Generating Exploration Mission-3 Trajectories to a 9:2 NRHO using Machine Learning

Esteban Guzman
NASA – JSC
Houston, Texas

December 2018
Since its founding, NASA has been dedicated to the advancement of aeronautics and space science. The NASA scientific and technical information (STI) program plays a key part in helping NASA maintain this important role.

The NASA STI program operates under the auspices of the Agency Chief Information Officer. It collects, organizes, provides for archiving, and disseminates NASA’s STI. The NASA STI program provides access to the NTRS Registered and its public interface, the NASA Technical Reports Server, thus providing one of the largest collections of aeronautical and space science STI in the world. Results are published in both non-NASA channels and by NASA in the NASA STI Report Series, which includes the following report types:

- **TECHNICAL PUBLICATION.** Reports of completed research or a major significant phase of research that present the results of NASA Programs and include extensive data or theoretical analysis. Includes compilations of significant scientific and technical data and information deemed to be of continuing reference value. NASA counterpart of peer-reviewed formal professional papers but has less stringent limitations on manuscript length and extent of graphic presentations.

- **TECHNICAL MEMORANDUM.** Scientific and technical findings that are preliminary or of specialized interest, e.g., quick release reports, working papers, and bibliographies that contain minimal annotation. Does not contain extensive analysis.

- **CONTRACTOR REPORT.** Scientific and technical findings by NASA-sponsored contractors and grantees.

- **CONFERENCE PUBLICATION.** Collected papers from scientific and technical conferences, symposia, seminars, or other meetings sponsored or co-sponsored by NASA.

- **SPECIAL PUBLICATION.** Scientific, technical, or historical information from NASA programs, projects, and missions, often concerned with subjects having substantial public interest.

- **TECHNICAL TRANSLATION.** English-language translations of foreign scientific and technical material pertinent to NASA’s mission.

Specialized services also include organizing and publishing research results, distributing specialized research announcements and feeds, providing information desk and personal search support, and enabling data exchange services.

For more information about the NASA STI program, see the following:

- Access the NASA STI program home page at [http://www.sti.nasa.gov](http://www.sti.nasa.gov)

- E-mail your question to help@sti.nasa.gov

- Phone the NASA STI Information Desk at 757-864-9658

- Write to: NASA STI Information Desk Mail Stop 148 NASA Langley Research Center Hampton, VA 23681-2199
GENERATING EXPLORATION MISSION-3 TRAJECTORIES TO A 9:2 NRHO USING MACHINE LEARNING

A Thesis

presented to

the Faculty of California Polytechnic State University,

San Luis Obispo

In Partial Fulfillment

of the Requirements for the Degree

Master of Science in Aerospace Engineering

by

Esteban Guzman

December 2018
COMMITTEE MEMBERSHIP

TITLE: Generating Exploration Mission-3 Trajectories to a 9:2 NRHO using Machine Learning

AUTHOR: Esteban Guzman

DATE SUBMITTED: December 2018

COMMITTEE CHAIR: Eric Mehiel, Ph.D.
Professor of Aerospace Engineering

COMMITTEE MEMBER: Kira Jorgensen Abercromby, Ph.D.
Professor of Aerospace Engineering

COMMITTEE MEMBER: Amelia Greig, Ph.D.
Professor of Aerospace Engineering

COMMITTEE MEMBER: Jeffrey P. Gutkowski, EG5 Deputy Branch Chief
NASA Johnson Space Center
Abstract

Generating Exploration Mission-3 Trajectories to a 9:2 NRHO using Machine Learning

Esteban Guzman

The purpose of this thesis is to design a machine learning algorithm platform that provides expanded knowledge of mission availability through a launch season by improving trajectory resolution and introducing launch mission forecasting. The specific scenario addressed in this paper is one in which data is provided for four deterministic translational maneuvers through a mission to a Near Rectilinear Halo Orbit (NRHO) with a 9:2 synodic frequency. Current launch availability knowledge under NASA’s Orion Orbit Performance Team is established by altering optimization variables associated to given reference launch epochs. This current method can be an abstract task and relies on an orbit analyst to structure a mission based off an established mission design methodology associated to the performance of Orion and NASA’s Space Launch System. Introducing a machine learning algorithm trained to construct mission scenarios within the feasible range of known trajectories reduces the required interaction of the orbit analyst by removing the needed step of optimizing the orbit to fit an expected translational response required of the spacecraft. In this study, k-Nearest Neighbor and Bayesian Linear Regression successfully predicted classical orbital elements for the launch windows observed. However both algorithms had limitations due to their approaches to model fitting. Training machine learning algorithms off of classical orbital elements introduced a repetitive approach to reconstructing mission segments for different arrival opportunities through the launch window and can prove to be a viable method of launch window scan generation for future missions.
ACKNOWLEDGMENTS

I would like to thank the following people for their guidance and support through the development of this thesis.

Jeff Gutkowski, for being an excellent mentor, helping me establish my knowledge in mission design development, and providing me with the opportunities to make an impact through the work he presented me.

Dr. Mehiel for your constant guidance through my post-graduate degree and for sharing your perspective on how to be successful in a professional environment.

The Orion Orbit Performance Team for helping me establish knowledge in mission considerations for the Orion vehicle.

Dave Lee, for providing me with launch window mission trajectories for EM-3.

Kelly Smith, for providing indispensable guidance regarding machine learning applications and how it can shape my thesis.

My friends and family for helping me maintain perspective and keeping me humble.

Maima, for your constant words of encouragement and support. Thanks for making me human.

Per Aspera, Ad Astra.
TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>List of Tables</td>
<td>ix</td>
</tr>
<tr>
<td>List of Figures</td>
<td>x</td>
</tr>
<tr>
<td>1 Introduction</td>
<td>1</td>
</tr>
<tr>
<td>1.1 Exploration Mission 3 Background</td>
<td>1</td>
</tr>
<tr>
<td>1.2 Launch Window Knowledge</td>
<td>3</td>
</tr>
<tr>
<td>1.2.1 Traditional Methods in Generating Launch Window Scans</td>
<td>3</td>
</tr>
<tr>
<td>1.2.2 Machine Learning as an Alternative to Generate Launch Window Scans</td>
<td>5</td>
</tr>
<tr>
<td>1.3 Outline of Thesis</td>
<td>6</td>
</tr>
<tr>
<td>2 Background and Literature Review</td>
<td>8</td>
</tr>
<tr>
<td>2.1 Current Numeric Methods used in Orbital Mechanics</td>
<td>8</td>
</tr>
<tr>
<td>2.1.1 Newton-Raphson Method</td>
<td>8</td>
</tr>
<tr>
<td>2.1.2 Extended Kalman Filter</td>
<td>9</td>
</tr>
<tr>
<td>2.1.3 SNOPTA</td>
<td>10</td>
</tr>
<tr>
<td>2.1.4 Orbital Trajectory Design and Optimization Platform</td>
<td>11</td>
</tr>
<tr>
<td>2.2 Applying Machine Learning in Orbital Mechanics</td>
<td>12</td>
</tr>
<tr>
<td>2.2.1 Orbit Determination in LEO</td>
<td>13</td>
</tr>
<tr>
<td>2.2.2 Modeling Circumbinary Orbit Stability</td>
<td>13</td>
</tr>
<tr>
<td>2.2.3 Modeling an Earth to Mars Trajectory</td>
<td>14</td>
</tr>
<tr>
<td>2.2.4 Modeling a Mission to an Asteroid Belt</td>
<td>15</td>
</tr>
<tr>
<td>2.3 Machine Learning Approaches Considered in this Study</td>
<td>15</td>
</tr>
<tr>
<td>2.3.1 k-Nearest Neighbor</td>
<td>16</td>
</tr>
<tr>
<td>2.3.2 Bayesian Linear Regression</td>
<td>19</td>
</tr>
<tr>
<td>2.4 Impact of Machine Learning in Mission Design</td>
<td>21</td>
</tr>
<tr>
<td>3 Approach</td>
<td>22</td>
</tr>
<tr>
<td>3.1 Mission Design Methodology</td>
<td>22</td>
</tr>
<tr>
<td>3.1.1 Force Models of the 9:2 NRHO</td>
<td>23</td>
</tr>
</tbody>
</table>
List of Tables

Table .......................... Page

3.1 NRHO Gravitational Parameters ........................................ 23
List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1 EM 3 Trajectory Overview</td>
<td>1</td>
</tr>
<tr>
<td>1.2 NRHO Comparisons</td>
<td>2</td>
</tr>
<tr>
<td>1.3 Traditional Method for Launch Window Scan Generation</td>
<td>4</td>
</tr>
<tr>
<td>1.4 Machine Learning in Launch Window Scan Generation</td>
<td>5</td>
</tr>
<tr>
<td>3.1 Family of Rendezvous with a 9:2 NRHO</td>
<td>22</td>
</tr>
<tr>
<td>3.2 COEs for Reference Launch Epochs during Pre-OPF</td>
<td>24</td>
</tr>
<tr>
<td>3.3 COEs for Reference Launch Epochs during NRHO</td>
<td>25</td>
</tr>
<tr>
<td>3.4 SOC Angles wrt to the VUW Frame</td>
<td>26</td>
</tr>
<tr>
<td>3.5 Linear Piecewise Boundary</td>
<td>30</td>
</tr>
<tr>
<td>3.6 Training Data by Segment Nodes vs Entire Mission</td>
<td>34</td>
</tr>
<tr>
<td>3.7 Argument of periapsis Before Pre-Processing</td>
<td>36</td>
</tr>
<tr>
<td>3.8 Argument of periapsis After Pre-Processing</td>
<td>37</td>
</tr>
<tr>
<td>3.9 RAAN Before Pre-Processing</td>
<td>38</td>
</tr>
<tr>
<td>3.10 RAAN After Pre-Processing: Unwrap</td>
<td>38</td>
</tr>
<tr>
<td>3.11 RAAN After Pre-Processing: Sine</td>
<td>39</td>
</tr>
<tr>
<td>3.12 Scaling Perifocal Distance</td>
<td>40</td>
</tr>
<tr>
<td>3.13 Trajectory Generation Flow Chart</td>
<td>42</td>
</tr>
<tr>
<td>4.1 k-NN Regression Fit for Perifocal Distance through Mission</td>
<td>45</td>
</tr>
<tr>
<td>4.2 k-NN Regression Fit for Perifocal Distance through Pre-OPF</td>
<td>46</td>
</tr>
<tr>
<td>4.3 Bayesian Regression Fit for Perifocal Distance through Mission</td>
<td>47</td>
</tr>
<tr>
<td>4.4 Bayesian Regression Fit for Argument of periapsis through Outbound Powered Flyby</td>
<td>48</td>
</tr>
<tr>
<td>4.5 Bayesian Regression Fit for Argument of periapsis through the Pre-OPF Coast</td>
<td>49</td>
</tr>
<tr>
<td>4.6 k-Nearest Neighbors Regression fit for Perifocal Distance through the Launch Window</td>
<td>52</td>
</tr>
<tr>
<td>4.7 k-Nearest Neighbors Regression fit for Perifocal Distance before and after the NRHO</td>
<td>53</td>
</tr>
<tr>
<td>B.3</td>
<td>Bayesian Regression Fit for Right Ascension through Mission</td>
</tr>
<tr>
<td>B.4</td>
<td>Bayesian Regression Fit for Argument of Perigee through Mission</td>
</tr>
<tr>
<td>B.5</td>
<td>Bayesian Regression Fit for Mean Anomaly through Mission</td>
</tr>
<tr>
<td>B.6</td>
<td>Bayesian Regression Fit for Epoch through Mission</td>
</tr>
<tr>
<td>B.7</td>
<td>Bayesian Regression Fit for MET through Mission</td>
</tr>
</tbody>
</table>
Nomenclature

Descriptive Mission Parameters

**COE** The oscillating classical orbital elements describing the state of a spacecraft at a given position during a mission

**DSG** Deep Space Gateway. NASA’s staging area for future human rated exploration missions - starting with EM-3

**IAUMoon** An inertial reference frame relative to the center of the moon

**JD** Julian Date. The number of days relative to noon UTC on January 1, 4713 BC. In this study, describes the time at launch for a reference trajectory.

