Simulated Onboard Autonomous Maneuver Design for Lunar IceCube and Lunar Reconnaissance Orbiter

Matthew D. Popplewell[1] and Nathan Parrish Ré[2]
Advanced Space, LLC, Westminster, CO 80234

Tyler Hanf[3]
Advanced Space, LLC, Westminster, CO 80234

This study evaluates the use of neural networks to perform autonomous onboard maneuver design for the Lunar Reconnaissance Orbiter (LRO) and Lunar IceCube (LIC) missions. A previous paper by the same authors presented a general framework, Neural Networks for Electric Propulsion (NNEP), for neural network-based autonomous maneuver planning and applied it to several astrodynamics problems, including a low-thrust cislunar transfer, a low-thrust interplanetary transfer, geostationary orbit station-keeping, and station-keeping in NASA Gateway's near-rectilinear halo orbit (NRHO). The framework also is implemented in flight software as a coreFlight System application. In this study, neural networks are used to recreate station-keeping operations of LRO in its low lunar mission orbit and simulate low-thrust trajectory corrections along LIC's lunar transfer.

I. Introduction

Current space mission operations require regular ground contacts for navigation and maneuver design. The typical navigation process, which includes telemetry downlink, orbit determination, maneuver design, hardware sequencing, and command uplink, requires several days to complete. Ground station scheduling and ground operator constraints can extend the timeline to over a week. During these delays, navigation errors propagate forward and uncertainties grow. At the time of execution, the maneuver instructions are based on old, imprecise navigation data.

As space missions trend toward operating in dynamically sensitive regimes and using constellations of small spacecraft, the need to automate operations increasingly becomes necessary. Using neural networks (NN) for onboard maneuver planning reduces the dependency on ground contacts and increases spacecraft robustness while delivering similar efficacy as ground team operations. In a previous paper, the authors presented a general neural network framework, Neural Networks for Electric Propulsion (NNEP), for onboard, autonomous maneuver planning [1]. NNEP leverages the powerful function approximation capabilities of NNs to learn the relationship between the spacecraft state and optimal control instructions. Ground computers generate tens of thousands of off-nominal trajectory corrections that cover the entire design space and train a NN using these samples. The NNEP framework requires minimal computational overhead and fits easily within modern flight hardware, allowing the NN to be evaluated onboard.

NNEP has been applied to a variety of astrodynamics problems successfully already and has been shown to deliver similar or better performance than a ground operations team. As part of ongoing efforts to mature and extend the technology in collaboration with NASA, this study initially was aimed at simulating trajectory correction operations during the cislunar transfer of the Goddard Space Flight Center (GSFC) Lunar IceCube (LIC) mission. Following the loss of the LIC mission, the focus of the study pivoted to recreating station-keeping (SK) operations of the Lunar Reconnaissance Orbiter (LRO) based on historical navigation data. These applications were selected because they are representative of upcoming civil, defense, and commercial space operations.

This paper presents the results of using NNs for simulated onboard autonomous maneuver planning, as applied to trajectory corrections along a cislunar transfer and station-keeping in a low lunar orbit (LLO). Section II provides an overview of the NNEP framework. Sections III and IV present the results of applying NNEP to the Lunar IceCube

[1] Staff Astrodynamics and Satellite Navigation Engineer, Advanced Space, LLC, 1400 W 122nd Ave, Westminster, CO 80234
[2] Principal Astrodynamics and Satellite Navigation Engineer, Advanced Space, LLC, 1400 W 122nd Ave, Westminster, CO 80234
[3] Software Engineer, Advanced Space, LLC, 1400 W 122nd Ave, Westminster, CO 80234

and Lunar Reconnaissance Orbiter missions, respectively. Section V introduces a future flight experiment involving NNEP onboard Advanced Space's CAPSTONE spacecraft.

II. Background

A. Neural Networks

Neural Networks (NNs) are mathematical models that serve as powerful function approximators. Inspired by biology, the models consist of a network of "neurons" that receive input, perform a simple mathematical operation, and pass the resulting output to the next neuron. Feed-forward NNs, the simplest type of NN, consist of several fully connected layers of neurons that pass information in one direction, from inputs to outputs. A diagram of a simple feed-forward model is given in Fig. 1. The model is trained by performing successive forward and backward passes of the neural network. During a forward pass, each neuron performs a linear operation on a given input $\vec{x}$ based on an associated weight and bias, then it applies a nonlinear activation function before the output is passed to the next neuron. During the backward pass, also known as back propagation, the partial derivatives of a defined cost function are computed with respect to each weight and bias. The weights and biases are then updated based on a gradient descent algorithm such that the final error between the predicted and desired model outputs is minimized.

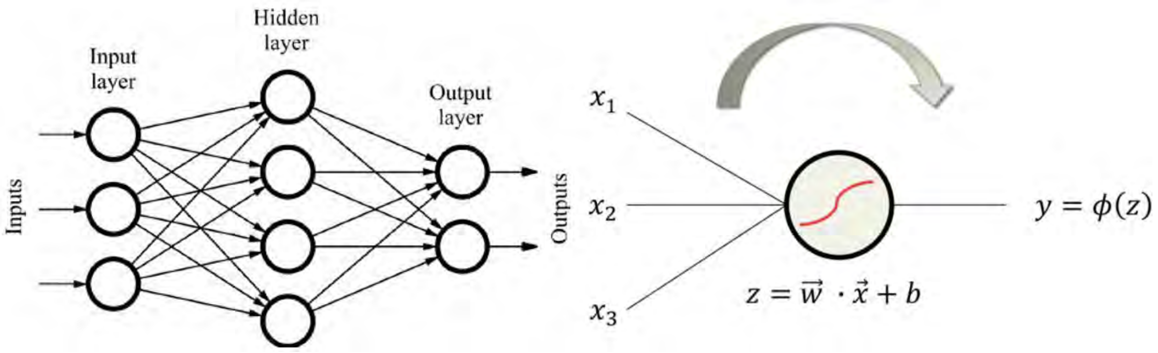


Fig. 1 Simple feedforward NN [2]. Neurons are arranged in fully connected layers, i.e., each neuron receives information from all neurons in the previous layer; performs a linear operation based on inherent weights and biases; and then applies a nonlinear activation function before passing the output to all neurons in the next layer. Information can be said to flow in one direction. The weights and biases are updated during the model's training loop.

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.

B. Neural Networks for Electric Propulsion

1. General approach for generating training data for electric propulsion applications

For low-thrust applications, the NNEP framework learns the relationship between a spacecraft's current state, which is perturbed from a nominal reference trajectory, and the optimal control theory costates (primer vector) for an optimal maneuver that returns to the reference trajectory. A reference trajectory must be available as an input to the NNEP framework but can be generated in any desirable manner, as the training sample generation pipeline only requires a time history of position and velocity reference states. A single, low-thrust training sample is generated by first choosing a time t^* at random from the reference trajectory's time span. The resultant position and velocity at time t^* are perturbed by $\delta\vec{r}$ and $\delta\vec{v}$, respectively. Each perturbation vector is drawn from random uniform distributions with lower and upper bounds $U(0, \delta r_{max})$ and $U(0, \delta v_{max})$. The choice of δr_{max} and δv_{max} define the size of the "training tube" shown in Fig. 2.

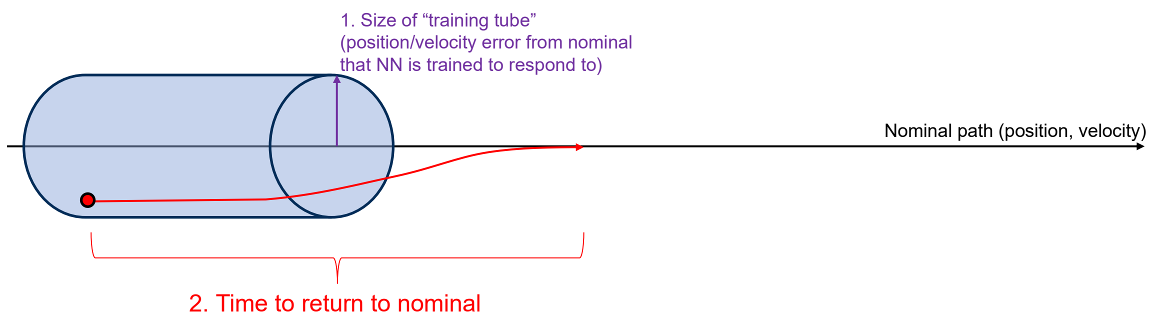

1. Size of "training tube"
(position/velocity error from nominal
that NN is trained to respond to)
Nominal path (position, velocity)
2. Time to return to nominal

Fig. 2 Illustration of the "training tube" around the reference path for low-thrust trajectory corrections. The size of the training tube is defined by the maximum state perturbation and is configurable for different missions or multiple models for one mission.

For the perturbed sample state, NNEP constructs the fixed time two-point boundary value problem (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 for missions in cislunar space or tens of days for interplanetary transfers. The use of a receding horizon always keeps the spacecraft close to the nominal path, thus keeping the spacecraft within the training tube.

An additional, configurable element of the training tube approach is the use of checkpoints at which the training samples are required to return to the reference path, as illustrated in Fig. 3. Prior work has shown that adding a few checkpoints at fixed times helps the spacecraft stay on a fuel-optimal trajectory. Checkpoints commonly are added immediately prior to sensitive events, such as a flyby (when small state errors can be hard to recover from) or at the start of long thrust arcs (when the control authority to respond to anomalies is more limited).

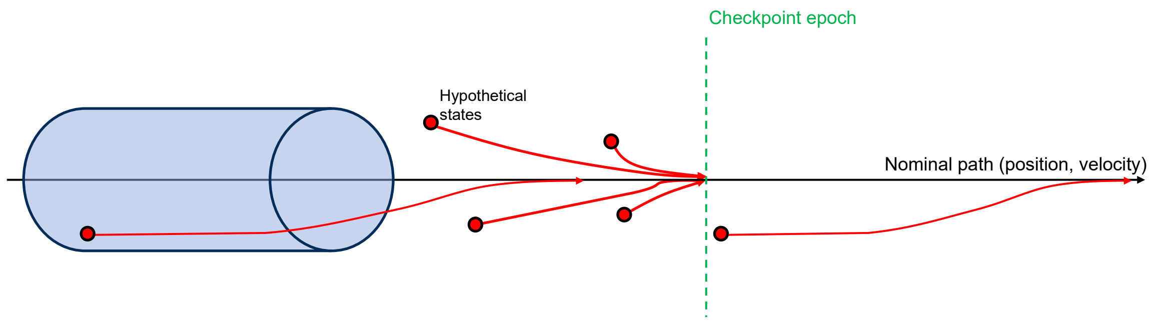


Fig. 3 Illustration of checkpoints with the "training tube." All training samples leading up to a checkpoint epoch are required to return to the nominal at the checkpoint, rather than at a receding time horizon.

Training samples are generated by solving the fixed time TPBVP for each perturbed training tube state using indirect single shooting. Partial derivatives for the trajectory solver are calculated using automatic differentiation, which allows the trajectory to be solved to numerical precision. Fig. 4 illustrates an example of the low-thrust training tube for an arbitrary reference path.

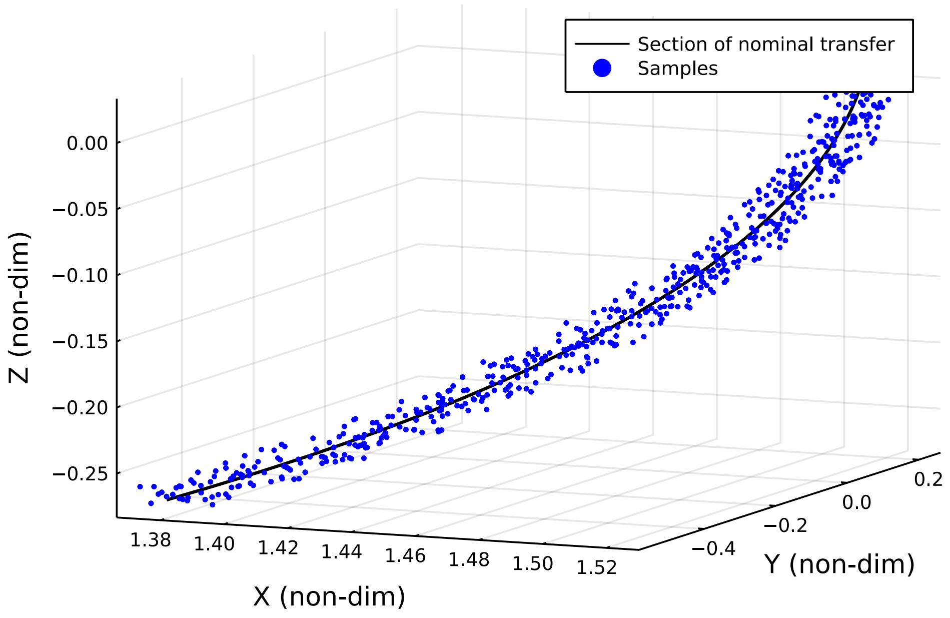


Fig. 4 Example training tube around an arbitrary section of a reference trajectory.

2. General approach for generating training data for chemical propulsion applications

For applications that can be modeled using impulsive maneuvers, such as station-keeping or trajectory correction maneuvers, NNEP learns the relationship between a spacecraft's current state and the desired control, which is commonly mission dependent. Therefore, the training sample generation pipeline for impulsive applications begins with first designing the nominal control strategy. For example, consider the application of autonomous station-keeping in the NRHO currently flown by Advanced Space's CAPSTONE mission [3]. The nominal SK strategy is to perform an orbital maintenance maneuver (OMM) up to once every revolution at an osculating true anomaly of 200°. Each maneuver is designed to target the velocity component in the $\hat{x}$ direction (v_x) of the Earth-Moon rotating frame and the time of periapsis (t_p) of the reference halo orbit at perilune passage 6.5 revolutions downstream.

Given the nominal control strategy, the sample generation can proceed in a manner similar to the low-thrust "training tube." Training samples are generated by perturbing the position and velocity of the reference orbit state at each OMM epoch by some $\delta\vec{r}$ and $\delta\vec{v}$, drawn from random uniform distributions. The nominal control strategy is applied to each perturbed state and the NN learns to map the relationship between perturbed state and control (ΔV) vector. Generating training samples in this way not only allows NNEP to be robust to errors that would force the spacecraft off its nominal trajectory, but also lets NNEP be adaptable to a variety of missions with unique control strategies and concept of operations (ConOps). Additional examples of applying NNEP to impulsive problems are presented in the authors' prior work [1], including station-keeping in a geostationary orbit, station-keeping in CAPSTONE's NRHO, and trajectory correction maneuvers for CAPSTONE's ballistic lunar transfer.

3. Machine learning framework

For the onboard implementation of NNEP, first a NN or sequence of NNs would be trained according to the strategies discussed previously. In both electric and chemical propulsion mission scenarios, ground computers are used to solve tens of thousands of perturbed trajectories. The solution for each provides a pair of input-output vectors: the current spacecraft state vector (potentially appended with mission dependent ancillary information) and the primer vector (for low-thrust applications) or ΔV vector (for impulsive applications) that represents the desired control strategy. Proprietary trajectory optimization tools developed by Advanced Space can generate tens of thousands of training samples in a few hours on a modern workstation computer. Additionally, the task can be parallelized across a distributed computing environment to reduce the computational time to minutes.

Once a sufficient dataset is generated—usually on the order of tens of thousands of training samples—a NN model is trained to map the relationship between input-output pairs using a custom version of the Levenberg-Marquardt (LM) algorithm. For small NNs containing up to tens of thousands of trainable parameters, the LM algorithm has been found to converge orders of magnitude faster than pure gradient descent methods [4]. At each training step the LM algorithm computes the Jacobian, a matrix of the partial derivatives of all training samples with respect to all trainable parameters, and updates the NN weights according to:

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

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 first- and second-order convergence; and $\vec{e}$ is the vector of residuals (the difference between the model output and the desired model output). The custom LM algorithm used by NNEP was built in the Julia language [5] with GPU acceleration via the CUDA.jl package [6]. A NN is trained for a fixed number of epochs, where an epoch represents passing the entire training set through the model and updating the model parameters once. A model can be trained for a few hundred training epochs on a modern workstation computer in only a few hours. Training time can be further reduced to minutes by leveraging powerful GPU architectures through cloud computing services.

A unique component of the NNEP architecture is the "NN Almanac" structure, a patent-pending structure that contains several NN models and ancillary information. With the Almanac structure, NNEP can switch readily between different NNs for different use cases entirely onboard without the need for multiple files. Examples include using different NN models for different mission phases (e.g., the mission transfer, orbit insertion, nominal orbit station-keeping, and extended mission operations all could be performed autonomously using different NN models) or switching to contingency models in the case of anomalies (e.g., a NN for nominal station-keeping operations and a backup NN if the spacecraft departs the orbit due to an anomaly). Several Almanac configurations are illustrated in Fig. 5. The Almanac's ancillary information maps each model within the structure to a corresponding use case, for example via epoch ranges or orbital geometries, and stores parameters used when simulating NNEP control. The individual models contain the NN weights and associated information, including the size of the model, input/output scaling, input/output shifting, and the model accuracy. The Almanac structure is saved as a binary file that is either loaded on the spacecraft prior to launch or uplinked to the spacecraft during flight.

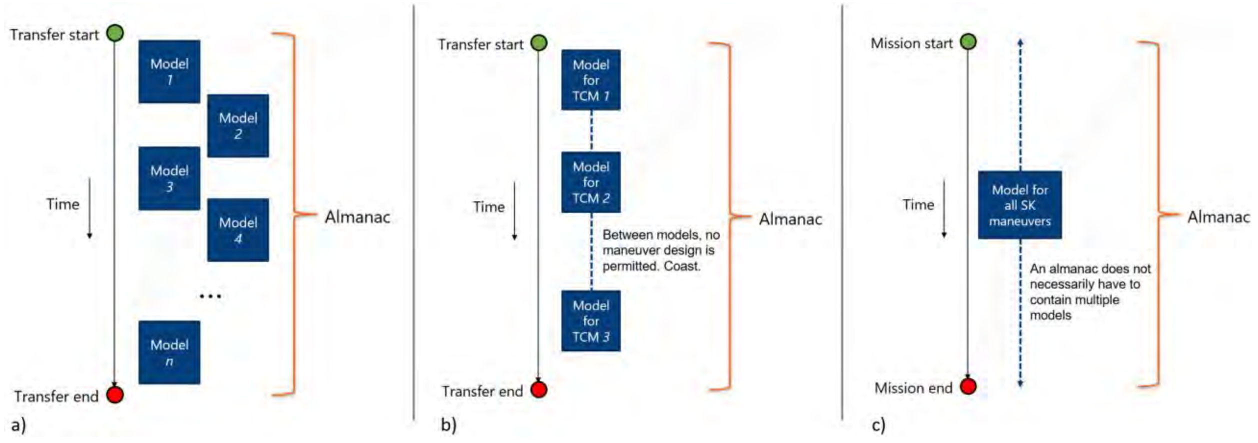


Fig. 5 Examples of different Almanac configurations. a) Low-thrust transfer - Each NN model is valid for a portion of the transfer. This helps the models be efficient in terms of memory use and computation cost. NNEP looks up the appropriate model based on the current epoch. b) Impulsive trajectory corrections – A different NN model is used for each maneuver. NNEP checks for a maneuver design only at specific epochs or ranges of epochs. c) Orbit station-keeping - An Almanac does not necessarily have to contain multiple models. The Almanac provides the model and any ancillary information, such as the model's valid operational lifetime.

4. ConOps for onboard operations

The spacecraft is assumed to perform onboard orbit determination (OD). Examples of onboard OD include the Cislunar Autonomous Positioning System (CAPS), which is currently being tested on CAPSTONE [3], optical navigation, and Global Navigation Satellite Systems (GNSS). At each navigation update, the spacecraft passes the best estimate of the current state into the NN model, which returns the primer vector (low-thrust applications) or ΔV vector (impulsive applications).

For low-thrust applications, trajectory corrections are assumed to occur continuously over the course of the transfer. The frequency of the correction “ticks” is configurable for each mission, but commonly occur with an update cadence on the order of tens of minutes. At each tick, NNEP evaluates the NN model, mapping the current state estimate to the costate vector; transforms the costates into the optimal thrust vector; and integrates the spacecraft state forward in time to the next tick with the control vector inertially fixed. The length of each tick is mission dependent, but often consists of less than a second of computation followed by several minutes of the spacecraft executing the commanded thrust. Fig. 6 illustrates the ConOps for continuous thrust missions.

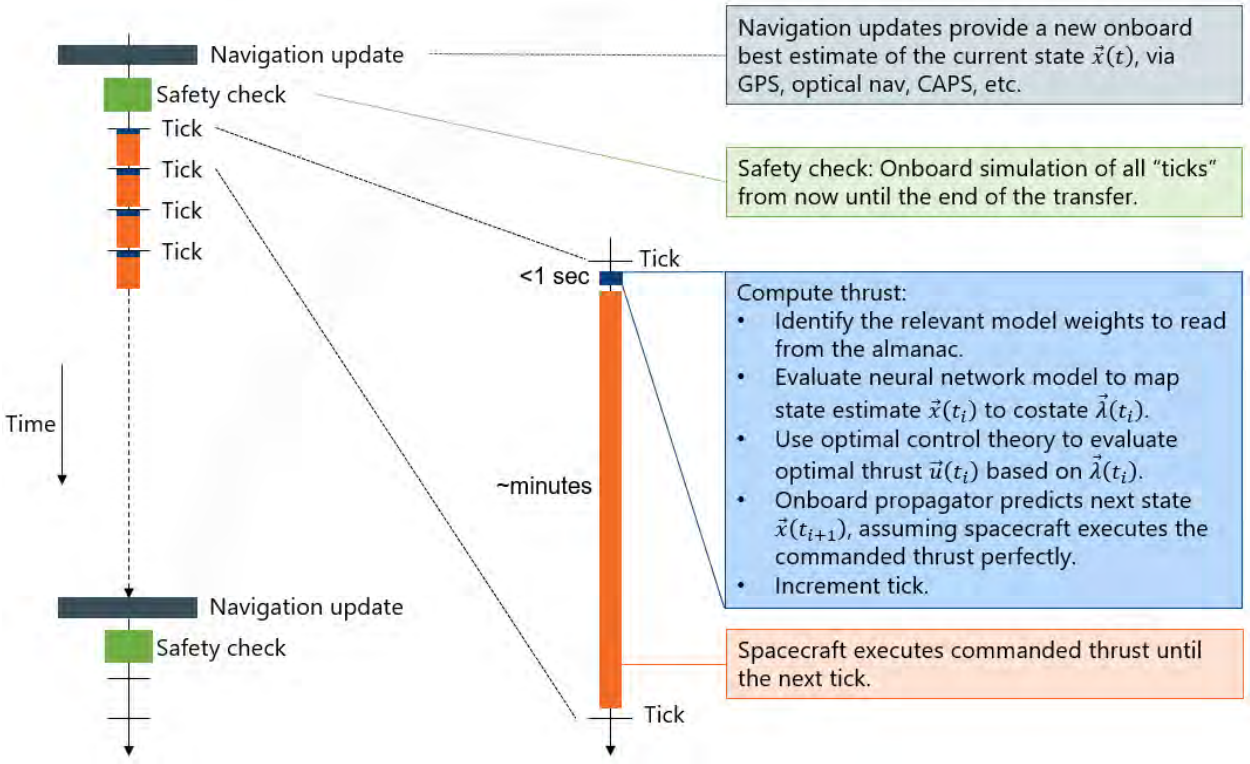
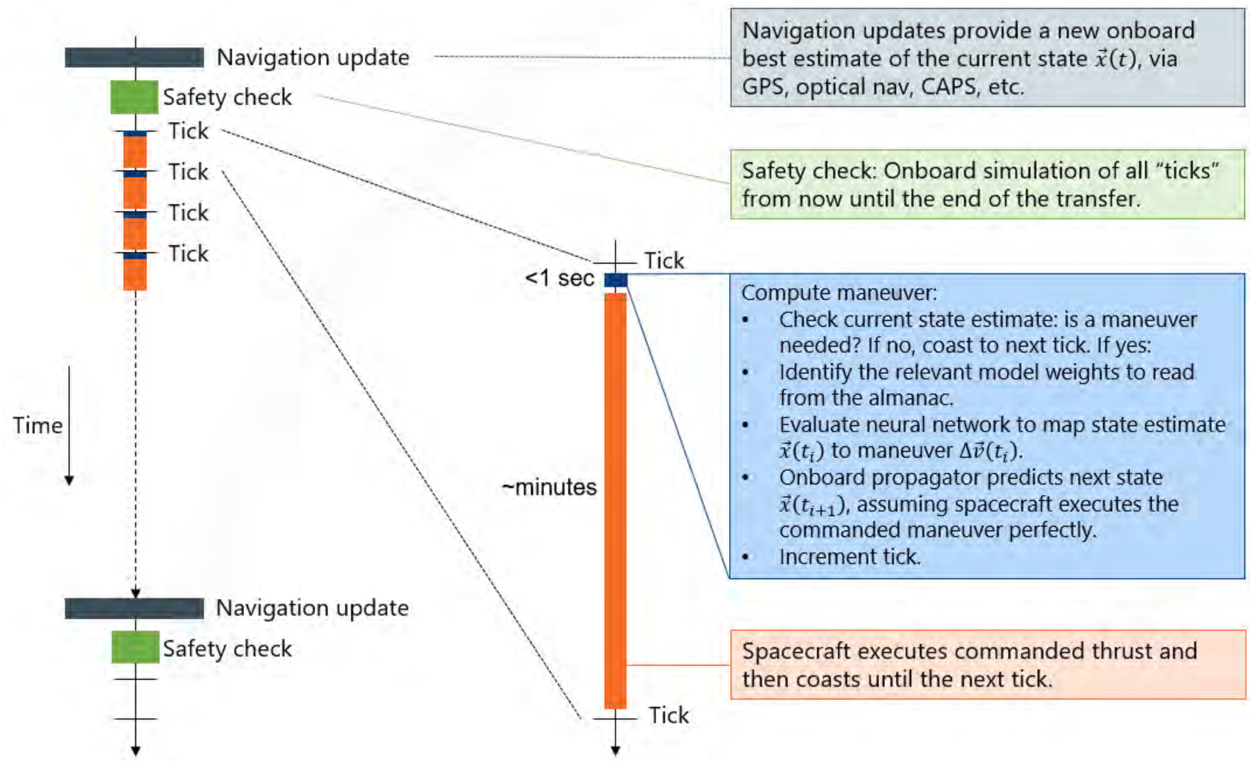

Navigation update
Safety check
Tick
Tick
Tick
Tick
Time
Navigation update
Safety check
<1 sec
Tick
~minutes
Tick
Navigation updates provide a new onboard best estimate of the current state $\vec{x}(t)$, via GPS, optical nav, CAPS, etc.
Safety check: Onboard simulation of all “ticks” from now until the end of the transfer.
Compute thrust:
• Identify the relevant model weights to read from the almanac.
• Evaluate neural network model to map state estimate $\vec{x}(t_i)$ to costate $\vec{\lambda}(t_i)$.
• Use optimal control theory to evaluate optimal thrust $\vec{u}(t_i)$ based on $\vec{\lambda}(t_i)$.
• Onboard propagator predicts next state $\vec{x}(t_{i+1})$, assuming spacecraft executes the commanded thrust perfectly.
• Increment tick.
Spacecraft executes commanded thrust until the next tick.

Fig. 6 NNEP onboard Concept of Operations for low-thrust applications. The frequency of NNEP ticks and navigation updates is mission dependent. The safety check is also configurable for each mission.

