. This weekly cycle made sure there was continuous progress.

The weekly planning structure complemented the long-term Gantt chart, which was updated periodically. Although the Gantt chart established overall milestones and work package sequences, the weekly objectives gave the actual steps required to achieve those milestones. This combination of long-term planning and short-term actions to be taken gave an effective management of the research project throughout the thesis period.