The Pennsylvania State University
The Graduate School

SPACECRAFT STATE ESTIMATION AND STEALTH THROUGH ORBIT-PERTURBING MANEUVERS: A GAME THEORY APPROACH

A Dissertation in
Aerospace Engineering
by
Jason A. Reiter

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Doctor of Philosophy

May 2020

The dissertation of Jason A. Reiter was reviewed and approved* by the following:

David B. Spencer
Professor of Aerospace Engineering
Dissertation Advisor
Chair of Committee

Robert G Melton
Professor of Aerospace Engineering

Puneet Singla
Associate Professor of Aerospace Engineering

Christopher C. Byrne
Associate Professor of Mathematics

Richard Linares
Charles Stark Draper Assistant Professor of Aeronautics and Astronautics
Massachusetts Institute of Technology
Special Signatory

Karl M. Reichard
Associate Research Professor
Penn State Applied Research Laboratory

Amy R. Pritchett
Professor of Aerospace Engineering
Head of the Department of Aerospace Engineering

Abstract

In space mission architectures, redundancy of subsystems mitigates the risk of isolated malfunctions, but not the risk of the entire platform being targeted by hostile forces. This dissertation explores the use of deception as a tactical defense mechanism. The scenario of a satellite accomplishing an unknown mission while evading a dedicated ground sensor is modeled as a two-player zero-sum game, which supports a robust performance assessment based on both the satellite and the sensor optimizing against each other. Moreover, the methodology determines the optimal strategies and associated performance based on assumed constraints that can be varied parametrically, so the method can be adapted to a range of specific scenarios as well as to advances in underlying technologies.

The two player game, which featured infinite strategy choices for both players, was solved using a reinforcement learning/neural network approach, specifically proximal policy optimization, capable of high-fidelity strategy tuning. Parametric sensitivity analysis on the results, computationally impractical with proximal policy optimization, was instead accomplished with coevolution, a genetic algorithm approach.

The primary result is that, under reasonable technological assumptions for both players, an evading spacecraft can expect to avoid detection by an optimally tracking optical sensor just under 60% of the time, and can accomplish this evasion expending, on average, just under 16 m/s of ΔV every 5 days. Analysis also indicated that this result is more sensitive to the sensor's parameters than the satellite's. Additional results compare the computational methodologies employed. The impact of the research as a methodological innovation is discussed and future direction is offered for applying the methodology to other problems, as well as possible refinements in the context of this application.

Table of Contents

List of Figures

List of Tables

List of Symbols

A	Advantage Function
D	Divergence
D_{TV}	Total variance divergence
D_{KL}	Kullback-Leibler Divergence
$\vec{F}$	Force vector $[N]$
G	Gravitational constant $[km^3/kgs^2]$
$\boldsymbol{H}$	Jacobian
K	Kalman gain
L	Loss
$L_\pi(\pi)$	Local approximation to the discounted reward $[km]$
M	Mass $[kg]$
$\boldsymbol{P}$	Error covariance matrix
$\boldsymbol{P}_i^-$	A priori covariance
$\boldsymbol{P}_i^+$	A posteriori covariance
P	Transition probability distribution
Q	State-action value function
$\boldsymbol{R}$	Variance of the measurement noise
$\vec{R}$	Observation site position vector $[km]$
S	Entropy bonus
V	Value Function

$\vec{Y}$ Vector of angles calculated from the state estimate [$radians$]

$\vec{a}$ Acceleration vector [km/s^2]

a Action

d Residual

e Eccentricity

$\hat{g}$ Gradient estimator

h Angular Momentum [kgm^2/s]

i Inclination [$radians$]

j Semilatus rectum [km]

l Probability ratio

m_t Moving average

p Probability evader chooses a specific cell

q Miss probability

$\vec{r}$ Position vector [km]

r Payoff/fitness/reward function [km]

r_a Radius of apogee [km]

r_p Radius of perigee [km]

s States

t Time

u Control Input

$\vec{v}$ Velocity vector [km/s]

v_t Squared gradient

w Neural Network weighting coefficients

$\vec{x}$ Full state

$\hat{x}_i^-$ A priori state estimate

$\hat{x}_i^+$ A posteriori state estimate

x	Function input
$\vec{y}$	Vector of angular observations $[radians]$
y	Function output
z	Nonlinear activation function
ΔV	Change in velocity magnitude $[km/s]$
$\Delta\vec{v}$	Change in velocity vector $[km/s]$
Δy	Observation residual $[radians]$
$\mathbf{\Phi}$	State Transition Matrix
Π	Polynomial basis functions
Υ	Range $[km]$
Φ	State transition matrix
Ψ	Nonlinear basis functions
Ω	Right Ascension of the Ascending Node $[radians]$
α	Hyperparameter
β	Hyperparameter
γ	Discount factor
δ_t^V	TD residual of the discounted value function
ϵ	Hyperparameter
η	Discounted reward $[km]$
θ	Policy parameters
λ	Reward steepness
$\lambda_\oplus$	Longitude $[radians]$
μ	Gravitational parameter $[m^3/s^2]$
ν	True anomaly $[radians]$
ξ	Resources allocated for sensor tasking
π	Strategy profile/policy

ρ Visitation frequency

ρ_0 Distribution of initial states

$\hat{\upsilon}$ Line-of-sight unit vector from the site to the spacecraft

ϕ_{trans} Flight path angle [*radians*]

$\phi_{\oplus}$ Latitude [*radians*]

χ Response function

ψ Effectiveness of spending researches in search process

ω Argument of Perigee [*radians*]

List of Acronyms

AFRL	Air Force Research Lab
BLS	Batch Least Squares
COE	Classical Orbital Elements
CUT	Conjugate Unscented Transfer
DMU	Drag Make-up Maneuver
ECEF	Earth-Centered Earth-Fixed
ECI	Earth-Centered Inertial
EKF	Extended Kalman Filter
EOM	Equations of Motion
EP	Electric Propulsion
FIG	Fisher Information Gain
FOR	Field-of-Regard
FOV	Field-of-View
GAE	Generalized Advantage Estimator
HEO	Highly Elliptical Orbit
IAM	Inclination Adjust Maneuver
IOD	Initial Orbit Determination
IMM	Interacting Multiple Model
ISM	Interval Similarity Model
KL	Kullback–Leibler

LEO	Low Earth Orbit
LLE	Largest Lyapunov Exponent
MDP	Markov Decision Process
NE	Nash Equilibrium
PE	Pursuit Evasion
PoL	Patterns of Life
PPO	Proximal Policy Optimization
RAAN	Right Ascension of the Ascending Node
ReLU	Rectified Linear Unit
RL	Reinforcement Learning
RMM	Risk Mitigation Maneuver
SBSS	Space Based Space Surveillance
SSA	Space Situational Awareness
SSN	Space Surveillance Network
STM	State Transition Matrix
SQP	Sequential Quadratic Programming
TOT	Time Over Target
TRPO	Trust Region Policy Optimization
UKF	Unscented Kalman Filter

Acknowledgments

I would like to start by thanking Dr. David Spencer for all of his help on this dissertation, as well my master's thesis and other endeavors, both academic and professional, these past six years. His wisdom and mentorship have proven to be invaluable time and time again.

Thank you, as well, to the members of my PhD committee, Dr. Robert Melton, Dr. Puneet Singla, Dr. Christopher Byrne, Dr. Richard Linares, and Dr. Karl Reichard, for their support and valuable feedback on the work I have conducted while at Penn State.

I would also like to thank one of my undergraduate professors at Cal Poly, Dr. Kira Abercromby, for inspiring me to pursue astrodynamics research through her engaging orbital mechanics courses and for continuing to support me past my undergraduate studies.

A special thanks goes to my family who have been encouraging me to pursue my dreams for as long as I can remember. My father, Adam, has been a constant source of inspiration by showing me what it means to put your all into your work through his own career. He has instilled in me so many important professional skills, such as public speaking, conversation skills, and networking. I wouldn't be who I am today without him. My mother, Dawn, showed me how to set standards for myself in everything that I do; higher than anyone else would expect of me. My sister, Amanda, has always been a major source of encouragement and someone I can look up to despite her lesser age. My grandma, Elanor, my number one fan, is the person I look forward to sharing details of my achievements with the most. Together, they are the best support system I could ever ask for.

Dedication

This dissertation is dedicated to the memory of my grandfather, Harvard, the original Dr. Reiter, who passed away from Alzheimer's disease just a few months ago.

Grandpa was the first person to introduce me to science. When my sister and I were young, he would take us to his lab, show us all of the shiny instruments, and let us play with his special wax paper. It was his passion for his work and pursuit of a singular goal that inspired me to do the same. Though he never got to see me fulfill the dreams that he inspired, I hope that he passed knowing the immense impact he has had on my life.

This dissertation is just a small example of how he shaped me, and he will continue to do so as I remember him fondly as the man I aspire to be.

Chapter 1
Introduction

1.1 Context: Spacecraft Detection Evasion

In space-based object behavior, the environment is typically unchanging, or at least predictable to a certain degree. When optimizing spacecraft maneuvers, unless a high-fidelity model of the space environment is desired, determining the maneuver is generally straight forward and limited only by the computational resources available. This is not the case for the sensor avoidance problem, though. When assuming that an optical sensor is interested in maintaining custody (knowledge of the spacecraft's state to a required level of accuracy) of a specific spacecraft, and the spacecraft is trying to prevent this custody by maneuvering, it can also be assumed that the sensor will adjust its tasking strategy to optimize its own reward. This leads to a back-and-forth of strategy adjustments that is generally unpredictable and non-trivial to optimize.

1.2 Game Theory Frames The Problem

The mathematical theory of games is perfectly suited to analyzing strategic interactions such as the satellite/sensor interaction studied here. Indeed, such "wargames" were a driving force behind much of the theory [1]. A mathematical game is defined by a set of players (actors), a set of strategies (actions), and a payoff function that assigns a valuation of any possible outcome that results from the collective strategy choices of the players (called a strategy profile).

The class of games especially fitting the sensor-satellite interaction is that of two-player zero-sum games. "Zero-sum" games model scenarios where the players' objectives are diametrically opposed, so that what is good for one is bad for the other. This means

that the payoff of one player (conventionally player 1) can be regarded as "the payoff" of the game, with player 1 striving to maximize this payoff and player 2 striving to minimize it. In the present context, the payoff is the distance between the spacecraft and the center of the sensor's FOV, so the satellite is treated as player 1 and sensor as player 2, striving to minimize successful evasion by the satellite.

Mathematical results from classical game theory summarized in Section 2.2 indicate that optimal strategies exist for the sensor and the satellite, such that each guarantees a certain level of performance, no matter what the other player does. These guarantees, player 1's lower bound on the payoff and player 2's upper bound on the payoff, meet, resulting in an expected payoff called the value of the game. This game theoretic approach avoids the risk of optimizing against a fixed optimistic assumption about one's opponent that might prove false in a real mission. Instead, it develops robust strategies based on an equally optimizing opponent.

Before introducing the player models of the sensor/pursuer and satellite/evader, the overarching solution approach is introduced next so that it may be referenced when discussing the players.

1.3 Reinforcement Learning Solves The Problem

Reinforcement learning (RL) is able to solve for the optimal strategies of the two players simultaneously and provide the resulting strategies in the form of neural networks. The RL approach is not mathematically proven to find the true optimum, but it has become a best practice in engineering for problems whose direct mathematical solution presents problems. For example, classical optimal control approaches require a fixed finite set of actions for the two players to follow, whereas the true strategy sets of the sensor and the satellite are both infinite. Quantizing them to a finite mesh would support a linear programming solution, but a mesh fine enough to match the fidelity of the overall model would render that approach computationally impractical if not intractable, and also less desirable in other ways. For example, the neural networks can be sampled repetitively to find the optimal strategies under different initial conditions. The networks can also be easily adjusted to changes in the constraints of one or both players.

Proximal Policy Optimization (PPO) [2], a type of RL known for its ease of implementation and high performance, is used to continually update the strategies employed by each player (discussed in the following sections). Within prescribed constraints, this process will result in optimal strategies for both the satellite and the sensor as well as the

value of game. This value can be compared to a mission specific threshold to determine who "wins"; that is, whether the evasion success of the satellite is sufficient to enable it to succeed in the larger context of its mission.

The following sections discuss the implementation of the two players in this game in greater detail, as well as the use of competitive coevolution to further evolve the players' strategies.

1.4 Player 1: Space/Evading Player

The success of the space/evading player, assumed to be a geosynchronous spacecraft, is a function of a handful of inputs and constraints as well. The inputs include the spacecraft's own state (known perfectly) and the location where the sensor is assumed to be pointing, while the constraints on the player include the maximum change in velocity available for an instantaneous maneuver as well as the minimum and maximum duration allowed between maneuvers. The position of the spacecraft is essentially unbounded, only limited by the most effective application of the maximum change in velocity. Note that the limit on the change in velocity is only considered for each individual maneuver, not the total velocity change over an entire mission. Though this could affect the operational lifetime of the spacecraft, that is not considered here.

In attempting to elude the pursuing player, the evading spacecraft player will perform maneuvers constrained in size and the time between them (applied to sufficiently bound the space and limit computation time). The maneuvers are allowed to be performed anywhere along the orbit and in any direction (see Section 2.1). The RL procedure employed (as described in Chapters 2 and 4) allows the spacecraft to learn from its mistakes. Based on the effectiveness of its maneuvers with respect to the pursuing player's tasking strategy, the timing, size, and direction of the maneuvers required to optimize its reward (the distance between the spacecraft and the sensor) can be determined.

1.5 Player 2: Ground/Pursuing Player

The success of the ground/pursuing player, assumed to be a single optical sensor, is a function of just a handful of inputs and constraints. The inputs include the Cartesian position where the sensor is pointed and the tracks of observation data for the spacecraft being tracked, while the constraints on the player are the sensor's field-of-view (FOV) and the region in space the sensor can task within, called the field-of-regard (FOR).

For the sake of simplicity and computational savings, the sensor is assumed to correctly observe the spacecraft whenever it is located within the sensor's FOV. This means that the angular measurements of the spacecraft are automatically correlated (matching an observation to previous ones) and associated (assigning observations to a specific object). However, to ensure that the sensor is not unfairly advantaged in this game, it is assumed that a sufficient volume of observations must be collected for the spacecraft's state to be known with enough certainty to initialize an extended Kalman filter (EKF). In addition, the sensor can miss observations of the spacecraft for only so long before it has to recollect observations for initialization. Though actual initialization and filtering of the observations are not included in the simulation, they are accounted for in the environment. This will be discussed in greater detail in Chapter 4.

Since observations made of the spacecraft are assumed to provide perfect estimates of its state, there is no need to use an EKF to filter through the observations, which means that an estimate of the covariance is not available. Though this would be ideal as an objective function/value of the game, it was deemed unfeasible. Instead, the objective function/value of the game was selected simply as the normalized distance between the center of the sensor's FOV and the spacecraft.

The tasking of the sensor is performed with respect to its previously known orbit of the spacecraft (assuming that the spacecraft's state is known at the beginning of each simulation). The known ("nominal") orbit is propagated forward and the sensor is tasked to a state within a set distance, in kilometers, from the nominal orbit (the effective FOR). When the spacecraft has been found again, and for long enough to effectively employ an EKF, the nominal orbit known for the spacecraft is then set to be the spacecraft's state.

Similar to with the space player, RL is used to optimize the actions of the sensor. As more observations are collected, especially after the spacecraft has maneuvered, the sensor becomes better at tasking to the correct location with respect to the spacecraft's previously known orbit. Ideally for the ground player, the sensor would become so adept at proactively tasking that the spacecraft's state would never need to be reinitialized.

1.6 Sensitivity Analysis / Competitive Coevolution Augmentation

As a computational optimization methodology, the success of the PPO process depends on the tuning of the hyperparameters and there is a possibility of the RL process getting

trapped in a local optimum rather than finding the global optimum when optimized under prescribed constraints. To guard against such a result, an alternative genetic algorithm approach, competitive coevolution, is introduced that does not have the fidelity or the continual learning capability of the RL method, but is efficient enough to support parametric sensitivity analysis of the results. Still a computational approach to optimization, competitive coevolution offers the benefit of quickly exploring a much greater region of the strategy space (with or without constraints), enabling discovery of any shortcomings, if they exist, in the RL solution. Furthermore, once the RL and competitive coevolution solutions are mutually validated, competitive coevolution enables efficient quantitative estimation of the impact of changes in the assumed constraints placed on the two players. For example, if the evading spacecraft has been unsuccessful in evading the pursuing sensor with a certain ΔV constraint, the constraint can be relaxed further until the spacecraft is successful and a ΔV requirement can be determined for such a mission. Conversely, if the pursuing sensor is unsuccessful in determining the spacecraft's position, the sensor's FOV or effective FOR can be increased until the threshold for the win condition is met. This has great applied value in prioritizing changes in ground and space architectures, and is discussed further in Section 2.2.3.3.

Competitive coevolution, in this context, will involve the simultaneous evolution of the strategies of the two players in the PE game in which the fitness of a maneuver strategy of the evading spacecraft is in direct competition with a sensor tasking strategy of the pursuing ground sensor(s), and vice versa. The key to avoiding a local optimum and efficiently optimizing the game is to start out with a smaller search space. Knowing the constraints on both players, the game is solved with tight constraints and the optimal strategies of both players are determined. Competitive fitness sharing, shared sampling, and/or hall of fame strategies (see Section 2.2.3.3) are then employed while relaxing the constraints, thus expanding the search space, and re-solving the game repeatedly until the desired constraints are met. This will result in a globally optimal strategy for both players with respect to each other and a final winner of the game quantified by the success of the pursuing player in predicting the spacecraft's state at any moment in time.

1.7 Document Organization

This dissertation is divided into seven chapters and an appendix. Figure 1.1 provides a visual description of how the information flows between the chapters in the main body of the dissertation.

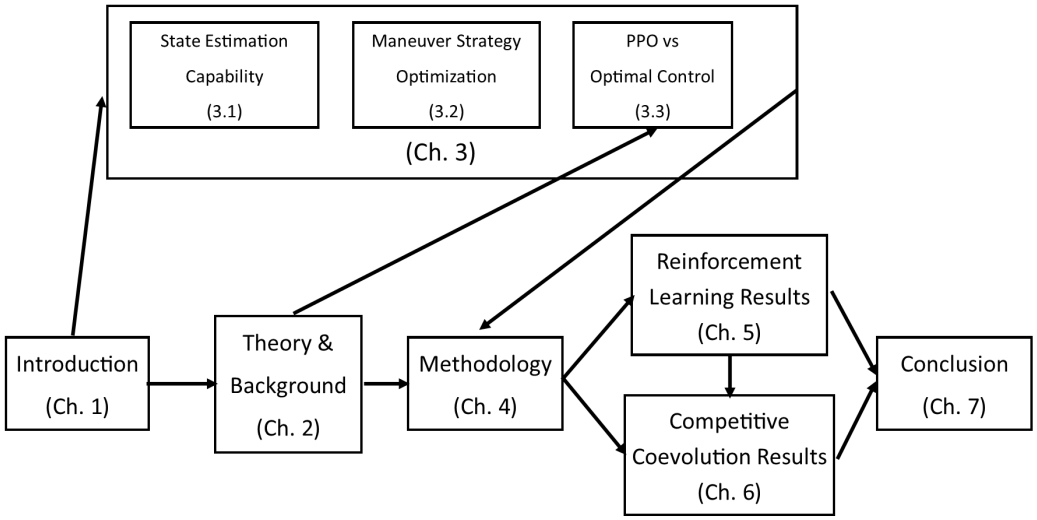


Figure 1.1. Information Flowchart

Chapter 1 here introduced the two-player zero-sum game to be solved, the evading spacecraft player and the pursuing ground sensor player, the PPO method that will be used to solve the game, and the competitive coevolution approach that can be used to perform additional analysis on these results. This information drives not just the theory and background presented in the next chapter, but directly justifies the limitations and capabilities explored in Chapter 3.

Chapter 2 explains in detail the theory employed in this dissertation. First, an overview of the two-body problem is presented to demonstrate how a spacecraft performing a maneuver affects its future orbital states. Then the game theory at play is reviewed, including two-player zero-sum games, the theory of fictitious play that serves as the foundation of this research, and the competitive coevolution methods explored. Next, the elements of RL are reviewed, including a comparison of different methods and a brief intro on the methods used in this dissertation. The next section presents a deeper examination of these methods, called policy optimization. The theory behind Trust Region Policy Optimization (TRPO), PPO, Generalized Advantage Estimation (GAE), and the Adam optimizer are all described in detail. Finally, the previously published work applying these theories is discussed.

Chapter 3 provides further foundation for this research by demonstrating the capabilities and limitations of using an optical sensor for tracking a maneuvering spacecraft, of a spacecraft attempting to simultaneously avoid detection while still meeting other mission objectives, and of using PPO for such a problem compared to traditional optimal control

methods. Though the theory presented in Chapter 2 is only necessary for the analysis of the capability of PPO, the results from the entirety of Chapter 3 sufficiently justifies the methodology applied in the next chapter.

Chapter 4 describes in detail the designed problem and how it is implemented in the RL environment. The combination of the different policy optimization methods detailed in Chapter 2 is then discussed. Finally, the contributions of this research are discussed with respect to previous applications of RL.

Directly building off of the methodology presented in the previous chapter, Chapter 5 presents the results of applying the combined policy optimization methods to the designed problem. Three different series of results are shown. First are the results optimizing only the strategy of the evading spacecraft, assuming that the sensor follows a prescribed tasking strategy. The next section shows the initial, non-converged, results from when the sensor was first introduced into the optimization. The last section shows the final results where lessons learned from the initial introduction of the sensor into the optimization are implemented and the strategies are allowed to further converge.

Chapter 6 follows the results of Chapter 5, applying the same methodology as was presented in Chapter 4, with the results found from implementing the competitive coevolution methods described in Chapter 2. Again, multiple stages of results are provided. First, competitive coevolution is used to optimize a continuous thrust trajectory that simultaneously minimizes propellant cost and maximizes the spacecraft's distance from a sensor, auch as that optimized using PPO in Section 3.4. Then the results are presented where competitive coevolution is used to determine the winner of the game, under set constraints, with a specific win-condition. Finally, the win condition is removed and the constraints are adjusted to perform sensitivity analysis on the RL results.

Chapter 7 summarizes the problem, methodology, and results presented in this dissertation and draws conclusions. Then potential future contributions towards this work are discussed.

Appendix A demonstrates the proof-of-convergence used to justify the application of policy optimization to this two-player PE problem.

Chapter 2
Theory and Background

This chapter covers the theory applied throughout the conducted research. Section 2.1 discusses the effects of maneuvers on a spacecraft's orbit under Keplerian dynamics. Section 2.2 introduces the game theory at play in this work. Section 2.3 begins the discussion on the fundamentals of RL. Section 2.4 summarizes the details of neural networks. Section 2.5 takes an in-depth look at the policy optimization methods used to encourage convergence of the players' strategies. Then Section 2.6 provides an overview of the previous work in related areas.

2.1 Spacecraft Maneuver Dynamics

To understand the interaction between the spacecraft and the sensor, one must first understand how the spacecraft behaves under Keplerian dynamics and the effects of maneuvers on the spacecraft's orbital path.

2.1.1 The Two-Body Problem

At its simplest, the motion of a spacecraft in orbit is known as the two-body problem, which considers the motion of a small object under the gravitational force of a second, much larger, object. Newton's second law and his universal law of gravitation are the starting points for constructing the required equations of motion (EOM). Newton's second law states that the time rate of change of linear momentum is proportional to the force applied. For a fixed-mass system, this gives the equation

$$\sum \vec{F} = \frac{d(M\vec{v})}{dt} = M\vec{a} \tag{2.1}$$

So, assuming the mass is constant, this means that the sum of all forces, $\vec{F}$ on the body are equal to the mass, M, multiplied by the acceleration of the body, $\vec{a}$.

Newton's law of gravitation allows for the components of the force to be found assuming that gravity is the only force affecting the small object. Figure 2.1 shows the geometry for a fixed-mass system in which a satellite of mass, m, is orbiting a body of mass, M, located at the center of mass of the system.

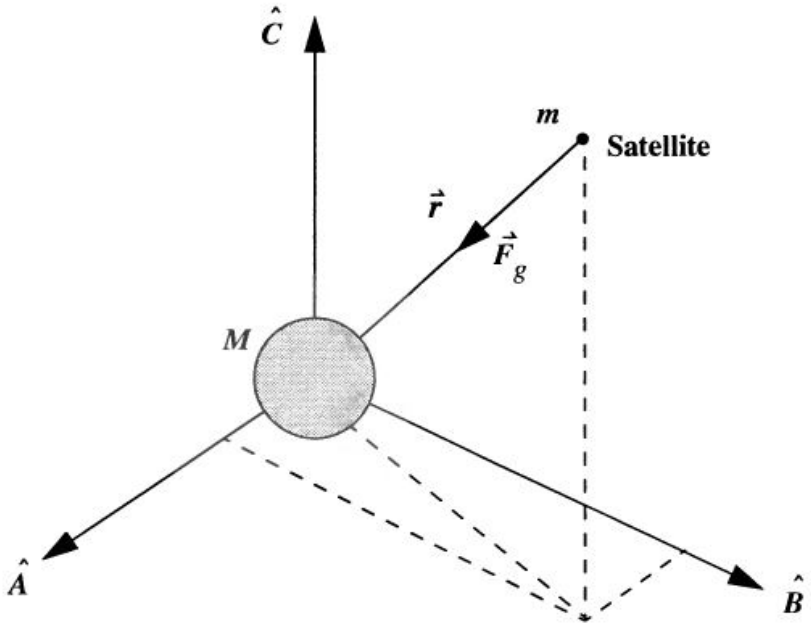


Figure 2.1. Gravitational Forces Acting on a Satellite [3]

When deriving the EOM, the system is assumed to be fixed in inertial space, or has a fixed orientation with an origin moving at a constant velocity. Figure 2.2 shows this system consisting of the Earth, $m_{\oplus}$, and a satellite, m_{sat}. The coordinate system $\hat{A}\hat{B}\hat{C}$, from Figure 2.1, is defined as an ideal inertial system displaced from the geocentric $\hat{I}\hat{J}\hat{K}$ coordinate system.

In the $\hat{I}\hat{J}\hat{K}$ system, Newton's law of gravitation for the force of gravity of the Earth acting on a satellite can be expressed as

$$\vec{F}_g = -\frac{Gm_{\oplus}m_{sat}}{||\vec{r}||^2}\left(\frac{\vec{r}}{||\vec{r}||}\right) \tag{2.2}$$

The geometry in Figure 2.2 gives the vector from the Earth to the satellite

$$\vec{r} = \vec{r}_{sat} - \vec{r}_{\oplus} \tag{2.3}$$

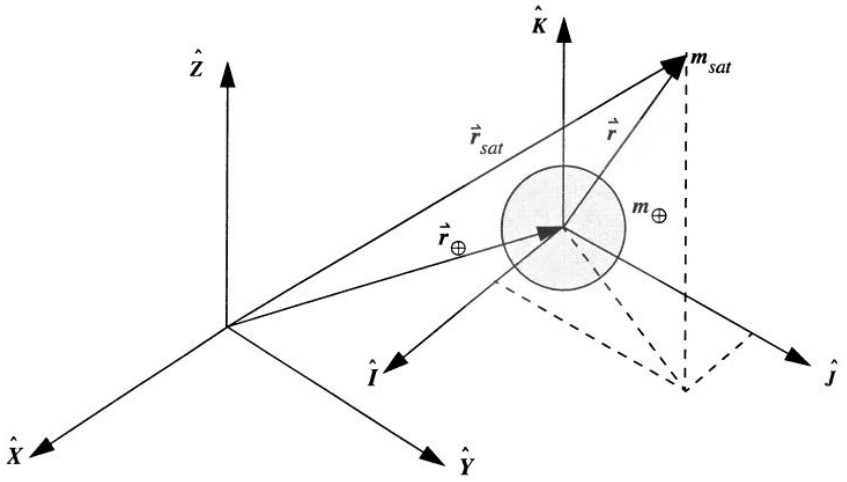


Figure 2.2. Geometry for Two Bodies in an Inertial Reference Frame [3]

where $\vec{r}_{sat}$ is the position vector of the satellite and $\vec{r}_{\oplus}$ is the position vector of the Earth, both with respect to the origin of the $\hat{X}\hat{Y}\hat{Z}$ coordinate system. Employing an inertial coordinate system allows for this vector to be differentiated without needing to consider the derivative of each axis of the coordinate system. Therefore, the second derivative of Eq. (2.3) can be written simply as,

$$\ddot{\vec{r}} = \ddot{\vec{r}}_{sat} - \ddot{\vec{r}}_{\oplus} \tag{2.4}$$

which is the acceleration of the satellite with respect to the center of the Earth. Combining this with Newton's second law and Eq. (2.2) gives

$$\begin{aligned} \vec{F}_{g_{sat}} &= m_{sat}\ddot{\vec{r}}_{sat} = -\frac{Gm_{\oplus}m_{sat}}{||\vec{r}||^2}\left(\frac{\vec{r}}{||\vec{r}||}\right) \\ \vec{F}_{g_{\oplus}} &= m_{\oplus}\ddot{\vec{r}}_{\oplus} = \frac{Gm_{\oplus}m_{sat}}{||\vec{r}||^2}\left(\frac{\vec{r}}{||\vec{r}||}\right) \end{aligned} \tag{2.5}$$

as the inertial forces. Note the difference between the two forces is only the sign, confirming that the force of the Earth is opposite to the direction of the satellite's force. Solving for the individual accelerations and using the differentiated form of the accelerations gives a solution for the relative acceleration:

$$\ddot{\vec{r}} = -\frac{G(m_{\oplus} + m_{sat})}{||\vec{r}||^2}\left(\frac{\vec{r}}{||\vec{r}||}\right) \tag{2.6}$$

This can then be re-written as

$$\ddot{\vec{r}} = -\frac{\mu}{||\vec{r}||^2}\left(\frac{\vec{r}}{||\vec{r}||}\right) \tag{2.7}$$

where μ is the gravitational parameter equal to $Gm_{\oplus}$ when assuming that the satellite's mass is significantly smaller than that of the Earth. [3]

2.1.2 Satellite Maneuvering

To find the states at every point in a satellite's orbit, Eq. (2.7) can be numerically integrated using an ordinary differential equation solver, such as MATLAB's ode45, given an initial state. When considering only two-body dynamics, the propagation of the orbit will return back to the same state after one orbital period and will infinitely traverse a closed path (circle or ellipse) assuming no outside forces are introduced.

For this research, only planar orbits and planar maneuvers are considered. When maneuvering between two orbits, the position at which the maneuver is applied remains unchanged while the velocity is changed to reflect the desired effect of the maneuver on the future orbit states. This change can be notated as

$$\Delta\vec{v}_b = \vec{v}_{final} - \vec{v}_{initial} \tag{2.8}$$

where $\vec{v}_{initial}$ is the velocity at the instant before the maneuver is applied and $\vec{v}_{final}$ is the velocity at the instant after the maneuver is applied. For this research, it is assumed that the maneuver is performed instantaneously such that the position is constant between $\vec{v}_{initial}$ and $\vec{v}_{final}$.

As the name implies, coplanar maneuvers don't change the orbital plane, but do change the orbit's size, shape, and the orientation of the line of apsides (the line connecting the orbit's periapsis and apoapsis where its orientation is defined by the argument of perigee, ω, as shown in Figure 2.3). The direction of applying a maneuver can be determined by calculating the flight path angle, ϕ_{trans}, or

$$\phi_{trans} = \arctan\frac{e\sin\nu}{1+e\cos\nu} \tag{2.9}$$

where ν is the angular position along the orbit with respect to perigee and e is the

eccentricity of the orbit, defined as

$$e = \frac{r_a - r_p}{r_a + r_p} \tag{2.10}$$

where r_a and r_p are the radius of apogee and perigee, the points on the orbit closest to the Earth (perigee) and farthest from the Earth (apogee). These, along with the rest of the classical orbital elements (COEs), can be seen visually defined in Figure 2.3).

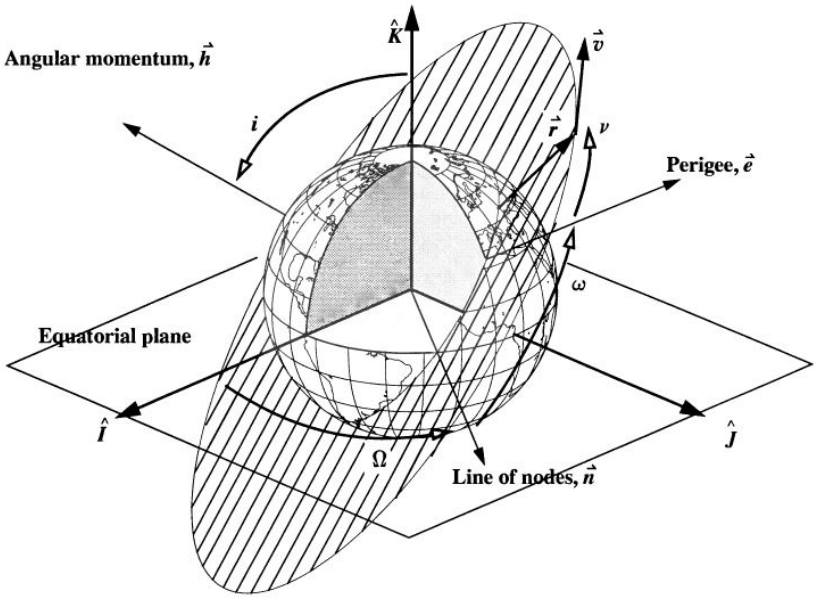


Figure 2.3. Classic Orbital Elements [3]

Mission planning and operations usually determines the location and direction of the maneuvers. When attempting to minimize the required ΔV, especially in maneuvers that change the size of the orbit, periapse and apoapse are the optimal locations to do so - where tangential burns (a flight path angle of zero) can be performed. However, in real missions, the objective is rarely that straightforward so most maneuvers are non-tangential with a non-zero flight path angle, as seen in Figure 2.4.

Figure 2.4 shows the effect of a maneuver, $\Delta\vec{v}_a$, applied tangentially at point a. Though not always the case, and not in this research, a second maneuver, $\Delta\vec{v}_b$, can also be performed (in this case at flight path angle, ϕ_{trans}) to re-adjust the orbital path if a transfer to a defined final orbit is desired. Otherwise, the spacecraft will remain in the "transfer" orbit until another maneuver is applied.

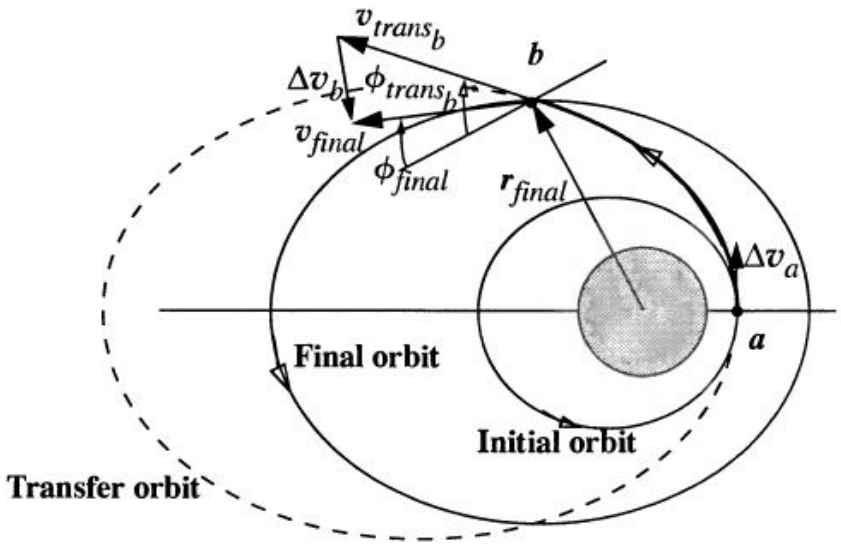


Figure 2.4. Nontangential Orbit Transfer [3]

2.2 Game Theory

2.2.1 Two-Player Zero-Sum Games

As discussed in the introduction, a mathematical game is defined by a set of players (actors), a set of strategies (actions), and a function that assigns a valuation to each player (called a payoff) of any possible outcome that results from the collective strategy choices of all the players (called a strategy profile).

Notwithstanding many detailed solution concepts for many specialized classes of interactions, the predominant solution concept for all games is known as the Nash equilibrium (NE), which is a strategy profile in which no player can improve their payoff through a unilateral change in strategy. The satellite-sensor interaction can be called a pursuit/evasion (PE) game. More generally, it belongs to a class of games called two-player zero-sum games, "zero-sum" referring to the diametrically opposed objectives of the players, which can be modeled as defining, in every outcome, the payoff of one player to be the negative of the other player's payoff. The result is that the payoff of one player (conventionally denoted as player 1) implicitly represents both players' payoffs, and is referred to as "the payoff," with player 2 striving to minimize the payoff that player 1 strives to maximize.

NE yields a much stronger solution concept in two-player zero-sum games than it does in general-sum games, where the payoffs of the players need not be in any particular

relation. If (x*,y*) and (x',y') are any two NE strategy profiles of a two-player zero-sum game, then (x*,y') and (x',y*) are also NE and the payoff of every NE is the same and is called the value of the game. The value of the game represents the best payoff player 1 can guarantee, as well as the best payoff player 2 can guarantee, and if both players play optimally, the value will be the payoff of the game. Because any NE strategy of either player guarantees the value of the game, which is the best payoff that can be attained against an optimizing opponent, NE strategies in two-player zero-sum games are called optimal strategies. These results are far stronger than in general-sum games, which might have a multitude of Nash equilibria, all with different payoffs, and no clear way to decide which, if any, is the best prediction. In a two-player zero-sum game, all predictions agree and represent a worst-case payoff guarantee, within the assumptions of the model.

Nash proved that all finite games have at least one NE, provided "mixed strategies" are allowed [4]. Mathematically speaking, mixed strategies are probability distributions over actual strategies (distinguished as "pure strategies" when mixed strategies are also considered) that can be used in a single play of the game. Nash's strong mathematical result has often led to people glossing over the fact that, in many games, mixed strategies cannot be implemented. However, in the present context, mixed strategies can be meaningfully interpreted/implemented as temporal distributions in the pursuit/evasion process. Exactly as in the mathematical model of independent randomizations of pure strategies by the two players, the uncertainty created by randomizing maneuvers yields a probabilistic result on the likelihood of detection and tracking, so mixed strategy solutions are considered in the present context with no compromise in realism.

The RL (specifically PPO) definition does not have the exact structure of the mixed strategies that Nash introduced in 1949, but Nash's existence proof remains valid for infinite games as long as the space of strategy profiles is compact, which it is in the present context. Both the sensor and satellite have closed and bounded real-valued choices in finitely many dimensions. In other words, each player's strategy set, and therefore the set of strategy profiles, is a closed and bounded subset of n-dimensional Euclidean space, and is therefore compact, so the game must have a Nash equilibrium, which in turn implies the existence of optimal strategies for both players. The PPO strategies have a similar element of randomness as Nash's original mixed strategies, because they are dynamically triggered by events that are inherently random, or at least unpredictable, in nature. It is this unpredictable quality of the PPO strategies that prevents an opponent from exploiting any exact sequences of maneuvers.

2.2.2 Fictitious Play and Reinforcement Learning

There are deep connections between mixed strategies, NE, and reinforcement learning. In 1949, George Brown and Julia Robinson [7] proposed the idea of fictitious play, an iterative method for solving discrete zero-sum games. Fictitious play is, in essence, the implicit dynamic process of which "Nash equilibrium" is an equilibrium. Players repeatedly play the game and after (arbitrary) moves in the first round, each player chooses a pure strategy best response to the averaged history of the opponent's moves, interpreted as a mixed strategy. Brown and Robinson proved that the averaged history of both players' (pure strategy) moves will always converge to a (mixed strategy) NE and the averaged history of the payoffs will converge to the value of the game [6–8]. Brown and Robinson's method of fictitious play could be regarded as the first modern reinforcement learning algorithm.

Beyond the inherent randomization of the PPO strategies, the entire RL approach used here employs Brown and Robinson's repeated play learning paradigm, albeit with a different learning algorithm suited to the infinite strategy spaces, and uses their assumption that both players have perfect knowledge of their opponent's strategy space and move history. Although perfect knowledge of one another's moves is not realistic in the real-life space-based game modeled here, the assumption is used only to find the optimal strategies, which, once found, do not require this perfect knowledge to implement. The idea is not for the satellite or the ground sensor to actually carry out the complete learning dynamic during its mission, but rather that once the optimal strategies have been found under the harsh conditions of extreme information exposure, the resulting strategies can be implemented by activating the fully trained neural network. The optimal strategies would be determined in the mission design phase based on information and/or assumptions about each player's architectural and technological constraints.

2.2.3 Coevolutionary Algorithms

2.2.3.1 Introduction

Evolutionary algorithms are often used in place of more traditional gradient methods, especially when search spaces are highly modal, discontinuous, or highly constrained. A coevolutionary algorithm, specifically, is an evolutionary algorithm in which individuals are evaluated based on their interactions with other individuals. Similar to traditional evolutionary algorithms, coevolutionary algorithms attempt to model concepts seen in

nature such as heredity and "survival of the fittest" for the purpose of solving computation problems. As with organic genetics, the computational solutions are altered using genetic variation like mutation and crossover to discover the optimal genetic compositions for modeled entities. In the computational application, the genetic compositions, or genotypes, are the independent variables being solved for in the optimization process.

Evolutionary algorithms typically begin with some defined fitness function, $r : G \rightarrow \mathbb{R}$, that assigns a real value to every possible genotype in G. The fitness in this case is then determined by comparing $r(g_1)$ and $r(g_2)$ to determine which genotype, where $g_1, g_2 \in G$, is the best fit. Coevolutionary algorithms are not able to use the same kind of direct comparison, though. Instead, two individuals are compared on the basis of their outcomes from interactions with other individuals [9].

How the individuals interact depends on the nature of the problem. Coevolutionary algorithms fall into two main groups: competetive coevolutionary algorithms and cooperative coevolutionary algorithms. In cooperative coevolution, individuals are rewarded when they perform well working among other individuals, and likewise, punished when they don't work well together. Conversely, competitive coevolution rewards individuals at the expense of others. This process is described in more detail in Section 2.2.3.3.

2.2.3.2 Elements of Evolutionary Algorithms

Before going into the specific details on how competitive coevolution is applied, this section first discusses the components found in evolutionary algorithms.

Representation: In order to transition from the natural world to the computational world, the representation - or encoding - of the system, a mapping from the phenotypes (possible solutions) onto a set of genotypes (individuals within the solution) that represent them, must be described. For instance, given an optimization problem where the possible solutions are integers, the given set of integers would be the set of phenotypes and the equivalent binary expression would be the genotype. The genotype space is where the evolutionary search takes place, where the best phenotype is decoded to provide the best solution upon completion of the search [10].

Fitness Function: The role of the fitness function is rather self-explanatory. It is a function used to assign a quality measure to genotypes, most often referred to as the objective in traditional optimization and the payoff in game theory. To continue with the example in the previous paragraph, the fitness function could be simply the square of the integer value. This would involve decoding the genotype from binary form to integer form and then taking the square of the integer phenotype [10].

Population: The population can be thought of as a set of possible solutions or, in this context, a multiset of genotypes. Individuals change and adapt throughout evolution as part of a greater population. Given a representation, defining a population may be as easy as specifying the number of individuals in it (the population size). In most evolutionary algorithms, the population size remains constant while the population itself varies. The population is used to select the parents in the genetic mutations that seed the next generation [10].

Selection: The goal of parent selection is to assess individuals given their quality in comparison to others. The best performing individuals (with respect to the fitness function) will be selected as parents and undergo variation (described in the following paragraphs) in order to create children. This selection process ensures a continual improvement in the fitness of the population [10].

Mutation: The goal of mutation and combination (discussed in the next paragraph) is to create new individuals from old ones (candidate solutions in the phenotype space). A mutation is a unary variation operator applied to slightly modify one individual genotype, creating a child. Mutation is a stochastic process intended to apply a random, unbiased change to the genotype. In the ongoing example, this would resemble a small probability that a digit in the binary genotype will be flipped. The objective here is to gradually explore the space so that the populations are able to represent the entire space with a finite, constant size [10].

Recombination: Recombination is a binary variation operator where information from two parent genotypes is merged to create one or two offspring genotypes. Like mutation, this process is stochastic and should have no inherent bias. While a mutation is intended to randomly search the space, recombination is intended to randomly combine two high-performing genotypes in an attempt to produce a better or uniquely good genotype. Applying this to the example problem would involve swapping randomly selected digits between two binary genotype parents. Though this is not guaranteed to produce desirable combinations of traits, some will have improved characteristics [10].

Replacement: As stated earlier in this section, the population sizes are assumed to remain constant. This means that, as mutation and recombination occurs, a process must be followed for replacing parents with their mutated offspring. This decision is usually based on their relative fitness values, favoring the higher quality offspring. While parent selection is usually stochastic, replacement is inherently deterministic. The remaining population should be an improvement on the previous one and serves as the parents for the next iteration of evolution [10].

2.2.3.3 Competitive Coevolution

Competitive coevolution, also known as host/parasite or "arms race", refers to the simultaneous evolution of two or more populations in which the fitness of an individual in one population is in direct competition with an individual from another population. It has traditionally been used in two kinds of problems. First, it can be used to evolve interactive behaviors that are difficult to evolve in terms of absolute fitness. Second, and in line with this research, it can be used to gain insight into the dynamics of game-theoretic problems. To this end, reference [11] details three techniques for use in competitive coevolution:

1. Competitive fitness sharing: Competitive fitness sharing is used to ensure the survival of important but poorly represented types. This is accomplished by treating each parasite (a strategy of player 1) as an independent resource to be shared by those hosts (a strategy of player 2) in the population that can defeat it. This results in essentially having a separate sharing function for each parasite in which the fitness assigned to the host defeating parasite with the set of indices J is expressed as

$$\Sigma_{j \in J} \frac{1}{N_j} \tag{2.11}$$

   where N_j is the total number of hosts in the population defeating parasite j. Even though the rewarded host might not have defeated as many parasites as others can, it is rewarded for defeating parasites that few others can.

2. Shared sampling: It is desirable to reduce the computational effort expanded in competition by testing each individual in the host population against only a limited sample of parasites from the other population. Between generations, this can mean selecting parasites with the best fitness for use in the next generation. However, if there are several niches in the parasite population, this would tend to pick similar individuals from the niche that happened to have the highest fitness, rather than making a diverse selection from all niches. Instead, new sample members are chosen such that the competitive shared fitness is maximized within the sample. This technique is called shared sampling, and is detailed further in Algorithm 1.

Algorithm 1 Shared Sampling Algorithm [11]

Initialize current sample to be the empty set
For each opponent i from previous generation,
    $beat_i = 0$ (# in current sample beating i)
while the current sample is not yet full **do**
    For each parasite j not yet in sample,
        $fit_j = 0$ (fitness within sample)
        For each opponent k beat by j last generation,
            $fit_j = fit_j + \frac{1}{1+beat_k}$
    Let j be such that fit_j is maximal
    Add individual j to current sample
    For each opponent i from previous generation,
        If j beat opponent i last generation,
            Increment $beat_i$
end while

3. Hall of Fame: In a finite population (set of all strategies), a genotype (strategy) must be successful almost every generation to stay in the population. In competitive coevolution, a lost but previously successful genotype might have become successful again. This could lead to less-than-optimal results. This is where Hall of Fame comes into play, in which the best individual from each generation is retained for future testing, as shown in Algorithm 2. In this setup, hosts are tested against both current parasites and a sample of the hall of fame. This results in a more optimal solution.

Algorithm 2 Hall of Fame [12]

$nCoev \leftarrow 0$; $A \leftarrow player_1$; $B \leftarrow player_2$; $c \leftarrow thresholdvalue$;
$HoF_A \leftarrow \varnothing$; $HoF_B \leftarrow$ InitialOpponent();
$pop \leftarrow$ EVALUATE(HoF_B); // Evaluate initial population
while $nCoev < MaxCoevolutions \wedge$ NOT($timeout$) **do**
    $pop \leftarrow$ RandomSolutions(); // pop randomly initialized
    $i \leftarrow 0$;
    while $(i < MaxGenerations) \wedge$ (fitness(best(pop)) $< c$) **do**
        $parents \leftarrow$ SELECT(pop);
        $childs \leftarrow$ RECOMBINE($parents, p_X$);
        $childs \leftarrow$ MUTATE($childs, p_M$);
        $pop \leftarrow$ REPLACE($childs$);
        $pop \leftarrow$ EVALUATE(HoF_B);
    if fitness(best(pop)) $\geqslant c$ **then** // winner found!
        $nCoevolutions \leftarrow 0$; // start new search
        $HoF_A \leftarrow HoF_A \cup best(pop)$
        $temp \leftarrow A$; $A \leftarrow B$; $B \leftarrow temp$; // interchange player roles
    else
        $nCoev \leftarrow nCoev + 1$; // continue search
    end if
end while

Although such competitive techniques allow sustaining the competition for longer, they do not directly encourage continual coevolution. With a fixed search space, a limit to the optimality will eventually be reached. In addition, it is still possible to get stuck in local optima. This can be fixed through complexification, a technique for elaborating strategies by adding new dimensions to the search space rather than searching for a new path through the original space. Even if a global optimum is reached in the search space of solutions, new dimensions can be added, opening up a higher-dimensional space where even better optima may exist [13]. Such a technique works well in applications of RL.

2.3 Reinforcement Learning

2.3.1 Introduction

RL, as the name implies, is the process of learning how to maximize a numerical reward by mapping situations to actions. The learner explores the state and action spaces, noting the rewards granted after each action. Another name for this state-action-reward connection is a Markov decision process (MDP), in which an agent visits a series of states connected by actions and each action-state pair has an associated reward. The state-action mapping is continually updated until the maximum reward is achieved.

Though most traditional machine learning is classified as either supervised learning or unsupervised learning, RL is different. Supervised learning is learning from a training set of labeled examples provided by a knowledgeable external supervisor. The objective there is to extrapolate from the provided data so that the learner can act appropriately in situations not covered in the training set. Unsupervised learning is different in that its objective is to find structure hidden in unlabeled data. Though this might appear to encompass RL, it does not. RL attempts to maximize a reward signal instead of trying to find hidden structure. Though uncovering structure could be useful in RL, it does not address the RL problem by itself.

Unlike supervised and unsupervised learning, RL has a unique challenge where it must attempt to balance exploration with exploitation. To obtain a lot of reward, a RL agent must prefer actions that it has tried in the past and found to be effective (exploitation). However, in order to discover these actions, it has to try new actions that have not been selected before (exploration). Neither can be pursued exclusively to be effective - the agent must try a variety of actions while simultaneously favoring those that produce the most reward.

RL also must, unlike other machine learning methods, consider the whole problem of a goal-directed agent interacting with an uncertain environment. Many researchers are able to plan with general goals without taking into account the planning's role in real-time decision making, or where the planning models come from. RL requires much more consideration, employing a complete, interactive, goal-seeking agent. RL agents can sense their environment and choose actions to influence that environment in pursuit of an explicit goal. In planning, RL has to consider how the planning and real-time action selection affect each other. This will produce results that can be assessed with real-world models in mind [14].

2.3.2 Elements of Reinforcement Learning

To use a RL algorithm, one must first understand the four major elements involved: policies, reward signals, value functions, and a model of the environment [14].

2.3.2.1 Policies

A policy, or strategy, defines the learning agent's behavior for a given state. As mentioned earlier, this is the mapping from the agent's state in the environment to an action to be taken at that state. This alone is sufficient to determine the agent's behavior. As will be discussed in the following section, policies are often represented as deep neural networks [14].

2.3.2.2 Reward Signals

A reward signal defines the goal (the objective function) of a RL problem. After an action is enacted at a state, the agent is sent a single number, the reward. The objective is to maximize the total reward that the agent receives over the entire simulation run, called an episode. The reward signal is what the agent uses to justify altering the policy, selecting similar states/actions in the future if the reward is desirable [14].

2.3.2.3 Value Functions

While the reward signal is used to provide feedback on the immediate quality of an action, the value function is used to represent the quality of an action in the long term. In other words, the value of a state is the total amount of reward an agent can expect to accumulate in the future. This is useful in RL problems where an action might earn a poor reward immediately, but prove to be highly rewarding as time goes on, or vice-versa. Using value functions allows for the total earned reward to be properly maximized over the entirety of an episode. Value functions will be discussed in greater detail in Section 2.5 [14].

2.3.2.4 Models

The final element of RL is the model of the environment. This is how, given a state, or a state-action pair, the next state visited is predicted. The model is used for planning a course of action for the agent given all possible future states before they are actually

experienced. In space-based environments, the model is the application of the Keplerian dynamics to determine future states, as described previously in Section 2.1 [14].

2.3.3 Reinforcement Learning Methods

Though the elements of RL are consistent between applications, the methods for solving the problems can vary highly. RL is often used for tasks in which the state space is combinatorial and large. In such cases, finding a truly optimal policy, or the optimal value function, is nearly impossible. Instead, the goal is to find a good approximate solution using limited computational resources. Examples of such approximate methods include on-policy, off-policy, or policy gradient optimization methods [14].

2.3.3.1 On-Policy vs Off-Policy Methods

There are two primary methods of RL: on-policy methods and off-policy methods. On-policy methods attempt to evaluate or improve the policy that is used to make decisions, whereas off-policy methods evaluate or improve a policy different from that used to generate the data. On-policy methods, like Monte Carlo control with Exploring Starts, are generally simpler to implement so they are considered first. Off-policy methods, on the other hand, are more complex since they require two policies, one that is learned about and becomes the optimal policy, and one that is more exploratory and is used to generate behavior. This usually creates greater variance in the results and requires more time to converge than on-policy methods, but is also more powerful [14].

2.3.3.2 Policy Gradient Methods

Though, similar in some ways to both on-policy and off-policy methods, policy gradient methods are unique in that they learn a parameterized policy that can select actions without consulting a value function. Though a value function may still be used to learn the policy parameter, it is not required for action selection like in on-policy and off-policy methods. Methods that learn approximations to both the policy and value functions are often called actor-critic methods, in which "actor" refers to the learned policy and "critic" to the learned state-value function.

Policy gradient methods are usually chosen over more traditional methods for a number of reasons:

1. The approximate policy can approach a deterministic policy. The action-value estimates will converge to their corresponding true values.

2. It allows for the selection of actions with arbitrary probabilities. This means it is possible to find stochastic optimal policies as well.

3. Policies may be easier to approximate than the action-value function.

4. Policy parameterization allows for prior knowledge about the desired form of the policy to be incorporated.

In addition, policy-based methods offer practical ways of dealing with large, continuous action spaces. Instead of computing learned probabilities for each of the many actions, it is possible to learn the statistics of a probability distribution that represents all possible actions. To produce a policy parameterization, the policy can be defined as the normal probability density over a real-value scalar action, with mean and standard deviation given by parametric function approximators that depend on the state. In this case, the mean is generally approximated as a linear function while the standard deviation must be positive and is usually approximated as the exponential of a linear function. These policy parameters are then learned by following the policy optimization process detailed in the following sections [14].

2.4 Neural Networks

The term "neural network" is based on its use in attempting to find mathematical representations of information processing in biological systems. Similar to biological systems, artificial neural networks are primarily used for statistical pattern recognition. In RL, neural networks are used to represent the policy and value functions and are trained during the update steps.

Neural networks are based on linear combinations of M fixed nonlinear basis functions, $\Psi_j(x)$ and take the form

$$y(x, w) = z\left(\sum_{j=1}^{M} w_j \Psi_j(x)\right) \tag{2.12}$$

where x is the input(s) to the network, w_j are the weighting coefficients and $z(\cdot)$ is the nonlinear activation function, which will be discussed in more detail later in this section. The basis functions are functions of a set number of parameters which are updated during the training steps. Neural networks use basis functions that resemble Eq. (2.12) such that each basis function is itself a nonlinear function of a linear combination of inputs. Thus, neural networks are simply a series of functional transformations.

To compose a neural network, M linear combinations of the input variables $x_1, ..., x_D$ are first constructed as

$$o_j = \sum_{i=1}^{D} w_{ji}^{(1)} x_i + w_{j0}^{(1)} \tag{2.13}$$

where $w_{ji}^{(1)}$ are the weights, $w_{j0}^{(1)}$ are the biases, and $j = 1, ..., M$. The superscript (1) is used to indicate the first "layer" in the network [15].

2.4.1 Activation Functions

The value o_j in Eq. (2.13) resulting from each layer is transformed using a differentiable, nonlinear activation function, $h(\cdot)$:

$$y_j = z(o_j) \tag{2.14}$$

where y_j corresponds to the outputs of the basis functions is Eq. (2.12).

The possible activation functions include:

1. Sigmoid Function: The sigmoid function is a smooth, gradual progression from 0 to 1 and is one of the most common activation functions, especially for linear regression. The equation takes the form [16]

$$z(x) = \frac{1}{1 + e^{-x}} \tag{2.15}$$

and resembles the curve shown in Figure 2.5.

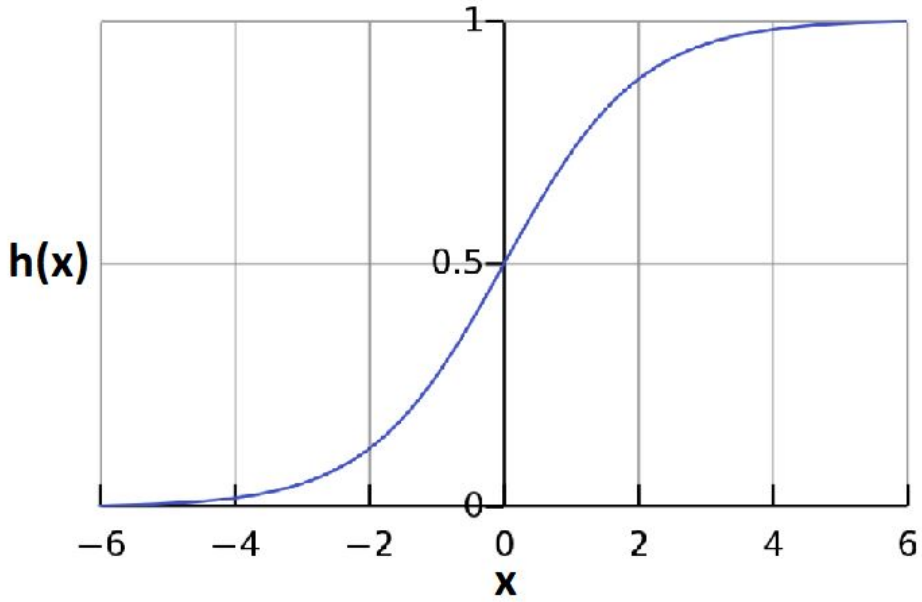


Figure 2.5. Sigmoid Activation Function

2. Hyperbolic Tangent Function: The hyperbolic tangent function resembles the sigmoid function but instead ranges from -1 to 1. This is generally more common than the sigmoid activation function and takes the form

$$z(x) = \tanh(x) = \frac{2}{1 + e^{-2x}} - 1 \tag{2.16}$$

as the name suggests [16].

3. ReLU Activation Function: The ReLU (Rectified Linear Unit) activation function can be simply represented as

$$\begin{aligned} z(x) &= 0 \text{ if } x < 0 \\ &= x \text{ if } x >= 0 \end{aligned} \tag{2.17}$$

The ReLU activation function is the most popular activation function in neural networks because it is efficient and easy to optimize since it doesn't require any complicated calculations. The primary downside of using a ReLU function is that once a neuron becomes zero, it is unlikely it will ever go back to a non-zero value [15].

2.4.2 Layers

The selected activation function can be linearly combined with Eq. (2.13) to give

$$o_k = \sum_{j=1}^{M} w_{kj}^{(2)} z_j + w_{k0}^{(2)} \tag{2.18}$$

where $k = 1, ..., K$ and K is the total number of outputs. This equation corresponds to the second layer of the network. Neural networks can have as many layers (called hidden layers) as necessary to achieve a robust model for the desired relationship it is defining. After each layer is processed, a final activation layer is used to generate the outputs

$$y_k = z(o_k) \tag{2.19}$$

The inputs, hidden layers, and outputs can be seen visualized in Figure 2.6. Here, each input x is applied its respective weights for each desired hidden layer. Between layers, the activation functions are applied. Then, finally, a last activation function is applied to generate the outputs.

Assuming only the two hidden layers as previously described, the overall network function becomes

$$y_k(x, w) = z\left(\sum_{j=1}^{M} w_{kj}^{(2)} h\left(\sum_{i=1}^{D} w_{ji}^{(1)} x_i + w_{j0}^{(1)}\right) + w_{k0}^{(2)}\right) \tag{2.20}$$

where the set of all weights and bias parameters have been grouped together into a vector, w, which are the learned parameters [15].

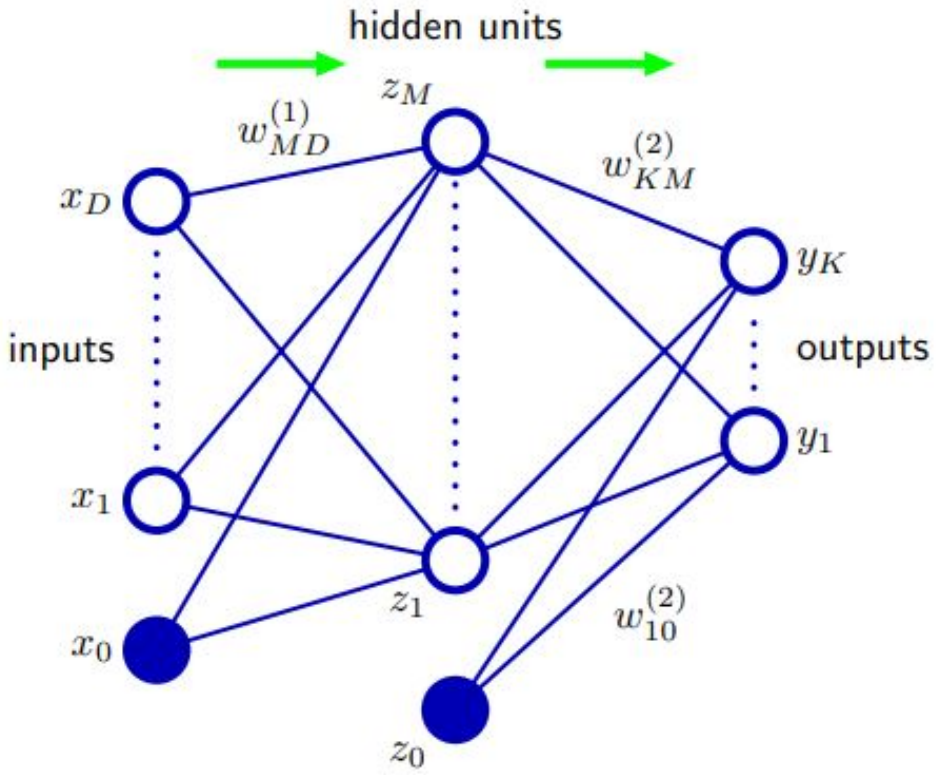


Figure 2.6. Neural Network Diagram - Two Hidden Layers [15]

2.5 Policy Optimization

Building onto the RL foundations, this section goes into greater detail on the policy optimization theory that will be required.

2.5.1 Trust Region Policy Optimization

TRPO, at its simplest, is an iterative process for optimizing policies while guaranteeing a monotonic improvement in the policies. Similar to natural policy gradient methods, TRPO is effective for optimizing large nonlinear policies such as neural networks (where the weights and biases are the search variables). The original work, Reference [17], demonstrated the robust performance of this method and its tendencies to give monotonic improvements with minimal tuning of the hyperparameters.

2.5.1.1 Preliminaries

Policy optimization methods like TRPO can be framed as discounted MDP, defined by the tuple $(\mathcal{S}, \mathcal{A}, P, r, \rho_0, \gamma)$ where $\mathcal{S}$ is a finite set of states, $\mathcal{A}$ is a finite set of actions, $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow \mathbb{R}$ is the transition probability distribution, $r : \mathcal{S} \rightarrow \mathbb{R}$ is the reward function, $\rho_0 : \mathcal{S} \rightarrow \mathbb{R}$ is the distribution of the initial state s_0, and $\gamma \in (0, 1)$ is the discount factor.

With each state/action pair, there is an associated stochastic policy, $\pi : \mathcal{S} \times \mathcal{A} \rightarrow [0, 1]$ to connect the two as well as an expected discount reward,

$$\eta(\pi) = \mathbb{E}_{s_0, a_0, \ldots} \left[\sum_{t=0}^{\infty} \gamma^t r(s_t) \right] \tag{2.21}$$

where $s_0 \sim \rho_0(s_0)$, $a_t \sim \pi(a_t|s_t)$, $s_{t+1} \sim P(s_{t+1}|s_t, a_t)$.

Moving forward, the value function, V_π, will be defined as

$$V_\pi(s_t) = \mathbb{E}_{a_t, s_{t+1}, \ldots} \left[\sum_{t=0}^{\infty} \gamma^l r(s_{t+1}) \right] \tag{2.22}$$

the state-action value function, Q_π, as

$$Q_\pi(s_t, a_t) = \mathbb{E}_{s_{t+1}, a_{t+1}, \ldots} \left[\sum_{t=0}^{\infty} \gamma^l r(s_{t+1}) \right] \tag{2.23}$$

and the advantage function, A_π, as

$$A_\pi(s, a) = Q_\pi(s, a) - V_\pi(s) \tag{2.24}$$

where $t \geqslant 0$.

The policy update step can then be expressed in terms of the advantage with the actions sampled using $a_t \sim \tilde{\pi}(\cdot|s_t)$,

$$\eta(\tilde{\pi}) = \eta(\pi) + \mathbb{E}_{s_0, a_0, \ldots \sim \tilde{\pi}} \left[\sum_{t=0}^{\infty} \gamma^t A_\pi(s_t, a_t) \right] \tag{2.25}$$

Meanwhile, the discounted visitation frequencies can be expressed as

$$\rho_\pi(s) = P(s_0 = s) + \gamma P(s_1 = s) + \gamma^2 P(s_2 = s) + \ldots \tag{2.26}$$

where $s_0 \sim \rho_0$ and the actions are chosen according to π. Equation 2.25 can then be rewritten as a sum over states instead of over timesteps:

$$\eta(\tilde{\pi}) = \eta(\pi) + \sum_s \rho_{\tilde{\pi}}(s) \sum_a \tilde{\pi}(a|s) A_\pi(s,a) \tag{2.27}$$

This can be used to demonstrate that any policy update with a non-negative expected advantage at every state, s, is guaranteed to increase the policy performance, η (or remain constant when the advantage is zero everywhere). However, having some states for which the expected advantage is negative is usually unavoidable. Since the dependency of $\rho_{\tilde{\pi}}(s)$ on $\tilde{\pi}$ makes the previous equation hard to optimize, a local approximation, $L_\pi(\tilde{\pi})$, to η can be used instead

$$L_\pi(\tilde{\pi}) = \eta(\pi) + \sum_s \rho_\pi(s) \sum_a \tilde{\pi}(a|s) A_\pi(s,a) \tag{2.28}$$

with the only difference being that changes in the state visitation density due to changes in the policy are ignored. In most cases, though, the interest will be in optimizing a parameterized policy, π_θ (where $\pi_\theta(a|s)$ is a differential function of the parameter vector, θ) which means that L_π will match η to the first order. That is, for any parameter value θ_0,

$$\begin{aligned} L_{\pi_{\theta_0}}(\pi\theta_0) &= \eta(\pi_{\theta_0}), \\ \bigtriangledown_\theta L_{\pi_{\theta_0}}(\pi_\theta)|_{\theta=\theta_0} &= \bigtriangledown_\theta \eta(\pi_\theta)|_{\theta=\theta_0} \end{aligned} \tag{2.29}$$

This implies that a small enough step $\pi_{\theta_0} \rightarrow \tilde{\pi}$ that improves upon $L_{\pi_{\theta_{old}}}$ will also improve η. It does not, however, suggest an appropriate step size. Reference [18] explored this issue further and found a policy updating scheme that allowed for specific lower bounds on the improvement of η to be provided. The policy update step was defined as

$$\pi_{new}(a|s) = (1-\alpha)\pi_{old}(a|s) + \alpha\pi'(a|s) \tag{2.30}$$

with the lower bound

$$\eta(\pi_{new}) \geqslant L_{\pi_{old}}(\pi_{new}) - \frac{2\epsilon\gamma}{(1-\gamma)^2}\alpha^2 \tag{2.31}$$

where α is a constant and $\epsilon = \max_s |\mathbb{E}_{\alpha\sim\pi'(a|s)}\left[A_\pi(s|a)\right]|$. However, this policy update scheme is not applicable to all general stochastic policy classes.

2.5.1.2 Monotonic Improvement Guarantee for General Stochastic Policies

Equation 2.31 provides a basis for expanding the policy update to all stochastic policy classes, rather than just mixture policies, by replacing α with a measure of the distance between π and $\tilde{\pi}$, and changing ϵ accordingly. In this case, the distance measure is the total variation divergence, defined as $D_{TV}(p||q) = \frac{1}{2}\sum_i |p_i - q_i|$ for discrete probability distributions p and q [19]. For simplicity moving forward, $\max_s D_{TV}(\pi(\cdot|s)||\tilde{\pi}(\cdot|s))$ will be written as $D_{TV}^{max}(\pi, \tilde{\pi})$. So if $\alpha = D_{TV}^{max}(\pi_{old}, \pi_{new})$, then the bound can be written as

$$\eta(\pi_{new}) \geqslant L_{\pi_{old}}(\pi_{new}) - \frac{4\epsilon\gamma}{(1-\gamma)^2}\alpha^2 \tag{2.32}$$

where $\epsilon = \max_{s,a} |A_\pi(s,a)|$. This theorem can then be applied directly to represent the bound in terms of D_{KL}^{max} when applying the known relationship between the total variation divergence and the KL divergence, $D_{TV}(p||q)^2 \leqslant D_{\mathrm{KL}}(p||q)$, giving

$$\eta(\tilde{\pi}) \geqslant L_\pi(\tilde{\pi}) - CD_{\mathrm{KL}}^{max}(\pi, \tilde{\pi}) \tag{2.33}$$

where $C = \frac{4\epsilon\gamma}{(1-\gamma)^2}$. This then provides an approximate policy iteration scheme as described in Algorithm 3 with the guarantee of monotonically improving sequences of policies such that $\eta\pi_0 \leqslant \eta\pi_1 \leqslant \eta\pi_2 \leqslant ....$ To see this, let $M_i(\pi) = L_{\pi_i\pi} - CD_{\mathrm{KL}}^{max}(\pi_i, \pi)$. Then

$$\begin{aligned} \eta(\pi_{i+1}) &\geqslant M_i(\pi_{i+1}) \text{ by Eq.2.33} \\ \eta(\pi_i) &= M_i(\pi_i), \text{therefore,} \\ \eta(\pi_{i+1}) - \eta(\pi_i) &\geqslant M_i(\pi_{i+1}) - M(\pi_i) \end{aligned} \tag{2.34}$$

Maximizing M_i at each iteration guarantees that the true objective η is non-decreasing. TRPO, discussed more in the following subsection, is an approximation to Algorithm 3, which uses a constraint on the KL divergence rather than a penalty to allow for large updates.

Algorithm 3 Policy Iteration Algorithm Guaranteeing Non-Decreasing Expected Return η [17]

Initialize π_0
for i = 0,1,2,...until convergence **do**
    Compute all advantage values $A_\pi(s, a)$,
    Solve the constrained optimization problem
    $\pi_{i+1} = \underset{\pi}{\text{argmax}} \left[L_{\pi_i}(\pi) - C D_{\text{KL}}^{max}(\pi_i, \pi) \right]$
        where $C = \frac{4\epsilon\gamma}{(1-\gamma)^2}$
        and $L_{\pi_i}(\pi) = \eta(\pi_i) + \sum_s \rho_{\pi_i}(s) \sum_a \pi(a|s) A_{\pi_i}(s, a)$
end for

2.5.1.3 Optimization of Parameterized Policies

The last step is to derive a practical algorithm based on the foundations built in the previous subsections. Moving forward, since all policies, π, will be parameterized by θ, the notation will be overloaded to express functions in terms of θ (such as $\eta(\theta) := \eta(\pi_\theta)$).

In the previous subsection, it was shown that $\eta(\theta) \geqslant L_{\theta_{old}}(\theta) - C D_{\text{KL}}^{max}(\theta_{old}, \theta)$ when assuming $\theta = \theta_{old}$. Thus, a maximization of the latter terms provides a guarantee to improve the objective η:

$$\max_\theta \left[L_{\theta_{old}}(\theta) - C D_{\text{KL}}^{max}(\theta_{old}, \theta) \right] \tag{2.35}$$

Instead of using the penalty coefficient, C, which usually would result in very small step sizes, larger steps can be taken by placing a constraint on the KL divergence between the new policy and the old policy instead. This is called a trust region constraint and is formulated as

$$\begin{gathered} \max_\theta L_{\theta_{old}}(\theta) \\ \text{subject to } D_{\text{KL}}^{max}(\theta_{old}, \theta) \leqslant \delta \end{gathered} \tag{2.36}$$

In this form, a constraint would be placed on the KL divergence at every point in space. This large number of constraints is impractical to satisfy so the same idea can be applied by instead considering the average KL divergence:

$$\max_\theta \left[L_{\theta_{old}}(\theta) - C D_{\text{KL}}^{max}(\theta_{old}, \theta) \right] \tag{2.37}$$

This then leads to optimizing as such,

$$\begin{aligned} &\max_{\theta} L_{\theta_{old}}(\theta) \\ &\text{subject to } \overline{D}_{\mathrm{KL}}^{\rho_{\theta_{old}}}(\theta_{old}, \theta) \leqslant \delta \end{aligned} \tag{2.38}$$

2.5.1.4 Practical Algorithm

The theory described in the previous subsections justifies optimizing a surrogate objective with a penalty on the KL divergence. The large penalty coefficient, C, leads to small steps, though, so it would be better to decrease the coefficient. This is where the hard constraint, the bound on the KL divergence, comes into play. Since the constraint on $D_{\mathrm{KL}}^{max}(\theta_{old}, \theta)$ is difficult to meet using numerical optimization, $\overline{D}_{\mathrm{KL}}^{\rho_{\theta_{old}}}(\theta_{old}, \theta)$ is constrained instead. With this in mind, the optimization is performed in distinct steps:

1. Collect a set of state-action pairs along with estimates of their Q-values
2. Averaging over samples, construct the estimated objective and constraint
3. Solve the constrained optimization problem to update the policy's parameter vector θ

The original paper on TRPO by Schulmann et. al. [17] demonstrated the robustness of this method with a number of designed experiments.

2.5.2 Proximal Policy Optimization

TRPO, considered easy to tune, serves as the foundation of the RL applied in this research. PPO has some of the benefits of same performance benefits of TRPO, but is even simpler to implement, can be applied more generically, and has better sample complexity. Unlike TRPO which performs one gradient update per data sample, PPO employs multiple epochs of minibatch updates, alternating between sampling data from the policy and optimizing from minibatches of the sample, to ensure quick convergence. PPO, though it uses only first-order optimization, has been shown to attain similar reliable performance and data efficiency as TRPO.

2.5.2.1 Policy Gradient Methods

In order to better understand how PPO works, it is necessary to first understand policy gradient methods in general. Policy gradient methods work by computing an estimator

of the policy gradient and plugging it into a stochastic gradient ascent algorithm. The most popular gradient estimator has the form

$$\hat{g} = \hat{E}_t[\nabla_\theta \log \pi_\theta(a_t|s_t)\hat{A}_t] \tag{2.39}$$

where $\hat{E}_t$ is the expectation over the batch of samples, π_θ is a stochastic policy (θ representing the policy parameters), and $\hat{A}_t$ is an estimator of the advantage function at time t. An objective function is then constructed whose gradient is the policy gradient estimator, $\hat{g}$, obtained by differentiating the objective, or loss. [2],

$$L^{\text{PG}}(\theta) = \hat{E}_t[\log \pi_\theta(a_t|s_t)\hat{A}_t] \tag{2.40}$$

Differentiation of this loss equation implies that it is the focal point of the optimization. PPO aims to optimize a set of NE strategies such that they result in the lowest possible loss values.

2.5.2.2 Clipped Objective

Trust Region Methods, like TRPO, can be applied within PPO to constrain the policy update based on the KL divergence. This gives the objective function

$$\begin{gathered} \max_\theta \hat{\mathbb{E}}_t \left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)} \hat{A}_t \right] \\ \text{subject to } \hat{\mathbb{E}}_t \left[\mathrm{D}_{KL} \left[\pi_{\theta_{old}}(\cdot|s_t), \pi_\theta(\cdot|s_t) \right] \right] \leqslant \delta \end{gathered} \tag{2.41}$$

Without the constraint, maximization of the above equation would result in an excessively large polcy update. PPO looks to restrict the change in the objective/loss, function $L^{\text{CLIP}}(\theta)$, by clipping the value at every update step,

$$L^{\text{CLIP}}(\theta) = \hat{E}_t[\min(l_t(\theta)\hat{A}_t, \text{clip}(l_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t] \tag{2.42}$$

in which l_t is the probability ratio

$$l_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)} \tag{2.43}$$

where ϵ is a hyperparameter, which is usually selected to be around 0.2 since this results in the policy changing by no more than 20% in each update. The second term, $\text{clip}(l_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t$, modifies the surrogate objective by clipping the probability ratio.

This essentially removes the incentive for moving r_t outside of the interval $[1-\epsilon, 1+\epsilon]$. Finally, the minimum of the clipped and un-clipped objective is taken so that the final objective is a lower bound on the un-clipped objective, effectively ignoring the change in probability ratio when it would make the objective improve. [2]

2.5.2.3 Adaptive KL Penalty Coefficient

In addition to clipping the surrogate objective, or alternatively to it, an adaptive penalty on the KL divergence can be used such that a target value for the divergence, d_{targ}, is achieved each policy update. In general, this performs worse than the clipped surrogate objective when applied individually, but both can be applied simultaneously to further encourage proper optimization of the policies.

Still implemented over several epochs of mini-batches, the objective is penalized as

$$L^{\text{KLPEN}}(\theta) = \hat{E}_t \left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)} \hat{A}_t - \beta \mathrm{D}_{KL} \left[\pi_{\theta_{old}}(\cdot|s_t), \pi_\theta(\cdot|s_t) \right] \right] \tag{2.44}$$

The divergence can then be calculated using

$$D = \hat{E}_t \left[\mathrm{D}_{KL} \left[\pi_{\theta_{old}}(\cdot|s_t), \pi_\theta(\cdot|s_t) \right] \right] \tag{2.45}$$

If $D < D_{targ}/1.5$ then β is set to $\beta/2$. If $D > D_{targ} \times 1.5$ then β is set to $\beta \times 2$. D_{targ} here is a hyperparameter provided as an ideal value for the KL divergence. This updated β value is then used in the next policy update.

2.5.2.4 Algorithm

The surrogate loss from the previous section can be computed and differentiated, using L^{CLIP} instead of L^{PG}. Multiple steps of stochastic gradient ascent are then performed on this objective.

If using a neural network architecture that shares parameters between the policy and value function, then a loss function must be used that combines the policy surrogate and a value function error term. In addition, an entropy bonus can be added to encourage exploration of the state and action spaces (as suggested in References [20] and [21]). Combined together, the objective to maximize becomes

$$L_t^{CLIP+VF+S}(\theta) = \hat{\mathbb{E}}_t \left[L_t^{CLIP}(\theta) - c_1 L_t^{VF}(\theta) + c_2 S \left[\pi_\theta \right] (s_t) \right] \tag{2.46}$$

where S is the entropy bonus, c_1 and c_2 are coefficients, and L_t^{VF} is the squared-error loss term $(V_\theta(s_t) - V_t^{targ})^2$.

One implementation of PPO demonstrating how it is well-suited for use with recurrent neural networks, as seen in Reference 12, runs the policy for T time-steps (where T is much less than the episode length), and uses the collected samples for an update. This style requires an advantage estimator that does not look beyond time-step T

$$\hat{A}_t = -V(s_t) + r_t + \gamma r_{t+1} + ... + \gamma^{T-t+1} r_{T+1} + \gamma^{T-t} V(s_T) \tag{2.47}$$

in which t specifies the time index in $[0, T]$, within a given length-T trajectory segment, and where V is the learned state-value function included to reduce the variance. [2]

A PPO algorithm [2] that uses fixed-length trajectory segments is shown below. Each iteration, each of N (parallel) actors collect T time-steps of data. Then the surrogate loss on these NT time-steps of data is constructed and optimized for K epochs using the Adam optimizer [22], explained further in Section 2.5.4.

Algorithm 4 PPO, Actor-Critic Style [2]

for iteration=1,2...**do**
    for actor=1,2...,N **do**
        Run policy $\pi_{\theta_{old}}$ in environment for T time steps
        Compute advantage estimates $\hat{A}_1, ..., \hat{A}_T$
    end for
    Optimize surrogate L wrt θ, with K epochs and mini-batch size $M \leqslant NT$
    $\theta_{old} \leftarrow \theta$
end for

The original authors demonstrated PPO's advantages over other algorithms and showed that the clipped objective was most effective at optimizing the policy.

2.5.3 Generalized Advantage Estimation

One primary hindrance in optimizing the objective in policy optimization is the long time delay between actions and their positive - or negative - effects on the reward, otherwise known as the credit assignment problem. This is where the value function, as mentioned in the previous subsection, comes into play. Value functions, which help to estimate how a policy should be improved, can effectively represent the quality of an action before the delayed reward is given.

The GAE was proposed as a method for optimizing the value function in a way that reduces variance while maintaining a tolerable level of bias, effects that are usually traded in related optimization approaches [23].

2.5.3.1 Preliminaries

Inspecting Eq. (2.39), it can be surmised that the policy gradient is not just a function of the policy parameters, but the advantage as well. The advantage function measures whether or not the action is better or worse than the policy's default behavior. This means that, in order to ensure that the policy is being improved upon, the gradient term should point in the direction of increasing $\pi_\theta(a_t|s_t)$ if and only if $A^\pi(s_t, a_t)$ is greater than zero.

The discount factor, γ, introduced in Section 2.5.1.1, is used to reduce the variance by downweighting any reward that correspond to delayed effects. This, however, will result in added bias. The discounted approximation to the policy gradient can then be defined as [23]

$$g^\gamma := \mathbb{E}_{s_{0:\infty}, a_{0:\infty}} \left[\sum_{t=0}^{\infty} A^{\pi,\gamma}(s_t, a_t) \bigtriangledown_\theta \log \pi_\theta(a_t|s_t) \right] \tag{2.48}$$

2.5.3.2 Advantage Function Estimation

To better understand the advantage estimator, specifically as presented in Eq. (2.47), one can start by formulating the TD residual of the discounted value function as

$$\delta_t^V = r_t + \gamma V(s_{t+1}) - V(s_t) \tag{2.49}$$

This is essentially an estimate of the advantage of the action, a_t. So, by definition, $\hat{A}_t^{(k)}$ is just the sum of k of thse δ terms:

$$\hat{A}_t^{(1)} := \delta_t^V = -V(s_t) + r_t + \gamma V(s_{t+1}) \tag{2.50}$$

$$\hat{A}_t^{(2)} := \delta_t^V + \gamma\delta_{t+1}^V = -V(s_t) + r_t + \gamma r_{t+1} \gamma^2 V(s_{t+2}) \tag{2.51}$$

$$\hat{A}_t^{(3)} := \delta_t^V + \gamma\delta_{t+1}^V + \gamma^2\delta_{t+1}^V = -V(s_t) + r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \gamma^3 V(s_{t+3}) \tag{2.52}$$

which leads to

$$\hat{A}_t^{(k)} := \sum_{l=0}^{k-1} \gamma^l \delta_{t+1}^V = -V(s_t) + r_t + \gamma r_{t+1} + ... + \gamma^{k-1} r_{t+k-1} + \gamma^k V(s_{t+k}) \tag{2.53}$$

$\hat{A}_t^{(k)}$ here involves a k-step estimate of the returns, minus a baseline term $-V(s_t)$. This suggests that, as $k \to \infty$, the bias becomes smaller since the term $\gamma^k V(s_{t+k})$ becomes more heavily discounted, and the term $-V(s_t)$ does not affect the bias. Taking $k \to \infty$ gives

$$\hat{A}_t^{(\infty)} = \sum_{l=0}^{\infty} \gamma^l \delta_{t+1}^V = -V(s_t) + \sum_{l=0}^{\infty} \gamma^l r_{t+1} \tag{2.54}$$

which is simply the empirical returns minus the value function baseline.

From this, the GAE (GAE) (γ, λ) is defined as the exponentially-weighted average of these k-step estimators

$$\begin{aligned}
\hat{A}_t^{\mathrm{GAE}(\gamma,\lambda)} &:= (1-\lambda)\left(\hat{A}_t^{(1)} + \lambda \hat{A}_t^{(2)} + \lambda^2 \hat{A}_t^{(3)} + ...\right) \\
&= (1-\lambda)(\delta_t^V + \lambda(\delta_t^V + \gamma \delta_{t+1}^V) + \lambda^2(\delta_t^V + \gamma \delta_{t+1}^V + \gamma^2 \delta_{t+1}^V) + ...) \\
&= (1-\lambda)(\delta_t^V(1 + \lambda + \lambda^2 + ...) + \gamma \delta_{t+1}^V(\lambda + \lambda^2 + \lambda^3 + ...) + \gamma^2 \delta_{t+2}^V(\lambda^2 + \lambda^3 + \lambda^4 + ...) + ...) \\
&= (1-\lambda)\left(\delta_t^V\left(\frac{1}{1-\lambda}\right) + \gamma \delta_{t+1}^V\left(\frac{\lambda}{1-\lambda}\right) + \gamma^2 \delta_{t+2}^V\left(\frac{\lambda^2}{1-\lambda}\right) + ...\right) \\
&= \sum_{l=0}^{\infty} (\gamma\lambda)^l \delta_{t+1}^V
\end{aligned} \tag{2.55}$$

There are two notable special cases of this formula, obtained by setting $\lambda = 0$ and $\lambda = 1$:

$$\mathrm{GAE}(\gamma, 0): \ \hat{A}_t := \delta_t \quad = r_t + \gamma V(s_{t+1}) - V(s_t) \tag{2.56}$$

$$\mathrm{GAE}(\gamma, 1): \ \hat{A}_t := \sum_{l=0}^{\infty} \gamma^l \delta_{t+1} = \sum_{l=0}^{\infty} \gamma^l r_{t+1} - V(s_t) \tag{2.57}$$

$\mathrm{GAE}(\gamma, 0)$ usually induces bias but has a low variance, while $\mathrm{GAE}(\gamma, 1)$ has less bias but a high variance due to the sum of the terms. Setting λ between zero and one for the GAE makes a compromise between bias and variance, controlled by λ. γ, while it most importantly determines the scale of the value function, also contributes to this bias/variance trade-off. While setting $\lambda < 1$ introduces bias only when the value function is inaccurate, setting $\gamma < 1$ introduces bias regardless. Due to this, the original authors found that the best value for λ is much lower than the best value for γ. Using the GAE, the discounted policy gradient can now be written as [23]

$$g^{\gamma} = \mathbb{E}\left[\sum_{t=0}^{\infty} \bigtriangledown_{\theta} \log \pi_{\theta}(a_t|s_t) \sum_{l=0}^{\infty} (\gamma\lambda)^l \delta_{t+1}^V\right] \tag{2.58}$$

2.5.3.3 Interpretation As Reward Shaping

Inspecting Eq. (2.58), it can be seen that both γ and λ are effectively implemented as discount factors after performing a reward shaping transformation on the MDP.

Reward shaping [24], refers to the transformation of the reward function of an MDP as

$$\tilde{r}(s, a, s') = r(s, a, s') + \gamma\Psi(s') - \Psi(s) \tag{2.59}$$

where $\tilde{r}$ is the transformed reward function and Ψ is an arbitrary scalar-valued function on the state space, $S \to \mathbb{R}$. This transformation results in the discounted advantage function remaining unchanged for any policy. This can be proven by manipulating the discounted sum of the rewards of a trajectory originating from state, s_t, as

$$\sum_{l=0}^{\infty} \gamma^l \tilde{r}(s_{t+l}, a_t, s_{t+l+1}) = \sum_{l=0}^{\infty} \gamma^l r(s_{t+l}, a_{t+l}, s_{t+l+1}) - \Psi(s_t) \tag{2.60}$$

The value and advantage functions of the transformed MDP can then be defined as

$$\begin{aligned} \tilde{Q}^{\pi,\gamma}(s, a) &= Q^{\pi,\gamma}(s, a) - \Psi(s) \\ \tilde{V}^{\pi,\gamma}(s, a) &= V^{\pi,\gamma}(s, a) - \Psi(s) \\ \tilde{A}^{\pi,\gamma}(s, a) = (Q^{\pi,\gamma}(s, a) - \Psi(s)) - (V^{\pi,\gamma}(s, a) - \Psi(s)) &= A^{\pi,\gamma}(s, a) \end{aligned} \tag{2.61}$$

Reward shaping can then be used to get a policy gradient estimate by constructing the policy gradient estimators that use discounted sums of the shaped rewards, $\tilde{r}$. Applying λ as, essentially, a second discount factor results in a "steeper" discount $\gamma\lambda$ when $0 \leqslant \lambda \leqslant 1$. In doing so, it's easy to see that the shaped reward from Eq. (2.60) is equal to the Bellman residual term δ^V from Eq. (2.49) when $\Psi = V$ [23]. Doing so gives

$$\sum_{l=0}^{\infty} (\lambda\gamma)^l \tilde{r}(s_{t+l}, a_t, s_{t+l+1}) = \sum_{l=0}^{\infty} (\lambda\gamma)^l \delta_{t+1}^V = \hat{A}_t^{GAE(\gamma,\lambda)} \tag{2.62}$$

Hence, by considering the $\gamma\lambda$-discounted sum of shaped rewards, the GAE from the previous section is obtained exactly [23].

Exploring the effect of the shaping transformation in more detail, a response function, χ can be introduced and defined as

$$\chi(l; s_t, a_t) = \mathbb{E}\left[r_{t+l} | s_t, a_t\right] - \mathbb{E}\left[r_{t+1} | s_t\right] \tag{2.63}$$

Note that $A^{\pi,\gamma}(s,a) = \sum_{l=0}^{\infty} \gamma^l \chi(l;s,a)$ so the response function decomposes the advantage function across timesteps. This allows for the temporal credit assignment problem to be properly quantified such that long range dependencies between actions and rewards correspond to non-zero values of the response function when $l \geqslant 0$. The discounted policy gradient estimator can then be expressed as

$$\bigtriangledown_\theta \log_{\pi_\theta}(a_t|s_t) A^{\pi,\gamma}(s_t,a_t) = \bigtriangledown_\theta \log_{\pi_\theta}(a_t|s_t) \sum_{l=0}^{\infty} \gamma^l \chi(l;s_t,a_t) \tag{2.64}$$

in which the error will be small if χ decays as l increases, as in the effect of an action is "forgotten" after about $1/(1-\gamma)$ time steps.

If the reward function, $\tilde{r}$, were obtained using $\Psi = V^{\pi,\gamma}$, then $\mathbb{E}\left[\tilde{r}_{t+l}|s_t,a_t\right] = \mathbb{E}\left[\tilde{r}_{t+1}|s_t\right] = 0$ for $l > 0$. Essentially, the response function would only be non-zero at $l = 0$ so the shaping transformation would turn a temporally extended response into an immediate response. This suggests that the rewards can be reshaped using V to shrink the temporal extent of the response function while the "steeper" discount $\gamma\lambda$ cuts off the noise resulting from long delays [23].

2.5.3.4 Value Function Estimation

The final step is applying GAE to the estimation of the value function. When employing a nonlinear function approximator to represent the value function, the easiest method for estimating it is to approach it as a nonlinear regression problem:

$$\min_\theta \sum_{n=1}^{N} ||V_\theta(s_n) - \hat{V}_n||^2 \tag{2.65}$$

where $\hat{V}_t = \sum_{l=0}^{\infty} \gamma^l r_{t+1}$ is the discounted sum of the rewards indexed by n over all timesteps in a batch of trajectories. When applying this to the trust region problem, the standard deviation is defined as $\sigma^2 = \frac{1}{N} \sum_{n=1}^{N} ||V_{\theta_{out}}(s_n) - \hat{V}_n||^2$. Then the optimization problem can be defined instead as

$$\begin{aligned} &\min_\theta \sum_{n=1}^{N} ||V_\theta(s_n) - \hat{V}_n||^2 \\ &\text{subject to } \frac{1}{N} \sum_{n=1}^{N} \frac{||V_\theta(s_n) - V_{\theta_{old}}(s_n)||^2}{2\sigma^2} \leqslant C \end{aligned} \tag{2.66}$$

Thus, like when updating the policy, the value function can be updated by applying a similar KL divergence constraint. [23]

2.5.4 Adam Optimizer

When training the policy and value functions, often represented as neural networks in policy optimization, the Adam optimizer is the most common method for doing so. Adam is an algorithm for first-order gradient-based optimization of stochastic objective functions based on adaptive estimates of lower-order moments. The method is straightforward to implement, has low memory requirements, is computationally efficient, is well suited for large problems, and is invariant to diagonal re-scaling of the gradients [22].

2.5.4.1 Algorithm

Algorithm 5 shows the pseudo-code for the Adam optimizer. The algorithm starts with $f(\theta)$, a noisy objective function differentiable with respect to θ, the policy parameters. The goal is to minimize the expected value of f, $\mathbb{E}[f(\theta)]$. With $g_t = \bigtriangledown_\theta f_t(\theta)$ as shown in line 10, the gradient of f is defined at time step t. The algorithm then presents the steps to update the exponential moving averages (m_t) and the squared gradient (v_t) where the hyperparemters $\beta_1, \beta_2 \in [0, 1)$ control the exponential decay rates of the averages. These estimate the 1st moment (mean) and 2nd raw moment (variance) of the gradient, respectively. Because, by initializing the averages at zero, they become biased towards zero, $\hat{m}$ and $\hat{v}$ are used to represent the necessary bias-corrected values. The last step in the algorithm then uses these bias-corrected averages to update the policy parameters.

Algorithm 5 Adam Pseudo Code. Good default settings are $\alpha = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-8}$. [22]

Require: α: Stepsize
Require: β_1,$\beta_2 \in [0, 1)$: Exponential decay rates for the moment estimates
Require: $f(\theta)$: Stochastic objective function with parameters θ
Require: θ_0: Initial parameter vector
    $m_0 \leftarrow 0$ (Initialize 1st moment vector)
    $v_0 \leftarrow 0$ (Initialize 2nd moment vector)
    $t \leftarrow 0$ (Initialize time step)
    while θ_t not converged **do**
        $t \leftarrow t + 1$
        $g_t \leftarrow \bigtriangledown_\theta f_t(\theta_{t-1})$ (Get gradients w.r.t. stochastic objective at time step t)
        $m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$ (Update biased first moment estimate)
        $v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$ (Update biased second raw moment estimate)
        $\hat{m}_t \leftarrow m_t / (1 - \beta_1^t)$ (Compute bias-corrected first moment estimate)
        $\hat{v}_t \leftarrow v_t / (1 - \beta_2^t)$ (Compute bias-corrected second raw moment estimate)
        $\theta_t \leftarrow \theta_{t-1} - \alpha \cdot \hat{m}_t / (\sqrt{\hat{v}_t} + \epsilon)$ (Update parameters)
    end while
    θ_t (Resulting parameters)

2.5.4.2 Update Rule

An important consideration in the update step of the Adam algorithm is the choice of step sizes. Assuming $\epsilon = 0$, the most effective step in parameter space, at time step t, is $\triangle_t = \alpha \cdot \hat{m}_t / \sqrt{\hat{v}_t}$ with upper bound $| \triangle_t | \leqslant \alpha \cdot (1 - \beta_1) / \sqrt{1 - \beta_2}$ when $(1 - \beta_1) > \sqrt{1 - \beta_2}$ and $| \triangle_t | \leqslant \alpha$ otherwise. This is the equivalent of establishing a trust region around the current parameter value, beyond which the current gradient estimate does not provide sufficient information. This allows for α to be set in advance since the viable parameter space is often known.

2.5.4.3 Initializing Bias Correction

As discussed in Section 2.5.4.1, bias correction terms are required since initializing the moving averages to zero inherently induces bias. The approach that is taken to accomplish this is unclear, though, as of yet. This section will show the derivation of the second

moment estimate (the derivation of the first moment estimate is completely analogous) in an effort to show how the bias correction term is calculated.

As used in Algorithm 5, let g be the gradient of the stochastic objective f with its second raw moment estimated using an exponential moving average of the squared gradient, with decay rate β_2. Then $g_1, ..., g_T$ are the gradients at subsequent time steps, each drawn from the underlying gradient distribution $g_t \sim p(g_t)$. As mentioned, the initialization of the exponential moving average as $v_0 = 0$ is what necessitates the bias correction. The update to this initialization, at time step t, is then performed by $v_t = \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$ where g_t^2 indicates the element-wise square $g_t \odot g_t$. This can be written as a function of the gradients at all previous time steps:

$$v_t = (1 - \beta_2) \sum_{i=1}^{t} \beta_2^{t-i} \cdot g_i^2 \tag{2.67}$$

From this, the relationship between $\mathbb{E}[v_t]$ and the true second moment $\mathbb{E}[g_t^2]$ will give the resulting bias correction term:

$$\begin{aligned} \mathbb{E}[v_t] &= \mathbb{E}\left[(1 - \beta_2) \sum_{i=1}^{t} \beta_2^{t-i} \cdot g_i^2\right] \\ &= \mathbb{E}[g_t^2] \cdot (1 - \beta_2) \sum_{i=1}^{t} \beta_2^{t-i} + \zeta \\ &= \mathbb{E}[g_t^2] \cdot (1 - \beta_2^t) + \zeta \end{aligned} \tag{2.68}$$

where $\zeta = 0$ if the true second moment, $\mathbb{E}[g_i^2]$, is stationary. Otherwise ζ is chosen as a small value since β_1 can be chosen such that the moving average selects small weights for gradients too far in the past. This leaves the term $(1 - \beta_2^t)$ as the bias correction.

2.5.4.4 Convergence Analysis

To properly analyze the convergence of the Adam optimizer, a prediction will be made for the parameter θ_t which will then be evaluated using the previously unknown cost function f_t. Since the nature of the sequence of cost functions $f_1(\theta), f_2(\theta), ..., f_T(\theta)$ is unknown in advance, the algorithm is instead evaluated using the regret (the sum of the difference between the online prediction $f_t(\theta_t)$ and the best fixed point parameter $f_t(\theta^*)$

from a feasible set χ for all previous steps). This regret is formulated as

$$R(T) = \sum_{t=1}^{T} [f_t(\theta_t) - f_t(\theta^*)] \tag{2.69}$$

where $\theta^* = \min_{\theta \in \chi} \sum_{t=1}^{T} f_t(\theta)$. The original authors [22] showed that Adam has $O(\sqrt{T})$ regret bound, which is comparable to the best known bound for general convex online learning problems.

To simplify notation moving forward, $g_t \triangleq \bigtriangledown f_t(\theta_t)$ and $g_{t,i}$ is the i^{th} element. $g_{1:t,i} \in \mathbb{R}^t$ is defined as a vector that contains the i^{th} dimension of the gradients over all iterations until $t, g_{t:t,i} = [g_{1,i}, g_{2,i}, ..., g_{t,i}]$. In addition, $\gamma \triangleq \frac{\beta_1^2}{\sqrt{\beta_2}}$ is also defined. Using these definitions, the following theorem holds when the learning rate, α_t, is decaying at a rate of $t^{-1/2}$ and first moment running average coefficient, $\beta_{1,t}$, decays exponentially with λ, which is typically close to 1.

Theorem 1 *[22] Assume that the function f_t has bounded gradients, $|| \bigtriangledown f_t(\theta)||_2 \leqslant G, || \bigtriangledown f_t(\theta)||_\infty \leqslant G_\infty$ for all $\theta \in R^d$ and distance between any θ_t generated by Adam is bounded, $||\theta_n - \theta_m||_2 \leqslant D, ||\theta_m - \theta_n||_\infty \leqslant D_\infty$ for any $m, n \in \{1, ..., T\}$, and $\beta_1, \beta_2 \in [0, 1)$ satisfy $\frac{\beta_1^2}{\sqrt{\beta_2}} < 1$. Let $\alpha_t = \frac{\alpha}{\sqrt{t}}$ and $\beta_{1,t} = \beta_1 \lambda^{t-1}, \lambda \in (0, 1)$. Adam achieves the following guarantee for all $T \geqslant 1$.*

$$R(T) \leqslant \frac{D^2}{2\alpha(1-\beta_1)} \sum_{i=1}^{d} \sqrt{T \hat{v}_{T,i}} + \frac{\alpha(1+\beta_1)G_\infty}{(1-\beta_1)\sqrt{1-\beta_2}(1-\gamma)^2} \sum_{i=1}^{d} ||g_{1:T,i}||_2 + \sum_{i=1}^{d} \frac{D_\infty^2 G_\infty \sqrt{1-\beta_2}}{2\alpha(1-\beta_1)(1-\lambda)^2}$$

The above theorem implies that when the data features are sparse and bounded gradients, the summation term can be much smaller than its upper bound $\sum_{i=1}^{d} ||g_{1:T,i}||_2 << dG_\infty \sqrt{T}$. From this, the Adam optimization method can achieve $O(\log d\sqrt{T})$, an improvement over $O(\sqrt{dT})$ for the non-adaptive method. The fact that $\beta_{1,t}$ decays towards zero matches previous empirical findings and suggest that reducing the momentum coefficient towards the end of the training can help to improve convergence. Finally, the corollary below shows that the average regret of Adam converges properly.

Corollary 1.1 *[22] Assume that the function f_t has bounded gradients, $|| \bigtriangledown f_t(\theta)||_2 \leqslant G, || \bigtriangledown f_t(\theta)||_\infty \leqslant G_\infty$ for all $\theta \in R^d$ and distance between any θ_t generated by Adam is bounded, $||\theta_n - \theta_m||_2 \leqslant D$, $||\theta_m - \theta_n||_\infty \leqslant D_\infty$ for any $m, n \in \{1, ..., T\}$. Adam achieves the following guarantee, for all $T \geqslant 1$.*

$$\frac{R(T)}{T} = O\left(\frac{1}{\sqrt{T}}\right)$$

This result can be achieved by using the Theorem above and $\sum_{i=1}^{d} ||g_{1:T,i}||_2 \leqslant dG_\infty\sqrt{T}$. Thus, $\lim_{T\to\infty} \frac{R(T)}{T} = 0$.

2.6 Previous Work

This section summarizes the previous work in astronautics related to the proposed research. The literature reviewed helped to determine the approach that proved to be most useful in this research.

2.6.1 Sensor Tasking

Sensor tasking is a subject that is discussed often and researched constantly, but has only recently come close to being truly optimized. Reference [25] does a good job of summarizing the Air Force's Space Surveillance Network's (SSN) sensor tasking strategy as of 1992, but no research has been published with public access on the matter since then. However, a number of papers have been published detailing new and improved strategies that the Air Force could implement.

For instance, in 2010, Erwin et al. published their work [26] on dynamic sensor tasking that establishes a simple benchmark problem for use to frame future research efforts in estimation and resource management for SSA applications. An extended Kalman filter (EKF) was used to provide state estimates of all tracked objects. These estimates and their associated covariances were used to execute a closed-loop sensor tasking approach to determine which sensors would track which objects.

Williams et. al published their work [27] on coupling estimation and sensor tasking. The purpose of the paper was to examine how this coupling can amplify performance differences between estimator types. This effect was investigated by modeling a simplified satellite-tracking scenario, in which the number of objects to track greatly outnumbers the sensor resources available to track them. The estimators used were an EKF and an unscented Kalman filter (UKF), while the tasking methods were based on an information-theoretic approach using Fisher information gain (FIG) [28] and a new stability approach using largest Lyapunov exponent estimation (LLE) [29]. They found that, for both filters, the LLE-based tasking provided more accurate estimates of both the satellite state and uncertainty.

Adurthi et al. [30] expanded upon Williams et al's work to apply Fisher Information and Mutual Information [31] to sensor tasking through the use of the Conjugate Unscented

Transformation (CUT) [32]. This method proved to be accurate as well at optimally tasking sensors, especially in the presence of non-Gaussian covariances of the spacecraft state estimates. Similarly, Gualdoni and DeMars found success [33] in utilizing information statistics [34] for handling multiple observations.

While each of the above methods have proven useful in their specific applications, no research has been published on their applicability to tracking maneuvering objects. This objective was solved, however, by Goff et al. in reference [35]. This paper demonstrates an information-based tasking strategy to track non-cooperative satellites using ground-based radars in which the sensors are tasked based on the Shannon information [36] of the state estimate. The maneuvers of the spacecraft to be tracked were estimated by using a covariance inflation filter-through approach within an Interacting Multiple Model framework, or IMM, for both continuous and instantaneous maneuvers. Figure 2.7 demonstrates this idea in the form of a flowchart where the red boxes represent propagation steps, the yellow boxes represent decisions within the strategy, and the green boxes are the options for each decision.

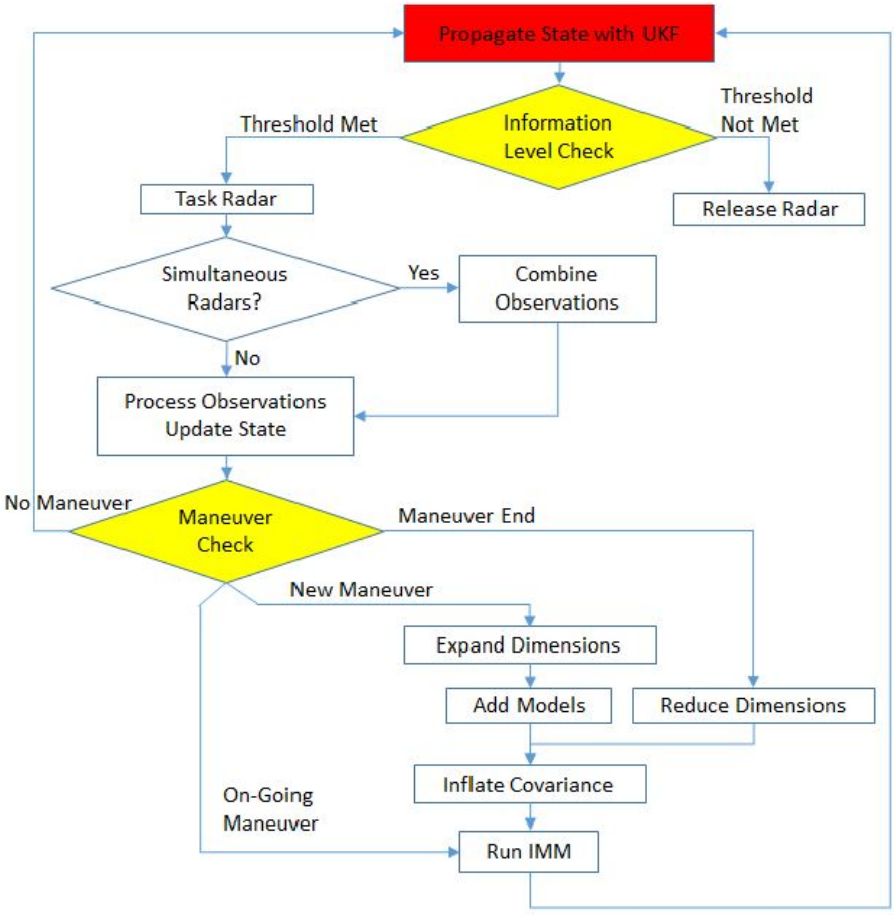


Figure 2.7. Dynamic Tasking Strategy Flowchart [35]

It was found that adaptive information-based tasking was successful in maintaining accurate state knowledge of maneuvering spacecraft. However, IMM approaches are less than ideal because of the required *a priori* information. This leaves room for additional improvements in the area of sensor tasking for tracking uncooperative maneuvering spacecraft.

2.6.1.1 Search Theory

Search theory is the study of an individual's optimal strategy for locating an opponent when choosing from a set of potential moves of random quality under the assumption that a delay in making a decision is costly. The result is an optimal strategy balancing the cost of delay with the value of trying again. Benkoski et al. [37] survey the field of search theory literature, which has not changed much since its publication in 1991. More recently, Chen et al. [38] describe the application of search theory to sensor management

for tracking evasive UAVs. This application translates well to the use of search theory in tracking uncooperative satellites.

As in PE game theory, search allocation games model two adversarial players. At the initial time, the searcher has a total energy constraint E. Using this total energy, the searcher has to allocate resources in the search space to detect the evader, which has an initial energy e_0. The flow of the subsequent game follows these steps:

1. At time k, the searcher obtains the information about the evader's position and their residual energy. At the same time, the evader is informed of the searcher's residual budget.

2. The evader then makes a decision to move from its current cell i to cell j in its neighborhood of cells $N(i)$ in a probabilistic manner, spending $e(i, j)$ energy to do so.

3. The searcher hypothesizes the evader's next cell and allocates resources accordingly, taking their own residual energy into account.

   (a) If n amount of resource is allocated to search cell i, then the searcher can detect the evader with probability $1 - q_i(n)$, with $q_i(n)$ being the miss probability modeled by

   $$q_i(n) = e^{-\psi_i n} \tag{2.70}$$

   where ψ_i is an indicator of how effective it is to spend resources to search in the i-th cell.

4. This process is then repeated until the evader is found.

The actions of the evader are determined slightly differently. Let p_i be the probability that the evader chooses cell i for its hiding location with cost c_i to get to there. With search resources allocated to cell i

$$\psi_i = \frac{1}{\psi_i}[\log \frac{r_i}{\rho}]^+ \tag{2.71}$$

where ζ_i is the value of the game when the evader is located in cell i and the searcher allocates ξ_i resource on it with miss probability criterion. For a single stage of the game, it can be modeled as a minmax problem

$$min_{\xi_i} max_{p_i} \sum_{i \in \mathscr{A}} p_i r_i e^{-\psi_i \xi_i} \tag{2.72}$$

subject to

$$p_i \geqslant 0, \sum_{i \in \mathscr{A}} p_i = 1 \tag{2.73}$$

$$\xi_i \geqslant 0, \sum_{i \in \mathscr{A}} c_i \xi_i = E \tag{2.74}$$

This results in a unique solution given by Eq. (2.71) and

$$p_i = \begin{cases} \frac{c_i/\psi_i}{\sum_{j, p \leqslant z_j} c_j/\psi_j} & \text{if } \rho \leqslant \zeta_i \\ 0 & \text{if } \rho > \zeta_i \end{cases} \tag{2.75}$$

where $[x]^+ = max\{x, 0\}$ and ρ is found from

$$\sum_{i \in \mathscr{A}} \frac{c_i}{\psi_i} [\log \frac{\zeta_i}{\rho}]^+ = E \tag{2.76}$$

This result can then be applied to similar multi-stage games. However, adaptations would need to be made to apply this to the proposed research. Most importantly, neither player will have knowledge about the resource availability of their opponent. The same search theory ideas and approach could prove useful, though, in weighing the cost of delay with resource utilization.

2.6.2 Game Theory Applied to Space Surveillance

Game theoretical methods have been applied for over a century, all starting with the prisoner's dilemma problem. Game theory is now applied to a wide range of behavioral relationships and has found a niche within the space situational awareness field. Most of this work has been published in joint papers from the Air Force Research Lab (AFRL) and Intelligent Fusion Technology, Inc (I-Fusion). Their work, like the vast majority of research in game theory applications for SSA including this dissertation, has focused on PE. Early work on problems of this type modeled the environment geometrically [40]. More recently, AFRL and I-Fusion have published a number of papers [41–43] on applying PE to sensor management. In each paper, the pursuer is represented by a space-based observer/sensor (SBSS) while the evader is represented by a low-thrust geosynchronous satellite being tracked.

The presented PE approach provides a method to solve the realistic SSA problem with imperfect state information, where the evader will exploit the sensing and tracking model to confuse their opponents by corrupting their tracking estimates, while the pursuer

wants to decrease the tracking uncertainties. In solving the game, fictitious play [6–8], as discussed in Section 2.2.2, was employed. In the papers by Shen et al. [41–43], the payoff takes the form of entropy, a function of the pursuers certainty in the evaders position. In the PE game, the evading spacecraft will try to increase the entropy by maneuvering while the evader will minimize the tracking uncertainties using a sensor-resource-efficient way. Since the pursuing spacecraft is assumed to not perform any maneuvers, this results in a sensor tasking schedule, as displayed in Figure 2.8. In the figure, SBSS can be seen simulated in a low-Earth polar orbit tracking a GEO spacecraft where the purple lines represent the pointing vector of the sensor when tasked. Though the sensor is not actively tasked to point in an optimal direction, the application of fictitious play to sensor management is useful beyond this application.

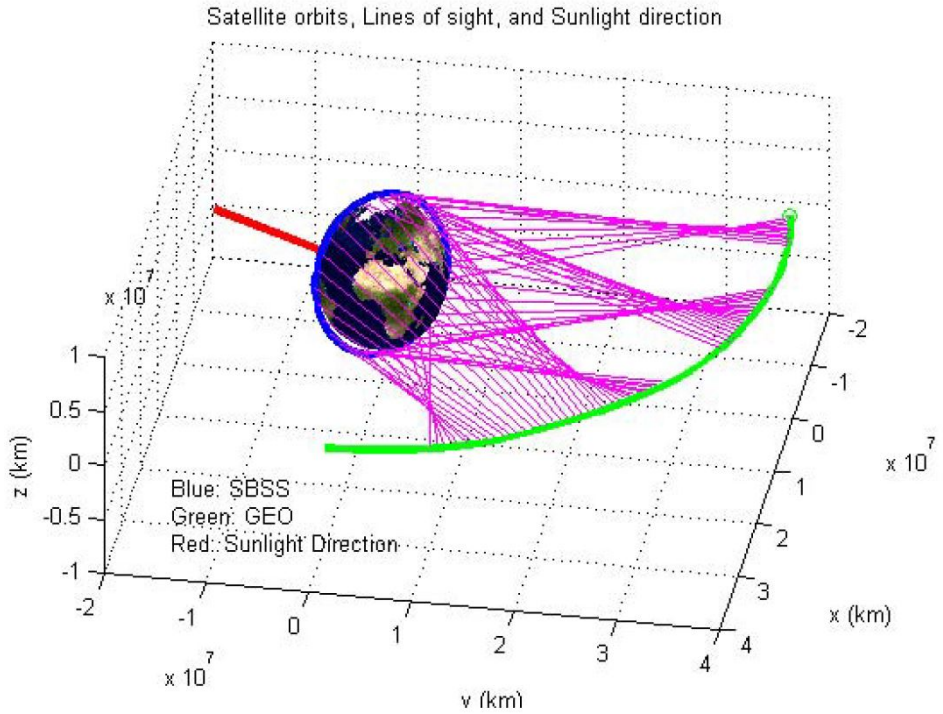


Figure 2.8. The Orbits of SBSS and GEO Under the Game Theoretic Sensor Management and Maneuver Strategies [43]

2.6.3 Machine Learning Applied to Space Surveillance

The application of machine learning to space surveillance has been even less explored than that of game theory, as described in the previous section. The work that has been done has focused primarily on supervised machine learning methods, such as neural

networks [47]. Neural networks have proven useful in adaptive estimation for tracking maneuvering targets, as in Reference [48], but require training that is not conducive for use in a catch-all tool.

Unsupervised machine learning would prove more useful in such a tool, but has only recently been considered for space surveillance applications. Starting in 2016, Shabarekh et al. published two papers [49, 50] on an unsupervised machine learning approach to probabilistically characterize Patterns of Life (PoL) for GEO satellites. Inspired by similarity-based clustering [51], an Interval Similarity Model (ISM) was used to predict future maneuvers learned from repeating intervals of unlabeled historical observations. Using this method, the authors were able to probabilistically predict maneuvers of the Galaxy 15 satellite with high confidence eight days in advance of the actual maneuver and detect deviations from its expected PoL within hours of the predicted maneuver. However, they also demonstrated that ISM is unable to predict future maneuvers for a spacecraft with an irregular PoL.

Linares and Furfaro [52] present a solution to sensor management formulated as a MDP, a mathematical framework for modeling decision making in situations where outcomes are partly random and partly under the control of a decision maker. The problem is then solved with RL using the actor-critic policy gradient approach and modeled using deep neural networks. However, as discussed above, neural networks aren't conducive to the research being proposed here. Fortunately, the authors followed this work with a second paper [53] and approached the same problem using inverse RL instead. As in reference [52], the problem is formulated as an MDP, but assume instead that a reward function is not explicitly provided. Rather, Feature Mapping [54] is employed to accurately estimate the reward function that the spacecraft is using while maneuvering by assuming that the observed trajectories are optimal with respect to the spacecraft's reward function. This approach allows for the determination of the reward function without the use of neural networks.

2.6.4 Proximal Policy Optimization in SSA

At the conception of this dissertation concept, some previous research had been done in applying RL to spacecraft dynamics problems, such as in References [52, 53, 61], but only References [85, 86], this researcher's own work, so far have focused on the detection avoidance problem and implemented PPO as the policy update algorithm. While other RL algorithms - like deep Q-learning [55], "vanilla" policy gradient methods [56], and trust region / natural policy gradient methods [17] - have their benefits, PPO has shown to be

the most promising with regards to data efficiency, scalability, and robustness. Q-learning (with function approximation) fails on many simple problems and is poorly understood while vanilla policy gradient methods have poor data efficiency and robustness. [2].

In the last few years, since the conception of this dissertation, a number of additional advancements have been made as well. In 2017, Reference [57] showed how Q-Learning can be used in conjunction with accessible regions to compute initial guesses for low-thrust transfers in the circular-restricted three-body problem. Then, in 2019, the authors continued with their work and used supervised learning to influence the resulting solution geometry [58]. Following that, they applied the same framework to contingency planning [59].

Besides applications in the circular-restricted three-body problem, PPO has been used for rendezvous in a cluttered environment [60], Martian landing guidance [61], multi-body station keeping [62], and lunar lander guidance [63]. In 2019, Miller and Linares demonstrated the use of PPO for low-thrust trajectory optimization in a multi-body regime [64] and then showed how PPO can be used in the decision-making process for interplanetary low-thrust transfers [65]. Miller and Linares then contributed to LaFarge et al's paper published in 2020 on applying PPO to Libration point orbits [66]. In addition, Gaudet et al moved on from their work with Martian landing guidance by applying PPO to asteroid proximity operations [67].

2.6.5 Spacecraft Maneuver Strategies

Spacecraft maneuver strategies have traditionally been predictable, repetitive, and designed to meet requirements of specific missions. For most missions this takes the form of drag makeup maneuvers (DMUs), risk mitigation maneuvers (RMMs), and inclination adjust maneuvers (IAMs) [69]. These maneuvers are designed anywhere from days to months in advance and are optimized to minimize the associated fuel costs while meeting mission specifications, such as ground track control requirements [70]. As of yet, no work has been made publicly available involving maneuver strategies for tracking evasion, which is a primary focus of this research.

Chapter 3
Capabilities and Limitations

The sensor avoidance problem, especially the application of policy optimization to such a real-world two-player game, is brand new territory in astrodynamics. Therefore, a preliminary assessment of the capabilities and limitations of both players, as well as the policy optimization method, must be performed to ensure that 1) the problem is worth pursuing, 2) the selected approach is the best way to go about solving the problem, and 3) as much realism is included in the sensor-satellite dynamic as possible.

3.1 State Estimation Capability

In state estimation using optical sensors, the visual magnitude of the tracked object, the inherent noise in the observations, and other environmental factors can effect the capability of the sensor to maintain custody of a spacecraft. Though it is mathematically true that only three angles-only optical observations is required to estimate a complete orbital state, this is not at all the case when including accepted values of noise in the observations. Once a sufficient number of observations are collected to get an estimate of the state, this is then used to initialize a batch least squares (BLS) estimator, which provides the covariance to initialize an extended Kalman filter (EKF). The EKF, though, is limited in its ability to process observations through a maneuver. This section aims to determine 1) the number of angle-only observations required to get an accurate estimate of the spacecraft's state, 2) how an EKF usually responds when a spacecraft is observed maneuvering, and 3) the maneuver magnitude required to confuse an EKF.

3.1.1 Gooding's Method for Initial Orbit Determination

As first discussed in Reference [71], Gooding's method is considered to be the most accurate angles-only Initial Orbit Determination (IOD) method for the majority of applications. Like all IOD techniques, Gooding's method is used to estimate the orbit of a spacecraft based on a set of measurements with no *a priori* information. However, the angles-only observations adds additional complexity to the problem due to the absence of range data. While other IOD methods can determine the orbit estimate from just one set of observation data with range, angles-only methods required three sets of observation data, each at different times.

Gooding's method requires three inputs:

1. Position vectors, $\vec{R}_j$, for the sites the measurements are taken from,
2. Unit vectors, $\hat{\upsilon}_j$, calculated from the angular measurement data, which are line-of-sight vectors from the site to the orbiting spacecraft,
3. Measurement times, t_j for each observation.

The algorithm computes the spacecraft position at t_2 using Lambert's problem (a method for solving the full state of an object given its position at two points in time) [3] while iterating on the assumed position vectors at t_1 and t_3 using the Newton-Raphson method [72]. The pseudo-code for this process can be found in Algorithm 6.

Depending on the level of noise in the measurement data, angles-only IOD algorithms can produce very poor estimates of the spacecraft's state if the results from a set of three observations converge at all. To ensure an accurate estimate of the spacecraft's state, practitioners employ a scheme which investigates all of the possible combinations of three measurements given the data set of N measurements. Running Gooding's algorithm on each combination results in a set of estimates for the spacecraft state from which the output can be chosen. If an accurate estimate of the range magnitude is known, this can be used to choose the output from the set of estimated states, otherwise the state corresponding to either the mean or maximum semimajor axis [3] is chosen. The result is an estimate of the position vector at t_2 that can be subsequently iterated upon using BLS and KF methods.

3.1.2 Batch Least Squares

The BLS method is used to improve upon the estimate produced from the IOD run(s) through the use of differential correction. Given the estimate of the state from the IOD

Algorithm 6 Gooding's Algorithm Pseudo-code [71]

1: Given values: $\hat{R}_j$, t_j, and $\hat{v}_j$, for j = 1,2,3
2: Assume a value for Υ_1 and Υ_3
3: **while** Not Maximum Iterations or Tolerance Reached **do**
4: Estimate the orbit by solving Lambert's problem using Υ_1, Υ_3, and t_{31}
5: Compute the error in the position of the spacecraft at t_2
6: Iterate Υ_1 and Υ_3 using the Newton-Raphson method
7: **end while**

and a batch of observation data, the BLS processor continues to iterate on the state estimate and produces the covariance that quantifies its certainty in the state.

First, the state estimate and state transition matrix (STM), $\mathbf{\Phi}$, are propagated to coincide in time with the second observation in the batch. At this time, the Earth Centered Inertial (ECI) state is converted to angular coordinates and compared to the observations angles at that time to determine the residual

$$\Delta y_i = \vec{y_i} - \vec{Y}(x_i^-, t_i) \tag{3.1}$$

where $\vec{y_i}$ is the vector of angular observations and $\vec{Y}(x_i^-, t_i)$ is the vector of angles calculated from the state estimate. In addition, the Jacobian, $\boldsymbol{H}_i$, is found from the state estimate for use in linearizing about the state at the time of the observation. Along with the variance of the measurement noise, $\boldsymbol{R}_i$, these values are used to calculate the state update variables, $\mathbf{\Lambda}$ and $\mathbf{\Xi}$ as such

$$\boldsymbol{T}_i = \boldsymbol{H}_i \mathbf{\Phi}(\boldsymbol{t_i}, \boldsymbol{t_0}) \tag{3.2}$$

$$\mathbf{\Lambda} = \mathbf{\Lambda} + (\boldsymbol{T}_i^T \boldsymbol{R}_i^{-1} \boldsymbol{T}_i) \tag{3.3}$$

$$\mathbf{\Xi} = \mathbf{\Xi} + \boldsymbol{T}_i^T \boldsymbol{R}_i^{-1} \Delta y_i \tag{3.4}$$

This process is repeated for each set of observation data and the variables $\mathbf{\Lambda}$ and $\mathbf{\Xi}$ are compounded at each step. Once the entire batch of observation data is processed, the new state estimate and its corresponding covariance are calculated from

$$\boldsymbol{P}_i^+ = \mathbf{\Lambda}^{-1} \tag{3.5}$$

and

$$\hat{x}_i^+ = \hat{x}_i^- + \boldsymbol{P}_i^+ \mathbf{\Xi} \tag{3.6}$$

where $\hat{x}_i^-$ is initial orbit estimate at the first observation time and $\hat{x}_i^+$ becomes the new $\hat{x}_i^-$. This process is then repeated, iterating on $\hat{x}_i^-$, until it converges within a suitable tolerance or the maximum number of iterations is hit. The result is an improved estimate of the spacecraft's state and its covariance [73]. This process is detailed in Algorithm 7.

Algorithm 7 Batch Least Squares Filter [73]

1: Propagate state and covariance (if known) to an epoch t_0 resulting in $\boldsymbol{P_0^-}$, $\boldsymbol{\hat{x}_0^-}$
2: Set $\boldsymbol{\Lambda = 1/P_0^-}$ and $\boldsymbol{\Xi} = 0$; **if** $\boldsymbol{P_0^-}$ unknown **then** $\boldsymbol{\Lambda} = 0$
3: Read in the next observation: t_i, $\vec{y}_i$, $\boldsymbol{R_i}$
4: Propagate reference trajectory $\boldsymbol{x_i^-}$ to t_i and calculate new $\boldsymbol{x_i^-}$ and $\boldsymbol{\Phi(t_i, t_0)}$
5: Initial conditions: $\boldsymbol{x_i^-}$, $\boldsymbol{\Phi(t_i, t_0)}$
6: Differential Equations: $\boldsymbol{\dot{x}_i} = f(\mathbf{x}, t)$, $\dot{\boldsymbol{\Phi}} = [\frac{f(\mathbf{x},t)}{\partial x}]\ \boldsymbol{\Phi(t_i, t_0)}$
7: Accumulate observations:
8: $\mathbf{H}_i = |\frac{\partial \vec{Y}(x_i^-, t_i)}{\partial x}|$
9: $\Delta y_i = \vec{y}_i - \vec{Y}(x_{ref,i}, t_i)$
10: $\mathbf{T}_i = \mathbf{H}_i \Phi(t_i, t_0)$
11: $\boldsymbol{\Lambda} = \boldsymbol{\Lambda} + \mathbf{T_i^T R_i^{-1} T_i}$
12: $\boldsymbol{\Xi} = \boldsymbol{\Xi} + \mathbf{T_i^T R_i^{-1}} \Delta y_i$
13: Return to step 3 and increment $i = i + 1$, continue until last observation (N)
14: Solve: $\mathbf{P}_i^+ = \boldsymbol{\Lambda}^{-1}$, $\delta\hat{x} = \mathbf{P}_i^+ \boldsymbol{\Xi}$
15: **if** converged **then** set epoch state and covariance estimate:
16: $\hat{x}_i^+ = \hat{x}_i^- + \delta\hat{x}$, $\boldsymbol{P}_i^+ = \boldsymbol{P}_i^+$
17: **else** update $\hat{x}_i^- = \hat{x}_i^- + \delta\hat{x}$ and $\boldsymbol{P}_i^- = \boldsymbol{P}_i^+$ and return to step 1

3.1.3 Extended Kalman Filter

While BLS is accurate enough for state estimation that it is used for standard catalog maintenance, the generated covariance is known to be an under-estimate and BLS cannot be used to process new measurements without restarting from the first observation. The EKF has become the industry standard for the continual processing of spacecraft observation data and for covariance generation. Its linearization process is similar to that of the BLS algorithm, but sequentially updates the state estimate instead of processing the observations as a batch [73]. At each observation step, the state and covariance are propagated, the residual and Jacobian are found, and then the Kalman gain is calculated from those values as

$$\boldsymbol{K_i} = \boldsymbol{P_i^-} \boldsymbol{H_i^T} (\boldsymbol{H_i} \boldsymbol{P_i^-} \boldsymbol{H_i^T} + \boldsymbol{R_i})^{-1} \tag{3.7}$$

The state and covariance can then be updated from

$$\hat{x}_i^+ = \hat{x}_i^- + \boldsymbol{K_i}\Delta y_i \tag{3.8}$$

$$\boldsymbol{P}_i^+ = (\boldsymbol{I} - \boldsymbol{K_i}\boldsymbol{H}_i)\boldsymbol{P}_i^- \tag{3.9}$$

The updated state and covariance are then propagated to the next time observation time and the process is repeated. As more observations are processed, the state estimate continues to evolve and the covariance steadily decreases as the confidence in the state estimate grows [73]. This process is detailed in Algorithm 8.

Algorithm 8 Extended Kalman Filter [73]

1: Define or update previous reference: $\boldsymbol{P_i^-}$, $\hat{x}_i^-$
2: Read in the next observation: t_i, $\vec{y}_i$, $\boldsymbol{R_i}$
3: Propagate state from t_{i-1} to t_i to get $\hat{x}_i^-$ and $\boldsymbol{\Phi}(t_i, t_0)$
4: Initial conditions: $\hat{x}_{i-1}^+$, $\boldsymbol{\Phi(t_{i-1}, t_{i-1})} = \mathbf{I}$
5: Differential Equations: $\dot{\hat{x}} = f(x,t)$, $\dot{\boldsymbol{\Phi}} = [\frac{\boldsymbol{f(x,t)}}{\boldsymbol{\partial x}}]\ \boldsymbol{\Phi(t, t_{i-1})}$
6: Update covariance:
7: $\mathbf{P}_i^+ = \boldsymbol{\Phi}(t_i, t_0)\mathbf{P}_i^-\boldsymbol{\Phi}(t_i, t_0)^T + \mathbf{Q}_i$
8: Accumulate observations:
9: $\mathbf{H}_i = |\frac{\partial \mathbf{G}(\mathbf{x}^-_i, t)}{\partial \mathbf{x}}|$
10: $\Delta y_i = \vec{y}_i - \vec{Y}(\mathbf{x_i^-}, t_i)$
11: Calculate observation covariance
12: $\mathbf{S}_i = \mathbf{H}_i\mathbf{P_i^-}\mathbf{H}_i^T + \mathbf{R}_i$
13: Calculate the Kalman gain and estimate the state and covariance
14: $\mathbf{K}_i = \mathbf{P_i^-}\mathbf{H}_i^T(\mathbf{S}_i)^{-1}$, $\hat{x}_i^+ = \hat{x}_i^- + \mathbf{K}_i\Delta y_i$, $\mathbf{P_i^+} = (\mathbf{I} - \mathbf{K}_i\mathbf{H}_i)\mathbf{P_i^-}$
15: Return to step 1 and process the next observation

3.1.4 Results

A proof-of-concept study was performed to demonstrate the validity of this research and characteristics of the state estimation process. The inputs for this study can be found summarized in Table 3.1.

Table 3.1. Inputs for Proof-of-Concept Study

Input	**Value**
Semimajor Axis, a	42164.137 km
Eccentricity, e	0
Inclination, i	50°
Right Ascension of Ascending Node, Ω	28°
Argument of Perigee, ω	0°
True Anomaly, ν	270°
Propulsion Type	Chemical
Sensor Latitude, ϕ	40° North
Sensor Longitude, λ	110° West
Sensor Altitude, h	2 km
Field-of-View	3.5°
Right Ascension Observation Noise	0.002 229°
Declination Observation Noise	0.002 229°

As stated in the preceding sections, the first step for the ground player is to initialize the spacecraft state and corresponding covariance for the orbit determination routine. To this end, Gooding's angles-only algorithm was used to perform the IOD. Using a 10-observation moving window for iterating through two passes (as the satellite crosses the equatorial plane) of 19 observations each, a file was created that lists the outputted orbital elements from each iteration of Gooding's method performed. Note that the required number of observations for a sufficiently accurate estimate of the state using Gooding's method was determined through trial and error. A portion of the file can be seen in Figure 3.2 in which the columns correspond to the measurement 1 index, measurement 2 index, measurement 3 index, semimajor axis, eccentricity, inclination (in radians), right ascension of the ascending node (in radians), argument of perigee (in radians), true anomaly (in radians), and the modified Julian date, in that order.

index1	index2	index3	semimajor axis	eccentricity	inclination	RAAN	arg. of perigee	true anomaly	MJD
1	2	3	8369.004666890	0.92960904	0.60122275	5.82189399	3.86667262	3.05316597	56160.20000000
1	2	4	11874.89752230	0.82678891	1.10035231	0.02768012	3.32789305	3.25770791	56160.20000000
1	2	5	13979.22933087	0.74610200	0.99907737	0.03548805	3.34539527	3.22767323	56160.20000000
1	2	6	15676.09464322	0.67803978	0.92181549	0.03276936	3.45532926	3.11628334	56160.20000000
1	2	7	15331.43973959	0.69072599	0.95450540	0.04005994	3.38523690	3.18253302	56160.20000000
1	2	8	16801.26673391	0.63395904	0.92773800	0.04735198	3.42679333	3.13491638	56160.20000000
1	2	9	17376.75458985	0.61197642	0.93882577	0.05657851	3.38264480	3.17304193	56160.20000000
1	2	10	19187.20386993	0.54393639	0.93454253	0.06976808	3.36386467	3.18247387	56160.20000000
1	2	11	-675.673422650	464.079233	0.87699950	0.33389886	3.27596359	6.22184548	56160.20000000
1	2	12	21880.12424617	0.44656640	0.92610124	0.08348449	3.36226233	3.17419923	56160.20000000
1	2	13	26992.28713750	0.28303100	0.92190001	0.10255570	3.30464624	3.21829423	56160.20000000
1	2	14	26560.71329178	0.29487549	0.91853938	0.10023884	3.36364950	3.16081932	56160.20000000
1	2	15	28567.90854790	0.23764447	0.91959117	0.10639994	3.30660735	3.21355776	56160.20000000
1	3	4	15438.89074836	0.91075852	0.28975349	1.01973856	5.52761604	3.42114843	56160.20000000
1	3	5	13789.46908261	0.86605549	0.62892190	0.69176848	5.92976297	3.30781191	56160.20000000
1	3	6	14280.34310638	0.81427102	0.79508528	0.62109855	6.06834172	3.22174351	56160.20000000
1	3	7	18702.43010454	0.69304931	0.78298625	0.57342617	5.96659852	3.35776108	56160.20000000
1	3	8	19147.61838398	0.64948988	0.82207236	0.56038077	6.04520523	3.28841970	56160.20000000
1	3	9	22451.88678055	0.55520526	0.81831709	0.54078420	5.94890585	3.39887523	56160.20000000
1	3	10	22312.52914328	0.55045892	0.82487159	0.54010200	5.97936341	3.36888203	56160.20000000
1	3	11	21448.45620513	0.56382672	0.83707523	0.54244140	6.05087470	3.29561905	56160.20000000
1	3	12	24377.66784538	0.47357275	0.83789618	0.52825839	5.98623843	3.37050402	56160.20000000
1	3	13	22205.01708689	0.53593088	0.84113367	0.53767410	6.05444370	3.29547291	56160.20000000
1	3	14	24494.09655220	0.45837013	0.84703496	0.52635040	6.04205773	3.31601735	56160.20000000
1	3	15	25384.59024030	0.43247412	0.84647981	0.52301926	6.01781257	3.34267670	56160.20000000
1	4	5	18661.21139381	0.66207374	0.94195272	0.55351328	0.08823211	2.96595393	56160.20000000
1	4	6	43000.47570345	0.02649944	0.87665647	0.48767614	2.10070579	1.00208871	56160.20000000
1	4	7	70115.07011300	0.42853747	0.87506199	0.13073714	5.84685622	0.65506740	56160.20000000
1	4	8	-29624.4769435	2.96377855	0.86710867	0.18285278	6.11795408	0.34685378	56160.20000000
1	4	9	122598.3798762	0.63683431	0.88324111	0.14388565	6.16992101	0.32286314	56160.20000000
1	4	10	-17043.2624456	5.20501396	0.88266192	0.20580082	0.06609777	0.09939582	56160.20000000
1	4	11	278813.7670243	0.82003244	0.88574556	0.45814966	2.91224639	0.21200779	56160.20000000
1	4	12	-13921.9575586	6.84061709	0.87854457	0.40227761	3.10481117	0.06012708	56160.20000000
1	4	13	59423.67833060	0.24682852	0.88150528	0.47686251	2.85729139	0.25334983	56160.20000000
1	4	14	522718.4016470	0.90172486	0.88121172	0.45577692	3.00506203	0.12092920	56160.20000000
1	4	15	343053.1049533	0.85100746	0.87835268	0.45692284	3.05631681	0.06884564	56160.20000000
1	5	6	80636.13009620	0.77887687	1.01624159	0.49681378	1.29382876	1.80113376	56160.20000000
1	5	7	20293.78124944	0.51068740	0.89313397	0.06214767	3.54525758	3.00516737	56160.20000000
1	5	8	32442.58990533	0.27686406	0.86820274	0.09564664	4.28094586	2.24554726	56160.20000000
1	5	9	28472.73143768	0.25013754	0.89664401	0.09840057	3.64924350	2.87592179	56160.20000000
1	5	10	55963.07373118	0.25852168	0.89623130	0.13266178	6.25039509	0.25061662	56160.20000000
1	5	11	-17713.0266322	5.01170589	0.88486334	0.20526288	0.07974560	0.08617188	56160.20000000
1	5	12	-24882497.3780	1.00196677	0.89308964	0.15886861	0.11210052	0.08707491	56160.20000000
1	5	13	-290513.833048	1.18601570	0.87679741	0.44910465	3.11302002	0.01783492	56160.20000000
1	5	14	-11031.5072167	9.00305782	0.87754803	0.39506960	3.14558580	0.02459410	56160.20000000

Table 3.2. IOD Output Data

Each row shows the orbital elements resulting from a single iteration of Gooding's algorithm, where the first three columns identify the observation indexes used for that iteration. Given the relatively short arcs between each set of three points, some of the resulting orbital elements are either physically impossible or correspond to hyperbolic orbits. Luckily, these are relatively uncommon and can be ignored when determining the correct state estimate. From this file, the output is selected by finding the semimajor axis value closest to 42,164 kilometers (assumed for any geosynchronous orbit), which, if only the iterations displayed were considered, would be the highlighted row. This result is then converted to Cartesian coordinates and propagated back to the time of the first observation, giving

$$\vec{r} = [-37124.9261, -19359.9490, 411.1446]\ km \tag{3.10}$$

$$\vec{v} = [0.9629, -1.7216, -2.3326]\ km/s \tag{3.11}$$

This state is then used to kickoff the BLS routine. The same set of 38 observations is used in the BLS routine to differentially correct the state estimate and produce a covariance for the first time. Over the course of just five iterations, the root mean square (RMS) of the measurement residual (see Figure 3.1) falls below 10^{-4} radians and the norm of the correction to the state estimate (see Figure 3.2) falls below 1 kilometer, the two requirements for convergence. The new state estimate is found to be

$$\vec{r} = [-37702.5028, -18831.1362, 1278.4266]\ km \tag{3.12}$$

$$\vec{v} = [0.81612, -1.8028, -2.3533]\ km/s \tag{3.13}$$

with a corresponding covariance matrix of

$$\left[\begin{array}{ccc:ccc} 587.618418 & 238.540693 & 57.689866 & -0.806995 & -0.333112 & -0.085022 \\ 238.540693 & 97.1337094 & 23.404550 & -0.326088 & -0.134922 & -0.034236 \\ 57.689866 & 23.404550 & 5.902464 & -0.077588 & -0.031910 & -0.008384 \\ \hdashline -0.806995 & -0.326088 & -0.077588 & 0.001155 & 0.000475 & 0.000120 \\ -0.333113 & -0.134922 & -0.031910 & 0.000475 & 0.000196 & 0.000049 \\ -0.085022 & -0.034236 & -0.008384 & 0.000120 & 0.000049 & 0.000013 \end{array}\right] \tag{3.14}$$

where the units of the top left quadrant are km^2, the units of the top right and bottom left quadrants are km^2/s, and the units of the bottom left quadrant are km^2/s^2.

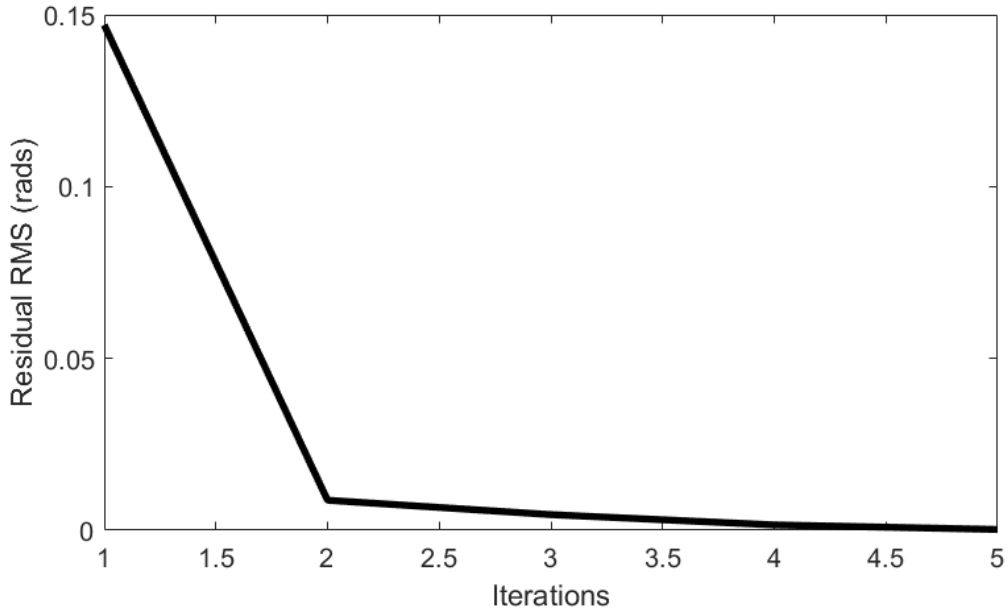


Figure 3.1. RMS of the Measurement Residual

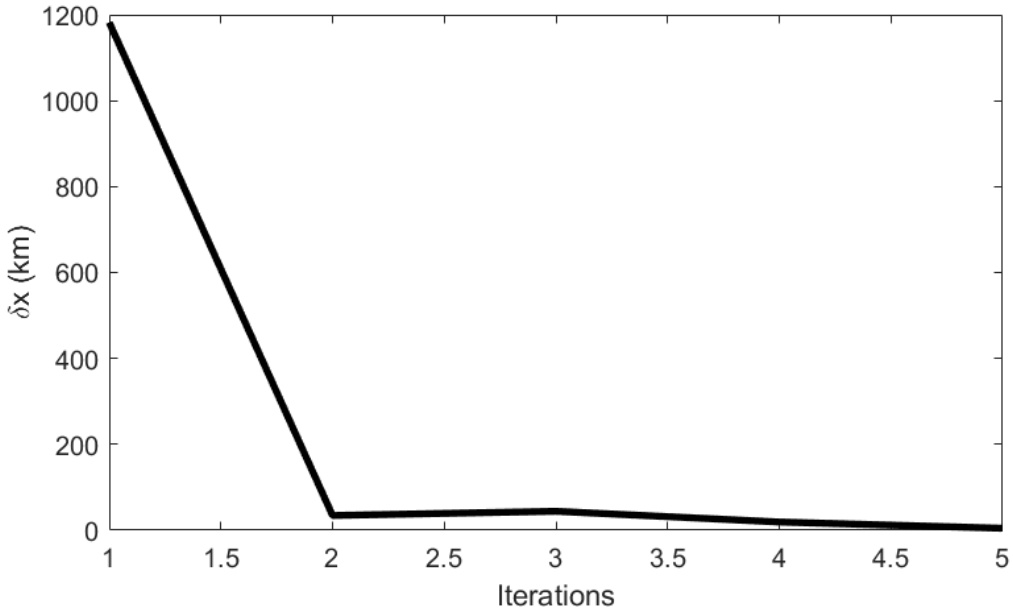


Figure 3.2. Norm of the Correction the State Estimate

From here, the state and covariance are used as inputs for the EKF and additional data can be processed beyond the batch of observations used for the IOD and BLS initialization. For this exercise, an additional two passes are used. The evolution of

the estimation errors bounded by the 3σ standard deviations can be seen in Figure 3.3 for the position and in Figure 3.4 for the velocity. After processing the four passes of observations, the state is now estimated to be

$$\vec{r} = [-36717.972560, -2000.073030, -1236.465823]\ km \tag{3.15}$$

$$\vec{v} = [1.027604, -1.690442, -2.353272]\ km/s \tag{3.16}$$

with a corresponding covariance matrix of

$$\left[\begin{array}{ccc:ccc} 12754.136377 & 5725.211339 & 1870.267444 & 1.023778 & -0.196015 & -0.738406 \\ 5725.211339 & 2590.695829 & 842.007653 & 0.458828 & -0.087082 & -0.333317 \\ 1870.267444 & 842.007653 & 291.904264 & 0.149749 & -0.030360 & -0.107549 \\ \hdashline 1.0237778 & 0.458828 & 0.149749 & 0.000101 & -0.000011 & -0.000056 \\ -0.196015 & -0.087082 & -0.030360 & -0.000011 & 0.000014 & 0.000007 \\ -0.738406 & -0.333317 & -0.107549 & -0.000056 & 0.000007 & 0.000050 \end{array}\right] \tag{3.17}$$

where the units of the top left quadrant are km^2, the units of the top right and bottom left quadrants are km^2/s, and the units of the bottom left quadrant are km^2/s^2.

Due to the limitations of processing angles-only observations without range data, the state estimate does not improve from the EKF compared to the BLS, and can actually be less accurate if the EKF is not tuned perfectly. However, the BLS is known to under-estimate the covariance so the covariance output from the EKF is considered to be much more accurate. The measurement and process noises can be used to tune how the covariance changes over time to account for the presence, or lack, of new observations, as seen in Figures 3.3 and 3.4.

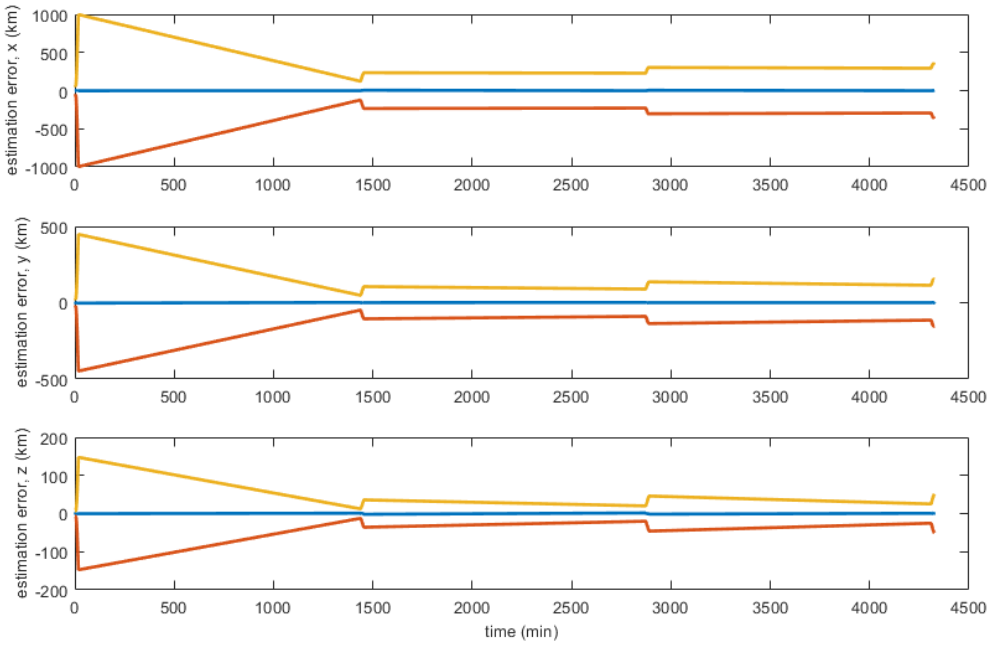


Figure 3.3. Position Estimation Error with 3σ Standard Deviation ($+3\sigma \equiv$ yellow, $-3\sigma \equiv$ red)

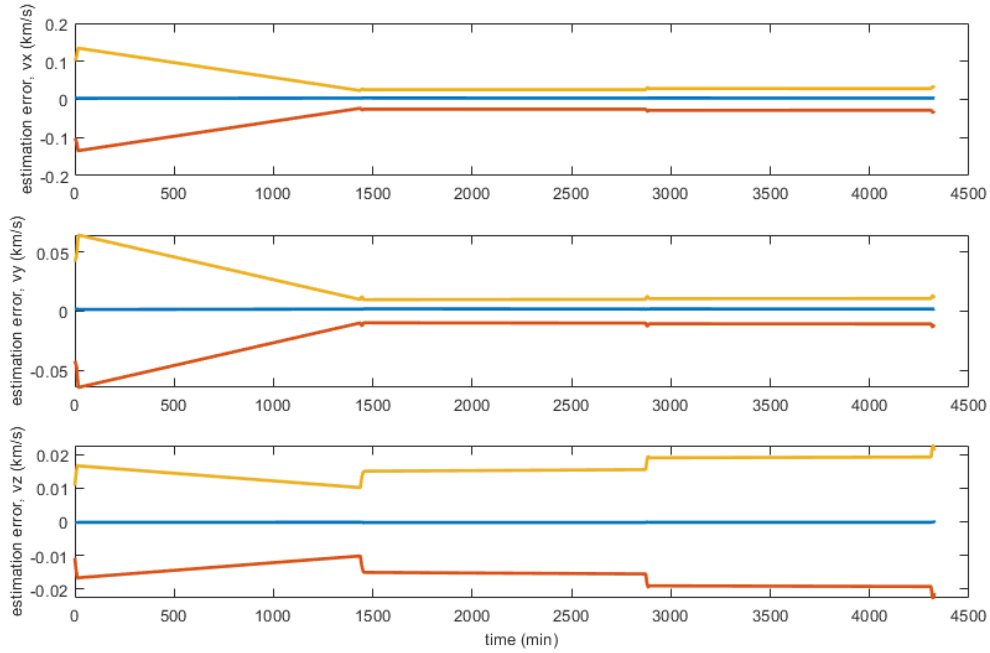


Figure 3.4. Velocity Estimation Error with 3σ Standard Deviation ($+3\sigma \equiv$ yellow, $-3\sigma \equiv$ red)

With the EKF ready to process additional data, the maneuvers of the evading spacecraft can be included. For this demonstration, both planar (change in semimajor axis) and plane change (change in right ascension of the ascending node, or RAAN) maneuvers are tested. These orbital elements were first introduced in Figure 2.3 back in Chapter 2. In order to demonstrate the validity of the sensor avoidance concept, there should exist a maneuver magnitude of each type that cannot be processed by the EKF alone and that has a reasonable ΔV cost such that it can be performed regularly through-out a mission. To test this, the maneuver magnitudes were gradually increased until the EKF failed to effectively process observations after the maneuver. In this scenario, the maneuvers are assumed to have occurred between observation passes, which is more likely than an impulsive maneuver occurring during the small 3.5° window.

To test the maneuver processing capability, semimajor axis changing maneuvers will be applied first. It was found that the EKF could successfully process through a change in semimajor axis of 200 kilometers. Figs. 3.5 and 3.6, the state and velocity estimates respectively, show that the estimated values stay well within the 3σ covariance values, meaning that the filter was able to successfully process observations and maintain an accurate estimate of the state after the maneuver was performed.

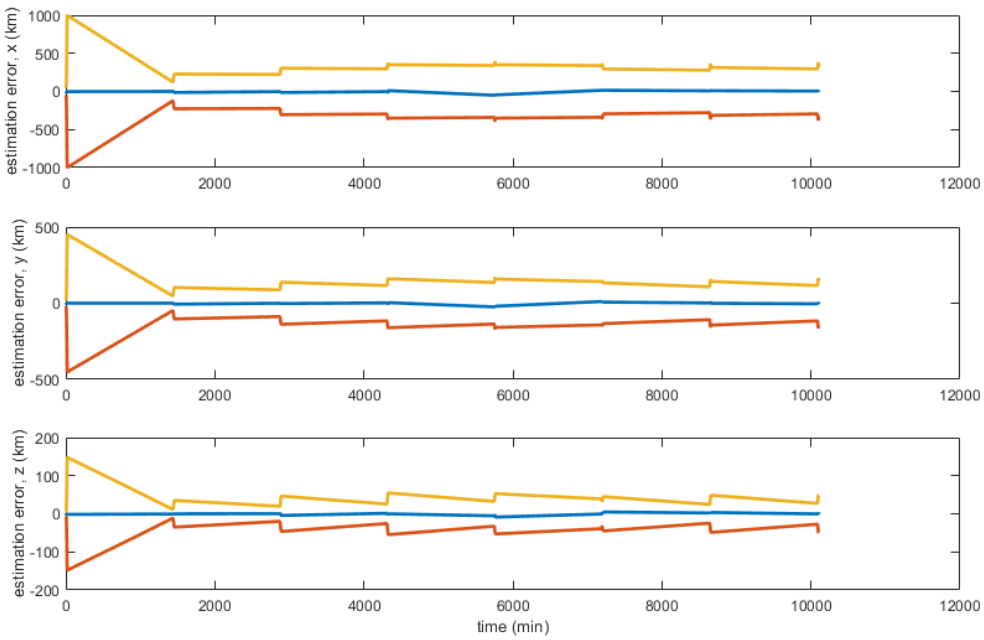


Figure 3.5. Position Estimation Error with a 200 km Semimajor Axis Change ($+3\sigma \equiv$ yellow, $-3\sigma \equiv$ red)

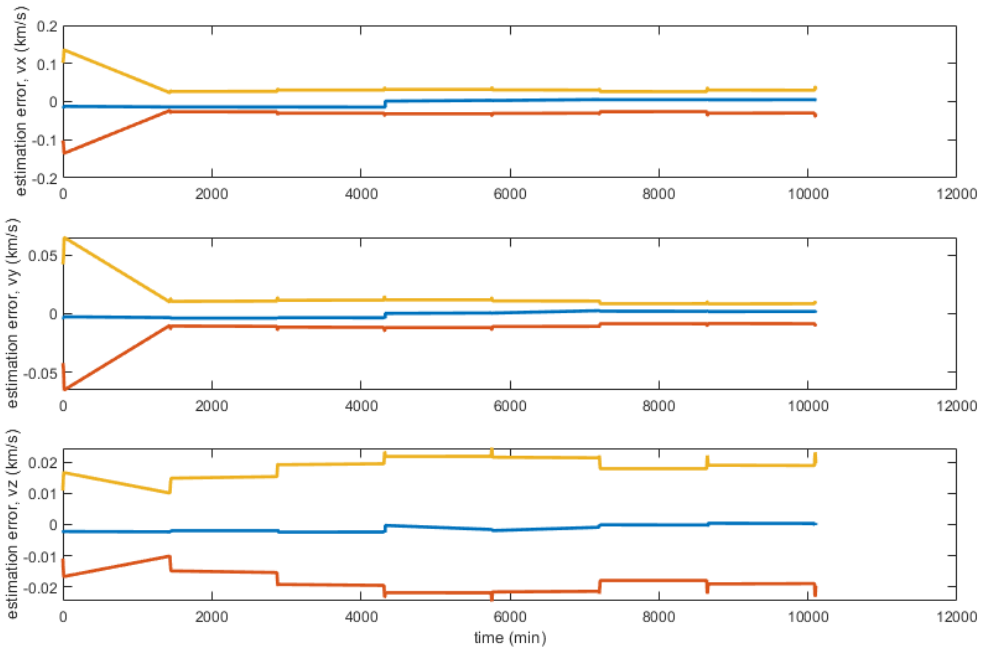


Figure 3.6. Velocity Estimation Error with a 200 km Semimajor Axis Change ($+3\sigma \equiv$ yellow, $-3\sigma \equiv$ red)

If the semimajor axis is instead changed by 300 kilometers, a grey area is found in which the estimation error begins to exceed the 3σ standard deviation even though the standard deviation again remains similar to as if no maneuver had been performed (see Figs. 3.7 and 3.8). With no change in the EKF, the pursuing player would have no idea that a maneuver, of 10.88 m/s magnitude, had occurred and that their state estimate was off by over 150 kilometers.

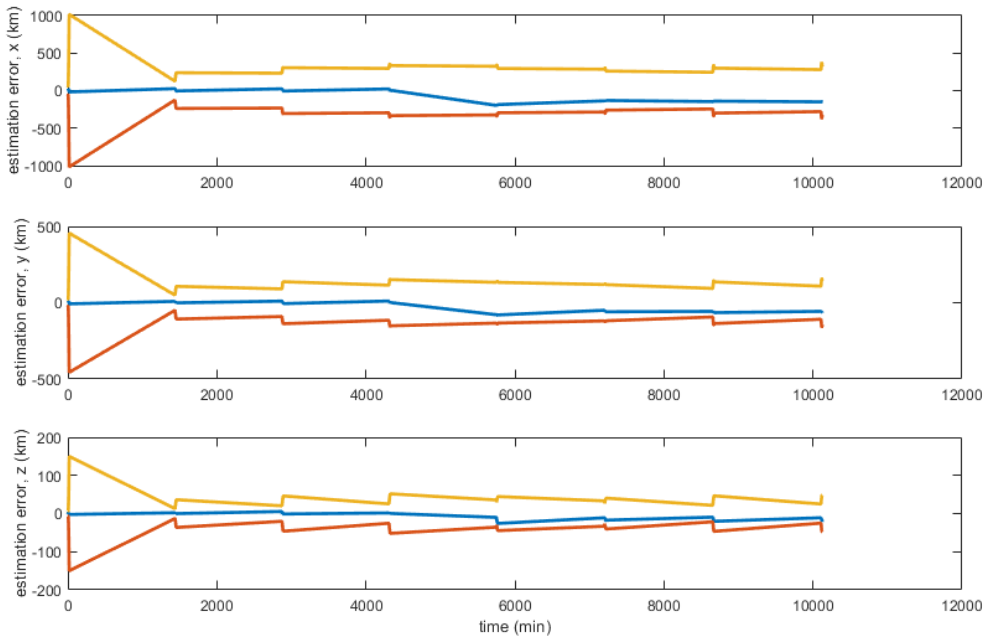


Figure 3.7. Position Estimation Error with a 300 km Semimajor Axis Change ($+3\sigma \equiv$ yellow, $-3\sigma \equiv$ red)

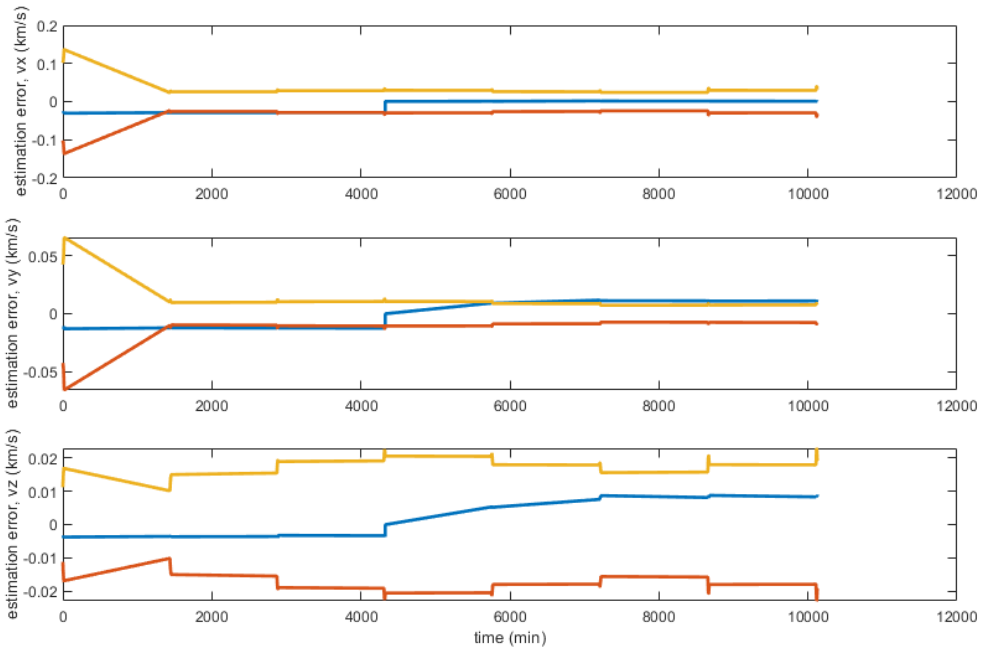


Figure 3.8. Velocity Estimation Error with a 300 km Semimajor Axis Change ($+3\sigma \equiv$ yellow, $-3\sigma \equiv$ red)

Next, an identical approach is taken for maneuvers that change the RAAN of the orbit. It was found that the EKF could successfully process through a change in RAAN of 0.2° (see Figs. 3.9 and 3.10). Like with the semimajor axis changing maneuver, the estimation error ended up being slightly greater at the conclusion of the data processing while the 3σ standard deviation mimicked the values found when no maneuver was performed.

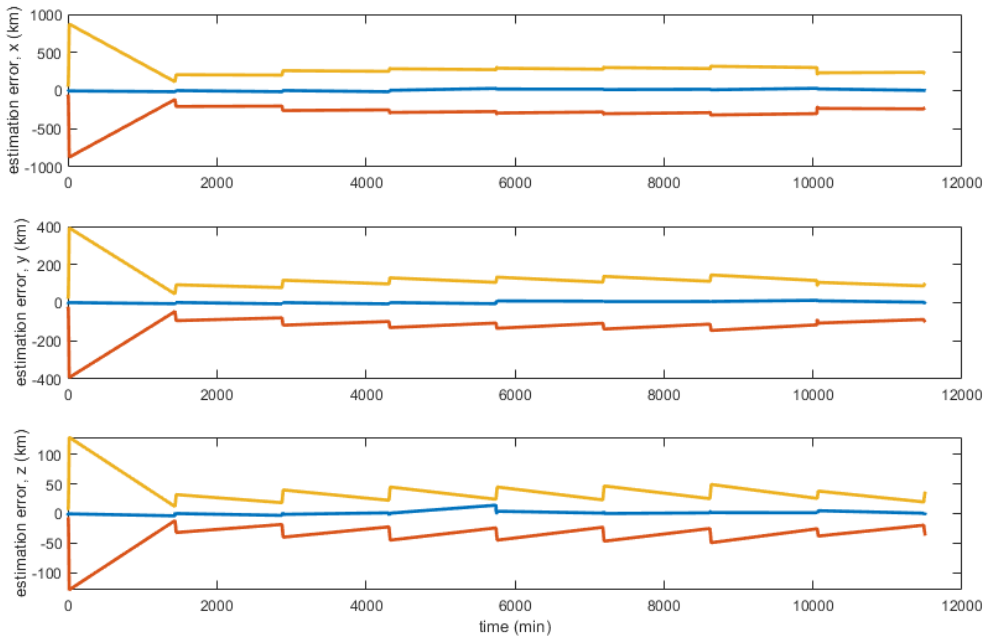


Figure 3.9. Position Estimation Error with a 0.2° RAAN Change ($+3\sigma \equiv$ yellow, $-3\sigma \equiv$ red)

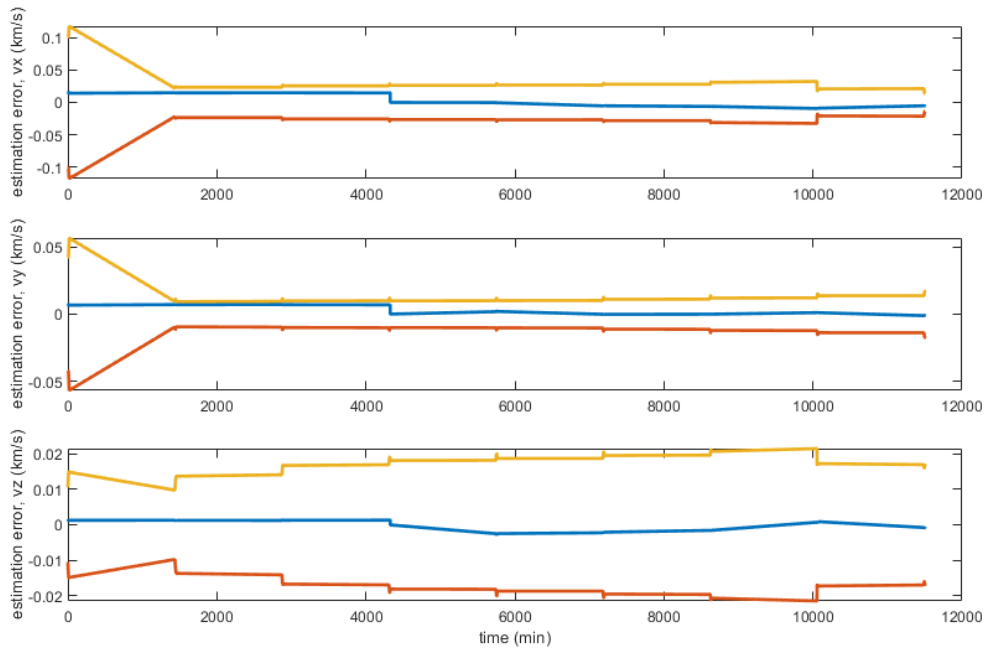


Figure 3.10. Velocity Estimation Error with a 0.2° RAAN Change ($+3\sigma \equiv$ yellow, $-3\sigma \equiv$ red)

If the RAAN change is increased to 0.3°, a similar result is found as when the

semimajor axis was increased - the estimation error begins to exceed the 3σ standard deviation and the standard deviation remains similar to as if no maneuver had been performed (see Figs. 3.11 and 3.12). Similar to the semimajor axis changing maneuver, the RAAN change cost 12.26 m/s in ΔV. This suggests that the maneuver magnitude, not type, is the primary cause for divergence in the processing of the observations through an EKF, allowing the evading player to prioritize the result of the maneuver over the maneuver selection itself.

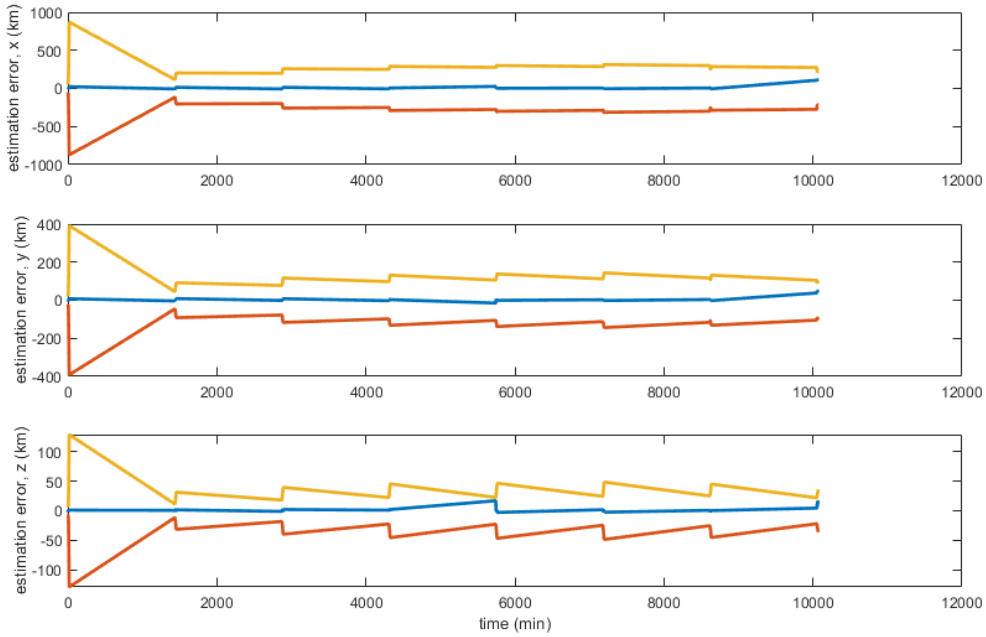


Figure 3.11. Position Estimation Error with a 0.3° RAAN Change ($+3\sigma \equiv$ yellow, $-3\sigma \equiv$ red)

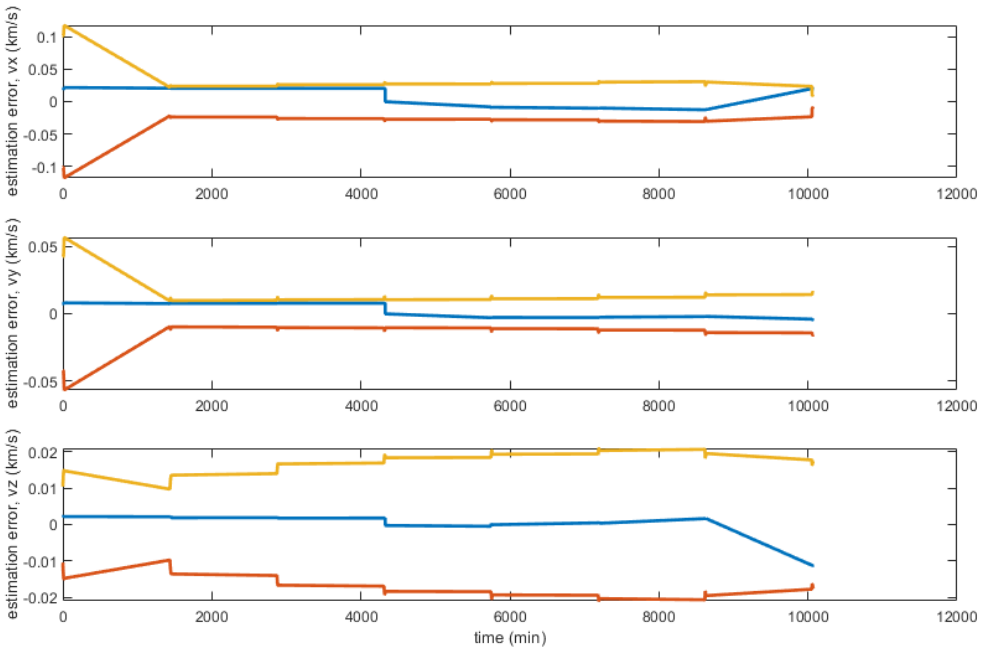


Figure 3.12. Velocity Estimation Error with a 0.3° RAAN Change ($+3\sigma \equiv$ yellow, $-3\sigma \equiv$ red)

Both the planar and plane change maneuvers were found to have been successful in deceiving the orbit determination system. This demonstrates that, with maneuvers about 10 m/s in magnitude, a spacecraft is able to successfully deceive an EKF. In addition, it takes about 30 minutes, at one observation a minute, for the state estimate error to exceed the filter's uncertainty in the state.

3.2 Maneuver Strategy Optimization Using Reachability Sets

The limitations and capabilities of the maneuvering spacecraft must be explored as well. Though the analysis in the previous section showed that a 10 m/s maneuver can successfully fool an EKF, the proposed research required optimizing an entire maneuver strategy. It is therefore prudent to demonstrate that optimizing the strategy as a whole is more effective than optimizing individual evasive maneuvers. This is accomplished by optimizing a series of maneuvers, with respect to the ΔV and the effect of the maneuver on the spacecraft's ground track, using reachability sets.

3.2.1 Reachability Sets for Ground Track Manipulation

A reachability set is the group of trajectories from a given initial condition that are possible under a specified range of control actions. Currently, exploration of a problem's state space is done heuristically, based on previously identified solutions, but this could result in designers completely missing improved mission design solutions that are not close to previous approaches. Reachability sets instead allow for regions to be identified in the system's state space where small maneuvers have large effects or, vice versa [74].

Reference [75] goes into greater detail on the reachable domain when applying single impulse maneuvers of arbitrary magnitudes at arbitrary locations. After the impulse is applied, the resulting trajectory can be described in the ECI frame by

$$\vec{r} = \begin{bmatrix} \cos\nu_0\cos\theta - \sin\nu_0\cos\Delta i\sin\theta \\ \sin\nu_0\cos\theta + \cos\nu_0\cos\Delta i\sin\theta \\ \sin\Delta i\sin\theta \end{bmatrix} \frac{j_1}{1+e_1\cos\theta+\nu_1} \tag{3.18}$$

where ν_0 is the departure true anomaly, θ is the angle between the vectors for a point on the trajectory and the point of application, Δi is the angle between the planes of the trajectory and the initial orbit, j_1 is the semilatus rectum of the trajectory, and e_1 is the eccentricity of the trajectory.

In reachability, the boundary of the reachable domain for planar orbital maneuvers, as modeled with the above equation, is defined by the intersection of a plane and an ellipsoid of revolution. The equation for the ellipsoid containing the trajectory can be described in terms of the angular momentum, h, and eccentricity vectors as

$$\vec{e}\cdot\vec{R} = \frac{h^2}{\mu} - ||\vec{r}|| \tag{3.19}$$

In ECI coordinates, this gives

$$R = f(k,l,\nu_0,\alpha,\beta) = \frac{j_1}{1+e_{1i}\cos k\cos l + e_{1,j}\cos k\sin l + e_{1,k}\sin k} \tag{3.20}$$

where k is the position elevation (angle between the position vector, $\vec{R}$, on the ellipsoid of revolution and its projection on the x-y plane), l is the position azimuth (the angle between the projection and the x axis), and e_{1i}, e_{1j}, e_{1k} are the components of the eccentricity vector (see Figure 3.13).

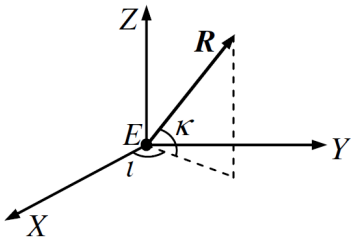


Figure 3.13. Definition of the Ellipsoid-of-Revolution Vector [75]

The equation of the orbital plane can be written as

$$\vec{h} \cdot \vec{r} = 0 \tag{3.21}$$

giving

$$\begin{aligned} g(\vec{r}, \nu_0, \alpha, \beta) = \sin\nu_0 \sin\beta \vec{r}_{1i} - \cos\nu_0 \sin\beta \vec{r}_{1j} + \\ \left(\sqrt{\frac{\mu}{p_0}} \frac{1 + e_0 \cos\nu_0}{V} + \sin\alpha \cos\beta\right) \vec{r}_{1k} = 0 \end{aligned} \tag{3.22}$$

where $\vec{r}_1 i, \vec{r}_1 j, \vec{r}_1 k$ are the components of $\vec{r}$ in the ECI frame, α and β are the body-centric angles that define the impulse vector (see Figure 3.14), and V is the magnitude of the impulse vector. The reachable domain for the spacecraft is contained in the torus formed by the intersection of these two envelopes.

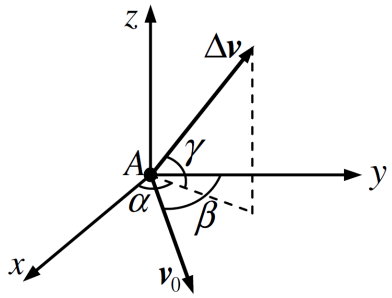


Figure 3.14. Initial Velocity and Impulse Vectors [75]

The ellipsoid-of-revolution envelope is defined by satisfaction of these equations

$$\begin{cases} (C_1 - A_1)s^2 + 2B_1 s + C_1 + A_1 = 0 \\ (C_2 - A_2)s^2 + 2B_2 s + C_2 + A_2 = 0 \\ (C_3 - A_3)s^2 + 2B_3 s + C_3 + A_3 = 0 \end{cases} \tag{3.23}$$

where $s = \tan l/2$ and

$$A_1 = \Big(\frac{\partial j_1}{\partial \nu_0} e_{1i} - j_1 \frac{\partial e_{1i}}{\partial \nu_0}\Big) \cos k \tag{3.24}$$

$$B_1 = \Big(\frac{\partial j_1}{\partial \nu_0} e_{1j} - j_1 \frac{\partial e_{1j}}{\partial \nu_0}\Big) \cos k \tag{3.25}$$

$$C_1 = \Big(\frac{\partial j_1}{\partial \nu_0} e_{1k} - j_1 \frac{\partial e_{1k}}{\partial \nu_0}\Big) \sin k + \frac{\partial j_1}{\partial \nu_0} \tag{3.26}$$

$$A_2 = \Big(\frac{\partial j_1}{\partial \alpha} e_{1i} - j_1 \frac{\partial e_{1i}}{\partial \alpha}\Big) \cos k \tag{3.27}$$

$$B_2 = \Big(\frac{\partial j_1}{\partial \alpha} e_{1j} - j_1 \frac{\partial e_{1j}}{\partial \alpha}\Big) \cos k \tag{3.28}$$

$$C_2 = \Big(\frac{\partial j_1}{\partial \alpha} e_{1k} - j_1 \frac{\partial e_{1k}}{\partial \alpha}\Big) \sin k + \frac{\partial j_1}{\partial \alpha} \tag{3.29}$$

$$A_3 = \Big(\frac{\partial j_1}{\partial \beta} e_{1i} - j_1 \frac{\partial e_{1i}}{\partial \beta}\Big) \cos k \tag{3.30}$$

$$B_3 = \Big(\frac{\partial j_1}{\partial \beta} e_{1j} - j_1 \frac{\partial e_{1j}}{\partial \beta}\Big) \cos k \tag{3.31}$$

$$C_3 = \Big(\frac{\partial j_1}{\partial \beta} e_{1k} - j_1 \frac{\partial e_{1k}}{\partial \beta}\Big) \sin k + \frac{\partial j_1}{\partial \beta} \tag{3.32}$$

The partials in the above equations are found by taking the derivative of these equations

$$j_1 = \frac{||\vec{r}||_0^2 V^2 \sin^2\beta + (h_0 + ||\vec{r}||_0 V \sin\alpha\cos\beta)^2}{\mu} \tag{3.33}$$

$$\begin{aligned} e_{1i} = \frac{h_0 + ||\vec{r}||_0 V \sin\alpha\cos\beta}{\mu}&[\nu_0 \cos(\gamma - \nu_0) + V\cos\beta\sin(\alpha + \nu_0)] \\ &+ \frac{||\vec{r}||_0 V^2 \sin^2\beta\cos\nu_0}{\mu} - \cos\nu_0 \end{aligned} \tag{3.34}$$

$$\begin{aligned} e_{1j} = -\frac{h_0 + ||\vec{r}||_0 V \cos\beta\sin\alpha}{\mu}&[\nu_0 \sin(\gamma - \nu_0) + V\cos\beta\cos(\alpha + \nu_0)] \\ &+ \frac{||\vec{r}||_0 V^2 \sin^2\beta\sin\nu_0}{\mu} - \sin\nu_0 \end{aligned} \tag{3.35}$$

$$e_{1k} = -\frac{r_0 V \sin\beta}{\mu}(\nu_0 \sin\gamma + V\cos\beta\cos\alpha) \tag{3.36}$$

with respect to ν_0, α, and β. The full partials can be found in reference [75].

The plane-family envelope is defined by satisfaction of these equations

$$\begin{cases} \sin\beta(e_0+\cos\nu_0)\vec{r}_{1i}+\sin\beta\sin\nu_0\vec{r}_{1j}+e_0\sin\nu_0\sin\alpha\cos\beta\vec{r}_{1k}=0 \\ \cos\alpha\cos\beta\vec{r}_{1k}=0 \\ \sin\nu_0\cos\beta\vec{r}_{1i}-\cos\nu_0\cos\beta\vec{r}_{1j}-\sin\alpha\sin\beta\vec{r}_{1k}=0 \end{cases} \tag{3.37}$$

The condition for the plane-family envelope is that the impulse is perpendicular to the initial orbit plane, so these equations simplify to

$$\begin{cases} (e_0+\cos\nu_0)\vec{r}_{1i}+\sin\nu_0\vec{r}_{1j}=0 \\ \sin\beta(\sin\nu_0\vec{r}_{1i}-\cos\nu_0\vec{r}_{1j})+\sqrt{\frac{\mu}{p_0}}\frac{1+e_0\cos\nu_0}{V}\vec{r}_{1k}=0 \end{cases} \tag{3.38}$$

These equations are solved by transforming the trigonometric forms into parametric forms to obtain the envelope of the family of the planes containing the trajectories generated after an arbitrary impulse is applied at any point on the initial orbit

$$\begin{cases} o_1=\sqrt{\frac{p_0}{\mu}}\frac{V[(1-n_1^2)\vec{r}_{1j}-2n_1\vec{r}_{1i}]}{1+n_1^2+(1-n_1^2)e_0} \\ o_2=-\sqrt{\frac{p_0}{\mu}}\frac{V[(1-n_1^2)\vec{r}_{1j}-2n_1\vec{r}_{1i}]}{1+n_1^2+(1-n_1^2)e_0} \\ o_3=\sqrt{\frac{p_0}{\mu}}\frac{V[(1-n_2^2)\vec{r}_{1j}-2n_2\vec{r}_{1i}]}{1+n_2^2+(1-n_2^2)e_0} \\ o_4=-\sqrt{\frac{p_0}{\mu}}\frac{V[(1-n_2^2)\vec{r}_{1j}-2n_2\vec{r}_{1i}]}{1+n_2^2+(1-n_2^2)e_0} \end{cases} \tag{3.39}$$

where

$$\begin{cases} n_1=\frac{\vec{r}_{1j}+\sqrt{\vec{r}_{1j}^2-(e_0^2-1)\vec{r}_{1i}^2}}{(1-e_0)\vec{r}_{1i}} \\ n_2=\frac{\vec{r}_{1j}-\sqrt{\vec{r}_{1j}^2-(e_0^2-1)\vec{r}_{1i}^2}}{(1-e_0)\vec{r}_{1i}} \end{cases} \tag{3.40}$$

Together, the two envelopes define the reachable domain. An example of a resulting torus can be found in Figure 3.15. Here, the four surfaces represent the boundaries of the reachable space. From any arbitrary point on the nominal orbit, an arbitrary single impulse can result in the spacecraft accessing any point that lies between these four boundaries.

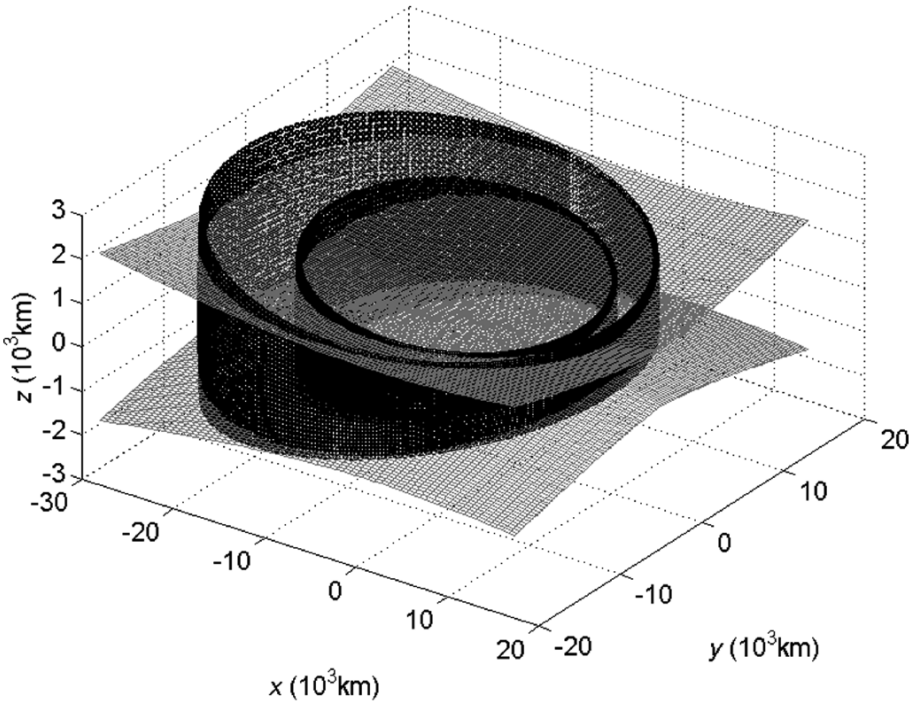


Figure 3.15. Intersection of the plane-family envelope and the ellipsoid-of-revolution envelope for arbitrary point of application and impulse direction [75]

In this demonstration, the interest lies in applying the above reachability theory to ground track manipulation based on the techniques presented in reference [76]. The authors demonstrated that significant ground track changes are possible without large expenditures of fuel. For example, Figure 3.16 demonstrates the capability of an electric propulsion (EP) system in a HEO orbit inclined at 85 degrees with a period of 2.7 hours. In this configuration, the system could change its time-over-target (TOT) by three hours after four days of thrusting and six hours after seven days. Given seven days of lead time, an EP system could change the TOT by six hours, ahead or behind, at two opposing locations in the orbit and would cover a total TOT change of 24 hours, which is equivalent to global reach.

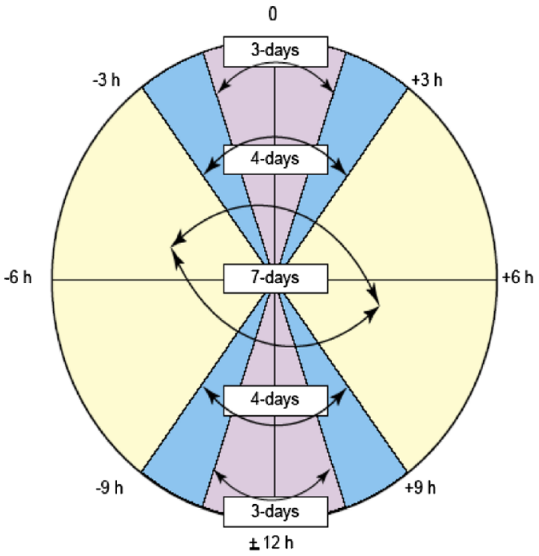


Figure 3.16. TOT Performance Based on Lead Time [76]

Terrestrial distance, the arc length between two points on Earth, was chosen as the metric to quantify the effects of a satellite maneuver:

$$D = 2R_{\oplus} \arcsin\left(\sqrt{\sin^2\left(\frac{\phi_2 - \phi_1}{2}\right) + \cos\phi_1 \cos\phi_2 \sin^2\left(\frac{|V_2 - V_1|}{2}\right)}\right) \tag{3.41}$$

where ϕ_1 is the latitude of point 1, ϕ_2 is the latitude of point 2, λ_1 is the longitude of point 1, and λ_2 is the longitude of point 2. These longitude values can be converted to an Earth-Centered-Earth-Fixed (ECEF), a coordinate frame with its origin at the center of the Earth and its axis aligned with the international reference pole and meridian that are fixed with respect to the Earth's surface, position vector using

$$\vec{r}_{1i} = ||\vec{r}|| \cos\lambda \cos\phi \tag{3.42}$$

$$\vec{r}_{1j} = ||\vec{r}|| \cos\lambda \sin\phi \tag{3.43}$$

$$\vec{r}_{1k} = ||\vec{r}|| \sin\lambda \tag{3.44}$$

From there, the ECI position vector can be found by applying a rotation about the z-axis through the location of the Prime Meridian. This gives a translation between a ground track target and ECI coordinates and a metric for quantifying the effects of a maneuver for use with the reachability domain theory above. Though reference [76] focuses on targeting specific ground locations in LEO and HEO orbits, the same theory can be

applied to achieve the opposite in GEO.

3.2.2 Computing Reachability Sets Using the Conjugate Unscented Transform

The core premise of the applied reachability set computation method is to include higher order terms of the Taylor series expansion for the final state, and use a non-product quadrature scheme known as CUT to compute the least squares coefficient corresponding to the expansion. The polynomial approximation method, as presented in Reference [77], solves the problem as visualized in Figure 3.17.

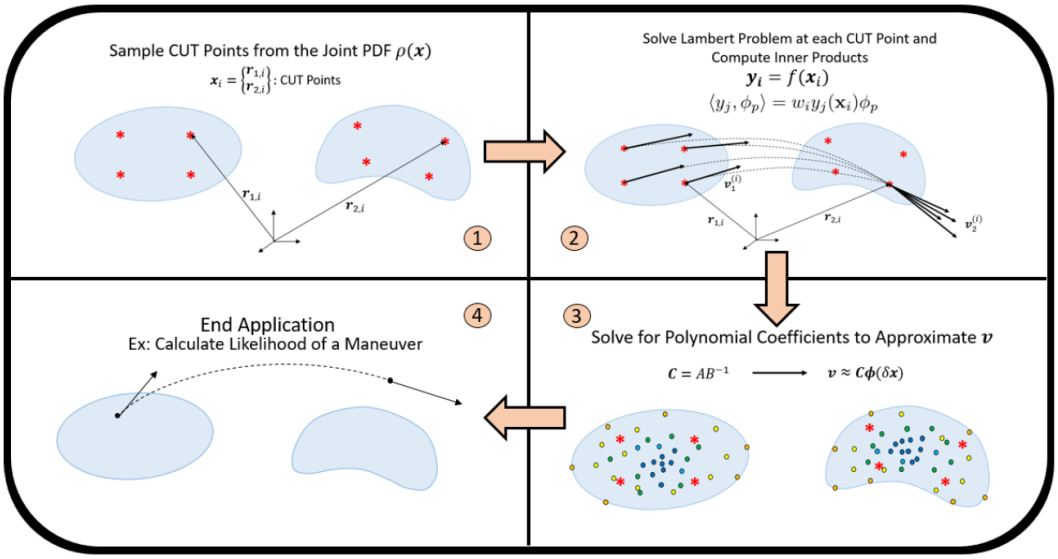
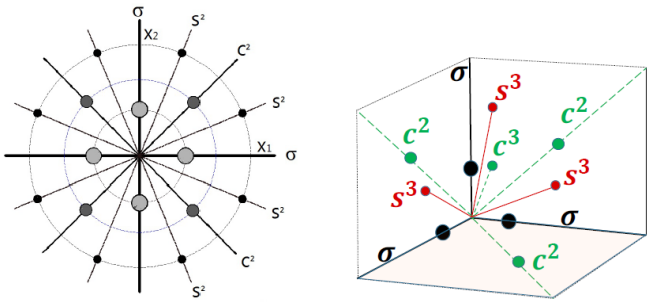


Figure 3.17. Polynomial Approximation Method Summary [77]

First, CUT points are sampled from the joint PDF [30,32,78,79]. The CUT approach can be considered an extension of the conventional UT method [80] that satisfies additional higher order moment constraints. Rather than using tensor products as in Gauss quadrature, the CUT approach judiciously selects special structures to extract symmetric quadrature points constrained to lie on specially defined axes as shown in Figure 3.18, allowing non-Gaussian distributions to be represented.

For each cubature point, two unknown variables, a weight w_i and a scaling parameter r_i are assigned. The moment constraints equations for the desired order are derived in terms of unknown variables r_i and w_i. Because of the symmetries of cubature points, the odd-order moment constraints equations are automatically satisfied, so the w_i and r_i are found by solving just the even-order equations. The order of these moment constraint



(a) CUT Points in 2D (b) CUT Points in 3D

Figure 3.18. Diagram of CUT Axes

equations dictates the set of cubature points. These new sets of so-called sigma points are guaranteed to exactly evaluate expectation integrals involving polynomial functions with significantly fewer points. With the application of the CUT approach, higher-order statistical equivalent transition matrices can be generated in a computationally efficient manner [77].

These points are then used to solve Lambert's problem. As described in detail in Reference 2, the nonlinear dynamics of the system can represented as

$$\bar{y} = \Theta(x^-) \tag{3.45}$$

where x^- and $\bar{y}$ denote the $(n \times 1)$ nominal input (position and velocity at point 1) and $(m \times 1)$ nominal output (position and velocity at point 2) vectors respectively, and f represents the function mapping the inputs to the outputs, as illustrated in Figure 3.19. In this diagram, the mean positions and velocities are denoted by $\boldsymbol{r}^*$ and $\boldsymbol{v}^*$ while $\boldsymbol{r}$ and $\boldsymbol{v}$ denote the states that fall within the distribution, in which the subscripts 1 and 2 denote the time steps.

It is desirable to find the solution for the same dynamics by substituting x into the mapping function where x is a continuous random variable with the pdf $P(x)$ such that,

$$y = \Theta(x) \tag{3.46}$$

The relationship between x and x^- is described by the $(n$ x $1)$ continuous random vector η with standardized pdf $P(\eta)$

$$x = x^- + T\eta = x^- + \delta x \tag{3.47}$$

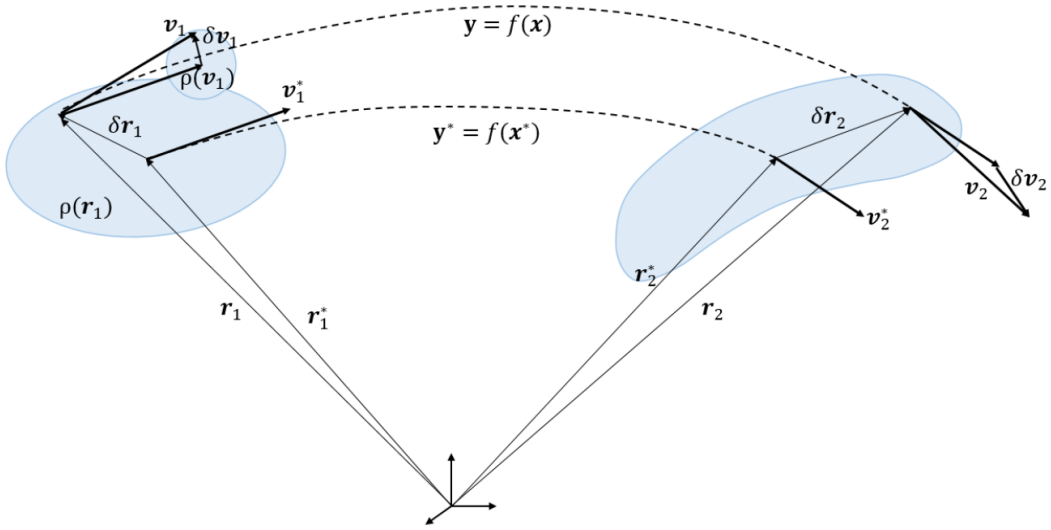


Figure 3.19. Reachability Problem Diagram

where x^- is defined as the (n x 1) mean vector of x, and T is a (nxn) scaling matrix dependent on the type of input uncertainty in a given problem. The term δx represents the scaled deviation in input due to uncertainty. In general, the function $y(x) = f(\bar{x} + \delta x)$ can be approximated as a q^{th} order Taylor series expansion, with the partial derivative terms grouped into constant matrices (C_i)

$$y \approx C_0 + C_1\delta x_1 + C_2\delta x_2 + C_3\delta x_3 + ...C_q\delta x_q \tag{3.48}$$

The coefficients can be rearranged into the full (nxM) coefficient matrix C and the deviation vectors ($\delta x_{(i)}$) into (M x 1) polynomial basis functions Π in which M is the total number of basis functions

$$y \approx C\Pi(n) \tag{3.49}$$

which can be rewritten in index notation as the approximation for the j^th element in terms of constant coefficients and arbitrary polynomial basis functions Π of maximum degree q as

$$y_j(x) \approx c_{j,k}\Pi_k \ for \ k = 1, 2, ...M \tag{3.50}$$

In order to solve for the coefficients, it is necessary to calculate the cost function, J, formulated as the error, ϵ, between the nonlinear output and the approximated output

squared, integrated over the domain of the PDF, P.

$$J = \frac{1}{2}\int \epsilon_j \epsilon_j P(\eta) d\eta \tag{3.51}$$

This cost function is minimized with respect to $c_{j,l}$, and the result can be written in compact form as

$$S = CR \tag{3.52}$$

where $S_{j,p} = \langle y_j(x), b_l \rangle_P$ and $R_{k,p} = \langle \Pi_k, b_l \rangle_P$. For a standard zero mean, unit variance Gaussian distribution the orthogonal polynomials will be the Hermite polynomials, and for a standard Uniform distribution in [-1,1] the orthogonal polynomials will be the Legendre Polynomials. The main difficulty in this method is computing the multi-dimensional integrals which appear in the expression for $S_{j,p}$. Generally these integrals are evaluated numerically by summing the arguments evaluated at p total points and multiplied by a weighting factor w_i, simplifying the inner product as

$$S_{j,p} =< y_j(x), b_l >_P = w_i y_j(x_i) b_p(\delta x_i) \ for \ i = 1, 2, ...N \tag{3.53}$$

These coefficients provide a quick numerical method for calculating a reachability set. Given an initial position and velocity, the CUT points are distributed around this coordinate using the desired maximum ΔV as the standard deviation in the velocity (the deviation in the position is set to zero). Lambert's problem is then solved at each CUT point and the inner products are found. The polynomial coefficients can now be calculated as described above and, for any number of desired Monte Carlo points, can be used to determine the reachable space after a pre-determined amount of time as passed.

Figure 3.20 shows a demonstration of this method for a maneuvering geosynchronous spacecraft with a Δt of one orbital period and a maximum ΔV of 10 m/s. The color-bar on the right shows the ΔV cost of each point in the set. Creating reachable sets for the same spacecraft with varying Δt values would create somewhat of a torus surrounding the nominal orbit.

3.2.3 Ground Track Manipulation

In applying this reachability set calculation method to detection avoidance, it is useful to explore how the reachable space behaves in topographic coordinates [3]. Converting the reachable set in Figure 3.20 into a ground track (ECEF coordinates bounded to the surface of the Earth) creates a series of "figure-8" patterns on the surface of the Earth, as

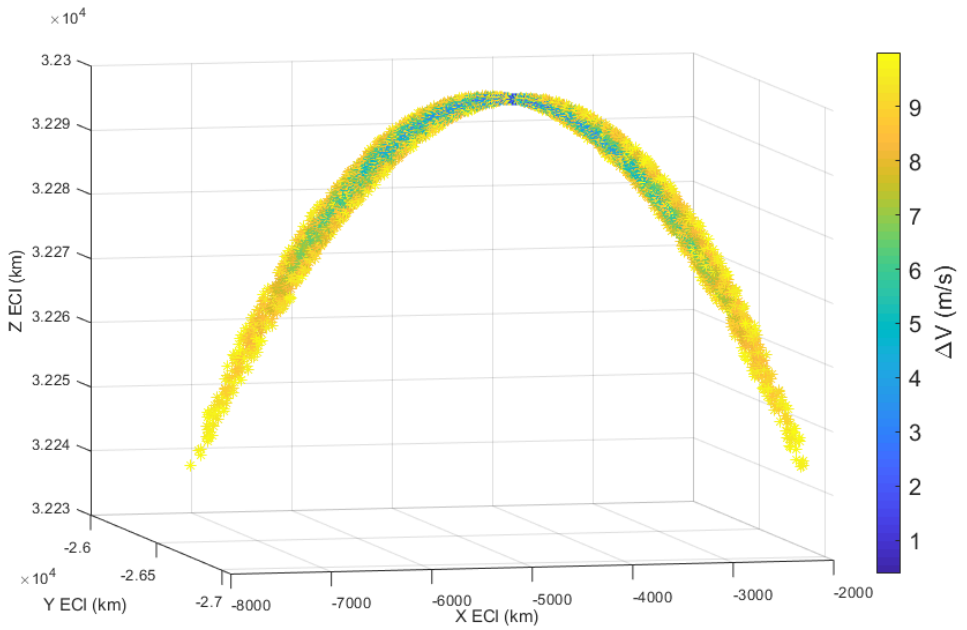


Figure 3.20. Reachable Set, in ECI, After 24 Hours with Max ΔV of 10 m/s

shown in Figure 3.21. These ECEF states can be readily converted into latitude/longitude and provide meaningful information about the location of the spacecraft with respect to the ground station of interest. Both will be useful for the multi-objective optimization of the spacecraft's detection avoidance maneuvers.

3.2.4 Multi-Objective Cost Function

With the computational burden of calculating reachability sets no longer an issue, this allows the problem to be extended further into multi-objective optimization. Spacecraft maneuver planning for detection avoidance is unique in that all objectives may not be met by moving some minimum distance from the nominal orbit. Analyzing the reachable sets in topographic coordinates instead presents a unique metric for quantifying detection avoidance.

For this study, three metrics are being considered:

1. Propellant usage, in the form of ΔV,
2. Ability to meet the nominal mission requirements, represented by the mean angle between the spacecraft and the ground station of interest,
3. Avoidance of the ground station's sensor, represented by the mean longitudinal change resulting from the maneuver.

The ideal end result is a maneuver that changes the spacecraft's orbit enough to

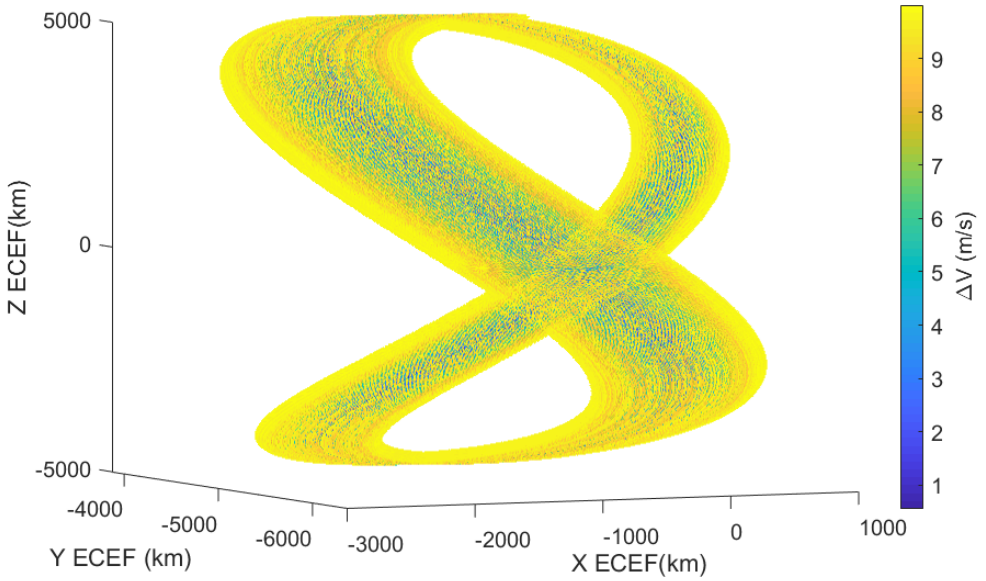


Figure 3.21. All Reachable Ground Tracks

confuse the ground station's sensor while simultaneously maintaining a proximity to the ground station and still sufficiently minimizing the ΔV cost. Through trial and error, a cost function that accomplishes this goal was found:

$$J = 2 \times 10^{-4}\left(\frac{\Delta V}{min(\Delta V)}\right)^4 + 60\left(\frac{\theta^2}{min(\theta^2)}\right) + 30\ln\left(\frac{\Delta\lambda^{-1}}{max(\Delta\lambda^{-1})}\right) \tag{3.54}$$

where θ is the mean angle in radians between the spacecraft and the ground station and $\Delta\lambda$ is the mean longitudinal change in radians resulting from the maneuver. The θ is squared in the function due to the loss that occurs as the angle increases between a spacecraft and its pointing location on the Earth's surface. In fact, if the angle is found to exceed the FOR of the spacecraft's sensor, the term is squared again. The entire ΔV term is set to the power of four to ensure that the cost is highly sensitive to the propellant usage. The inverse of the longitudinal change is used in the cost to ensure that the cost decreases as $\Delta\lambda$ increases. Lastly, the natural log is taken of the λ term to better distribute the values about the mean. Notice as well that each of the terms is normalized by either the maximum value or the minimum value according to their individual objectives. This is done to simultaneously guarantee that the minimum/maximum for each is equivalent to one, leaving the cost value unitless. The resulting cost function creates a wide distribution of values among the Monte Carlo points and results in a minimum cost that simultaneously meets each of the three objectives.

3.2.5 Maneuver Strategy Optimization

Though the method presented above allows for a much quicker calculation of a reachability set than previously thought possible, the computation time increases exponentially as more maneuvers are added in series. In order to consider all possible paths, each point in the reachable space must seed a reachability set of its own. For just three maneuvers, this would require 10^8 reachability set calculations with the current method formulation, something not feasible for a personal computer. To save on computation time without a significant loss in accuracy, a method was devised in which only a pre-defined percentage of the resulting solutions, close in cost to the optimal solution (for the single maneuver), is considered. A pre-defined number of points (say, 30, for example) is then sampled from within this space to seed the next calculation of reachability sets. Repeating this process for three maneuvers requires only 900 reachability set calculations to produce a $30 \times 30 \times 30$ space of possible optimal solutions. Though this does not guarantee global optimality for the three-maneuver series, the relative accuracy of the solution can be verified by repeating the calculation and collecting statistical data on the results, as presented in the next section.

3.2.6 Results and Analysis

The goal of this study is to prove that making use of a maneuver strategy could be more cost effective than optimizing a single maneuver in sensor detection avoidance. It is important to note that the purpose is to minimize the overall cost, calculated using Eq. (3.54), for the entire maneuver set. This method is different than minimizing the cost of each individual maneuver. Minimizing the cost per strategy may result in a reduced total cost, but there may be circumstances in which increasing the cost of a single maneuver may lower the overall strategy cost. This difference can be seen in Figures 3.22 and 3.23. Figure 3.22 is an example of three consecutively ("locally") optimized maneuvers where as Figure 3.23 is an optimized set ("globally") of three maneuvers. In both plots, the blue line is the initial ground track, the red line is the new ground track after the first maneuver, the yellow line is the new ground track after the second maneuver, and the purple line is the final ground track after the last maneuver. The global optimization of the maneuvers allows for larger changes and ultimately, more meaningful maneuvers.

Optimizing a set of maneuvers using the same reachability set method as done in prior research will rapidly lead to computational issues since a reachability set contains all of

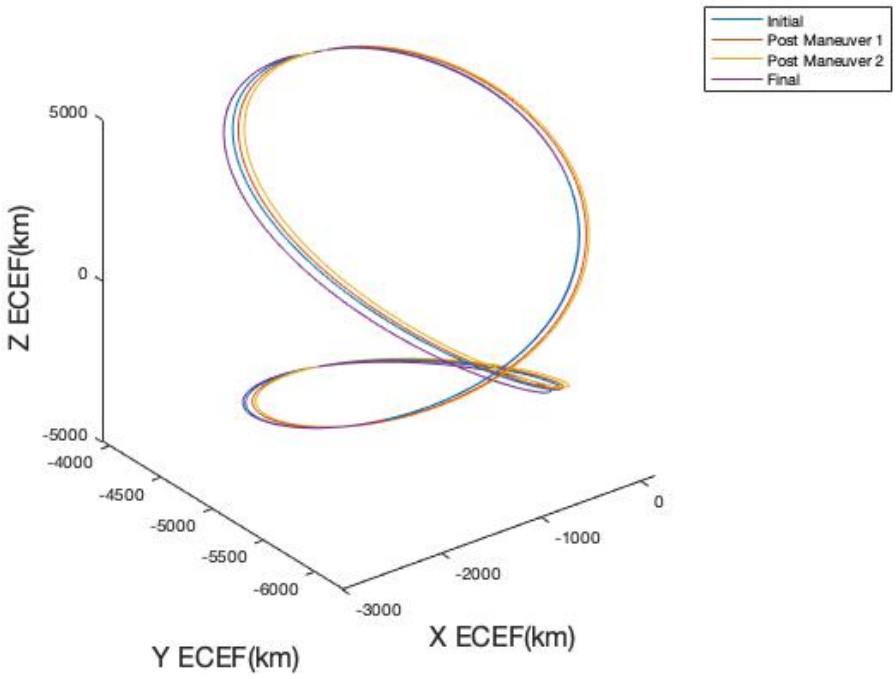


Figure 3.22. Local Optimization of Three Maneuvers

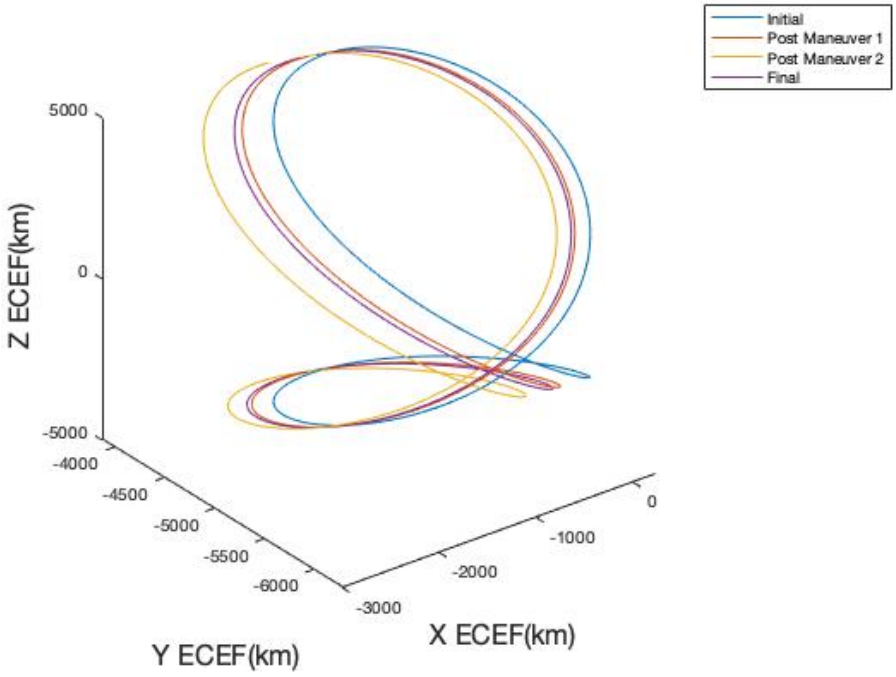


Figure 3.23. Global Optimization of Three Maneuvers

Table 3.3. Single Maneuver vs. Strategy Optimal Parameters

	Single Optima				**Strategy Optima**			
Maneuver #	1	2	3	**Avg**	1	2	3	**Avg**
ΔV **(m/s)**	3.089	3.089	3.091	**3.090**	3.079	3.084	3.080	**3.081**
Longitude (degs)	0.709	0.586	0.710	**0.668**	0.677	0.827	0.919	**0.808**
Sat-Sensor (degs)	4.703	4.844	4.721	**4.746**	4.746	4.725	4.734	**4.735**

a spacecraft's accessible locations given its initial conditions. To find the full reachable space after the second maneuver, new reachability sets must be found for each original ending location. Even using the CUT method, this process would grow computationally taxing at an increasing rate. In order to mitigate this issue, a small number of samples were collected from a percentage of the points close in cost to the optimal solution for the single maneuver. These randomly sampled points would then seed the next maneuver and the process is repeated for each consecutive maneuver, converting an exponential growth to a relatively linear one. This percentage of area will be referred to as the "percent considered" and will be variable. Selecting a percentage which is too low has the potential to cut off the actual optimal location with respect to the entire series and create false optimal points. Selecting too high of a percentage, however, could cause the too much variation in the samples taken, leading to inaccuracies. Similar to the percent considered, the number of samples is also variable.

After these parameters are implemented, it is necessary to create a test to compare relative costs of a single maneuver as opposed to a set of maneuvers. Through pre-testing it was determined that the minimum number of Monte Carlo points needed to ensure repeatable results is $10,000$. The test is then designed to allow for variability in the number of sample taken and the percentage of points considered. These variables range from 5 to 30 and 5 to 20 in steps of 5, respectively. Every permutation of the data points are considered and simulated to optimize the maneuver set by minimizing the overall cost. It is expected that with less samples taken and a higher percentage considered, the data will display a much higher rate of variability. Once the tests are run, it is possible to compare the single maneuver optimization (three individually optimized maneuvers) results to the new strategy optimization results. Overall the cost of running a strategy optimization, as opposed to the previously standard single maneuver, reduced the cost by 6.5%. These results are detailed in Figure 3.24 where the cost of each individual run is plotted with a linear fit to show the difference in the average values of the two approaches.

Table 3.3 shows three metrics: propellant (modelled as a change in velocity), longitude,

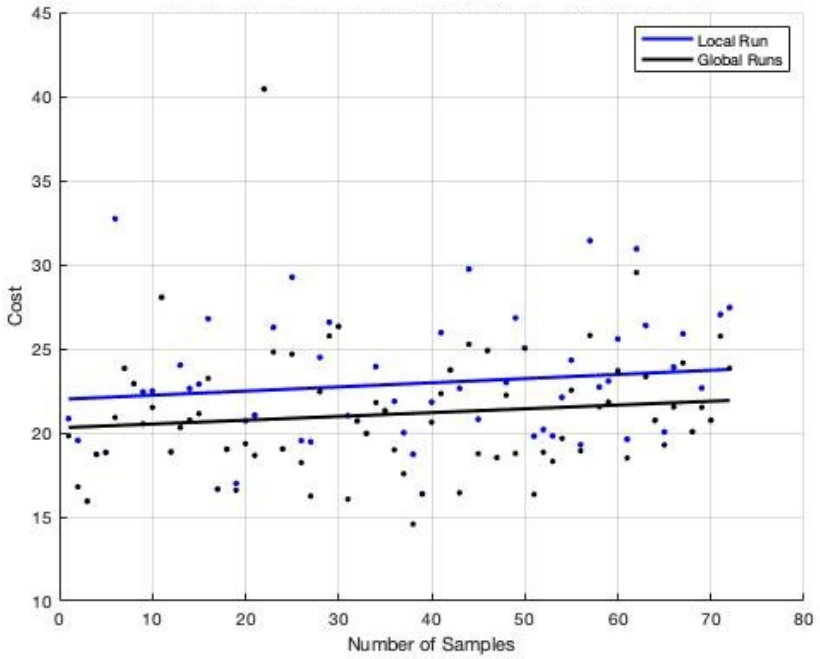


Figure 3.24. Optimized Cost of Single Maneuver Series vs Maneuver Strategy

and the angle of the satellite to the sensor. These metrics are the basis of the cost function and give meaning to further testing. The table displays the data of a simulation with 20 samples and 15 percent considered. The reasoning for this decision is discussed below. Although minimal, the global optima displayed a lower ΔV for each of the three maneuvers. The longitude of the maneuvers displayed a more apparent trend, where on average, the strategy optima's longitude was larger. However, the satellite angle seemed to have little trend. These patterns may be a result of random behavior because of the small sample size so further testing is needed to draw a significant conclusion.

By expressing the results in a whisker plot (Figure 3.25), a slight variation of the results with respect to cost from each of the six numbers of samples can be seen. Although each whisker plot only includes 12 individual data points, a trend appears to be forming. With 20 samples reaching a minimum cost, higher samples seem to keep a lower variability but with a higher cost. With a number of samples lower than 20, costs remain low but there is a large variance. This seems to agree with earlier expectations that a higher sampling will reduce the total variability in the data, removing some of the randomness or "luck". By adding more data points, this theory could be verified further and cut out some of the seemingly random deviations that are seen. The current results lead to the belief that a sampling of 20 would be sufficient to determine the minimized cost without

a large computational burden.

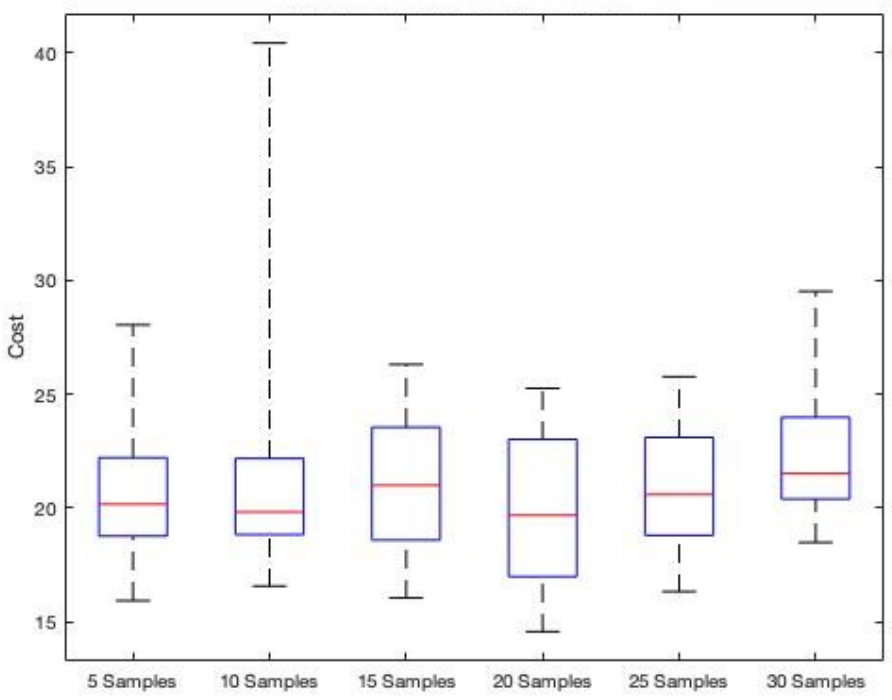


Figure 3.25. Variability of Cost (Eq. (3.54)) by Number of Samples

To narrow down a considered percentage, a similar plot is made to relate the percent considered to its resulting variation in cost. This new diagram (Figure 3.26) illustrates a similar but less obvious trend. That is, away from the center of the figure, somewhere between 10 and 15%, the variation increases. Although 5% considered does boast an overall lower mean cost, this cannot be proven to be a repeatable result because of its higher than usual standard deviation. Each of these plots contain 18 individual data points, which could be increased further to decrease noise.

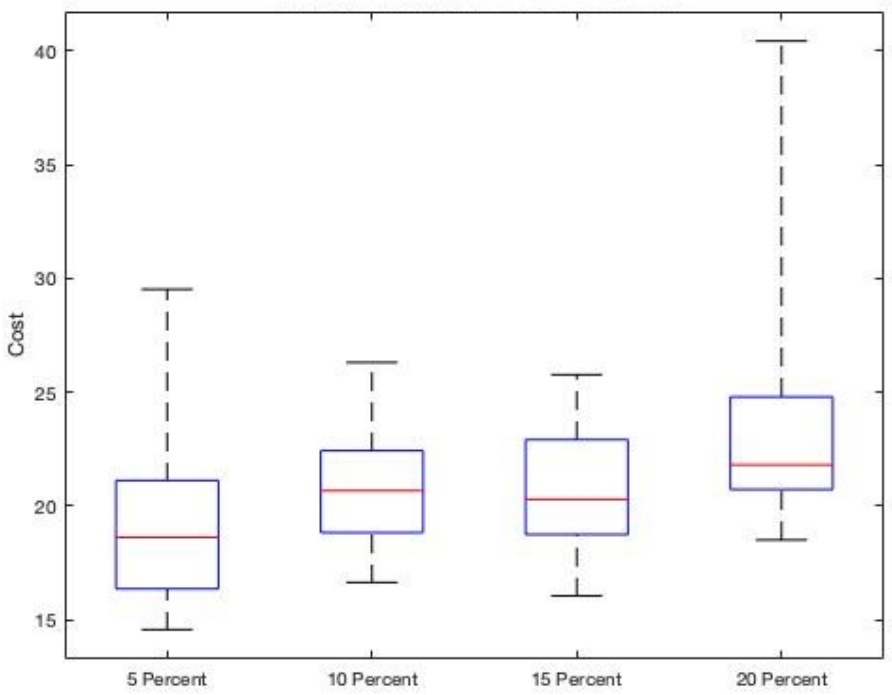


Figure 3.26. Variability of Cost (Eq. (3.54)) by Percent Considered

It can be concluded, at the very least, that the results presented in Table 3.3 (the 20 samples and 15 percent considered scenario), are signs of a larger trend: maneuvers performed in series must be optimized as a strategy to truly optimize their combined effectiveness in sensor avoidance optimization.

3.3 Proximal Policy Optimization vs Optimal Control

Now that the limitations and capabilities of the pursuing ground player and the evading space player have been determined, it is important to be able to justify the use of PPO in comparison to traditional optimal control methods.

3.3.1 Methodology

Though the PPO theory was presented in Section 2.5.2, traditional optimal control approaches have yet to be discussed. In this case, direct transcription was selected for the comparison.

3.3.1.1 Optimal Control

Direct transcription [81] methods allow for the discretization of a continuous space with the Keplerian dynamics serving as the constraints on the state. The maneuvering spacecraft's objective and gradient are a function of both the control vector at every time step and the distance between the spacecraft and the center of the sensor's FOV.

The objective and constraints are applied to the optimization problem using MATLAB's fmincon [82] with its Sequential Quadratic Programming (SQP) algorithm [83,84]. In doing so, the objective is formulated using the standard additive-cost optimal control objective

$$J_{\vec{u}(\cdot)}(\vec{x}_0) = \int_{t_0}^{t_f} g(\vec{x}(t), \vec{u}(t))dt \tag{3.55}$$

where $\vec{x}_0$ is the initial condition and $\vec{u}_t$ is the input defined over a finite interval $[t_0, t_f]$. This allows the trajectory optimization problem to be written as

$$\text{minimize}_{\vec{u}(\cdot)} \int_{t_0}^{t_f} g(\vec{x}(t), \vec{u}(t))dt \tag{3.56}$$

$$\text{subject to } \forall t, \dot{\vec{x}}(t) = f(\vec{x}(t), \vec{u}(t)), \tag{3.57}$$

$$\vec{x}(t_0) = \vec{x}_0 \tag{3.58}$$

in which $f(\cdot)$ is the state equation and $g(\cdot)$ is the running cost equation.

This is an optimization over continuous trajectories. In order to formulate this as a numerical optimization, it must then be parameterized with a finite set of numbers. Though there are many different ways to write down this parameterization, direct transcription was chosen for this application.

In direct transcription, the finite-time trajectories, $\vec{x}(t)$ and $\vec{u}(t) \forall t \in [t_0, t_f]$ are represented by their values at a series of break points, t_0,t_1,t_2,t_3, and denoting the values at those points $\vec{x}_0, ..., \vec{x}_N$ and $\vec{u}_0, ..., \vec{u}_N$, respectively. A simple mapping of the optimization problem onto a nonlinear program is to fix the break points at even intervals, dt, and use Euler integration to get

$$\text{minimize}_{\vec{x}_0,...,\vec{x}_N,\vec{u}_0,...,\vec{u}_{N-1}} \sum_{n=0}^{N-1} g(\vec{x}_n, \vec{u}_n)dt \tag{3.59}$$

$$\vec{x}_{n+1} = \vec{x}_n + f(\vec{x}_n, \vec{u}_n)dt, \ \forall n \in [0, N-1] \tag{3.60}$$

By satisfying the dynamic constraints, the optimization is effectively solving the dif-

ferential equation. However, instead of moving forward through time, it is minimizing the inconsistency at each of the break points simultaneously. This provides the optimal solution to the objective that meets all prescribed constraints. [81]

3.3.1.2 The Problem

In the devised dimensionless toy problem, the optimizer is setup with one of the three following terminal constraint scenarios:

1. Return to the initial orbit after some fixed time,
2. Arrive at a specified state (position and velocity) after some fixed time. Figure 3.27 shows an example of this where the spacecraft starts at the point $[1, 0]$ on the blue orbit and transfers to the red orbit by way of the black orbit, ending around the coordinate $[-1, -0.3]$.
3. No terminal constraint provided.

Note that non-dimensional units, scaled such that the initial orbit has a radius of one, are used to improve the accuracy and speed of the optimization and that the maneuver is assumed to be planar.

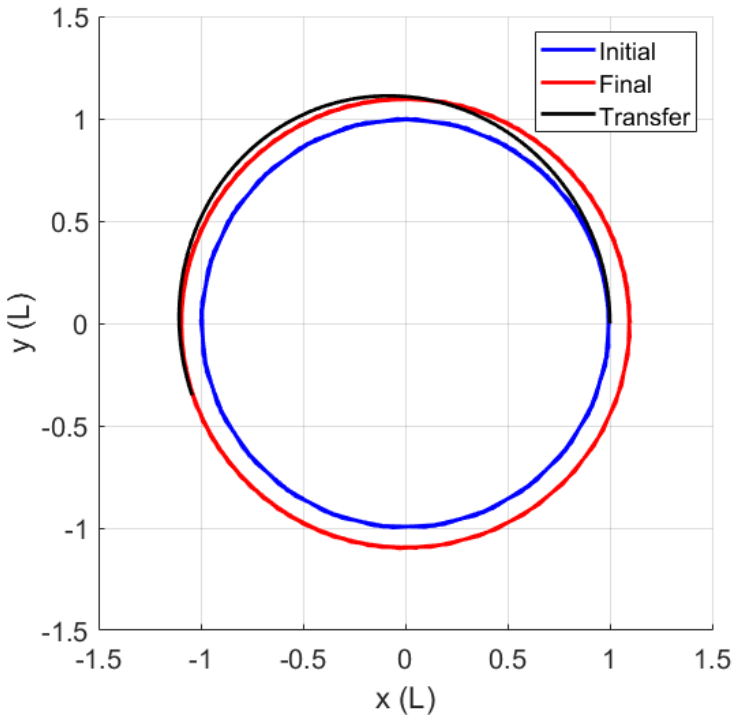


Figure 3.27. Arrival at a Specified Point After Some Fixed Time

Though minimizing the control input (propellant usage) is important, the primary

objective is to maximize the distance between the spacecraft and the sensor. The sensor itself is assumed to be optical with a defined FOV. By default the sensor will follow along the trajectory that the spacecraft, propagated from its previous point with no additional input, is assumed to be following. If at any point the spacecraft is not found within the sensor's FOV, then the sensor will move in a sinusoidal pattern of gradually increasing amplitude along the nominal path until the spacecraft is found again. Figure 3.28 shows this pattern, as zoomed-in on the orbital transfer (green line) just after departing from the initial orbit (red line).

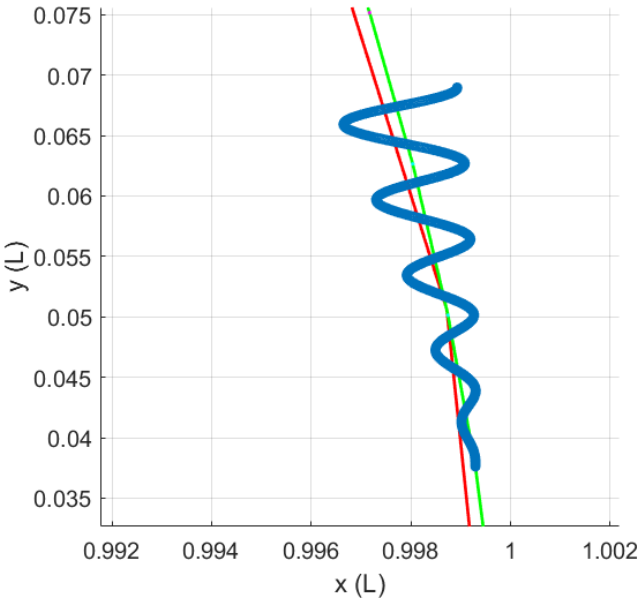


Figure 3.28. Example of the Sinusoidal Tasking Strategy

The end result will be a trajectory that diverges slightly from the minimum-fuel trajectory (like in Figure 3.27) as the sensor continues to lose and regain custody of the spacecraft over the specified time interval.

3.3.2 Results and Analysis

The outputs for both the optimal control solution and the RL solution is an optimal policy that the spacecraft can follow to ensure it avoids detection by the optical sensor while simultaneously minimizing propellant use.

3.3.2.1 Optimal Control

In implementing the toy problem in MATLAB as described above, the function tolerance was set to $1x10^{-4}$, the step tolerance was set to $1x10^{-6}$, and the constraint tolerance was set to $1x10^{-3}$. These ensured timely convergence while still satisfying the dynamical and terminal constraints.

Scenario 1, a terminal constraint requiring that the spacecraft return back to the initial orbit after some fixed time (known as station keeping), produced the trajectory found in Figure 3.29. The initial and final orbits are identical, but the transfer orbit can be seen in green. The dots overlaying the transfer orbit show the time periods where the spacecraft is found (in cyan) and not found (in magenta). It was determined that the spacecraft is found by the sensor about half of the time.

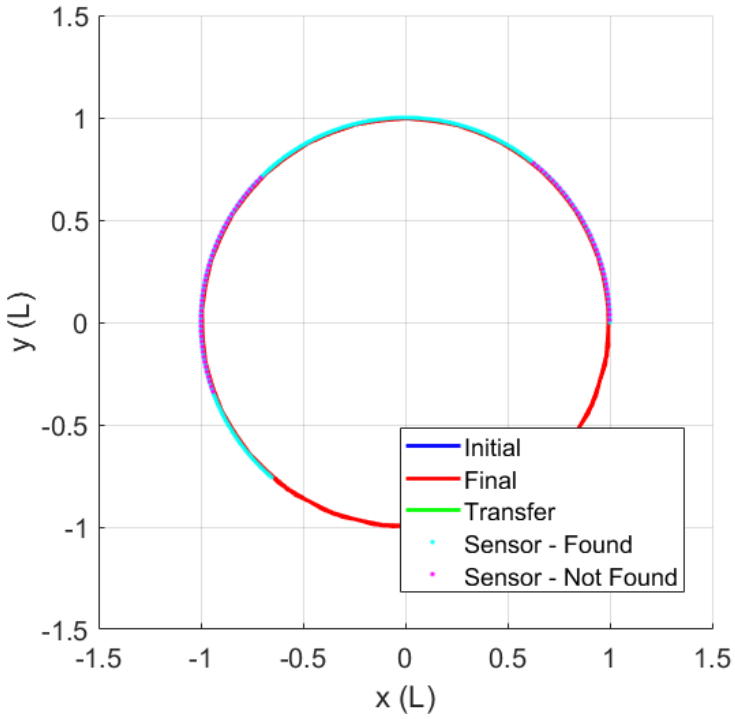


Figure 3.29. Optimal Trajectory for Scenario 1 - Optimal Control

Scenario 2, a terminal constraint requiring that the spacecraft arrive at a specified state (position and velocity) after some fixed time, produced the trajectory found in Figure 3.30. Again, the success of the avoidance maneuver/sensor tasking is seen in cyan and magenta colors. In this scenario, the spacecraft is found with more frequency but is able to evade the sensor more as it approaches the terminal constraint. The tasking

strategy is robust enough that the sensor maintains custody through-out most of the maneuver, likely due to the lack of curvature in the transfer. The sensor losing custody towards the end of the transfer, however, suggests that with more time, or slightly greater propellant use, the spacecraft could successfully maneuver such that the sensor could not find it without a more adaptable tasking strategy.

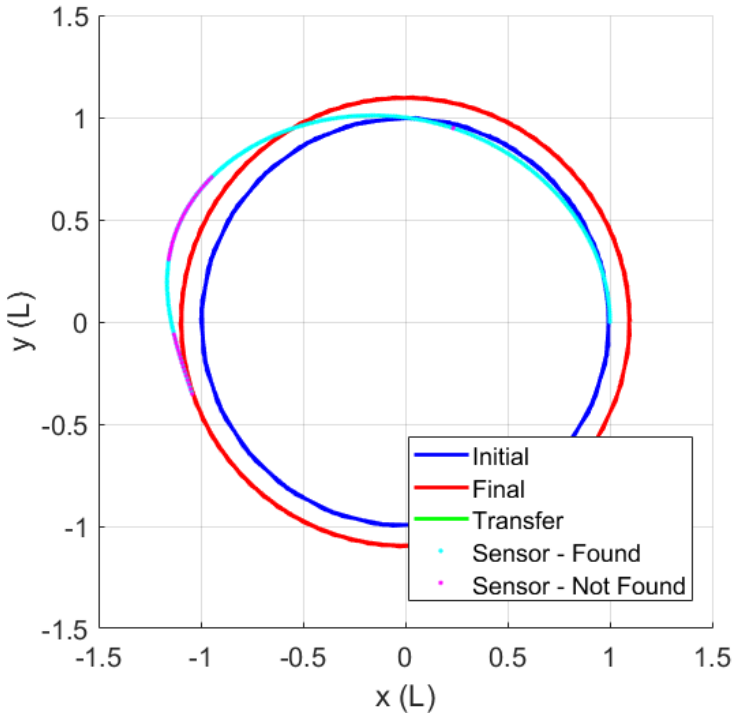


Figure 3.30. Optimal Trajectory for Scenario 2 - Optimal Control

Scenario 3, the scenario without a terminal constraint, produced the trajectory found in Figure 3.31. Without a specified final state, the spacecraft is free to maneuver in whichever direction most effectively avoids the tasked sensor. However, it appears that the optimal control approach chosen limits deviation of the spacecraft from the nominal orbit. Unsurprisingly, however, the spacecraft manages to continually lose the sensor throughout the maneuver given the lack of constraints on its final position (as is evident by the decrease in cyan color on the plot).

Though more scenarios could be tested with this toy problem, the capability of a maneuvering spacecraft to avoid a tasked sensor has been demonstrated. These results now serve as a basis for comparison with the PPO results when optimizing the same scenarios.

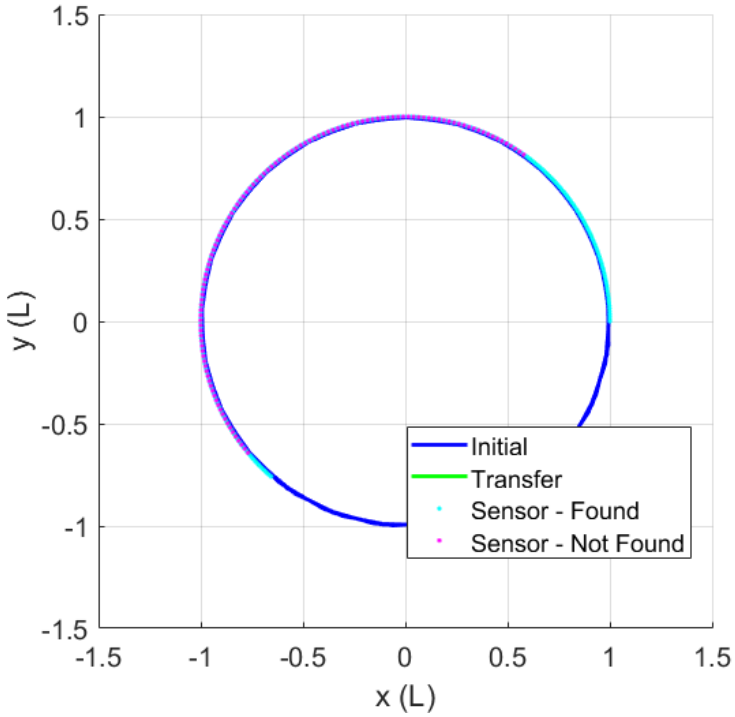


Figure 3.31. Optimal Trajectory for Scenario 3 - Optimal Control

3.3.2.2 Proximal Policy Optimization

The PPO methodology described in Chapter 2 was implemented with the goal of optimizing the same three scenarios.

Scenario 1, the station keeping test case, produced the trajectory found in Figure 3.32. This resulting trajectory looks very similar to the one produced using the optimal control method, but notice that the colors are different. Since a control strategy is sampled to produce this trajectory, though the overall transfer is optimal with respect to the objective function, the input at each time step is not. Therefore, a pattern develops in the transfer in which the spacecraft is overall successful in evading the sensor for varying time periods.

Unfortunately, PPO was unable to produce a converged solution for Scenario 2 (a terminal constraint requiring that the spacecraft arrive at a specified state (position and velocity) after some fixed time). This is almost certainly due to the fact that PPO can not take into account hard constraints. Instead, the terminal constraint is applied as a soft constraint in the objective function. No weight applied to the soft constraint was found to be sufficient enough to overcome to minimization of the propellant usage and maximization of the distance between the spacecraft and the sensor.

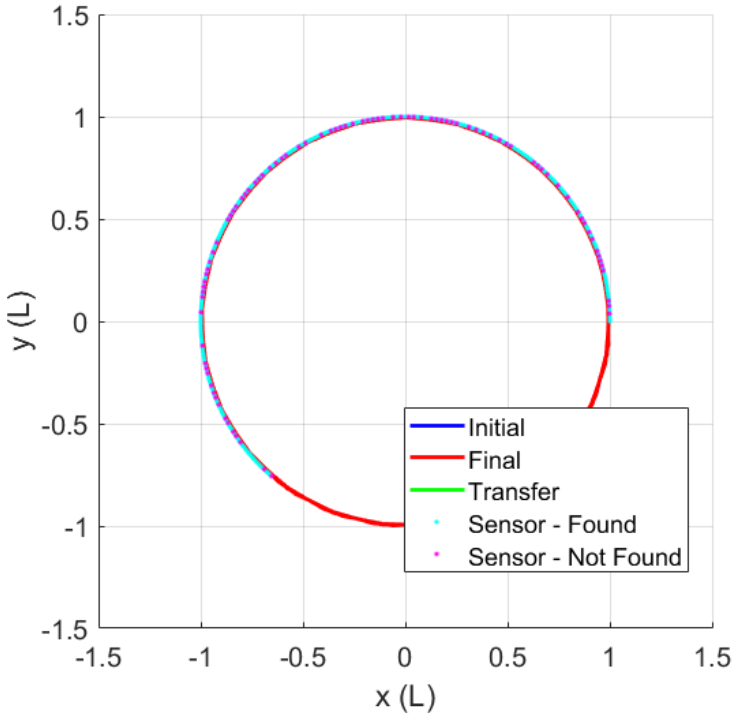


Figure 3.32. Optimal Trajectory for Scenario 1 - PPO

Scenario 3, the scenario without a terminal constraint, produced the trajectory found in Figure 3.33 when using PPO. Originally, PPO was producing an intuitive result where a trajectory was produced that varied greatly from the nominal orbit - as would be expected when a primary objective is to avoid detection by a sensor. However, in the interest of comparison with the optimal control solution, the objective function was changed to put additional emphasis on the propellant usage. This resulted in a plot similar to that of the station keeping scenario.

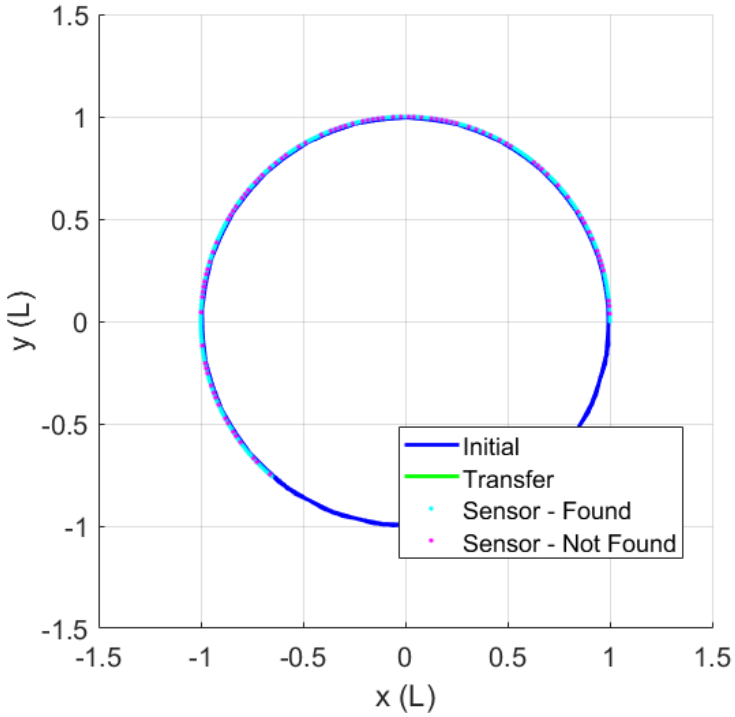


Figure 3.33. Optimal Trajectory for Scenario 3 - PPO

3.3.2.3 Comparison

In the course of implementing PPO for optimizing the same scenarios, it was discovered that, as expected, PPO has some drawbacks compared to direct transcription. Though the Keplerian dynamics are seamlessly maintained in the environment, terminal constraints on the dynamics can only be implemented as soft constraints in the objective function. It is also relevant that the optimization is stochastic so the learning does not converge every time. In addition, though the solution runs very fast on a trained policy, the training itself takes about twice as long than when using the optimal control approach. However, it is also true that PPO is global over the theatre of operations, meaning that similarly accurate results can be expected when the problem is expanded.

Most significantly, PPO was successfully used to optimize the trajectories in two of the three scenarios. The question that remains is how the PPO results look when compared to the optimal control solutions. To compare the results of Scenario 1, the two plots can be overlapped, with the PPO solution in black, as seen in Figure 3.34. The plots look to overlap completely, but can't say for certain without zooming in. Taking a look at the very end of the trajectory (around $[-0.7, -0.7]$) in Figure 3.35, its obvious

that the two solutions are exactly the same (the error is essentially zero). Note that the multiple lines in red are from multiple revolutions of the initial trajectory, the black line is the transfer orbit found using PPO, and the green line is the transfer orbit found using SQP.

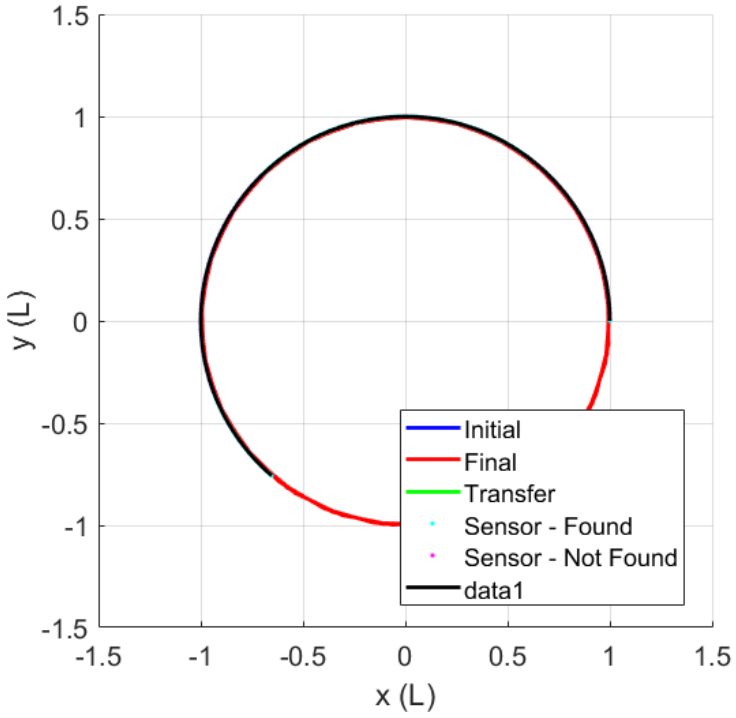


Figure 3.34. Solution Comparison for Scenario 1

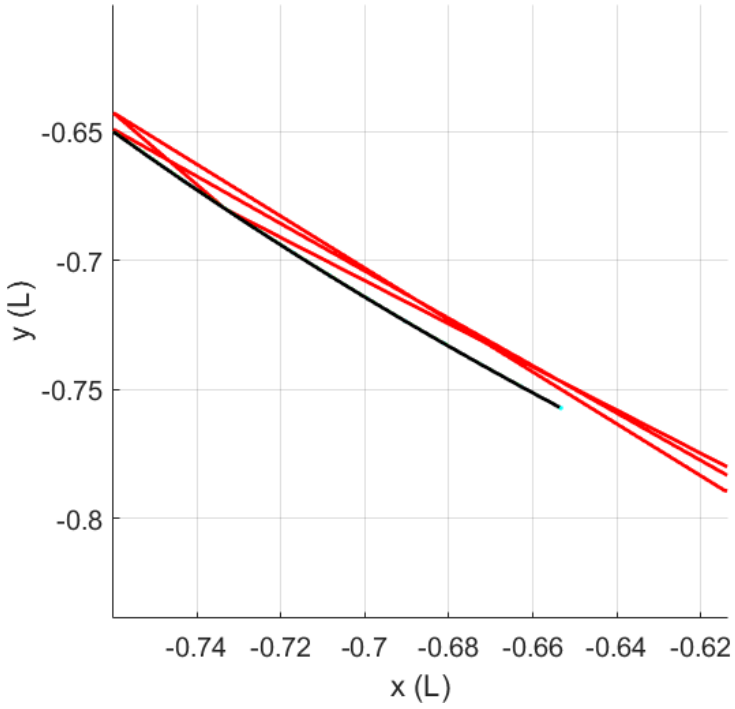


Figure 3.35. Solution Comparison for Scenario 1 (Zoomed)

A similar comparison was performed for Scenario 3, skipping Scenario 2 since PPO was unable to produce meaningful results. Again, the two solutions can be seen overlapped in Figure 3.36 and, again, the end of the trajectory is zoomed in on (Figure 3.35). In this scenario, there is a slight deviation in the two solutions, but the maximum error was found to be less than 0.5% at the final time.

This justifies the use of PPO over traditional optimal control methods for optimizing the two-player sensor avoidance problem as long as no terminal conditions are required.

Now that the capabilities and limitations of both players as well as the selected policy optimization method have been explored and defined, the decision process and methodology employed can be describe in more detail before presenting the results.

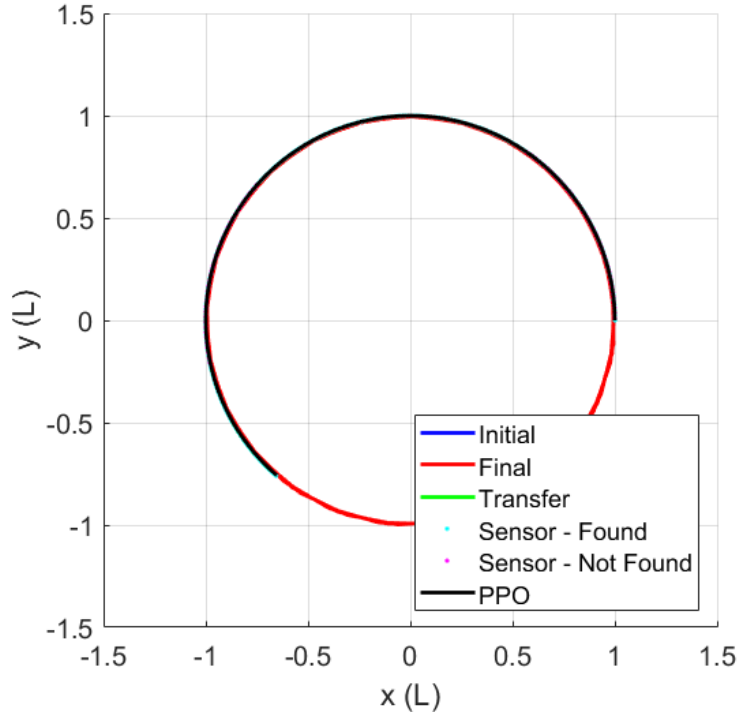


Figure 3.36. Solution Comparison for Scenario 2

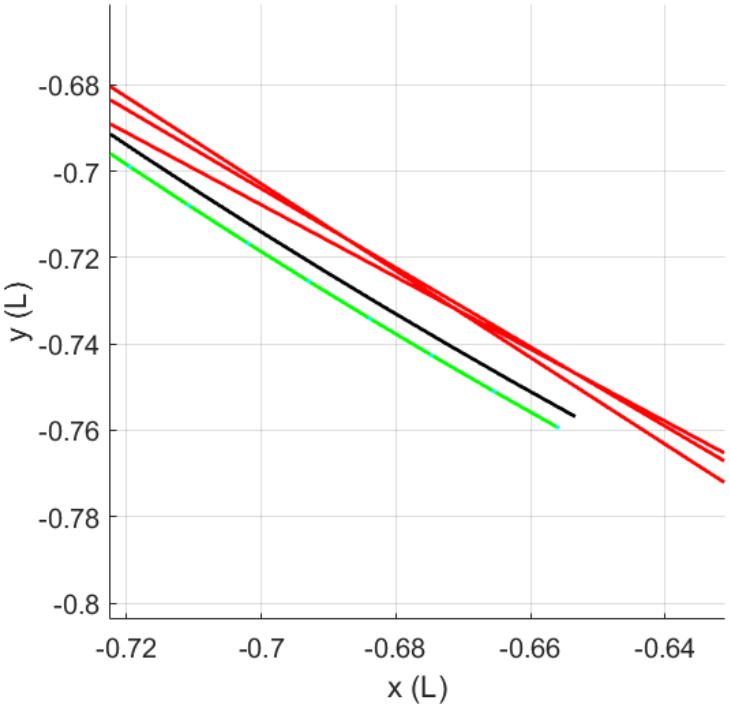


Figure 3.37. Solution Comparison for Scenario 2 (Zoomed)

Chapter 4
Methodology

This chapter goes into greater detail on the two-player game that is being solved. First, the specifics of the problem and the general approach to determining the optimal solution are discussed in Section 4.1. Then, the implementation of the problem specifics into the RL environment is described in Section 4.2. Section 4.3 follows this by showing how the policy optimization methods first introduced in Section 2.5 are combined together. Finally, Section 4.4 goes into the contributions of this research with respect to previous work performed in related areas.

4.1 The Designed Problem

In the devised problem, the primary objective is for a geosynchronous spacecraft to continually avoid detection by a sensor with no attention paid to the final spacecraft state. Though minimizing the control input (propellant usage) is usually important, the sole objective in this case is to minimize the number of succesful observations of the spacecraft. The sensor itself is assumed to be optical, meaning it is constrained by the position of the sun. Since spacecraft in geosynchronous orbit can be viewed only at night, the sensors are constrained to a 12-hour window to take observations, with a defined FOV. In previous work by the author [85, 86], presented in Section 6.1, the sensor followed along the trajectory that the spacecraft, propagated from its previous point with no additional input, was assumed to be following. If at any point the spacecraft was not found within the sensor's FOV, then the sensor would move in a sinusoidal pattern of gradually increasing amplitude along the nominal path until the spacecraft was found again. Figure 4.1 shows this pattern, as zoomed-in on the orbital transfer (green line) just after departing from the initial orbit (red line).

For the final iterations (the results can be seen in Sections 6.2 and 6.3), like presented

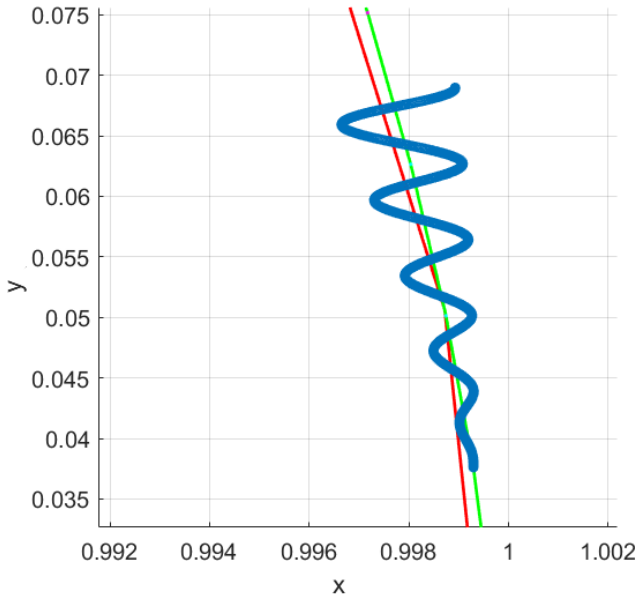


Figure 4.1. Example of the Sinusoidal Tasking Strategy - Non-dimensionalized

in Reference [87], the sensor is no longer assumed to follow a prescribed pattern. Instead, RL is used to optimize the strategies representing the state-actions pairs for both the spacecraft and the sensor simultaneously.

The spacecraft is prescribed a maximum ΔV of 20 m/s in each direction. An infinite time-horizon is assumed, but the time between each maneuver is bounded between 12 minutes and 10 days. With the number of maneuvers capped at 30, this places an upper limit on the time-horizon at 300 days. Each maneuver is applied instantaneously and is represented as a change in the velocity in the $\hat{x}$ and $\hat{y}$ directions in the ECI frame. Meanwhile, the sensor collects observations every 12 minutes during the 12-hour observation window and is assumed to have a FOV of 4°. Since there are no Keplerian dynamics to constrain the sensor's path, each action is performed with respect to the previous known orbital path of the spacecraft. These actions are bounded between 0 and $4,216$ km (10% of the initial orbital radius) in the $\hat{x}$ and $\hat{y}$ directions. Similar to the described previous work, the sensor is assumed to follow along the spacecraft's path until it is not found within the sensor's FOV. At this point, it is assumed that the sensor has lost custody of the spacecraft and searches the nearby space according to the learned strategy. If the spacecraft is successfully found in the sensor's FOV for 15 consecutive observations (to simulate angles-only IOD initialization), custody is regained and the sensor can begin to follow the spacecraft along its new path without applying the search

strategy. In an attempt to simulate the nuances of state estimation, the sensor is afforded some flexibility in the search process. If the spacecraft has been successfully found for 7 consecutive observations after custody is lost, a missed observation does not restart the count unless the spacecraft is not located for 3 observations in a row, in which case the count to 15 must restart in order to regain custody.

The strategies for both the spacecraft and sensor are represented as state-action pairs. The inputs for both strategies are identical $5x1$ vectors containing the spacecraft's position, the sensor's position, and the simulation time. Both strategies utilize a deep neural network with 4 layers, each employing a sigmoid activation function, to find the desired output action. The spacecraft's strategy outputs a $3x1$ vector containing the change in velocity in the $\hat{x}$ and $\hat{y}$ directions and the time until the next maneuver. The sensor's strategy outputs a $2x1$ vector containing the variation of the search state in the $\hat{x}$ and $\hat{y}$ directions with respect to the suspected location of the spacecraft based on its previously known state. After each strategy is applied, both players are given a reward, opposite in value from each other (as required for a zero-sum game). Since the reward must directly relate to the apparent success of the applied strategies, the normalized distance between the center of the sensor's FOV and the spacecraft's position was selected, represented as

$$r = ||\vec{x}_{S/C} - \vec{x}_S|| \tag{4.1}$$

where $\vec{x}_{S/C}$ is the spacecraft's state and $\vec{x}_S$ is the sensor's state. PPO is then applied on batches of these state-action-reward sets in order to update the strategies. As more and more updates are performed, the strategies will converge to a NE for the given state and action spaces and constraints.

Since these spaces and constraints are highly subjective, competitive coevolution is then applied to investigate how the strategies can be adapted, free of bounds and constraints, to achieve specific goals (the percentage of successful observations). For this research, the population size was set to 20, the maximum number of generations was set to 3, and the maximum number of coevolutions was set to 10. Though these are arbitrary and slight adjustments could result in smoother convergence, these values appear to be acceptable.

In applying the competitive coevolution in Chapter 6, as discussed in Section 2.2.3.3, competitive fitness sharing, shared sampling, and Hall of Fame are all valid strategies for ensuring global optimality in competitive coevolution. However, it was found in the authors' previous work applying competitive coevolution [88] that, given the inherent continuity in the search space, competitive fitness sharing was found to be unnecessary.

In addition, poorly represented types was not an issue with this specific game as parasites were never found that could only be defeated by specific hosts. Similarly, shared sampling was also found to be an unnecessary burden on the optimization. With a relatively small number of iterations required to reach convergence, the Halls of Fame for the two players remained rather small. Instead of speeding up convergence, sampling from the opponents Hall of Fame each iteration greatly slowed down the convergence since less optimal strategies were being added to the current player's Hall of Fame. However, applying the Hall of Fame technique alone was found to be more than sufficient. With a win condition set based on the percentage of observations that successfully find the spacecraft, the Hall of Fame approach collects a mixture of the most uniquely successful strategies of both players and evolves this mixture free of constraints until one of the players can no longer evolve its mixed strategy in such a way that allows it to meet the win condition.

In Section 6.2, the win condition is a pre-defined threshold with respect to the value of the game and each players' sole objective is to meet this condition. The winning player, and the NE optimal strategies, can be determined when one player can no longer evolve to pass the threshold value. In Section 6.3, the win condition is no longer fixed and is instead based on the other player's current best value of the game. The players will continue to evolve until the value of the game converges to a single point and the winner will be declared based on subjective factors.

4.2 Reinforcement Learning Environment

The designed problem described in the previous section is implemented primarily in the RL environment. The pseudo-code for this implementation can be found in Algorithm 9. In the algorithm,

- δt is both the time step taken by the spacecraft propagator and the time between observations collected by the sensor,
- FOV is the field-of-view in kilometers,
- $max - man$ is the maximum number of maneuvers the spacecraft is allowed to take,
- $\vec{X}_{s/c}$ is the spacecraft state,
- $\vec{X}_{ref}$ is the spacecraft orbit previously known by the sensor,

- $\vec{X}_s$ is the position of the center of the sensor's FOV,
- num_{man} is the number of maneuvers previously applied by the spacecraft,
- $found$ is the logical flag for if the spacecraft was most recently found within the sensor's FOV,
- $count_{found}$ is the number of observations that the spacecraft has been found in the sensor's FOV in a row,
- $count_{not-found}$ is the number of observations that the spacecraft was not found in the sensor's FOV in a row,
- $\Delta\vec{V}$ is the change in the spacecraft's velocity vector applied as a maneuver,
- $\delta\vec{x}_s$ is the change in the sensor's position with respect to the $\vec{X}_{ref}$,
- $apply - man$ is the logical switch for whether or not the spacecraft is applying a maneuver this step through the environment, and,
- $lighting$ is the logical check for if the spacecraft is located inside the Earth's shadow such that the sensor can observe it.

This environment is run through once every time step and the outputs are returned to the main function that is performing the policy optimization as described in the next section.

Algorithm 9 Reinforcement Learning Environment

Given constants δt, FOV, $max - man$
Initialize $\vec{x}_{s/c} = [1, 0, 0, 1]$, $\vec{x}_{ref} = \vec{x}_{s/c}$, $num_{man} = 0$, $found = 1$, $count_{found} = 30$, $count_{not-found} = 0$
Variables from Main Function: ΔV, $\delta\vec{x}_s$, $apply - man$, $lighting$
if $apply - man$ **then** $num_{man} += 1$
if $lighting$ **then** $\delta t = \delta t$ **else** $\delta t = \delta t \times 10$
Propagate $\vec{x}_{s/c} + \delta V$ for δt
Propagate $\vec{x}_{ref}$ for δt
    $\vec{x}_s = \vec{x}_{ref} + \delta\vec{x}_s$
if $lighting$ **then**
    $d = \text{norm}(\vec{x}_{s/c} - \vec{x}_s)$
    if $d \leqslant$ FOV **then**
        if $d == 0$ **then**
            $count_{found} = 0$, $\vec{x}_{ref} = \vec{x}_{s/c}$
        else
            $count_{found} += 1$
            if $count_{found} \geqslant 15$ **then** $\vec{x}_{ref} = \vec{x}_{s/c}$
        $found = 1, count_{not-found} = 0$
    else
        $found = 0$
        if $count_{found} \geqslant 7$ **then**
            $count_{not-found} += 1$
            if $count_{not-found} \geqslant 3$ **then** $count_{found} = 0$
        else
            $count_{found} = 0$
else
    $found = 0$
    if $count_{found} \geqslant 7$ **then**
        $count_{not-found} += 1$
        if $count_{not-found} \geqslant 3$ **then** $count_{found} = 0$
else
    $count_{found} = 0$
$cost_{s/c} = \text{norm}(\vec{x}_{s/c} - \vec{x}_s)$
$cost_s = cost_{s/c}$
if $num_{man} += 1 \geqslant max - man$ **then** done
return $cost_{s/c}, cost_s, done$

4.3 Combining Policy Optimization Methods

In most policy optimization uses, PPO is employed on its own with the Adam optimizer used to train the policy and value functions. In some cases, TRPO is implemented alongside PPO to improve the optimization process. Reference [89], however, implemented the combined PPO and TRPO optimization algorithm while using GAE to improve the advantage and value function estimation. This unique combination of these three methods proved useful in optimizing the strategies for this two-player game.

As discussed earlier in this chapter, the policies for both the spacecraft and sensor are represented as state-action pairs. The inputs for both policies are identical $5x1$ vectors containing the spacecraft's position, the sensor's position, and the simulation time. Both policies utilize a deep neural network with 4 layers, each employing a sigmoid activation function at the final layer, to find the desired output action. The spacecraft's policy outputs a $3x1$ vector containing the change in velocity in the $\hat{x}$ and $\hat{y}$ directions and the time until the next maneuver. The sensor's policy outputs a $2x1$ vector containing the variation of the search state in the $\hat{x}$ and $\hat{y}$ directions with respect to the suspected location of the spacecraft based on its previously known state. The values produced by both networks resemble the expected values (means) of the outputs where their respective standard deviations continually decrease as the optimization progresses and the confidence in the means improves. The means and standard deviations are then used to produce a normal distribution from which the actions (the outputs) are sampled. These neural networks can be seen visualized in Figures 4.2 and 4.3, respectively. Note that the output layer employs a sigmoid activation function while the hidden layers all use tanh activation functions.

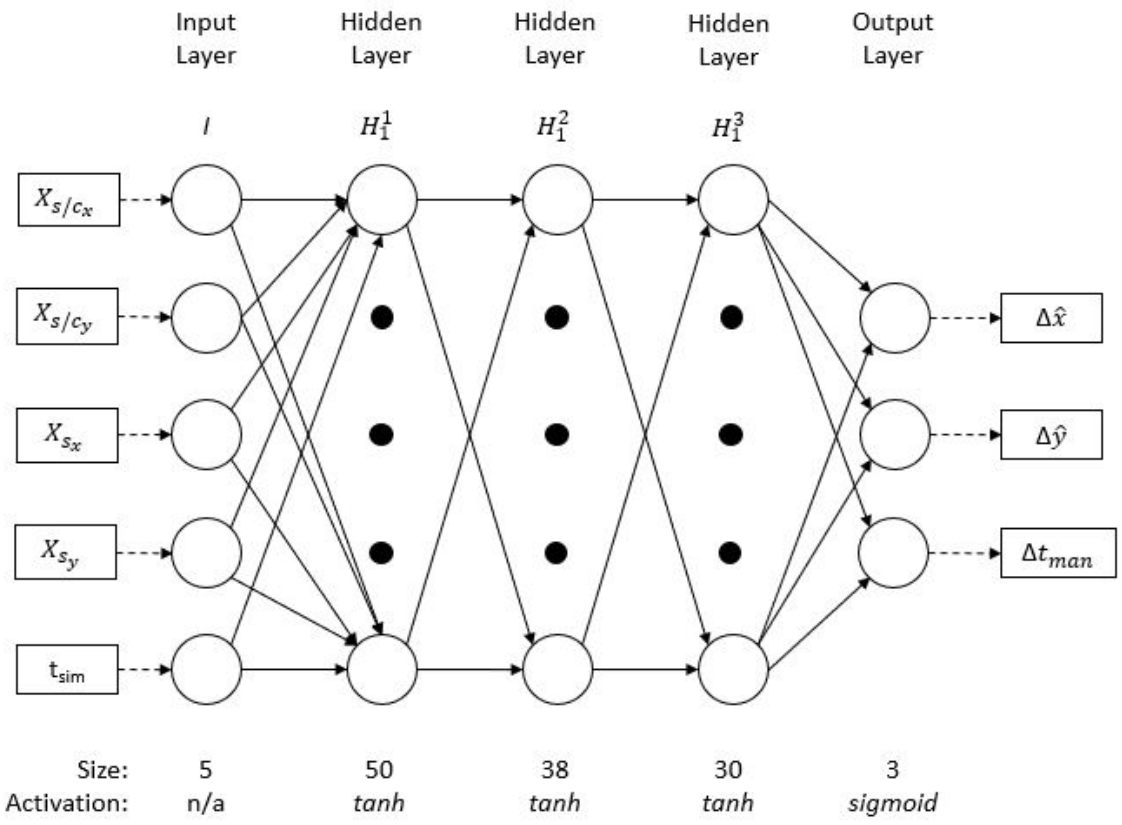


Figure 4.2. Spacecraft Policy Neural Network

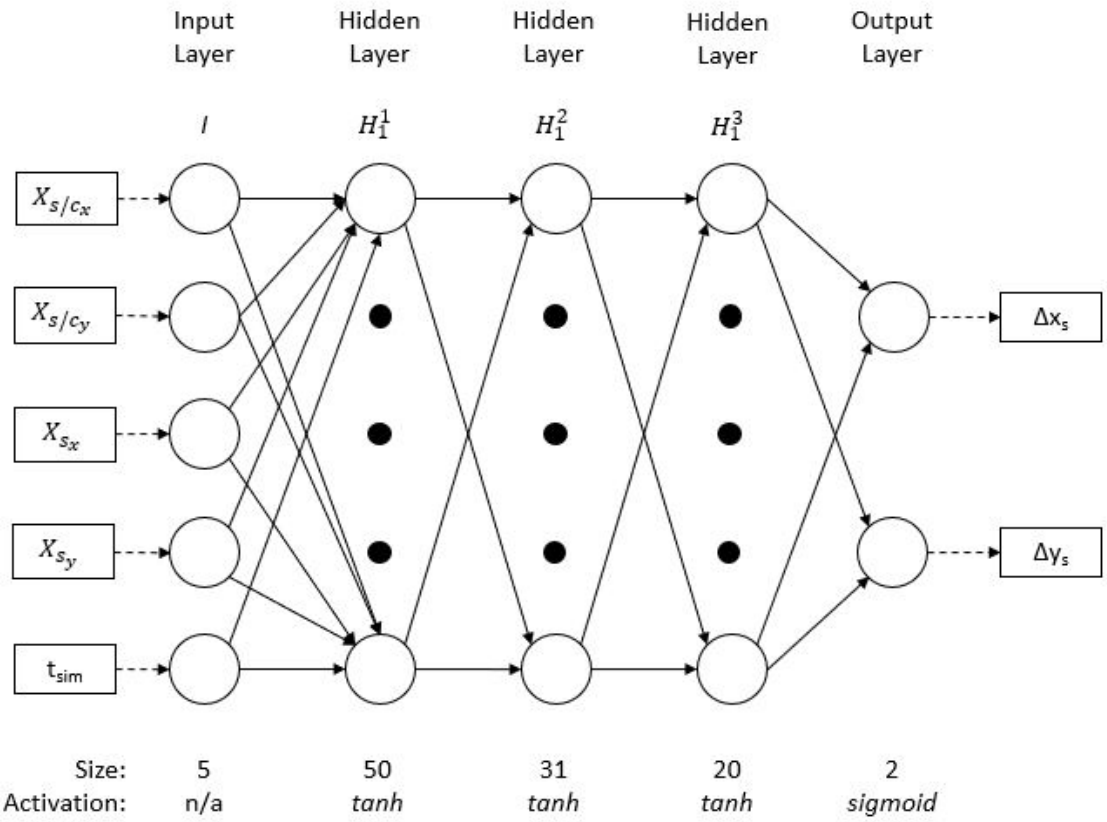


Figure 4.3. Sensor Policy Neural Network

As discussed in Section 2.5.2.3, the policies are updated by minimizing their respective loss values. Using the combined PPO/TRPO approach, the loss values are formulated as

$$L^{\text{KLPEN}}(\theta) = \hat{E}_t \left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)} \hat{A}_t - \beta \text{KL}\left[\pi_{\theta_{old}}(\cdot|s_t), \pi_\theta(\cdot|s_t)\right] \right] \tag{4.2}$$

where the advantage, A_t, is calculated using the GAE method as

$$\hat{A}_t^{\text{GAE}(\gamma,\lambda)} = \sum_{l=0}^{\infty} (\gamma\lambda)^l \delta_{t+1}^V \tag{4.3}$$

and the divergence can then be calculated using

$$D = \hat{E}_t \left[\text{KL}\left[\pi_{\theta_{old}}(\cdot|s_t), \pi_\theta(\cdot|s_t)\right] \right] \tag{4.4}$$

Then, if $D < D_{targ}/1.5$ then β is set to $\beta/2$. If $D > D_{targ} \times 1.5$ then β is set to $\beta \times 2$. This updated β value is then used in the next policy update. The loss values are then

minimized using the Adam optimizer, as explained in Section 2.5.4. The learning rate of the optimizer is initialized to be 2.37×10^{-5} for the spacecraft optimization and 2.9×10^{-5} for the sensor optimization, determined somewhat subjectively based on the size of the neural network layers. The learning rate is then continually adjusted based on the value for β such that if $D > D_{targ} \times 2$, $\beta > 30$ and the learning rate multiplier is currently > 0.1, then the multiplier is divided by 1.5. Otherwise, if $D < D_{targ}/2$, $\beta < 1/30$, and the learning rate multiplier is currently < 10, then the multiplier is multiplied by 1.5. This ensures that the learning rate moves in the opposite direction of β and encourages slower updates when the loss is making larger changes based on the KL divergence value.

The value function neural networks are designed similarly to the policies except the output layers are both linear. The neural networks for the two player look identical and can be seen in Figure 4.4.

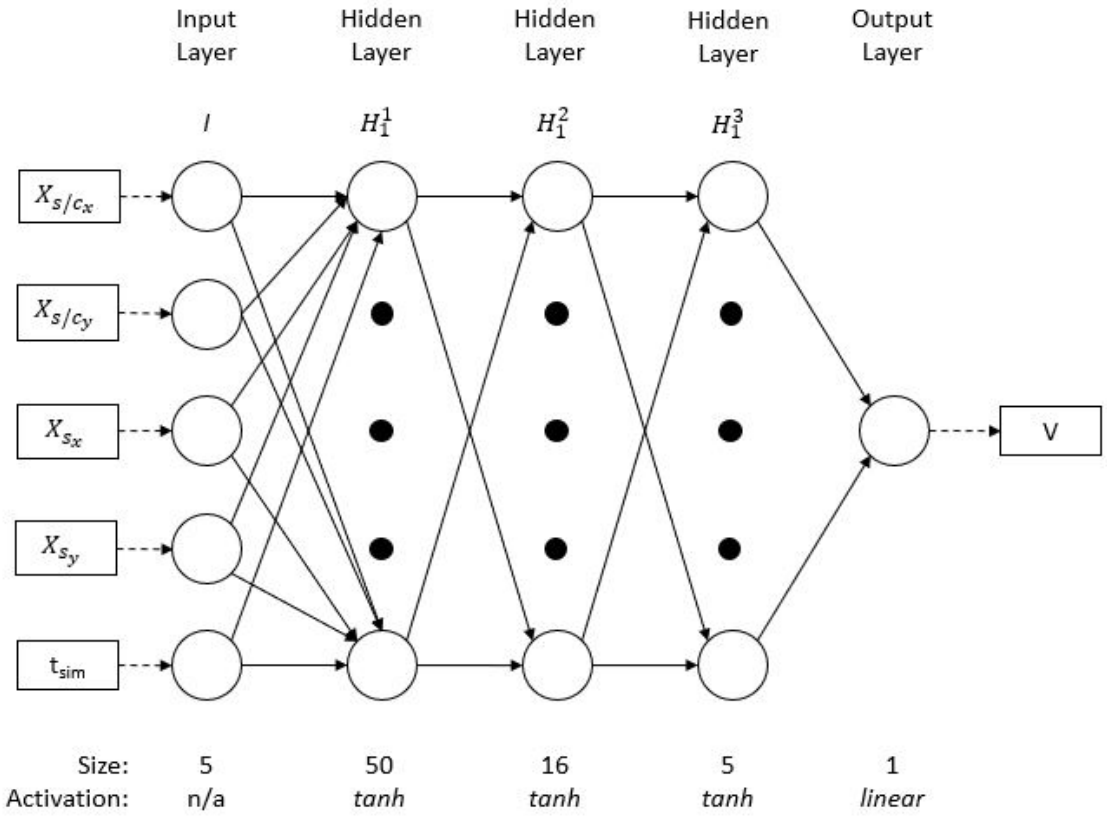


Figure 4.4. Value Function Neural Network

The value functions are optimized using the GAE process explained in Section 2.5.3.

Using this method, the value function is updated using

$$\begin{aligned} &\min_{\theta} \sum_{n=1}^{N} ||V_\theta(s_n) - \hat{V}_n||^2 \\ &\text{subject to} \frac{1}{N} \sum_{n=1}^{N} \frac{||V_\theta(s_n) - V_{\theta_{old}}(s_n)||^2}{2\sigma^2} \leqslant \epsilon \end{aligned} \tag{4.5}$$

where a similar KL divergence constraint is applied as is used in the policy optimization.

Prescribing values for discount factor, γ, the reward steepness, λ, the KL divergence target, and the batch size provides the required inputs to begin the optimization process. As the policies and value functions are continually updated, the spacecraft and sensor learn from their identical reward value and their policies converge toward their respective NE.

4.4 Variations From Previous Application

This research is original in a number of distinct ways. First, as of the conception of this dissertation, no other research had been published applying PPO to astrodynamics problems. Since then, a handful of papers have been published doing just that while this research was being performed. Applying PPO to environments constrained by Keplerian dynamics has some inherent benefits and drawbacks. On the negative side, the application of a change in velocity at one point can have unpredictable consequences at much later times in the simulation so the value function neural network has to attempt to estimate this. Notably, this results in much smaller values of γ being used to ensure that a longer time frame of data is included when updating the policy and value functions. On the positive side, as long as the value function can learn to approximate the Keplerian propagation, having constraints on the spacecraft's state limits its state space at any moment in time.

The unique contribution of this work primarily lies in the application to sensor avoidance. All other applications of PPO within astrodynamics as of yet involve a single player optimizing a trajectory with a defined time or location to trigger the end of an episode, usually on the order of less than an hour to a couple of months. In these scenarios, continuous thrust transfers are generally considered with reasonable step sizes. Unfortunately, none of those assumptions could be made for this research. Most significantly, this problem requires simultaneously optimizing the strategies of two players. So instead of optimizing an objective like propellant use or time-of-flight, both of

which depending only on the actions of the spacecraft and its interaction with the natural environment, both players are attempting to optimize their strategies with respect to an objective that is dependent on the other player's actions, creating an environment that is constantly changing unpredictably. Even in non-astrodynamics applications, applying PPO to multi-player games has only gotten so far as to optimize board, card, and video games - nothing so far in the real-world environment.

A secondary effect of optimizing this two-player game is that the speed at which the two players make decisions is different. While the sensor is constantly being tasked in search of the spacecraft and has a largely finite number of moves, the spacecraft is limited to a finitely small number of moves (30 in this case). This means that, though the two players will be updating their strategies at the same rate, the pursuing player will have much more information to update its strategy based off of. However, given the fact that the spacecraft is only maneuvering 30 times, the amount of effectively new information provided to the sensor every update will be equally small.

Furthermore, most astrodynamics applications of PPO have had maximum simulation times multiple orders of magnitude less than what is required for this type of application. Given the sheer number of orbit propagation calls required, a larger step size must be chosen to achieve a barely-even-close-to-reasonable computation time. With a step size on the order of 6 to 12 minutes selected for these simulations, this results in rather sparse observations of the spacecraft's state. With maneuvers being performed on the order of days, meaningful observations are even more sparse. This requires extremely large batch sizes and small KL divergence targets to ensure that the Adam optimizer has enough meaningful information to perform an effective update.

Along these same lines, basing the length of each episode off of the number of maneuvers rather than fuel expenditure or time-of-flight required an alteration to the policy optimization approach. The original publications on PPO and similar policy optimization methods dictate that the reward signal be the sum of each individual reward provided to the player throughout the entire episode. While this may be effective when optimizing the total ΔV, it is counter-productive when optimizing the distance between the center of the spacecraft and the sensor's FOV. Doing so would mean that, while the spacecraft is trying to maximize the objective, it could do so by increasing the length of the variable-length episode. Conversely, the sensor would be interested in decreasing the episode length to decrease the total value. Therefore, instead of using the total reward for each episode, the average reward is used for the optimization.

Finally, no open source PPO code, including the code published by the original authors,

has found an effective way of applying bounds on the selected actions to be performed by the two players. Most implementations simply apply an arctangent activation function to bound the outputs of the neural networks modeling the mean to between -1 and 1, scale the outputs by a scalar value, and then clip the resulting values to meet the desired bounds. Doing this, however, can potentially result in statistically significant portions of the probability distribution being ignored since the neural network determines only the mean, not its corresponding standard deviation, which is added in later. While this may not negatively affect the results in other applications of PPO with smaller standard deviations, that was not the case here. The results from this approach, tested on a simpler problem presented in the Appendix, often "converged" to the extremes of the action space instead of to the actual optimal values. Instead, the arctangent activation function was first changed to a sigmoid activation function to bound the outputs of the mean to between 0 and 1 (as seen in Figures 4.2 and 4.3). Then the resulting probability distribution (the neural network) is combined with the standard deviation to produce a truncated normal distribution with ranges specified individually for each output. This allows for the entirety of the distribution to be represented in the outputted values. The results with this adjustment were much closer to the expected optimal values and sufficiently converged without approaching the extremes of the action space.

The results of the contributions and variations from previous applications of PPO can be seen applied to this problem in the following chapter.

Chapter 5
Reinforcement Learning Results

Section 4.1 detailed the designed problem in all steps of implementation while the Appendix demonstrated that a simplified version of this two-player zero-sum game will converge using PPO. This chapter presents the results through three different iterations of the problem. First, the results are explained when only the spacecraft maneuvers are being optimized by PPO and the sensor follows a prescribed tasking pattern. Next, the sensor tasking is added to the optimization process and the initial results are presented. Finally, lessons learned from the previous iterations are applied to the final iteration and the nearly fully converged strategies are discussed.

5.1 Avoiding A Sensor Using A Prescribed Sensor Tasking Pattern

5.1.1 Problem Description

The first iteration when applying PPO attempts to optimize only the spacecraft's maneuvers while assuming that the sensor tasking pattern is pre-defined and unchanging.

The spacecraft player is attempting to simultaneously minimize the total ΔV applied while maximizing the distance between it and the center of the sensor's FOV. Because the sensor is not yet included in the optimization, the ΔV can be included in the objective function. This is primarily implemented in these results to demonstrate how it affects the maneuver decisions with respect to the sensor avoidance goals.

The sensor itself is assumed to be optical with a defined FOV. By default the sensor will follow along the trajectory that the spacecraft, propagated from its previous point with no additional input, is assumed to be following. If at any point the spacecraft is

not found within the sensor's FOV, then the sensor will move in a sinusoidal pattern of gradually increasing amplitude along the nominal path until the spacecraft is found again. Figure 5.1 shows this pattern, as zoomed-in on the orbital transfer (green line) just after departing from the initial orbit(red line).

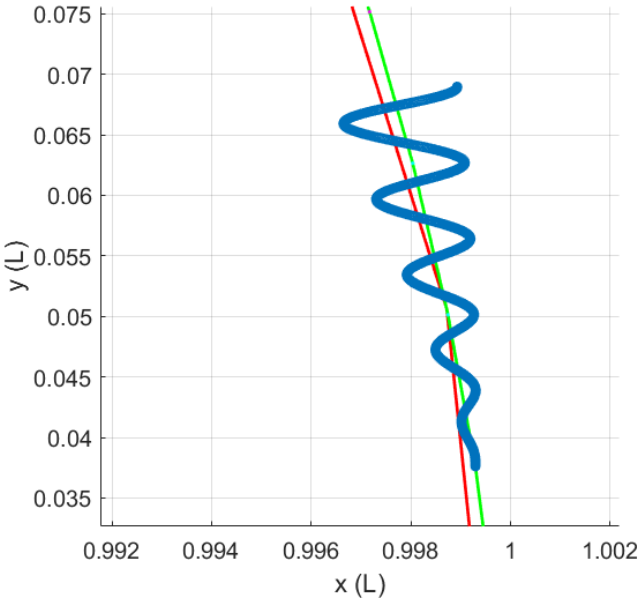


Figure 5.1. Example of the Sinusoidal Tasking Strategy

Meanwhile, the spacecraft is attempting to avoid detection by the tasked sensor. The spacecraft is allowed to perform a set number of impulsive maneuvers, applied instantaneously and represented as a change in the velocity in the $\hat{x}$ and $\hat{y}$ directions, each with a maximum time allowed before the next maneuver and a maximum ΔV. This ensures that propellant is used most efficiently and allows for the entire maneuver strategy to be optimized. For this exercise, the numbers of maneuvers was set at 30, each with a maximum ΔV of 12 m/s and a maximum time between maneuver of 8 days.

The hyper-parameters used for this optimization, based on those used in Reference [89] and altered through trial and error, can be found in Table 5.1.

Table 5.1. Hyperparameters Used in PPO Training

Parameter	Value
Discount Factor, γ	0.8
Batch Size	320
Reward Steepness, λ	0.98
KL Divergence Target, D_{targ}	0.003

5.1.2 Results

Given the lack of eccentricity in the nominal orbit, the impulsive maneuvers are not expected to be performed at a specific location along the orbital path. Instead, the timing of the maneuvers should depend purely on the relative weighting placed on the propellant use and the distance between the spacecraft and the sensor FOV in the objective function. If more emphasis is placed on the propellant use, the time between maneuvers should remain close to the maximum time allowed. If more emphasis is placed on the distance between the spacecraft and the sensor FOV, then the maneuvers should be relatively close together in time. The opposite would be true for the ΔV applied for each maneuver. The lesser the weighting on propellant, the smaller the ΔV, and vice versa. Ideally, the objective function weightings would be selected such that a balance is struck with respect to both the time between maneuvers and the maneuver magnitude. For these results, the objective function is formulated as

$$J = ||d|| + \frac{1}{100}||\Delta V||^2 \tag{5.1}$$

where d is set to 100 if $||V_{s/c} - V_s||$ is less than the FOV, expressed in kilometers, and set to $\frac{1}{d}$ otherwise. The units of the objective are irrelevant for the optimization since the relative magnitude of the two values is the only concern.

From the resulting NE, the strategies are sampled for one episode to produce a single set of actions for both players. Figure 5.2 shows the specific locations in the orbit that each maneuver was performed, where each star is an individual impulsive maneuver. As expected, no pattern emerges here. However, Figure 5.3 depicts the resulting time placed between each maneuver is performed. It is obvious here that a pattern has developed. Though not consistently at the maximum, the ΔV appears to have been given too large of a weighting in the objective function as the time between maneuvers remains close to the maximum 8 days. Interestingly, it appears that the times decrease as the simulation

nears its end. This will be explored further below.

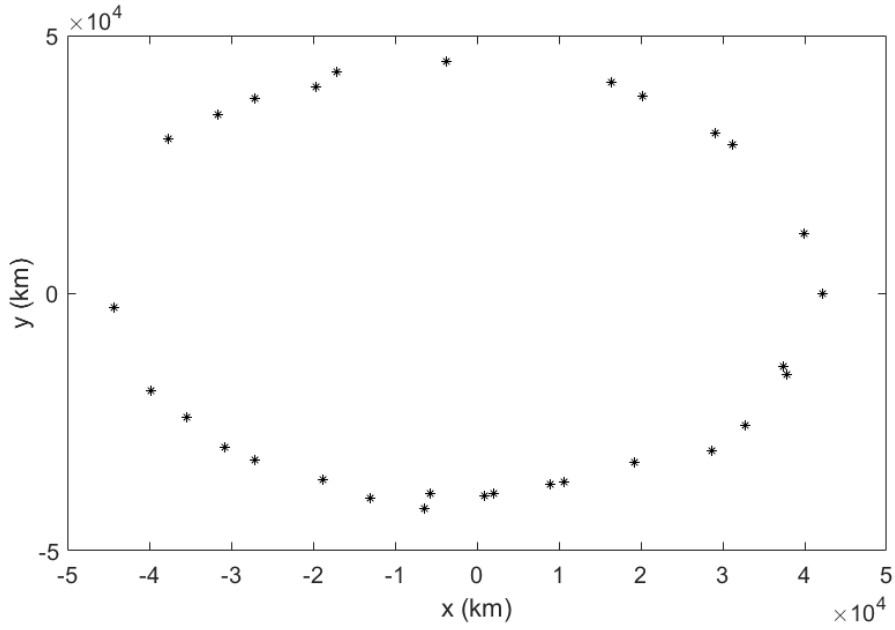


Figure 5.2. Maneuver Locations

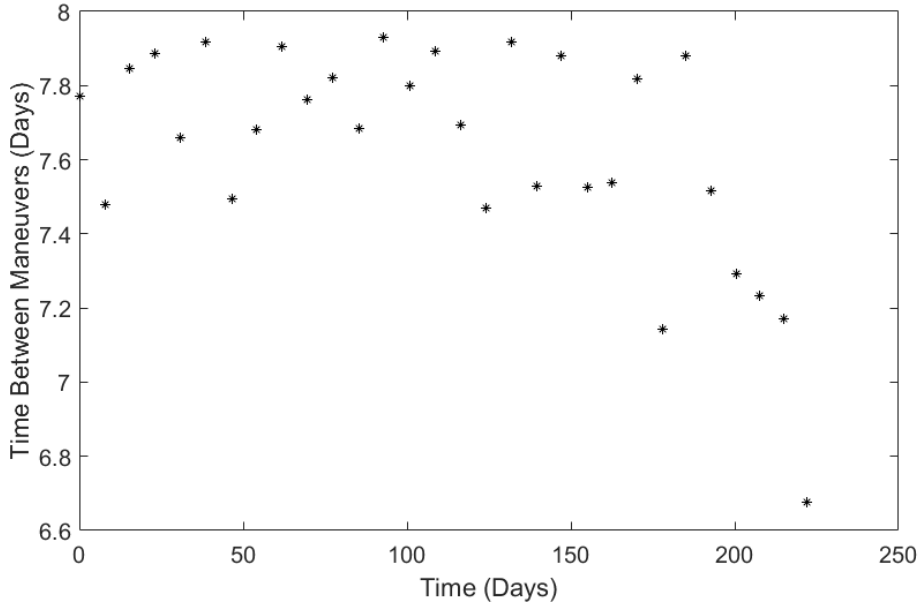


Figure 5.3. Time Between Maneuvers

The ΔV in the $\hat{x}$ and $\hat{y}$ directions (in the Earth Centered Inertial frame) can also be seen in Figure 5.4. Like the time between maneuvers, these also decrease as the total

time increases, starting out near the maximum allowed 12 m/s and decreasing down to -2 m/s in either direction. Looking at the ΔV magnitudes and directions (represented as the angle between the ΔV vector and the ECI x-axis) separately in Figures 5.5 and 5.6, it is clear that this change over time occurs in both the maneuver magnitude and maneuver direction, not just the magnitude as one might expect.

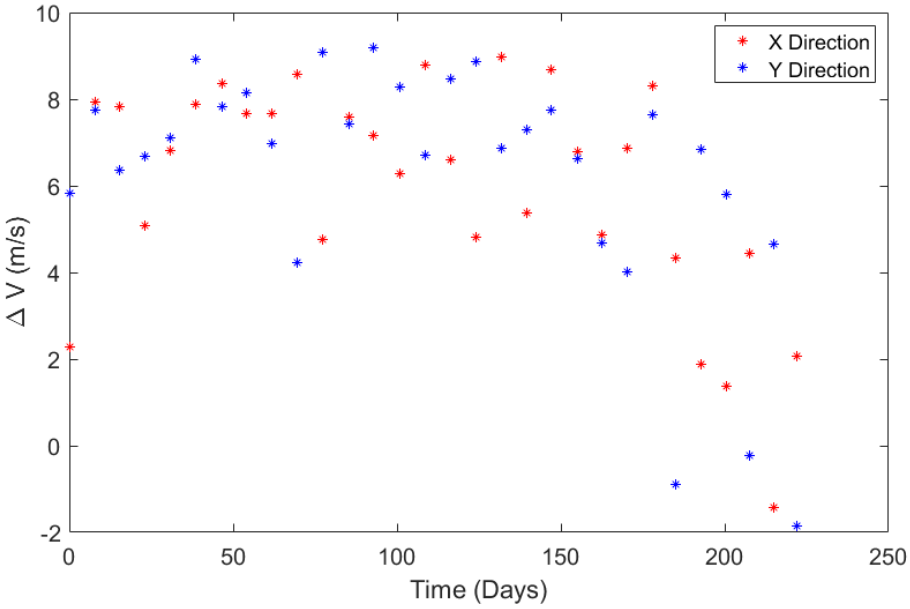


Figure 5.4. Maneuver Magnitude in the $\hat{x}$ and $\hat{y}$ Directions

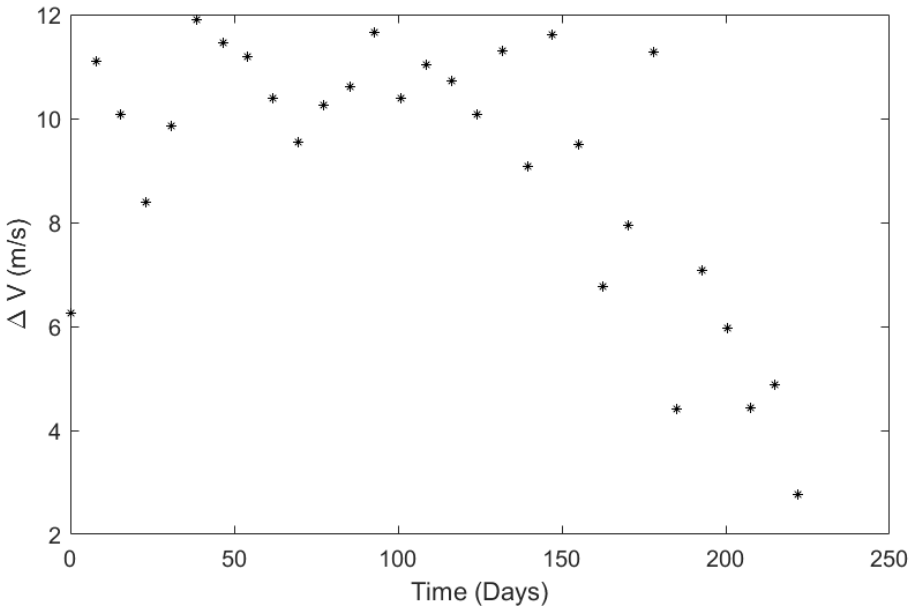


Figure 5.5. Maneuver Magnitudes

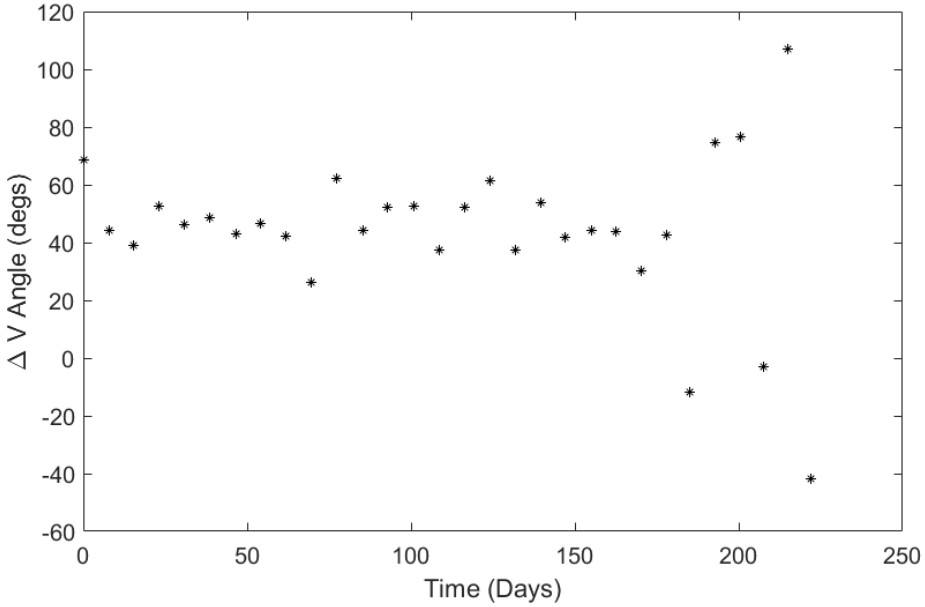


Figure 5.6. Maneuver Angle From the ECI X-Axis

Though both the time between maneuvers and the ΔV decrease together, this is not meaningful without more information. Larger ΔVs at larger intervals can be equivalent to smaller ΔVs at smaller intervals. To get an idea for the true tendency over time, these

values are divided into each other (Figure 5.7). Though it can't be known exactly given the non-linearity of the dynamics, the resulting curved line suggests that the decrease in the time between the maneuvers does not do so enough to offset the simultaneous decrease in the ΔV - the maneuvers are actually tapering off. This suggests that, when avoiding a sensor with a fixed search pattern, it is possible to successfully avoid being tracked after enough maneuvers of a sufficient magnitude have been performed.

It turns out that a 10.3 m/s maneuver every 7.6 days is most effective at avoiding detection by the sensor with the selected objective function weights. In addition, the results suggest that the most effective maneuver direction for avoiding a sensor is 45° from the horizontal.

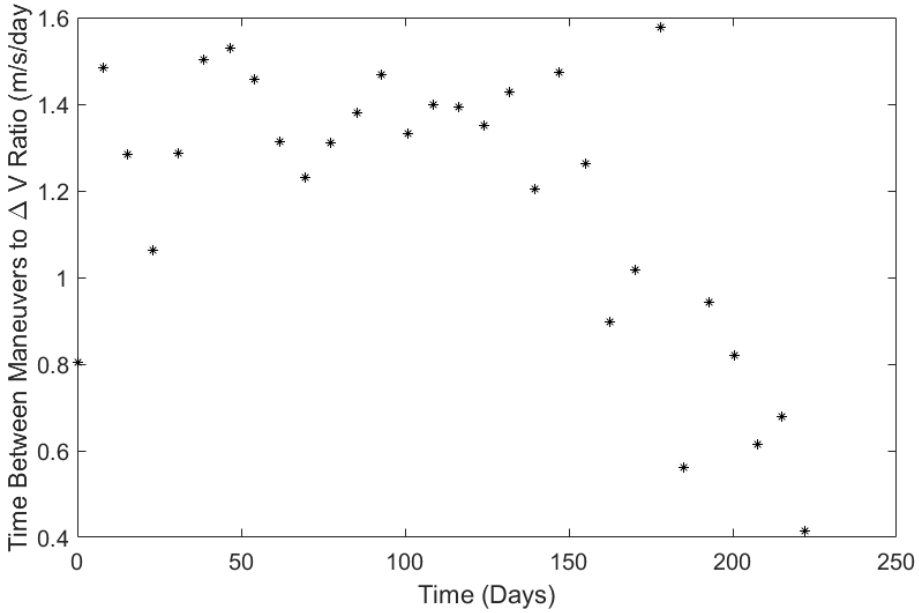


Figure 5.7. Ratio Between The Maneuver Times and ΔV

Figure 5.8 shows the resulting trajectory from the 30 maneuvers. Here, the green markers denote when the spacecraft was found by the sensor, while the red markers denote when the spacecraft successfully evaded detection. Overall, the sensor was successful 72% of the total duration, meaning that the spacecraft went undetected for over 61 days. Looking more closely at these results, it was discovered that the periods of successful evasion occurred primarily in large chunks spread evenly throughout the transfer (Figure 5.9).

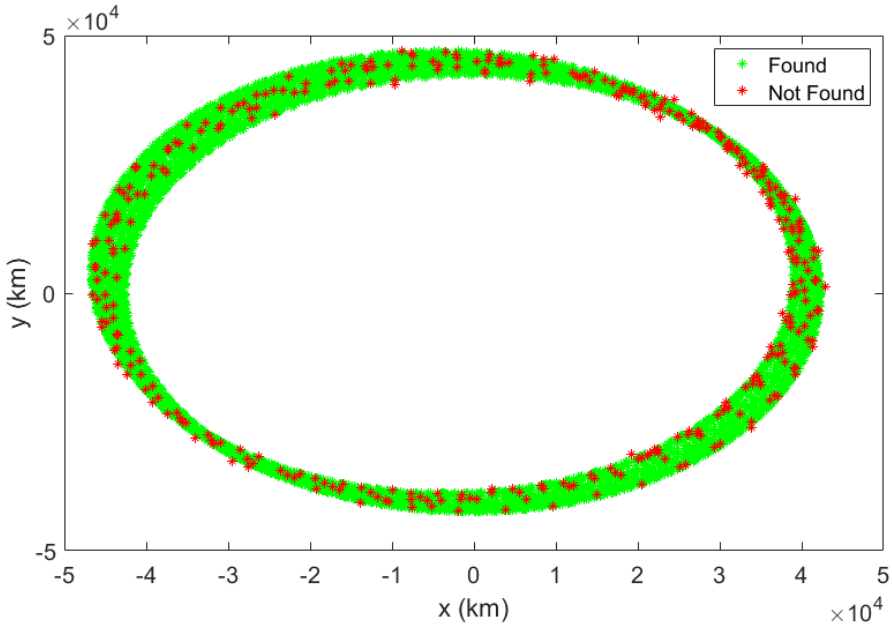


Figure 5.8. Spacecraft Path With Sensor Success

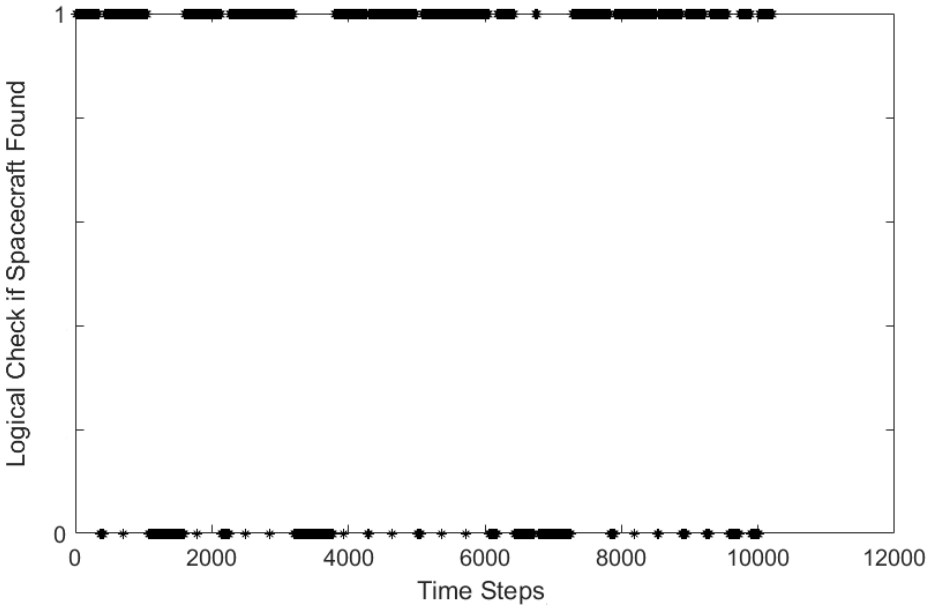


Figure 5.9. Time History of the Sensor's Success

The reason a predictable pattern emerged in the maneuver strategy is because the sensor tasking was programmed with a fixed search strategy. Because of this, the RL is able to find an optimal interval and magnitude at which to perform the maneuvers

that the sensor is not well suited to track. Though this was expected, it is likely that real-world sensors would not share the same flaws and would react and adapt when the spacecraft maneuvers. Including the sensor tasking in the RL is the most logical way to realistically model the interaction between the two players.

Playing with the objective function weightings would likely result in a similar but different pattern emerging, likely with different time between maneuver and ΔV values. Unfortunately, there is no obvious way to select the desired objective function weightings prior to performing the optimization.

5.2 Preliminary Results of Two-Player Optimization

5.2.1 Problem Description

The primary change implemented in this iteration is that the policy optimization now includes the sensor as well, simultaneously updating the policies and value functions of both players. While assuming a pre-defined tasking strategy was effective, it does not challenge the spacecraft to learn to predict changes in the search strategy and is not representative of tasking performed by a human-in-the-loop or autonomous sensor. Thus, the next step was taken to incorporate the sensor tasking into the RL process. Assuming the initial state of the spacecraft is known, the sensor will learn to predict maneuvers performed by the evasive spacecraft, given only a maximum position variation (or, effective FOR) and FOV.

The objective function was changed as well so that the two players are attempting to optimize the normalized distance between the spacecraft and the center of the sensor's FOV. Though most trajectory optimization aims to minimize ΔV, it was elected to remove that consideration from this research and the ΔV is only included as a constraint. This is because, not only is the sensor avoidance the primary concern and the focus of this research, but optimizing the problem as a zero-sum game is a required premise to assume that the strategies will converge to a NE. The sensor player, though potentially interested in the spacecraft exhausting its resources, is not directly involved in the ΔV application.

Additionally, the position of the spacecraft with respect to the Earth's shadow is considered to determine when the optical observations can be collected. To stress test how this affects the sensor's capability, 2-hour windows after sunset and before sunrise are set where the sensor is allowed to collect observations.

For this iteration of results, the spacecraft is prescribed a maximum ΔV of 20 m/s in each direction. An infinite time-horizon is assumed, but the time between each maneuver is bounded between 6 minutes and 6 days. With the number of maneuvers capped at 30, this places an upper limit on the time-horizon at 180 days. These values were adjusted based on the results from the previous iteration in an effort to simultaneously include a more meaningful action space (the increase in the maximum ΔV) and decrease the computation time (the decrease in the maximum time between maneuvers). Each maneuver is applied instantaneously and is represented as a change in the velocity in the $\hat{x}$ and $\hat{y}$ directions. Meanwhile, the sensor collects observations every 6 minutes during the 2-hour windows after sunset and before sunrise and is assumed to have a FOV of 4°. Since there are no Keplerian dynamics to constrain the sensor's path, each action is performed with respect to the previous known orbital path of the spacecraft. These actions are bounded between 0 and $4,216$ km (10% of the initial orbital radius) in the $\hat{x}$ and $\hat{y}$ directions. Similar to the previous iteration where the tasking strategy is prescribed, the sensor is assumed to follow along the spacecraft's path until it is not found within the sensor's FOV. At this point, it is assumed that the sensor has lost custody of the spacecraft and searches the nearby space according to the learned strategy. If the spacecraft is successfully found in the sensor's FOV for 10 consecutive observations, custody is regained and the sensor can begin to follow the spacecraft along it's new path without applying the search strategy. In an attempt to simulate the nuances of state estimation, the sensor is afforded some flexibility in the search process. If the spacecraft has been successfully found for 5 consecutive observations after custody is lost, a missed observation does not restart the count unless the spacecraft is not located for 3 observations in a row, in which case the count to 10 must restart in order to regain custody.

The hyper-parameters used for this optimization can be found in Table 5.2.

Table 5.2. Hyperparameters Used in PPO Training

Parameter	**Value**
Discount Factor, γ	0.8
Batch Size	320
Reward Steepness, λ	0.98
KL Divergence Target, D_{targ}	0.003

5.2.2 Results

Initialized from random normal distributions, the strategies employed by the two players were iterated upon using PPO for just over $100,000$ iterations. More iterations would be ideal, though, as the great computation time involved did not allow for a complete convergence of the strategies to their NE. The lack of convergence is also because the strategies are superficially clipped to meet the constraints rather than using the truncated normal distributions, as discussed in Section 4.4. This was done so that the strategies could be easily used to initialize the competitive coevolution optimization process (results presented in the following Chapter). It was not discovered until after the fact that this could bring about difficulties with the convergence. To inspect the quality of the convergence, the loss (normalized by the largest magnitude value) through all updates for the sensor value function, spacecraft policy, and sensor policy can be seen in Figure 5.10 .

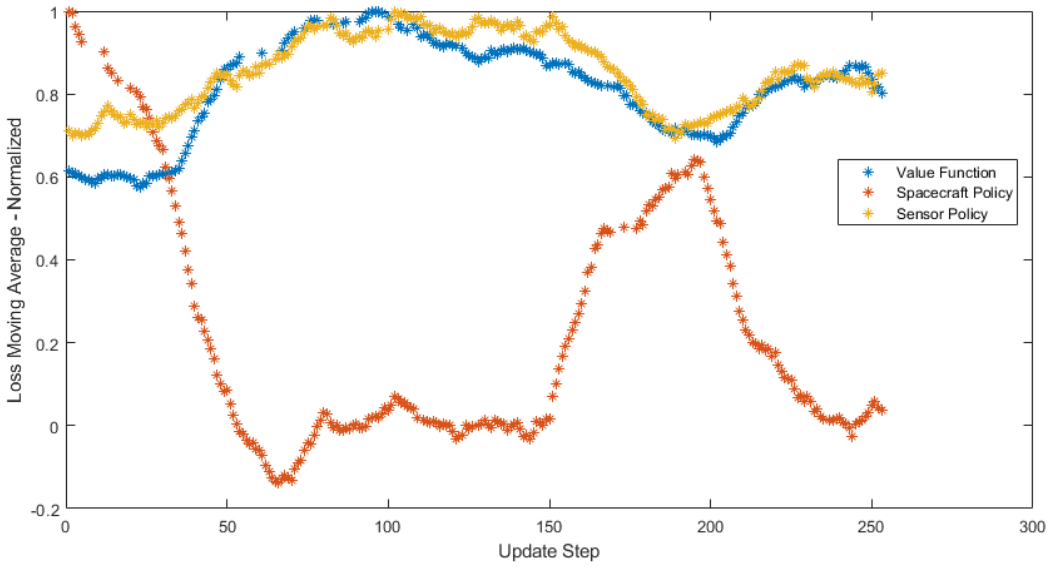


Figure 5.10. Moving Average of the Loss Values Through $100,000$ Iterations

As expected, the spacecraft and sensor policy losses follow opposite patterns - as the sensor policy loss decreases, the spacecraft policy loss increases. With this, the value function loss follows a similar pattern to that of the sensor loss, just with the changes slightly delayed. Inspecting the variation in the plots, it is obvious that the loss values have not converged. In fact, the current policies will likely favor the spacecraft over the sensor since the spacecraft loss most recently decreased. It is likely that the two policies will continue to drive each other to increase and decrease in succession until both

players reach steady state loss values after another likely 500,000 iterations or more (if at all given the superficial clipping implemented). However, the resulting strategies after 100,000 iterations can still shed light on this unique space-based game and the inherent limitations and advantages of both players.

From the resulting NE, the strategies are sampled for one episode to produce a single set of actions for both players. A simulation of these actions is shown in Figure 5.11. In this plot, the spacecraft can be seen in black while the sensor is green at observations where it successfully found the spacecraft and red at observations where it did not. In applying these two strategies, the sensor was determined to have successfully found the spacecraft 68.8% of the observations taken.

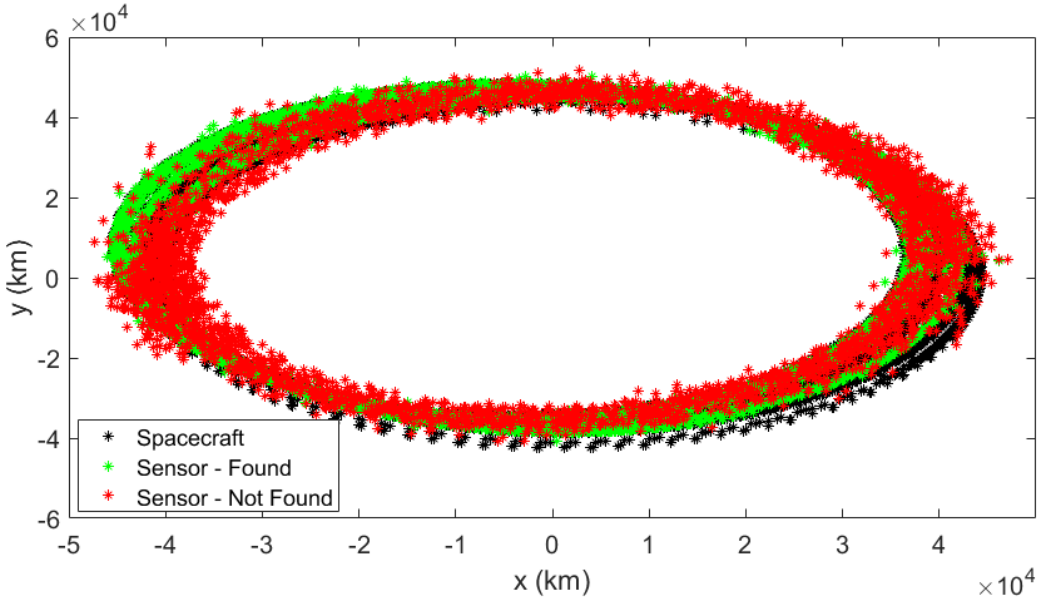


Figure 5.11. Spacecraft and Sensor States Through One Simulation

Though not necessarily optimal, the fact that this is likely close to the optimal result is encouraging. Successfully evading the sensor for even 31.2% of the observations taken is no small feat. The question is, though, how expensive is it for the spacecraft to be this successful? With the maximum ΔV per maneuver bounded at 20 m/s in each direction (28.28 in magnitude), the total for all 30 maneuvers could not possibly exceed approximately 848 m/s over the maximum 180 day simulation. In addition, the time between maneuvers could not exceed 6 days. Figures 5.12 and 5.13 show the ΔV values and times between maneuvers, respectively.

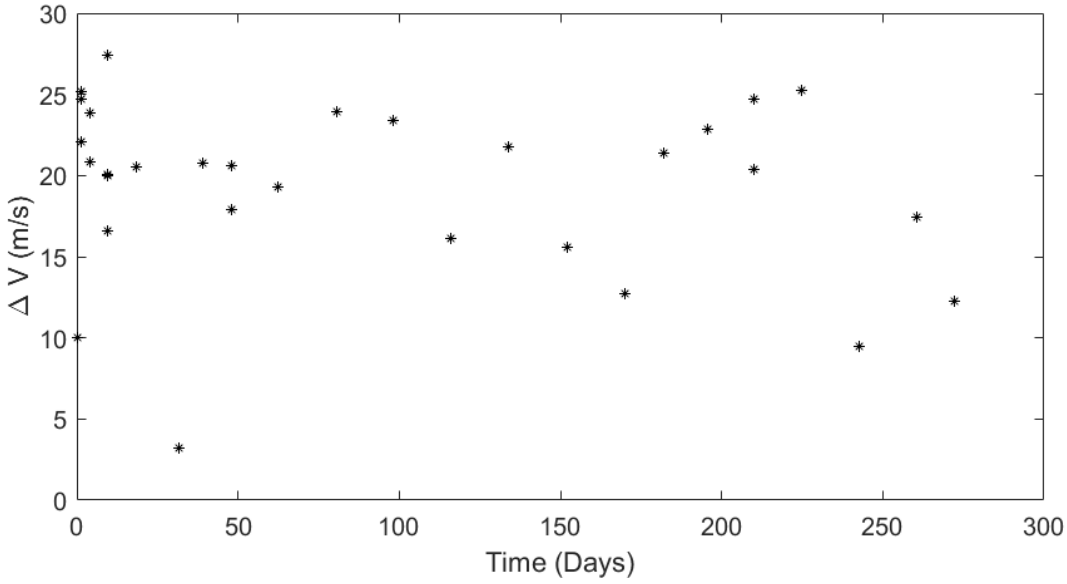


Figure 5.12. ΔV Applied for Each Maneuver

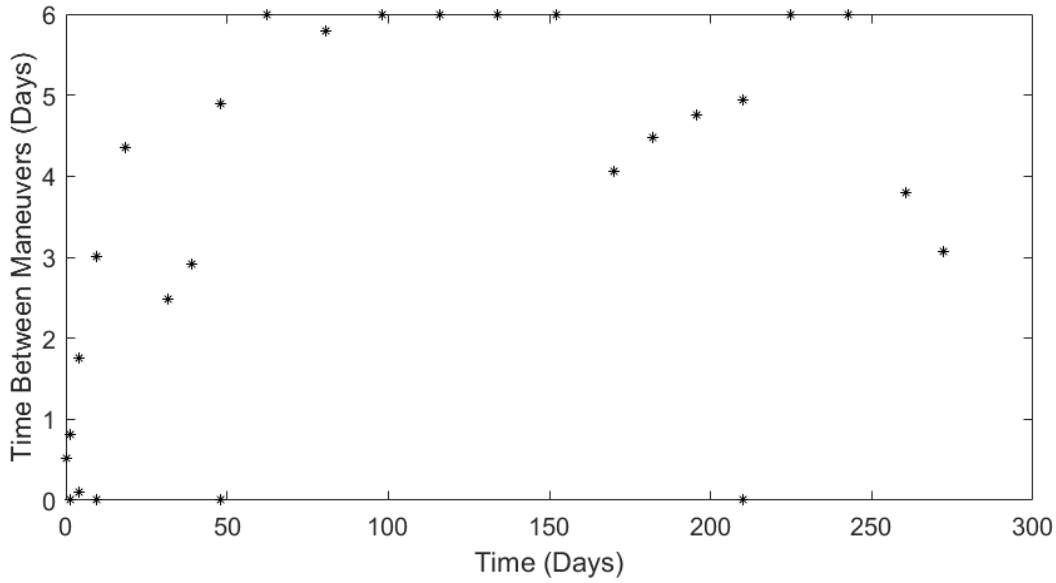


Figure 5.13. Time Between Each Maneuver

As expected, the ΔV values were close to the maximum allowed in order to achieve this result since the objective function selected was a function solely of the distance from the spacecraft to the center of the sensor's FOV. In the end, it took 580 m/s and an average of 3.12 days between each maneuver to attempt to win the game with this condition. It can also be noticed that a pattern is forming in these plots. Though 30

data points are not conclusive, it appears that the most effective ΔV values for sensor avoidance average out to about 19 m/s. Interestingly, after the first 10 maneuvers, the values drop from about 25 m/s to below 20 m/s and continue to decrease and vary further. This is to be expected since the sensor has a much more difficult time regaining custody of the spacecraft once it is initially lost. In addition, the time between maneuver values were not minimized to near-zero as one might expect when attempting to avoid detection by a sensor. This could be because of the 20 hours each day when the spacecraft is free to maneuver or coast without fear of being observed. Therefore, the spacecraft would optimize its objective by maneuvering between observation periods. Like seen with the ΔV values, the time between maneuver values start out at the extrema at the beginning of the simulation. These large, successive maneuvers ensure that the spacecraft can quickly gain separation from the tasked sensor. As the spacecraft becomes successful in doing so, the time between maneuvers starts to increase towards the upper bounds as the simulation progresses and less proximity between maneuvers is required to be successful in avoiding the sensor.

When conducting further analysis on the time between maneuver values, it was determined that the most likely cause for this is actually the fact that, as incorporated in other PPO applications, the reward provided to both players at the end of each episode is the sum of each individual reward rather than the average over all values. This means that the reward is also affected by the resulting simulation time, which is not at all desired. Using the average reward over each episode instead should encourage the optimal time between maneuver values to decrease. This is implemented in the next iteration of results.

Additional analysis can be performed by inspecting the ΔV magnitudes along its $\hat{x}$ and $\hat{y}$ components, shown in Figure 5.14. The opposite signs in the magnitudes here suggest that the optimal thrust angle for sensor avoidance is around 0° consistently. Looking closely at the angles of each maneuver, as seen in Figure 5.15, this can be confirmed to be true. Though the average maneuver angle falls around $-17°$, the values seem to follow a similar pattern to that of the ΔV and time between maneuvers, changing from around $-45°$ maneuvers to 0° as the maneuver magnitudes decrease.

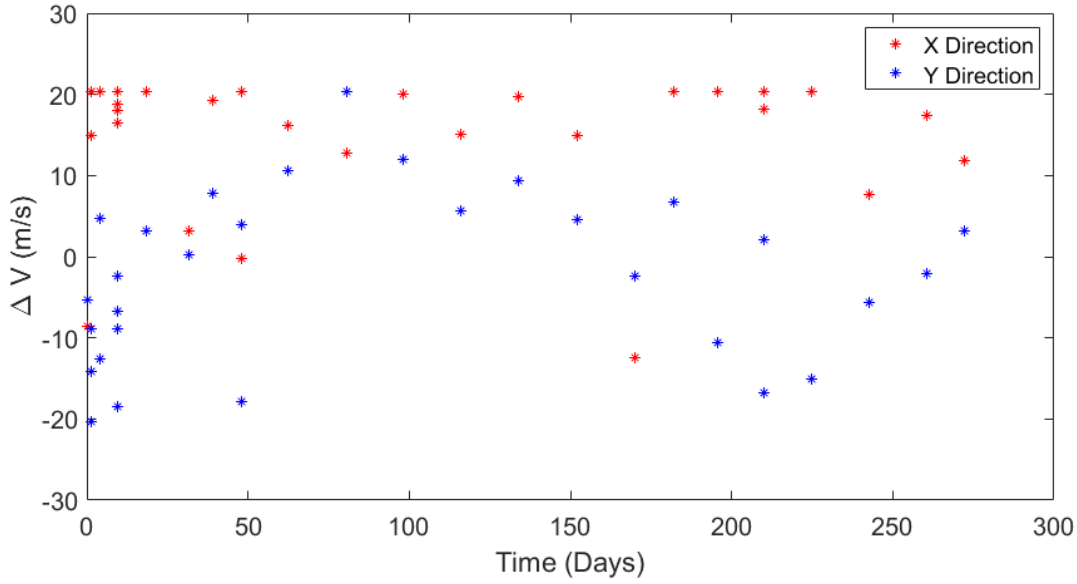


Figure 5.14. ΔV Components

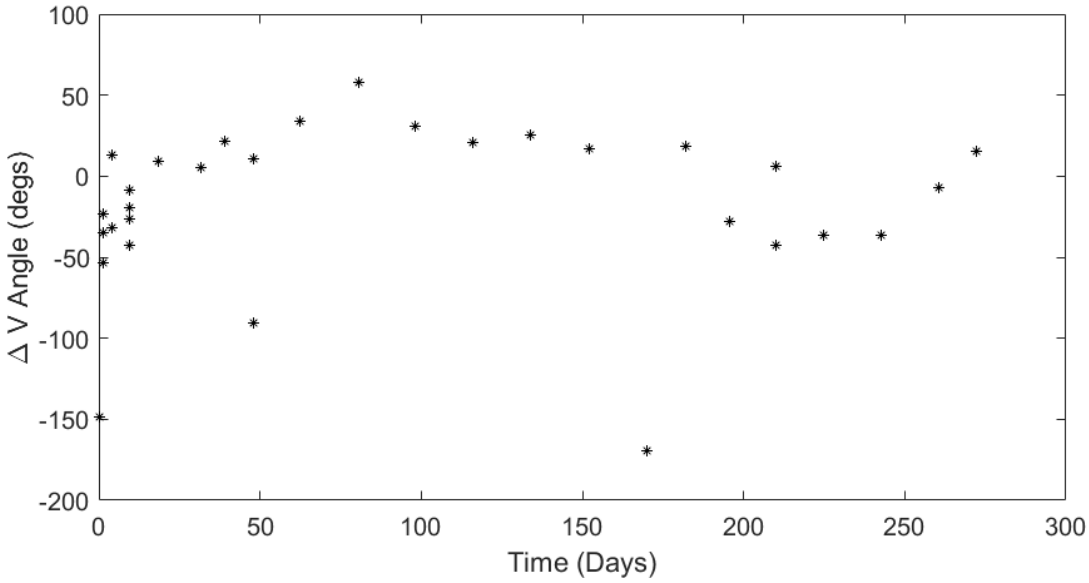


Figure 5.15. ΔV Angle

It's also useful to examine the pattern by which the sensor successfully finds the spacecraft. Looking closely at Figure 5.16, the spacecraft quickly evades detection, as is expected given the large, successive maneuvers at the beginning of the simulation. It takes almost 15 days for the sensor to regain custody of the spacecraft. Over the next 64 days, though, the sensor successfully finds the spacecraft at least once every

5 days. From there, another 66 days pass where the spacecraft goes undetected. The sensor then regains custody and continues to successfully observe the spacecraft without significant gaps for the remainder of the simulation. Zooming in on days 0 to 50, as shown in Figure 5.17, it can be seen that, even when no weeks-long gaps in observations exist, multiple observations periods can pass in a row with no successful observations. With days, and sometimes even months without the sensor regaining custody, this gives the spacecraft plenty of confidence in its evasive capabilities.

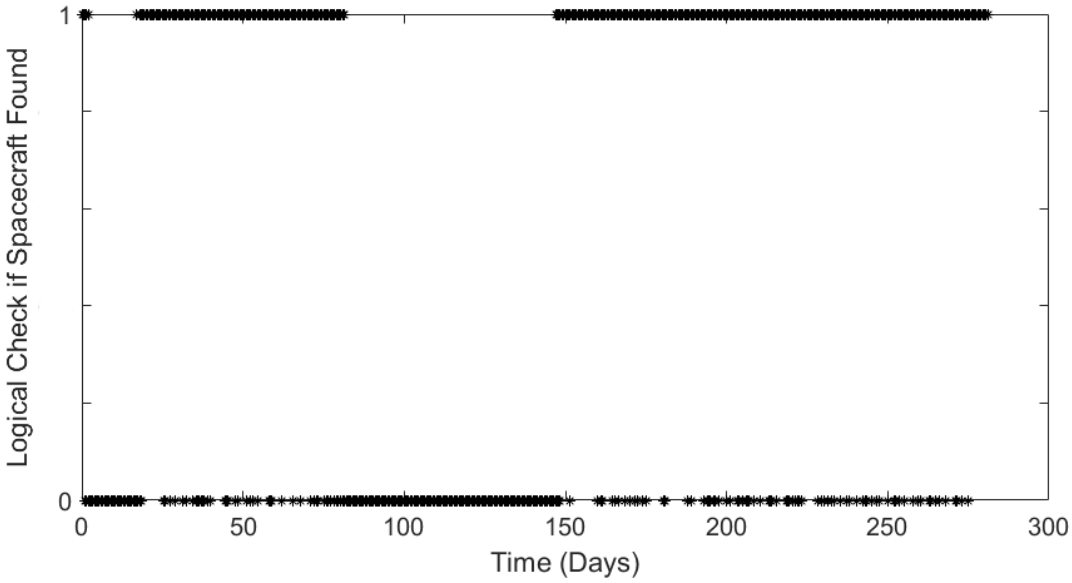


Figure 5.16. Time History of Succesful and Unsuccesful Observations

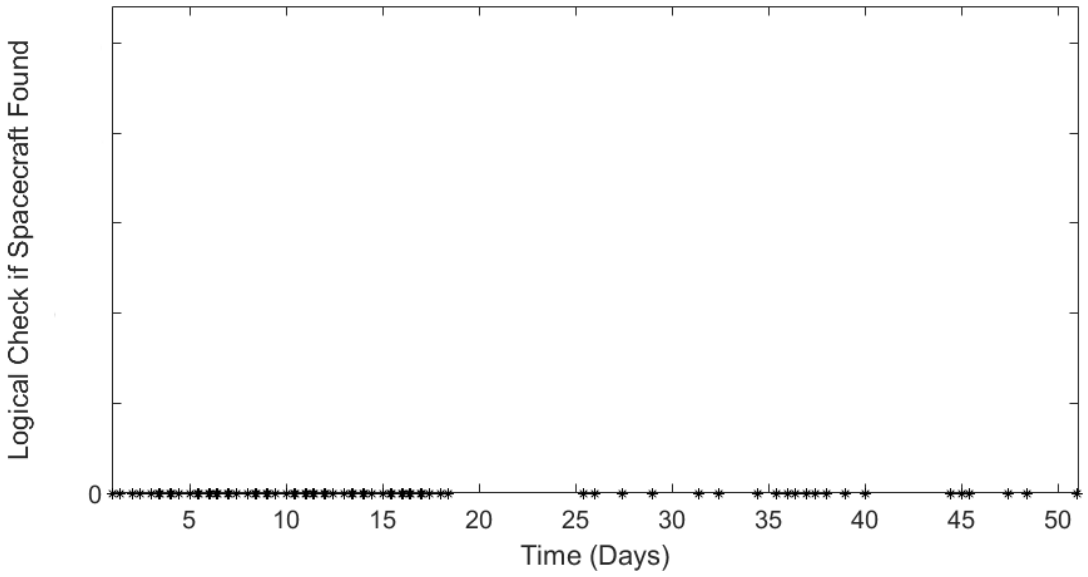


Figure 5.17. Time History of Succesful and Unsuccessful Observations - Days 0 to 50

5.3 Final Results of Two-Player Optimization

5.3.1 Problem Description

The final iteration of results primarily implements changes based on lessons learned from the first two iterations. To begin with, the observation window is expanded to account for modern optical sensing capabilities. For simplicity, the window is set at 12-hours per day. In addition, some of the contributions discussed in Section 4.4 are implemented. Most importantly, the truncated normal distribution is implemented to ensure that the entirety of the policy distribution is represented in the selected actions. This helps to improve the convergence of the solution. Additionally, as discussed in the previous section, the reward provided at the end of each episode is changed to be the average of all instantaneous rewards rather than the sum. This ensures that the spacecraft is not favoring longer episode lengths to increase its payout (and the opposite for the sensor). These changes make for a much more realistic solution which is able to fully converge to the optimal strategies for both players.

For this iteration of results, the spacecraft is prescribed a maximum ΔV of 20 m/s in each direction. An infinite time-horizon is assumed, but the time between each maneuver is bounded between 12 minutes and 10 days, increased from the last iteration because the previous 6-day upper bound was assumed to be too low since it was commonly selected in

the strategy. With the number of maneuvers capped at 30, this places an upper limit on the time-horizon at 300 days. These actions are again bounded between 0 and $4,216$ km in the $\hat{x}$ and $\hat{y}$ directions. In order to help ease some of the computational burden, the time step is doubled from 6 minutes to 12 minutes compared to the previous iterations.

If the spacecraft is successfully found in the sensor's FOV for 15 consecutive observations, custody is regained and the sensor can begin to follow the spacecraft along it's new path without applying the search strategy. In an attempt to simulate the nuances of state estimation, the sensor is afforded some flexibility in the search process. If the spacecraft has been successfully found for 7 consecutive observations after custody is lost, a missed observation does not restart the count unless the spacecraft is not located for 3 observations in a row, in which case the count to 15 must restart in order to regain custody. These values were changed from the previous iterations to attempt to better model real-world limitations of the sensor. As discussed in Section 3.1, Gooding's algorithm requires two "passes", or 38 observations, to properly initialize the BLS process. To replicate this, 15 observations is subjectively set as the minimum for an initialization of the filtering process. Since the time between maneuvers is significantly higher in this analysis, 15 was selected as an approximation of the same statistical significance. It is assumed that 7 observations, or about 84 minutes of observations, is sufficient for the filter to obtain a low-uncertainty estimate of the state assuming no maneuvers are performed in that time. In addition, it was determined in Section 3.1 that it takes about 30 minutes for the filter's state error to exceed the 3σ uncertainty after a maneuver, or just over 3 observations. Hence the change in the implementation.

The hyper-parameters used for this optimization can be found in Table 5.3.

Table 5.3. Hyperparameters Used in PPO Training

Input	**Value**
Discount Factor, γ	0.85
Batch Size	3200
Reward Steepness, λ	0.98
KL Divergence Target, D_{targ}	0.0003

5.3.2 Results

Now that a solution has been obtained that was able to nearly fully converge to a NE, a proper analysis can be conducted on the resulting strategies and their implications.

First, to verify convergence, the loss values for the spacecraft's and sensor's policies can be seen plotted together in Figure 5.18, normalized by their highest magnitude values. As expected, both are converging to steady state values, but are slightly offset from zero which is not ideal. This can be explained by the inherent unpredictability of the actions of the two players. The non-zero loss suggests that, while the strategies have converged, the confidence that the strategies are globally optimal is low. However, the presence of convergence at the very least allows for further analysis of the resulting strategies.

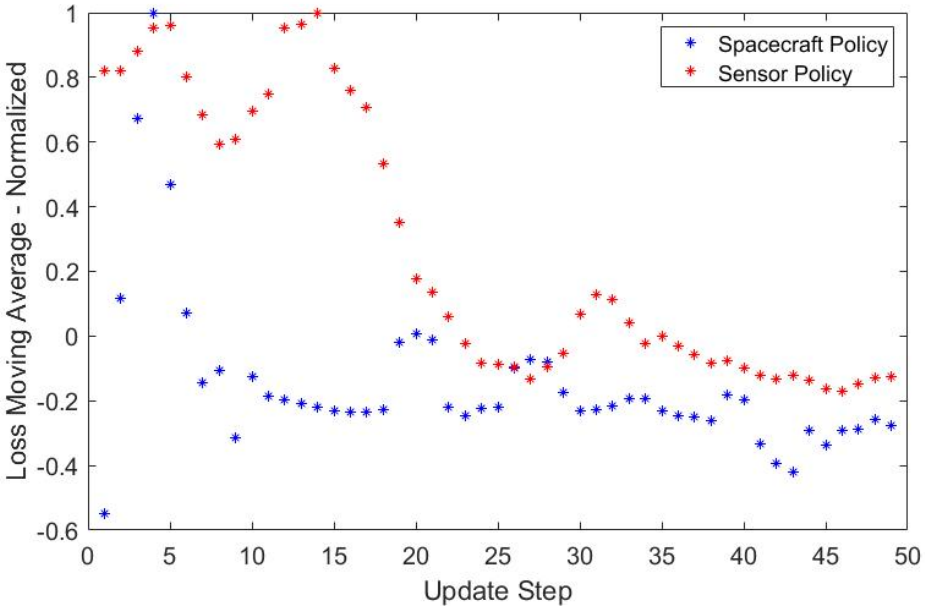


Figure 5.18. Loss Values for Spacecraft and Sensor Policies

From the resulting NE, the strategies are again sampled for one episode to produce a single set of actions for both players. As was presented in the previous two sections, the resulting ΔV and time between maneuver values are a good place to start for a proper analysis. Figure 5.19 shows the ΔV components in the ECI $\hat{x}$ and $\hat{y}$ axes and Figure 5.20 shows the magnitude of the maneuvers. Interestingly, very little pattern appears to emerge here other than that the components of the maneuvers appear to favor the positive direction and that the largest maneuvers are performed near the beginning of the episode. The lack of a distinct pattern is likely because the spacecraft has succeeded in randomizing its maneuvers such that it cannot be easily predicted by the sensor. It is noteworthy, though, that the ΔV magnitudes rarely approach the maximum allowed. This means that the maximum ΔV could be constrained further without a significant loss in capability. Altogether, the spacecraft expends a total of 474 m/s of ΔV over the

course of about 160 days in its attempt to evade the sensor, an average of 15.8 m/s per maneuver.

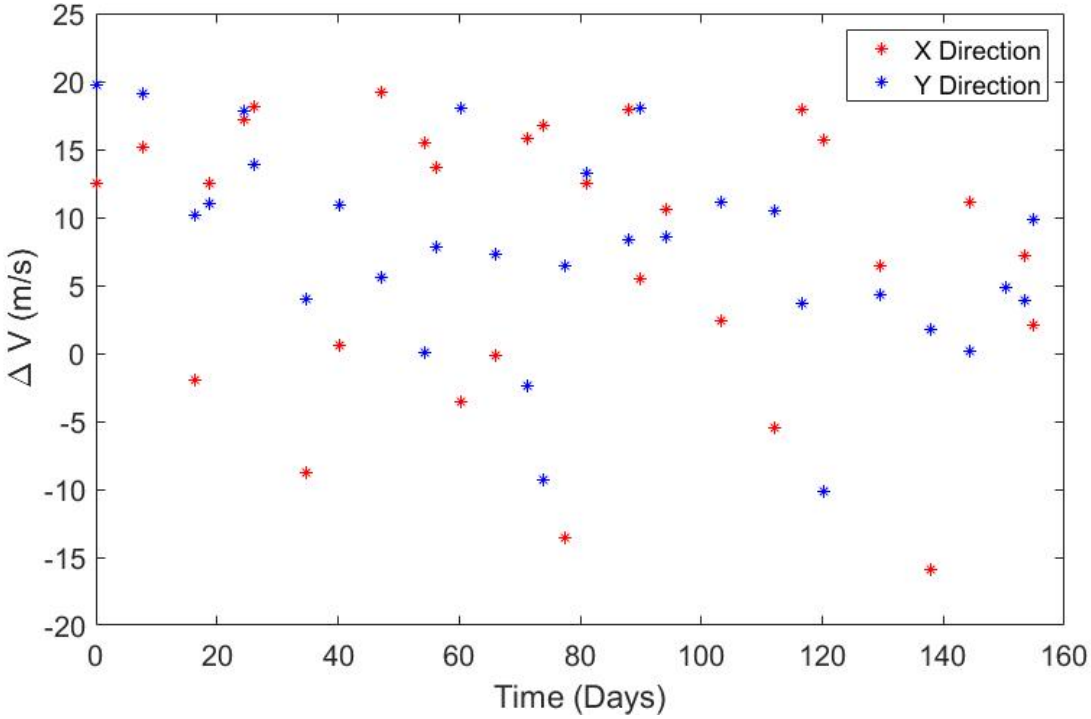


Figure 5.19. ΔV Components

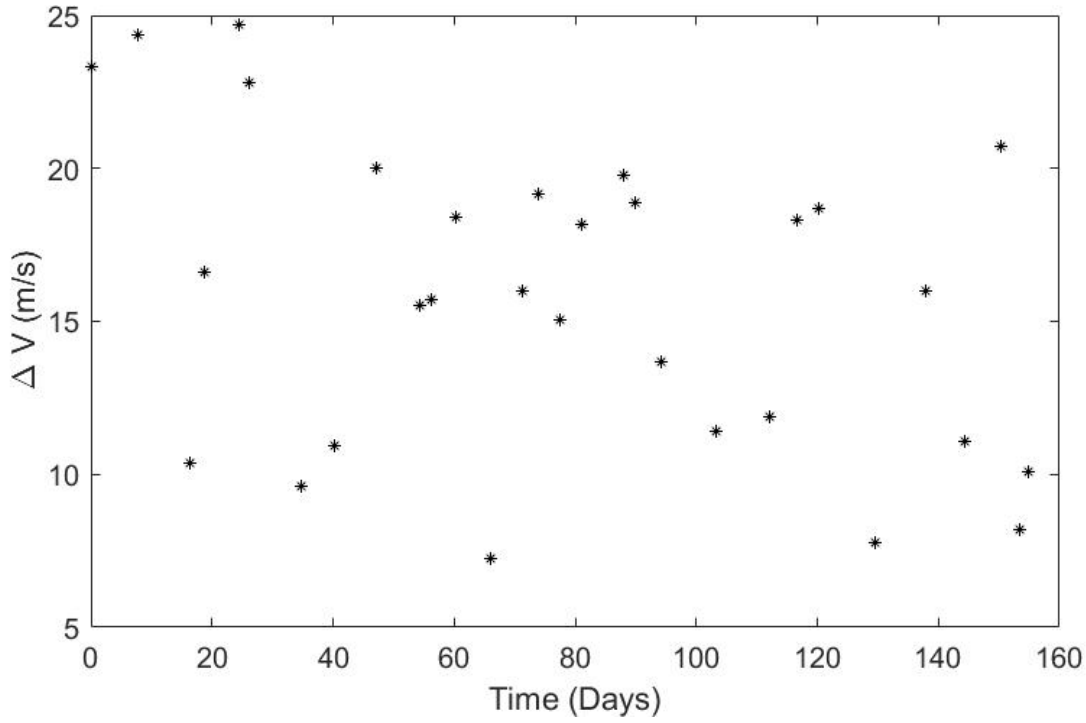


Figure 5.20. ΔV Magnitudes

The time between maneuver values can be seen in Figure 5.21. Unlike in the previous results, the values do not approach the upper bound as much. This is because of the change in the reward implementation such that the value provided at the end of each episode is the average of the individual rewards as opposed to the total. Again, there appears to be no obvious pattern in the data, even less so than the ΔV plot, though 30 points may not be enough to identify one. The large variation in the times between the maneuvers suggests that the spacecraft player found unpredictability to be more important than placing maneuvers consistently close together or far apart, and that even large successive maneuvers would not avoid detection for long enough to make the large propellant expenditure worth the cost. Over the entire episode, the maneuvers are performed on average every 5.3 days.

It is noteworthy too when the maneuvers are performed. Looking at Figure 5.22 the same plot is produced but with each maneuver colored according to the lighting condition at the time. Though it would be expected for each maneuver to be performed when the spacecraft is not visible, this is not the case. This implies that the spacecraft is indifferent to the lighting condition when the maneuver is performed, likely because the maneuvers are large enough in magnitude that locating it immediately after a maneuver

is performed is nearly impossible no matter its visibility.

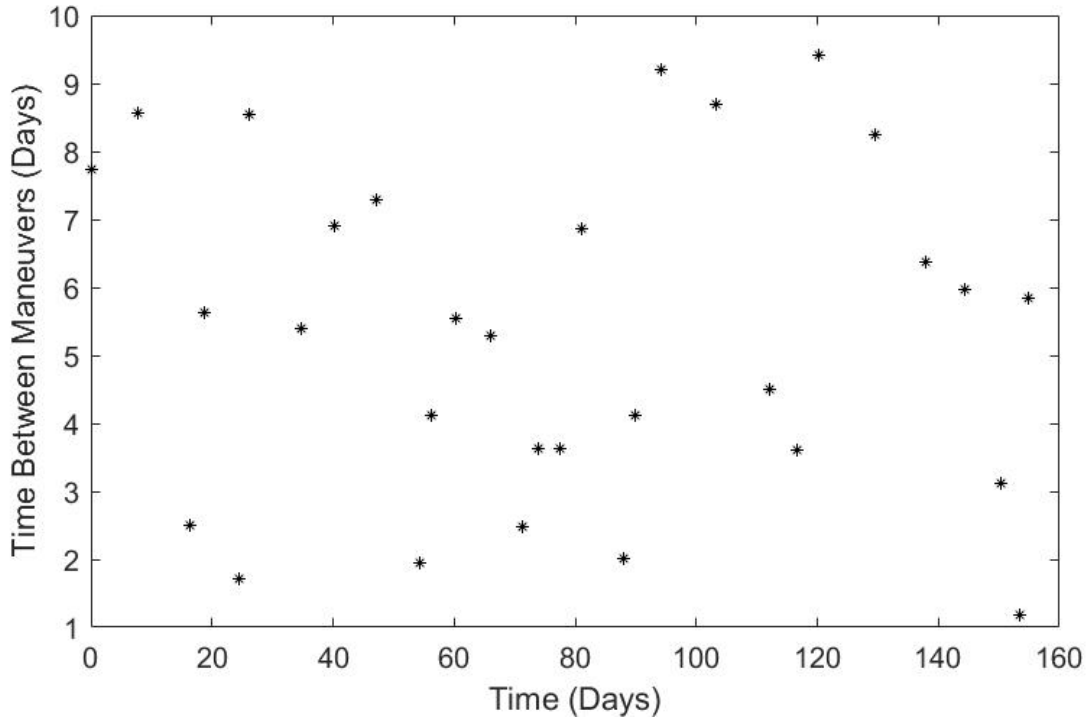


Figure 5.21. Time Between Each Maneuver

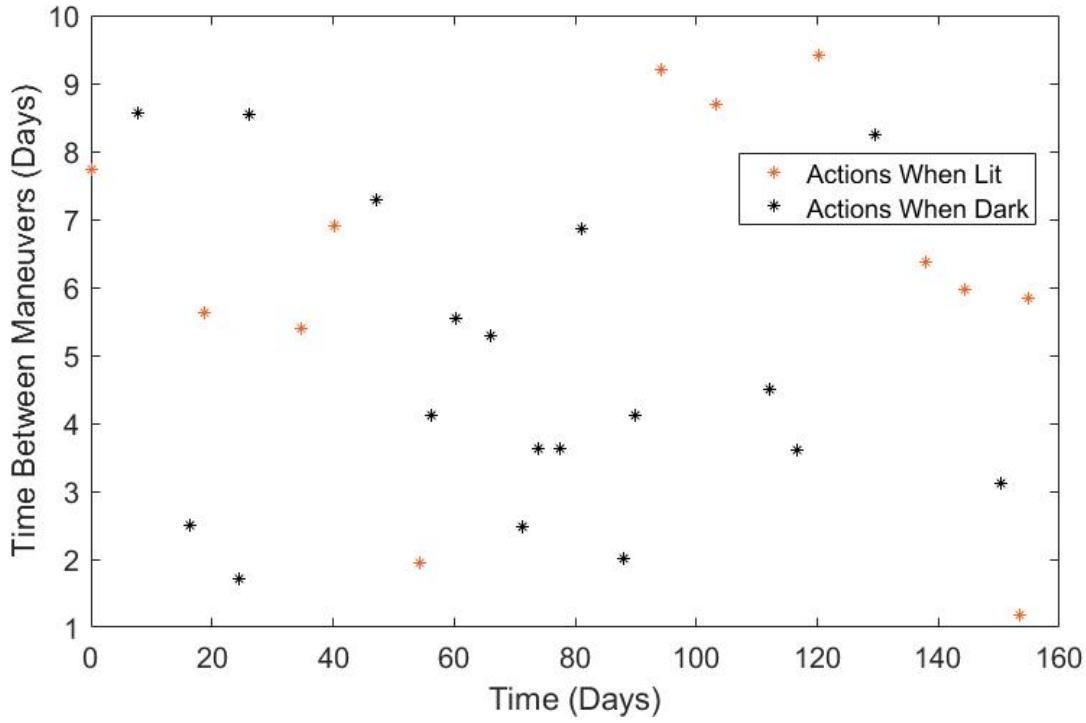


Figure 5.22. Time Between Each Maneuver - With Lighting Condition

It is also useful to inspect the direction of each of the maneuvers. Figure 5.15 shows the angular values between the ΔV vectors and the horizontal in the ECI frame. Unlike in the previous iterations of results, an optimal angle cannot be easily determined, at least not by visual inspection. The only clear interpretation is that maneuvers applied at positive angles are better for sensor avoidance than those applied at negative angles. More analysis would be required to determine why this is the case, though. However, the average angle was actually found to be close to 45°, same as the optimal angle found in the previous results. So, though the strategy prefers unpredictability over a specific pattern, the average is still meaningful.

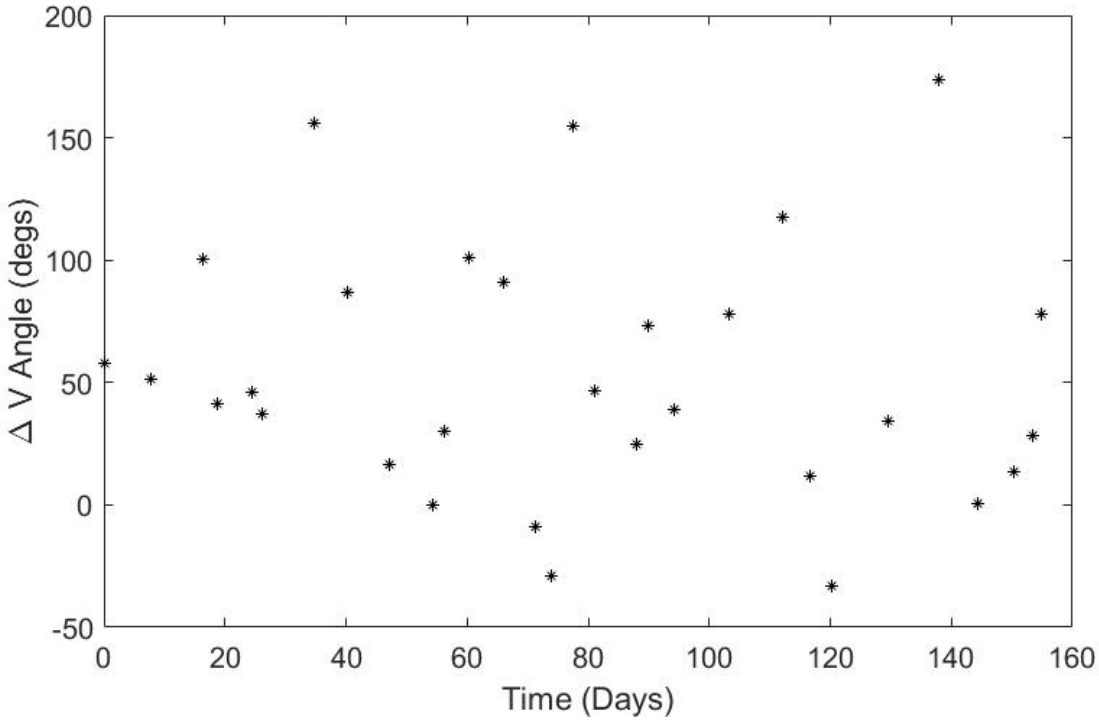


Figure 5.23. ΔV Angle

One data set that was not discussed in the previous sections is the sensor tasking pattern. While attempting to track the satellite, the sensor's position with respect to the last known orbit of the satellite, in the $\hat{x}$ and $\hat{y}$ directions, can be seen in Figure 5.24 for one observation period, with the magnitude values of the change presented in Figure 5.25. Given the seemingly random strategy employed by the spacecraft, it is not surprising that the sensor's strategy similarly follows no obvious pattern as well. The only noteworthy observation is that the components rarely fall between -0.5 and -1.0. It is noteworthy, though, that it explores the entire space, including near both extremes. This means that the constraint on the sensor's position may be a major factor in it's ability to track the spacecraft.

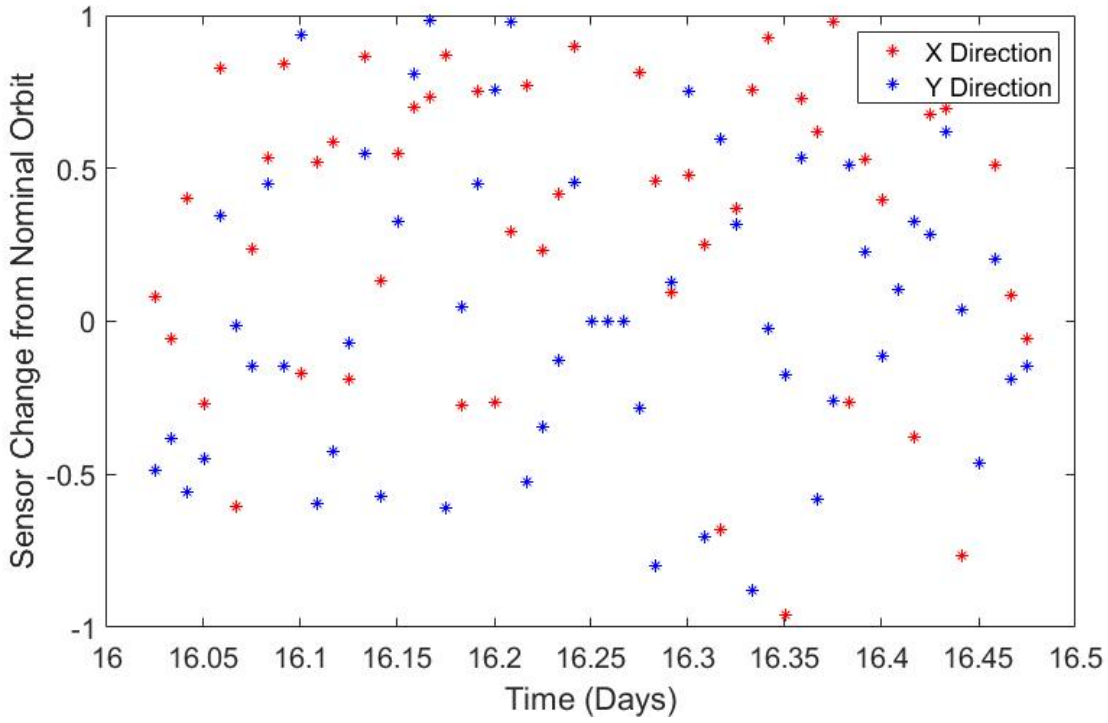


Figure 5.24. Sensor Distance From the Nominal Spacecraft Orbit for One Observation Period - In Components

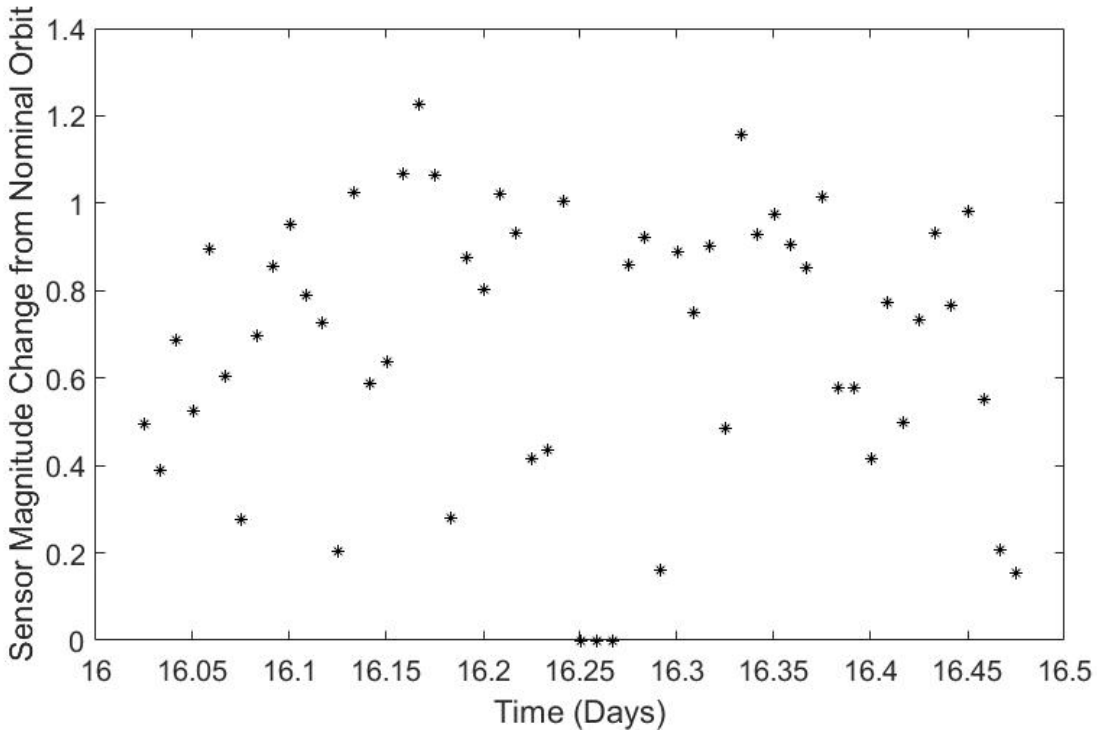


Figure 5.25. Sensor Distance Magnitude From the Nominal Spacecraft Orbit for One Observation Period

Implementing the strategies of the two players together, a single simulation of the game is performed. Figure 5.26 shows the states of the spacecraft and the sensor throughout one episdoe, in which the spacecraft can be seen in black, the sensor is shown in green at observations where it successfully found the spacecraft, and the sensor is shown in red at observations where it did not. Though, a vast majority of the points are red, note that this is somewhat misleading since it overlaps a large number of green and black points. Still, the spacecraft was successful at evading detection more than half of the time. In applying these two strategies, the sensor was determined to have successfully found the spacecraft 41.4% of the observations taken.

From Figure 5.26, it is not obvious whether a pattern develops in the timing of when the spacecraft is and is not successfully located by the sensor. Figure 5.27 shows the logical values for this analysis, where a value of 1 means that the spacecraft was successfully found. Inspecting this plot, it can be seen that the sensor successfully located the spacecraft for the majority of the 25 days but soon lost custody for over 90 days. Custody was then regained for most of the remaining 45 days. It would take a longer simulation to determine if a pattern has formed, but it does appear that when

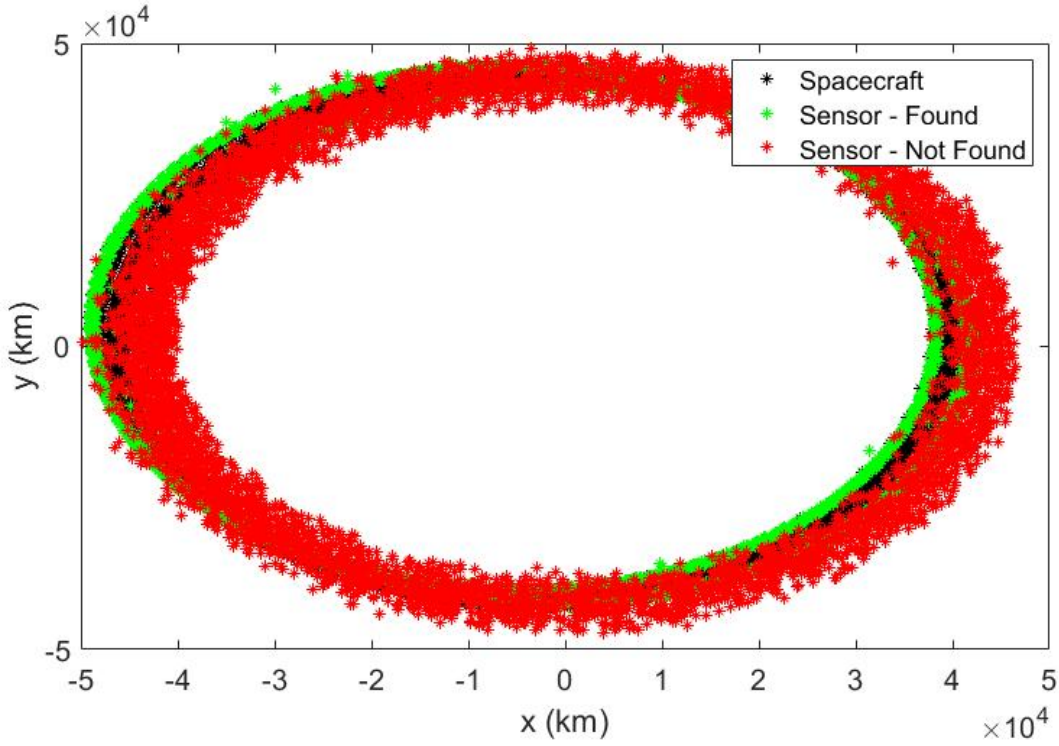


Figure 5.26. Spacecraft and Sensor States Through One Simulation

the spacecraft avoids detection by the sensor it does so for long periods of time. It's also interesting to note that, even when the sensor manages to maintain custody of the spacecraft, the longest time period of uninterrupted observations is just 9 days. Additionally, whether a correlation exists between the timing and size of the spacecraft's maneuvers and its success at evading detection should be investigated. Figure 5.28 shows the logical values for whether the spacecraft is found overlaid with the maneuver magnitudes on the same time scale. Interestingly, it does appear that there is a slight correlation. It is only after a series of five up to 25 m/s maneuvers is performed that the spacecraft starts to fully avoid detection. It then requires less than 20 m/s per maneuver to maintain this custody avoidance for the 90 day period. This suggests that at least some maneuvers above 20 m/s would be required for the spacecraft to fully prevent custody though additional analysis would be required to verify this.

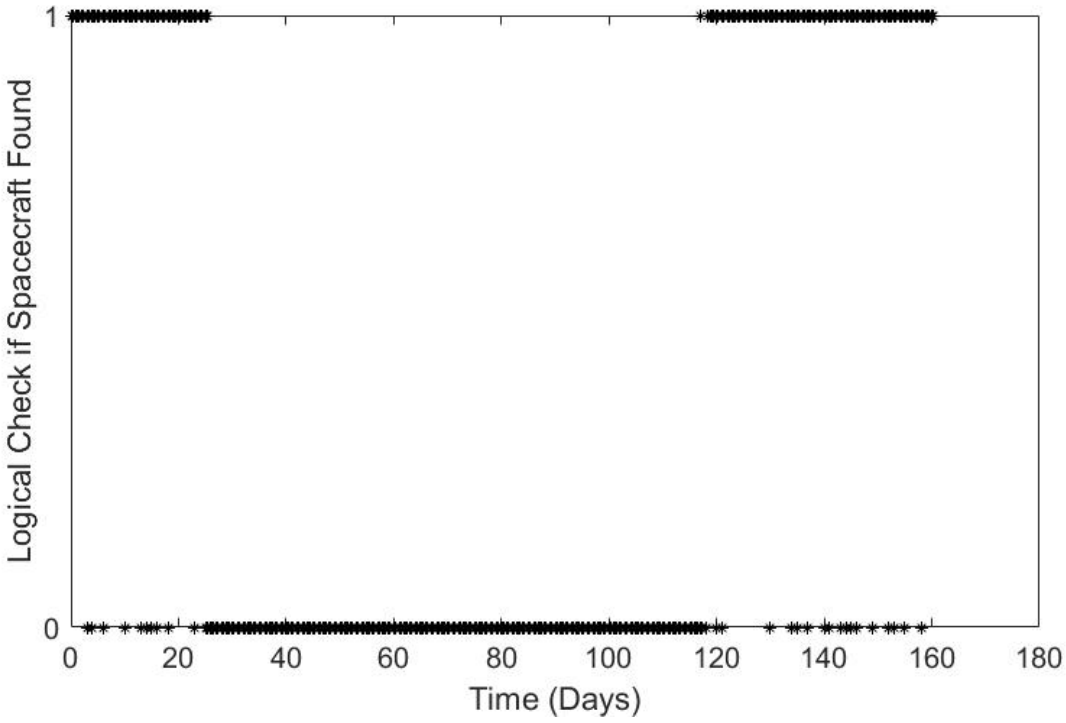


Figure 5.27. Time History of Succesful and Unsuccessful Observations

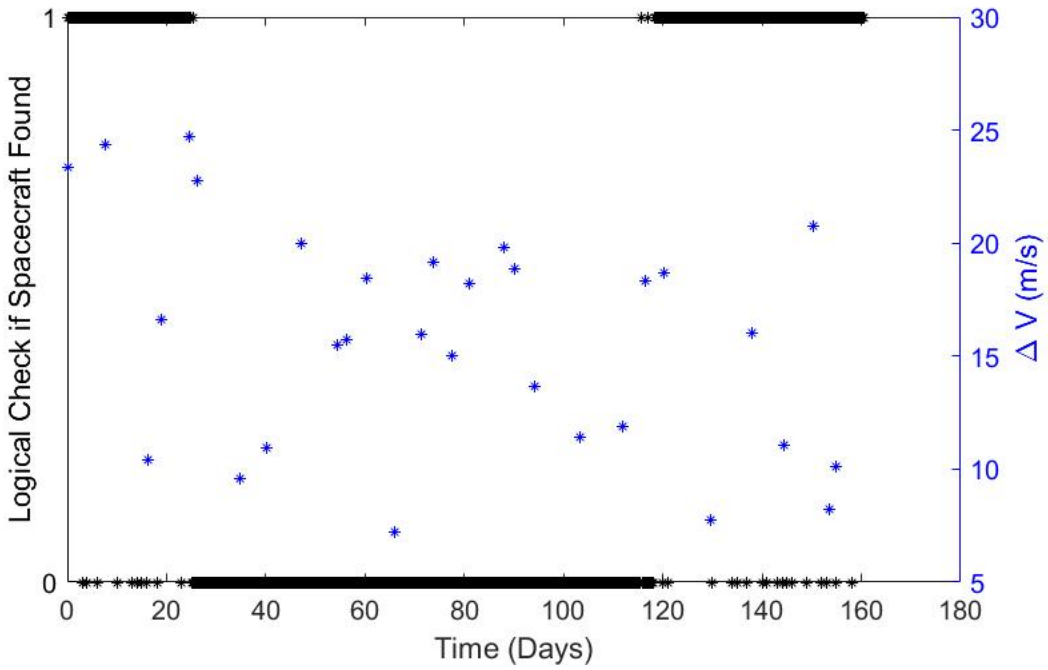


Figure 5.28. Time History of Observations w/ Maneuvers Performed

Overall, it was determined that the spacecraft could expend 15.8 m/s of ΔV every 5.3 days on average to successfully evade the sensor for 58.6% of the simulation, or 94 out of the total 160 days. This resulted in up to 90 days where the spacecraft went completely undetected. Though this is a rather significant amount of fuel, it is worthwhile to note that the objective function used does not take ΔV into account at all, so this is not the most ΔV-conscious solution possible. In addition, evading the sensor for 58.6% of time is a rather lofty, and potentially unnecessary goal. Significantly less ΔV could be used and still obtain a reasonable amount of time outside of the sensor's FOV. Furthermore, evading detection by the sensor is much harder than evading custody of the sensor. Using the sensor's uncertainty in its estimate of the spacecraft's state, rather than the current distance objective, would lend itself well to a custody evasion analysis and would likely require much less propellant to accomplish.

Chapter 6
Sensitivity Analysis Using Competitive Coevolution

The competitive coevolution research was conducted alongside the policy optimization research. These sections mirror the applications of PPO in Sections 3.3, 5.2, and 5.3. First, the results are provided when competitive coevolution is applied to continuous thrust sensor avoidance trajectories as a proof-of-concept. Next, the same methodology is applied as an augmentation to Section 5.2 in which the strategies of both players are evolved to meet a defined win condition. Finally, the same scenario used in Section 5.3 is applied in competitive coevolution to demonstrate how the approach can be used to perform sensitivity analysis on the results.

6.1 Preliminary Results: Continuous Thrust Trajectories

6.1.1 Problem Description

In the devised toy problem first optimized in Section 3.3, the spacecraft is instructed to arrive at a specified state (position and velocity) after some fixed time. For this exercise, a 10% orbit raise is optimized with the arrival occurring in just under two-thirds of an orbital period, as in Figure 6.1 Here the spacecraft starts at the point $[1, 0]$ on the blue orbit and transfers to the red orbit by way of the black orbit, ending around the coordinate $[-1, -0.3]$. Note that non-dimensional units, normalized to the initial orbit radius, are used to improve the accuracy and speed of the optimization and that the maneuver is assumed to be planar.

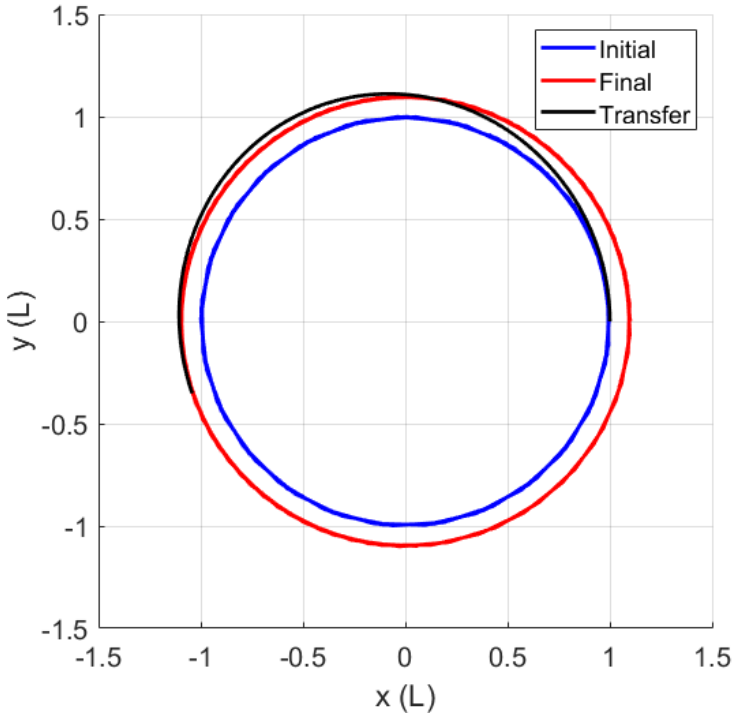


Figure 6.1. Arrival at a Specified Point After Some Fixed Time - Minimum Propellant

Though minimizing the control input (propellant usage) is important, the primary objective is to maximize the distance between the spacecraft and the sensor. The sensor itself is assumed to be optical with a defined FOV. By default, the sensor will follow along the trajectory that the spacecraft, propagated from its previous point with no additional input, is assumed to be following. If at any point the spacecraft is not found within the sensor's FOV, then the sensor will move in a sinusoidal pattern of gradually increasing amplitude along the nominal path until the spacecraft is found again (like presented in Sections 3.3 and 5.1).

The end result will be a trajectory that diverges slightly from the minimum-propellant trajectory (like in Figure 6.1) as the sensor continues to lose and regain custody of the spacecraft over the specified time interval.

In an effort to optimize the PE game, competitive coevolution is applied to continually adjust two inputs for each of the players. For the space player, the weights applied to the ΔV and d, the distance between the spacecraft and the sensor, in the objective function, are adjusted between runs. For the ground player, the frequency and amplitude of the sinusoidal search pattern are adjusted between runs. The result is a NE of sets of these four inputs in which neither player can unilaterally change their inputs to increase their

payoff (fitness).

6.1.2 Results

As discussed previously, competitive fitness sharing, shared sampling, and Hall of Fame are all valid strategies for ensuring global optimality in competitive coevolution. However, given the inherent continuity in the selected search space, competitive fitness sharing was found to be unnecessary. Poorly represented types was not an issue with this specific game as parasites were never found that could only be defeated by specific hosts. Similarly, shared sampling was also found to be an unnecessary burden on the optimization. With a relatively small number of iterations required to reach convergence, the Halls of Fame for the two players remained rather small. Instead of speeding up convergence, sampling from the opponents Hall of Fame each iteration greatly slowed down the convergence since less optimal strategies were being added to the current player's Hall of Fame. However, applying the Hall of Fame technique alone was found to be more than sufficient.

Due to the computation time required to solve each iteration, the maximum number of generations and coevolutions (applied in Algorithm 2) were set to three and five, respectively. It was found that, in general, re-sampling the population between coevolutions was more effective at producing better fitness values than recombining and mutating the population between generations. In addition, it typically only took a coevolution or two to successfully sample a population with a better fitness prior to convergence being reached.

The fitness itself, a single value in a zero-sum game, was simply calculated as the fraction of the transfer time that the spacecraft was successfully located by the tasked sensor. The closer to 1, the better the result for the pursuing sensor. The closer to 0, the better the result for the evading spacecraft. For this exercise, the fitness threshold, ϕ, was chosen to be 0.5 so as not to induce any bias. If the strategies converge such that the sensor is successful in locating the spacecraft more often than it is not, the sensor wins. Otherwise, the spacecraft wins.

Figure 6.2 shows the progression of the fitness by iteration as the competitive coevolution was performed. The red points are when the sensor is finding a new strategy to beat the spacecraft's Hall of Fame, while the blue points are when the spacecraft is finding a new strategy to beat the sensor's Hall of Fame. After nearly 100 iterations, the strategies converged to a NE and the spacecraft is declared the winner. Though the spacecraft is usually considered to be the disadvantaged of the two in a PE game, this result is to be expected in this case considering that the spacecraft's trajectory is being

optimized at each time step, while the sensor is not allowed to adapt and has to follow a pre-defined search pattern when custody is lost. In addition, no constraints were placed on the spacecraft's propellant use, usually considered to be the greatest limiting factor in space-based games.

Looking closely at Figure 6.2, the competitive coevolution is performing as expected. At first both players are able to find strategies that dominate the other since the Halls of Fame are small and sub-optimal. However, as more capable strategies are added to the Halls of Fame, the players have a harder and harder time finding new strategies that can improve their payoff. As is expected, the payoff of one player eventually drops below the 0.5 fitness threshold and the evolution of the strategies begins. Eventually, a successful strategy is found and the fitness once again exceeds the threshold. This process repeats over and over again until the Halls of Fame are filled with the most capable strategies possible, the fitness converges to a steady state value, and a winner is declared.

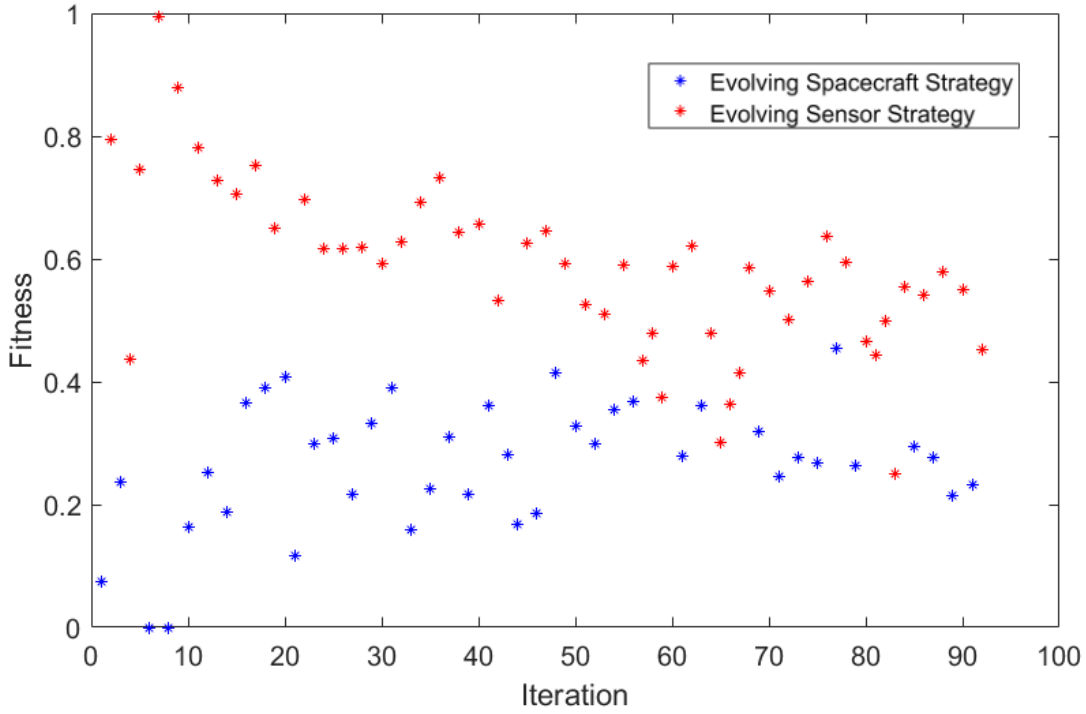


Figure 6.2. Fitness At Each Competitive Coevolution Iteration

Now, the converged solution can be compared with the pre-coevolution baseline. Figure 6.3 shows the resulting transfer when the converged strategies are applied. The location of the magenta points (when the spacecrft is not found) suggest that the sensor often has to search large areas of the space in order to find the spacecraft. Despite this, it

was still successful at relocating the spacecraft and maintaining custody for the latter part of the simulation (the cyan points). As expected given the converged fitness value, the spacecraft is found by the sensor less than 30 percent of the transfer. However, this was only accomplished by performing a propellant-cost-prohibitive maneuver. Constraining the propellant use of the spacecraft would likely result in a converged fitness value more in favor of the sensor.

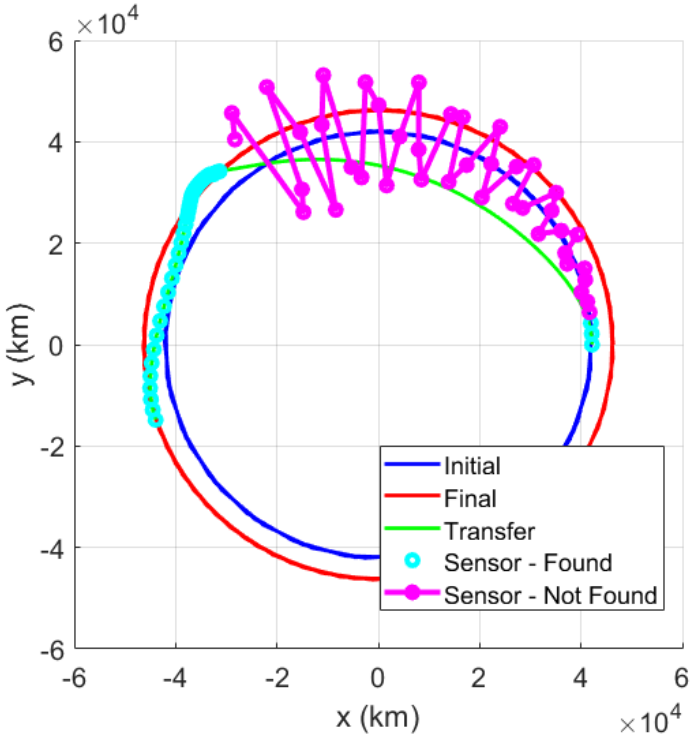


Figure 6.3. Optimal Sensor Avoidance Path at Convergence

6.2 Secondary Results: Initial Two-Player Optimization

6.2.1 Problem Description

The scenario solved in this section is the same as was presented in Section 5.2. The difference here is that, instead of optimizing the distance between the spacecraft and the center of the sensor, the competitive coevolution approach considers only the percentage of the time that the spacecraft is successfully found by the sensor (called the fitness) since it is easier to conceptualize and set a win condition.

Like in the PPO approach, the spacecraft is prescribed a maximum ΔV of 20 m/s in

each direction. An infinite time-horizon is assumed, but the time between each maneuver is bounded between 6 minutes and 6 days. With the number of maneuvers capped at 30, this places an upper limit on the time-horizon at 180 days. Each maneuver is applied instantaneously and is represented as a change in the velocity in the $\hat{x}$ and $\hat{y}$ directions. Meanwhile, the sensor collects observations every 6 minutes during the 2-hour windows after sunset and before sunrise and is assumed to have a FOV of 4°. Since there are no Keplerian dynamics to constrain the sensor's path, each action is performed with respect to the previous known orbital path of the spacecraft. These actions are bounded between 0 and $4,216$ km (10% of the initial orbital radius) in the $\hat{x}$ and $\hat{y}$ directions. Similar to the previous iteration where the tasking strategy is prescribed, the sensor is assumed to follow along the spacecraft's path until it is not found within the sensor's FOV. At this point, it is assumed that the sensor has lost custody of the spacecraft and searches the nearby space according to the learned strategy. If the spacecraft is successfully found in the sensor's FOV for 10 consecutive observations, custody is regained and the sensor can begin to follow the spacecraft along it's new path without applying the search strategy. In an attempt to simulate the nuances of state estimation, the sensor is afforded some flexibility in the search process. If the spacecraft has been successfully found for 5 consecutive observations after custody is lost, a missed observation does not restart the count unless the spacecraft is not located for 3 observations in a row, in which case the count to 10 must restart in order to regain custody.

Since the state/action spaces and constraints are highly subjective, competitive coevolution is then applied to investigate how the strategies can be adapted, free of bounds and constraints, to achieve specific goals (the percentage of successful observations). For this analysis, the population size was set to 10, the maximum number of generations was set to three, the maximum number of coevolutions was set to 10, and the maximum Hall of Fame size was set to 50. Though these are arbitrary and slight adjustments could result in smoother convergence, these values appear to be acceptable.

As discussed in the previous section, it was again found that applying the Hall of Fame technique alone was found to be more than sufficient. With a win condition set based on the percentage of observations that successfully find the spacecraft, the Hall of Fame approach collects a mixture of the most uniquely successful strategies of both players and evolves this mixture free of constraints until one of the players can no longer evolve its mixed strategy in such a way that allows it to meet the win condition. Thus, a winner, and both player's optimal strategies, can be found for the specified win condition.

6.2.2 Results

Ideally, the results in Section 5.2 would be used to initialize the strategies to be evolved here. Unfortunately, due to the computation time required to run the RL process to completion, a truly optimal NE of strategies was not available at the time to use as an input to the competitive coevolution algorithm. Instead, non-optimal strategies were collected towards the beginning of the PPO simulation. Though not ideal, competitive coevolution was found to be robust enough to still optimize these strategies for a given win condition, all be it in greater computation time.

For this demonstration, a win condition of 0.5 was selected. That is, in order to be deemed the winner, the spacecraft must successfully evade detection by the sensor for greater than 50% of the observations taken. Conversely, the sensor must successfully find the spacecraft within its FOV greater than 50% of the observations to be considered the winner. The resulting plot of the fitness values can be seen in Figure 6.4.

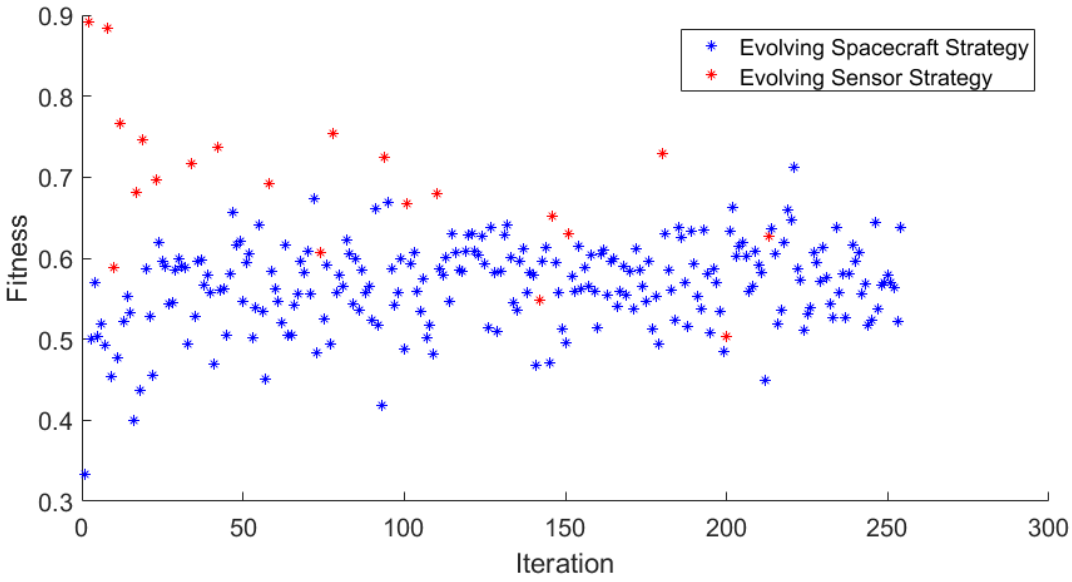


Figure 6.4. Fitness Value Throughout The Coevolution Process

Here, the red points are when the sensor's strategy is coevolving while the blue points are when the spacecraft's strategy is coevolving. Since the initialized strategies are no where near optimal, it was easy for the first round of coevolution to find significantly superior strategies. This is why the first two points are so close to the extrema. The strategies in the set of 10 populations that led to these fitness values are added to their respective Halls of Fame and the process continues. It can be seen early in the evolution

that the spacecraft is having difficulty achieving a fitness of less than 0.5. By the 20^{th} iteration, it takes three generations each with three coevolutions for the spacecraft to find a strategy that can beat all of the sensor's six Hall of Fame strategies. As the sensor adds more strategies to its Hall of Fame, it becomes increasingly harder for the spacecraft to find a winning strategy. After about 250 iterations, the spacecraft can no longer find a winning strategy and the sensor is declared the winner. Though the sensor's fitness values were driven below 0.6 twice, it never once fell below the 0.5 threshold. Given the inherent imbalance in the sensor avoidance game, this is not at all surprising. More so, this does not mean that the sensor could not further improve upon its own strategy and best the spacecraft by an even larger margin. Still, a lot can be learned from this.

Applying the optimal mixed strategies, a single simulation is performed, as seen in Figure 6.5. In this plot, the spacecraft can be seen in black while the sensor is green at observations where it successfully found the spacecraft and red at observations where it did not. By inspection, the win condition of a 50% success rate can be confirmed and the sensor is determined to be the winner.

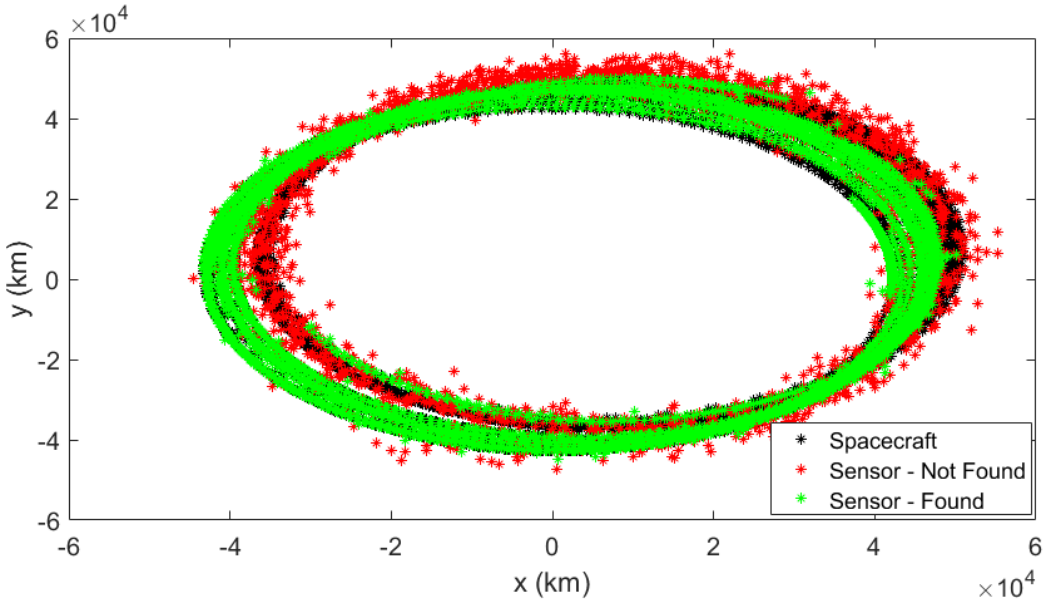


Figure 6.5. Spacecraft and Sensor States Through One Simulation

Successfully evading the sensor for even 50% of the observations taken is no small feat. The question is, though, how expensive is it for the spacecraft to be this successful? In the PPO approach, the maximum ΔV per maneuver is bounded so the total for all 30 maneuvers could not possibly exceed approximately 636 m/s over the approximately 180 day simulation. In addition, the time between maneuvers could not exceed 6 days.

When applying competitive coevolution, though, this is no longer the case. Figures 6.6 and 6.7 show the ΔV values and times between maneuvers, respectively.

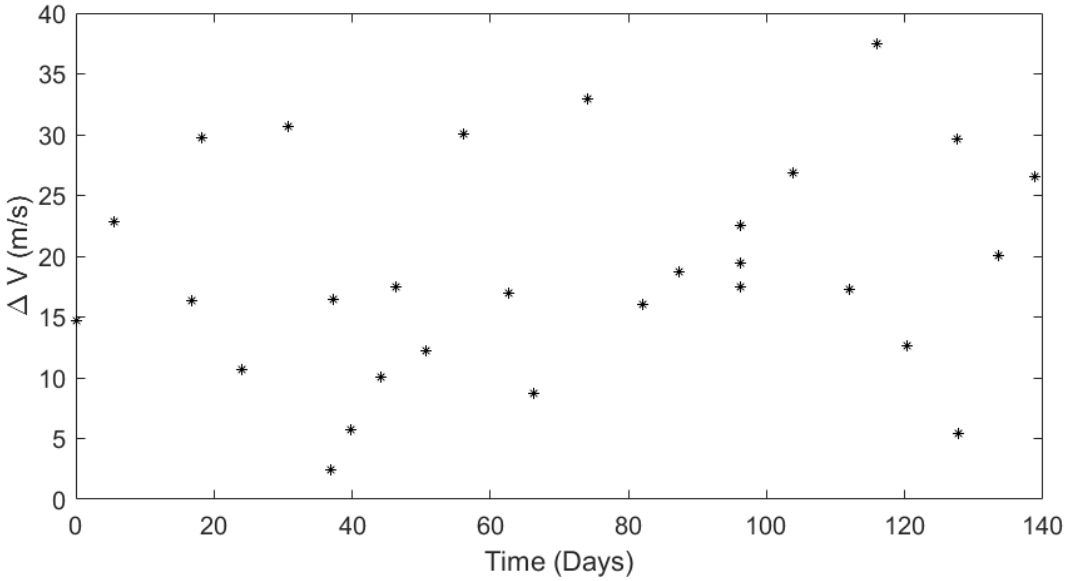


Figure 6.6. ΔV Applied for Each Maneuver

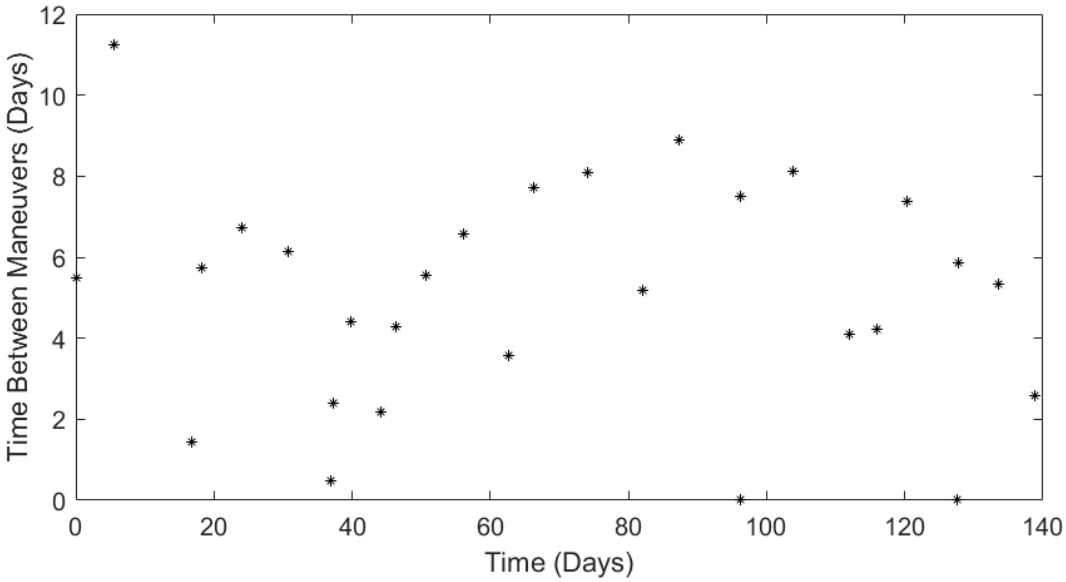


Figure 6.7. Time Between Each Maneuver

As expected, in an attempt to meet the 50% win condition, the ΔV values broke their original constraints. Originally constrained to maneuver magnitudes of 28.28 m/s,

competitive coevolution encouraged the values to increase to almost 40 m/s. In the end, it took 566 m/s and an average of 4.72 days between each maneuver to attempt to win the game with this condition. It can also be noticed that a pattern is forming in these plots. Though 30 data points are not conclusive, it appears that the most effective ΔV values for sensor avoidance fall between 10 and 25 m/s, with an average of 18.87 m/s. In addition, the time between maneuver values were not minimized to near-zero as one might expect when attempting to avoid detection by a sensor. This is likely because of the 20 hours each day when the spacecraft is free to maneuver or coast without fear of being observed. Therefore, the spacecraft optimizes its objective by maneuvering between observation periods. Interestingly, though, the bounds on the time between maneuvers was also broken in the coevolution. The previous maximum of six days between maneuvers was exceeded to allow up to almost 12 days at times. This suggests that it took the sensor more than 6 days at times to regain custody of the spacecraft.

On this note, additional analysis can be performed by inspecting the pattern by which the sensor successfully finds the spacecraft. Looking closely at Figure 6.8, the sensor seems to have no difficulty in maintaining custody of the spacecraft for most of the simulation. However, the spacecraft does meet the 0.5 success rate threshold by evading observation by the sensor for 38 days in a row towards the end of the simulation. Depending on whether it is preferred that the spacecraft evades the sensor for consistent short periods or a singular large period, this could be desirable. Exploring further its capability to achieve the former, Figure 6.9 shows the first ten days of observations zoomed in on the x-axis. Though the sensor maintains custody of the spacecraft throughout this period, it appears that the spacecraft successfully evades the sensor at least twice a day (not shown are the observations where the spacecraft is found), preventing perfect confidence in the sensor's custody of the spacecraft at the very least.

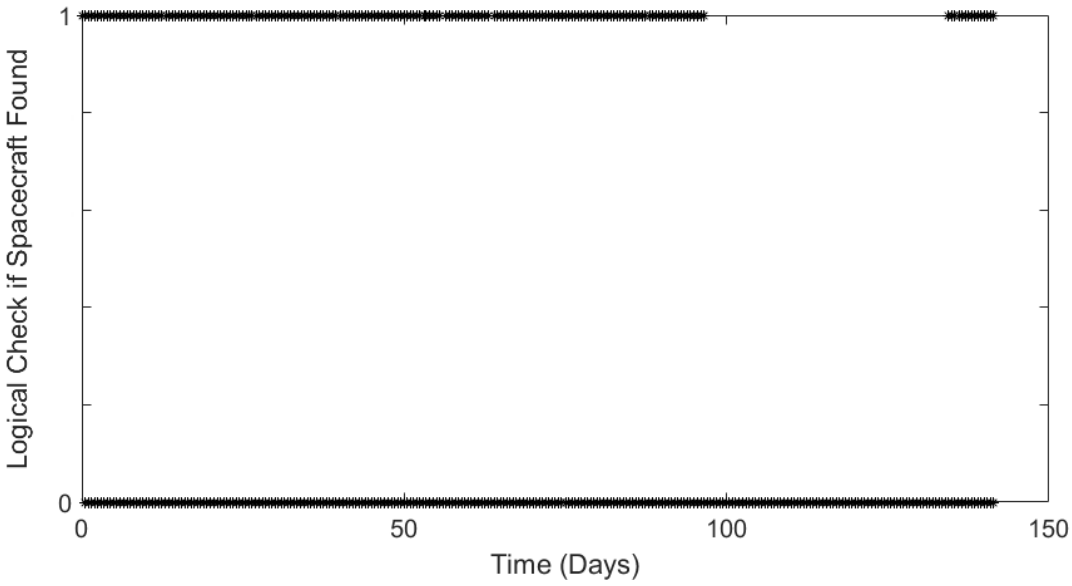


Figure 6.8. Time History of Succesful and Unsuccesful Observations

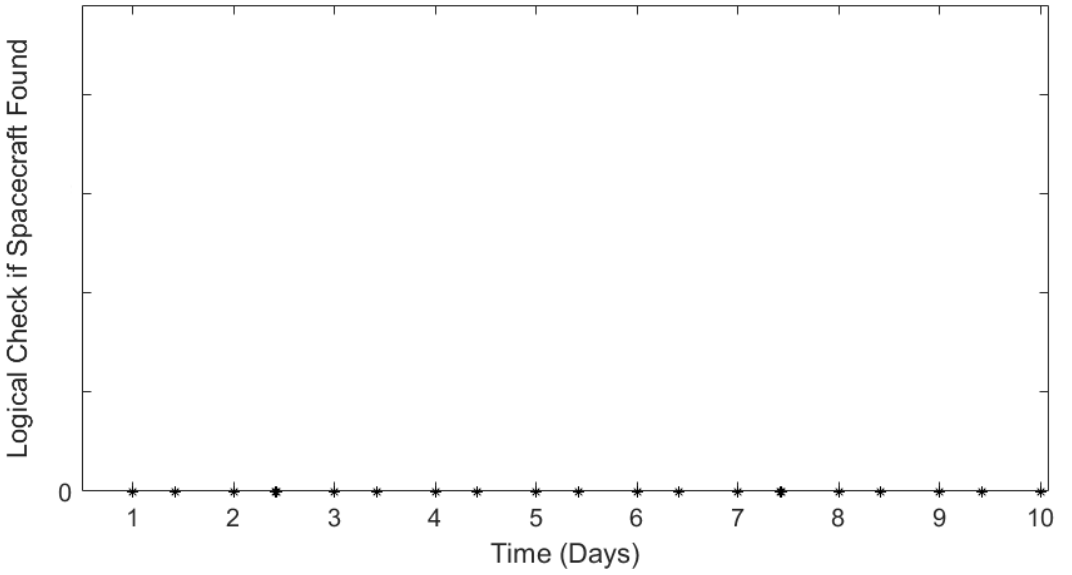


Figure 6.9. Time History of Succesful and Unsuccesful Observations - Days 0 to 10

6.3 Final Results: Final Two-Player Optimization

6.3.1 Problem Description

The scenario optimized in this section very closely mirrors that of Section 5.3 for the first part. Ideally, the resulting strategies from the PPO optimization would be used to initialize the strategies for the competitive coevolution optimization. However, the transformation of the strategies to truncated normalized distributions does not allow for them to be easily re-purposed. Instead, the strategies will be randomly initialized and fully optimized using the competitive coevolution approach. This has the secondary benefit of allowing the results of the PPO and competitive coevolution approaches to be directly compared. Therefore, instead of using the percentage of the observations where the spacecraft is found in the sensor's FOV as the fitness function, the distance between the sensor and the spacecraft will be used just as it serves as the objective function in the PPO optimization.

In the previous section, the competitive coevolution process requires a defined win condition that the two players are attempting to meet. In order to optimize the strategies such that the process resembles PPO, though, the approach must be adapted. Instead of using a constant win condition through-out the optimization, the win condition is allowed to vary as the evolution occurs and is defined as the previous best value achieved by the other player. This allows the players to evolve their strategies only with respect to each other.

Like in Section 5.3, the spacecraft is prescribed a maximum ΔV of 20 m/s in each direction. An infinite time-horizon is assumed, but the time between each maneuver is bounded between 12 minutes and 10 days. With the number of maneuvers capped at 30, this places an upper limit on the time-horizon at 300 days. These actions are again bounded between 0 and $4,216$ km in the $\hat{x}$ and $\hat{y}$ directions. In order to help ease some of the computational burden, the time step is doubled from six minutes to 12 minutes compared to the previous iterations.

If the spacecraft is successfully found in the sensor's FOV for 15 consecutive observations, custody is regained and the sensor can begin to follow the spacecraft along it's new path without applying the search strategy. In an attempt to simulate the nuances of state estimation, the sensor is afforded some flexibility in the search process. If the spacecraft has been successfully found for 7 consecutive observations after custody is lost, a missed observation does not restart the count unless the spacecraft is not located for

3 observations in a row, in which case the count to 15 must restart in order to regain custody.

Once the optimal strategies are found, competitive coevolution can be applied again to perform sensitivity analysis. To do this, individual constraints are relaxed/tightened as desired and the competitive coevolution process picks up where it left off. The strategies will then continue to evolve until a new NE is found.

6.3.2 Results

The strategies for both the sensor and the spacecraft take the same form as the 4-layer neural networks used by the policy optimization. For this analysis, all values are randomly initialized on a normal distribution. Besides the change in the implementation of a win condition compared to the previous sections, the population and Hall of Fame sizes were changed as well. It was found that increasing the population size helps to improve continuity in the optimization so a value of 20 was selected this time. In addition, some trial and error showed the setting the maximum Hall of Fame size to 30 was no less beneficial than setting it to 50 and provided significant computational cost savings. It would take significantly larger, computationally prohibitive, Hall of Fame sizes to perfectly represent the fully converged results for minimal improvement.

From the initialization, the strategies of the two players were allowed to co-evolve using the Hall of Fame method. As previously discussed, the fitness value in this optimization is the magnitude of the distance between the spacecraft and the center of the sensor's FOV, where the spacecraft is attempting to maximize the fitness while the sensor simultaneously tries to minimize it. Each iteration, the players are provided an updated win condition of the fitness based on the success of the opposite player in its previous iteration. This results in the win condition fitness becoming increasingly more difficult to achieve as the strategies evolve, eventually converging to the optimal value of the game. This process can be seen in Figure 6.10.

In Figure 6.10, the top figure shows the evolution of the fitness, normalized by the radius of the nominal orbit, and the bottom figure shows the resulting evolution of the fraction of the observations that the spacecraft is successfully found by the sensor (called "fraction found" here). Though the selected fitness function allows for a more continuous convergence of the strategies, the fraction found presents a value that is much easier to conceptualize. Inspecting these two plots, it can first be verified that the fitness and fraction found values move in the opposite direction for each player. This is to be expected since the spacecraft is attempting to maximize the distance from the sensor

while minimizing the frequency with which it is found by the sensor, and vice versa for the sensor. Additionally, since the two dependent variables are related, they generally follow the same pattern as the strategies evolve. The fitness plot just progresses more smoothly overall. Though there is still a fair amount of variation in the values at the end of the optimization, the variation is consistent and the mean of the values appears to be well defined. In the end, the fitness converges to a value of 0.72 while the fraction found converges to a value of 0.37 This means that the sensor is unsuccessful at locating the spacecraft in its FOV 63% of each episode.

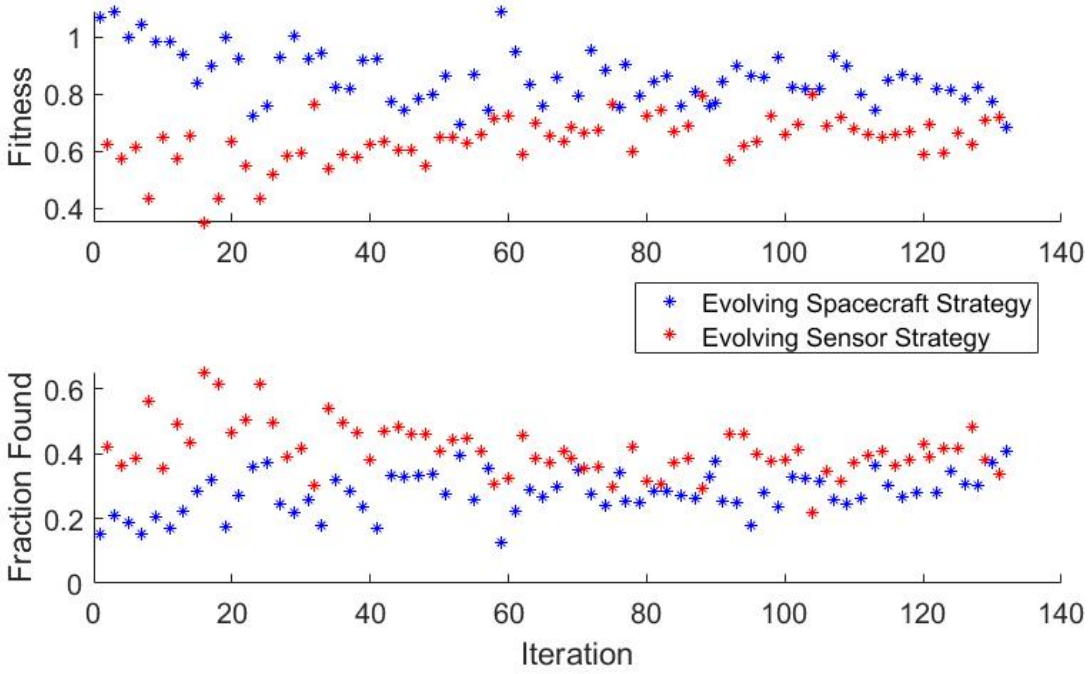


Figure 6.10. Fitness and Fraction Found At Each Competitive Coevolution Iteration

Since this scenario is identical to the one optimized using PPO and presented in the Section 5.3, the results can be compared rather easily. Earlier, the PPO approach found that the spacecraft was successful at evading the sensor 58.6% of each episode. These values vary between the two methods by less than 5%, close enough to verify a consistency between the two methods. The strategies themselves can be compared as well.

Inspecting the ΔV values, the components and magnitudes are plotted in Figures 6.11 and 6.12, respectively. While the fraction found values match that of the PPO closely,

the strategies vary highly. Competitive coevolution appears to have a difficult time optimizing the relative successes of the two players while still maintaining the desired randomization in their strategies. Therefore, the spacecraft's strategy favors consistently large maneuvers - in both Cartesian directions - over randomization. In fact, only a single maneuver fell below 20 m/s in magnitude. Altogether, the spacecraft expends a total of 765 m/s of ΔV over the course of 163 days in its attempt to evade the sensor, an average of 25.5 m/s per maneuver.

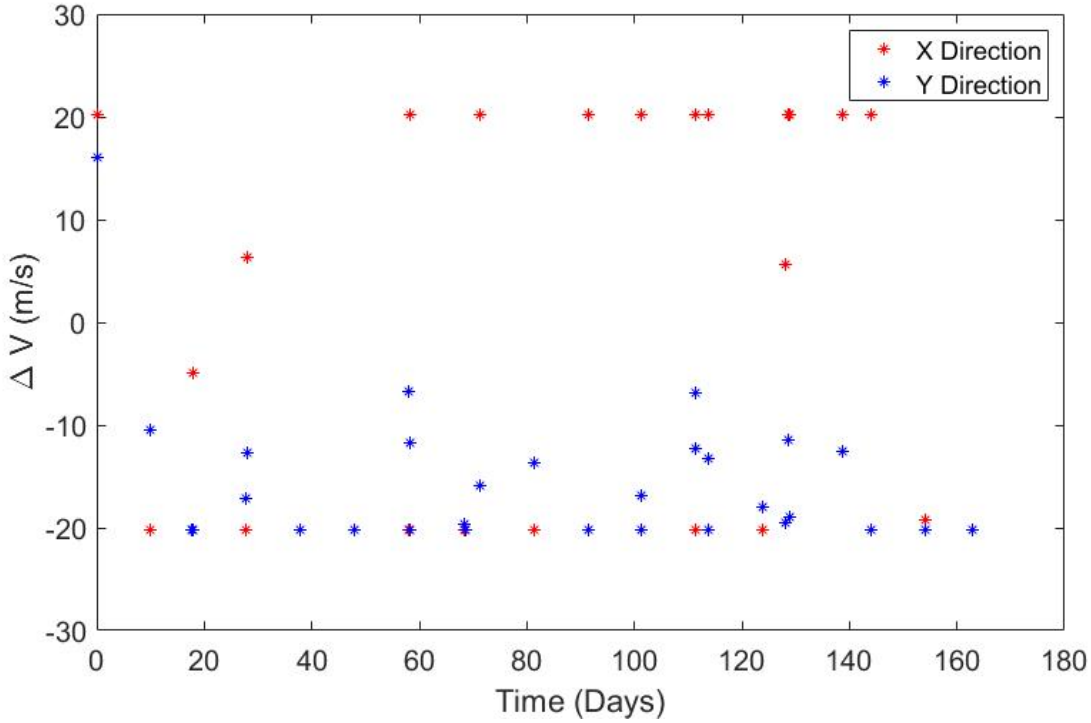


Figure 6.11. ΔV Components

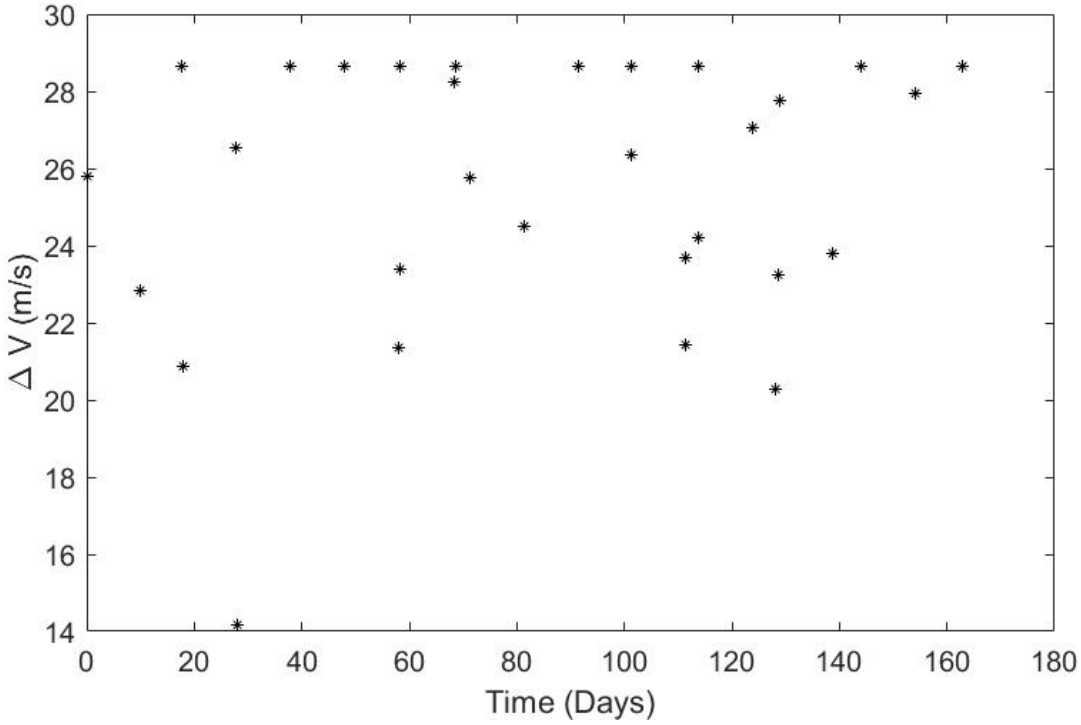


Figure 6.12. ΔV Magnitudes

The time between maneuver values can be seen in Figure 6.13. Similarly to the ΔV values, a pattern emerges where the times between maneuvers favors the bounds, in this case both the lower and upper. This suggests that most maneuvers will be placed in clusters of successive maneuvers, with the clusters spaced 10 days apart. Over the entire episode, the maneuvers are performed on average every once every 5.5 days.

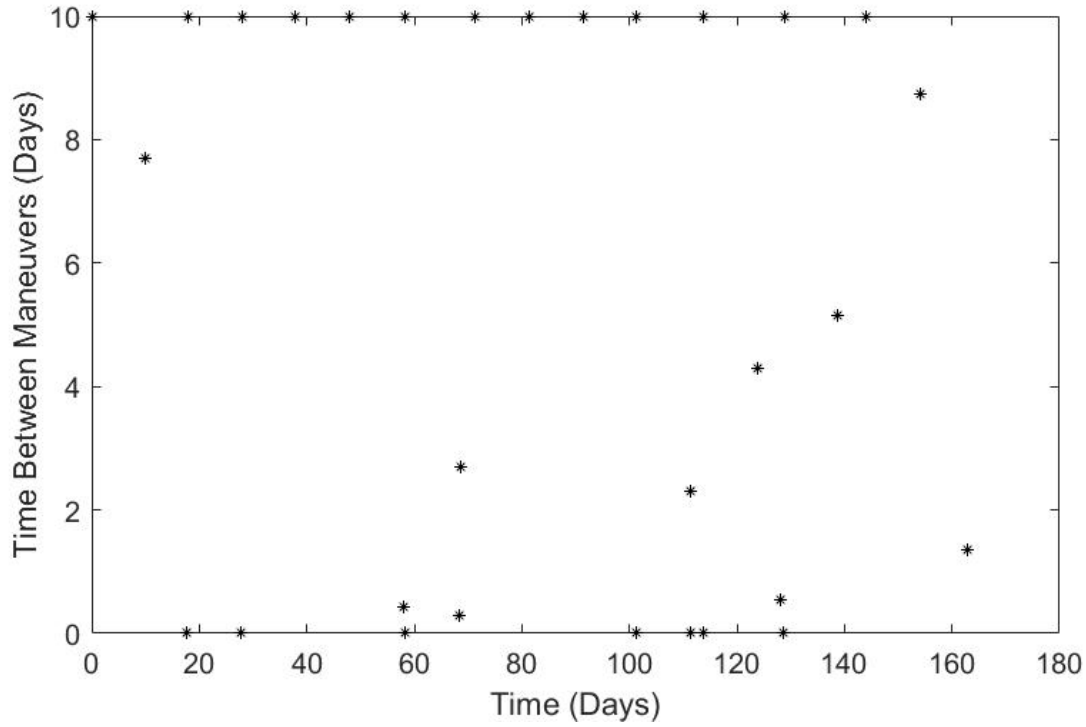


Figure 6.13. Time Between Each Maneuver

It is useful, again, to inspect the direction of each of the maneuvers. Figure 6.14 shows the angular values between the ΔV vectors and the horizontal in the ECI frame. Unlike when using PPO, a pattern is obvious in the angular values. Interestingly, two optimal angles for performing sensor avoidance maneuvers have been identified: $-45°$ and $-135°$. These two values are not surprising, though. The negative sign suggests, as is verified by 6.11, that all maneuvers are performed in the negative $\hat{y}$ direction. The magnitudes of these values place both angles $45°$ from the horizontal, just in different quadrants. This means that there is a single optimal angle to perform sensor avoidance maneuvers, just with the reference axis varying.

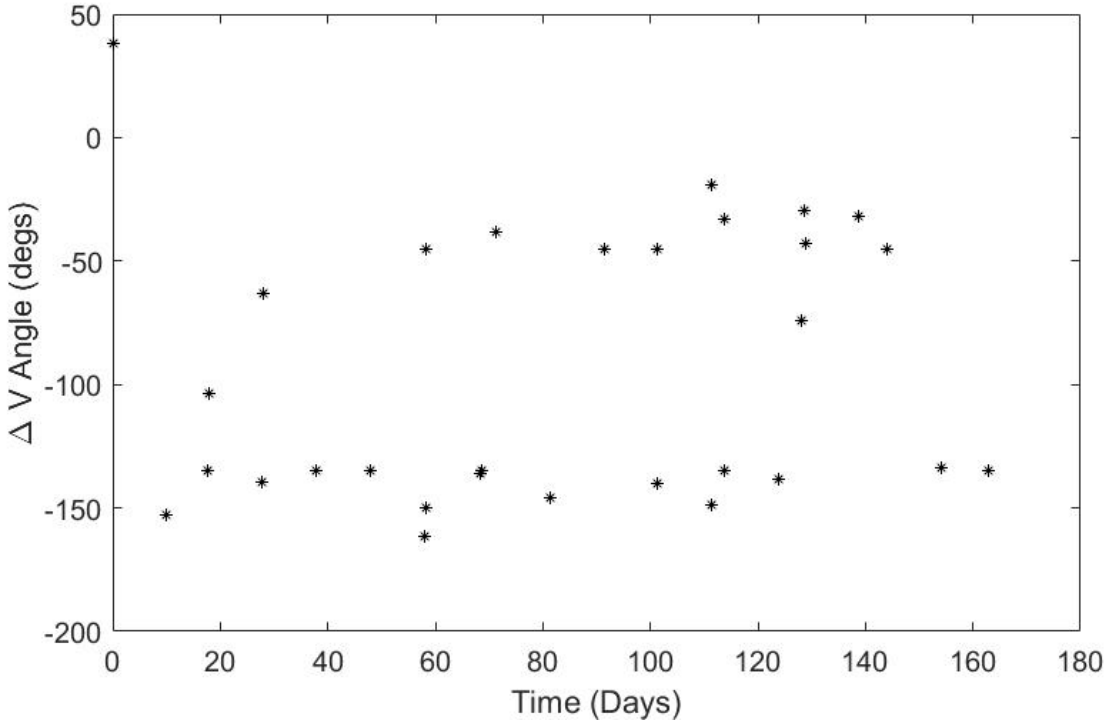


Figure 6.14. ΔV Angle

While attempting to track the satellite, the sensor's position with respect to the last known orbit of the satellite, in the $\hat{x}$ and $\hat{y}$ directions, over the course of half of a day can be seen in Figure 6.15, with the magnitude values of the change are presented in Figure 6.16. From these plots, it can be seen that, while the sensor does search most of the state space, it does spend most of its time near the maximum distance allowed by its constraint. Given the spacecraft's pattern of performing large maneuvers, this is not surprising. It also means that relaxing this constraint would likely help the sensor to improve the percentage of the episode in which it successfully locates the spacecraft.

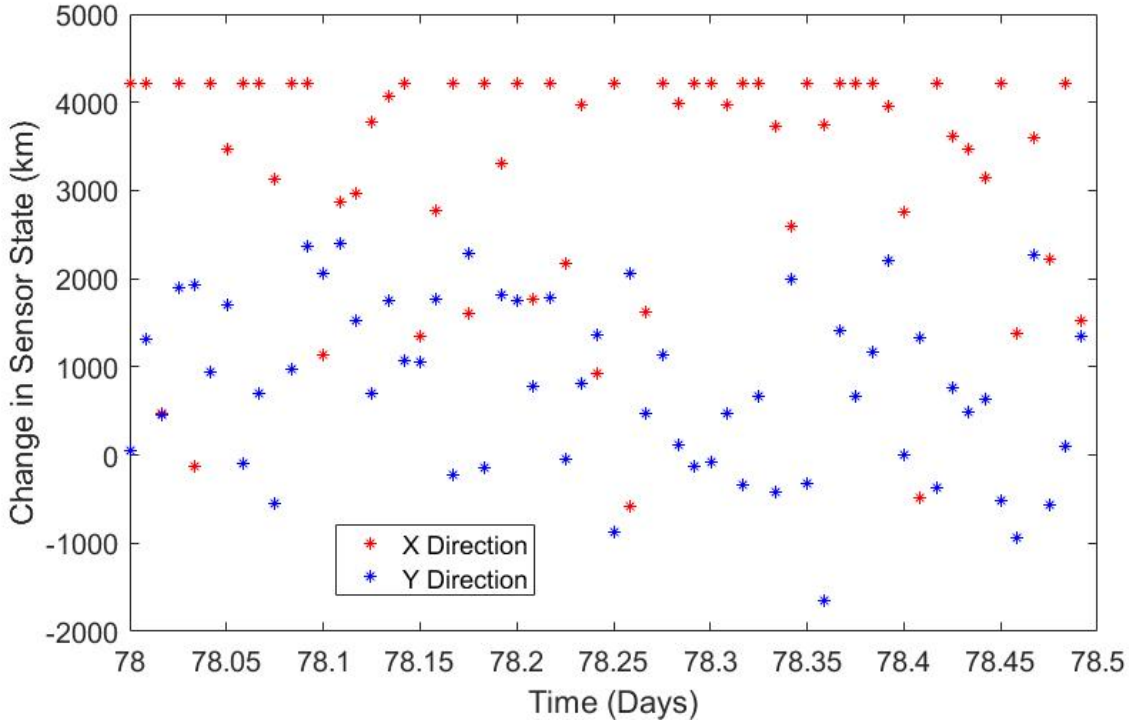


Figure 6.15. Sensor Distance From the Nominal Spacecraft Orbit for One Observation Period - In Components

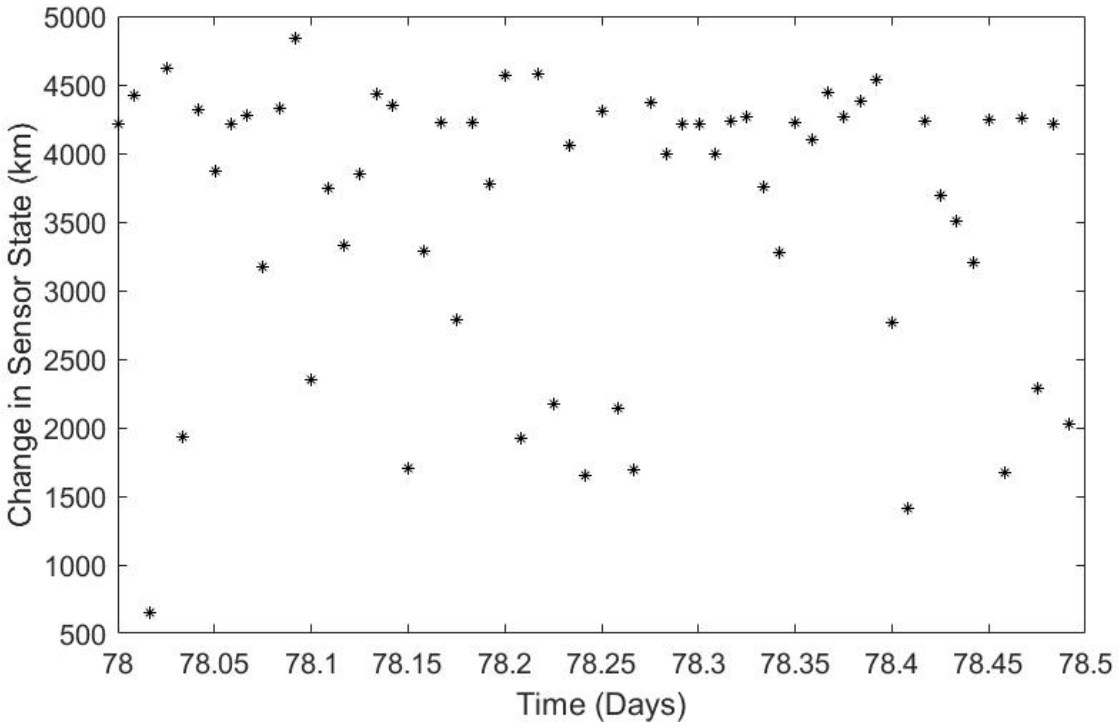


Figure 6.16. Sensor Distance Magnitude From the Nominal Spacecraft Orbit for One Observation Period

Sampling from the Hall of Fame, a single simulation of the game is performed. Figure 6.17 shows the states of the spacecraft and the sensor throughout one episdoe, in which the spacecraft can be seen in black, the sensor is shown in green at observations where it successfully found the spacecraft, and the sensor is shown in red at observations where it did not. As mentioned earlier, the sensor was determined to have successfully found the spacecraft 37% of the observations taken.

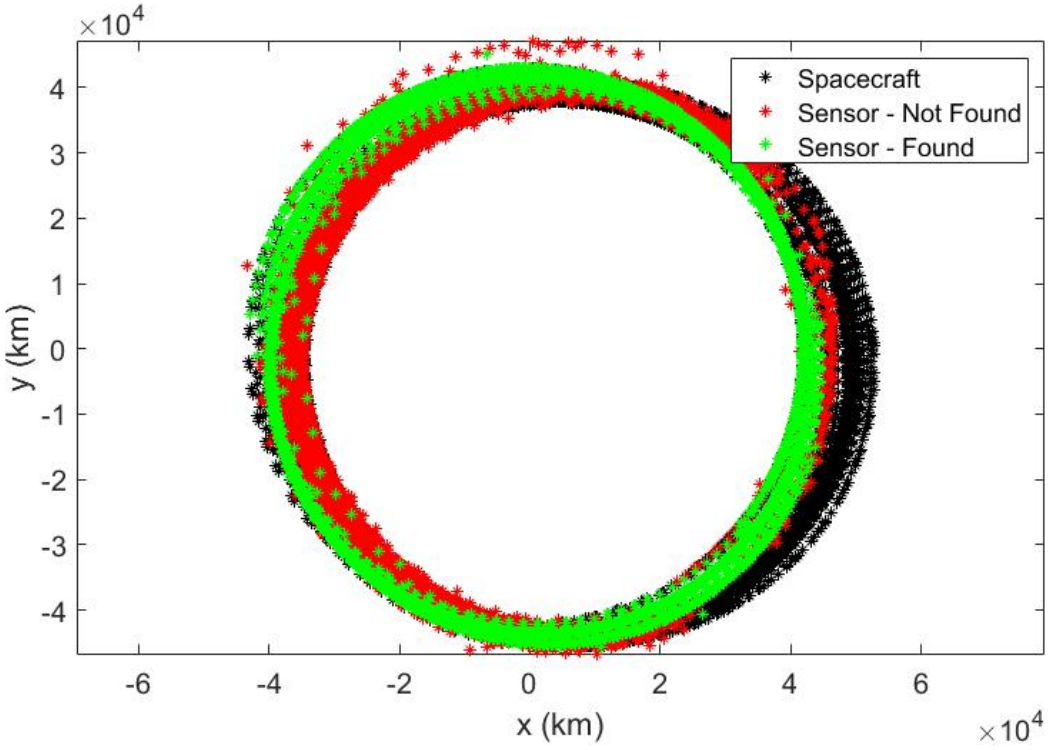


Figure 6.17. Spacecraft and Sensor States Through One Simulation

From Figure 6.17, it is not obvious whether a pattern develops in the timing of when the spacecraft is and is not successfully located by the sensor. Figure 6.18 shows the logical values for this analysis, where a value of 1 means that the spacecraft was successfully found. Similar to the results when applying PPO to the problem, the periods where the sensor does and does not find the spacecraft are both long in duration. However, there are still times within the period that the sensor maintains custody of the spacecraft in which it successfully avoids detection.

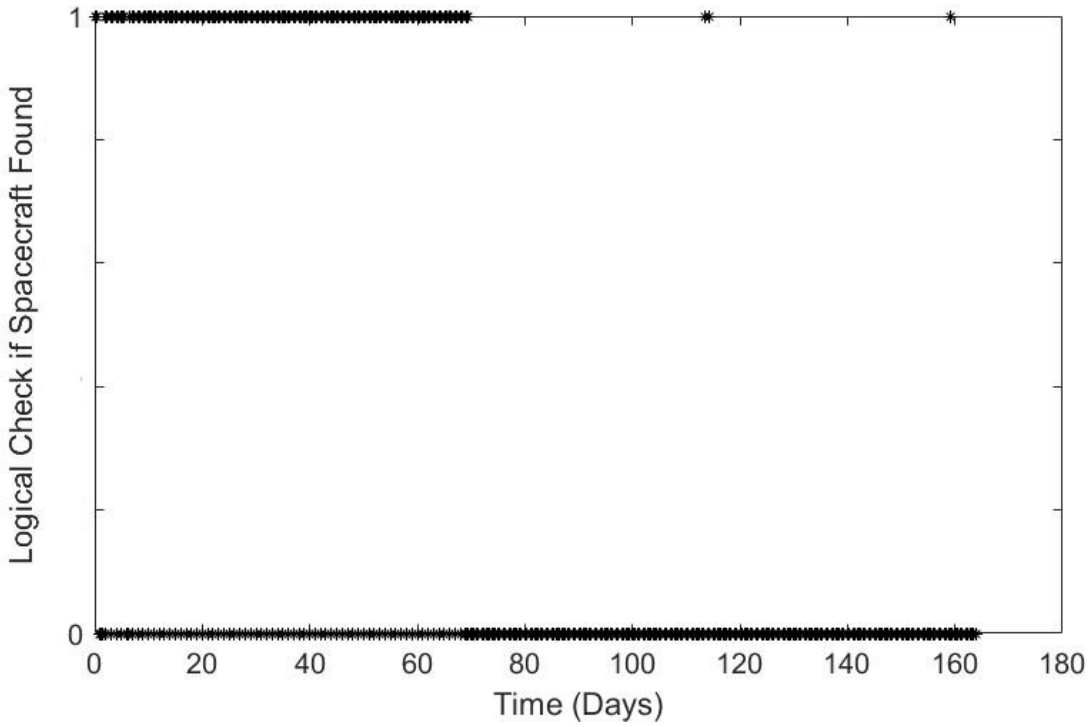


Figure 6.18. Time History of Succesful and Unsuccesful Observations

Overall, it was determined that the spacecraft could expend 25.5 m/s of ΔV every 5.5 days to successfully evade the sensor for 63% of the simulation, or 103 out of the total 163 days. This resulted in up to 44 days where the spacecraft went completely undetected. Though this is a *possible* strategy set for the two players to employ, it is unlikely to be the optimal NE. The PPO solution presented in Section 5.3 found a maneuver strategy for the spacecraft requiring almost half the amount of ΔV to be nearly just as successful in evading the sensor. In addition, the resulting competitive coevolution strategies lack the randomness that would be expected for a spacecraft that desires to remain un-trackable. However, the two methods do manage to produce similar "percent found" values, which mirrors the value of the game. Altering the constraints on the problem, irrespective of the solution method, should then affect the "percent found" value in a similar way.

The next step, and the primary motivation behind using competitive coevolution to begin with, is to perform sensitivity analysis on the presented results. The required size and timing of the maneuvers has been determined, but it is unknown how changing the constraints on both players affects their relative successes and the result of the game. To begin with, the constraint on the maximum ΔV is decreased by 25% from $20 m/s$ to 15 m/s in each direction. To accomplish this, the competitive coevolution optimization

is re-initialized using the previously determined optimal strategies and picks up right where it left off under the new constraints.

After an additional 75 iterations, the solutions re-converged to a new NE, as seen in Figure 6.19, where the cyan and magenta points represent the spacecraft and sensor's attempts at improving their fitness after the constraint was tightened. In this analysis, the goal is not to compare the resulting strategies, but to simply compare the effects of this change on the fitness and fraction found values. Here, the fitness converges to a value of 0.65 while the fraction found converges to a value of 0.44 Compared to the previous values, this is a change by 8.7% in the fitness and 9% in the fraction found. So a 25% decrease in the maximum ΔV increases the percentage the spacecraft is found by 9%. This suggests that continuing to constrain the spacecraft's ΔV may not have an equally adverse effect on the spacecraft's relative success in evading the sensor. In fact, producing more runs of varying ΔV constraints may result in a Pareto front of ΔV and fraction found values from which an "optimal" result could be chosen.

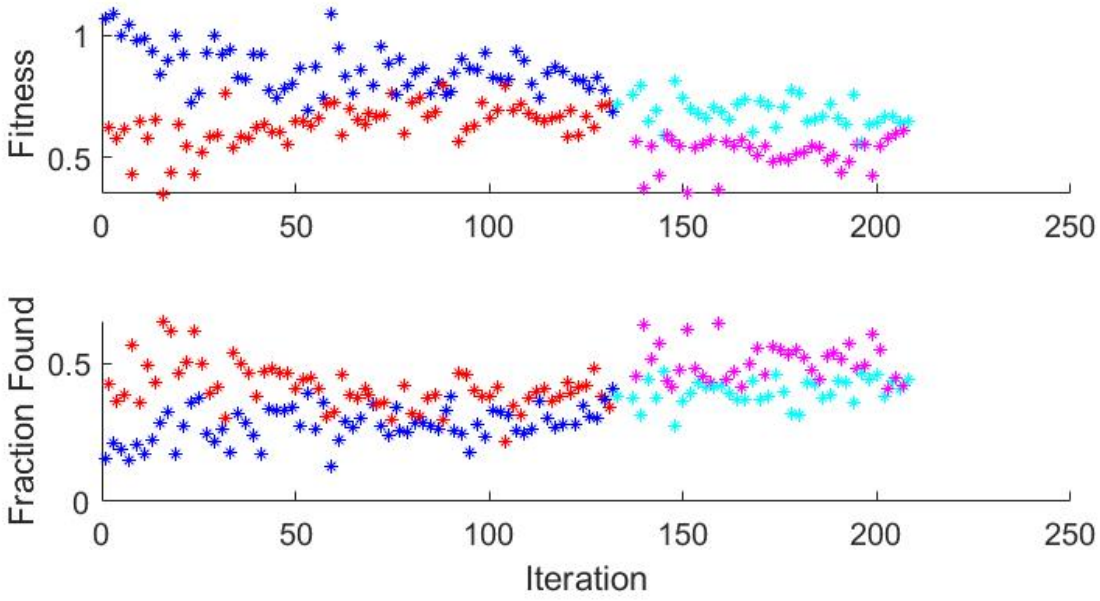


Figure 6.19. Fitness and Fraction Found At Each Competitive Coevolution Iteration

It is also beneficial to determine if altering the constraints on the sensor's strategy leads to a similar relationship between the constraint and the fraction found values for the sensor. To that end, the sensor's FOV is expanded from 4° to 5° while maintaining the spacecraft's constraints at their original values. After an additional 70 iterations, the solutions re-converged to a new NE, as seen in Figure 6.20. In this run, the fitness

converges to a value of 0.57 while the fraction found converges to a value of 0.47. Compared to the previous values, this is a change by 20% in the fitness and 21% in the fraction found. Interestingly, a 25% change in the sensor's constraint leads to over twice as large of a change in the fitness and fraction found values as an equally large change in the spacecraft's constraint. This suggests that the results are significantly more sensitive to the sensor's constraints than the spacecraft's. In addition, this shows how important technology advances are to the ground player. While the constraints of Keplerian dynamics and the laws of gravity will never make it more feasible for a spacecraft to expend large amounts of propellant in orbit (barring a major leap in propulsion technology), the sensor technology employed by the ground player continues to improve. This suggests that this game will favor the ground player more and more as technology advances. In addition, the results of this research could be used to justify a necessity for such advances.

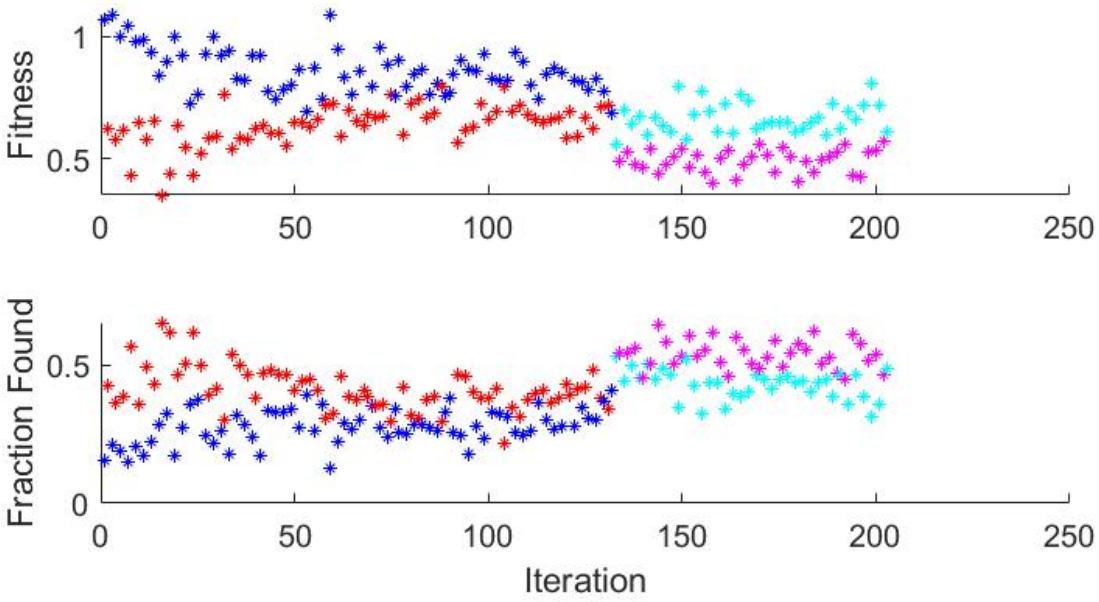


Figure 6.20. Fitness and Fraction Found At Each Competitive Coevolution Iteration

Chapter 7
Conclusions and Potential Future Directions

This chapter summarizes the main motivations, contributions, and results presented in this dissertation and draws conclusions from them. Suggestions for future work are also presented.

7.1 Conclusions

This dissertation explored, through the use of reinforcement learning, the process for optimizing a maneuver strategy for a satellite such that it is essentially un-trackable by an actor that may wish to disable it, tackled with a pursuit/evade game theory approach. The theory of fictitious play, discussed in Chapter 2, proves that a two-player game in which both players have perfect knowledge of the other player's move history will always converge to a single optimal value of the game. This means that, for a two-player game between an evasive spacecraft and a tasked sensor in pursuit, the game will converge to an optimal value where one player can be declared the winner, if given enough time.

To further validate the problem and the selected approach, Chapter 3 presented a series of experiments that were conducted to explore the capabilities and limitations of the two players and the reinforcement learning approach. First, industry accepted state estimation techniques were used to determine how a state estimation filter responds when it encounters a maneuvering spacecraft. Second, it was demonstrated using reachability sets that optimizing a series of maneuvers for multi-objective sensor avoidance produces better results than optimizing individual maneuvers. Finally, a continuous-thrust transfer while avoiding detection was optimized using both reinforcement learning and optimal control to demonstrate the capability of the proposed methodology compared

to traditional methods.

Chapter 4 then discussed the designed sensor avoidance problem in greater detail, how it is implemented in the reinforcement learning environment, and how the policy optimization methods detailed in Chapter 2 are combined to produce the best approach for optimizing the two players' strategies. This chapter concluded by discussing the contributions of this research with respect to previous work performed in related areas. Most notably, this included the first application of PPO to a large, real-world two-player game, adjusting the reward signal to account for the mean of each individual reward earned rather than the total, and adjusting the structure of the neural networks that represent the policies of the two players to ensure better convergence. These contributions, in the context of space object tracking, are summarized in Table 7.1.

Table 7.1. Summary of Contributions

These contributions include:
• First application of RL to sensor avoidance
• First application of PPO to a real-world two-player game
• Solved a non-trivial, long time-duration, sparse data problem with RL
• Unique use of mean reward objective function in PPO
• First documented use of truncated normal distribution to bound actions in PPO

Implementing these advances, Chapter 5 presented the results for applying reinforcement learning to three different iterations of the designed problem. The first iteration demonstrated the effectiveness of PPO in optimizing the spacecraft's maneuvers without the sensor tasking included in the optimization and discovered that, without the sensor's strategy optimized alongside the spacecraft's, the spacecraft is able to identify an optimal interval and magnitude at which to perform the avoidance maneuvers. The second iteration presented initial results when the sensor is optimized in the PPO algorithm as well. Between these two iterations, the latter especially, the importance of the advances presented in Chapter 4 were discovered and emphasized. It was only once the reward signal was adjusted to account for the mean instead of the total reward, and the neural networks were modeled using truncated normal distributions, that the loss values were found to be properly converging. The third and final iteration presented converged solutions with these advancements incorporated. In a single simulation of the strategies, it was determined that the spacecraft could expend 15.8 m/s of ΔV every 5.3 days on average (with no discernable pattern) to successfully evade the sensor for 58.6% of the

simulation, or 94 out of the total 160 days using seemingly random maneuvers. This resulted in up to 90 days where the spacecraft went completely undetected with a strategy likely to not be predicted accurately by the sensor. Though this is a rather significant amount of fuel, it is worthwhile to note that the objective function used does not take ΔV into account at all, so this is not the most ΔV-conscious solution possible. In addition, evading the sensor for 58.6% of time is a rather lofty, and potentially unnecessary goal. Significantly less ΔV could be used and still obtain a reasonable amount of time outside of the sensor's FOV. Furthermore, evading detection by the sensor is much harder than evading custody of the sensor. Using the sensor's uncertainty in its estimate of the spacecraft's state, rather than the current distance objective, would lend itself well to a custody evasion analysis and would likely require much less propellant to accomplish. This is discussed further in the next section.

Chapter 6 summarized the results from applying competitive coevolution to solve the same problem. Unlike the PPO results, the competitive coevolution strategies did not favor randomization and instead opted to perform large, successive maneuvers in an attempt to brute force a favorable result. Despite the variation between the optimal strategies produced from the PPO and competitive coevolution methods, they do produce similar "percent found" values, which suggests that the sensitivity analysis performed using competitive coevolution would translate directly to the PPO results. Performing the sensitivity analysis, it was determined that a 25% decrease in the maximum ΔV constraint on the spacecraft increases the percentage of observations that the spacecraft is found by 9%. This suggests that continuing to constrain the spacecraft's ΔV may not have an equally adverse effect on the spacecraft's relative success in evading the sensor. In fact, producing more runs of varying ΔV constraints may result in a Pareto front of ΔV and fraction found values from which an "optimal" result could be chosen. Conversely, expanding the constraint on the sensor's FOV by 25% had over twice the effect on the percentage of observations that the spacecraft is found. This suggests that the results are significantly more sensitive to the sensor's constraints than the spacecraft's. In addition, this shows how important technology advances are to the ground player. While the constraints of Keplerian dynamics and the laws of gravity will never make it more feasible for a spacecraft to expend large amounts of propellant in orbit (barring a major leap in propulsion technology), the sensor technology employed by the ground player continues to improve. This suggests that this game will favor the ground player more and more as technology advances. In addition, the results of this research could be used to justify a necessity for such advances.

The results presented in Chapters 5 and 6 can be seen summarized in Table 7.2. Note, though, that these are known only to be true for the considered test cases.

Table 7.2. Summary of Results

Notable results include:
• PPO can simultaneously optimize both strategies and achieve convergence
• Spacecraft meets its objective by evading sensor months at a time
• PPO is superior to competitive coevolution in optimizing unpredictability
• PPO and competitive coevolution produce varied strategies but similar values
• Results are twice as sensitive to the sensor's constraints than the spacecraft's

7.2 Potential Future Directions

Given the significant computing time required to produce the results presented in this dissertation, there are many possible extensions of this work that simply could have feasibly been included.

Most importantly, it can be argued that solving the sensor avoidance problem is not as practical as solving the custody avoidance problem, the difference in implementation being the objective function selection. While using the distance between the sensor and the spacecraft as the objective proved useful, it could be more beneficial to instead use the uncertainty of the sensor's estimate of the state of the spacecraft. Unfortunately, this requires a filter to be constantly running to provide updates to the uncertainty every time an observation is collected. This would add significantly to the computation time which could not be afforded for this research. Implementing this would likely result in much more realistic ΔV values for the spacecraft since evading custody can be accomplished with much smaller maneuvers than evading detection.

In addition, the ΔV could be decreased further by incorporating it into the objective function. A combination of the current distance metric and ΔV would allow for mission designers to subjectively weight the two variables to produce mission-specific results that would likely be more reasonable to implement in real missions. One potential issue with this, though, is that the fundamental theory that proves the problem will converge requires that a zero-sum game be modeled. Therefore, incorporating ΔV in the objective function requires that the sensor operator has an equally vested interest in increasing the spacecraft's propellant expenditure. Though this is not an unreasonable assumption, it

could be mission dependent. With this in mind, the objective function could be expanded further to include the sensor's resource use (most likely time) as well.

Improvements could also be made to the realism of the scenario as well. While using a filter to track the sensor's uncertainty in the spacecraft's state, it would also be prudent to include both track association and track correlation steps. This would result in the spacecraft potentially determining unique strategies for taking advantage of inherent gaps in the sensing and state estimation capability.

An alternative approach, that may provide computational savings if approached correctly, would be to separate out the optimization of the best response strategies from the fictitious play convergence process. A more traditional optimal method could be used to optimize the strategy of one player while keeping the other player's strategy constant. This process would be repeated, alternating which player is optimizing their strategy each iteration, using fictitious play to optimize with respect to the strategy histories of both players. Though this approach would not lend itself well to the same sensitivity analysis process explored in this dissertation, it could prove useful in optimizing the strategies if the constraints on both players are known.

Along these lines, if the use of orbital perturbations or continuous control actions are not desired, propagating with numerical integration could be replaced with an analytic approximation, which would provide significant computation time savings.

Expanding the problem even further, it would be helpful to incorporate additional spacecraft or sensors into the scenario. With additional sensors especially, leveraging different types (optical vs radar) and different theaters (ground, space-based, stratospheric) could result in optimal architectures being determined for tracking an evasive spacecraft. Similarly, multiple spacecraft could be used to simulate space architectures involving either cooperative and/or competitive spacecraft, or using multiple spacecraft to hide the identity of a singular one.

Lastly, the potential applications for competitive coevolution in simulating similar two-player games have barely been touched. Not only could competitive coevolution be used to perform simple sensitivity analysis, but it could also be applied to design entire ground or space architectures. The possibilities are endless.

Appendix
Proof of Convergence

1 Justification

Given the computational requirements when optimizing the final version of the reinforcement learning approach, it was not expected to be able to reach convergence. When running the code on a personal laptop, the memory capacity was exhausted given the large batch size necessary to ensure quality updates of the policies. When running the code on the MIT Supercloud cluster, the limitation is simply time. Running thousands of integration calls per episode, necessary to simulate an episode length on the order of hundreds of days, takes over a month even on a 64-core node. In addition, given the convergence time for the full scenario, it was difficult to determine sufficient batch sizes and KL divergence target values. Conducting this proof-of-convergence helped to determine the values used in the final iteration of results presented in Section 5.3.

2 Simulated Problem

The primary cause of the extreme computational requirements is the necessity to propagate the spacecraft's state under Keplerian dynamics. To avoid this, a proof-of-convergence problem is simulated in which the change in both the sensor and the spacecraft's states are presented with respect to nominal Cartesian states along a 2-D circle with a normalized radius of one (no continuity between states is enforced). With 120 points traversed per revolution and a prescribed episode length of 600 steps, the pursuit evasion game between the two players is simulated for five complete orbits. Like the bounds placed on the sensor in the Keplerian-based simulations, both players are allowed to move only within 10% of the nominal orbit radius in each direction (14% in magnitude). Similarly, the

objective function for both players is set as simply the distance between their positions at any given point.

2.1 Results

With the hyperparameters, based on the values used in Reference [89] and adjusted though trial and error, set according to Table A.1, the convergence of this simplified problem can be demonstrated.

Table A.1. Proof-of-Convergence Hyperparameters

Parameter	**Value**
Discount Factor, γ	0.9
Batch Size	3200
Reward Steepness, λ	0.98
KL Divergence Target, d_{targ}	0.0003

Though most applications of PPO involving one player require at most $100,000$ episodes to converge, it was found that a simple version of this two-player game took over $1,000,000$ episodes to converge. As discussed in the main chapters, this is likely because of the back-and-forth process required to optimize the strategies of both players simultaneously. In addition, most applications require batch sizes on the order of 32 to 64 episodes to ensure that enough data points are collected to encourage proper convergence. However, for this game, it was determined that even 320 episodes per batch was too small. This resulted in the strategies converging quickly to unrealistic distributions. Instead, a batch size of 3200 is used for this proof-of-convergence, and subsequently, for the final iterations of the full, real-world scenario. In addition, the default value for the KL divergence target of 0.003, as used in Reference [89], was found to be insufficient as well. Though the strategies appeared to head towards convergence overall, they had difficulty taking small enough step sizes when required. Instead, a value of 0.0003 for the KL divergence target was selected. Meanwhile the discount factor was set to 0.9, smaller than most applications but slightly larger than what was used in the real scenario. This is because the lack of Keplerian dynamics in this study requires less consideration of points further in the future. Combined with a reward steepness of 0.98 (unchanged through-out this research), these hyper-parameters resulted in a fully converged solution.

To verify that convergence has indeed occurred, the moving average of the loss values

for the policies of the two players, normalized by their largest magnitude values, are plotted together in Figure A.1. Though the losses do not steadily decrease, they do both converge to values very close to zero. This is the desired result and demonstrates that PPO is capable of solving for a Nash Equilibrium in complex two-player zero-sum games.

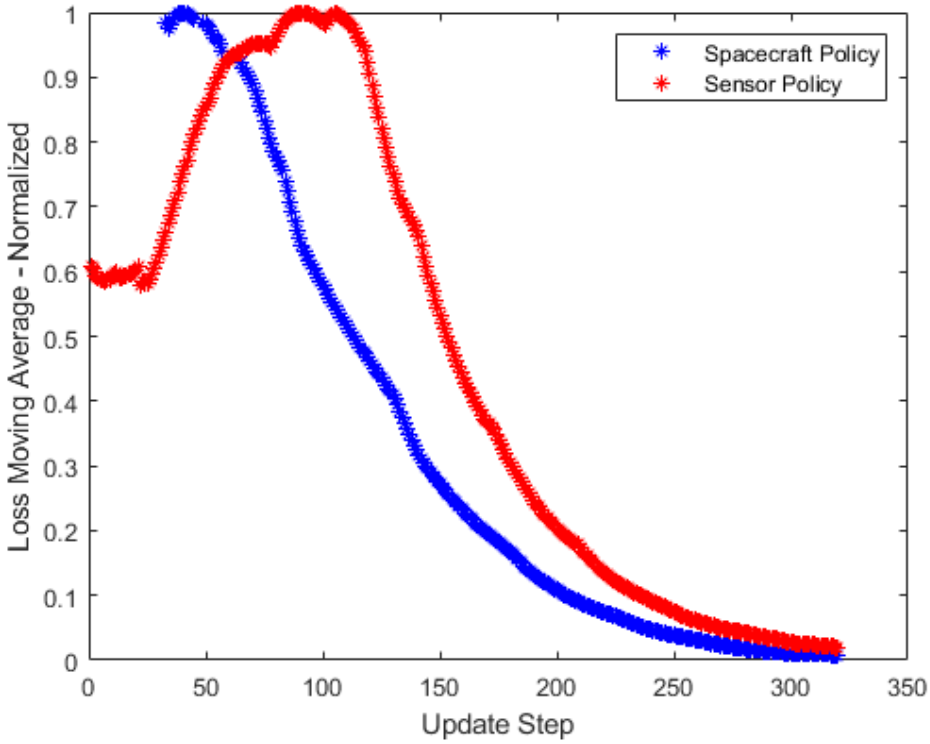


Figure A.1. Loss Values for Spacecraft and Sensor Policies

Beyond the losses, it's also valuable to inspect the resulting optimal strategies for the two players. A plot of the positions of the two players, normalized to one, in the final episode of the optimization can be seen in Figure A.2. Interestingly, the sensor's states appear to converge to a ring within a couple percent of the nominal orbit, while the spacecraft continues to explore the entire state space. This is likely because both players are equally advantaged in this scenario. Therefore, the objective function (a function of only the distance between the two players) is bound to converge to a mean distance value somewhere close to half of the max distance. This is accomplished if one of the two players, the spacecraft in this case, explores the entire space while the other, the sensor, remains close to the nominal orbit. In this scenario, if the spacecraft were to attempt to learn from its relative success and adapt to only travel near the maximum distance allowed by the constraints, then the sensor would react accordingly and search a larger

region of the space. That is the beauty of PPO and Nash Equilibria.

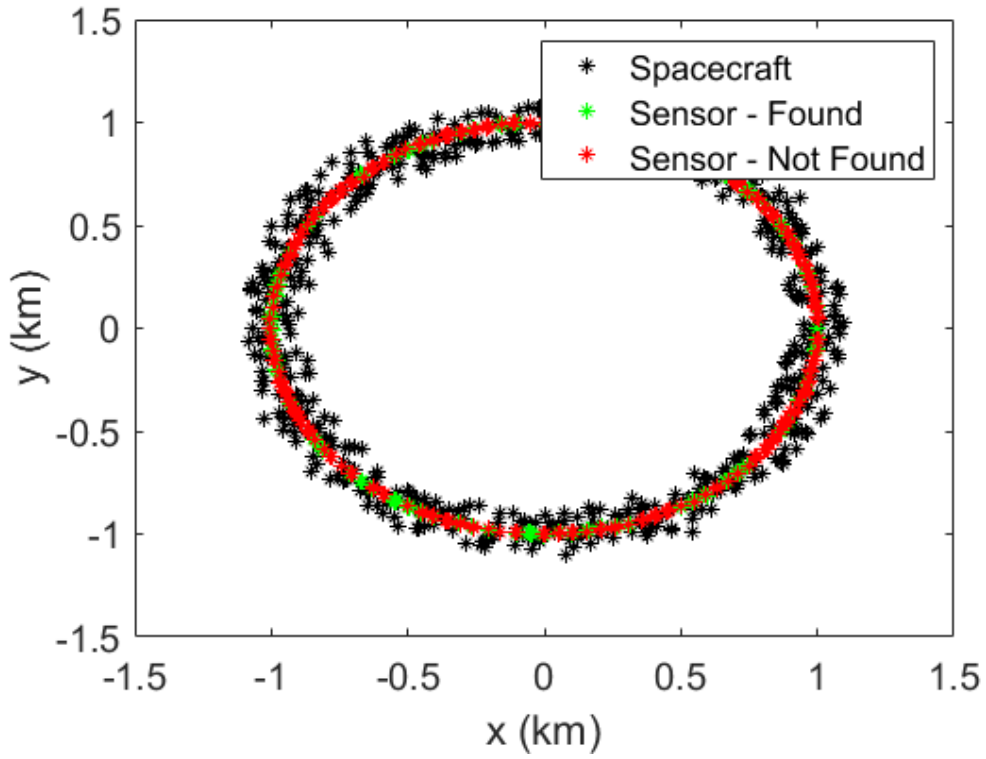


Figure A.2. Simulation of the Spacecraft and Sensor States for One Episode

Figures A.3 and A.4 verifies these analysis by inspection. Figure A.3 shows the magnitude of the states selected for the sensor, expressed in percentage of the radius of the nominal orbit. These values lie exclusively within 1.25% of the nominal orbit magnitude. The spacecraft, shown in Figure A.4, meanwhile, covers the entire space available within 14% of the nominal orbit. Figure A.5 shows how the states of the two players overlap in this space, with the sensor shown in red and the spacecraft shown in blue.

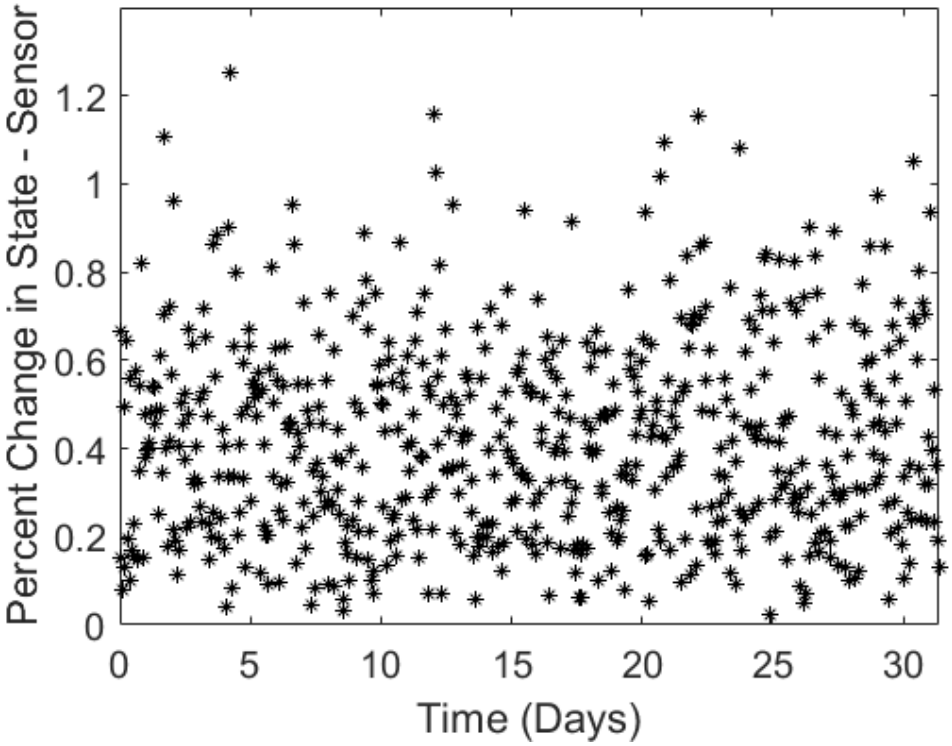


Figure A.3. Magnitude of Sensor State in Percentage Variation from Nominal

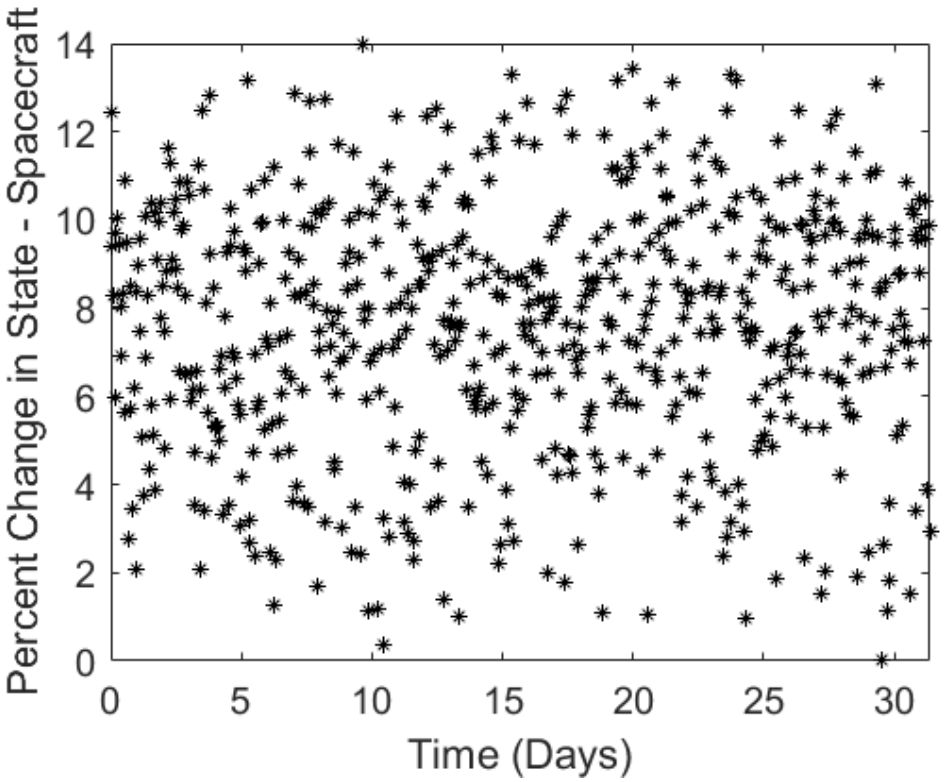


Figure A.4. Magnitude of Spacecraft State in Percentage Variation from Nominal

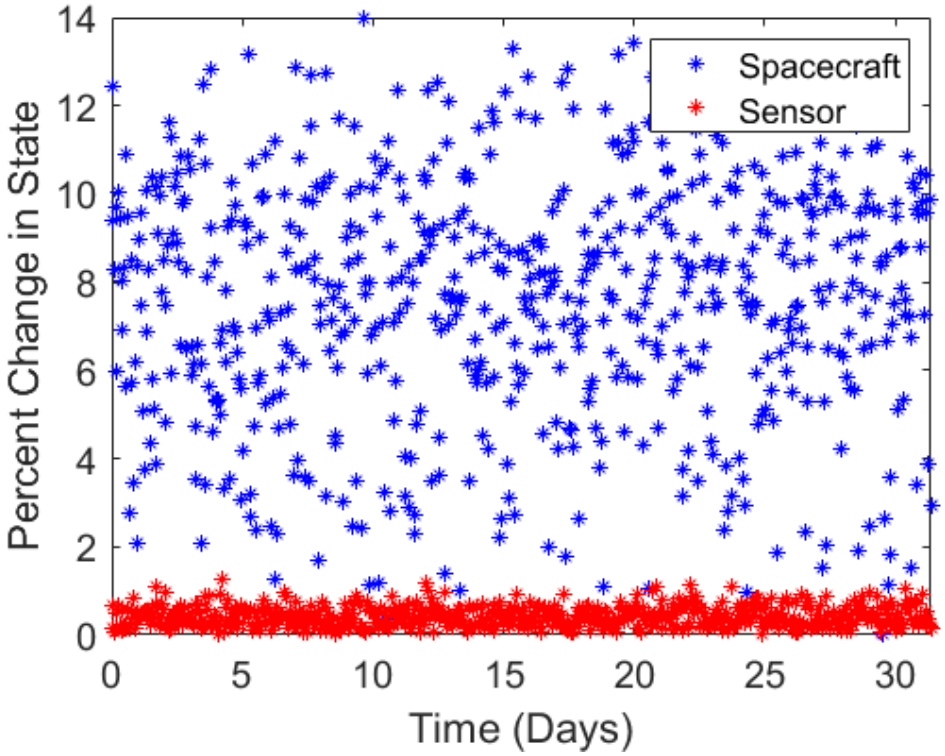


Figure A.5. Magnitude of Spacecraft and Sensor States in Percentage Variation from Nominal

It's not obvious from this plot, though, that the game converged to strategies as equally matched as the problem formulation would suggest. Figure A.6 shows the times where the spacecraft is found, and not found, by the sensor, where a value of 1 signifies that the spacecraft was found in the sensor's FOV. Though not necessarily obvious from this plot, the points are evenly distributed between the two logical values. It was determined that a FOV of 0.08 in the normalized Cartesian coordinates results in the spacecraft being located in the sensor's FOV around 50% of the episode. This corresponds to about 4.5° if the orbital radius were set to that of a geosynchronous spacecraft - close to the 4° FOV used in the actual real-world scenario. This suggests that, not only should the real-world strategies converge to realistic distributions using PPO, but that the relationship between the two players is fairly represented as well.

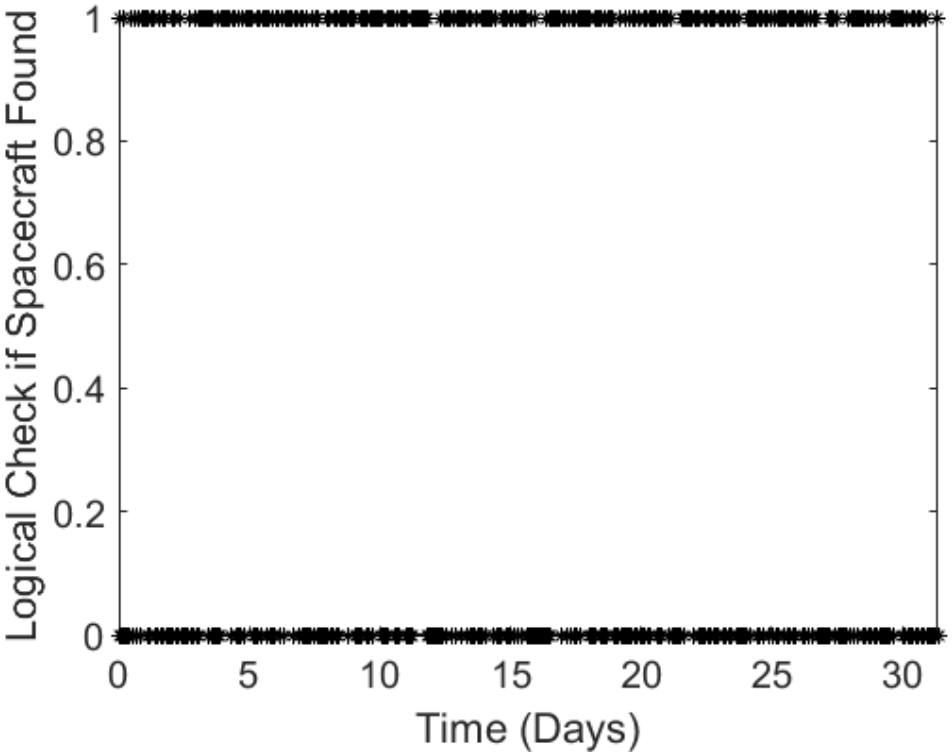


Figure A.6. Found History for One Episode

References

[1] Corp., RAND. Project AIR FORCE. 1996.

[2] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O., Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347, 2017.

[3] Vallado, D. A., Fundamentals of Astrodynamics and Applications, Springer, New York, NY, 4th ed., 2013.

[4] Nash, J. "Non-Cooperative Games." Annals of Mathematics, vol. 54, no. 2, 1951, pp. 286–295.

[5] Owen, G., *Game Theory*. Academic Press Inc., 1995

[6] Fudenberg, D., and Levine, D., The Theory of Learning in Games. Cambridge, MA, USA: MIT Press, 1998.

[7] Brown, G.W., "Iterative Solutions of Games by Fictitious Play", In Activity Analysis of Production and Allocation, T.C. Koopmans (Ed.), New York: Wiley. 1951.

[8] Berger, U., "Brown's original fictitious play," *Journal of Economic Theory* 135:572-578, 2007.

[9] Popovici, E., Bucci A., Wiegand R.P., De Jong E.D. (2012) Coevolutionary Principles. In: Rozenberg G., Back T., Kok J.N. (eds) Handbook of Natural Computing. Springer, Berlin, Heidelberg

[10] Eiben, A.E., and Smith, J.E. (2015). Introduction to Evolutionary Computing (2nd ed.). Springer Publishing Company, Incorporated.

[11] Rosin, C.D., and Belew, R.K., "New methods for competitive coevolu-tion," *Evolutionary computation*, 1997 - MIT Press, 1997, pp. 1-29.

[12] Nogueira M., Cotta C., Fernández-Leiva A.J. (2013) An Analysis of Hall-of-Fame Strategies in Competitive Coevolutionary Algorithms for Self-Learning in RTS Games. In: Nicosia G., Pardalos P. (eds) Learning and Intelligent Optimization. LION 2013. Lecture Notes in Computer Science, vol 7997. Springer, Berlin, Heidelberg

[13] Stanley, K.O., Miikkulainen, R., "Competitive coevolution through evolutionary complexification," *Journal of Artificial Intelligence Research*, v.21 n.1, p.63-100, January 2004

[14] Sutton, R.S., and Barto, A.G. (2018). Reinforcement Learning: An Introduction. A Bradford Book, Cambridge, MA, USA.

[15] Bishop, C.M. (2006). Pattern Recognition and Machine Learning (Information Science and Statistics). Springer-Verlag, Berlin, Heidelberg.

[16] Goodfellow, I., Bengio, Y., Courville, A., *Deep Learning*, The MIT Press, 2016

[17] Schulman, J., Levine, S., Moritz, P., Jordan, M.I., and Abbeel, P. "Trust region policy optimization". In: CoRR, abs/1502.05477 (2015).

[18] Kakade, S. and Langford, J. Approximately optimal approximate reinforcement learning. In ICML, volume 2, pp. 267–274, 2002.

[19] Bellemare, M. G., Naddaf, Y., Veness, J., and Bowling, M. The arcade learning environment: An evaluation platform for general agents. Journal of Artificial Intelligence Research, 47:253–279, jun 2013.

[20] Williams, R.J. "Simple statistical gradient-following algorithms for connectionist reinforcement learning". In: Machine learning 8.3-4 (1992), pp. 229–256.

[21] Mnih, V., Badia, A.P., Mirza, M., Graves, A., Lillicrap, T.P., Harley, T., Silver, D., K. Kavukcuoglu, K. "Asynchronous methods for deep reinforcement learning". In: arXiv preprint arXiv:1602.01783 (2016).

[22] Kingma, D., and Ba, J. "Adam: A method for stochastic optimization". In: arXiv:1412.6980 (2014).

[23] Schulman, J., Moritz, P., Levine, S., Jordan, M., and Abbeel, P. High-dimensional continuous control using generalized advantage estimation. arXiv preprint arXiv:1506.02438, 2015.

[24] Ng, A.Y., Daishi, H., and Russell, S. Policy invariance under reward transformations: Theory and application to reward shaping. In ICML, volume 99, pp. 278–287, 1999.

[25] Berger, J.M., Moles, J.B., and D.G Wilesey. *An Analysis of USSPACECOM's Space Surveillance Network Sensor Tasking Methodology.* MS Thesis. Air Force Institute of Technology. 1992.

[26] Erwin, R. S., Albuquerque, P., Jayaweera, S., and Hussein, I., "Dynamic Sensor Tasking for Space Situational Awareness," *American Control Conference (ACC)*, Institute of Electrical and Electronics Engineers, Jun 2010, pp. 1153-1158, doi: 10.1109/ACC.2010.5530989.

[27] Williams, P. S., Spencer, D. B., and Erwin, R. S., "Coupling of Estimation and Sensor Tasking Applied to Satellite Tracking," *Journal of Guidance, Control, and Dynamics, Vol. 36*, Jul, pp. 993-1007.

[28] Tian, K., and Zhu, G., "Sensor Management Based on Fisher Information Gain," *Journal of Systems Engineering and Electronics, Vol. 17, No. 3*, 2006, pp. 531–534. doi:10.1016/S1004-4132(06)60091-1

[29] Wolf, A., Swift, J., Swinney, H., and Vastant, J., "Determining Lyapunov Exponents from Time Series," *Physica D, Vol. 16, No. 3*, 1985, pp. 285–317. doi:10.1016/0167-2789(85)90011-9

[30] Adurthi, N., Singla, P., and Singh, T. "Conjugate unscented transform and its application to filtering and stochastic integral calculation." AIAA Guidance, Navigation, and Control Conference, August 2012.

[31] Guestrin, C., Krause, A., and Singh, A.P., "Near-optimal sensor placements in gaussian processes," in *Proceedings of the 22nd international conference on Machine learning*. ACM, 2005, pp. 265–272.

[32] Adurthi, N., Singla, P., and Majji, M., "Mutual Information Based Sensor Tasking with Applications to Space Situational Awareness," Journal of Guidance, Control, and Dynamics, Feb. 2020.

[33] Gualdoni, M.J., and DeMars, K.J., "The Information Content of Data Arcs for Multi-Step Sensor Tasking", 2018 Space Flight Mechanics Meeting, AIAA SciTech Forum, (AIAA 2018-0727)

[34] DeMars, K. J. and Jah, M. K., "Evaluation of the information content of observations with application to sensor management for orbit determination," *Advances in the Astronautical Sciences, Vol. 142*, 2011, pp. 3187-3206.

[35] Goff, G.M., Black, Beck, J.A., and Hess, J.A., "A Dynamic Sensor Tasking Strategy for Tracking Maneuvering Spacecraft using Multiple Models", *AIAA Guidance, Navigation, and Control Conference*, *AIAA SciTech Forum*, (AIAA 2016-1859).

[36] Williams, P. S., Coupling Between Nonlinear Estimation and Dynamic Sensor Tasking Applied to Satellite Tracking, Ph.D. dissertation, Department of Aerospace Engineering, Pennsylvania State University, University Park, PA, Dec 2012.

[37] Benkoski, S.J., Monticino, M.G., and Weisinger, J.R., "A survey ofthe search theory literature," Naval Research Logistics, vol. 38, no. 4,pp. 469-494, 1991.

[38] Chen, H., Shen, D., Chen, G., and Blasch, E., "Tracking Evasive Objects via A Search Allocation Game," *IEEE ACC*, 2010.

[39] "Prisoner's Dilema." Team: Tokyo Tech. `http://2015.igem.org/Team:Tokyo_Tech`

[40] Isaacs, R. (1965). "Differential Games: A Mathematical Theory with Applications to Warfare and Pursuit, Control and Optimization." New York: John Wiley & Sons. OCLC 489835778.

[41] Shen, D., Jia, B., Chen, G., Blasch, E., Pham, K., "Pursuit-evasion games with information uncertainties for elusive orbital maneuver and space object tracking," *Proc. SPIE*, Vol. 9469, 2015.

[42] Shen, D., Jia, B., Chen, G., Blasch, E., Pham, K., "Space Based Sensor Management Strategies Based on Informational Uncertainty Pursuit-Evasion Games," *IEEE Nat. Aerospace and Electronics Conf.*, 2015.

[43] Shen, D., Jia, B., Chen, G., Blasch, E., Pham, K., (2016) Pursuit-evasion game theoretic uncertainty oriented sensor management for elusive space objects. *2016 IEEE National Aerospace and Electronics Conference (NAECON) and Ohio Innovation Summit (OIS)*, 156-163.

[44] Jia, B., Pham, K.D., Blasch, E., Shen, D., Wang, Z., and Chen, G., "Cooperative space object tracking using consensus-based filters," *Int'l Conf. on Information Fusion*, 2014.

[45] Jia, B., Pham, K.D., Blasch, E., Shen, D., Wang, Z., Tian, X., and Chen, G., "Information Weighted Consensus-based Cooperative Space Object Tracking to overcome Malfunctioned Sensors and Noisy Links," *Int'l Conf. on Information Fusion*, 2015.

[46] Jia, B., Pham, K.D., Blasch, E., Shen, D., Wang, Z., and Chen, G., "Cooperative Space Object Tracking using Space-based Optical Sensors via Consensus-based Filters," *IEEE Transactions on Aerospace and Electronics Systems*, 2016.

[47] Samarasinghe, S.(2007) Neural networks for applied sciences andengineering: From fundamentals to complex pattern recognition. Boca Raton, FL: Auerbach.

[48] Amoozegar, F., Sundareshan, M.K., "Adaptive Kalman filter implementation by a neural network scheme for tracking maneuvering targets", *Proc. SPIE 2485, Automatic Object Recognition V*, (5 July 1995); doi: 10.1117/12.213077

[49] Shabarekh, C. Kent-Bryant, J, Garbus, M., Keselman, G., Baldwin, J. and B. Engberg (2016). Efficient Object Maneuver Characterization to Support Space Situational Awareness. In proceedings of the Technical Track at the 32nd Space Symposium, Colorado Springs, CO.

[50] Shabarekh, C., Kent-Bryant, J., Keselman, G., & Mitidis, A. A Novel Method for Satellite Maneuver Prediction. *Advanced Maui Optical and Space Surveillance Technologies Conference*, 2016.

[51] Balcan, M.F., Blum, A., and Vempala, S., Clustering via Similarity Functions: Theoretical Foundations and Algorithms. *In Proceedings of the 40th ACM Symposium on Theory of Computing (STOC)*, 2008.

[52] Linares, R., Furfaro, R., Dynamic Sensor Tasking for Space Situational Awareness via Reinforcement Learning, *Advanced Maui Optical and Space Surveillance Technologies Conference*, Maui, HI, Sept. 2016.

[53] Linares, R., Furfaro, R., Space Objects Maneuvering Detection and Prediction via Inverse Reinforcement Learning, *Advanced Maui Optical and Space Surveillance Technologies Conference*, Maui, HI, Sept. 2017.

[54] Abbeel, P. and Ng, A.Y., "Apprenticeship learning via inverse reinforcement learning," *Proceedings of the twenty-first international conference on Machine learning*, ACM, 2004, p. 1.

[55] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G., et al. "Human-level control through deep reinforcement learning". In: Nature 518.7540 (2015), pp. 529–533.

[56] Mnih, V., Badia, A.P., Mirza, M., Graves, A., Lillicrap, T.P., Harley, T., Silver, D., and Kavukcuoglu, K. "Asynchronous methods for deep reinforcement learning". In: arXiv preprint arXiv:1602.01783 (2016).

[57] Das-Stuart, A., Howell, K. C., and Folta, D. C., "A Rapid Trajectory Design Strategy for Complex Environments Leveraging Attainable Regions and Low-Thrust Capabilities," 68th International Astronautical Congress, Adelaide, Australia, 2017, pp. 1–19.

[58] Das-Stuart, A., Howell, K. C., and Folta, D. C., "Rapid Trajectory Design in Complex Environments Enabled by Reinforcement Learning and Graph Search Strategies," Acta Astronautica, Available Online April 25, 2019. https://doi.org/https://dx.doi.org/10.1016/j.actaastro.2019.04.037.

[59] Das-Stuart, A., and Howell, K., "Contingency Planning in Complex Dynamical Environments via Heuristically Accelerated Reinforcement Learning," AAS/AIAA Astrodynamics Specialist Conference, American Astronautical Society, Portland, Maine, 2019, pp. 1–21.

[60] Broida, J., and Linares, R., "Spacecraft Rendezvous Guidance in Cluttered Environments via Reinforcement Learning," 29th AAS/AIAA Space Flight Mechanics Meeting, American Astronautical Society, Ka'anapali, Hawaii, 2019, pp. 1–15.

[61] Gaudet, B., Linares, R., and Furfaro, R., "Integrated Guidance and Control for Pinpoint Mars Landing Using Reinforcement Learning," Proceedings of the AAS/AIAA Astrodynamics Specialist Conference, AAS Paper 18-290, Snowbird, UT, Aug. 2018.

[62] Guzzetti, D., "Reinforcement Learning And Topology Of Orbit Manifolds For Station-keeping Of Unstable Symmetric Periodic Orbits," AAS/AIAA Astrodynamics Specialist Conference, American Astronautical Society, Portland, Maine, 2019.

[63] Furfaro, R., and Linares, R., "Waypoint-Based generalized ZEM/ZEV feedback guidance for planetary landing via a reinforcement learning approach," 3rd International Academy of Astronautics Conference on Dynamics and Control of Space Systems, DyCoSS, Univelt Inc., Moscow, Russia, 2017, pp. 401–416.

[64] Miller, D., and Linares, R., "Low-Thrust Optimal Control via Reinforcement Learning," 29th AAS/AIAA Space Flight Mechanics Meeting, American Astronautical Society, Ka'anapali, Hawaii, 2019.

[65] Miller, D., Englander, J. A., and Linares, R., "Interplanetary Low-Thrust Design Using Proximal Policy Optimization," AAS/AIAA Astrodynamics Specialist Conference, American Astronautical Society, Portland, Maine, 2019.

[66] LaFarge, N., Miller, D., Howell, K., and Linares, R., "Guidance for Closed-Loop Transfers using Reinforcement Learning with Application to Libration Point Orbits," AIAA SciTech, Orlando, FL, 2020.

[67] Gaudet, B., Linares, R., and Furfaro, R., "Seeker-based Adaptive Guidance via Reinforcement Meta-learning Applied to Asteroid Close Proximity Operations," AAS/AIAA Astrodynamics Specialist Conference, American Astronautical Society, Portland, Maine, 2019.

[68] Fu, J., Lou, K., and Levine, S., "Learning Robust Rewards with Adversarial Inverse Reinforcement Learning." CoRR abs/1710.11248 (2017)

[69] "Maneuver Types." *Satellite Safety*, NASA GSFC, `satellitesafety.gsfc.nasa.gov/maneuvers.html`. April 2018.

[70] Mckinley, D.P., "Maneuver Planning for Conjunction Risk Mitigation with Groundtrack Control Requirements". 18th AAS/AIAA Space Flight Mechanics Meeting Proceedings. 27-31 January 2008.

[71] Gooding, R. H. (1990). A Procedure for the Solution of Lambert's Orbital Boundary-Value Problem. *Celestial Mechanics and Dynamical Astronomy*. 48(2): 145-165. [6]

[72] Henderson, T.A., Mortari, D., Davis, J., *Modifications to the Gooding Algorithm for Angles-Only Initial Orbit Determination*. AAS 10-238, AAS/AIAA Space Flight Mechanics Meeting, San Diego (2010)

[73] Tapley, B., Schutz, B., and Born, G. H., *Statistical Orbit Determination*, Elsevier, