73rd International Astronautical Congress (IAC), Paris, France, 18-22 September 2022.

IAC-22-C1,3,3,x73038

Neural Networks for Onboard Maneuver Design

Nathan Parrish Ré[a]*, Timothy M. Sullivan[b], Matthew D. Popplewell[c], Kirk S. Roerig[d], Clayton Michael[e], Tyler Hanf[f], Tyler Presser[g]

[a] *Advanced Space, LLC, 1400 W 122nd Ave, Westminster, CO 80234*, nathan.re@advancedspace.com
[b] *Advanced Space, LLC, 1400 W 122nd Ave, Westminster, CO 80234*
[c] *Advanced Space, LLC, 1400 W 122nd Ave, Westminster, CO 80234*, matthew.popplewell@advancedspace.com
[d] *Advanced Space, LLC, 1400 W 122nd Ave, Westminster, CO 80234*, kirk.roerig@advancedspace.com
[e] *Advanced Space, LLC, 1400 W 122nd Ave, Westminster, CO 80234*
[f] *Advanced Space, LLC, 1400 W 122nd Ave, Westminster, CO 80234*, tyler.hanf@advancedspace.com
[g] *Advanced Space, LLC, 1400 W 122nd Ave, Westminster, CO 80234*, tyler.presser@advancedspace.com
* Corresponding Author

Abstract

This paper presents a general neural network framework, Neural Networks for Electric Propulsion (NNEP), for spacecraft autonomous onboard maneuver design and its application to several astrodynamics regimes. A previous version of this work applied a neural network (NN) model to making a single low-thrust trajectory correction in cislunar space. This paper extends the prior work to allow any number of low-thrust trajectory corrections in cislunar space and implements it in a general framework. This framework also allows space missions to offload the computational "heavy lifting" to ground-based computers. Ground systems generate training data (consisting of tens of thousands of off-nominal maneuver designs) and train a series of NNs, where each NN is applicable to a predefined range of states and/or epochs. The framework's computational burden for the spacecraft is minimal and easily fits within most current flight computers. The NN framework is also implemented in prototype flight software as a coreFlight System (cFS) app with minimal external dependencies. Simulation results show accuracy and propellant use comparable to or better than the best human-in-the-loop ConOps. This framework is extended further to apply to interplanetary low-thrust trajectory correction, geostationary orbit (GEO) station keeping, and station keeping in Gateway's near-rectilinear halo orbit (NRHO).

Keywords: Neural Networks, Maneuver Design, Autonomous, Onboard

Acronyms/Abbreviations
AI = Artificial Intelligence
AoL = Argument of Latitude
BLT = Ballistic Lunar Transfer
CAPS = Cislunar Autonomous Positioning System
cFS = coreFlight System
CRTBP = Circular-Restricted Three Body Problem
ECEF = Earth-centered Earth-fixed
ECI = Earth-centered inertial
EP = Electric Propulsion
FSW = Flight Software
GEO = Geostationary Orbit
Isp = Specific Impulse
LM = Levenberg-Marquardt
MLP = Multilayer Perceptron
NIM = NRHO Insertion Maneuver
NN = Neural Network
NNEP = Neural Networks for Electric Propulsion
NRHO = Near Rectilinear Halo Orbit
OMM = Orbit Maintenance Maneuver
TCM = Trajectory Correction Maneuver
TPBVP = Two Point Boundary Value Problem
TIP = Trajectory Interface Point
VNB = Velocity-Normal-Binormal

1. Introduction & Background

1.1 Motivation

The state of the art for deep space mission navigation and operation is to have a dedicated team of 2-3 people for a single spacecraft, plus a crew to run the ground station. The deep space navigation process consists of spacecraft downlink, state estimation, trajectory re-optimization, hardware sequencing, and data uplink require two to three days. Ground station tasking constraints delay the process further, so spacecraft instructions are often days to weeks old by execution time. Aggressive ground scheduling reduces the delay to several days but further reduction requires automation. Two examples of this are the Dawn and BepiColombo missions. The BepiColombo mission (arriving at Mercury in 2025) uses ground timelines of 1 week [1]. The Dawn mission used timelines of 1-5 weeks during its the interplanetary transfer, 3 days during normal operations at Vesta, and 36 hours while passing through unstable resonances with Vesta's gravity field [2]. Regardless of the length of the timeline, the spacecraft always used thruster instructions based on out-of-date navigation results.

In addition to deep space missions, the space industry is trending towards a larger number of small spacecraft, enabling new capabilities at lower costs. This trend will also necessitate automating operations.

In response to the growing need for real-time and onboard maneuver planning for spacecraft, this paper presents a general framework using neural networks for the maneuver planning process. The framework leverages the powerful fundamental principles of optimal control and a rich field of recent advancements in the area of artificial intelligence (AI) to automate spacecraft maneuver correction, resulting in improved spacecraft maneuver accuracy, lowered operations complexity, and cost savings. Specifically, NNs are used as function approximators, learning the complex relationship between spacecraft state and the costates defining the optimal control to return to a reference trajectory.

Typically, when optimizing spacecraft maneuvers, a mission designer is tasked with solving a two-point boundary value problem (TPBVP). Solving the TPBVP is a numerically sensitive task that takes significant computational resources to solve. Solving such a problem also requires large numerical optimization software libraries and long runtimes. Both requirements make it unappealing to solve the TPBVP in flight software (FSW) running on a limited-performance flight computer. The framework presented in this paper solves these problems by "learning" the optimal solution as a function of the state which can then be uploaded and evaluated onboard the spacecraft. In this deployment, the computationally challenging training is done on the ground and the evaluation of the NN is done onboard the spacecraft. To demonstrate the viability of the framework for onboard spacecraft computations, it is implemented in prototype flight software as a cFS app.

This paper provides an overview of the neural network architecture as well as the framework built around it to develop maneuvers. Several applications of this framework are also analyzed with simulation results provided for each. Results of tests using the framework as a cFS app are also discussed.

