ARL-TR-10413 • AUG 2026

DEVCOM ARMY RESEARCH LABORATORY

Pontryagin Neural Network Optimized Terrain Following Paths for Supersonic Atmospheric Vehicles

by Bradley T. Burchett

NOTICES

Disclaimers

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

Citation of manufacturer's or trade names does not constitute an official endorsement or approval of the use thereof.

Destroy this report when it is no longer needed. Do not return it to the originator.

ARL-TR-10413 ● AUG 2026

Pontryagin Neural Network Optimized Terrain Following Paths for Supersonic Atmospheric Vehicles

Bradley T. Burchett
DEVCOM Army Research Laboratory

REPORT DOCUMENTATION PAGE

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED	
		START DATE	END DATE
August 2026	Technical Report	4/5/2026	7/17/2026

4. TITLE AND SUBTITLE
Pontryagin Neural Network Optimized Terrain Following Paths for Supersonic Atmospheric Vehicles

5a. CONTRACT NUMBER	5b. GRANT NUMBER	5c. PROGRAM ELEMENT NUMBER
5d. PROJECT NUMBER	**5e. TASK NUMBER**	**5f. WORK UNIT NUMBER**

6. AUTHOR(S)
Bradley T. Burchett

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)	8. PERFORMING ORGANIZATION REPORT NUMBER
DEVCOM Army Research Laboratory ATTN: FCDD-RLA-WD Aberdeen Proving Ground, MD 21005	ARL-TR-10413

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
DISTRIBUTION STATEMENT A. Approved for public release: distribution unlimited.

13. SUPPLEMENTARY NOTES
ORCID: Bradley T. Burchett, 0000-0002-1934-0537

14. ABSTRACT
For many years, tactical aircraft have used terrain following or nap-of-the-Earth flight to avoid enemy detection. With advances in fixed-wing drones, such tactics are used by fast, small, unmanned vehicles as well. This report explores path planning for such vehicles using Pontryagin Neural Networks—a method previously used for space and low-speed atmospheric vehicles. This work includes several extensions from previous work, such as more realistic 3D flight dynamics and realistic supersonic aerodynamics. Multiple obstacles are combined into a single, smooth 2D function using Fourier sine series. By encoding actual Earth topography using the Fourier approximation, we demonstrate optimal flyable trajectories over actual terrain.

15. SUBJECT TERMS
path planning, optimization, drones, neural networks, guidance, Weapons Sciences

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES
a. REPORT	b. ABSTRACT	C. THIS PAGE		
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UU	34

19a. NAME OF RESPONSIBLE PERSON	19b. PHONE NUMBER *(Include area code)*
Bradley T. Burchett	(520) 691-5110

STANDARD FORM 298 (REV. 5/2020)
Prescribed by ANSI Std. Z39.18

Contents

List of Figures

List of Tables

1. Introduction

Modern optimization methods were developed largely from works by Hamilton,[1] Pontryagin et al.,[2] and Bellman.[3] Their foundational work led to the two main approaches to optimization: direct methods and indirect methods. Indirect methods for continuous time systems include the Euler–Lagrange[4] and Hamilton–Jacobi–Bellman methods.[3] Direct methods include pseudo-spectral[5] and orthogonal collocations. The Pontryagin Neural Network (PoNN)[6–9] combines features of both indirect and direct methods with notable improvements over each. The PoNN loss functions are found by applying the Euler–Lagrange equations to derive the necessary conditions for optimality. These conditions take the form of a system of nonlinear ordinary differential equations (ODEs) and hence a two-point boundary value problem (TPBVP). In the PoNN framework, this TPBVP is solved by discretizing along the time axis and using a set of basis functions to represent the solution. The PoNN also leverages two key innovations. First, it uses the Theory of Functional Connections (TFCs)[10,11] to automatically satisfy boundary conditions. The PoNN is an extreme learning machine, so it uses extreme TFCs. Second, PoNN uses the Fischer–Burmeister (FB) function to convert inequality constraints into equality constraints. Since the resulting loss functions are nonlinear, we use a Levenberg–Marquardt gradient search to iterate the loss functions until convergence.

In this report, we extend our previous work[12] by using a flight dynamics model that represents the vehicle as a point mass in coordinated flight, including realistic aerodynamics, and including multiple obstacles that may or may not overlap. The aerodynamics (lift and drag) reflect those of a small boost-glide vehicle. The results in this report use the High-Speed Army Reference Vehicle (HARV)[13] for dimensions, mass properties, and aerodynamic coefficients. The aerodynamic model was found by sampling three aerodynamic prediction codes (Missile DATCOM, Cart 3D, and Kestrel) and combining the results through recursive co-kriging. Aerodynamic look-up tables were generated by sampling the co-kriging surrogate model with an evenly spaced full-factorial grid of 13 points in Mach, 13 in angle of attack (AoA), and 11 in pitch deflection all at zero sideslip. The surrogate renders predictions of $C_L, C_D,$ and C_m at each point plus the nine possible sensitivities $\partial C_X/\partial\alpha, \partial C_X/\partial M, \partial C_X/\partial\delta, X \in \{L, D, m\}$ at each point. Although we use a particular vehicle, the methods presented here are platform agnostic, as long as the vehicle flies in a coordinated manner such that it maneuvers by bank-to-turn.

2. Optimal Control with Path Constraints Using PoNNs: Euler–Lagrange Optimal Control

In this report, we consider the optimal control problem,[4]

$$\min_{\mathbf{u}} J = \boldsymbol{\varphi}\left(\boldsymbol{x}(t_0), \boldsymbol{x}(t_f)\right) + \int_{t_0}^{t_f} \boldsymbol{L}(\boldsymbol{x}, \boldsymbol{u}, t)\, dt \tag{1}$$

$$\text{s.t. } \dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, t) \tag{2}$$

$$\boldsymbol{A}(\boldsymbol{x}(t_0), t_0) = \mathbf{0} \tag{3}$$

$$\boldsymbol{\Omega}(\boldsymbol{x}(t_f), t_f) = \mathbf{0} \tag{4}$$

$$\boldsymbol{\Gamma}(\boldsymbol{x}, \boldsymbol{u}, t) \leq \mathbf{0} \tag{5}$$

The necessary conditions for optimality can be found by defining the Hamiltonian as

$$H = L + \boldsymbol{\lambda}^T \boldsymbol{f} + \boldsymbol{\mu}^T \boldsymbol{\Gamma} \tag{6}$$

These conditions are then found from

$$\frac{\partial H}{\partial \boldsymbol{u}} = \mathbf{0} \tag{7}$$

and

$$\dot{\boldsymbol{\lambda}}^T = -\frac{\partial H}{\partial \boldsymbol{x}} = -\frac{\partial L}{\partial \boldsymbol{x}} - \boldsymbol{\lambda}^T \frac{\partial f}{\partial \boldsymbol{x}} - \boldsymbol{\mu}^T \frac{\partial \Gamma}{\partial \boldsymbol{x}} \tag{8}$$

In addition, Equation 5 is satisfied by enforcing the complementary slackness conditions

$$\mu_i \geq 0, \mu_i \Gamma_i = 0 \; \forall i = 1, \ldots, p \tag{9}$$

Equation 9 may be satisfied by using the FB function[8]

$$\phi(a, b) = \sqrt{a^2 + b^2} + a - b\,, \tag{10}$$

which, is zero only when conditions described by Equation 11 are satisfied.

$$\phi(a, b) = 0 \Leftrightarrow \begin{cases} a \leq 0 \\ b \geq 0 \\ ab = 0 \end{cases} \tag{11}$$

Thus, any inequality constraints such as Equation 5 may be enforced by an equality constraint $\phi(a, b) = 0$, where b is the nonnegative multiplier and a is the left-hand side (LHS) of inequality 5.

3. Flight Dynamics: 3 Degrees of Freedom (DOF) Point Mass in Coordinated Flight

3.1 Dynamics

A boost-glide vehicle in supersonic flight may be modeled by six nonlinear ODEs with a state vector consisting of downrange distance x, crossrange distance y, altitude z (positive toward zenith), air speed V, heading angle χ (positive counterclockwise from bird's eye), and flight path angle γ (climbing flight positive).

$$\dot{x} = V \cos\chi \cos\gamma \tag{12}$$

$$\dot{y} = V \sin\chi \cos\gamma \tag{13}$$

$$\dot{z} = V \sin\gamma \tag{14}$$

$$\dot{V} = F_x/m \tag{15}$$

$$\dot{\chi} = F_y/(mV \cdot \cos\gamma) \tag{16}$$

$$\dot{\gamma} = F_z/(mV) \tag{17}$$

where the forces are expanded as

$$F_x = T \cos\alpha - D - W \sin\gamma \tag{18}$$

$$F_y = (L + T \cdot \sin\alpha) \sin\mu \tag{19}$$

$$F_z = (L + T \cdot \sin\alpha) \cos\mu - W \cos\gamma \tag{20}$$

and lift and drag are $L = QSC_L$ and $D = QSC_D$, respectively. S is the reference area $S = \pi D^2/4$ and Q is the dynamic pressure $Q = \rho V^2/2$. We assume the flight is at low altitude and thus density ρ and speed of sound c are constant. The vehicle and atmospheric properties are shown in Table 1.

Table 1. HARV properties.

Mass (kg)	Characteristic length (cm)	Atmospheric density (kg/m^3)	Gravity (km/s^2)	Speed of sound (km/s)
m = 2.867	D = 5.08	ρ = 0.243	$g = 9.81(10)^{-3}$	c = 0.331

To keep quantities in the state vector close to an order of magnitude of unity, we use a length unit of kilometer. Thus V is in km/s, x in km, etc. For consistent units in Equations 12–17, we divide force in N by 1000, i.e.,

$$F\ [\text{N}/1000] = F\ [\text{kg} \cdot \tfrac{\text{km}}{\text{s}^2}].$$

The control vector is defined as $\mathbf{u} = [\mu\ \alpha\ T]^T$, where μ = bank angle, α = AoA, and T = thrust, which can be negative if drag devices need to be extended. Bank-to-turn is used to control the flight path in the horizontal plane. Angle-of attack modulates lift which controls the flight path in the vertical plane, including providing the load factor necessary for turns in the horizontal plane. The optimal path planning problem is to find the time history of the control vector such that the control vector 2-norm is minimized while satisfying all constraints (dynamic, geometric, and flight envelope).

3.2 Optimal Control: Euler–Lagrange Necessary Conditions

The optimal control problem is

$$\min_{\mathbf{u}} J = \int_{t_0}^{t_f} \boldsymbol{u}^T \boldsymbol{u}\, dt = \int_{t_0}^{t_f} w_\mu \mu^2 + w_\alpha \alpha^2 + w_T T^2\, dt \tag{21}$$

$$\textbf{s.t. } \dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, t) \qquad \text{(see 12–17)}$$

$$\boldsymbol{x}(t_0) = \boldsymbol{x_0}, \boldsymbol{x}(t_f) = \boldsymbol{x_f} \tag{22}$$

$$x(t_0) = \{x_0 \quad y_0 \quad z_0 \quad V_0 \quad \chi_0 \quad \gamma_0\} \tag{23}$$

$$C_{\boldsymbol{th}} \leq \mathbf{0} \tag{24}$$

$$C_{\boldsymbol{M}} \leq \mathbf{0}\ , \tag{25}$$

where the 3 DOF dynamics given in Equations 12–17 determine the vehicle motion (point mass in coordinated flight). We seek a path planning algorithm that will guide the vehicle above the terrain while maintaining a flyable air speed. Thus, constraint 24 ensures the vehicle will not impact the ground. Constraint 24 ensures that the vehicle maintains an air speed above stall. For computational purposes, this limit is placed at the minimum Mach number available in the aerodynamic tables. The terrain can be an arbitrary surface $\boldsymbol{Z}(\boldsymbol{x}, \boldsymbol{y})$ that is defined pointwise on a grid. This surface is transformed into a Fourier sine series, such that the algorithm sees a constraint defined by a smooth, easily differentiated function.

$$C_{\boldsymbol{th}} = Z_{th} - z, Z_{th} = \sum_{n=1}^{\infty} \sum_{m=1}^{\infty} c_{nm} \sin\left(\frac{n\pi x}{a}\right) \sin\left(\frac{m\pi y}{b}\right) \tag{26}$$

Thus C_{th} is negative for flight above the ground. The sine series is described in detail in the Appendix, Section A.2. The LHS of constraint 24 is the difference between minimum Mach number and the current Mach number:

$$C_M = M_{min} - M \tag{27}$$

such that C_M is negative for flight within the envelope.

The Hamiltonian is

$$H = w_\mu \mu^2 + w_\alpha \alpha^2 + w_T T^2 + \boldsymbol{\lambda}^T \boldsymbol{f} + \mu_{th} C_{th} + \mu_M C_M \tag{28}$$

The costate dynamics are derived from the necessary condition, $\dot{\boldsymbol{\lambda}}^T = -\frac{\partial H}{\partial \boldsymbol{x}}$. To facilitate this derivative, expand the terms in H that are dependent upon the states:

$$\begin{aligned} \boldsymbol{\lambda}^T \boldsymbol{f} &= \lambda_1 f_1(V,\chi,\gamma) + \lambda_2 f_2(V,\chi,\gamma) + \lambda_3 f_3(V,\gamma) + \lambda_4 f_4(F_x(V,\alpha,T,\gamma)) \\ &+ \lambda_5 f_5(V,\gamma,F_y(V,\alpha,T,\mu)) + \lambda_6 f_6(V,F_z(V,\alpha,T,\mu,\gamma)) + \mu_{th} C_{th}(x,y,z) + \mu_M C_M(z) \end{aligned} \tag{29}$$

Thus,

$$\dot{\lambda}_i = -\mu_{th} \frac{\partial C_{th}}{\partial x_i}, i = 1,2,3, x_i \in \{x,y,z\} \tag{30}$$

$$\dot{\lambda}_4 = -\frac{\partial H}{\partial V} - \mu_M \frac{\partial C_M}{\partial M}\frac{\partial M}{\partial V} = -\sum_{\boldsymbol{k=1}}^{\boldsymbol{6}} \lambda_k \left(\frac{\partial f_k}{\partial V}\right) - \mu_M \frac{\partial C_M}{\partial M}\frac{\partial M}{\partial V} \tag{31}$$

Where $\frac{\partial f_k}{\partial V}, k \in \{4,5,6\}$ has terms such as $\frac{\partial f_k}{\partial V} = \frac{\partial f_k}{\partial V} + \frac{\partial f_k}{\partial F_j}\frac{\partial F_j}{\partial V}$.

$$\dot{\lambda}_5 = -\frac{\partial H}{\partial \chi} = -\lambda_1 \frac{\partial f_1}{\partial \chi} - \lambda_2 \frac{\partial f_2}{\partial \chi} - \lambda_3 \frac{\partial f_3}{\partial \chi} \tag{32}$$

$$\dot{\lambda}_6 = -\frac{\partial H}{\partial \gamma} = -\lambda_1 \frac{\partial f_1}{\partial \gamma} - \lambda_2 \frac{\partial f_2}{\partial \gamma} - \lambda_3 \frac{\partial f_3}{\partial \gamma} - \lambda_4 \frac{\partial f_4}{\partial F_x}\frac{\partial F_x}{\partial \gamma} - \lambda_6 \frac{\partial f_6}{\partial F_z}\frac{\partial F_z}{\partial \gamma} \tag{33}$$

The second Euler–Lagrange necessary condition is $\frac{\partial H}{\partial \boldsymbol{u}} = 0$. Since the elements of the control vector only appear in the running loss and f_4, f_5, and f_6, we can write the following three equations:

$$0 = 2w_\mu \mu + \lambda_5 \frac{\partial f_5}{\partial F_y}\frac{\partial F_y}{\partial \mu} + \lambda_6 \frac{\partial f_6}{\partial F_z}\frac{\partial F_z}{\partial \mu} \tag{34}$$

$$0 = 2w_\alpha \alpha + \lambda_4 \frac{\partial f_4}{\partial F_x}\frac{\partial F_x}{\partial \alpha} + \lambda_5 \frac{\partial f_5}{\partial F_y}\frac{\partial F_y}{\partial \alpha} + \lambda_6 \frac{\partial f_6}{\partial F_z}\frac{\partial F_z}{\partial \alpha} \tag{35}$$

$$0 = 2w_T T + \lambda_4 \frac{\partial f_4}{\partial F_x}\frac{\partial F_x}{\partial T} + \lambda_5 \frac{\partial f_5}{\partial F_y}\frac{\partial F_y}{\partial T} + \lambda_6 \frac{\partial f_6}{\partial F_z}\frac{\partial F_z}{\partial T} \tag{36}$$

These three algebraic constraints will be solved simultaneously with the other constraints to determine the time history of the control vector.

From Equations 10 and 26, we can write

$$\phi(C_{th}, \mu_{th}) = 0 \tag{37}$$

and from Equations 10 and 27 likewise

$$\phi(C_M, \mu_M) = 0 \tag{38}$$

From Equations 12–17, 30–33, and 34–38, we can write 17 loss functions as follows:

$$\mathcal{L}_{1-6} = \boldsymbol{w_v}(\dot{\boldsymbol{x}} - \boldsymbol{f}) \tag{39}$$

$$\mathcal{L}_{7-9} = \boldsymbol{w_{\lambda r}}\left(\dot{\boldsymbol{\lambda}}_{\boldsymbol{i}} + \mu_{th}\frac{\partial C_{th}}{\partial \boldsymbol{x_i}} + \mu_M\frac{\partial C_M}{\partial \boldsymbol{x_i}}\right), \boldsymbol{i} \in \{1,2,3\} \tag{40}$$

$$\mathcal{L}_{10-12} = \boldsymbol{w_{\lambda v}}\left(\dot{\boldsymbol{\lambda}}_{\boldsymbol{i}} + \partial H/\partial \boldsymbol{x}_i\right), \boldsymbol{i} \in \{4,5,6\} \tag{41}$$

$$\mathcal{L}_{13-15} = \boldsymbol{w_{acc}}(\partial H/\partial \boldsymbol{u}_i), \boldsymbol{u_i} \in \{\mu, \alpha, T\} \tag{42}$$

$$\mathcal{L}_{16} = \boldsymbol{w_{th}}\left(\phi(C_{th}, \mu_{th})\right) \tag{43}$$

$$\mathcal{L}_{17} = \boldsymbol{w_{hd}}\left(\phi(C_{hd}, \mu_{hd})\right), \tag{44}$$

where $[\boldsymbol{w_v}, \boldsymbol{w_{\lambda r}}, \boldsymbol{w_{\lambda v}}, \boldsymbol{w_{acc}}, \boldsymbol{w_{th}}, \boldsymbol{w_{hd}}]$ are weights that determine the relative importance of each constraint, and the total loss is found by concatenating Equations 39–44 as

$$\mathcal{L} = [\mathcal{L}_{1-6}^T, \mathcal{L}_{7-9}^T, \mathcal{L}_{10-12}^T, \mathcal{L}_{13-15}^T, \mathcal{L}_{16}^T, \mathcal{L}_{17}^T] \tag{45}$$