For impulsive applications, trajectory corrections occur at fixed events, for example, specific epochs in the case of trajectory corrections along a cislunar transfer or specific orbit geometries for station-keeping in an NRHO. Therefore, while the ConOps for impulsive applications is similar to that for low-thrust cases, each tick contains an initial check to determine if a maneuver is needed. This check is separate from the “safety check” that occurs after each navigation update and instead checks for predetermined conditions within the current state estimate, such as whether a specific epoch has been reached. If a maneuver is required at the current tick, NNEP continues as before, evaluating the NN model to receive the optimal ΔV vector and using the onboard propagator to integrate the spacecraft state to the next tick. If a maneuver is not required, the spacecraft does not execute a maneuver and instead coasts to the next tick. Fig. 7 illustrates the ConOps for impulsive control missions.



Navigation update
Safety check
Tick
Tick
Tick
Tick
Time
Navigation update
Safety check
Navigation updates provide a new onboard best estimate of the current state $\vec{x}(t)$, via GPS, optical nav, CAPS, etc.
Safety check: Onboard simulation of all "ticks" from now until the end of the transfer.
Tick
<1 sec
~minutes
Tick
Compute maneuver:
• Check current state estimate: is a maneuver needed? If no, coast to next tick. If yes:
• Identify the relevant model weights to read from the almanac.
• Evaluate neural network to map state estimate $\vec{x}(t_i)$ to maneuver $\Delta\vec{v}(t_i)$.
• Onboard propagator predicts next state $\vec{x}(t_{i+1})$, assuming spacecraft executes the commanded maneuver perfectly.
• Increment tick.
Spacecraft executes commanded thrust and then coasts until the next tick.

Fig. 7 NNEP onboard Concept of Operations for impulsive applications. The frequency of NNEP ticks and navigation updates is mission dependent. The safety check also is configurable for each mission.

A key component of NNEP for onboard operations is the "safety check" that occurs after each navigation update. After receiving a new onboard state estimate, NNEP simulates all ticks from the current time to the end of the transfer, or, in the case of station-keeping applications, the end of the mission orbit lifetime. At each iteration, the onboard software checks the simulated state estimates or spacecraft hardware against predefined, mission dependent constraints. Examples of safety checks include an altitude constraint for station-keeping to ensure the spacecraft does not impact the surface; an orbital elements constraint for station-keeping to ensure the spacecraft does not depart the mission orbit; a position error constraint for transfers to ensure the spacecraft does not deviate too far from the nominal transfer; and a hardware check to ensure thrusters are operating nominally. If a mission-defined constraint is violated, NNEP enters a standby mode and sends a message to ground operators. The interim operation of NNEP during the standby mode is mission dependent. For example, during transfers when there is sufficient time to remedy a ground solution, the default action after failing a safety check might be to command the spacecraft to coast until ground operators can diagnose any issues. In contrast, in a low lunar orbit where time is a constraint and the spacecraft may impact the surface, the default action might instead be to execute a maneuver sequence that places the spacecraft into a higher, safer orbit while ground operators design a solution. While the NN models used for NNEP are fully deterministic and rigorously tested on the ground to ensure proper operation in all foreseeable situations, the NNEP safety check provides additional security against any potentially unseen anomalies.

5. *Flight software implementation*

The flight software (FSW) implementation embeds core functionality into a C++ library. The library allows system and functional interfaces to be overridden to accommodate unique mission constraints and operations. For example, the safety check implementation may be overridden to perform custom checks given a specific NN and mission. A C-language wrapper also is provided to embed NNEP into C-specific applications.

NASA GSFC's coreFlight System (cFS) was leveraged to embed NNEP functionality into a flight-safe ecosystem. cFS provides a platform-agnostic framework and environment to enable the reuse of software across missions and environments by compartmentalizing units of functionality into "apps.". The implemented NNEP cFS app handles core operating functionality along with features for commanding NNEP, requesting telemetry, and downlinking results. It is designed as a state machine with standby and error states in cases of off-nominal behavior. Fig. 8 provides an overview of the app as a state machine.

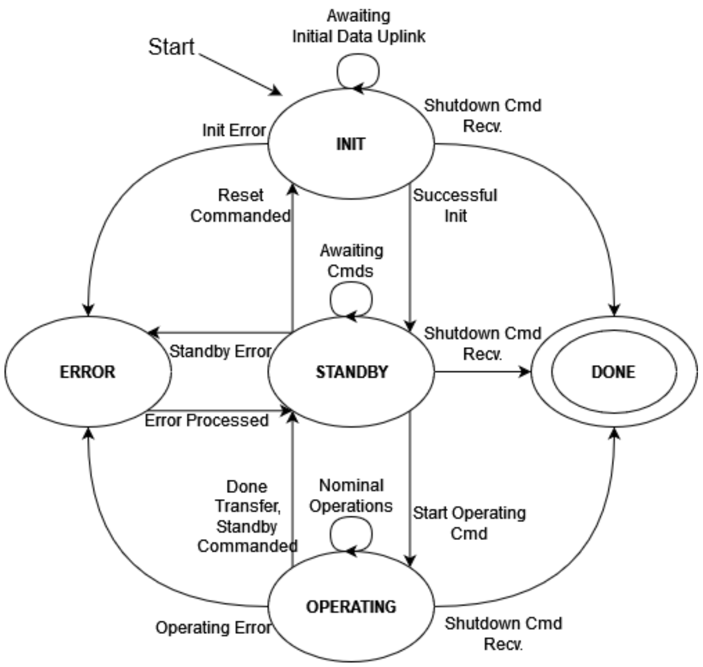


Fig. 8 The NNEP coreFlight app as a state machine.

Although the exact size and resource requirements of the NNEP app depend on mission-specific implementation details, the NNEP app ran multiple implementations on a Picozed 7030 flight board comprised of an 800 MHz ARMv7 processor without vectorization capabilities, 1 GB of RAM, and 10 MB of storage. NNEP app implementations are typically about 1 MB in size, including all dependencies. Likewise, Almanacs can vary largely in size for a given mission. Past implementations have used Almanacs on the order of tens to hundreds of KB. Small Almanacs allow for low memory footprints, typically faster predictions, and quicker Almanac uplink times in cases of Almanac updates. Tests have been conducted using a mock ground station (COSMOS v4.5) and test flight boards (Picozed 7030) on multiple mission implementations. The NNEP app has been tested with nominal and off-nominal behavior and data, as well as across simulated multiday executions. The app has proven to run as expected in the lab for each mission.

III. Lunar IceCube

A. LIC Mission Overview

In collaboration with NASA GSFC, as part of ongoing efforts to mature and transition the technology for flight opportunities, NNEP was adapted to NASA's Lunar IceCube mission. The objective was to conduct "shadow operations" during LIC's cislunar transfer. In other words, NNEP was to be implemented on the ground alongside the traditional human-in-the-loop ConOps to validate the performance of the NN architecture.

Training samples and NN models are generated from a GSFC-provided reference trajectory for LIC with a launch date of November 16, 2022. The nominal trajectory consists of a lunar flyby followed by a low-energy transfer to an eventual NRHO insertion (the final NRHO is the same NRHO that is currently occupied by Advanced Space's CAPSTONE and that will be flown by NASA's Gateway). GSFC also provided ancillary information about the LIC spacecraft's characteristics: an initial spacecraft mass of 13.487 kg, a maximum thrust of 1.1 mN, and a specific impulse (Isp) of 2156.0 seconds (s). The reference trajectory in an Earth-centered J2000 frame is illustrated in Fig. 9.

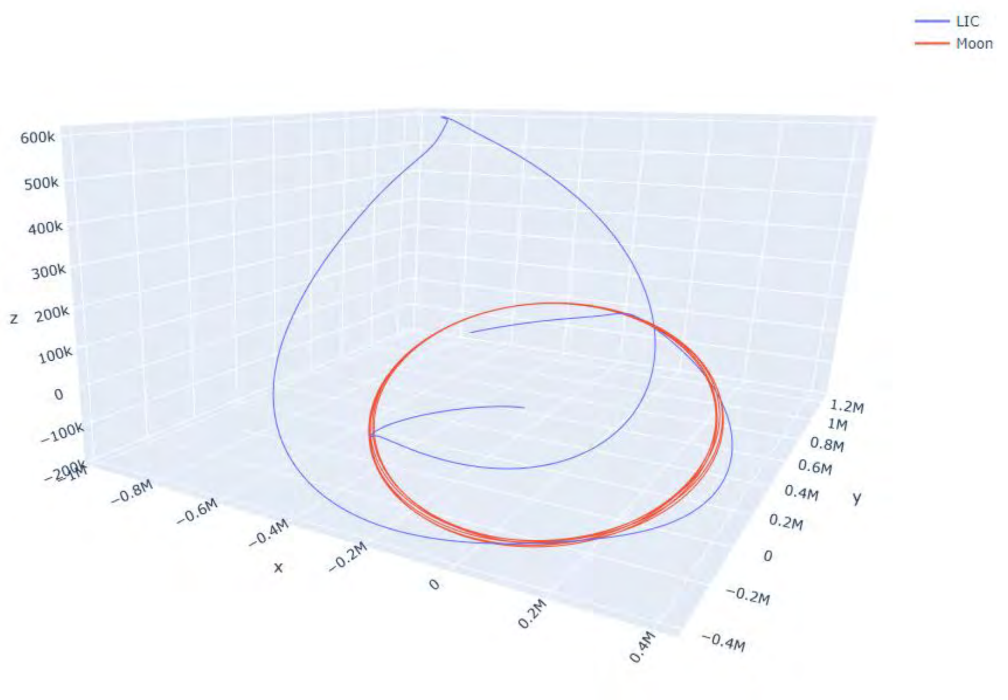


Fig. 9 Lunar IceCube reference trajectory for a November 16, 2022 launch. The transfer consists of a lunar flyby followed by a low-energy transfer to an NRHO.

Training samples for the NN are generated following the “training tube” approached described in *General approach for generating training data for electric propulsion applications*. A useful component of the low-thrust sample generation is the ability to declare “checkpoints” at which the training samples must return to the nominal path. In this application, it is critical to be on the nominal trajectory for the lunar flyby as failing to do so can cause a significant state deviation post-flyby. Therefore, a checkpoint is inserted just prior to the flyby epoch to ensure NNEP rendezvouses with the reference trajectory for this critical event.

Sixteen NN models, each responsible for approximately a 10-day segment of the transfer, are trained using the resulting training samples. Using several NNs, each covering a small section of the transfer, has been shown to produce better results than attempting to train one NN for the entire transfer. Each NN maps epoch and state error to the appropriate costate correction that rendezvouses with the reference path Δt = 30 days later. Each NN is a simple feedforward model consisting of three hidden layers of 30 neurons each.

B. LIC Results

To evaluate the performance of NNEP for trajectory corrections along the LIC transfer, simulated autonomous operations for LIC are conducted in a high-fidelity environment using NNEP to supply control vectors. The initial post-deployment state is generated from a random distribution informed by GSFC. The simulation includes navigation errors of 5 km/s, 5 cm/s 1σ that mimic onboard navigation errors for a system like Advanced Space’s Cislunar Autonomous Positioning System (CAPS). To model thrust execution errors, an error vector time series is created for the length of the transfer at the start of each trial. The timeseries consists of randomly drawn error vectors every 8 hours, which perturb any desired thrust occurring during that time span in a consistent direction and magnitude. Table. 1 summarizes the simulated errors.

Table. 1 Simulation error summary

Error	Magnitude
Onboard navigation	5 km, 5 cm/s
Thrust execution	2%

Results show NNEP can deliver sufficient, autonomous trajectory corrections for the LIC transfer. Table. 2 gives the results of a 1000-trial Monte Carlo simulation using NNEP as a controller from Earth launch to just before NRHO insertion.

Table. 2 Summary of results for Monte Carlo simulation of NNEP (n=1000)

Mean fuel consumption (kg)	0.111
Std. fuel consumption (kg)	0.011
Mean final position error (km)	130.302
Std. final position error (km)	150.992
Mean final velocity error (m/s)	1.675
Std. final velocity error (m/s)	1.598

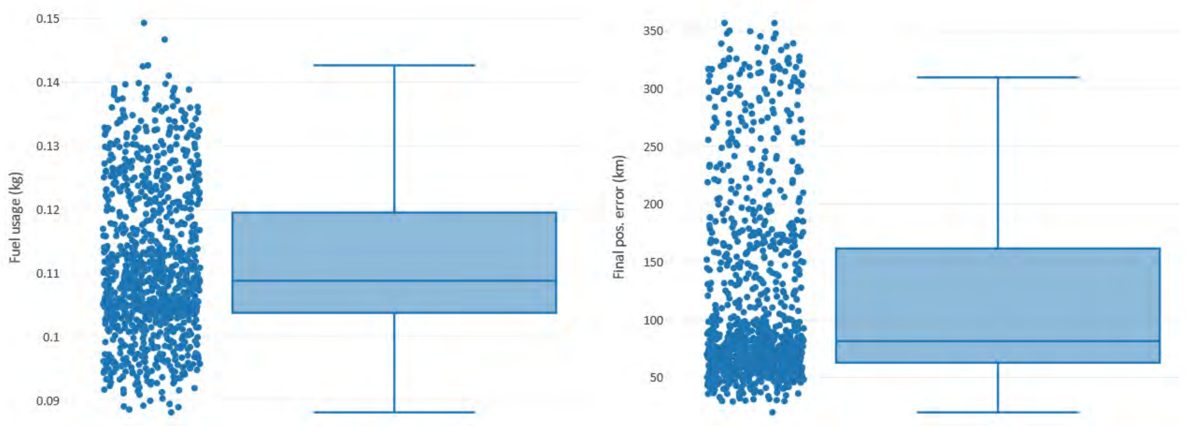


Fig. 10 Plot of results for Monte Carlo simulation of NNEP (n=1000). Each individual point represents a single simulated run of using NNEP for trajectory corrections along LIC's transfer.

A reoptimized nominal trajectory using the Nov. 16th launch LIC reference consumes a total of approximately 0.085 kg. NNEP uses, on average, about 0.026 kg more fuel than the nominal trajectory. Experience has shown that the fuel consumption of NNEP can be improved by fine-tuning the parameters of the framework, such as the rendezvous checkpoints and NN hyperparameters. Unfortunately, due to the loss of the LIC mission, the subject of the "shadow operations" effort changed to LRO. Therefore, the authors were unable to fine-tube this application of NNEP to achieve the best performance. Future work would reduce the mass difference and final state errors between NNEP and LIC further.

IV. Lunar Reconnaissance Orbiter

A. LRO Mission Overview

With the loss of the Lunar IceCube mission, Advanced Space could not complete the intended LIC transfer shadow operations. Instead, the application was quickly pivoted to replicating historical operations of the Lunar Reconnaissance Orbiter. Specifically, NNEP was applied to station-keeping operations in the 50 km low lunar orbit LRO occupied during its primary mission from September 2009 through December 2011. This orbit and time period were chosen because they are representative of the operations of future NASA missions, such as the planned Lunar Exploration and Science Orbiter (LExSO) mission. LExSO plans to have an onboard navigation capability via optical navigation. If paired with an onboard maneuver design capability such as NNEP, it would be capable of autonomous station-keeping in LLO.

During its primary mission phase from September 2009 to December 2011, LRO flew in a 50 km low lunar polar orbit. The dynamic environment of this orbit causes the spacecraft to follow a natural eccentricity (ECC) and argument of periapsis (AOP) pattern, when viewed in a Moon body-fixed frame, which repeats every lunar period. A polar plot of the natural motion of the 50 km LLO over one lunar period is shown in Fig. 11. The pattern begins in the upper left and "drifts" to the upper right over one lunar period (~27.4 days).

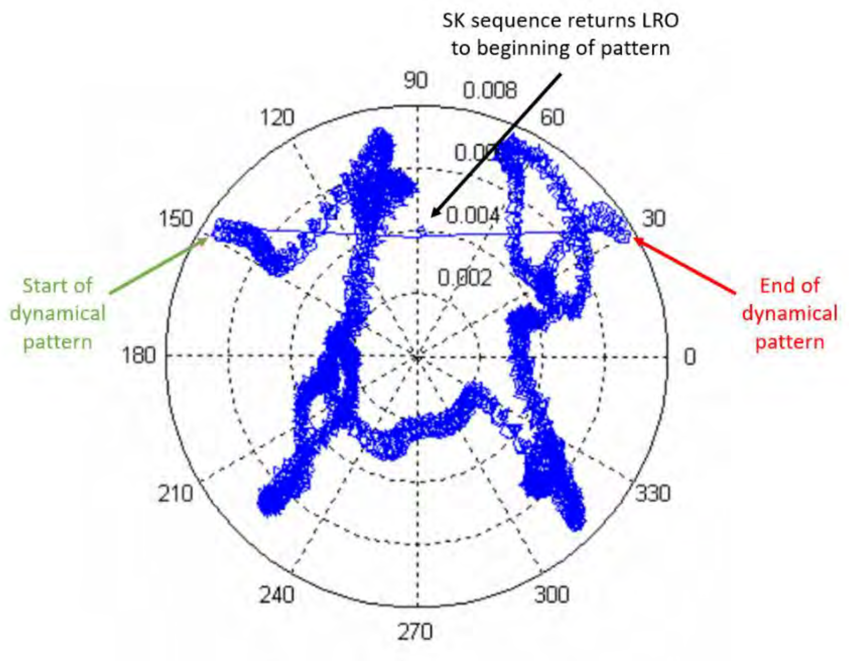


Fig. 11 Natural dynamical pattern of LRO as viewed in a lunar-fixed frame over one lunar period. The radial and theta axes are eccentricity and argument of periapsis respectively. The pattern begins in the upper left and "drifts" to the upper right. A SK sequence then returns the spacecraft to the beginning of the pattern. [7]

For a supervised learning approach such as NNEP, the accuracy of the trained models is highly dependent on the quality of the training samples. Therefore, it is critical that the samples provided to the model during training are representative of LRO's operations. With feedback from NASA LRO operators, the dynamical environment and SK algorithm of LRO was recreated in MONTE, a robust astrodynamics Python library developed by the Jet Propulsion Laboratory (JPL). The dynamical model for LRO operations is described in Table 3.

Table 3. Dynamical model for simulated LRO operations.

Parameter	Value
Lunar gravity	GRAIL-900, 150x150
Planetary ephemeris	DE430
Point mass bodies	Earth, Sun, Jupiter Barycenter
Solar radiation pressure	Cr = 1.0, Flat area = 14 m^2

The nominal LRO SK strategy is a two-burn line-of-apsides rotation maneuver sequence. The first burn, designated Ska (ex. SK2a, SK3a, …, SK25a), is primarily in the spacecraft's velocity direction and places the spacecraft into a transfer orbit with a larger semi-major axis. The second burn, designated SKb, occurs approximately three hours after the corresponding SKa and is primarily in the spacecraft's anti-velocity direction. The SK sequence occurs approximately once per lunar period, with some variance due to operator scheduling and ground station constraints, and returns LRO to the start of the dynamical pattern. The spacecraft then follows the same natural motion until the next SK maneuver.

The results of recreating the dynamics of the LRO reference trajectory in MONTE are presented in Fig. 12 and Fig. 13. Fig. 12 shows the ECC versus AOP dynamical pattern of the LRO recreation in MONTE compared to the LRO reference ephemeris (BSP). Fig. 13 plots the position and velocity error magnitudes between the MONTE recreation and the reference ephemeris. The MONTE recreation closely follows the reference (errors on the order of less than one meter) until approximately halfway into the lunar period. At this point, the reference trajectory performs a momentum desaturation (desat) maneuver which imparts a small change in velocity. Momentum desats are not modeled in the MONTE simulation (as historical data on LRO's executed desats is not readily available) and therefore, at this point, the MONTE recreation begins to diverge from the reference. The sample generation strategy and simulation approach are designed to account for these unmodeled desats while still testing the efficacy of NNEP.

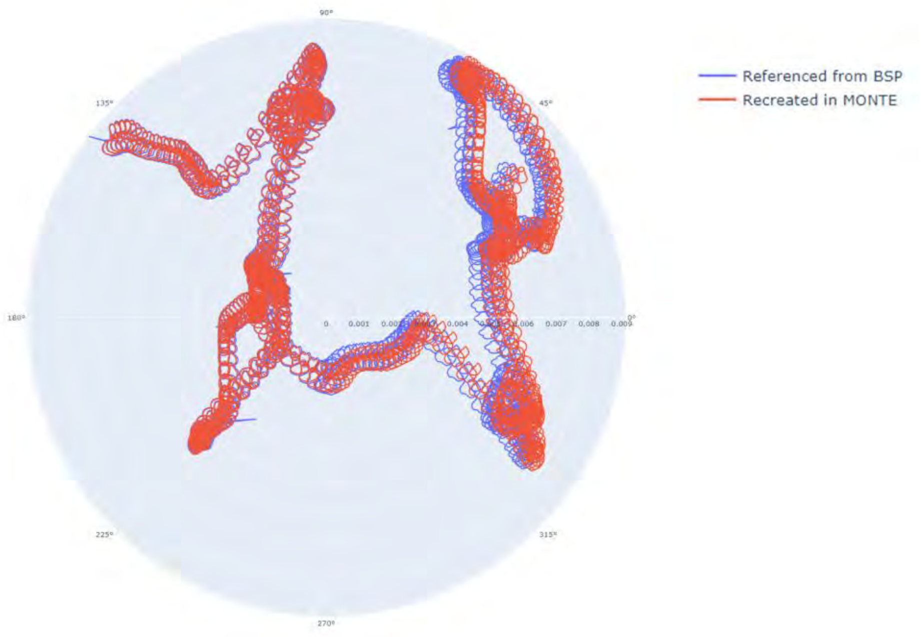


Fig. 12 Polar plot of reference orbit versus MONTE recreation in a Moon body-fixed frame.

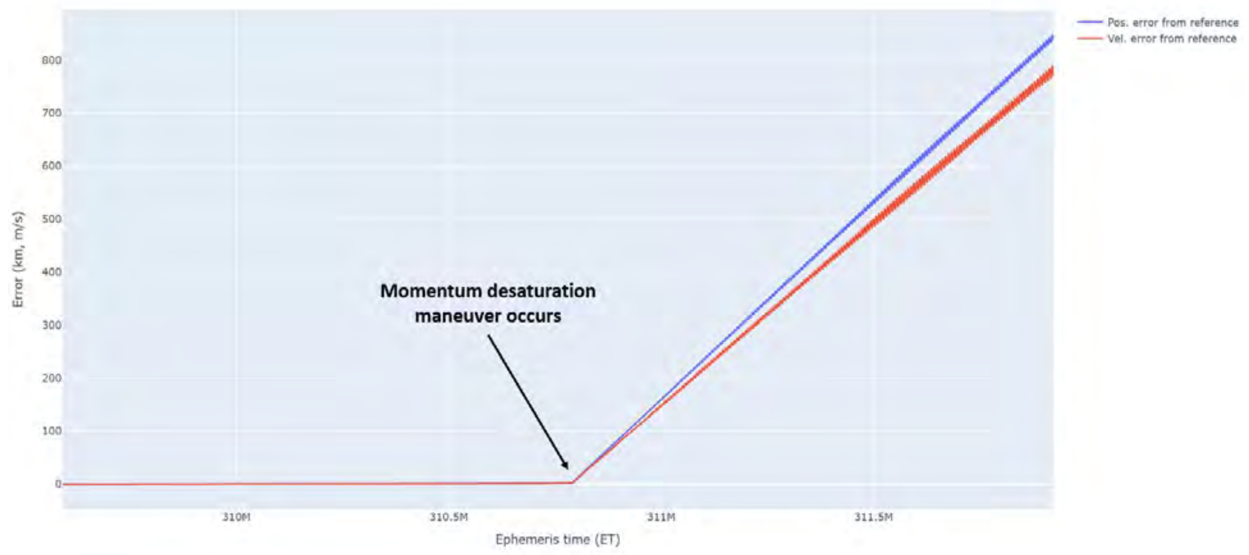


Fig. 13 Position and velocity error between reference and MONTE recreation. Approximately 14 days into the simulation, the reference trajectory contains a desat maneuver.

The training samples generation strategy is to generate a distribution of "off-nominal" states for each SK maneuver around the initial reference maneuver state and to apply the two-burn line-of-apsides SK strategy such that the reference target state is achieved at the reference target epoch. This approach is similar to the "sample tube" strategy used by low-thrust transfer applications, except samples are generated at defined times rather than across the entire reference trajectory.

One NN model is trained to provide SK maneuvers for the entire nominal 50 km mission orbit. The NN accepts 12 inputs: the six-dimensional Keplerian orbital elements describing the spacecraft state before a given SKa maneuver, the three-dimensional Moon-to-Sun position vector, and the three-dimensional Moon-to-Earth position vector. Each input is based in the International Astronomical Union (IAU) Moon-Fixed reference frame. The NN delivers seven outputs: the three-dimensional SKa impulsive delta-V (DV) vector, the three-dimensional SKb impulsive DV vector, and the time between SKa and SKb. Other than the input and output layers, the model contains three hidden layers of 30 neurons each.

B. LRO Results

To evaluate the performance of NNEP, simulated operations for LRO are performed using NNEP to supply SK maneuvers. For each SK maneuver during the September 2009 to December 2011 nominal mission orbit period, a navigation update is simulated by referencing the LRO ephemeris. This navigation state, along with the necessary ancillary information, is provided to the NN. NNEP returns the DV vectors for SKa and SKb, as well as when to perform SKb. Monte is used to execute the SKa maneuver; propagate LRO to the SKb epoch in a high-fidelity environment; execute SKb; and then propagate one hour past the SKb epoch. At SKb plus one hour, the "NNEP state" described above is compared to a "reference state" pulled from the LRO ephemeris to produce a metric of error. Fig. 14 and Fig. 15 show this error metric and the maneuver magnitudes, respectively, for each LRO SK maneuver as flown by NNEP. Fig. 16 shows the eccentricity versus argument of periapsis dynamical pattern across a full lunar period after NNEP's design of the SK5 maneuver. The simulated LRO operations show that NNEP is capable of delivering sufficient SK maneuvers to maintain the nominal 50 km LRO lunar orbit. For each SK maneuver, the positional error one hour after the corresponding SKb is less than 2 km and each maneuver is of an expected magnitude (flown LRO maneuvers were, on average, ~6 m/s).

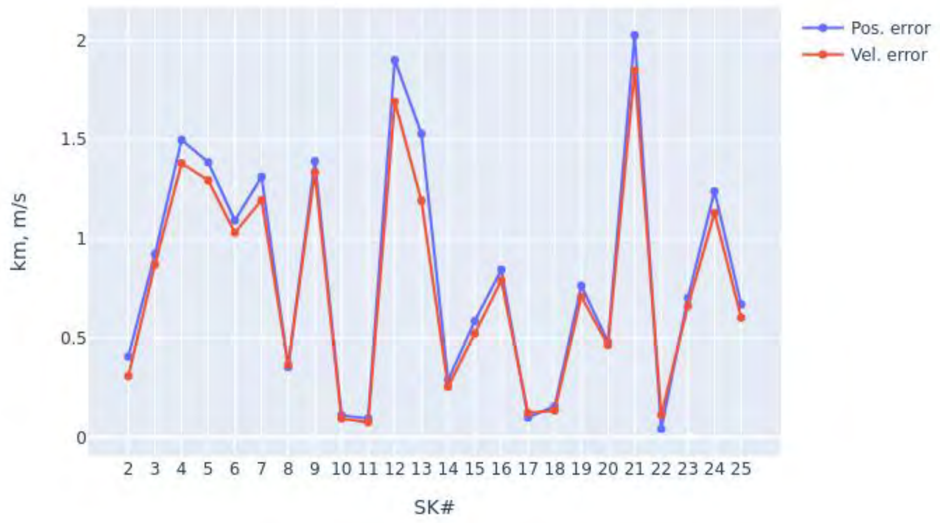


Fig. 14 Position and velocity error of NNEP versus LRO reference one hour after each SKb maneuver. SK1 is not simulated, as it was used to evaluate hardware performance and fell outside of the nominal SK strategy.

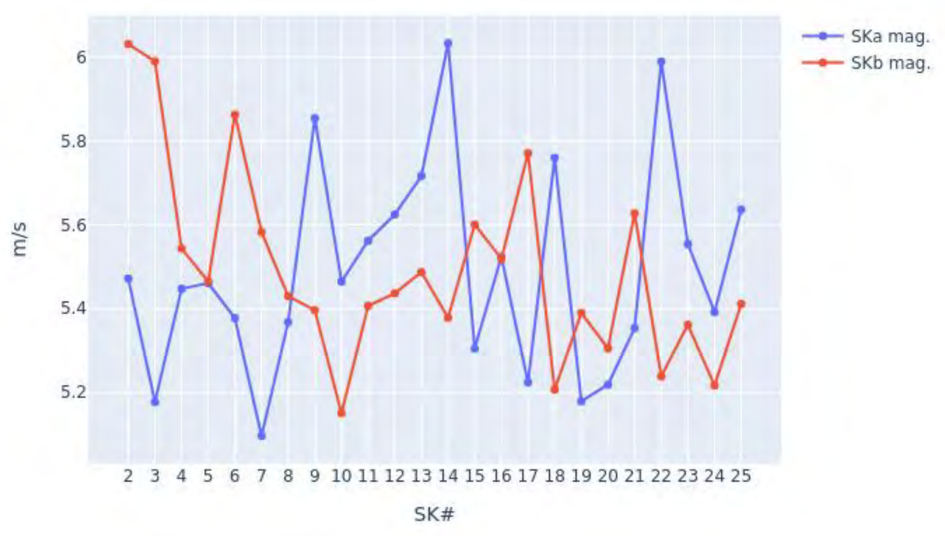


Fig. 15 Magnitudes of each SKa and SKb maneuver as delivered by NNEP. SK1 is not simulated, as it was used to evaluate hardware performance and fell outside of the nominal SK strategy. The magnitudes of the actual LRO maneuvers were about 6 m/s. NNEP delivers maneuvers of a similar magnitude.

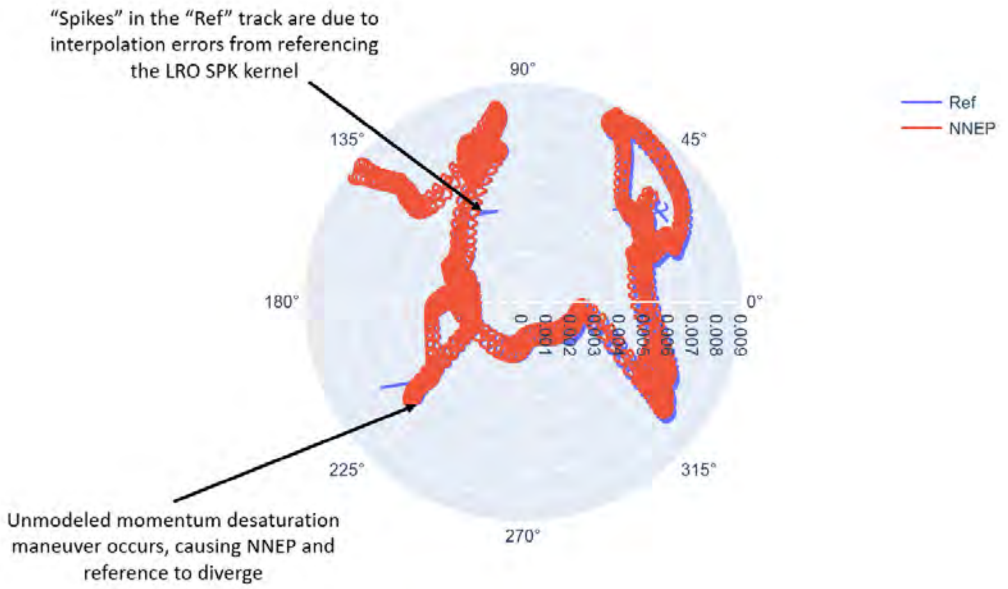


Fig. 16 Polar plot of reference orbit versus NNEP as-flown in a Moon body-fixed frame

C. Flight Software Changes for LRO

Prior to this effort, the onboard FSW was capable of handling only low-thrust applications of NNEP. As such, the LRO study serves as an interesting test of the FSW flexibility because it differs significantly from all previous electric propulsion mission tests. That is, impulsive maneuvers had to be converted to finite burns; the safety-check implementation had to be adapted for ensuring correctness based on the LRO constraints; the neural net required different inputs and preprocessing; and most importantly, the core execution logic had to be adapted to handle a two-burn maneuver sequence. Such changes tested the flexibility of the flight software and provided insights into how the software could be generalized further in future maturation periods. The final implementation was completed without modifying any of the core components that comprise the original FSW libraries and cFS application. It also demonstrated how the Almanac specification could be modified in the future to accommodate per-model preprocessing constants. Unlike previously tested missions, the LRO mission required neural net inputs to be scaled and shifted based on the model. For this specific training case, weights were added manually in a separate file. However, in a real mission, the preprocess scaling values would be encoded in the Almanac.

V. Future Work

Internal efforts at Advanced Space to mature the NNEP technology for flight readiness are underway. The primary objective is to implement NNEP onboard Advanced Space's CAPSTONE spacecraft for autonomous station-keeping in its NRHO. Initial tests will ensure the software integrates correctly with the rest of the CAPSTONE system. Preliminary functional tests will use ground generated state data to inference the NN models onboard the spacecraft. This ensures the software can run on a spacecraft while minimizing the risk and complexities of interacting directly with other on-board systems. Additional tests will work towards a complete onboard integration, relying entirely on NNEP for onboard autonomous maneuver planning. Deploying the NNEP flight software on CAPSTONE will further mature the technology and demonstrate its efficacy and robustness. The deployment of NNEP onboard CAPSTONE is planned for early 2024.

Conclusion

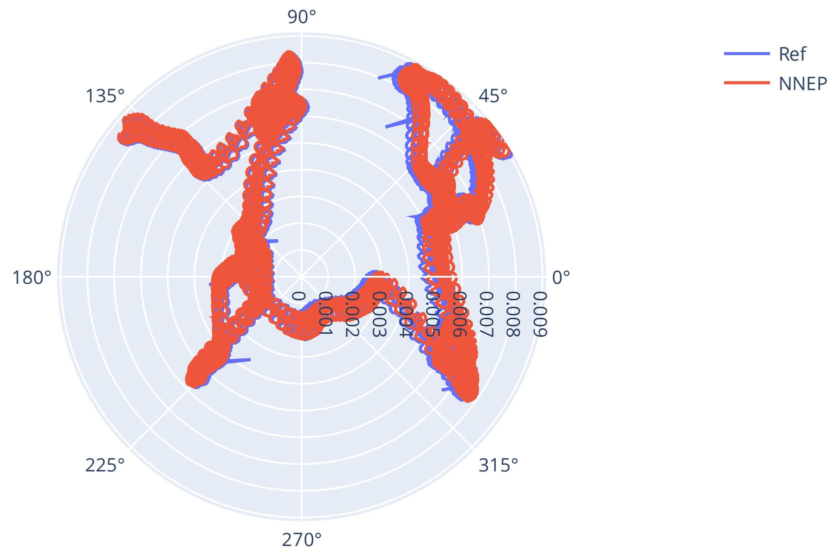
This paper presents the results of applying NNEP [1], a neural network framework for autonomous onboard maneuver planning, to the Lunar IceCube and Lunar Reconnaissance Orbiter missions. Each component of the framework is discussed, including sample generation strategies and examples for various mission types, NN training pipelines, and proprietary features such as the NN Almanac structure and NN safety check. Additionally, a concept of operations for onboard implementation and operations is outlined, along with a description of the flight software application. NNEP initially was adapted for simulated trajectory correction operations along LIC's low-energy cislunar transfer. However, following the loss of the mission, the focus of the study shifted to recreating station-keeping operations in LRO's low lunar orbit. The results demonstrate that NNEP can autonomously maintain the desired reference trajectory as effectively as human-in-the-loop ground operations when operating onboard a spacecraft.

The adaptation of NNEP to these missions is part of ongoing efforts to mature the technology for flight readiness. The applications are representative of future NASA missions, such as LExSO, which will fly in an orbit similar to LRO. Internal initiatives at Advanced Space also are underway to deploy NNEP onboard CAPSTONE, currently operating within the same NRHO that will be used by the Lunar Gateway, for autonomous onboard station-keeping. Demonstrating NNEP across various operational and dynamical environments serves as a crucial test for the robustness and efficacy of the architecture.

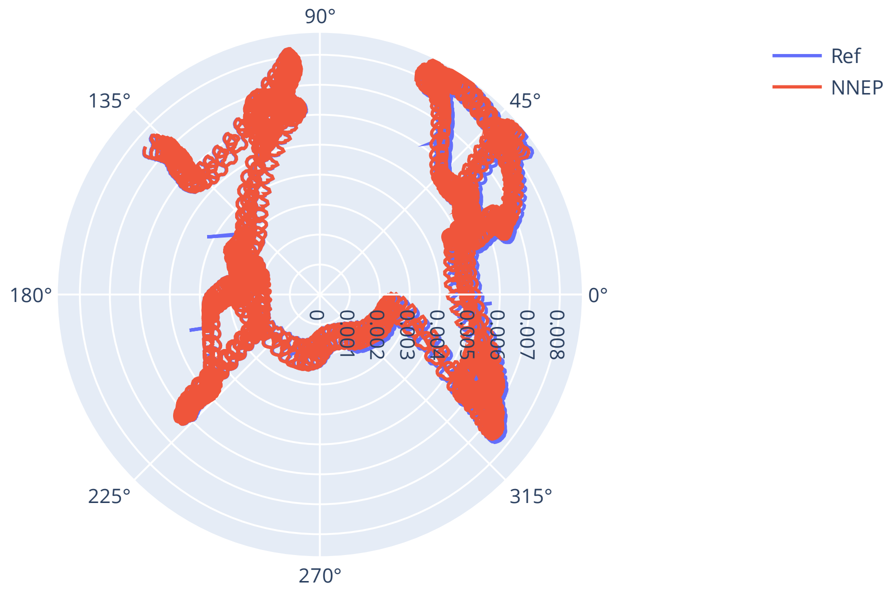
Appendix

Plots of the ECC and AOP pattern after each SK maneuver provide a good visualization of the full LRO shadow operations. The figures show the ECC and AOP, plotted on the radial and theta axes respectively, of LRO in a lunar-fixed frame for one lunar period after each SK maneuver. The plots contain a "Ref" trace, which shows the LRO as-flown ephemeris, and a "NNEP" trace, which shows the results of using NNEP to deliver SK maneuvers. Observe that NNEP closely follows the reference trace until approximately halfway through the trajectory, at which point NNEP begins to diverge due to LRO performing an unmodeled momentum desaturation ("desat") maneuver. The reference traces also frequently contain "spikes;" these "spikes" appear due to interpolation errors (polynomial ringing) when referencing the LRO SPICE ephemeris binary and do not represent the actual trajectory. Finally, SK1 is not included in the NNEP simulation because it was used to evaluate hardware performance and fell outside of the nominal SK strategy.

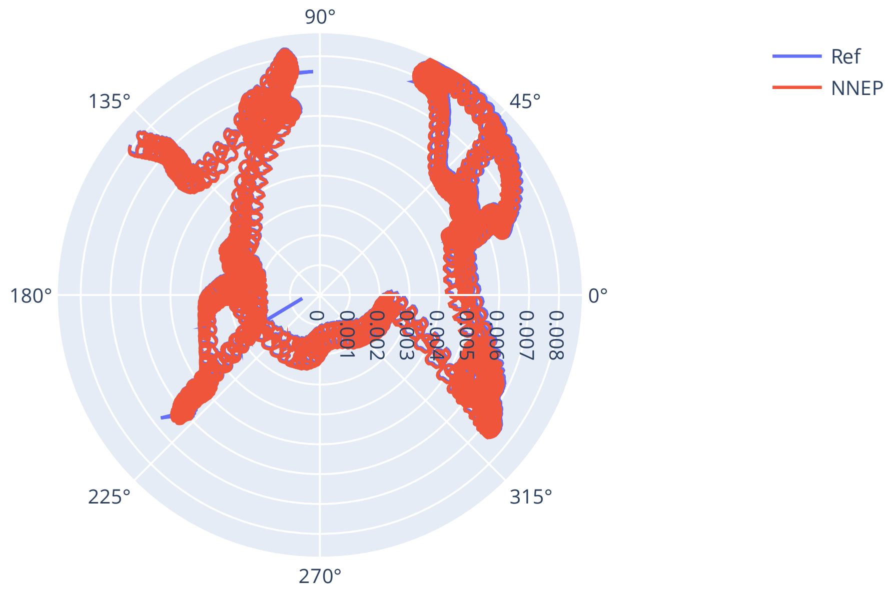
SK2


90°
135°
45°
180°
0°
225°
315°
270°
0
0.001
0.002
0.003
0.004
0.005
0.006
0.007
0.008
0.009
Ref
NNEP

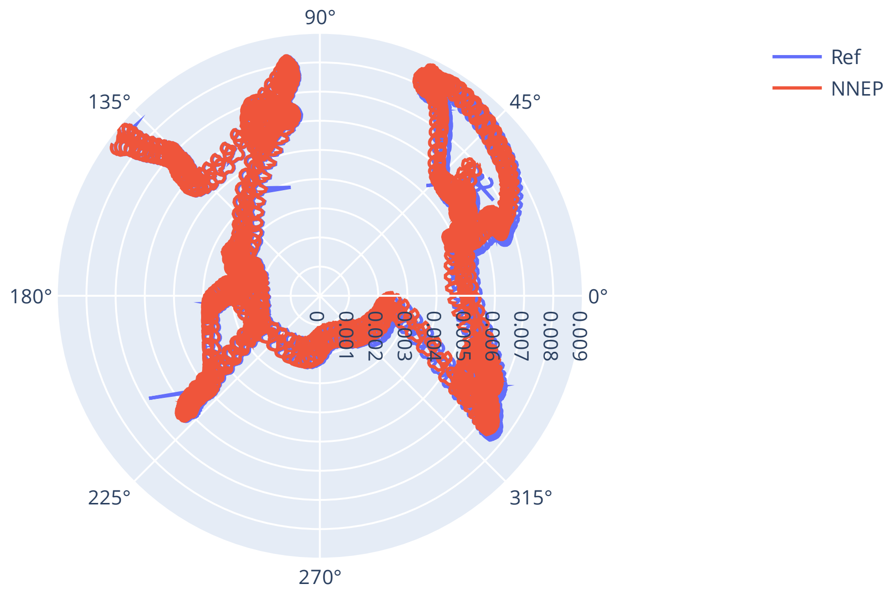
SK3


90°
135°
45°
180°
0°
225°
315°
270°
0
0.001
0.002
0.003
0.004
0.005
0.006
0.007
0.008
Ref
NNEP

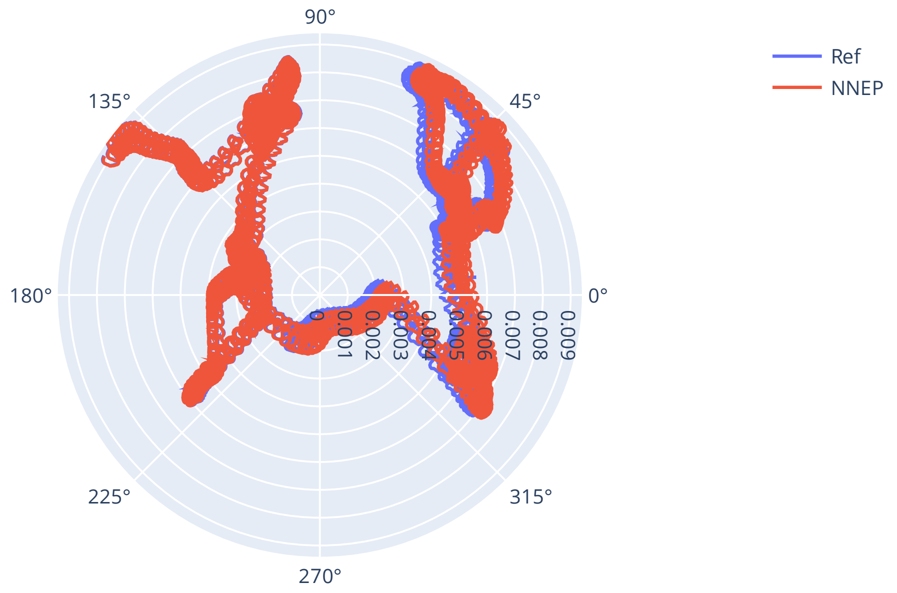
SK4


90°
135°
45°
180°
0°
225°
315°
270°
0
0.001
0.002
0.003
0.004
0.005
0.006
0.007
0.008
Ref
NNEP

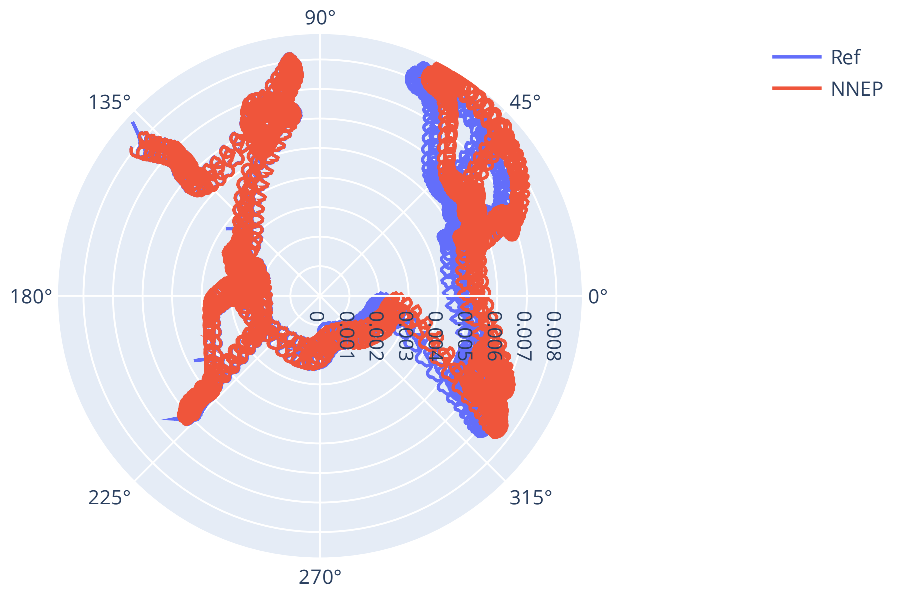
SK5


90°
135°
45°
180°
0°
225°
315°
270°
0
0.001
0.002
0.003
0.004
0.005
0.006
0.007
0.008
0.009
Ref
NNEP

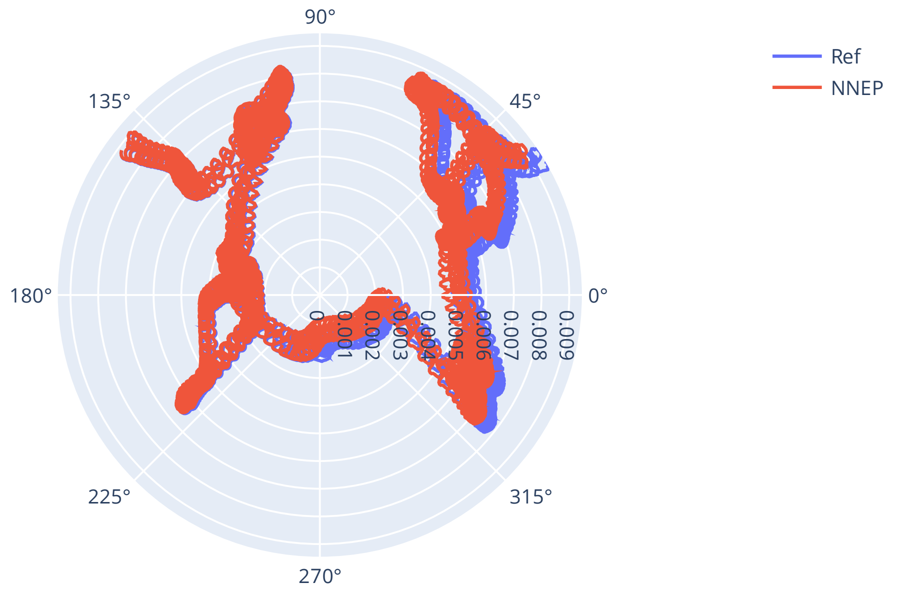
SK6


90°
135°
45°
180°
0°
225°
315°
270°
0
0.001
0.002
0.003
0.004
0.005
0.006
0.007
0.008
0.009
Ref
NNEP

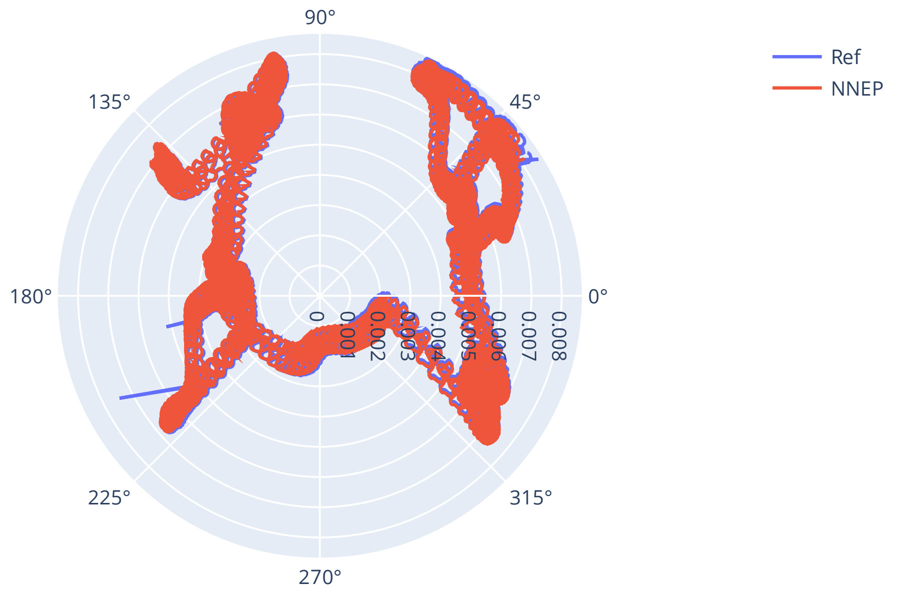
SK7


90°
135°
45°
180°
0°
225°
315°
270°
0
0.001
0.002
0.003
0.004
0.005
0.006
0.007
0.008
Ref
NNEP

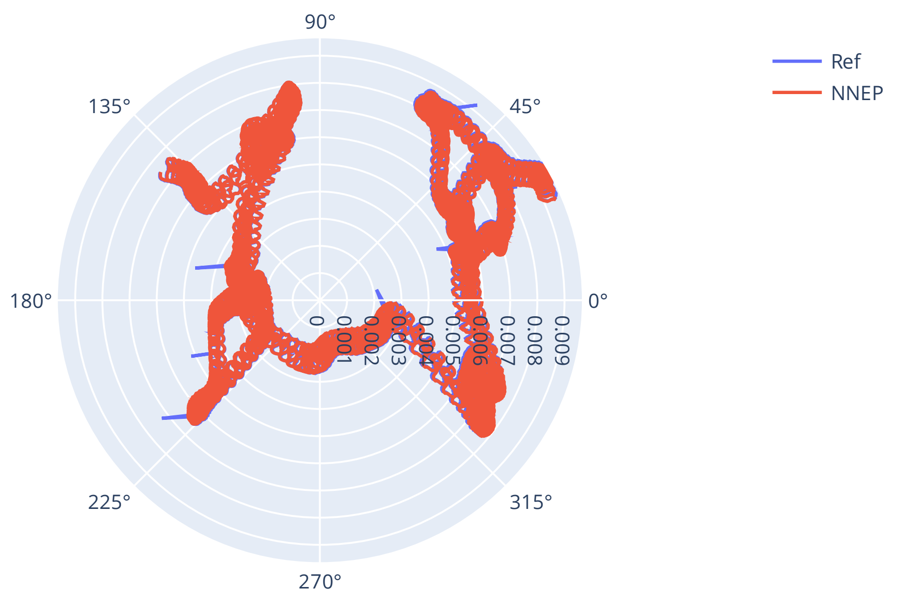
SK8


90°
135°
45°
180°
0°
225°
315°
270°
0
0.001
0.002
0.003
0.004
0.005
0.006
0.007
0.008
0.009
Ref
NNEP

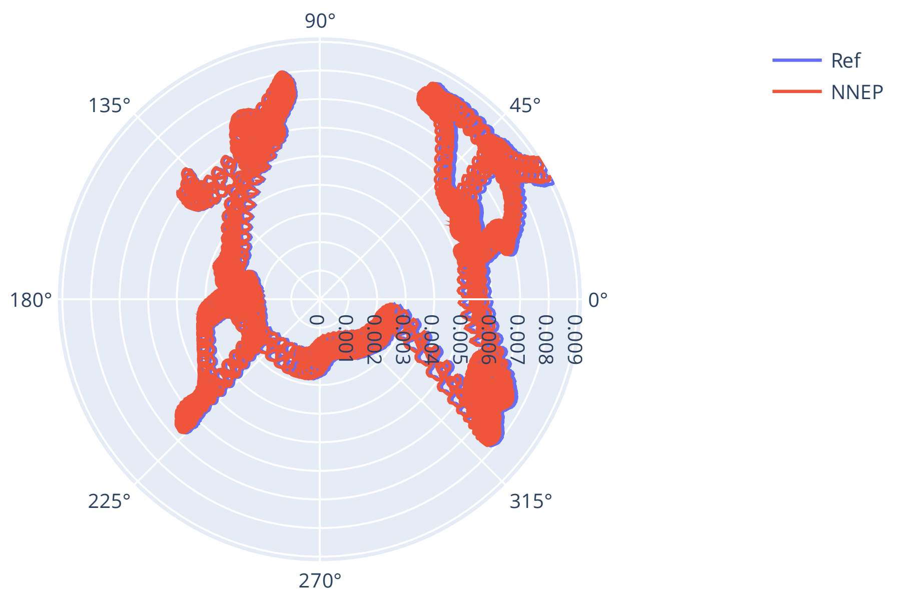
SK9


90°
135°
45°
180°
0°
225°
315°
270°
0
0.001
0.002
0.003
0.004
0.005
0.006
0.007
0.008
Ref
NNEP

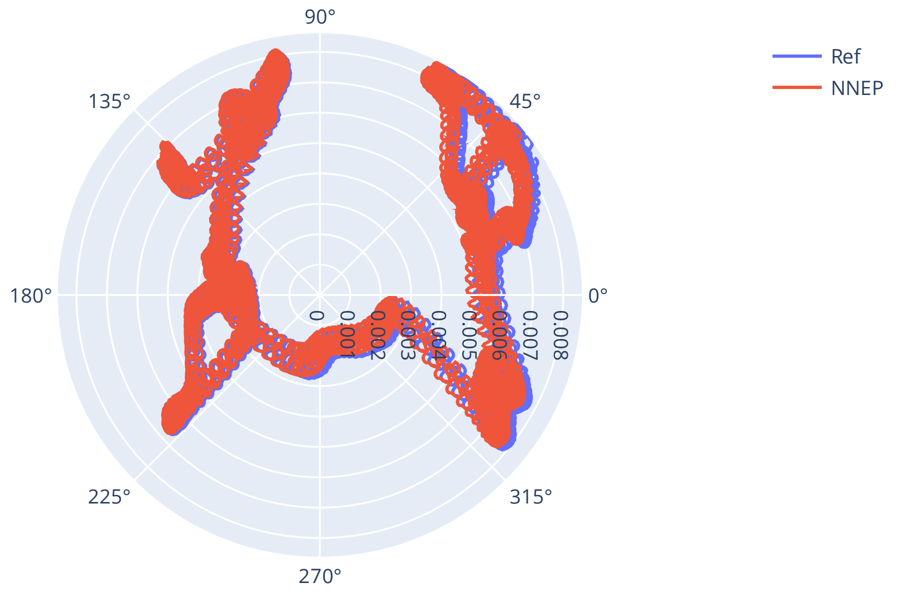
SK10


90°
135°
45°
180°
0°
225°
315°
270°
0
0.001
0.002
0.003
0.004
0.005
0.006
0.007
0.008
0.009
Ref
NNEP

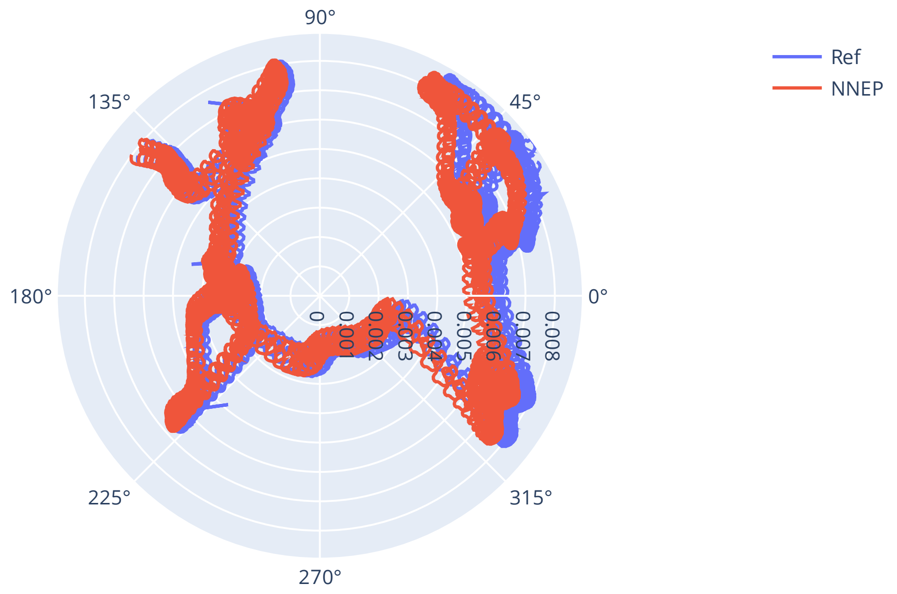
SK11


90°
135°
45°
180°
0°
225°
315°
270°
0
0.001
0.002
0.003
0.004
0.005
0.006
0.007
0.008
0.009
Ref
NNEP

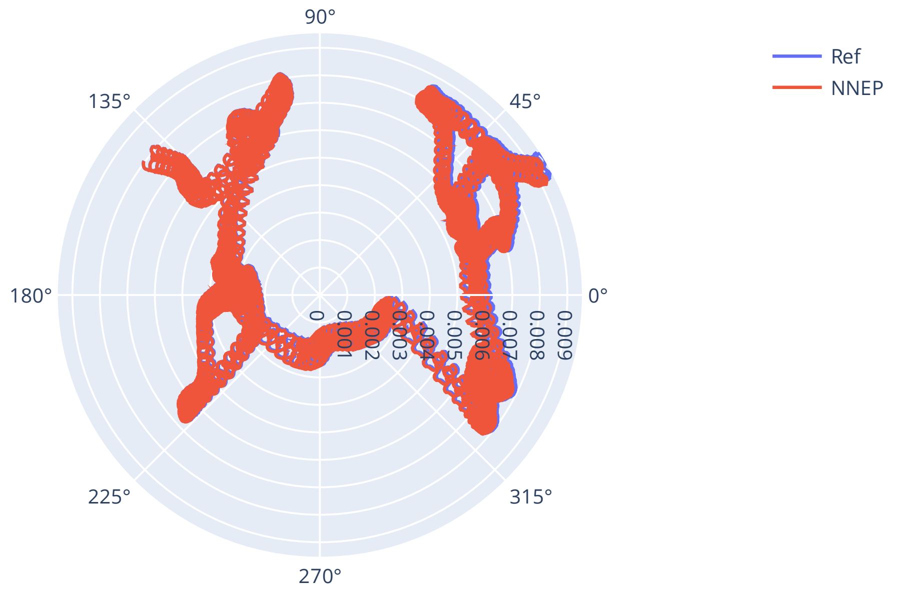
SK12


90°
135°
45°
180°
0°
225°
315°
270°
0
0.001
0.002
0.003
0.004
0.005
0.006
0.007
0.008
Ref
NNEP

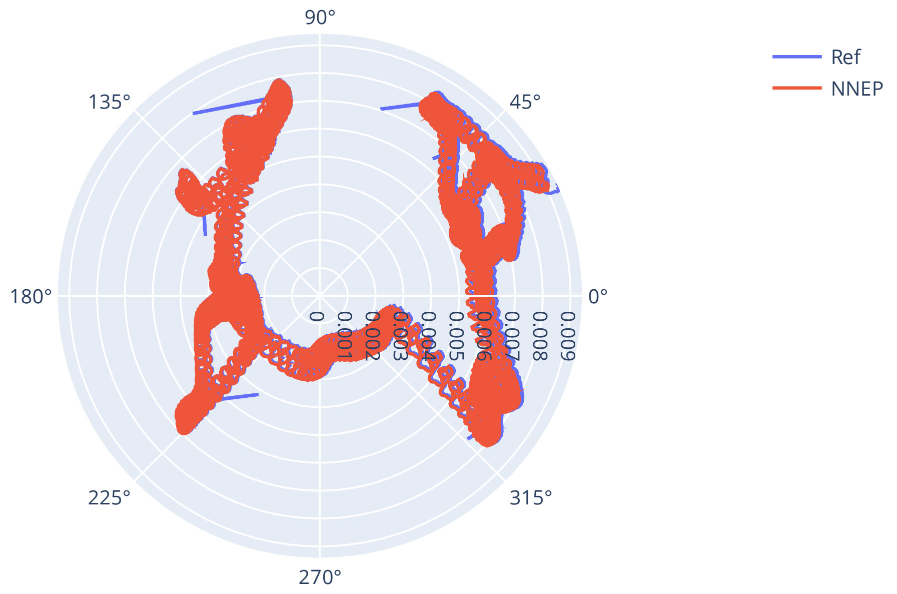
SK13


90°
135°
45°
180°
0°
225°
315°
270°
0
0.001
0.002
0.003
0.004
0.005
0.006
0.007
0.008
Ref
NNEP

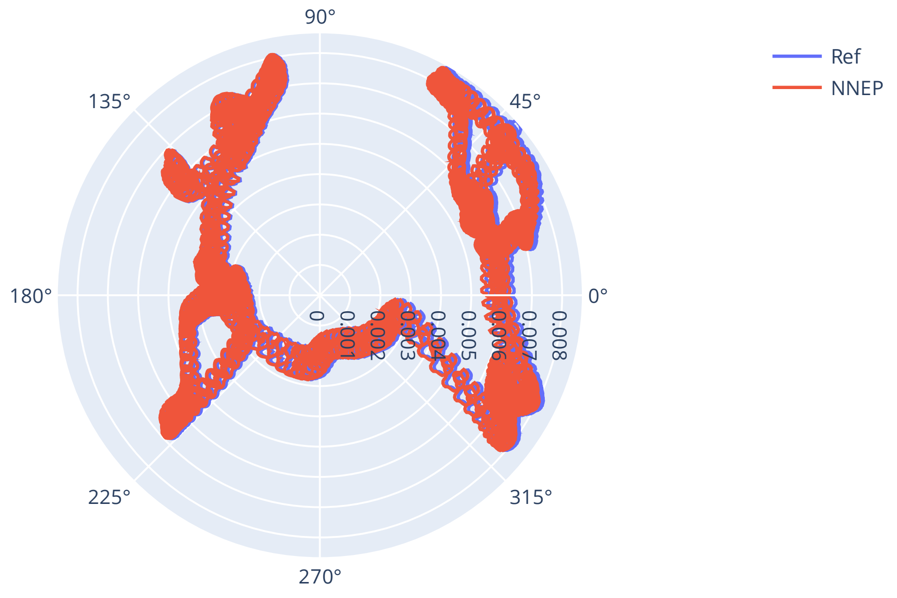
SK14


90°
45°
135°
180°
0°
225°
315°
270°
0
0.001
0.002
0.003
0.004
0.005
0.006
0.007
0.008
0.009
Ref
NNEP

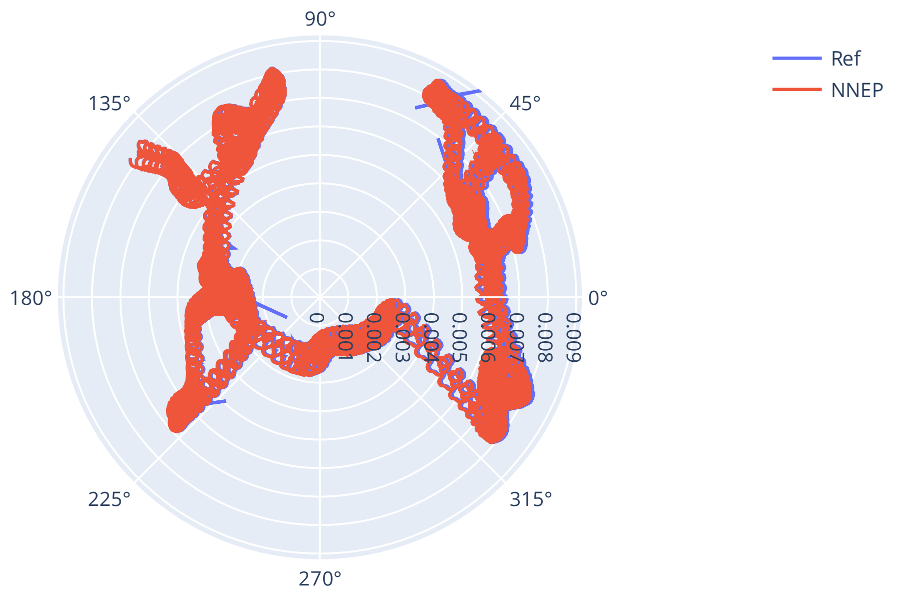
SK15


90°
45°
135°
180°
0°
225°
315°
270°
0
0.001
0.002
0.003
0.004
0.005
0.006
0.007
0.008
0.009
Ref
NNEP

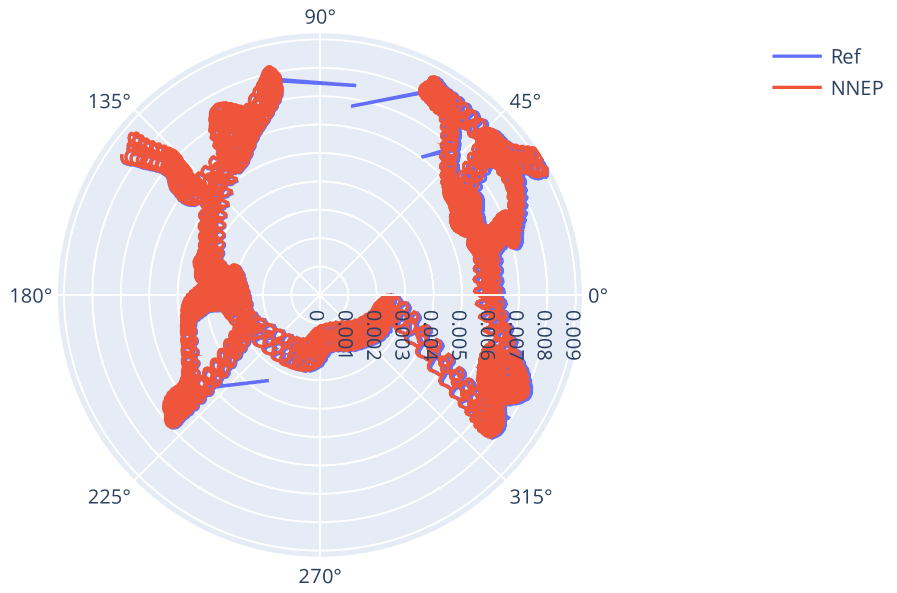
SK16


90°
135°
45°
180°
0°
225°
315°
270°
0
0.001
0.002
0.003
0.004
0.005
0.006
0.007
0.008
Ref
NNEP

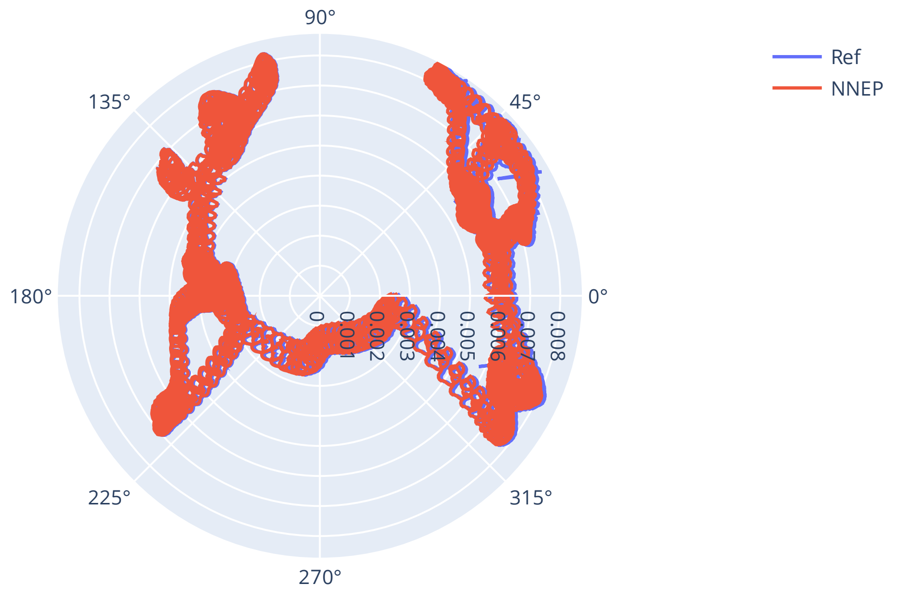
SK17


90°
135°
45°
180°
0°
225°
315°
270°
0
0.001
0.002
0.003
0.004
0.005
0.006
0.007
0.008
0.009
Ref
NNEP

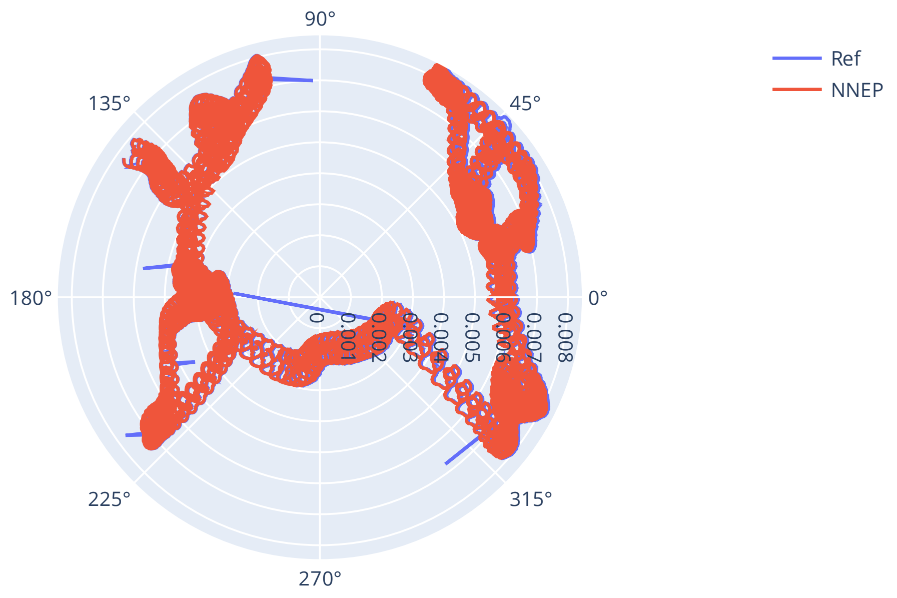
SK18


90°
135°
45°
180°
0°
225°
315°
270°
0
0.001
0.002
0.003
0.004
0.005
0.006
0.007
0.008
0.009
Ref
NNEP

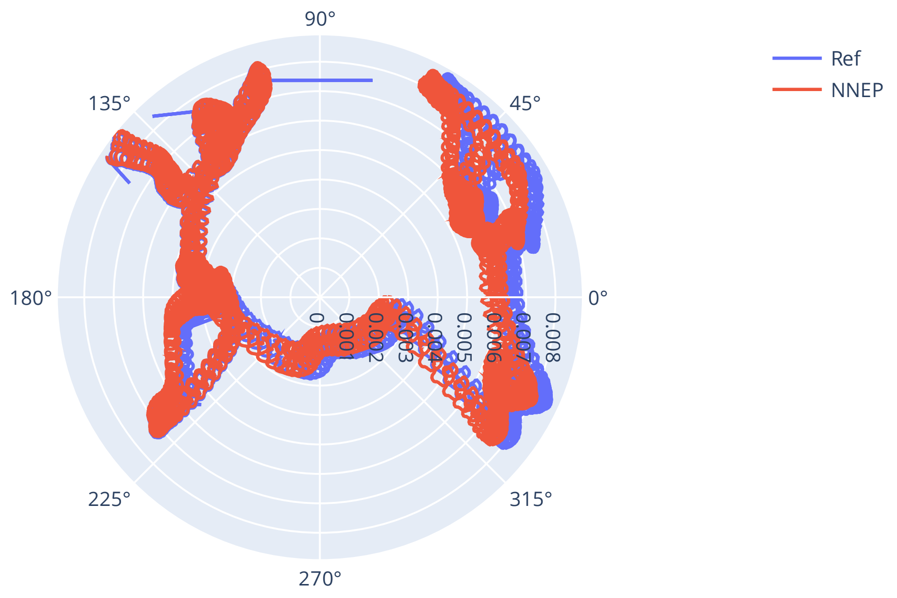
SK19


90°
135°
45°
180°
0°
225°
315°
270°
0
0.001
0.002
0.003
0.004
0.005
0.006
0.007
0.008
Ref
NNEP

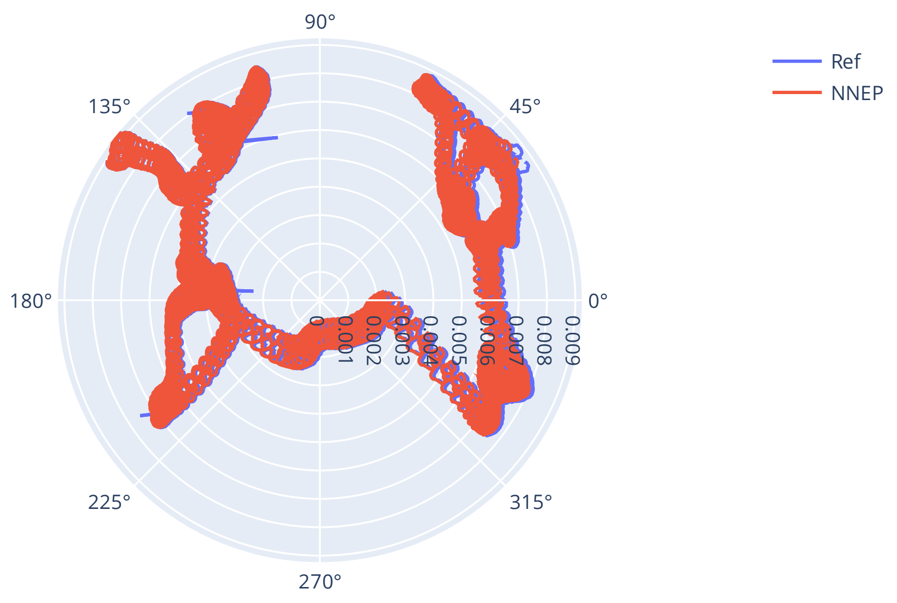
SK20


90°
135°
45°
180°
0°
225°
315°
270°
0
0.001
0.002
0.003
0.004
0.005
0.006
0.007
0.008
Ref
NNEP

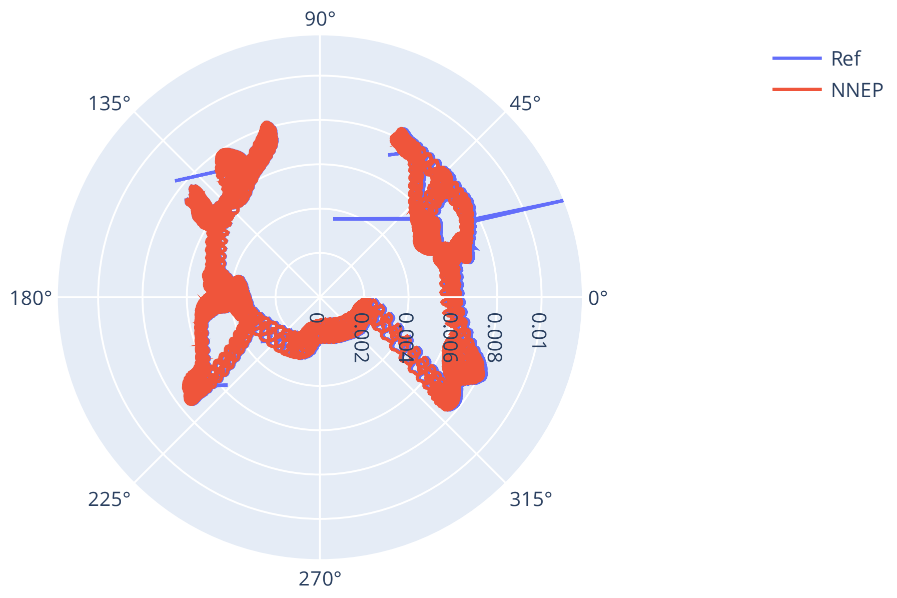
SK21


90°
135°
45°
180°
0°
225°
315°
270°
0
0.001
0.002
0.003
0.004
0.005
0.006
0.007
0.008
Ref
NNEP

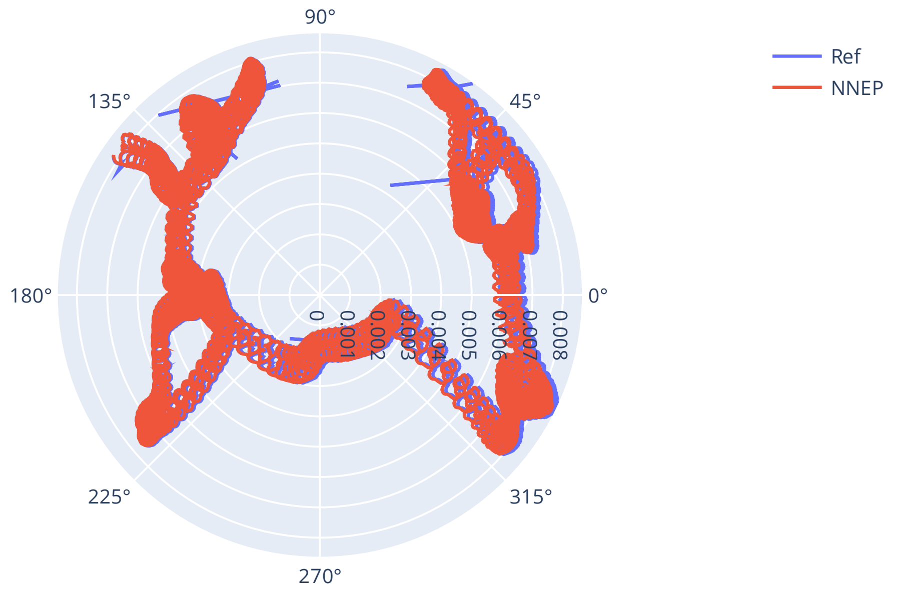
SK22


90°
45°
135°
180°
0°
225°
315°
270°
0
0.001
0.002
0.003
0.004
0.005
0.006
0.007
0.008
0.009
Ref
NNEP

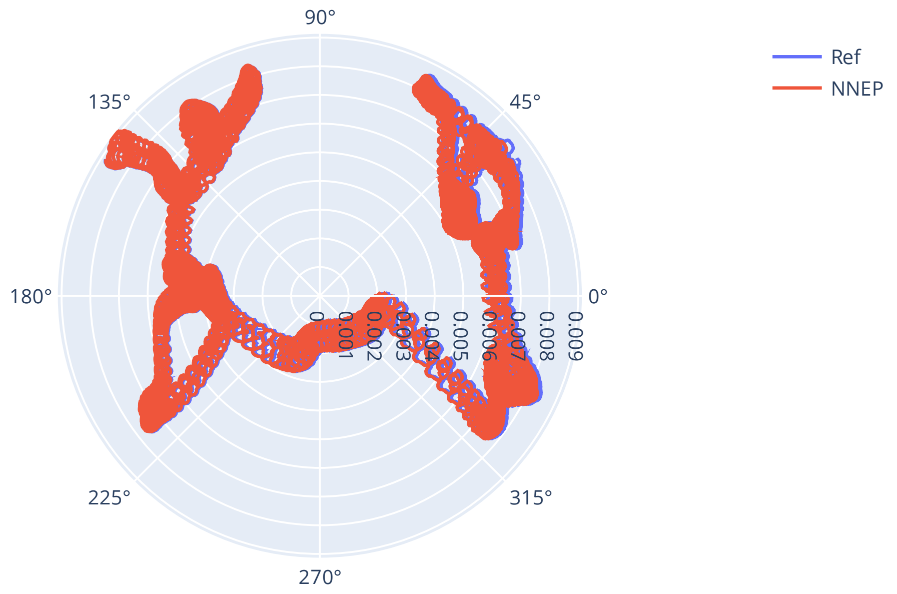
SK23


90°
45°
135°
180°
0°
225°
315°
270°
0
0.002
0.004
0.006
0.008
0.01
Ref
NNEP

SK24


90°
135°
45°
180°
0°
225°
315°
270°
0
0.001
0.002
0.003
0.004
0.005
0.006
0.007
0.008
Ref
NNEP

SK25


90°
135°
45°
180°
0°
225°
315°
270°
0
0.001
0.002
0.003
0.004
0.005
0.006
0.007
0.008
0.009
Ref
NNEP

Acknowledgments

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

References

[1] Parrish Ré, N., Sullivan, T. M., Popplewell, M. D., Roerig, K. S., Michael, C., Hanf, T., and Presser, T., "Neural Networks for Onboard Maneuver Design," *73rd International Astronautical Congress*, Paris, 2022.

[2] Quiza, R., and Davim, J. P., "Computational Methods and Optimization," *Machining of Hard Materials*, edited by J. P. Davim, Springer London, London, 2011, pp. 177–208. https://doi.org/10.1007/978-1-84996-450-0_6

[3] Cheetham, B., Gardner, T., and Forsman, A., "Cislunar Autonomous Positioning System Technology Operations and Navigation Experiment (CAPSTONE)," *ASCEND 2021*, Las Vegas, Nevada, 2021.

[4] Parrish, N. L. O., "Low Thrust Trajectory Optimization in Cislunar and Translunar Space," Ph.D. Dissertation, University of Colorado Boulder, 2018.

[5] Bezanson, J., Edelman, A., Karpinski, S., and Shah, V. B., "Julia: A Fresh Approach to Numerical Computing."

[6] Besard, T., Foket, C., and De Sutter, B., "Effective Extensible Programming: Unleashing Julia on GPUs," *IEEE Transactions on Parallel and Distributed Systems*, Vol. 30, No. 4, 2019, pp. 827–841. https://doi.org/10.1109/TPDS.2018.2872064

[7] Beckman, M., and Lamb, R., "Stationkeeping for the Lunar Reconnaissance Orbiter (LRO)," *20th International Symposium on Space Flight Dynamics*, Annapolis, MD, 2007.