1.2 Background: optimal control

Optimal control theory forms the basis of the neural network framework. An optimal control problem within the presented framework is defined as: Minimize the performance index J given as:

$$J = K(x_0, t_0, x_f, t_f) + \int_{t_0}^{t_f} L(x, u, \tau) d\tau \quad (1)$$

where K is the cost of the endpoints and L is the cost of the path. In addition, x is the state, u is the control, and t is time. The subscript 0 represents initial, and the subscript f represents final. The state vector is subject to differential constraints (the state dynamics) given as:

$$\dot{x}(t) = f(x(t), u(t), p, t) \quad (2)$$

where p is a vector of constant parameters. Terminal constraints are applied as:

$$g(x_0, t_0, x_f, t_f) = 0 \quad (3)$$

As trajectory optimization is nearly always an underdetermined problem, additional constraints must be introduced to find an optimal trajectory. The constraints are derived via the Hamiltonian as follows:

$$H = L(x, u, t) + \lambda \cdot \dot{x}(t) \quad (4)$$

where λ are the costates. The costate dynamics are then given by

$$\lambda = -\partial H / \partial x \quad (5)$$

The Hamiltonian Minimization Condition readily yields the optimal control policy

$$u^*(x, \lambda, t) = argmin_u H(x, \lambda, u, t) \quad (6)$$

For the minimum fuel transfer, the L term of the objective function is chosen as

$$L = u(t) \quad (7)$$

However, this results in a numerically sensitive problem. To make the indirect optimal control problem easier to solve, the L term of the objective function is chosen differently as:

$$L = u + \epsilon[u \ log \ u + (1-u) \ log \ (1-u)] \quad (8)$$

where ϵ is a homotopy parameter. When ϵ is ~1, the optimal control is very smooth. As ϵ is reduced to, say, 10^{-4}, the optimal control becomes nearly indistinguishable from the minimum fuel solution. This modification to the objective function was studied by Bai, Turner, and Junkins [3], then further refined by Bertrand and Epenoy [4], and studied in the context of NN optimal control by Parrish [5]. To keep the problem formulation simple and generic to different dynamical environments, additional path constraints can be introduced.

For low thrust trajectories, the NN is tasked with learning the relationship between a spacecraft's current state, which is perturbed from a nominal reference trajectory, and the costates, λ , for an optimal maneuver that returns to the reference trajectory.

1.3 Background: neural networks

Neural networks are a mathematical tool capable of approximating arbitrarily complex functional

relationships. NNs are inspired by biological brains and consist of a network of "neurons", where each neuron performs a basic mathematical operator on its inputs, passing its output to the next neuron.

The NNs used in this paper are simple feedforward networks consisting of an input layer, a series of hidden layers, and an output layer. This network formulation is also known as a multilayer perceptron (MLP). A generic feedforward neural network with a single hidden layer is shown below in Figure 1.

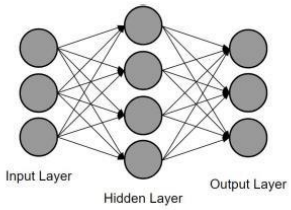

Figure 1. A simple neural network with one hidden layer.

One of the advantages of NNs is their ability to approximate arbitrarily complex functions given sufficient network sizes and training samples. Additionally, while the training process takes significant time, the evaluation of the network given a set of inputs is extremely fast and requires few computational resources, making NNs suitable for flight-approved hardware.

2. Methods

2.1 Training data generation

Much of the complexity of the framework is in the generation of quality training data. This section describes how those simulated data are created for each type of application.

2.1.1 Low thrust transfer general approach to training data generation

A reference trajectory must be available for the spacecraft to follow. The reference trajectory can be generated by any means, as the training sample generation only requires the position and velocity over time. In this work, we use a proprietary tool to build an optimal trajectory with the following high-level steps:

- Direct multiple shooting in circular-restricted three body problem (CRTBP) dynamics, with mesh refinement to add more nodes near lunar close approaches.
- Indirect multiple shooting in the CRTBP with the smoothed minimum fuel objective function.
- Convert solution in CRTBP to an ephemeris model.
- Direct multiple shooting in the ephemeris model.
- Indirect multiple shooting in the ephemeris model with the smoothed minimum fuel objective function.

Training data are generated by first choosing a time t^* at random from the time span of the reference trajectory. At time t^* the position and velocity are perturbed by some $\delta\vec{r}$ and $\delta\vec{v}$, drawn from random uniform distributions with lower and upper bounds $U(0, \delta r_{\max})$ and $U(0, \delta v_{\max})$ .The choice of $\delta r_{\max}$ and $\delta v_{\max}$ define the a "training tube" size illustrated in Figure 2.

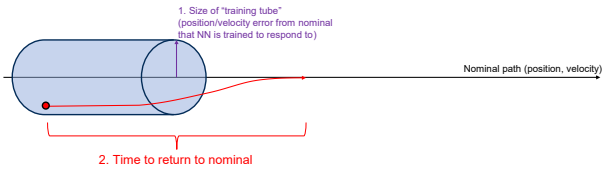

Figure 2. Illustration of the "training tube" around the reference path, used for low-thrust trajectory corrections.

With the training tube defined, the final step is to solve the fixed time TPBVP for the transfer back to the nominal trajectory at time $t^* + \Delta t$. The time interval Δt is fixed and chosen for the problem, typically on the order of a few days or tens of days. The use of a receding horizon keeps the spacecraft close to the nominal path at all times, thus keeping the spacecraft within the training tube.

A core element of this approach is the use of checkpoints at which the training samples are required to return to the reference path. This is illustrated in Figure 3. We found that adding ~2-5 checkpoints with fixed final times helped guide simulated spacecraft to stay on a fuel-optimal trajectory. Checkpoints are added immediately prior to sensitive events such as a flyby (when small state errors can be hard to recover from) or the start of a long thrust arc (when the control authority to respond to anomalies is more limited).

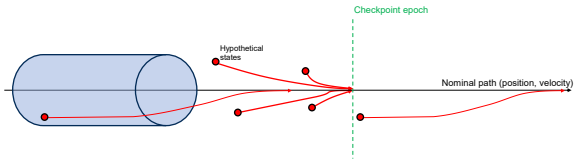

Figure 3. Illustration of checkpoints with the "training tube". All training samples leading up to a checkpoint epoch are required to return to the nominal at the checkpoint, rather than at a receding time horizon.

Training samples can now be generated with indirect single shooting. Partial derivatives for the trajectory solver are computed via automatic differentiation, which allows the trajectory to be solved to numerical precision. An example of the training tube is given in Figure 4.

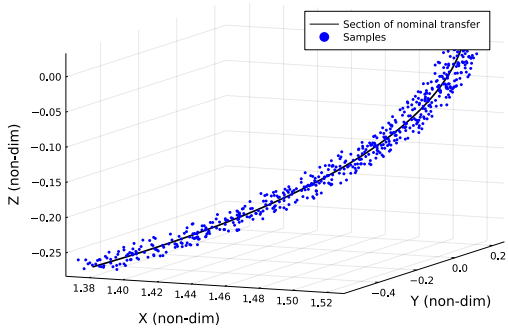

Figure 4. Example training tube around an arbitrary section of a nominal transfer.