3.3 PoNN Collocation: Reducing the Problem to Nonlinear Least Squares

In the PoNN method, states and co-states are solved through a collocation. However, to ensure that boundary conditions are satisfied, the solutions are contrived in the form of constrained expressions (CEs). These CEs consist of a free function and a linear combination of basis functions that automatically satisfy the boundary conditions, i.e.,

$$y_j^l(t) = g_j^l(t) + \sum_{\boldsymbol{k=1}}^{\boldsymbol{n_j}} \eta_{k,j} s_k^l(t)\,, \tag{46}$$

where the l superscript denotes the lth derivative with respect to time, n_j is the number of constraints for state j, $g_j^l(t)$ is the free function, and $s_k^l(t)$ are basis functions with properties that keep the problem well-conditioned (orthogonal polynomials, neural net activation functions, etc.). For a PoNN, the free function is the output of a single-layer neural network, such that,

$$g_j(t) = \sum_{q=1}^{L} \beta_{j,q} \sigma(w_q^T z + b_q),$$

where σ is the activation function, L is the number of hidden units, w_q^T are the input weights, and b_q are the input biases. Input weights and biases are chosen to span the input space and are held constant. Only the output weights $\beta_{j,q}$ are trained. Since the free function is linear in $\beta_{j,q}$, this training requires little computing time and $g_j(t)$ may be written compactly as $g_j(t) = \sigma^T \beta_j$. Note that as a collocation method, the independent variable z is scaled such that $-1 \leq z \leq 1$. Thus, for time-based problems, the temporal horizon is mapped to the domain of z by

$$z = -1 + \frac{2}{t_f - t_0}(t - t_0)$$

Thus, the scaling constant $b^2 = \frac{2}{t_f - t_0}$ must be applied to time derivatives to map Equation 46 from time to z domain:

$$g_j^l(t) = \frac{d^l g_j(z)}{dt^l} = \frac{d^l g_j(z)}{dz^l}\left(\frac{dz}{dt}\right)^l = \frac{d^l g_j(z)}{dz^l}(b^2)^l = (b^2)^l \left(\frac{d^l \boldsymbol{\sigma}}{dz^l}\right)^T \boldsymbol{\beta}_j$$

In this report, the second term in Equation 46 is expanded into products of switching functions and the basis functions at a given boundary condition. These switching functions equal 1 at the time where the respective boundary condition is in effect and 0 elsewhere and are denoted as $\boldsymbol{\Phi}_i$ hereafter. The switching functions used in this report are defined in the Appendix. See D'Ambrosio and Furfaro[6] for a full catalog of switching functions. For the application highlighted in this report, the initial and final positions are specified. The initial and final velocities are free. The $\boldsymbol{\Phi}_i$ from Table A5 of D'Ambrosio and Furfaro[6] are used in Equations 47 and 48, and are presented in the Appendix, Table A-1 for convenience.

Now, we can build the solution for the states and co-states in terms of CEs. These CEs will be substituted into Equations 39–44 to compute the losses, and the optimal solution will be found by minimizing the losses through adjusting the free variables $(\boldsymbol{\beta}_j, \mu, \alpha, T, \mu_{th}, \mu_{hd})$.

$$\boldsymbol{X}_i^T(z) = \boldsymbol{\beta}_{1:6}\left(\boldsymbol{\sigma} - \boldsymbol{\sigma}_0 \boldsymbol{\Phi}_1 - \boldsymbol{\sigma}_f \boldsymbol{\Phi}_2\right) + \boldsymbol{X}_0 \boldsymbol{\Phi}_1 + \boldsymbol{X}_f \boldsymbol{\Phi}_2 \tag{47}$$

$$\dot{\boldsymbol{X}}_i^T(z) = \boldsymbol{b^2}\left(\boldsymbol{\beta_{1:6}}\left(\boldsymbol{\sigma'} - \boldsymbol{\sigma}_0\boldsymbol{\Phi'_1} - \boldsymbol{\sigma}_f\boldsymbol{\Phi'_2}\right) + \boldsymbol{X_0}\boldsymbol{\Phi'_1} + \boldsymbol{X_f}\boldsymbol{\Phi'_2}\right) \quad (48)$$

$$\boldsymbol{\lambda_{xi}} = \boldsymbol{\beta_{7:12}}\boldsymbol{\sigma} \quad (49)$$

$$\dot{\boldsymbol{\lambda}}_{\boldsymbol{xi}} = b^2\boldsymbol{\beta_{7:12}}\boldsymbol{\sigma'} \quad (50)$$

In the equations, $\boldsymbol{\sigma}$ is the vector of neural net activation functions evaluated over the z domain, and $\boldsymbol{\sigma}_0$ and $\boldsymbol{\sigma}_f$ are the activation functions evaluated at $z = -1$ and $z = 1$, respectively. In this report, we chose the activation function $\boldsymbol{\sigma}$ to be hyperbolic tangent (tansig). The input weights and biases were chosen as $w_q = 5.2$, $b_q = w_q\left(-1 + \frac{2(q-1)}{L-1}\right), q = 1, \ldots, L$. Output weights were initially set to uniform random $-0.5 \leq \beta_{ij} \leq 0.5$, and multipliers μ_{th}, μ_{hd} were all initially set to unity. Equations 47–50 are substituted into Equations 39–44 to evaluate the losses using the PoNN collocation. The optimal solution is then found by a Levenberg–Marquardt search for $(\boldsymbol{\beta}_j, \mu, \alpha, T, \mu_{th}, \mu_{hd})$.

4. Results

We used analytic derivatives to find the gradients needed for the Levenberg–Marquardt search. Finite differences were not accurate enough to guide the search to adequate convergence. One must take care in finding analytic derivatives because the aerodynamic coefficients are functions of several variables and some second derivatives are needed to populate the Jacobian matrix. For instance, $L = f(M(V), Q(V), \alpha)$. Thus, one could find $\partial L/\partial\alpha$; $\frac{\partial L}{\partial V} = \frac{\partial L}{\partial M}\cdot\frac{\partial M}{\partial V} + \frac{\partial L}{\partial Q}\cdot\frac{\partial Q}{\partial V}$, and **J** has terms such as

$$\frac{\partial^2 L}{\partial\alpha\partial V} = \frac{\partial}{\partial\alpha}\left(\frac{\partial L}{\partial M}\cdot\frac{\partial M}{\partial V} + \frac{\partial L}{\partial Q}\cdot\frac{\partial Q}{\partial V}\right)$$

Although the surrogate model provides gradients of the aerodynamic coefficients with respect to Mach, AoA, and deflection, second derivatives were required for terms such as $\frac{\partial}{\partial\alpha}\left(\frac{\partial L}{\partial M}\right)$. Thus, we chose to use finite differencing for $\partial C_X/\partial\vartheta$ $X \in \{L, D, m\}$, $\vartheta \in \{M, \alpha, \delta\}$, and the corresponding second derivatives. First derivatives are found from first order forward differences. Second derivatives are found using, for instance,

$$\frac{\partial^2 C_L}{\partial\alpha\partial M} \approx \frac{C_L(M+\delta_M, \alpha+\delta_\alpha) - C_L(M+\delta_M, \alpha-\delta_\alpha) - C_L(M-\delta_M, \alpha+\delta_\alpha) + C_L(M-\delta_M, \alpha-\delta_\alpha)}{4\delta_M\delta_\alpha}$$

4.1 One Terrain Feature

Figure 1 shows the optimal trajectory over flat terrain with one spherical dome feature at 22.6-km downrange from launch. The vehicle is launched 10 m above the surface on a flight path approximately 15° above the horizon. It must reach an objective 53.2-km downrange on centerline with speed equal to initial speed and flight path angle depressed 15° below the horizon. The arrival time at the objective is fixed such that the vehicle could reach it easily by flying a trajectory in the line-of-sight plane without thrust and in the absence of drag. Two trajectories are depicted. In Figure 1a, the final time is extended by approximately 10 s, allowing freedom to maneuver in the horizontal plane, and choose a slower cruising speed. In Figures 1c–1f, a subset of state and control trajectories are displayed. In Figure 1a, the vehicle uses the extra time to maneuver horizontally around the obstacle. In Figures 1d and 1e, the algorithm takes advantage of the extra time by slowing to the minimum air speed, thus reducing drag and required thrust. In Figure 1f, the bank angle is very small and the AoA is small. The algorithm commands limited maneuvering in the vertical plane since the AoA is penalized 1) directly via the cost function and 2) indirectly because high AoA would induce additional drag and hence additional required thrust. To meet target air speed efficiently, the algorithm pushes thrust up on the final descent, at the opportune time while AoA is small and gravity enables forward acceleration—similar to the technique of a human pilot.

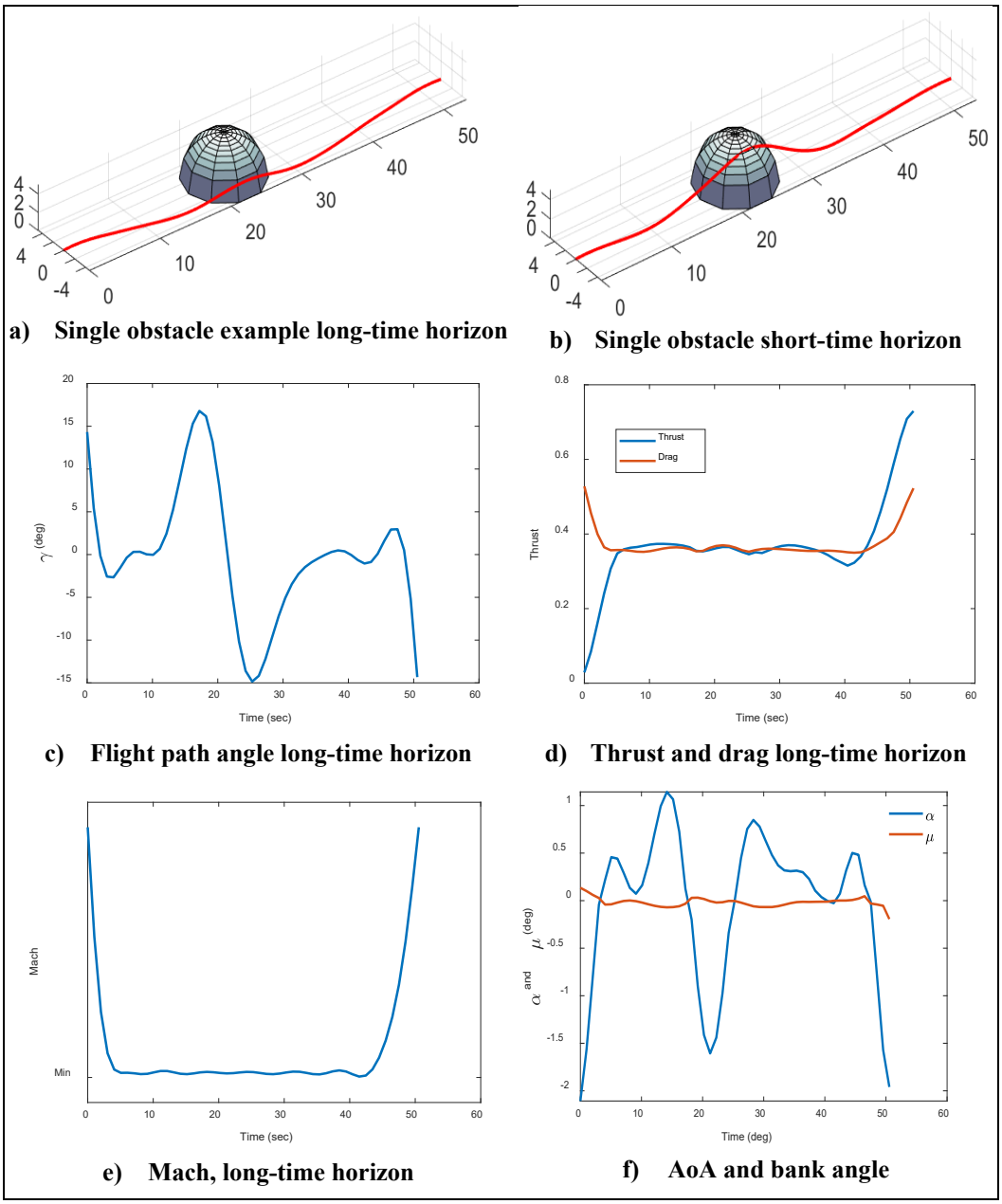


Figure 1. (a–f) Single-obstacle example solutions for long- and short-time horizons, point mass model with coordinated aerodynamics.

Figure 1b displays the trajectory for the short time horizon. The algorithm commands a more direct route due to the limited time. The state histories follow the same trends as for the long-time horizon. In other words, the algorithm tells the vehicle to slow down to minimum flyable speed and maintain steady air speed with thrust equal to drag. To meet the more stringent time requirement, it accelerates sooner, overshoots the terminal speed, and slows down during the last 5 s of flight. The flight path angle shows a striking resemblance to Figure 1c, only with larger minimum and maximum to reflect the steeper climb and dive while surmounting the obstacle closer to its center.

4.2 Four Terrain Features

Figure 2 shows an example in which four terrain features are modeled using the Fourier sine series. Figure 2a shows the actual terrain and the optimal trajectory. Figure 2b shows the Fourier approximation—a single 2D function capturing all four features. The Fourier terrain model includes the datum plane, so no additional constraint is required to keep the vehicle above the "hard deck." Due to the height and density of terrain features, the optimal trajectory takes the vehicle over three of the hills and around the fourth. In this case, the objective is 6 km left of centerline, making circumnavigation nearly impossible within the time constraint. The remaining launch and objective conditions are unchanged from the previous example.

Like the previous example, the vehicle slows down to reduce drag and hence thrust. It maintains a steady air speed with thrust approximately equal to drag for a large portion of the flight, and bank and AoA remain small throughout the flight.

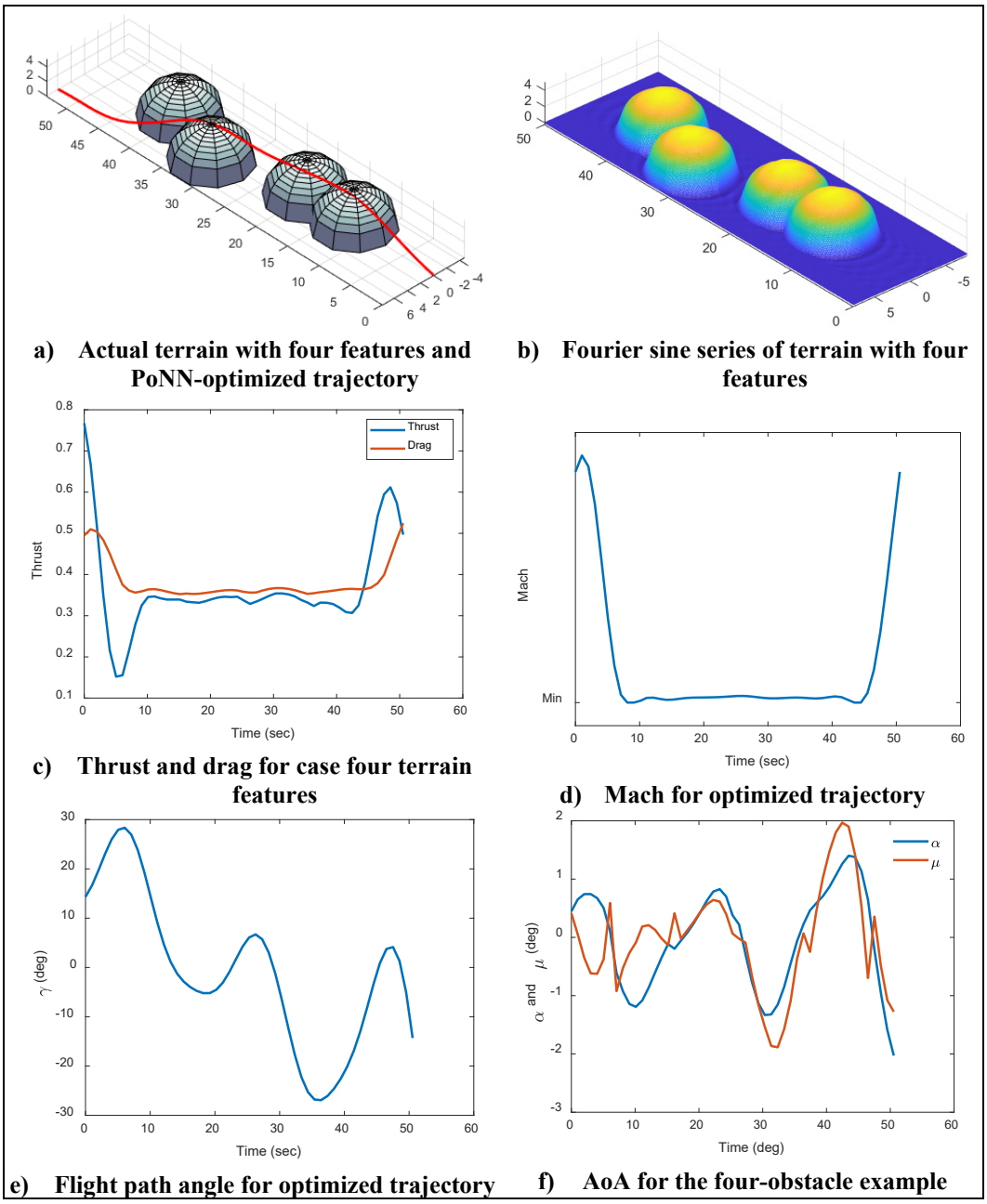


Figure 2. (a–f) Examples with four terrain features.

4.3 Flight through Actual Terrain

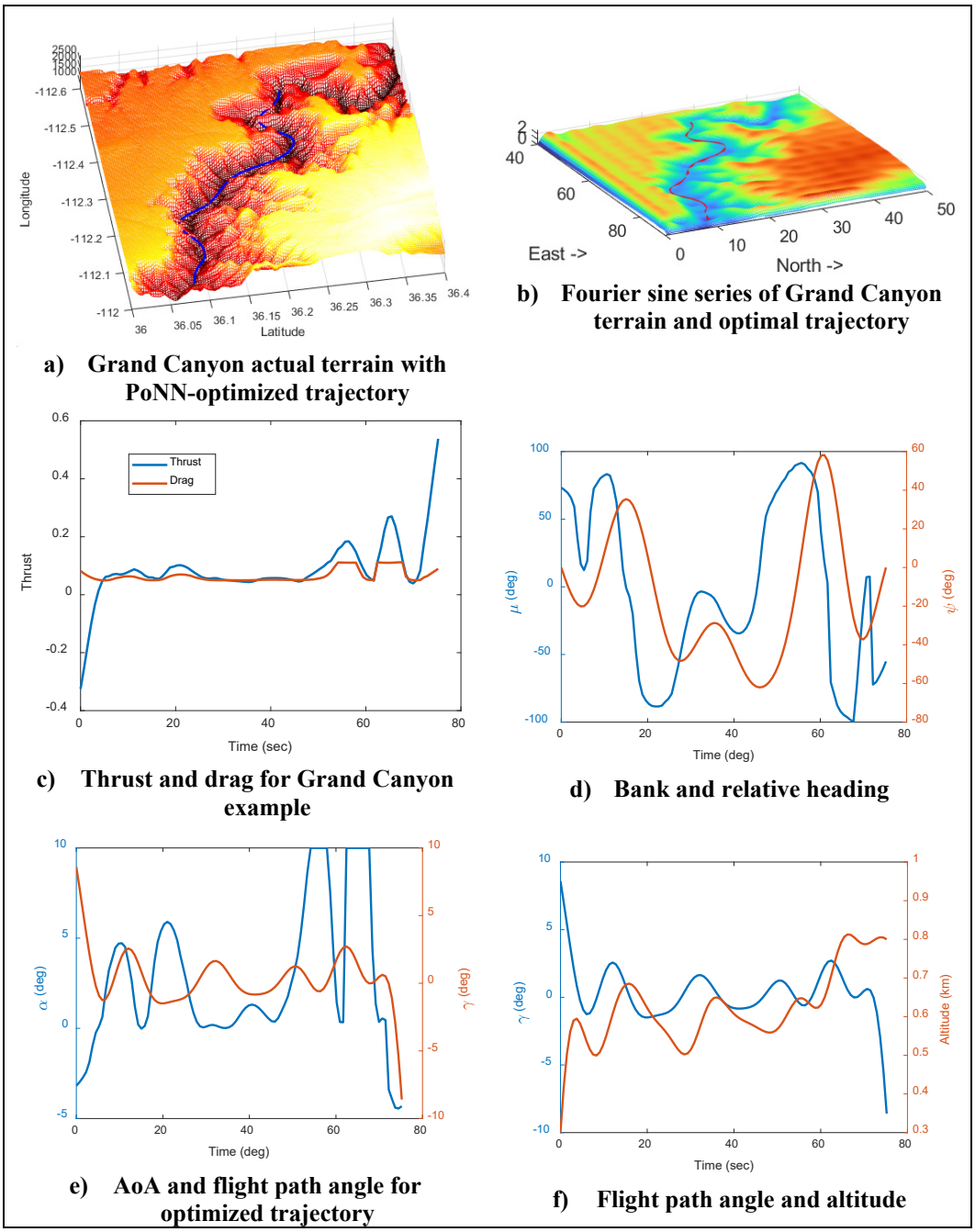
To simulate flight through actual terrain, we downloaded Earth topography and used the Fourier sine series (Appendix A-2) to model it as a smooth function. We downloaded small regions of terrain elevation from the MathWorks web map server. Each region is approximately 1 degree of latitude and longitude in height and width. We asked for samples spaced 10 arc seconds in latitude and longitude so the terrain elevation data fits in a matrix of dimension 360 × 360. We then used

features of the MATLAB mapping toolbox to convert from geodetic to East-North-Up (ENU) coordinates. We apply a 51 term sine series to convert the ENU elevation data into smooth functions.

Next, we used the PoNN optimization algorithm to find an optimal, flyable path through the terrain. Optimality is defined as maintaining the lowest possible altitude while maintaining a minimum air speed using the minimal thrust and keeping AoA and bank angle within the flight envelope. The penalty weights were set to $w_v = w_a = w_\alpha = 0.25, w_T = w_{\lambda_v} = w_{acc} = 0.125, w_{th} = 0.5, w_{\lambda_r} = 0.0125, w_\mu = 0.02, w_Z = 10$. We applied the algorithm to flight through the Grand Canyon (Arizona). Results are illustrated in Figures 3 and 4. Launch and target conditions were chosen approximately 60 km apart along the canyon floor. The time of flight was set to allow maneuvering at minimum air speed while reaching the target in time.

An additional constraint was added to restrict AoA to positive values. This constraint was included to prevent engine stall on vehicles with chin inlet ramjets or turbojets. The PoNN converged in approximately 20 iterations. Figure 3a shows the flight path through the terrain in geodetic coordinates. The terrain height is exaggerated due to axis scaling. Figure 3b plots the trajectory over the Fourier approximation of the terrain in ENU coordinates.

The vehicle traces a path close to the canyon floor. The minimum thrust condition causes the vehicle to slow down, reducing drag (Figure 3c). Figure 3d shows the bank angle and relative heading angle time histories in degrees. Note that bank reaches ±90° during the tightest turns. Figure 3e shows the AoA and flight path angle. The AoA is positive for most of the trajectory except at the very beginning and ending. In addition, the AoA is limited to 10° and reaches this limit during the final two turns. Figure 3f shows the flight path angle and altitude. Note that trends in flight path angle lead trends in altitude as $\dot{z} \propto \sin\gamma$.



a) Grand Canyon actual terrain with PoNN-optimized trajectory

b) Fourier sine series of Grand Canyon terrain and optimal trajectory

c) Thrust and drag for Grand Canyon example

d) Bank and relative heading

e) AoA and flight path angle for optimized trajectory

f) Flight path angle and altitude

Figure 3. (a–f) Actual terrain examples, flying through the Grand Canyon.

The result in Figure 3 represents the best possible outcome. Because we are using a gradient-based optimization, convergence is susceptible to getting trapped in local minima depending on the initial guess. This problem is exacerbated by the Fourier approximation of the terrain. If the path found at iteration k flies between two local peaks, that portion of the solution will be trapped in a suboptimal path due to local gradients. For the terrain shown in Figure 3a, the trajectory would typically take a shortcut over the topographical pass at 36.2 latitude, −112.4 longitude. The

corresponding point in Figure 3b has a positive definite Hessian—that is, a local minima. Attempts to overcome this difficulty using particle swarm optimization induced other problems such as control signal chatter. For terrain with many local minima, it is best to break the trajectory into several segments to be solved separately. Unfortunately, this requires repeated user intervention.

To validate the flyability of the optimal (Grand Canyon) trajectory, we devised a closed-loop simulation where the vehicle is expected to fly the 3D path designed using PoNN. The closed-loop simulation uses the dynamic model of Equations 12–20 and aero database used in PoNN optimization. It uses an outer guidance loop where all six states from the PoNN trajectory are available as reference signals. Equations 12–14 are invoked to compute $\dot{x}_{ref}$, $\dot{y}_{ref}$, and $\dot{z}_{ref}$. The outer loop guidance is computed as $\mathbf{v} = \mathbf{K}\boldsymbol{\epsilon}$, where

$$\boldsymbol{\epsilon} = \left[x_{ref} - x\ \dot{x}_{ref} - \dot{x}\ y_{ref} - y\ \dot{y}_{ref} - \dot{y}\ z_{ref} - z\ \dot{z}_{ref} - \dot{z}\right]^T$$

We designed a feedback linearization/backstepping controller to determine the physical controls from virtual controls. The details of this controller are included in the Appendix, Sections A.3 and A.4.

Results of the closed-loop simulation are compared with the PoNN predictions in Figure 4. Figure 4a shows the AoA time history for the PoNN prediction, closed-loop integration (FB Lin), and a neural net reconstruction method (NNet) described in the Appendix, Section A.5. The AoA is limited to 12° for the PoNN prediction. All methods prefer positive angles of attack. The "real-time" integration requires higher angles of attack to converge to the specified trajectory. Figure 4b shows the bank angle. The reconstruction methods predict higher bank angles approaching ±90° six times while the PoNN solution only reaches ±90° four times. The bank angle from feedback linearization real-time prediction and offline neural net predictions are strikingly similar. Figure 4c shows the thrust required to fly the optimal trajectory. Neural net and feedback linearization results match exactly. Both show trends and amplitudes similar to the PoNN prediction. Figure 4d shows the PoNN trajectory and real-time tracking solution. The real-time tracker lags somewhat in altitude with approximately 10-m maximum tracking error. Horizontal tracking error is negligible.

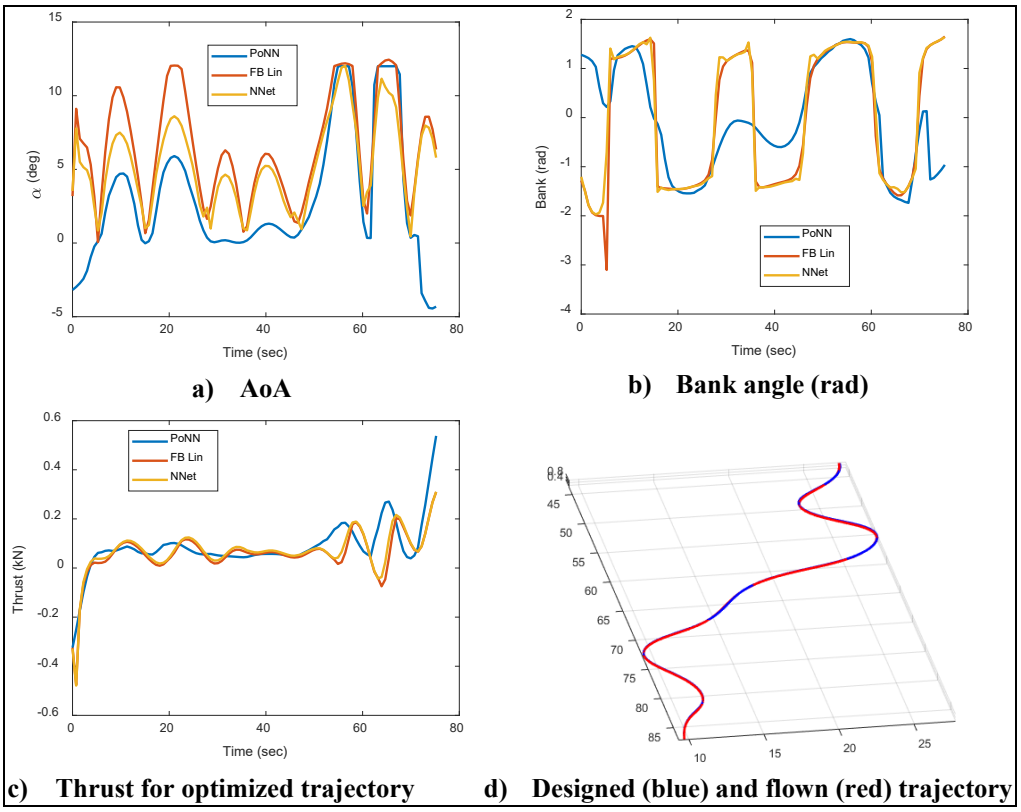


Figure 4. (a–d) Flyability study, Grand Canyon example.

Figure 5 shows two examples of the solution getting trapped in local minima largely due to undulations in terrain near the lowest corridor. Figures 5a and 5b show a trajectory over Lake Powell, Utah. The lakebed is clearly the lowest path; however, minor peaks near the lake push the suboptimal trajectory away from the lakebed and through an adjacent low pass near (38.15, −110.8). Figures 5c and 5d show a suboptimal trajectory over terrain near Lake Lucerne, Switzerland. The lowest path is evident in Figure 5d somewhat to the right of the plotted suboptimal path. Due to the many local maxima and minima in the Fourier approximation (Figure 5d), the trajectory gets trapped near the plotted trajectory, and then it makes small oscillations left and right with subsequent search iterations.

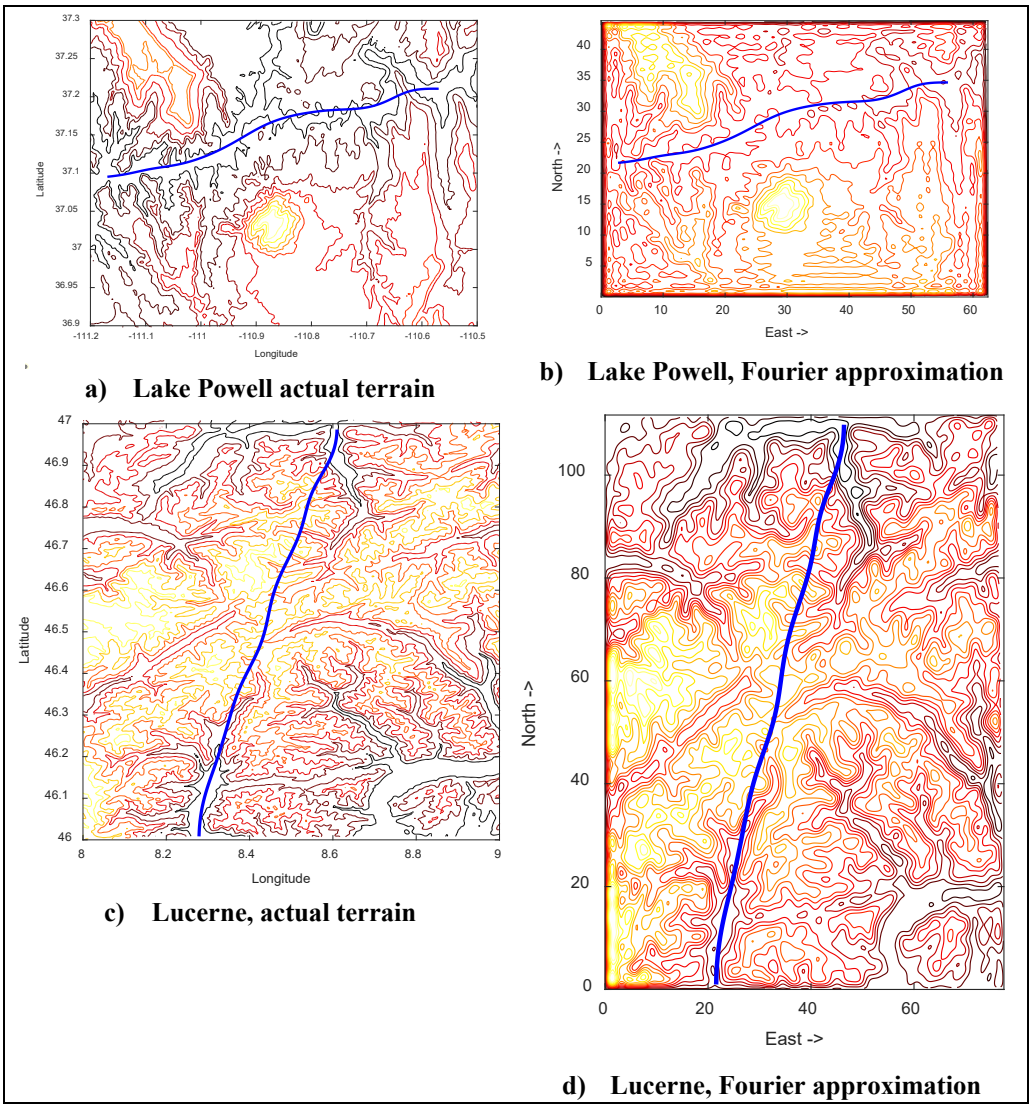


Figure 5. (a–d) Solutions trapped in local minima due to Fourier aliasing.

5. Conclusion

In this report, we extended previous work in optimal path planning. We incorporated more realistic point mass dynamics, more realistic speeds, and realistic aerodynamics into the PoNN optimization algorithm. We demonstrated optimal path planning over terrain with multiple features up to 5 km in height.

We used Fourier terrain encoding to demonstrate our approach using actual topographical data. The algorithm found a low, flyable trajectory through a portion of the Grand Canyon. We validated the flyability of this trajectory using a closed-loop time simulation with feedback linearization and backstepping guidance. The PoNN-predicted control signals were similar to those generated during the closed-loop simulation. Closed-loop simulation control signals were somewhat larger in amplitude to achieve convergence to the commanded trajectory.

The method presented will render a flyable trajectory if the problem is feasible. User-selected constraints (time of flight and initial and final conditions) could make the problem infeasible. Since the optimization search is gradient based, the method can get trapped in local minima, most notably saddle points in the terrain. Once it is trapped in a local minimum, the algorithm cannot converge to the lowest available path.

6. References

1. Hamilton WR. On a general method of expressing the paths of light, & of the planets, by the coefficients of a characteristic function. P.D. Hardy; 1833.
2. Pontryagin LS et al. The mathematical theory of optimal processes. Interscience; 1962.
3. Bellman R. Dynamic programming. Princeton University Press; 1957.
4. Bryson AE, Ho Y. Applied optimal control. Ginn and Company; 1969.
5. Elnagar G, Kazemi MA, Razzaghi M. The pseudospectral legendre method for discretizing optimal control problems. IEEE Trans Auto Control. 1995;38(10):1793–1796.
6. D'Ambrosio A, Furfaro R. Learning fuel-optimal trajectories for space applications via Pontryagin neural networks. Aerospace. 2024;11(3):228. https://doi.org/10.3390/aerospace11030228
7. D'Ambrosio A, Schiassi E, Curti R, Furfaro R. Pontryagin neural networks with functional interpolation for optimal intercept problems. Mathematics. 2021;9(9):996. https://doi.org/10.3390/math9090996
8. D'Ambrosio A, Benedikter B, Furfaro R. Physics-informed Pontryagin neural networks for path-constrained optimal control problems. J Guid Cont Dyn. 2025;48(8):1861–1877. https://doi.org/10.2514/1.G008854
9. Conti M, D'Ambrosio A, Circi C, Furfaro R. Pontryagin neural networks for high-fidelity cislunar orbital transfers. J Guid Cont Dyn. 2026;29(7). https://doi.org/10.2514/1.G009693
10. Mortari M. The theory of connections: connecting points. Mathematics. 2017;5(4):57. https://doi.org/10.3390/math5040057
11. Mortari M. Least-squares solution of linear differential equations. Mathematics. 2017;5(4):48. https://doi.org/10.3390/math5040048.
12. Burchett BT. Optimal trajectory design for vehicles navigating around obstacles using Pontryagin neural networks. Proceedings of the AIAA SciTech Forum 2026; 2026 Jan 12–16; Orlando, FL.
13. Vasile JD et al. High-speed Army reference vehicle. DEVCOM Army Research Laboratory (US); 2022 Aug. Report No.: ARL-TN-1128.

Appendix. Additional Details

A.1 Switching Functions

Table A-1 shows the switching functions used to assure that the Pontryagin Neural Network (PoNN) solution intersects the prescribed initial and final conditions. Switching functions apply the theory of functional connections to replace the free function trajectory solution with the prescribed end conditions.

Table A-1. Switching functions for two constraints, one on $x(z_0)$ and one on x(z_f), defined on domain of $z \in [z_0, z_f]$.

Derivative	Initial value $\boldsymbol{\Phi_1(z)}$	Final value $\boldsymbol{\Phi_2(z)}$
(.)	$\frac{(z_f - z)}{\Delta z}$	$\frac{(z - z_0)}{\Delta z}$
$\frac{d}{dz}(.)$	$\frac{-1}{\Delta z}$	$\frac{1}{\Delta z}$
$\frac{d^2}{dz^2}(.)$	0	0

A.2 Fourier Sine Series Coding of Terrain

To represent composite arbitrary terrain features as a single 2D smooth function, we used a 2D real-valued Fourier sine series. The terrain altitude $Z(x, y)$ is encoded into a sum of harmonics with coefficients c_{nm} determined by Equation A-1.

$$c_{nm} = \frac{4}{ab}\int_0^b \int_0^a Z(x,y) \sin\left(\frac{n\pi x}{a}\right) \sin\left(\frac{m\pi y}{b}\right) dx\, dy \tag{A-1}$$

The integrals may be evaluated in MATLAB using trapz. $Z = Z(x, y)$ is a pointwise grid of the highest terrain at (x, y). The series can be truncated at any arbitrary number of harmonics. For the examples shown in this report, we used $n, m = 1,2, \ldots ,51$.

Once the terrain is encoded in c_{nm}, this mapping is fixed during the optimization. Given a candidate trajectory $(x(t), y(t), z(t))$, the height of terrain along this path may be computed as a batch using Equation A-2:

$$Z_{th} = \sum_{n=1}^{\infty} \sum_{m=1}^{\infty} c_{nm} \sin\left(\frac{n\pi x}{a}\right) \sin\left(\frac{m\pi y}{b}\right), \tag{A-2}$$

where $0 \le x \le a$, $0 \le y \le b$, and the series is truncated at 51 terms. Since Equations A-2 through A-7 may be evaluated in batch mode, the computational burden of 51 terms does not significantly slow the search. Note that per Equation 30, first derivatives $\frac{\partial C_{th}}{\partial x_i} = \frac{\partial Z_{th}}{\partial x_i}$ will be needed to define the co-state dynamics,

$$\frac{\partial Z_{th}}{\partial x} = \frac{\pi}{a}\sum_{n=1}^{\infty}\sum_{m=1}^{\infty} nc_{nm}\cos\left(\frac{n\pi x}{a}\right)\sin\left(\frac{m\pi y}{b}\right) \tag{A-3}$$

$$\frac{\partial Z_{th}}{\partial y} = \frac{\pi}{b}\sum_{n=1}^{\infty}\sum_{m=1}^{\infty} mc_{nm}\sin\left(\frac{n\pi x}{a}\right)\cos\left(\frac{m\pi y}{b}\right) \tag{A-4}$$

and $\frac{\partial C_{th}}{\partial z} = -1$.

Since the loss functions contain first derivatives of C_{th}, the Jacobian matrix will contain the second derivatives

$$\frac{\partial^2 Z_{th}}{\partial x \partial y} = \frac{\pi^2}{ab}\sum_{n=1}^{\infty}\sum_{m=1}^{\infty} nmc_{nm}\cos\left(\frac{n\pi x}{a}\right)\cos\left(\frac{m\pi y}{b}\right) \tag{A-5}$$

$$\frac{\partial^2 Z_{th}}{\partial x^2} = -\frac{\pi^2}{a^2}\sum_{n=1}^{\infty}\sum_{m=1}^{\infty} n^2 c_{nm}\sin\left(\frac{n\pi x}{a}\right)\sin\left(\frac{m\pi y}{b}\right) \tag{A-6}$$

$$\frac{\partial^2 Z_{th}}{\partial y^2} = -\frac{\pi^2}{b^2}\sum_{n=1}^{\infty}\sum_{m=1}^{\infty} m^2 c_{nm}\sin\left(\frac{n\pi x}{a}\right)\sin\left(\frac{m\pi y}{b}\right) \tag{A-7}$$

with $\frac{\partial^2 C_{th}}{\partial z \partial z} = \frac{\partial^2 C_{th}}{\partial z \partial y} = \frac{\partial^2 C_{th}}{\partial z \partial x} = 0$.

A.3 Feedback Linearization Guidance

To explore the flyability of an optimal trajectory, we built a closed-loop simulation based on the dynamics of Equations 12–17. Angle of attack (α), bank angle (μ), and thrust (T) are chosen as control inputs. This simulation has multiple loops. The outermost loop uses the cartesian coordinates (x, y, z) from the optimal trajectory as functions of time to guide the vehicle. Flyability is defined as the ability of the vehicle to reach the specified waypoints at the given times with control inputs within the vehicle capability, for instance, $-12^\circ \leq \alpha \leq 12^\circ$.

To determine the control signals required to track the prescribed waypoints, we use a two-step feedback linearization scheme. For the outer loop, we define three outputs ($x, y,$ and z) and three inputs ($F_x, F_y,$ and F_z), such that the nonlinear system of Equations 12–17 is "square" and affine in the inputs. This allows us to apply feedback linearization directly. See Slotine and Li[1] and Burchett[2] for a full treatment as our description here will be terse.

Apply Equation 9 from Burchett[2] to Equations 12–14:

$$\ddot{y}_j = L_{\mathbf{f}}^2 h_j + \sum_{i=1}^{m} L_{g_i} L_{\mathbf{f}}^1 h_j u_i \tag{A-8}$$

[1] Slotine JJ, Li W. Applied nonlinear control. Pearson; 1991.

[2] Burchett BT. Feedback linearization guidance for approach and landing of reusable launch vehicles. Proceedings of the 2005 American Control Conference; 2005 June 8–10; Portland, OR. vol. 3, p. 2093–2097. https://doi.org/10.1109/ACC.2005.1470279

resulting in

$$\ddot{x} = \cos\chi\cos\gamma\frac{F_x}{m} - \sin\chi\frac{F_y}{m} - \cos\chi\sin\gamma\frac{F_z}{m} \quad \text{(A-9)}$$

$$\ddot{y} = \sin\chi\cos\gamma\frac{F_x}{m} + \cos\chi\frac{F_y}{m} - \sin\chi\sin\gamma\frac{F_z}{m} \quad \text{(A-10)}$$

$$\ddot{z} = \sin\gamma\frac{F_x}{m} + \cos\gamma\frac{F_z}{m} \quad \text{(A-11)}$$

and yielding the decoupling matrix

$$\mathbf{E} = \frac{1}{\mathrm{m}}\begin{bmatrix} \cos\chi\cos\gamma & -\sin\chi & -\cos\chi\sin\gamma \\ \sin\chi\cos\gamma & \cos\chi & -\sin\chi\sin\gamma \\ \sin\gamma & 0 & \cos\gamma \end{bmatrix} \quad \text{(A-12)}$$

Applying decoupling transforms the system of Equations 12–17 into the linear system:

$$\dot{\mathbf{z}} = \mathbf{A}\mathbf{z} + \mathbf{B}\mathbf{v} \qquad \mathbf{y} = \mathbf{C}\mathbf{z} \quad \text{(A-13)}$$

where

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & & & & & \\ 0 & 0 & & & & & \\ & & 0 & 1 & & & \\ & & 0 & 0 & & & \\ & & & & 0 & 1 & \\ & & & & 0 & 0 & \end{bmatrix} \quad \mathbf{B} = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ & 0 & 0 \\ & 1 & 0 \\ & & 0 \\ & & 1 \end{bmatrix}$$

$$\mathbf{C} = \begin{bmatrix} 1 & 0 & 0 & & & \\ 0 & 0 & 1 & 0 & & \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

We can design feedback for the system of Equation A-13 using $\mathbf{v} = -\mathbf{K}\mathbf{z}$, where $\mathbf{K}$ is found as the linear-quadratic regulator (LQR) gain. In fact, we use the LQR with integrators method described in Burchett[2] and Kuo and Golnaraghi.[3] This leads to a linear feedback gain of the form

$$\mathbf{K} = \begin{bmatrix} K_{px} & K_{dx} & 0 & 0 & 0 & 0 & K_{ix} & 0 & 0 \\ 0 & 0 & K_{py} & K_{dy} & 0 & 0 & 0 & K_{iy} & 0 \\ 0 & 0 & 0 & 0 & K_{pz} & K_{dz} & 0 & 0 & K_{iz} \end{bmatrix}$$

The physical control $\mathbf{u}$ is then found from the pseudo control $\mathbf{v}$ as $\mathbf{u} = \mathbf{E}^{-1}\mathbf{v}$, where $\mathbf{u} = \begin{bmatrix} F_x & F_y & F_z \end{bmatrix}^T$.

An inner loop is needed to convert the force vector to control inputs. We explore a backstepping method in Section A.4 and a neural network approach in Section A.5.

A.4 Ad Hoc Backstepping

Since the vehicle cannot command $\mathbf{u} = \begin{bmatrix} F_x & F_y & F_z \end{bmatrix}^T$ directly, we need to design an inner guidance loop that will map $\begin{bmatrix} F_x & F_y & F_z \end{bmatrix}$ to $[\alpha \ \mu \ T]$. We achieve this through an ad hoc backstepping loop. The backstepping loop is illustrated in Figure A-1. This

[3] Kuo BC, Golnaraghi F. Automatic control systems. 8th ed. Wiley; 2003.

inner loop is intended to drive the control vector $[\alpha\ \mu\ T]$ to values that will yield the desired force vector. The desired force vector $\vec{\mathbf{F}}_{des}$ is provided by the feedback linearization loop described in Section A.3. A nonlinear block $\mathbf{g}(\mathbf{x}, \mathbf{u})$ applies Equations 18–20 to the current state and control vectors to find the current force $\vec{\mathbf{F}}_{act}$.

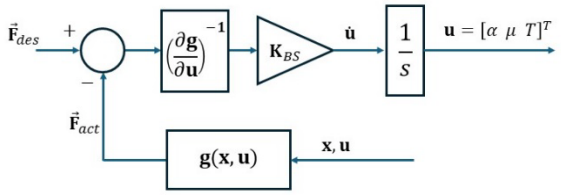


Figure A-1. Ad hoc backstepping loop.

The forward path of this loop takes the difference between desired and actual force. A Taylor series expansion is used to determine the direction that the control vector should be modified to generate the desired force vector

$$\dot{\mathbf{u}} = \mathbf{K}_{BS}\left(\frac{\partial \mathbf{g}}{\partial \mathbf{u}}\right)^{-1}\left(\vec{\mathbf{F}}_{des} - \vec{\mathbf{F}}_{act}\right),$$

where $\partial \mathbf{g}/\partial \mathbf{u}$ is a Jacobian matrix derived from Equations 18–20.

$$\frac{\partial \mathbf{g}}{\partial \mathbf{u}} = \begin{bmatrix} -T \cdot \sin\alpha - \dfrac{\partial D}{\partial \alpha} & 0 & \cos\alpha \\ \dfrac{\partial L}{\partial \alpha}\sin\mu + T\cos\alpha\sin\mu & L\cos\mu + T\sin\alpha\cos\mu & \sin\alpha\sin\mu \\ \dfrac{\partial L}{\partial \alpha}\cos\mu + T\cos\alpha\cos\mu & -L\sin\mu - T\sin\alpha\sin\mu & \sin\alpha\cos\mu \end{bmatrix}$$

$\mathbf{K}_{BS}$ is a positive diagonal gain matrix such that the inner loop converges to $\vec{\mathbf{F}}_{act} \approx \vec{\mathbf{F}}_{des}$ much faster than the vehicle dynamics. We set $\mathbf{K}_{BS} = 300\mathbf{I}$.

A.5 Neural Net Model Inverting Equations 18–20

We developed a second, offline method to map the desired force vector to a corresponding control vector. This method used the "distal teacher" neural network architecture[4]—an early predecessor to adaptive-critic methods. This approach uses neural network function approximation to find the inverse of a "many-to-one" algebraic system. In other words, the force vector computed by Equations 18–20 is unique, but its inverse is not unique. For instance, if we flip the signs of both α and μ in Equation 19, we will render the same F_y. I.E. $-\sin\mu = \sin(-\mu)$ and $L(-\alpha) = -L(\alpha)$.

[4] Jordan MI, Rumelhart DE. Forward models: supervised learning with a distal teacher. Cogn Sci. 1992;16(3):307–354.

To find one of the many inverse solutions, we first train two feedforward networks: one that models the aerodynamic tables for C_L and C_D (Aero Network), and one that models the forward solution of Equations 18–20 (Physics Network). After these networks are adequately trained, we form the series of networks shown in Figure A–2. This network is further trained such that the full path from left to right acts as an identity function. See Jordan and Rumelhart[4] for full details on this method.

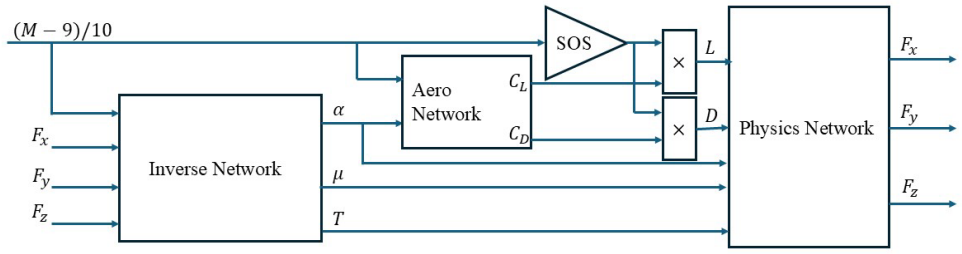


Figure A-2. Training the inverse network.

After the Inverse Network is trained, it can be used in a stand-alone fashion to invert Equations 18–20, that is, to compute the control vector that will render the desired forces. We found this approach to be successful in recreating the control vector offline. However, it was not accurate enough to be used in place of the backstepping loop described in Section A.4. Thus, closed-loop flyability simulations were performed using the feedback linearization/backstepping combination. The results of offline control reconstruction using the Inverse Network are plotted in Figure 4 with the control signals found during closed-loop simulation. The neural net result compares well with the backstepping result.

List of Symbols, Abbreviations, and Acronyms

2D/3D	Two-/Three-Dimensional
AoA	Angle of Attack
CE	Constrained Expression
DOF	Degrees of Freedom
ENU	East-North-Up
FB	Fischer–Burmeister
HARV	High-Speed Army Reference Vehicle
LHS	Left-Hand Side
LQR	Linear-Quadratic Regulator
NNet	Neural Net Reconstruction Method
ODE	Ordinary Differential Equation
PoNN	Pontryagin Neural Network
TFC	Theory of Functional Connection
TPBVP	Two-Point Boundary Value Problem

NOMENCLATURE

$\{x,y,z\}$	Vehicle *cg* Position in Ground Frame (km)
$\{\chi,\gamma,\mu\}$	Heading Angle, Flight Path Angle, and Bank in Ground Frame (rad)
V	$\|\mathrm{V}\|_2$, Speed Relative to Ground Frame (km/s)
$\boldsymbol{f}$	System Dynamics $\mathbb{R}^n$
Γ	Path Constraints $\mathbb{R}^p$
L	Cost Functional
m	Vehicle Mass (kg)
x	System State Vector $\mathbb{R}^n$
t	Time (s)
0	Constraint at Initial Time, t_0

$\mathbf{u}$	Control Vector $\mathbb{R}^m$
$\boldsymbol{\lambda}$	Lagrange Multipliers (Dynamics)
$\boldsymbol{\mu}$	Lagrange Multipliers (Path Constraints)
Φ_i	Switching Function
$c_\gamma, s_\gamma, t_\gamma$	cos(γ), sin (γ), tan(γ)
SUBSCRIPT	
$_0$	Initial Conditions
$_f$	Final (Objective) Conditions
SUPERSCRIPT	
^{T}	Matrix Transpose

1 (PDF)	DEFENSE TECHNICAL INFORMATION CTR DTIC OCA
1 (PDF)	DEVCOM ARL FCDD RLB CB TECH LIB