**LEO** A low earth orbit is one in which a vehicle maintains an altitude between 400 and 1,000 miles about the earth’s surface

**MET** The mission elapsed time for a given trajectory as an ephemeris time

**NRHO** Near Rectilinear Halo Orbit. DSG’s trajectory about the moon

Translational Maneuvers

**NRD** A departure translational maneuver from the Near Rectilinear Halo Orbit

**NRI** An insertion translational maneuver to the Near Rectilinear Halo Orbit

**OPF** An outbound powered flyby aligning a translunar injection burn from earth with an entry orbit at the moon

**RPF** A return powered flyby aligning a return trajectory from the target trajectory back to earth
Numeric Evaluation Methods

*MSE*  Mean Square Error. A metric comparing a projected range of regression values versus the known span of training data for machine learning algorithms

*SNOPT*  Sparse Nonlinear Optimizer. A software package evaluating large-scale nonlinear optimization problems

Classical Orbital Elements

\( \mu \)  Gravitational parameter of the primary body

\( \Omega \)  Right ascension of the ascending node is a horizontal measure of the ellipse’s ascending node from the vernal point

\( \omega \)  Argument of periapsis is the orientation of the orbital ellipse in the orbital plane from the ascending node to the periapsis

\( a \)  The semimajor axis is the sum of apoapsis and periapsis divided by two

\( e \)  Eccentricity measures the elongation of an ellipse as compared to a reference circle

\( i \)  Inclination describes the tilt of a trajectory with respect to a reference frame

\( M \)  Mean anomaly describes the angle from the periapsis for the ellipse relative to a circular orbit with the same period

\( T_0 \)  The epoch of the trajectory at a given step through the mission as an ephemeris time
Chapter 1

INTRODUCTION

1.1 Exploration Mission 3 Background

Exploration mission 3 is the first step in establishing the Deep Space Gateway (DSG) in a 9:2 synodic resonance near rectilinear halo orbit. Figure 1.1 below illustrates key mission events between launch from earth, achieving the target trajectory, and return to earth. Near rectilinear halo orbits are quasi-stable in nature. As a result, station keeping is relatively cheap at 254 mm/s over the span of 500 revolutions for a 9:2 NRHO. These values and further reasoning for selection of the 9:2 NRHO as the target trajectory for DSG come from a study observing cis-lunar exploration trajectories in reference [1].

![Figure 1.1: EM 3 Trajectory Overview](image)

A 9:2 near rectilinear halo orbit is one in which nine NRHO orbits are completed over the span of two lunar months. At a 9:2 resonance, perilune for the 9:2 NRHO is 3200 kilometers from the center of the moon as illustrated in figure 1.2.
The target orbit was additionally selected to mitigate loss of signal between DSG and earth. For thermal considerations of the power systems on board, the 9:2 orbit was selected to reduce the risk of being in eclipse for too long with maximum totality experienced a maximum of 1.2 hours over the span of the 20 year study completed.

The stability of the NRHO over time and distance to the moon provides easy access to a low lunar orbit or for a descent to the lunar surface. This periodicity provides a region which can be predictably targeted during rendezvous opportunities for future missions to the DSG. The Orion Multi-Purpose Crew Vehicle (MPCV) will transport crew and payloads to the DSG. Existing mission data considers the performance of the MPCV as reflected in the required duration and pointing angles for finite burns. Since this is the designed method of reaching DSG’s trajectory from EM-3 onwards (EM-3+), finite burns to enter and rendezvous with DSG in future missions consider the existing MPCV performance parameters. By examining the mission trajectory elements that comprise an insertion into this target orbit, spawned trajectories from existing data provide knowledge on what maneuvers are possible should the nominal window slip and on what maneuvers can be achieved in future rendezvous attempts with DSG.
1.2 Launch Window Knowledge

Much like the mission itself, designing the trajectory profile for an exploration vehicle is comprised of satisfying a collection of objectives. Analysts at NASA Johnson Space Center’s Orion Performance Team employ a methodical approach to the design of mission trajectories accounting for mission objectives and vehicle constraints. Any number of adverse conditions during launch can postpone the nominal timeline of a target mission. To account for the inevitable variability that comes with finding the ideal opportunity to launch, mission design analysts generate trajectories at defined time intervals throughout the span of a launch season. Trajectory design comes as a collaborative effort between the analyst, flight operations, and vehicle designers to account for required mission constraints, vehicle performance parameters, and logistical timing.

1.2.1 Traditional Methods in Generating Launch Window Scans

With all mission ground rules and assumptions applied, mission analysts generate a trajectory for a given reference launch epoch in a three degrees of freedom (3 DOF) orbital optimization platform. The established trajectory has associated with it a given cost function be it for shortest mission time, least amount of fuel consumed, most delta-V, etc, depending on the study performed. Launch opportunities are spawned from this nominal mission trajectory by using a wrapper script which interfaces with an informational input deck defining the mission at different intervals. The wrapper script changes the launch epoch to the desired step through the season and modifies the provided orbital parameters, treated as optimization variables, to meet the desired cost function while retaining a desired mission design methodology. The collective data package reflecting these feasible launch opportunities comprises a scan through the launch window. An overview of the traditional procedures currently
used to generate launch window scans is highlighted in figure 1.3.

![Figure 1.3: Traditional Method for Launch Window Scan Generation](image)

The process to generate said scans is computationally demanding and, as a result, time-consuming with regards to the mission analyst’s availability. The means of scripting up the wrapper code to generate the scan can also be an abstract process to generate the combination of optimization variables to satisfy the constraints of the mission for the varied time step through the launch season. By the nature of the problem, mission trajectory design is no small feat yet there exists an opportunity to alleviate the workload required to generate launch season scans by looking into alternative methods, namely machine learning.

The baseline feasibility time history values describing launch opportunities to target trajectories were treated as a training set for machine learning algorithms in this study. Pattern recognition algorithms can then impose a regression fit based off observed trends in the existing time history files. Ultimately, feasible launch trajectories can then be generated in keeping with the expected values from the analyzed family of solutions. The resolution of feasible trajectories can then be improved from a 1.03505 day resolution to a launch opportunity every 3 hours thus improving the resolution for potential rendezvous opportunities to the desired trajectory. The new methods incorporating machine learning are reflected in figure 1.4.
1.2.2 Machine Learning as an Alternative to Generate Launch Window Scans

Machine learning is an area of study focused on automating a variety of tasks by training an algorithm to generate a desired output either by supervised or unsupervised methods. For the purposes of this study, machine learning has been employed in a pattern recognition capacity. As will be further discussed in section 2.2, machine learning can be used to fit existing trends when provided with knowledge of launch opportunities (attributes) and their related classical orbital elements (labels.) Knowledge of feasible alternative mission trajectories is then expanded by generating mission ephemeris times and COEs for different reference launch epochs. This meets the need of an orbital analyst on the Orion Performance Team who must generate high enough resolution in mission availability to support off-nominal alternative missions. For example in a scenario where weather conditions delay launch from the nominal time, generating mission trajectories through a given window provides knowledge regarding how to achieve the target trajectory for different launch times.

Machine learning algorithms do not make assumptions regarding relations between the provided attributes and labels. This provides a helpful functionality when observing unique classical orbital elements where the trends observed throughout a span of mission opportunities may not follow a single distinct pattern. In this way, machine learning provides a robust and repetitive approach to what was previously an arbitrary approach in generating launch mission scans.
1.3 Outline of Thesis

This thesis captures the design of machine learning algorithms incorporated in the generation of mission availability scans to a 9:2 NRHO. Machine learning can help alleviate the workload otherwise experienced by an orbital analyst and reduces the computational resources required when generating mission availability feasibility knowledge.

This thesis discusses the traditional numerical methods currently used when generating the orbital parameters that comprise mission trajectories. Giving an overview of machine learning will lay the ground work for effective methods in which machine learning has been incorporated through various industries. A survey of various applications of machine learning in an orbital mechanics perspective will further establish why machine learning is a viable candidate for tasks considered through mission design and help validate why the decisions to use the selected ML algorithms were made. Exact explanations are provided regarding the driving mathematical concepts, regions of validity, and computational performance of k-Nearest Neighbor and Bayesian Linear Regression. This thesis then covers how these two models performed in aiding mission design from a machine learning performance metric perspective.

The current mission design methodology used by NASA to define translational maneuvers to the moon will be discussed to give an understanding of the source data and the impact of generating trajectories using machine learning. Section 3.1 walks through what reference frames, suboptimal control pointing angles, classical orbital elements, and force model were considered in describing the desired mission trajectory and why. In section 3.2, discussion is provided on how trajectory data must be formatted when training and validating a machine learning algorithm; namely the required filtering, segregation, time-scale presentation, pre-processing steps, and degree-fitting steps completed when constructing a ML model. Machine learning in
this study has been incorporated as an augmenting support function which expands on existing knowledge. Section 3.2.5 covers the steps taken when constructing a code interface to classical orbital elements and tying back code considerations to mission requirements.

Chapter 4 covers the results observed for k-Nearest Neighbor and Bayesian Linear Regression. Both models have regions of reliable performance and limitations through the epochs observed. In observing the generated data, a discussion is provided on the expected behavior of the models versus the nature of pattern fitting methods employed through the known epoch. Additional discussion is provided in chapter 5 discussing how the constructed models performed in reference to expected machine learning metrics.

Though this study has provided valuable insight into potential methods of machine learning launch window scans, there exist areas for improvement. Section 6.1 discusses how the current methods can be improved through refined model fitting methods. Other machine learning models may better suit the problem posed in this thesis so a recommendation on potential candidate models is made in section 6.2. Section 6.3 discusses how the methodology of incorporating a pattern recognition approach can be further expanded to other missions outside of the mission observed in this study.

In closing, chapter 7 contains remarks on the effectiveness of the machine learning platform constructed, how it aligns with NASA mission requirements, and how it performs in the context of machine learning.
Chapter 2

BACKGROUND AND LITERATURE REVIEW

2.1 Current Numeric Methods used in Orbital Mechanics

Understanding the dynamic interactions of exploration vehicles in a celestial mechanics perspective relies on numerical analysis to achieve feasible trajectories. Numeric computations have aided the design of orbital trajectories throughout many different regions of astrodynamics studies. Provided with discrete observation data, numerical methods have been used in orbit determination providing analyst with a method of projecting the expected trajectory of a given spacecraft. Planetary stability analysis is further simulated using finite difference computations and cross comparative approaches with analytic data when considering the habitability of planets. Numeric methods support onboard guidance methods which require real time, low resource operations to understand a spacecraft’s orientation in orbit. Vehicle translation maneuvers are calculated with an associated cost function which reflects a performance parameter required to minimize a given metric through an interplanetary maneuver. For each of these scenarios, the orbital mechanics interaction is tailored to meet a mission specific operational requirement.

2.1.1 Newton-Raphson Method

The Newton-Raphson Method handles root finding by evaluating a line tangent to a function and extrapolating data to achieve an intersection with the x-axis. In application, Newton-Raphson has been successfully employed in initial orbital determination and has provided rapid convergence when observed satellite positions reflect the orbit well. Newton-Raphson provides a robust solution method capable of
determining an initial orbit through iterative refinement [2]. The Newton-Raphson method works best when the trend of the trajectory is provided with a good initial guess. As time shifts, the motion of the planetary objects considered begins to affect the feasibility of a given trajectory.

By shifting the launch epoch to the next step in the desired scan, the Newton Raphson method becomes limited by the initial guesses provided for a family of solutions constrained to launch geodetic parameters near the initial guess. When knowledge is desired on multiple steps through a launch window, initial guesses must account for the shift in launch geodetic parameters from the nominal launch epoch. For launch opportunities where initial launch parameters shift through a launch season, the initial guess posed is no longer a viable starting point and will then become computationally intensive.