2.1.2 Cislunar low thrust transfer training data generation

The Earth-Moon system was chosen due to its sensitive dynamics which makes learning accurate corrections more difficult. Algorithms developed for this dynamical environment will necessarily work for simpler dynamics. Initial tests were conducted using the CRTBP and then extended to a point-mass ephemeris model using the JPL DE430 ephemeris.

The test transfer is from an Earth-Moon NRHO (Gateway's planned orbit) to a larger Earth-Moon L2 halo orbit. The nominal transfer's time of flight is 24 days. The spacecraft has an initial mass of 1,000 kg and a maximum thrust of 300 mN with an Isp of 2,000 sec. This hypothetical spacecraft propulsion system is equivalent to the high end of currently-feasible thrust-to-weight ratios of electric propulsion (EP) spacecraft. The maximum thrust is limited to 240 mN for the nominal transfer to build in margin for recovery from errors. The nominal transfer requires 5.4 kg of propellant. Spacecraft thrusting is not allowed within a radius of 20,000 km from the Moon to avoid the additional operational complexity of a powered lunar flyby. The trajectory is defined in the Moon-centered J2000 inertial frame. The example NRHO to L_2 transfer is shown in Figure 5. In addition, the nominal transfer's thrust profile is computed as in Figure 6.

The nominal transfer and all NN training trajectories are modeled with the point mass gravity of Earth, Moon, and Sun from the DE430 ephemeris. The spacecraft specific impulse (Isp) and maximum thrust are assumed constant throughout the nominal transfer.

The training tube for this transfer is defined as $\delta r = 500\,km$ and $\delta v = 5\,m/s$ relative to the nominal trajectory. Training samples return to the nominal trajectory after 4 days or at the next checkpoint epoch, whichever is earlier.

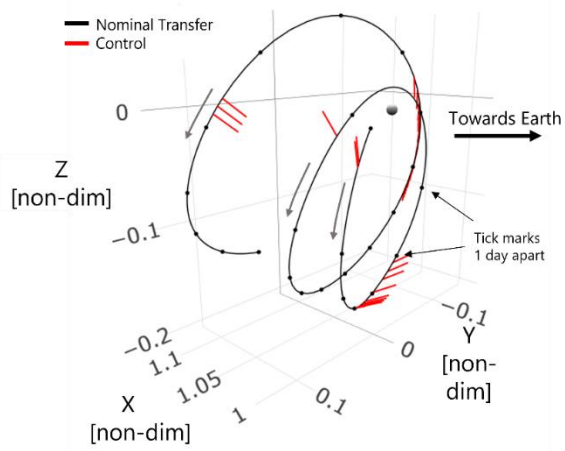

Figure 5. NRHO to L_2 nominal low thrust transfer, viewed in the Earth-Moon rotating frame.

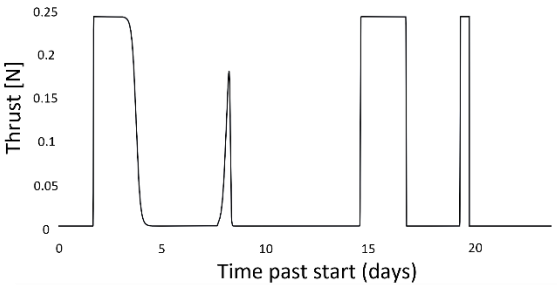

Figure 6. Nominal thrust profile vs. time for NRHO to L_2transfer.

2.1.3 Interplanetary low thrust transfer training data generation

The same method for training sample generation is also applied to an interplanetary, heliocentric transfer. For this transfer, a solar electric propulsion model is used to provide realistic spacecraft control authority. The nominal transfer consists of an Earth launch, Mars flyby, and finally Mars rendezvous, of which only the first leg (Earth to Mars flyby) will be used to train the neural network. This leg was designed with a duty cycle of 80%, has a time-of-flight of 290 days, and has a total mass drop of approximately 64.5 kg. The transfer uses near bang-bang control to model real-world operations. The force model used for the nominal trajectory and all NN trajectories includes the point mass gravities of Earth, Jupiter, Mars, and the Sun from the DE430 ephemeris.

Figure 7 illustrates a heliocentric inertial view of the nominal interplanetary trajectory on which the framework was trained.

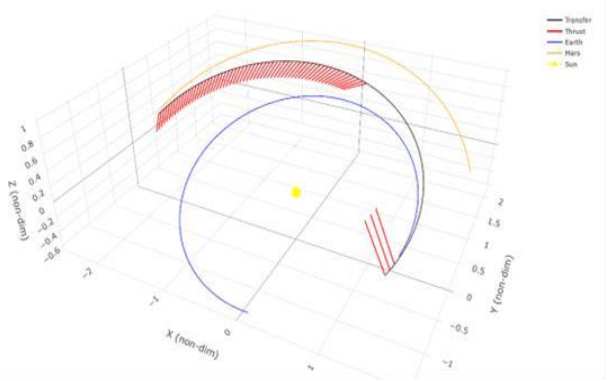

Figure 7. Sun-centered, inertial view of nominal transfer.

The available thrust is a function of a realistic power model. The power available to the electric propulsion engine is inversely proportional to the square of the distance from the Sun, with a maximum of 3 kW at 1 AU. The engine model has a fixed Isp and jet efficiency (1500 s and 50%, respectively) and includes losses for regulator efficiency, off-pointing, solar cell degradation, and bus power requirements. The thrust curves for the interplanetary transfer are given in Figure 8.

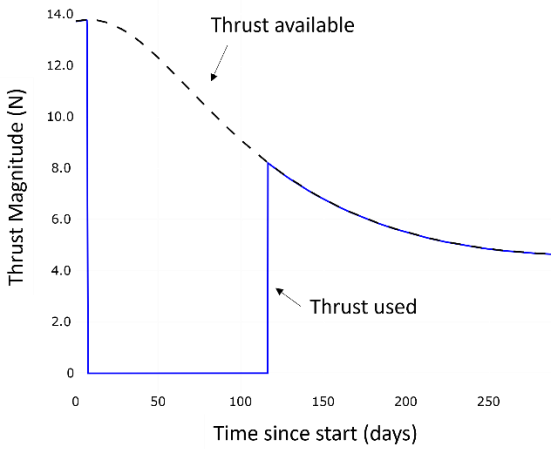

Figure 8. Available and nominal thrust used by interplanetary transfer.

Training samples for the NN are continuously generated in a defined region around the nominal transfer according to the "training tube" described previously. For this problem, we generated 150,000 samples between the start and end of the nominal transfer in a uniform distribution up to 5,000 km and 50 m/s in position and velocity magnitude, respectively. The single shooting algorithm then solves for the minimum fuel, smoothly-varying-thrust costate solution that allows each sample to rendezvous with the nominal transfer either 30 days later or at the end of the transfer, whichever is sooner, from the sample epoch.

2.1.4 GEO station keeping training data generation

The GEO station keeping problem is based on the idea of keeping a spacecraft within an operational slot defined by latitude and longitude bounds. Perturbations from solar and lunar point mass gravities, solar radiation pressure, and the non-uniformity of Earth's gravitational field cause a GEO spacecraft to drift in the longitudinal and latitudinal directions from its nominal nadir location. The effects of these perturbations are illustrated in Figure 9.

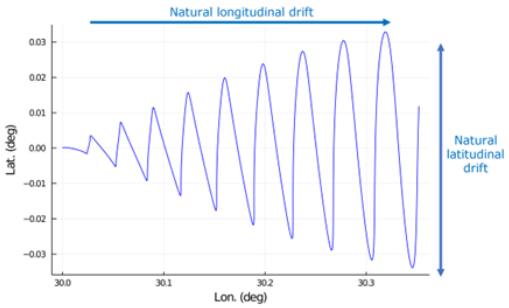

Figure 9. Natural longitudinal and latitudinal drift due to perturbations.

Autonomous maneuver planning for GEO spacecraft has been studied extensively in literature [6], [7]. However, existing strategies rely on linearized dynamical models or approximate optimization methods. Neural Networks provide a means for on-board maneuver correction without over-simplifying the problem.

GEO station keeping using chemical propulsion consists of two separate maneuver types: an east-west maneuver that corrects longitudinal drift and a north-south maneuver that corrects latitudinal drift. The maneuvers are described in the Velocity-Normal-Binormal (VNB) local frame. East-west maneuvers are performed solely in the velocity direction which increases or decreases the orbit semi-major axis, consequently decreasing or increasing the orbit angular speed and causing the spacecraft to drift westward or eastward, respectively. Eventually, the influence of the orbit perturbations creates a turnaround point, at which the spacecraft begins drifting in the opposite direction.

For a spacecraft that experiences a natural eastward drift, an optimal east-west maneuver places the turnaround point at the westerly boundary of the operating slot in order to maximize the time between maneuvers. North-south maneuvers return the inclination of the orbit to zero. Maneuvers that only change inclination require a component in both the velocity and orbit normal directions. However, for this application, the north-south maneuvers consist of a burn solely in the orbit normal direction. Performing such a maneuver causes a deviation from the nominal orbit semi-major axis and, consequently, an east-west maneuver frequently immediately follows a north-south burn. Experience has

shown that a NN can more easily learn the north-south maneuvers when the problem is structured this way. An example EW maneuver is shown in Figure 10.

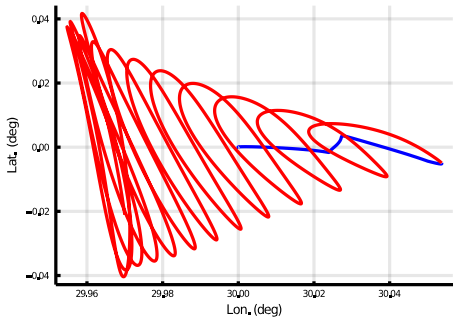

Figure 10. An example east-west maneuver – the spacecraft (originally blue) approaches the easterly boundary of the operational box (30.05°) causing a maneuver to be performed. The spacecraft (now red) is propagated to the turnaround point near the westerly boundary of the box (29.95°).

For this problem, the spacecraft is initialized in GEO with a nadir point of 30° East and 0° North and an epoch of 29373 UTC MJD. The operational slot is defined with longitudinal and latitudinal bounds of ±0.1° to model realistic application. To generate samples, a targeting scheme in GMAT delivers the optimal station keeping maneuvers for the nominal GEO spacecraft over ten years. Additional samples are created by perturbing the initial state off the nominal and again finding the optimal station keeping history for ten years. The perturbed initial states consist of combinations of initial epoch, latitude, and longitude values in the ranges of [29373, 29391 UTC MJD], [-0.3, 0.3°], and [29.96,30.02°] respectively.

A separate NN is trained for each maneuver type. The input layer of the NN takes 14 quantities: the spacecraft's latitude, longitude, Cartesian velocity components in an Earth-centered Earth-fixed (ECEF) frame, longitude rate, orbit eccentricity, and the Cartesian position components of the Moon and Sun in an Earth centered inertial (ECI) J2000 frame. The longitude rate is found by taking the difference of the Earth's rotation rate and the spacecraft's mean motion. The output layer of the NN is a single quantity: the burn magnitude of the appropriate maneuver. A NN with three hidden layers of 30 neurons each and these input and output layers (total of 3,271 learned NN weights) is capable of learning both maneuver types well.

To evaluate the performance of the NN, it is desirable to propagate a GEO spacecraft in GMAT using a high-fidelity environment. However, the NN is built and trained using the Julia language and GMAT lacks a built-in interface with Julia. In order to pass the NN evaluations to GMAT, a TCP socket is used to send information between a Julia script, which evaluates the NN, and a Python script, which interfaces with the GMAT API.

A GEO spacecraft, with an initial state at 29373 UTC MJD and a sub-satellite point of 30° East and 0° North, is propagated using the above simulation scheme for 10 years. The spacecraft is propagated in GMAT until an EW or NS station keeping maneuver is required. A Python script collects the end state using the GMAT API and sends the state to Julia over the TCP socket. A Julia script then determines which maneuver type is required, evaluates the correspond NN, and sends the burn magnitude over the socket to Python. Python then reinitializes the GMAT simulation with the previous end state and corresponding burn. GMAT performs the NN burn before again propagating until the next maneuver. To make the simulation more realistic, navigation and thruster errors are simulated. The navigation error is modeled as Gaussian noise $\mathcal{N}(0, \sigma 2)$ where σ is the standard deviation according to the values given in Table 1.

Table 1. GEO simulation setup.

State Component	σ
Latitude (deg)	5e-5
Longitude (deg)	5e-5
Cartesian velocity components (cm/s)	1
Eccentricity	1e-6
Orbit period (seconds)	0.5

The thruster error is modeled at 2% for this simulation. The Moon and Sun Cartesian positions in the ECI frame are obtained through JPL's SPICE toolkit and thus no error is applied to these values.

2.1.5 NRHO station keeping training data generation

Within the Cislunar system, the station keeping problem was also analyzed using the framework. The authors specifically chose to test the framework to simulate station keeping within an NRHO due to its relevance for the upcoming NASA Lunar Gateway. NASA's upcoming Lunar Gateway is planned to operate in a southern L_2 NRHO. The orbit has close approaches to the lunar surface allowing low-cost access to the lunar surface and consistent line-of-sight with Earth ground stations. The orbit is phased such that it avoids all total Earth eclipses [8]. Advanced Space's CAPSTONE mission launched on June 28, 2022 and will demonstrate operations within the same NRHO that will be used by the Gateway.

Existing station keeping strategies for the designated NRHO target a velocity vector component in the Earth-Moon rotating frame and a time of perilune passage several revolutions downstream of a given state.

Targeting the velocity component in the $\hat{x}$ direction (v_x) at perilune passage 6.5 revolutions downstream is currently favored as it delivers a minimum ΔV and maintains the NRHO near its reference. The planned station keeping strategy for CAPSTONE follows the strategy planned for Gateway: perform an orbital maintenance maneuver (OMM) up to once every revolution that targets both v_x and T_p at an osculating true anomaly of 200° [9]. The v_x used for targeting is the velocity in the $\hat{x}$ direction of the Earth-Moon rotating frame at perilune. This v_x and the time of perilune passage (T_p) are compared to those of the reference orbit. When T_p is within a given threshold, the maneuver only targets v_x.

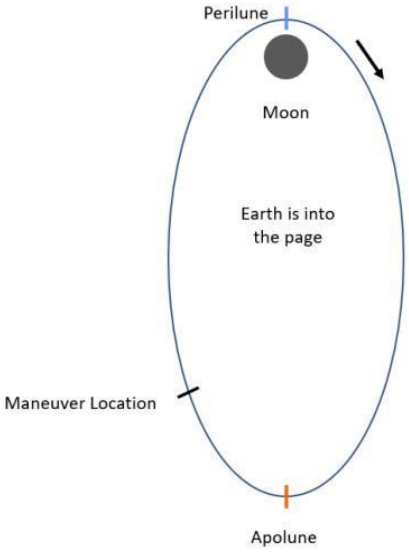

Figure 11. Maneuver location on NRHO viewed from the Y-Z plane of the Earth-Moon rotating frame.

Training samples for the NN are generated by perturbing the six-dimensional state of the orbit at each OMM epoch for the full reference period (15 years, up to 2035). At each OMM epoch, the reference state is determined from ephemeris and 500 samples are generated by perturbing the position and velocity vectors such that they form a uniform spherical distribution with 3σ of 3,000 km and 30 cm/s respectively. For each sample, the corresponding station keeping maneuver that targets VX and TP 6.5 revolutions downstream is designed using the Jet Propulsion Laboratory's Monte library. The OMM is designed using a realistic dynamical model with the forces modeled in Table 2 through Table 4.

Table 2. Point masses used for NRHO dynamics.

Point Masses
Sun
Mercury
Venus
Mars Barycenter
Jupiter Barycenter
Saturn Barycenter
Neptune Barycenter
Uranus Barycenter
Pluto Barycenter

Table 3. Spherical harmonics used for NRHO dynamics.

Body	Dataset	Order
Earth	GGM03C	16
Moon	GL900D	16

Table 4. Solar Radiation Pressure parameters for NRHO dynamics.

Parameter	Value
Flat Area (m^2)	0.6
Mass (kg)	25
C_r	1.5

A single feedforward neural network is trained for all OMM epochs. The NN consists of two hidden layers each with 40 neurons. The input layer has a size of 42 consisting of:

- the six-dimensional perturbed sample state, six-dimensional reference state,
- six-dimensional error vector between the perturbed and reference states,
- three-dimensional relative spacecraft-Sun position vector in the synodic frame,
- three-dimensional relative spacecraft Earth position vector in the synodic frame,
- three-dimensional vector of Euler angles describing the Moon orientation in inertial space,
- six-dimensional perturbed state 6.5 revolutions downstream,
- six-dimensional reference state 6.5 revolutions downstream,
- the v_x difference between the perturbed state and reference 6.5 revolutions downstream, and
- the T_p difference between the perturbed state and reference 6.5 revolutions downstream.

The output layer has a size of three consisting of the impulsive three-dimensional maneuver vector in the Earth-Moon rotating frame. In total, there are 3,483 learned NN weights.

2.1.6 TCM training data generation

As mentioned in section 2.1.5, Advanced Space's CAPSTONE mission will demonstrate operations within the same NRHO that will be used by the Gateway. In order to reach and eventually insert into the target NRHO, CAPSTONE follows a ballistic lunar trajectory (BLT), a fuel-efficient path that leverages the Sun's gravity to increase radius of periapsis and inclination. Along the BLT, several trajectory correction maneuvers (TCMs) occur to clean up launch vehicle insertion errors, maneuver errors, and navigation errors. Nominally, five TCMs will occur along the BLT but the exact number can increase to eight depending on vehicle parameters, insertion errors, maneuver errors, and navigation errors.

The TCMs are designed using a three-burn optimization process [10]. When designing any given TCM *n*, the backpropagated state from the NRHO at the next TCM *n+1* epoch is targeted and the total ΔV of TCM *n*, TCM *n+1*, and the NRHO Insertion Maneuver (NIM) is minimized. For example, when designing TCM1, the total ΔV of TCM1, TCM2, and NIM is minimized while the targeted state is that of the reference trajectory at the time of TCM2. Nominally, TCM1 through TCM5 are designed in this manner with TCM5 also designing an optional TCM6. Additionally, if TCM1, which cleans up initial launch errors, is above a threshold defined by the spacecraft parameters then optional maneuvers TCM1b and TCM1c are also designed. The approximate locations of the TCMs for a representative BLT are given in Figure 12.

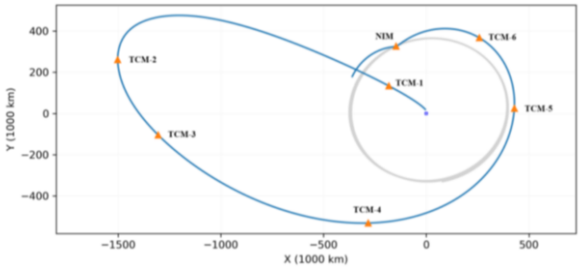

Figure 12. Approximate locations of nominal TCMs for a given BLT [10].

TCM training data is generated via Monte Carlo simulation of the operational maneuver design process. Realistic state errors are sampled relative to the reference trajectory at each pre-planned TCM epoch, then the 3-burn optimization described above is carried out to generate the ΔV vector corresponding to the state error.

For this test case, the training dataset only includes the nominal TCM1 through TCM5. Any TCM1 maneuver that is over 20 m/s, and would normally require a secondary TCM1 burn, is filtered from the dataset before training. A BLT reference trajectory corresponding to a previously designed Trajectory Interface Point (TIP) of June 2, 2022, and the corresponding TCM maneuver schedule is used to determine the reference TCM states. Sample states are generated by uniformly perturbing each TCM reference state in position and velocity such that they form a spherical distribution. At each sample state the corresponding TCM is designed using the aforementioned process. The TCM design, and later simulation, is performed in the realistic dynamical model identical to the NRHO dynamics outlined in Table 2 through Table 4.

A separate feedforward NN is used to learn each of the five nominal TCMs. The NN consists of three hidden layers with 30 neurons each. The input layer is of dimension 30 consisting of the six-dimensional perturbed sample state, six-dimensional reference state, six-dimensional error vector between the perturbed and reference states, six-dimensional relative spacecraft-Sun state vector in inertial space, and six-dimensional relative spacecraft-Earth state vector in inertial space. The output layer is of size four consisting of the TCM burn duration and three-dimensional unit direction vector. In total, there are 2,914 learned NN weights.

2.2 Machine learning framework

Based on experience from Parrish [5], we chose to use the Levenberg-Marquardt (LM) optimization algorithm. NNs are trained using a custom version of the Levenberg-Marquardt (LM) algorithm. At each iteration, the LM algorithm computes the Jacobian of all training samples with respect to all trainable parameters and uses this to adjust the trainable parameters to minimize the sum of squared errors. When the objective is to minimize the sum of squared errors, the Jacobian can be used to accurately approximate the Hessian as well, leading to second-order convergence on the optimal NN weights. The LM algorithm is suited for regression problems with NNs containing up to tens of thousands of weights. It is not used for classification problems such as natural language processing or image recognition because those require NNs with millions or even billions of weights. Use of the LM training algorithm on such a large NN would require computing an excessive number of partial derivatives and is thus computationally intractable. For small NNs, such as those commonly used in regression problems, the LM algorithm has been found to converge orders of magnitude faster than methods that only compute a vector gradient.

The LM algorithm makes updates to the weights as follows:

$$W' = W - (J^T J + \mu I)^{-1}(J^T \vec{e}) \quad (9)$$

where W is the vector of model weights, J is the Jacobian of model outputs with respect to model weights, I is an appropriately-sized identity matrix, μ is an adjustment factor to smoothly vary between 1st and 2nd order convergence, and $\vec{e}$ is the vector of residuals (the

difference between the model output and the desired model output).

At the start of this work, the only performant, publicly-available LM implementation was in MATLAB, and even that implementation was not capable of GPU acceleration. We traded all the available implementations and ultimately decided to write our own in the Julia language [11] with GPU acceleration via the CUDA.jl package [12]. Our implementation uses analytically-defined partial derivatives that we developed and implemented with great regard to speed. The Jacobian matrix is strategically broken into sections, and each section is computed by a separate GPU thread. The performance bottleneck is the low-level cuBLAS matrix pseudo-inverse in equation 9.

Our implementation is based on the work by Tomislav et al [13], particularly with regard to the automatic weight decay variation. Testing on a variety of problems has found this implementation to converge more quickly and robustly than naïve choice of the μ parameter.

2.3 ConOps for onboard implementation

The proposed framework would be implemented on board a spacecraft as follows. First, ground operators design a nominal transfer from the current state to some target state, taking into consideration all necessary constraints. Then, the same ground software is used to solve tens of thousands of similar, perturbed transfers.

The solution of each perturbed transfer results in a pair of input-output vectors: given the current state vector (and problem-specific additional context), return the primer vector (for continuous-thrust) or the ΔV vector (for impulsive maneuvers) corresponding to the optimal path to rendezvous with the target. The ground software then trains a NN to approximate the relationship between the input and the output.

Since each of the perturbed trajectories is short and has a good guess from the nominal path, the perturbed optimal trajectories can be generated quickly. Proprietary trajectory optimization tools at Advanced Space can generate these training samples in a few hours on a modern workstation computer. If needed, the task can be parallelized to hundreds or even thousands of CPUs in a distributed computing environment to generate sufficient training samples within minutes.

Once training samples are generated, the NN model is trained. The model is saved as a binary file dictating the shape of the model and the model weights, and the binary file is either loaded on the spacecraft prior to launch or uplinked to the spacecraft during flight.

Trajectory corrections are made continuously over the course of an orbit transfer. The frequency of corrections is tunable for the mission, with tests here using a 10–60-minute update cadence.

The spacecraft is assumed to perform onboard orbit determination which returns a state error signal. Example onboard orbit determination technologies include the Cislunar Autonomous Positioning System (CAPS) which is under development at Advanced Space [10] , GNSS, optical navigation, one-way ranging with an atomic clock, and other technologies in development. The spacecraft passes the best estimate of the current state into the NN model, which returns the costate vector or ΔV vector. The spacecraft numerically integrates the state forward in time to the next tick, with control held inertially fixed. Other elements of the FSW turn the control vector into thruster instructions which are executed to fly the trajectory. A schematic of the framework's concept of operations is shown below in Figure 13.

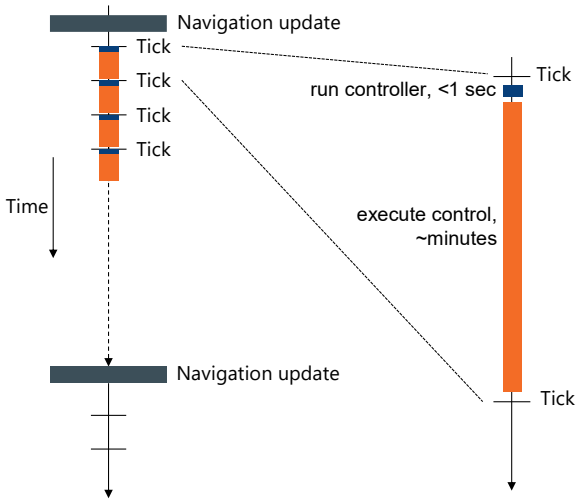

Figure 13. Schematic of onboard ConOps.

3. Results: Low Thrust Control

To evaluate the performance of the NNs, a spacecraft is flown along the reference trajectory using the NNs to deliver trajectory corrections. The simulation propagates two spacecraft concurrently, a "truth" and a "predicted" spacecraft. The predicted spacecraft's state is periodically updated to that of the truth, plus some navigation noise, according to a navigation frequency parameter. At a defined NN evaluation frequency, the predicted spacecraft's state is used to evaluate the appropriate NN and deliver the desired control. To translate the costate outputs of the NNs to thrust commands, the NN is evaluated frequently, and the spacecraft thruster is only fired when the magnitude of the costate control law is above a defined threshold parameter. The truth spacecraft executes the desired control commands from the predicted spacecraft plus some thruster error.

3.1 Earth-Moon 3-body transfer

The results from Monte Carlo simulation of a NRHO to L_2 transfer with the proposed framework and simulated ground control are provided in Table 5, for a range of navigation-update cycle speeds. Historically, achieving a 2-day navigation and control update loop with human-in-the-loop control has required 24-hr coverage from operations teams and is only practical for short periods of critical operations. However, the speed of the control loop in space is limited only by the onboard navigation requirements. Once a state estimate is available for NNEP, it can update the thruster commands immediately.

Table 5. Comparison of neural net framework and simulated ground control.

Simulation type (nav update cycle)	Final Error (km)	Average Propellant (kg)
NNEP 2-day	39 ±15	6.03±0.24
NNEP 4-day	54±23	6.16±0.33
NNEP 6-day	129±84	6.22±0.32
Ground 2-day	22±11	6.1±1.1
Ground 4-day	75±63	6.8±1.3

3.2 Interplanetary transfer

Table 6 shows the results for a 200-trial Monte Carlo simulation of the NNEP and ground control strategies for different navigation update frequencies. NNEP outperforms the ground control in terms of final error from the reference trajectory but uses slightly more fuel in the process. The NNEP software makes up for larger navigation errors with a nearly instantaneous navigation solution to control update cadence.

Table 6. Neural net framework and human control comparison (N = 200). Data provided as mean ± standard deviation.

Simulation type (nav update cycle)	Final Error (km)	Propellant use (kg)
NNEP 8-day	167±19	64.46±0.09
NNEP 16-day	205±60	64.50±0.10
NNEP 28-day	184±62	64.49±0.10
Ground 8-day	364±252	64.22±0.03
Ground 16-day	525±321	64.25±0.06
Ground 28-day	879±516	64.28±0.13

4. Results: Impulsive Control

4.1 GEO station keeping

Results for the GEO station keeping trial show that the NN framework can deliver necessary station keeping maneuvers such that the spacecraft maintains its operational slot for the entire 10-year simulation time. The total ΔV for all maneuvers over 10 years using NNEP is 521.365 m/s. In comparison, a ground simulation, in which GMAT propagates the spacecraft and calculates optimal station keeping burns, results in a total ΔV of 520.481 m/s over 10 years. The NN delivers near optimal maneuvers instantaneously without the need for a human operator or optimization software.

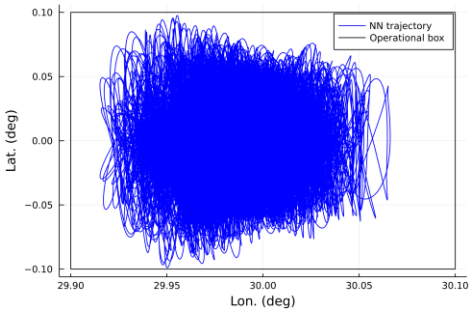

Figure 14. Motion of a GEO spacecraft in the ECEF frame propagated for 10 years using the NN for station keeping predictions.

4.2 NRHO station keeping

To evaluate the performance of the NN, a spacecraft in the defined NRHO is propagated for the full reference period (15 years) using the NN to deliver OMMs every revolution. The spacecraft is propagated in JPL's Monte Python package. Since the NN is constructed and trained in the Julia language but the simulation occurs in Python, a TCP socket is used to pass information between a Julia script running concurrently with the Python simulation. Two spacecraft are simulated, a "NN" spacecraft that evaluates the trained NN for OMMs and a "truth" spacecraft that uses SNOPT and the aforementioned station keeping strategy to deliver OMMs. Both are initialized with the same state which is randomly generated and slightly off-reference. To make the simulation more realistic, navigation and thruster errors are simulated. The navigation noise is modeled as a uniform spherical perturbation to the six-dimensional spacecraft state with 3σ of 1 km and 1 cm/s for position and velocity respectively. The thruster error is modeled as a uniform spherical distribution in direction with a random magnitude according to a uniform distribution with an upper bound of 2% of the maneuver magnitude.

Results in Table 7 show that the NN can deliver sufficient OMMs such that the NN spacecraft remains in the vicinity of the reference NRHO for the entire 15-year reference period. A Monte Carlo analysis was performed by running the above simulation 100 times and collecting the results.

The NN delivers OMMs to remain within the vicinity of the NRHO over 15 years without the need for a ground team to design maneuvers. The NN, while using slightly more total DV over the simulation time, outperforms the ground control in terms of error from the reference orbit. It should be noted that the amount of difference in ΔV

between the NN control and ground control is negligible compared to effect of momentum desaturation maneuvers, which were not modeled here.

Table 7. NN and ground truth control comparison (N = 100). Data provided as mean ± standard deviation.

Performance Metric	NN Control	Ground Control
Annual ΔV (m/s)	0.31 ± 0.04	0.25 ± 0.02
Maximum position deviation (km)	554 ± 33	780 ± 12

4.3 TCM's for CAPSTONE

To evaluate the performance of the NN, a spacecraft is flown on the reference BLT using the NNs to deliver each nominal TCM. The spacecraft is propagated in JPL's Monte Python package from TIP to TCM6. Since the NN is constructed and trained in the Julia language but the simulation occurs in Python, a TCP socket is used to pass information between a Julia script running concurrently with the Python simulation. To make the simulation more realistic, navigation error is simulated. The navigation noise is modeled as a uniform spherical perturbation to the six-dimensional spacecraft state with 3σ of 5 km and 5 cm/s for position and velocity respectively.

Table 8 shows the results of a 200 trial Monte Carlo simulation of the CAPSTONE BLT with NNs delivering the necessary TCMs. The NNs can provide sufficient maneuvers such that the spacecraft remains close to the reference trajectory autonomously.

Table 8. NN control Monte Carlo results (N = 200). Data provided as mean ± standard deviation.

Performance Metric	NN Control
Total ΔV (m/s)	83 ± 5
Final position error (km)	372 ± 20

5. Results: FSW implementation

The on-board neural network inferencing functionality is designed to be platform independent. Thus, NASA's cFS framework is used to host the neural network software. To maintain low-compute requirements, trained models are serialized and then uplinked to the on-board system. As model integrity and model relevance are key factors to nominal operations, the FSW is designed to allow for model uplinks to occur at any time to replace the current model. To handle off-nominal exceptions during flight, the FSW is implemented with a standby mode as shown in Figure 15, allowing operators to provide recovery commands as needed without model evaluations.

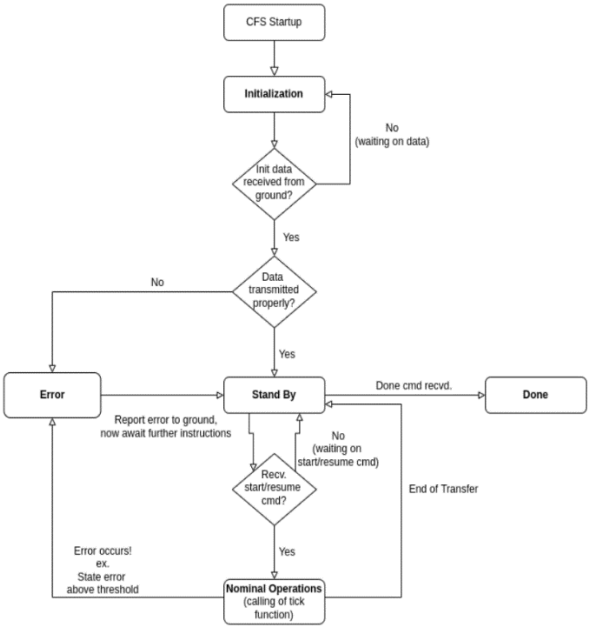

Figure 15. NNEP cFS App Order of Operations

The FSW implementation is designed to be minimally intrusive for any spacecraft. We developed a custom inference engine that is highly memory-efficient. Profiling was carried out on a NN model representative of a low-thrust trajectory corrector. Profiling of model loading, de-serializing, and inference showed a peak memory use of 86.5 kB. Software dependencies are also minimal: the entire cFS app is dependent only on NASA's CSPICE.

The FSW has been simulated using a mock ground-station to uplink models, send commands, and to receive telemetry. Tests were conducted with nominal and off-nominal initial conditions, as well as with varying simulation durations. Furthermore, standby mode and in-flight FSW reinitialization were also tested to ensure correct recovery behavior. The simulation results were compared with the non-FSW implementation simulation results and proved to be equivalent. We hope to be able to share more details of the FSW implementation in a future publication.

6. Conclusion

This paper presents a framework for autonomous and onboard maneuver planning in both low thrust and impulsive thrust spacecraft built upon NNs. To illustrate the NN's ability to generate optimal control response in different dynamical regimes, simulations were conducted with both two- and three- body trajectories. To support the framework's usability in flight, a concept of operations is designed to utilize the framework with data that would traditionally be available to the spacecraft and to ground operators.

The next contribution is the sample generation method for the NN. To generate training samples for the NN, the sampling tube method is proposed where a nominal trajectory is designed and thousands of perturbed states are used to compute optimal return maneuvers to the reference trajectory. With the neural networks trained for different tasks, simulation results are then presented for each maneuver design task in each dynamical environment.

Finally, this paper also briefly summarizes a process by which the framework can be implemented onboard a spacecraft as a cFS application. Altogether, the proposed framework and findings from subsequent analysis aim to provide a highly generalizable architecture based on neural networks for automated and onboard maneuver planning in multiple dynamical environments.

Acknowledgements

This research was sponsored by the NASA SBIR program, contract number 80NSSC20C0139.

References

[1] F. Castellini, G. Bellei and F. Budnik, "BepiColombo Orbit Determination Activities During Electric Propulsion Arcs," in *AIAA SciTech Forum*, 2020.

[2] D. W. Parcher and G. J. Whiffen, "Dawn Statistical Maneuver Design for Vesta Operations," *Advances in the Astronautical Sciences,* vol. 140, no. 818, pp. 1159-1176, 2011.

[3] X. Bai, J. D. Turner and J. L. Junkins, "Bang-bang Control Design by Combing Pseudospectral Method with a novel Homotopy Algorithm," in *AIAA, GNC-36: Hypersonic Vehicles and Spacecraft Control I*, Chicago, 2009.

[4] R. M. Byers, S. R. Vadali and J. L. Junkins, "Near-minimum time, closed-loop slewing of flexible spacecraft," *Journal of Guidance, Control, and Dynamics,* vol. 13, no. 1, pp. 57-65, 1990.

[5] N. Parrish., *Low Thrust Trajectory Optimization in Cislunar and Translunar Space,* dissertation: University of Colorado Boulder, 2018.

[6] F. Brujin, S. Theil, D. Choukroun and E. Gill, "Collocation of geostationary satellites using convex optimization," *Journal of Guidance, Control, and Dynamics,* vol. 39, 2016.

[7] D. Losa, M. Lovera, J. Mrmorat, T. Dargent and J. Amalric, "Station keeping of geostationary satellites with on-off electric thrsuters," in *IEEE Conference on Control Applications* , Germany, 2006.

[8] M. Thompson, M. Bollinger, N. Ré and D. Davis, "An Analysis of Downstream Uncertainty in NRHO Stationkeeping Strategies," in *AIAA SciTech Forum*, National Harbor, 2022.

[9] D. C. Davis, F. S. Khoury, K. C. Howell and D. J. Sweeney, "Phase Control and Eclipse Avoidance in Near Rectilinear Halo Orbits," in *43rd AAS Guidance, Navigation, and Control Conference*, Breckenridge, 2020.

[10] E. W. Kayser, J. S. Parker, M. Bollinger, T. Gardner and B. Cheetham, "The Cislunar Autonomous Positioning System Technology Operations Navigation Experiment," in *Ascend 2020*, Virtual Event, 2020.

[11] J. Bezanson, A. Edelman, S. Karpinski and V. B. Shah, "Julia: A Fresh Approach to Numerical Computing," *SIAM Review,* no. 59, pp. 65-98, 2017.

[12] T. Besard, C. Foket and B. De Sutter, "Effective Extensible Programming: Unleashing Julia on GPUs," *IEEE Transactions on Parallel and Distributed Systems,* no. 1045-9219, 2018.

[13] B. Tomislav, D. Majetic and D. Brezak, "GPU Implementation of the Feedforward Neural Network with Modified Levenberg-Marquardt Algorithm," in *2014 International Joint Conference on Neural Networks (IJCNN)*, Beijing, 2014.

[14] E. W. Kayser, J. S. Parker, M. Bolliger, T. Gardner and B. Cheetham, "The Cislunar Autonomous Positioning System Technology Operations and Navigation Experiment," in *Ascend 2020*, Virtual Event, 2020.