2.1.2 Extended Kalman Filter

The Kalman filter is a recursive algorithm which employs a prediction and correction method in determining a desired state based off of a series of measurements with known uncertainty. Kalman filters do not require storage of prior state knowledge as the uncertainty of a range of measurements is reflected in an uncertainty matrix. Kalman filtering is common in guidance, navigation and control due to the ability to account for noise and measurement error inherent in state vector readings. The definition of the Kalman filter relies on understanding how reliable a given sensor is. Performance from a Kalman filter is additionally dependent on the complexity of the expected perturbations and dynamic models. The generated Kalman filter output can be further refined by establishing and tuning a covariance matrix over time. For a standard Kalman filter, the prediction step calculates a state transition matrix at the initial step whereas an Extended Kalman filter calculates the state transition matrix
at every time step.

In an effort to limit on-board memory storage associated to position and velocity, the author of [3] employed an Extended Kalman filter to accurately predict the attitude and orbital parameters of a spacecraft in a Molniya orbit. In this study, the author developed state matrices considering the perturbational forces from earth’s oblateness, the moon’s gravitational force and solar radiation pressure. As more measurements are collected, the error and processing time of the state decreases. The authors of the study used readings from GPS, magnetometers, earth sensors, sun sensors, and star trackers with varying degrees of accuracy. The study was successful in predicting the orbit based off of ground station tracking measurements of the satellite and sensor readings.

2.1.3 SNOPTA

The method used for the scan of missions on which this study is based uses SNOPTA [4], a Sequential Quadratic Programming system which supports optimization variables and convergence constraints for mission definitions. Under SNOPTA, the orbit is propagated at variable time steps and finds extrema, minimum or maximum depending on the objective. Searching for extrema in SNOPTA is done using sequential quadratic programming - a method of minimizing a quadratic sub-problem tied to the cost function. SNOPTA is well suited to handle the linear and non-linear constraints required for patch point convergence of the mission.

Python wrapper scripts used to generate launch window scans, as they currently exist, invoke SNOPTA by way of the Copernicus executable file, resolving the problem for each new reference launch epoch through the window. In line with satisfying the cost function through SNOPTA, optimization variables are modified in the script to ensure convergence of the trajectory to within a kilometer in position and a
centimeter per second in velocity. While using SNOPTA is necessary in establishing an initial trajectory, it is the goal of this paper to find a method which can alleviate the amount of processing required by using a less computationally intensive approach while retaining the constraints and rules for the trajectory.

2.1.4 Orbital Trajectory Design and Optimization Platform

The Orion Orbit Performance Team at NASA Johnson Space Center currently utilizes Copernicus, an orbit design and optimization platform when generating mission trajectories. Copernicus is a software platform with an entire catalog of targeting and optimization methods for minimizing cost functions, reflecting vehicle performance, and factoring in body-force models for planetary interactions. The exact decisions and constraints imposed when developing a mission per NASA standards is captured in Mission Design Methodology on page 22.

In Copernicus, users have the ability to select from a range of targeting and optimization solution methods. Different portions of a mission are defined in a segment, an informational data object retaining values for sequence duration, orientation with respect to a reference frame, classical orbital elements in that reference frame, mass properties of the vehicle, and any other relevant mission information. Convergence for a given segment is defined as falling within tolerance of either an existing state or a user defined range. In order to constrain missions to predictable arrival and departure opportunities, input decks are constructed to reflect forward and backward propagating segments associated to patch points to other segments where the need arises.
2.2 Applying Machine Learning in Orbital Mechanics

Machine learning has been around since 1959 where it was first used to train a computer the game of checkers [5]. In this study, a computer stored the rules for checkers and established weights for piece movement based off of an incentivized direction based off of board position, color, and opposing piece discrimination. The model trained between 8 and 10 hours and employed a winning strategy to beat the third ranked checker’s player in America at the time. Since then, machine learning has been incorporated in a variety of industries. In the auto industry, machine learning algorithms have been incorporated to train a car to maintain lane keeping, can predict when a car will need maintenance, and is used in hands off interaction via voice recognition services[6]. With access to large patient historic data, professionals in the medical industry have began research on incorporating machine learning to determine successful treatment and medication plans[7].

As it pertains to orbital mechanics, machine learning provides the service of generating classification and regression fits. Classification in an orbital mechanics perspective has been used to determine the type of orbit associated to repeating classical orbital elements. Regression has been applied in orbit determination, minimization of objective functions, and spacecraft orientation during maneuvers. Both methods have provided the utility of introducing robust, versatile solutions capable of handling a range of linear and non-linear regions of a trajectory while not encumbered by the burden of data bias. Where a model can fit, a model is applied which also doubles as an efficient method of computation limiting either the post-processing execution once the model is set up or the configuration of the model itself during training.
2.2.1 Orbit Determination in LEO

In a collaborative study performed between the University of Michigan Ann Arbor and NASA’s Jet Propulsion Laboratory [8], transfer learning methods were used in the orbital determination of a CubeSat in LEO. When knowing the parameters for a given spacecraft and after having established a model to fit the provided data, the learned model can be transferred over to fit a model for which prior data is provided. The study observed the Keplerian orbital elements, \([A \text{ (km)}, e, I \text{ (deg)}, \Omega \text{ (deg)}, \omega \text{ (deg)}, M \text{ (deg)}]\), as a feature vector for which a training model was constructed off of 3000 pass observations. The estimation was then generated for two passes.

The vector of COEs was determined using a simulated radio frequency (RF) signal for which a bias, band frequency, and sample rate were simulated by a software defined radio system. The model then serves as a classification model which can be utilized in regression and to determine sources of noise. The model constructed in the study was successful at estimating the orbital parameters despite variance in provided data. Machine learning was also effectively used to identify and classify the orbits of specific spacecraft transmissions.

2.2.2 Modeling Circumbinary Orbit Stability

In a classification sense, machine learning can be utilized to capture the stability behavior for orbiting planets described as being in Circumbinary orbits[9]. Circumbinary orbits have the complication of being on the cusp of instability per parametric relations. The approach presented in this paper provided valid regression estimates in confirming whether a given system was stable while not generating false positives of instability associated to a polynomial fit with known regions of instability.

The target problem was structured as analyzing systems initially coplanar
with circular orbits. The resulting deep neural network provided better performance in accuracy, precision and memory recall when compared against the existing method of stability calculations.

2.2.3 Modeling an Earth to Mars Trajectory

A nominal mission from Earth to Mars was treated as the baseline trajectory in [10]. The authors of the study noted how research into machine learning applications for orbital trajectories has been limited due to the lack of available data provided by the aerospace industry. The trajectory was initially optimized using the quadratic optimization control approach and then minimized prop mass consumed by running through as a mass optimizing scheme. The data set was expanded by perturbing orbital parameters in the nominal trajectory. From the data generated, an optimal state feedback deep network was constructed generating a relation for thrust vectoring in polar coordinates throughout the mission.

In neural networks, activation functions must be selected to incorporate weights based off of displacements from an epoch perspective. Input and output values were both processed through activation functions. Machine learning techniques can provide a better output when the input is initially normalized as was done in this study. The number of layers in a neural network define what weights and biases are attributed to a signal ultimately resulting in what will be sent into the activation function. The more refined the number of layers in a model, the higher the fidelity of the model.

When selecting the number of layers in a neural network, there is a trade-off between the accuracy of the generated parameter and the run-time of the neural network. The authors of the paper decided that the returns on accuracy for the model stopped providing added benefit after the mean square error associated to an
extra layer was within $10^{-4}$ of an MSE from a previous layer. The resulting neural network model utilized multiple network approaches in generating the different guidance parameters. It was found that individual parameters had better performance by incorporating unique neural network activation functions. Overall, the neural network system successfully constructed mission guidance parameter projections within a reasonable error.

2.2.4 Modeling a Mission to an Asteroid Belt

In a study performed as part of a orbit optimization competition [11], the European Space Agency constructed a machine learning algorithm to determine the ideal opportunity to rendezvous to an asteroid based off of a phasing hyperparameter. The intention of this study was to determine when a combination of asteroids produced a combined position ideal for rendezvous with multiple asteroids at a given time. To do this, a k-Nearest Neighbor algorithm was employed to calculate a euclidean phase indicator. The phase indicator reflected the state vectors for a cluster of asteroids at a given time thus defining the indicator in a 6-dimensional space.

Incorporating the indicator calculated by k-Nearest Neighbor provided a useful metric for reflecting how successful a given rendezvous opportunity was at fulfilling multiple objectives for a mission through use of a decision tree.

2.3 Machine Learning Approaches Considered in this Study

The scope of this study focuses on applying machine learning to improve understanding on launch window availability by spawning missions off of existing trajectory data through a launch season scan. In order to do so, the machine learning algorithms considered should project inferential data based off of sparse data provided for multiple arrival opportunities at the NRHO for varied launch trajectories.
The two algorithms observed in this study achieve inferential projections in similar but unique ways. Both k-Nearest Neighbors and Bayesian Linear Regression are supervised learning methods which require analysis beyond initial exposure to the data to provide the ideal fit of an inferred guess not in the existing data set. Where they differ is how this is achieved. k-Nearest Neighbor establishes a model fit based off of weighted contributions from nearby label/attribute combinations and serves as a fit to the data. Bayesian Linear Regression projects the data set into a Gaussian probabilistic function and fits the data to a function based off of likelihood of recurrence from data analysis. Both were successful in generating orbital parameters for future mission availability knowledge within certain regions of epochs.

2.3.1 k-Nearest Neighbor

Overview of k-Nearest Neighbor Regression

k-Nearest Neighbor is a machine learning algorithm which produces a regression function estimate based off of the k nearest elements to a given epoch and label combination [12]. For a set of data \((X_1, Y_1)\ldots(X_n, Y_n)\), the default estimate of a given point is generated from the regression function \(m_n\):

\[
m_n = \frac{1}{k_n} \sum_{i=1}^{k_n} Y_{(i,n)}(x)
\]

where \(k_n\) reflects the impact of the k nearest elements on the estimated value of a desired unknown epoch. During evaluation, distance for a given label is relative to the specific epoch under consideration. The fraction \(\frac{1}{k_n}\) reflects the default weight as an inverse proportionality between the surrounding labels for a given set of attributes.
Search Radius

The k-Nearest Neighbor algorithm utilizes a radial search pattern to either provide classification or regression of an unknown epoch inferential from a known data set. The search radius imposed in this study is euclidean. A euclidean radius has a uniform linear reference path and can be applied in this study due to the uniform resolution available in the reference training data. The radius used in k-Nearest Neighbor can also be constrained to a defined function when influence disbursement may not follow a linear trend. Since evaluation of the orbital parameters is generated at the time step resolution, the euclidean search radius is considered in a two-dimensional space and is defined by the following formula:

\[
distance = \sqrt{(epoch_{ref} - epoch_{target})^2 + (parameter_{ref} - parameter_{target})^2}
\]

The neighbors with the lowest evaluation of this distance are then considered when establishing weights during an inference.

Support for Different Numerical Systems

k-Nearest Neighbor is a non-parametric machine learning algorithm which means it does not make assumptions regarding the relation between an attribute and the associated label. Because of this, k-Nearest Neighbors reflects the non-linear relations that are inherent in orbital mechanics while still maintaining the effects of body forces, perturbations, signal errors, etc. Knowledge of the distribution is determined upon each evaluation since k-Nearest Neighbors produces instance based evaluations. The regression generated then fits the distribution of data observed at each step providing support for changes in the patterns observed in future time steps.
This instance based evaluation is particularly beneficial when considering the data set provided for EM-3 where values of the classical orbital element sets tend to have a larger span of differences earlier in the mission and converge to a constant range after arriving at the NRHO as is observed in figures 3.2 and 3.3.

**Computational Performance Considerations**

k-Nearest Neighbor is a supervised learning scheme which requires feedback to generate the best possible fit for the spread of data. Selecting too few or too many neighbors when constructing the regression can result in a model which is respectively under or over fitting the data. A model which inappropriately fits the data may capture the training points effectively but would do poorly in generating a guess in the testing or validation sets. A valuable metric in considering how well a model is performing is the misclassification error (MSE) which reflects how well selection of the number of neighbors, k, performs in capturing the trend of the data. MSE then captures the trend for how many points are valuable in constructing a model. For an understanding of what MSE looked like when reconstructing COEs in this study, see figure

**Limitations of k-Nearest Neighbor**

Due to the instance based evaluation employed, the regression fit generated from k-Nearest Neighbor provides a fit specific to the data observed. As such, k-Nearest Neighbor does not perform well in projecting attributes for points outside of the span of epochs on which knowledge is established. Instead, k-Nearest neighbor will fit the guess generated to its nearest points which can erroneously capture a trend not in keeping with the expected pattern. Because of this, regressions formed by k-Nearest Neighbors are only reliable through the known span of epochs for which
labels are provided. k-Nearest Neighbor has a gap in performance when considering forecasting of trajectories as a result.

2.3.2 Bayesian Linear Regression

With k-Nearest Neighbor lacking in performance outside of the known range of epochs, it was of interest, in this study, to find a model which had the ability to adequately infer posterior trends from a sparse prior information set. Bayesian regression defines predictive functions by projecting data into a multivariate Gaussian distribution [13]. Fitting the data to a multivariate distribution then allows for the calculation of a probability disbursement for a distribution of attributes within the known and unknown data set. In practice, this relation provides an inferential relation to project some label \((B)\) provided a set of attributes \((D)\) as a function of the attributes and some vector of adjustable parameters in the model \((w)\): \(P(B|D) = f(D, w)\)

**Gaussian Probabilistic Inferential Projection**

As a Gaussian distribution, the relation between the proportionality to the probability of a data set given a \(w\), and the prior probability distribution of \(w\), \((p(w))\), a predictive projection function can be defined. This inferential projection is capable of generating new values outside of the known range of data epochs. This posterior probability is defined as:

\[
p(t|\tilde{t}, \alpha, \beta) = \int p(t|w, \beta)p(w|\tilde{t}, \alpha, \beta)dw
\]

where \(\tilde{t}\) is the set of training points from the existing data, \(p(t|w, \beta)\) is the conditional distribution, and \(p(w|\tilde{t}, \alpha, \beta)\) is the posterior distribution of the model parameters given the training points.
A helpful metric in minimizing error in the model is considering the maximum a posteriori distribution (MAP). The MAP is the mode of the posterior distribution and is established off of prior knowledge from the provided data. Given that the Bayesian regression model incorporates a maximum a posteriori estimate, the model avoids overfitting to specific data by instead matching the likelihood of an attribute given a distribution. Calculating the MAP establishes a point inference which trends towards the maximum likelihood value. The maximum likelihood value is associated to the minimum error since error is the negative log of likelihood. By the design of Bayesian inference, the projected value minimizes error while accounting for likelihood of a label given an attribute.

Because the model of projection used in Bayesian regression imposes a fit to a Gaussian distribution, the Bayesian regression algorithm also calculates a standard deviation. This standard deviation also serves as a measure of precision for the fit along the data set based off of the degree of trust in the model.

**Generalized Model Fit**

Due to the generic Gaussian fitting employed, the projected model fits the trend of the values by design. This may lead to value discrepancies at certain epochs when projecting values off of the trained model. The projections generated by BRR have proven effective in capturing the expected trend from the sparse classical orbital element set on which the model is trained.

**Computational Demand of BRR**

Initial inference of the model can be time consuming at first as BRR attempts to fit a data set to a Gaussian distribution which may not be immediately discernible. After training the model to fit Bayesian ridge, retaining the model parameters can
make for quicker evaluation in future runs. Imposing a Gaussian distribution on a sparse data set provides a generic trend which would benefit from higher precision if provided more data points. With higher data availability, future analysis would require the trade-off between available precision and runtime performance as was discussed in 2.3.1

2.4 Impact of Machine Learning in Mission Design

This thesis is structured around constructing an algorithm which will generate mission trajectories through an NRHO. Orbital parameters observed at a time step resolution reflected a periodic behavior throughout different portions of the missions indicating compatibility with a pattern recognition machine learning algorithm. Using supervised learning methods, a model fit can be generated to provide a regression evaluation unique to each orbital parameter at each time step specific to an arrival opportunity at the DSG’s NRHO trajectory. The generated orbital parameters are then compiled and comprise a mission trajectory, providing mission trajectory information for a launch epoch not previously present in the data set.

k-Nearest Neighbor and Bayesian Linear Regression were both selected because of their support in fitting models when sparse data is provided. Both k-Nearest Neighbors and BLR are used in projecting orbital parameters for reference launch epochs provided to the code. Existing launch window knowledge is provided at a 1.03505 day interval however inferential models for launch epochs in between these regions is desired to understand off-nominal mission opportunities. Performance for the two machine learning approaches are then evaluated against common machine learning metrics with discussion on reliability given the expected evaluation regions.
Chapter 3

APPROACH

3.1 Mission Design Methodology

Training the machine learning algorithm will focus on mission segments associated to the insertion, stay and departure to and from an NRHO closest to the moon within a body fixed-moon centered inertial reference frame. These portions of the mission can be seen in figure 1.1 as the region enclosed by the red dashed line. An orbit is considered thoroughly defined when the classical orbital elements, mission ephemeris time, and delta-V maneuvers required to achieve a desired trajectory are defined. Mission availability knowledge is directly improved by machine learning algorithms when structured together to describe reference launch epochs. The generated trajectories achieve an arrival at an NRHO within the family of known opportunities in the existing data. Different launch opportunities comprising a family of rendezvous trajectories to the NRHO are illustrated in figure 3.1 below.

![Figure 3.1: Family of Rendezvous with a 9:2 NRHO](image)

Figure 3.1: Family of Rendezvous with a 9:2 NRHO
3.1.1 Force Models of the 9:2 NRHO

Dynamics about the moon have been simulated as a circular restricted three body problem in a moon-centered earth-moon rotating frame. The planetary bodies considered in the design of this trajectory are the earth, moon, and sun in line with the DE421 ephemeris. DE 421 contains a fully converged solution and comes recommended as the ephemeris to use for lunar missions [14]. Gravity about the moon is represented using GRAIL spherical harmonic model at degree and order 8. The gravitational parameters for the earth and sun are represented as point masses captured below in table 3.1

<table>
<thead>
<tr>
<th>Body</th>
<th>Gravitational Parameter ((km^3/s^2))</th>
</tr>
</thead>
<tbody>
<tr>
<td>Earth</td>
<td>398600.436233</td>
</tr>
<tr>
<td>Moon</td>
<td>4902.800076</td>
</tr>
<tr>
<td>Sun</td>
<td>132712440040.944</td>
</tr>
</tbody>
</table>

Table 3.1: NRHO Gravitational Parameters

3.1.2 Orbital Parameters in Orion Body Fixed-Moon Centered Inertial Frame

Similar to what the authors of [11] saw when attempting to rendezvous with asteroids from an ephemeris set, the orbital parameters describing the trajectory to an NRHO maintain a periodic pattern. The classical orbital elements considered in the mapping of Orion at a given point in an EM-3+ trajectory are perifocal distance, eccentricity, inclination, right ascension of the ascending node, argument of periapsis, mean anomaly, the epoch at the point in the mission, and the gravitational parameter of the primary planetary body. These elements along with the mission elapsed time as an ephemeris time are required when propagating a trajectory through an orbits software package.

Coupling the quasi-stability of the NRHO, and establishing a desired rendezvous phase region as described in 1.1 results in a repeating pattern for all orbital
parameters observed through a given launch season. Through the entirety of the mission trajectory, the repeating patterns observed in the classical orbital elements are associated to families of arrival epochs at the NRHO. A family of arrivals comprises multiple launch opportunities that arrive at a specific rendezvous opportunity with the NRHO at its perilune.

When converged in Copernicus, mission segments utilize different reference frames through the various translational maneuvers. For consistency, the orbital parameters for a given trajectory were extracted from a SPICE kernel generated in Copernicus. A SPICE kernel is a trajectory data file following the format established by NASA’s Jet Propulsion Laboratory. The classical orbital elements were in a body-fixed inertial frame relative to the center of the moon (IAU MOON). In this reference frame, orbital parameters maintained discernible trends compatible with a pattern recognition algorithm. For a visual representation of what this looks like, reference figure 3.2 where the converged data set obtained from Copernicus is plotted against different launch epochs for the Pre-OPF maneuver. Notice that each classical orbital element sets have unique patterns associated to different launch opportunities yet they all follow some distinguishable pattern.

Figure 3.2: COEs for Reference Launch Epochs during Pre-OPF
Additionally, the classical orbital element through the stay at the NRHO, reflected in figure 3.3, maintain their own unique periodic patterns. With the COEs trending towards values related to arrival opportunities at the NRHO, it is more apparent here that the orbital parameters are converging to a fixed phase entry to the NRHO as required for a rendezvous with DSG.

![Figure 3.3: COEs for Reference Launch Epochs during NRHO](image)

### 3.1.3 Suboptimal Control Thrust Parameters

Translational maneuvers were modeled in Copernicus using sub-optimal control in the VUW reference frame relative to Orion’s velocity vector. For insertion and departure burns, it is of interest to capture right ascension ($\alpha$) and declination ($\beta$) to describe the in plane and out of plane angles for a translational maneuver. Right ascension and declination, illustrated in figure 3.4, describe the required orientation of the MPCV during translational maneuvers for insertion into the fixed phase rendezvous with DSG.
Angle values for each of the major translational burns are captured in the generated time history files and the machine learning algorithm trains off of the provided data in the same way a model is constructed for the orbital parameters.

3.1.4 Key Mission Events

In the design of a mission to the NRHO, NASA has maintained a methodical approach to reach the target orbit comprised of four key translational maneuvers around the moon[1]. These translational maneuvers are periods during the mission where finite burns are necessary to achieve rendezvous with the DSG orbit. The desired pattern is as follows:

- **OPF** an outbound powered flyby following a translunar injection maneuver from earth to align with the moon
- **NRI** a near rectilinear insertion burn to arrive at the NRHO
- **NRD** a near rectilinear departure burn to leave the NRHO
- **RPF** a return powered flyby to return to earth
Arrival and departure opportunities are made uniform when attempting to reach DSG. From an orbital mechanics perspective, it is of interest to have a predictably defined region where an arrival and departure can be initiated. In the context of mission design, this is considered a fixed insertion phase where the position of the orbit is in line with a rendezvous opportunity with the DSG.

In the interest of capturing this rendezvous as a converged, continuous trajectory, the coast states between the executed finite burns are modeled as optimization variables resulting in patch points. At these patch points, position must converge for two mission segments within one kilometer for position and one cm/s for velocity. Convergence to an angle has a tolerance of +/- 0.01 degrees. The following coast segments exist throughout the span of the mission to aid in patch point convergence:

- **Pre-OPF** a coast period aligning with a translunar injection maneuver
- **Post-OPF to Pre-NRI** a patch point aligning the coast state after outbound powered flyby coast state to the coast state before near rectilinear insertion finite burn
- **NRHO Stay** the stay period at the 9:2 near rectilinear halo orbit
- **Post-NRD to Pre-RPF** a patch point aligning the coast state after near rectilinear departure to the coast state before return powered flyby

A trajectory is comprised of the translational maneuvers (4) and the coast periods (6) resulting in ten segments describing a continuous trajectory to and from the NRHO. Each segment has enough nodes to reflect the position through the mission at an average of 15 second intervals.
3.1.5 Existing Mission Availability Scan

The existing mission data available for EM-3 has a baseline NRHO stay of about 5 days. Missions have been generated for a range of dates between October 2025 and November 2026 resulting in 511 unique reference launch epoch trajectories. These mission trajectories have associated with them 57 unique NRHO arrival opportunities. For a given range of reference launch epochs, a family of solutions is defined as any span of trajectories which arrive at the NRHO on a given day. Each of the trajectories for a given arrival family have associated orbital parameters describing the dynamics of Orion through an EM-3+ type mission.

3.2 Incorporating ML Algorithms for Mission Trajectory Generation

Although machine learning methods can be viable sources for recognizing patterns in data sets, the effectiveness of the model relies on how the attributes and labels are reflected numerically. For most parameters through the duration of a mission, there is little work that needs to be done to capture and adequate periodic pattern. Due to the quasi-stability of the NRHO, unique families of arrival opportunities maintain periodicity locally. Fitting a model to the entire launch season would not adequately reflect these local patterns and would instead overfit a model through the season. Overfitting can be avoided by only considering local launch epochs which maintain the expected trend during training. Trajectories throughout the scan exist where discontinuities or scaling must be resolved when training a model. The data can be pre-processed ensuring that the attribute and label relation is accurately reflected while providing a format more compatible with the algorithm considered.
3.2.1 Filtering Data Using Piecewise Linear Boundaries

When multiple families of solutions are present, a model fit must be restrained to within a defined region using linear piecewise boundaries. Segregating data in this manner provides the benefit of generating regressions based off of relevant family members associated to a given distribution as reflected in figure 3.5. When used in k-Nearest Neighbor and Bayesian Linear Regression, linear piecewise boundaries confine the influence of the data to a defined region and will not erroneously construct a trend of labels as continuous if there are multiple families of arrivals present.
Both families reflect convergence to unique trends for right ascension through the coast to outbound powered flyby. Restraining the fit to the piecewise linear boundaries provides two unique regressions avoiding potential inaccurate fit of attributes between the region. A reference launch epoch which can align with two separate arrival opportunities must treat these scenarios separately since the local pattern is associated to the quasi-stability of the NRHO.
3.2.2 Filtering Mission Data for Machine Learning

A machine learning algorithm is only as reliable as the data it is provided because the model fit estimations treat the data provided as absolute truth; the attribute and label combination will always be true for any other observations. With a sparse data set, the impact of removing an attribute and label combination from the expected data can be significant since a model should retain as many points as possible and not erroneously negate what few data points exist. Filtering data requires a trade off between establishing an initial model fit while retaining enough data points to provide intermediate and final validation of a model. Data is filtered into three different groups when constructing a model: training, validation, and testing.

The weights and biases the machine learning algorithms produces to fit the presented data to a given model are developed during the training step. As is seen in k-Nearest Neighbors, the model directly captures the pattern of the data provided considering the impact of other data points most closely associated to a desired epoch. In Bayesian Linear Regression however, the model seeks to reflect the likelihood of a value given prior knowledge for the entire distribution. In either case, the fidelity of the generated model is directly correlated to the number of training points on which the algorithm is trained.

The validation step considers just how well the produced fit meets the expected trend of the data provided. Since validation data is not present during the training step, comparing the attribute and label combinations generated by the model for a given epoch versus the true value provides an unbiased comparison of just how well the data is performing. The performance of the model can be tuned by observing the performance through the validation step. Validation then helps a machine learning designer understand how the number of training points used affect the hyperparameters generated by the model without helping the model "learn."
The testing data set is a final evaluation of the fit produced by the model. Ideally, the fit should be finely tuned enough to generate an accurate estimate for the test epochs which would have not been used to establish or refine the model up to this point. If instead evaluation of the test epochs results in a discrepancy between the generated value and the truth, the model is considered to have fitting issues (see 3.2.2). Learning performance metrics generated during the testing phase describe how different hyperparameter configurations for a given model reflect the expected pattern through analysis of resulting bias and variances. Once the model meets an expected tolerance through the testing phase, it is considered ready for production and can be used to reliably infer epochs not previously present in the observed data.

Filtering Trade-Offs to Consider

An issue to keep in mind when developing a machine learning algorithm is the computational efficiency of the algorithm. After a certain amount of training data, no extra knowledge is garnered by observing extra attribute-label combinations. With no further precision generated, the machine learning algorithm can be over-encumbered in label generation for no added benefit.

Machine learning algorithms can additionally suffer from ineffective fitting when employed in pattern recognition. With too few points, the real trend of the data provided is not accurately captured. Too many points, on the other hand, can result in overfitting of a model. A by-product of machine learning, overfitting is observed where the produced model specifically captures the label of each attribute for each epoch of the training data. Overfitting comes at the cost of losing generality of the trend and thus losing the capacity to accurately infer an epoch outside of the data set provided.

Mean square error (MSE) reflects the trade-off between the average error
points for labels generated by the ML model and serves as a useful metric when
determining how many data points to consider when constructing an algorithm. MSE
helps the ML algorithm designer avoid overfitting by considering the bias and variance
of the generated model. Overfitting is reflected in MSE as a divergence away from an
otherwise converged error in bias. Selecting the number of training points to avoid
inefficient processing can be done by determining when the MSE figure has converged
to a given tolerance after which the model will no longer produce additional precision.

3.2.3 Time Scale Considerations during Training Step

An orbit generated using machine learning would need to follow the mission
trajectory trends exhibited in the existing data. Focus must be taken to capture the
scale of orbital parameters for a given segment instead of imposing comparative scales
throughout the entire mission trajectory. With this in mind, a time node segment
approach would provide a more effective means of replicating the mission data.

Considering a model fit at the time-step node level provides a relative com-
parison of reference epochs associated to a position of a mission parallel to the position
in the desired epoch. Weights and offsets generated by the function fit provided by the
machine learning approach would be associated to a given time-step node rather than
establishing the weights and offsets for the entire mission. In figure 3.6 below, this is
illustrated where the zoomed in segment has a more distinct shape than would have
otherwise been captured by the machine learning algorithm if scaling was established
for the entire trajectory.
3.2.4 Preprocessing Data

A critical step in establishing a machine learning algorithm is presenting the data set in a format that provides an easily discernible pattern. Prior to training an algorithm, the data must be scrutinized for numeric discontinuities. Deviance of a given epoch’s attribute and label combination can reflect noise present in the analyzed signal. This study considers simulation of a given trajectory for which no signal noise exists. Data biases, such as angle phase offset, in this scenario are a byproduct of numeric processing. Without first pre-processing the data, a fit would be imposed accounting for larger differences that exist numerically but not geometrically.

Pre-processing transforms data by removing discrepancies while retaining the base geometric representation of the state captured by the attribute and label combination. Additionally, some methods of machine learning benefit by training off of regions of data standardized to within a unity scale. In so doing, the machine learning algorithm is more effective in capturing what can otherwise be a subtle
difference when comparing a span of epoch data. Specific to this study, pre-processing was employed in reshaping the classical orbital elements for compatibility with k-Nearest Neighbor and Bayesian Linear Regression. For certain angle parameters throughout the launch window, there existed angle discontinuities which required further attention to be properly reflected by a model fit. Additionally, to better support the machine learning algorithms, the orbital parameters were standardized between 0 and 1.

**Angle Phase Considerations**

There exist phase offsets when observing some angular parameters in the classical orbital element set. While, geometrically, the values reflect the same angular displacement with respect to a given reference frame, the phase offset introduces a periodic offset. Such a phase angle shift is visible in figure 3.7 where the argument of periapsis is portrayed for a family of NRHO insertion opportunities.

Training a machine learning algorithm off of the data as it is currently reflected would introduce a model fit to asymptotic trends. Generating an estimate from such a model introduces potential inconsistent estimates where certain time steps may reflect a deceptively large magnitude not in keeping with local epochs. The values must be processed through either unwrapping or by calculating the sine of the angle.
In the case for argument of periapsis, the machine learning algorithm would provide a more reliable output if it did not have to account for observable asymptotes in the data. Instead, unwrapping the angle from what would have otherwise been a value contained between 0 and $2\pi$ provides a smoother transition region among the different segments of the mission as observed in figure 3.8.
An estimate for an epoch within this trained region reflects the same angle displacement while following the expected trend when compared against local expected arrival COEs. Unwrapping the angle has proven effective when constructing models for argument of periapsis, and inclination. Another orbital parameter which needs pre-processing is right ascension of the ascending node (RAAN) as seen in figure 3.9. For RAAN however, unwrapping the angle does not remove all discontinuities, as observed in figure 3.10. Instead, unwrapping the angle for RAAN introduces a different type of discontinuity.
Rather than unwrapping the angle in the case of right ascension, the angle was pre-processed by calculating the sine of the angle, as is observed in figure 3.11. A model was then fit to the periodic value observed in the sine of RAAN, which maintained a semi-stable pattern through the NRHO. Generating the estimated value after training for the true value of right ascension was done by taking the inverse sine to revert back to an angle value.
Machine learning models can produce a better fit when the model is trained off of standardized data. Standardizing the data means transforming the incoming data between zero and one. In calculations, each parameter in a family of arrival opportunities was standardized by subtracting the minimum point observed in the time step, shifting the bias to zero. Next, all values were divided by the maximum value in the data set. An example of this is shown in figure 3.12 which shows perifocal distance, previously between approximately 180 to 375 kilometers, scaled between 0 and 1.

Figure 3.11: RAAN After Pre-Processing: Sine

Scaling Time Step Values

...
By accounting for the weights and biases in the data, all labels fell within zero and one. The generated estimate from the model was then multiplied by the weight and added to the bias to regain the true parameter value of the calculation for each orbital element.

### 3.2.5 Computational Trajectory Generation

In this study, a mission generating script was developed using the Python \[15\] language. Orion’s feasible trajectories to and from the NRHO in EM-3 for an entire launch season were generated using SNOPTA in Copernicus. A python wrapper script then extracted the trajectories to a binary SPICE kernel file (BSP) which held trajectories in a spice kernel compatible format. Classical orbital elements were then generated by using the position data in the BSP files for the desired frame using SpicyPy, a spice kernel module for python.
The COEs were generated at a defined resolution for each of the segments described in section 3.1.4. Generating the COEs with a fixed resolution ensured that each time step was relative to a fixed position in the mission for the different families of arrival opportunities considered when constructing a model fit. For example the $n^{th}$ element in a time history file would always be in the pre-outbound powered flyby segment of each launch opportunity through the season. The orbital elements describing were then stored in unique comma-separated value (CSV) time history files. The python script interfaced with the existing time history files by extracting classical orbital element parameters at each time step.

Piecewise linear boundaries define arrival families in the python script by extracting the arrival date at the NRHO from all of the trajectories in the launch season. Running the python script from a command prompt, the user is asked to put in a target reference epoch. Upon receiving the target epoch within the defined range, the algorithm determines which family of launch epochs the provided epoch can be compared against. There are trajectories associated to reference launch days on which a given mission can arrive at an NRHO for two separate opportunities. When facing such an opportunity, the machine learning algorithm generates both feasible trajectories and leave decision making regarding the desired NRHO insertion day to the discretion of the orbital analyst.

Once the relevant boundaries are determined for a target epoch, the python script generates a model fit for the family of arrival opportunities at each time step using machine learning algorithms. k-Nearest Neighbor and Bayesian Linear Regression were both incorporated using the Scikit-learn package[16] in Python. The configuration parameters for k-Nearest Neighbors and Bayesian Linear Regression were determined based off the sparsity of the data set provided and effective range of where the selected models were expected to project feasible values. The configuration parameters which define an algorithm were decided on during the validation
and testing phase along with the filtering trade-off considerations discussed in section 3.2.2. The script then compiles all classical orbital elements for the target launch epoch into a vector describing the trend through the mission trajectory. Figure 3.13 illustrates the steps described in this section as a flow chart.

![Figure 3.13: Trajectory Generation Flow Chart](image-url)
Machine learning was incorporated in this study with the intention of expanding mission availability knowledge through a launch season. A model shaped around machine learning introduces a tool capable of refining knowledge on classical orbital element parameters. This machine learning tool can refine Orion mission trajectory knowledge for an entire launch season for which mission data exists at a reference epoch resolution every 1.03505 days. Additionally, understanding what maneuvers and orbital parameter combinations defining mission trajectories outside of a given span of launch data provides a method of launch season forecasting. The reliability of the models depend on the capability of each algorithm in generating a fit for each of the known epochs through the scan.

The model fits constructed by both k-Nearest Neighbor and Bayesian Linear Regression had varying degrees of success. k-Nearest Neighbors imposes a regression fit based off of known combinations of attributes and labels. Because of this, k-Nearest Neighbors had limited success when providing a regression fit in a forecasting capacity. Due to the sparsity of the mission scan data provided, the Bayesian Linear Regression model projections captured the generic trend of the parameters to the span of associated reference launch epochs. The result was a model which captured a forecast in keeping with the provided data while losing fidelity at the individual epoch resolution. This chapter will provide discussion on the epoch regions where the models provided the best fit and the associated trade offs in relying on the constructed models.
4.1 Machine Learning Trends at each Time Step through the Mission

This section discusses the performance of k-Nearest Neighbors and Bayesian Linear Regression in reflecting the patterns observed throughout the launch window. The perifocal distance throughout the mission provides a useful illustration of the differences between the two models considered. The trends present through the launch window result in model fits which exhibited behaviors associated to the machine learning algorithms. The other orbital elements are captured in appendix A and B for k-Nearest Neighbor and Bayesian Linear Regression Respectively. In the figures provided, a solid black point reflects a known converged solution from SNOPTA evaluation. Additionally, tolerance bars are provided reflecting parameter convergence as discussed in 3.1.4.

4.1.1 k-Nearest Neighbors

k-Nearest Neighbor provided a robust means of generating mission trajectories for the entirety of the launch season based off of relative neighbors to a provided launch epoch. Figure 4.1 below reflects the true values of perifocal distance at different time steps through the duration of the mission compared against the model fit provided by k-Nearest Neighbor. The resolution of the model has been improved from 1.03505 days to every 3.10515 hours (1.03505/8).

Because of the sparsity of the families on which k-Nearest Neighbors is trained, the trend captured by k-Nearest Neighbors reflects a fit to the perifocal distance data as a jagged pattern through the launch window. Through different mission segments, k-Nearest Neighbor provides better performance when the training data does not have oscillations as is seen in the Pre-OPF, Pre-NRI, NRI, and NRHO Stay Segments. Fit to a constant trajectory after the NRHO Stay results in a forgiv-
ing portion of the mission where the model fit adequately reflects the trend to within tolerances described in section 3.1.4 for NRD through RPF.

![Figure 4.1: k-NN Regression Fit for Perifocal Distance through Mission](image)

By design, an inferred epoch for k-Nearest Neighbors considers the weights provided by the closest elements. This introduces a "notched" discontinuity where local epochs have more influence on the generated trend rather than following the expected pattern through the launch window. This exists through the model fit generated during the Pre-OPF segment for perifocal distance, illustrated in figure 4.2 below.
Figure 4.2: k-NN Regression Fit for Perifocal Distance through Pre-OPF

Rather than having a continuously decreasing trend like in the data it is presented, k-Nearest Neighbor reflects a larger influence from the nearest launch epoch data for the epochs circled in red. The distribution of launch epoch and parameter data introduces a scenario where the training launch epochs and parameters to the right of the target epoch have a shorter Euclidean distance. This results in a weighted decision with higher influence from training combinations for launch opportunities with shorter difference to the desired projection rather than keeping with the trend of known data for a launch opportunity chronologically further away to the left. The trend captured by k-Nearest Neighbors is then influenced more by epoch data closer to 1.03505 days rather than the prior available epoch, which is further away.
4.1.2 Bayesian Linear Regression

Bayesian linear regression considers the orbital element set as a sparse subset of a Gaussian distribution. The fit generated then infers estimates at each epoch step through a launch window by calculating the probability of an attribute and label combination based off the training set provided. In figure 4.3 below, the mean of this gaussian projection is reflected at different segments through the trajectory. Where k-Nearest Neighbors considers each neighbor’s impact on an epoch, Bayesian regression considers the distribution of the entire data set provided. The mean value generated in the bayesian regression provides a generic trend meeting the expected values of perifocal distance through different mission segments.

Figure 4.3: Bayesian Regression Fit for Perifocal Distance through Mission
The initial data set from which this model is trained was solved for feasibility of the trajectory, which can be prone to falling into local minimums when evaluating the cost function. Given more attention, the trajectory could be modified to fit a global minimum for which less observable discrepancies would exist through the launch window. Posing an inference based off of a probability function results in a generic trend through the window rather than meeting the epoch values at each launch opportunity. As observed in figure 4.3, this can be helpful in reducing the impact of data points with deviations from a family of solutions.

The generic inferential distributions can result in a model fit which does not capture significant resolution at individual epochs when presented data with oscillating values. This is portrayed in figure 4.4 where the argument of periapsis during the outbound powered flyby maneuver oscillates around a range of values and does not follow a fixed trend. Also captured in figure 4.4 is the standard deviation associated to the distribution as generated by the Bayesian regression module in SciKitLearn.

![Figure 4.4: Bayesian Regression Fit for Argument of periapsis through Outbound Powered Flyby](image)

Figure 4.4: Bayesian Regression Fit for Argument of periapsis through Outbound Powered Flyby
The maximum deviation for argument of periapsis through the launch window is projected to be 6.2 degrees which does not fall within the confidence interval expected for state convergence as defined in 3.1.4. For comparison, the model fit during the pre-OPF coast is illustrated in figure 4.5. Due to the generic fit for the data presented, the Bayesian regression still does not quite meet each epoch in the launch window but the standard deviation is lower at approximately 0.9 degrees. The consistent trend that exists between the orbital parameters through each step in the window results in a generic trend which more accurately reflects a discernible pattern.

Figure 4.5: Bayesian Regression Fit for Argument of periapsis through the Pre-OPF Coast

As mentioned in section 3.2.2, a model fit from a given machine learning algorithm is only as reliable as the data provided during training. The impact of training data’s effect on the model is reflected in perifocal distance during post-OPF coast in figure 4.1 and the argument of periapsis during outbound powered flyby in figure 4.4 where oscillations in mission data generate unique gaps in model regression. The orbital parameters can be modified until a global minimum is found for a given
launch window. After finding the global minimum, the observed oscillations can be minimized resulting in a more reliable fit from Bayesian regression. Further work is required to understand if the resulting projection from the algorithm’s model fit could comprise a valid parameter value to capture the state of an EM-3 trajectory at the given mission segment and time step.

In addition to fitting a model to the provided data through a launch window, Bayesian regression also generates a realistic forecast of orbital parameters outside of the known range of epochs in the training data. The projected elements provide new launch opportunities in keeping with the expected trends for parameters associated to a rendezvous with DSG. Within the scope of this study, forecasting has provided predictions for launch opportunities 1.03505 days prior to and following the nominal window. Due to the quasi-stable nature of the 9:2 NRHO, mission projections are specific to an arrival opportunity at the rendezvous with DSG. Forecasting using Bayesian regression is limited to launch opportunities near in time to the observed launch window. Logistically, compressing or expanding the timeline also has an impact on vehicle performance. Additionally, imposing a projection beyond a tight resolution would operate under the assumption that the numeric trends of orbital parameters for launch opportunities would be continuous. This assumption is not inherently true, especially in the case of an NRHO where orbital parameters are close but not equal at various rendezvous opportunities.

4.2 Improving Resolution for Mission Availability Scans

The discussion so far has been focused on the results at the time step resolution however the regressions generated by the machine learning algorithms must be consistent through the entirety of the mission trajectory. The plots in figures 4.6 and 4.8 show the relationship between the attribute (a reference launch epoch) and
the labels associated to the trajectory (classical orbital elements and mission elapsed time) for reference launch epochs spanning from November 7th through November 12th, 2025 (JD = 2460897 to 2460992). Since mission elapsed time is a label during the machine learning process, plotting mission elapsed time and perifocal distance against the reference launch epoch attribute showcases the multidimensional relationships that exist in the mission trajectory estimates.

4.2.1 k-Nearest Neighbor through the Launch Window

Figure 4.6 reflects the perifocal distance for inferred k-Nearest Neighbor regression epochs alongside the known trajectory data through an entire launch window. In this study, the application of k-Nearest Neighbors has been limited to fill in knowledge between existing reference launch epoch trajectories because of the limitations of its reliability to the known span of epochs. Due to the sparsity of the data provided during training, regression estimates at the time step resolution resulted in regions where closer neighbors affected the estimate more than the expected trend. Overall, these regions of increased influences do not overpower the estimates through the entire trajectory and the generated inferences trend towards the expected launch epochs.
As described in section 3.2.3, it is of equal importance to capture trends specific to a segment of a mission. Figure 4.7 (A) and (B) illustrate the perifocal distance through the Near Rectilinear Insertion and Departure finite burn maneuvers, respectively. The relationship between mission elapsed time, the COE, and the label is further showcased in figure 4.7.A where the regression introduces a sharp displacement at different reference launch epochs. Despite these sharp discontinuities, perifocal distance maintains continuity at the segment resolution. Figure 4.7.B further illustrates convergence to the rendezvous with DSG through the constant trend exhibited in perifocal distance regardless of reference launch epoch after reaching the NRHO.
Figure 4.7: k-Nearest Neighbors Regression fit for Perifocal Distance before and after the NRHO
4.2.2 Bayesian Linear Regression through the Launch Window

The scope of trajectory forecasting in this study has been limited to launch epochs 1.03505 days outside of the known family of trajectories. With this range, Bayesian regression has successfully inferred projections in keeping with the training data. As discussed at the time step resolution in section 4.1.2, the projected model fits the presented data as a generalized gaussian distribution rather than focusing on a given epoch. Figure 4.8 illustrates the projected trend for perifocal distance through the launch window.

As in k-Nearest Neighbors, the Bayesian regression considers mission elapsed time as another label for the associated reference launch epoch. The projection of estimated MET versus perifocal distance results in a distribution through the launch window which captures the expected mission availability. The combined estimates for MET and perifocal distance also provide a reasonable estimate forecasting trajectories before November 7th and after November 12th.

Figure 4.8: Bayesian Linear Regression fit for Perifocal Distance through the Launch Window
In contrast to the discontinuous trends exhibited by k-Nearest Neighbors in figure 4.7, figures 4.9.A captures the continuous relationships of both MET and perifocal distance during insertion to the NRHO. Through NRI, projections provided by Bayesian regression satisfy the known epochs, however forecasted estimates require further processing when considering whether the orbital elements comprise a valid trajectory. As a constant forecast, predictions made through NRD must be valid as they comprise an already proven trajectory provided by the solved reference epochs. The regression estimates provided by the Bayesian model maintain the expected convergence of trajectory after the NRHO as illustrated in figure 4.9.B which captures the perifocal distance through the NRD finite burn.

Figure 4.9: Bayesian Linear Regression fit for Perifocal Distance before and after the NRHO

(A) Near Rectilinear Insertion

(B) Near Rectilinear Departure
4.3 Consolidating ML Projections to Construct a Trajectory

Up to this point, the discussion has been focused on individual orbital parameters. The goal of this thesis is improving mission availability knowledge by generating entire descriptive trajectories for previously unknown launch epochs. As an example, suppose an orbital analyst was tasked with determining mission feasibility for April 5th, 2026 for a reference launch epoch of 8:20 EST. The generated scan from Copernicus has converged solutions for reference launch epochs at April 5th at 2:09 EST (6:09 UTC) and April 6th at 3:00 EST (7:00 UTC). Due to visibility constraints for launch, neither of these opportunities provide the desired environment for the required support efforts. By imposing a model fit to the trends of the known launch epochs, a new mission trajectory is generated which was not previously captured in time history files.

Continuing along with the familiar discussion regarding classical orbital elements, each descriptive orbital parameter can be propagated based off trends through the window. Figure 4.10 illustrates this during Pre Outbound Powered Flyby at the time-step resolution for a k-Nearest Neighbors model fit.

Figure 4.10: k-Nearest Neighbor Model at a Time Step
The time step evaluation is then appended to a vector representing the trajectory through the entire mission. The projected value for argument of periapsis for the desired trajectory is plotted alongside the known values through the window in figure 4.11. As discussed in section 3.1.4, the models developed in this study reconstruct mission trajectories between Pre-OPF to the end of returned powered flyby. The pattern observed in 4.11 shows that the fitted projection follows the trend through the launch window. Projections are made for each of the orbital elements discussed in section 3.1.2.

Figure 4.11: k-Nearest Neighbor Model through Mission

The combination of all projected orbital elements then describe the position and orientation of Orion throughout a newly constructed trajectory. When combined into a time history csv file, these orbital parameters can be optimized in Copernicus, or some other orbit analysis platform, through numeric analysis. For the intention of illustrating what this looks like graphically, the orbital parameters were converted into a state vector using SpiceyPy’s conics routine. The trajectory is illustrated in figure 4.12 below where the reference launch epoch trajectories for April 5th at 2:00 EST and April 7th at 3:00 EST are also reflected. The trajectories reflected converge to the same NRHO orbit defined as the rendezvous arrival opportunity with DSG.
The inferred orbital parameters generated using Bayesian regression produce similar results. With a Gaussian distribution fit imposed, the model fit produced by the Bayesian regression algorithm follows the generic trend seen through a launch window as discussed in section 4.1.2. Argument of perigee at the time-step resolution reflects the trend through the observed window as seen in figure 4.13. The trend for argument of perigee through the mission also fits well when observed alongside other launch opportunities in the window as seen in figure 4.14. 

Figure 4.12: Trajectory Generated by k-Nearest Neighbors
However, when propagating all elements into a state vector, the generated trajectory begins to show the impact of the generalized fit. In figure 4.15, the trajectory for the desired launch epoch overlaps with the reference trajectory associated with a launch at 2:06 despite being 6 hours later. While presumably the trajectory would follow the closer epoch, further numeric processing is required to determine if the generated trajectory is truly feasible for the new launch epoch.
4.4 Overall Performance of Machine Learning

Both k-Nearest Neighbors and Bayesian regression impose assumptions when constructing a predictive model. The machine learning algorithms in this study considered sparse data sets with divergences in values associated to numerical processing. When applying machine learning analysis to an entire launch season, measures must be taken to understand and work around the limitations present in k-Nearest Neighbors and Bayesian Regression. In the context of orbital trajectories, an appropriately trained model can result in orbital parameter combinations which successfully represent an Orion mission to an NRHO orbit about the moon. Despite their limitations, both algorithms observed in this study provided valuable trajectory knowledge at the time step resolution and throughout the duration of a launch window.
At this point of the study, machine learning has proven to be a promising method of generating orbital elements tied to trajectories for off-nominal launch epochs. In this section, further discussion will be provided regarding the observed performance of the machine learning algorithms through the launch season. By providing a uniformly distributed training set, k-Nearest Neighbor and Bayesian regression generated reliable results. Additionally, the algorithms proved to have regions through the launch windows where the generated parameters fit well within the expected trends and others where the model fit suffered from artifacts of model fitting. Along with providing the desired resolution, the algorithms were designed with the intention of limiting computational demand. Observing metrics useful in describing the performance of a machine learning algorithm provides information on how accurate the resulting fit to the existing data was. Finally, the intention of this study was to construct mission trajectories which can be incorporated in the analysis of a launch window by orbital analyst. A recommendation is made on how the results generated by the machine learning algorithms can be utilized.

5.1 Capturing Performance through Mean Square Error

A useful metric in describing the performance of an inferential model is the mean square error. The mean square error describes the sum of errors between individual epochs squared divided by the number of sample points. The specific equation is as follows:
\[ MSE = \frac{1}{n} \sum_{i=1}^{n} (Y_i - \hat{Y}_i)^2 \]

where \( \hat{Y}_i \) is the inferred value and \( Y_i \) is the known parameter value at a given epoch. Because MSE is evaluated at each epoch, it represents the performance of the generated model through the region considered during estimation. In this section, MSE provides understanding of how a model performs when evaluated against certain configuration parameters incorporated in the design of the ML algorithms.

5.2 Reliability of ML Algorithm in Reconstructing COEs

As has been stated throughout this paper, the results of a machine learning algorithm are only as good as the data provided. The data on which the algorithm is trained should retain enough of the expected trend throughout the window to effectively reconstruct the distribution of parameters without favoring a given region. Additionally, the math models, as described in section 2.3, are governed by assumptions resulting in numeric artifacts from the algorithms. Understanding the limitations of the algorithms through the application in mission trajectory analysis provides further support for how and where a given model can be considered reliable.

5.2.1 Reliably Capturing COEs through Uniform Launch Epochs

The constructed machine learning algorithm must provide reliable estimates for the values of classical orbital elements throughout the duration of an observed launch window. An important aspect of attribute and label reconstruction for a given launch epoch is the relative distribution of the training epochs. With improper distribution, a model can train off of launch epochs that favor a given region rather
than reflecting the entirety of a launch window. The model would be effective at reconstructing the parameters through the trained regions and lose reliability elsewhere.

In figure 5.1 and 5.2 below, the model fits provided by two training sets for argument of periapsis at the first time-step through the launch are compared. The plots illustrate the epochs off of which the model is trained in the known data and the projected values for epochs within the known region for the given launch window. Figure 5.1 illustrates a fit for the algorithm trained off of randomly selected epochs and 5.2 reflects the inferred epoch solutions for a model presented with uniformly distributed epochs.

Figure 5.1: Non-uniform Training Distribution
Distribution can be seen to have a distinct impact during the model reconstruction process. Because of this, algorithms were designed to train off of sets of data with uniform distribution providing a model which captured the entire trend of the launch window.

5.2.2 Regions of Confident Performance

As described in section 2.3, k-Nearest Neighbors can reliably infer data trends within a known region of epochs. As generated epochs leave the known region, k-Nearest Neighbors creates guesses for the parameters off of the closest neighbors and diverts from the expected trend. This model fit from k-Nearest Neighbors is not effective as a method of forecasting through the launch window. The break from the trend through the launch window is observed in figure 5.3 where the algorithm generates an expected trend within the region of known epochs and begins to divert.
once the epochs continue into regions outside of the training data. By comparison, the Bayesian projection maintains the generic decreasing pattern though it loses resolution due to an improper degree fit for the time step.

Bayesian regression performs poorly when the training data does not have a distinct pattern. An example of an over generalized model is illustrated in figure 4.4 which reflects the argument of periapsis from November 18th through the 28th. The inability to capture specific values for individual epochs introduces discrepancies in orbital element combinations when propagating the state of Orion through the orbit. The impact of orbital parameter mismatching is shown in figure 5.4 below which illustrates the propagation of the same over generalized orbit from the launch window in figure 4.4. This introduces noticeable discontinuities through the Pre-OPF segment and an entire break from what should be a rendezvous with the DSG through the NRI segment. Since the reference training trajectories were solved specifically for feasibility, certain orbital elements do not maintain a continuous trend. Further processing would be required on the reference trajectories prior to obtaining a trend for which Bayesian regression would work.
5.3 Performance Considerations for ML Algorithms

When developing a machine learning algorithm, design decisions are made based off the model’s support for reconstructing epochs through a range of data. Specifically, performance is dictated by variance at individual epochs as reflected by the calculated mean square error. MSE can be used when comparing performance of a model through specific training resolution and model configurations such as the number of neighbors to consider or the degree to impose on the data. In this section, the performance of k-Nearest Neighbor and Bayesian ridge will be analyzed in their ability to reconstruct the launch window parameter values. The observed performance introduces trade-off opportunities which will be further discussed in section 5.4.
5.3.1 ML Model Fidelity vs Training Resolution

An area of interest when considering the application of machine learning in generating viable inference models for orbital parameters is the resolution required to reflect the observed trend through a launch window. The historic data off of which this model was trained has trajectory information every 1.03505 days. Using less data points when constructing a model alleviates the computational demand necessary when processing data through the algorithms. The application of machine learning has been structured as a supplemental tool with the ability to infer knowledge from existing SNOPTA converged solutions. Observing the performance from a limited number of training points benchmarks what resolution of data points is required to reliably reconstruct a trajectory. As an example, the performance for a family of solutions spanning from November 17th through November 27th, 2025 spanning 11 launch opportunities is described in this section.

In figure 5.5 below, the mean square error is plotted against the number of training points used to construct the k-Nearest Neighbor model. The red bar indicates the standard error associated to a variance of +/-0.01 degrees as was previously discussed as the expected tolerance for angles. At eight training points, the model generates a standard error related to 0.2 degrees. It is not until 10 training points that the required convergence to +/- 0.01 degrees is met. This provides a trade-off between faster computational performance versus obtaining a higher resolution from the model which will be further discussed in section 5.4.
Comparatively, performance of Bayesian regression as illustrated in figure 5.6 shows the projections generated by the model fit do not achieve the tolerance expected regardless of the number of training points. Using 11 training points generates the lowest magnitude of MSE which is associated to a variance of approximately +/- 0.1 degrees. While not quite meeting the expected tolerance to provide an exact trajectory, the model can provide an initial guess on which a future trajectory could be based with further processing confirming convergence.
k-Nearest Neighbors requires an input of neighbors to consider when establishing weights during the learning process. In figure 5.7 below, MSE is plotted against the number of possible neighbors when reconstructing the launch window values of argument of perigee from November 18th through the 28th. Typically, design of a nearest neighbor algorithm considers an odd number of neighbors to avoid potential ties when generating weights at different epochs. At 7 neighbors, the MSE is associated to a variance of +/- 0.05 degrees which does not fall within tolerance of 0.01 degrees but may potentially provide a good initial estimate from which a feasible trajectory can be obtained. The trend constructed when considering 7 neighbors is illustrated in figure 5.8.
Figure 5.7: Number of Neighbors versus k-NN MSE

Figure 5.8: k-Nearest Neighbors with 7 Neighbors
Due to the sparse data available, the imposed fit can only be interpreted as overfitting the existing trends. The constructed model fits every available epoch but does not provide reasonable trends in between known launch epochs. The most robust performance was experienced using 2 nearest neighbors. Although odd neighbors should be considered for tie breaking purposes, the uniform distribution discussed in section 5.2.1 ensures that even contribution is provided from the available elements. The model constructed using 2 neighbors achieved the required tolerance of 0.01 degrees.

The Bayesian model requires the definition of relevant samples when constructing a Gaussian distribution. As seen in figure 5.9 below, the best performance is achieved when using 6 samples to define the Bayesian model. With 7 samples onwards, the Bayesian algorithm plateaus to a biased value and provides no added benefit.

![Figure 5.9: Number of Samples versus Bayesian Regression MSE](image)

Figure 5.9: Number of Samples versus Bayesian Regression MSE
5.3.3 Machine Learning Performance through the Launch Window

Models configured in this study were defined at each family based off of the observed MSE through the window. Distribution of the training samples and the machine learning parameters used during configuration contributed to the established performance of both k-Nearest Neighbors and Bayesian regression. The model fit at the time step resolution can be utilized to propagate the entirety of the mission trajectory as observed in figures 4.12 and 4.15. However over generalized models lacking fidelity at the time step resolution can result in gaps through the expected trajectory performance as observed in figure 5.4. By considering the required performance and regions of reliable epoch generation, machine learning can help expand mission trajectory knowledge for launch seasons.

5.4 Practical Application of ML during Mission Design

When designing a mission, orbital analysts at NASA consider scenarios beyond the nominal launch window. The best case scenario for a launch opportunity is governed by performance cost of the mission in obtaining a desired trajectory. From this reference trajectory, orbital analyst are tasked with generating feasible trajectory scans associated to different launch initiation opportunities. The construction of this initial scan can be computationally demanding. This study covered the practicality of applying machine learning in reconstructing and filling in mission knowledge for the desired scans without resorting to computationally intensive operations. Thanks to the quasi-stable nature of near rectilinear halo orbits, defining orbital parameter observed in most missions through a launch window provided trends which were reasonably mapped to pattern recognition algorithms.
Machine learning, as it has been applied in this study, can be incorporated in the launch window development process for standardized missions similar to EM-3. An initial spread of time history files associated to feasible launch trajectories can serve as training points for a given launch window. The existing time history spread has been improved from mission data every 1.03505 days to launch opportunities every 3 hours. The launches must be broken up into mission segments defined by the standardized translational maneuvers. For each window, the algorithms must be configured to obtain the desired performance as benchmarked through the MSE figures in this chapter.

During development of a launch window scan for any type of mission, families of solutions arise in the orbital parameters. Beyond missions to an NRHO or within the vicinity of the moon, machine learning algorithms can be applied expanding knowledge of feasible trajectories capturing the pattern of the existing family. The code developed for this thesis relies on analyzing orbital elements at the time step resolution in a format compatible with regression fit calculations. Numeric trends through a given launch window will exist regardless of the mission. An analyst developing a launch window scan using machine learning is tasked with capturing the desired trajectories in a relevant reference frame. Additional work may also exist where the orbital elements must be made compatible through pre-processing. Once the orbital elements are reflected in a format with discernible trends, the pattern recognition methods employed in this study can be incorporated.

Additionally, machine learning design must meet the defined tolerance that constrains a continuous trajectory. With certain model configurations, performance might not meet the absolute desired tolerance for a mission, as observed in figure 5.9, but can generate a reasonable estimate with less computational demand. The orbital analyst is presented with a trade-off regarding parameter generation. Should an exact trajectory be required, finer tuning of the model is necessary. However, machine
learning algorithms can also provide reasonable estimates which can be further processed in an orbits analysis platform. Once the desired performance is obtained from the existing launch windows, mission trajectory candidates can then be provided for launch epochs not part of the initial data set.

When considering launch availability through a window, knowledge on how to achieve mission trajectory opportunities is required at the hour, if not minute, resolution. The model constructed from a given family of mission trajectories can supplement knowledge by establishing candidates at a finer resolution than what currently exists. These candidates could replace the existing method of orbital parameter modification and introduce a structured approach for orbit optimization trajectory design. An orbital analyst with access to time history data for a given mission can generate rendezvous trajectories formed from the existing trends observed through a base feasibility scan. The orbital analyst can then incorporate these candidate models into an orbit optimization platform to minimize the spacecraft’s performance cost function further.
Chapter 6

FUTURE WORK

The mission trajectory results generated by machine learning provided promising performance in reconstructing and projecting mission trajectories through a launch window. Analysis of the performance from the observed algorithms in mapping unique launch opportunities has introduced areas where improvements can be made in generating a more robust solution. Specific to k-Nearest Neighbors, application of a modified search radius would alleviate artifacts observed when considering data sets with non-uniform launch epoch distribution. In future applications of Bayesian regression, automating the selection of the number of samples when constructing the model fit would require less tailoring to obtain a reasonable projection. Computational performance from both models can be improved by incorporating parallel processing into model fitting. Beyond the two algorithms observed, there exist other candidate models for which model fitting may be more robust. Finally, application of machine learning could aid the development of missions beyond trajectories to a 9:2 NRHO to the moon.

6.1 Improving Models Utilized

Both k-Nearest Neighbors and Bayesian Regression are supervised learning methods which require post-processing analysis when designing the best configuration. The decisions made when selecting parameters for both algorithms were tailored around achieving desired performance for specific launch windows. Parameters selected in the development of machine learning algorithms impact weighted decisions while potentially introducing numeric artifacts in the inferential projections.
they generate. Further understanding of both models can be achieved by expanding the analysis of performance for different model configurations.

### 6.1.1 Modifying the Search Radius for k-Nearest Neighbors

The k-Nearest Neighbors model fit utilized the standard Euclidean search radius when establishing weights for the desired epochs. Euclidean search radii are most effective when the distribution of training epochs is uniform. Launch opportunities had data available at a fixed resolution so application of a euclidean search radius was justified. However time history data may not always be available in such a convenient format. The algorithm can be improved by incorporating a search radius which does not rely on uniform distribution of data. A given search radius must support the reconstruction of orbital parameters which exist in real-valued vector space.

Orbital parameters exhibit unique trends at different training epochs related to specific sample variances. To better reflect the weighted impact of variance at each time step, it can be helpful to scale the difference of parameters at each epoch by its respective sample variance. k-Nearest Neighbors can incorporate a search radii which scales the data set off of individual sample variances through standardized Euclidean. Standardizing the orbital parameters based off of sample variance will introduce influence from unique epochs with consideration for a single epoch’s impact on the entire window. The standardized euclidean search radius is then:

$$\text{distance} = \sqrt{\frac{(\text{epoch}_{\text{ref}} - \text{epoch}_{\text{target}})^2 + (\text{parameter}_{\text{ref}} - \text{parameter}_{\text{target}})^2}{\text{variance}_{\text{target}}}}$$
An assumption made based off the application of euclidean in the design of k-Nearest Neighbors is that the influences can be reflected as a linear relation. A linear assumption between attributes and labels during training may not be a valid approach for all distributions of orbital parameters or training resolutions. The Minkowski distance is a method of calculating the distance between two training points as a combination of defined exponents. Specifically, the Minkowski calculation is as follows:

\[
\text{distance} = \left( (\text{epoch}_{\text{ref}} - \text{epoch}_{\text{target}})^p + (\text{parameter}_{\text{ref}} - \text{parameter}_{\text{target}})^p \right)^{\frac{1}{p}}
\]

The euclidean distance is a specific instance of the Minkowski distance for which \( p = 2 \). By applying the more generalized distance metric, a better relation may be reached when considering weighted distributions for orbital parameters for which data distributions may have exponential trends.

### 6.1.2 Automatically Picking Configuration Parameters

The training data describing different rendezvous opportunities with DSG maintained unique trends throughout the launch season observed. The model fits generated capturing these unique trends were themselves distinct from window to window. As the windows changed, the configuration parameters defining the designed model fits were tailored to support each unique trend. The scope of this study was focused on how a machine learning algorithm needed to be configured when generating a launch window scan.

Parameters such as the number of samples to consider when designing a Bayesian regression algorithm were selected based off of observed performance through
the launch window. The generalized model would remove the tailored design at individual launch windows and provide support for the unique trends observed. Where six samples may be effective at capturing the trend through one specific window, the same assumption may not be relevant for another unique rendezvous with the DSG trajectory.

Automatically selecting configuration parameters would be computationally intensive since multiple runs of machine learning training and performance evaluation would be required. The models considered were supervised learning algorithms requiring further designer input when refining a given model. Counteracting measures can be taken when evaluating such a generic model. A production model must reliably reconstruct orbital elements through the entirety of a launch trajectory however an evaluation model could compare performance of configured models at a lower resolution of time steps through the mission.

6.1.3 Improving Computational Demand

One of the end goals of this study was the construction of a tool which would alleviate computational demand in projecting mission trajectories through a launch window. Inferring orbital parameters on the fly as has been done can result in models which can be computationally demanding. The SciKit Learn API used was incorporated with the intention of running on a single computer. In future iterations, application should be considered in a processing environment where model fit calculations can have allocated parallel jobs. A true comparison of computational performance between a machine learning algorithm versus the traditional methods would require that both scan generation methods be run on similar computational architectures. Processing performance will need to be captured for the algorithms observed beyond the application on a personal computer.
6.2 Alternative Machine Learning Applications to Launch Window Construction

Machine learning algorithms provided promising results when inferring orbital elements through the observed launch windows. Their design came with constraints regarding the relations between parameters through the window. k-Nearest Neighbors and Bayesian regression also introduced numeric artifacts when processing orbital parameters based off of the mathematical evaluation methods used. Future studies in launch window inferences could consider candidate models for which the method of trajectory inference provide a better suited fit to the trends observed through a family of solutions.

The application of machine learning considers launch window resolution improvement through regression methods. The knowledge gained from the projected results provides an understanding for how many opportunities exist for reaching a given trajectory. In a situation where an arrival epoch for rendezvous with the DSG were provided, classification could target the family of solutions achieving the desired trajectory.

The problem shifts from knowing when a launch is desired and not knowing what was feasible for arrival to knowing where the arrival is desired without knowing what launch date is required to achieve rendezvous. Figure 6.1 below illustrates reference launch epochs versus orbital element combinations for a 30 day scan. The arrival opportunities through a season maintain trends through the defined launch windows. By applying a classification method through the season and a resulting regression, machine learning can have expanded impact beyond the analysis of a unique launch window.
Support vector machines can generate the best fit for the desired arrival rendezvous by establishing a hyperplane comprised of the orbital elements with the arrival epoch describing the position on the trajectory. The orbital element combinations can then be evaluated against a kernel function providing a numeric cost function with the intention of matching the desired rendezvous position as best as possible. Support vector machines can apply linear regression for data processed through classification schemes. The resulting classification could provide potential launch dates and generate along with it the associated orbital elements.

6.3 Applying Methodology to Other Missions

The mission observed consists of a nominal stay at the NRHO of about 10 days. Machine learning algorithms reconstructed families of solutions associated to a
fixed phase arrival at the NRHO. Off-nominal mission trajectories to the rendezvous orbit with DSG must also account for potential early departures in the event of an emergency. Between the ability to generate mission trajectories through machine learning and the quasi-stability of the NRHO, an additional study can analyze spliced trajectories piecing together a reasonable departure trajectory associated to a specific arrival opportunity.

In figure 6.2 below, perifocal distances for two families of NRHO stays are illustrated. There exists an overlap period between the first and second family where the trajectory from the November 5th arrival is preparing to depart as the trajectory from the November 11th arrival orbit has just arrived. In the scenario presented, the November 11th arrival trajectory could depart the NRHO as early as November 15th rather than staying the entire duration of the nominal mission. A machine learning classification algorithm could search for and spawn a return trajectory at the nearest departure opportunity. The resulting trajectory would help orbital analyst prepare for an off-nominal scenario in which the crew would depart the NRHO early.

Figure 6.2: Perifocal Distance versus Arrival and Departure Opportunities
Chapter 7

CONCLUSION

With EM-3, NASA is establishing the ground work for understanding how to support life in deep space. The DSG’s 9:2 NRHO provides a unique exploration opportunity with advantages in station keeping and quasi-stability in our general planetary vicinity. Developing mission trajectories to and from a NRHO through the span of a launch season is instrumental in understanding the logistical demands, performance requirements, and feasibility of achieving rendezvous with the DSG. The application of machine learning during the mission development process provides unique utilities and insight into reliability to support exploration mission design.

In this study, two machine learning algorithms, k-Nearest Neighbors and Bayesian Regression, were configured with the intention of generating reasonable mission launch trajectories through individual launch windows in a season. Analyzing the numeric performance of the two algorithms and observing the orbital trends through a mission provided an understanding for how machine learning algorithms could be utilized by orbital analyst when generating launch window scans. Specifically, the machine learning algorithms enabled projection of orbital parameters associated to new launch epoch trajectories in the family of solutions of arrivals at the NRHO.

Propagating the state vector at each time step illustrated the utility of inferring individual parameters through the mission. Capturing the performance of the machine learning algorithms through mean square error provided insight for how the configuration of the two algorithms fared when generating new orbital parameters related to the true, known, data. In closing, introducing machine learning in the mission design process makes the development of launch season mission feasibility knowledge a repeatable, standardized approach while alleviating computational demand.
Bibliography


APPENDICES

Appendix A

K-NEAREST NEIGHBOR TIME STEP REGRESSION FIGURES

Figure A.1: k-Nearest Neighbors Fit for Eccentricity through Mission

Figure A.2: k-Nearest Neighbors Fit for Inclination through Mission
Figure A.3: k-Nearest Neighbors Fit for Right Ascension through Mission

Figure A.4: k-Nearest Neighbors Fit for Argument of Perigee through Mission
Figure A.5: k-Nearest Neighbors Fit for Mean Anomaly through Mission

Figure A.6: k-Nearest Neighbors Fit for Epoch through Mission
Figure A.7: k-Nearest Neighbors Fit for MET through Mission
Appendix B

BAYESIAN LINEAR TIME STEP REGRESSION FIGURES

Figure B.1: Bayesian Regression Fit for Eccentricity through Mission

Figure B.2: Bayesian Regression Fit for Inclination through Mission
Figure B.3: Bayesian Regression Fit for Right Ascension through Mission

Figure B.4: Bayesian Regression Fit for Argument of Perigee through Mission
Figure B.5: Bayesian Regression Fit for Mean Anomaly through Mission

Figure B.6: Bayesian Regression Fit for Epoch through Mission
Figure B.7: Bayesian Regression Fit for MET through Mission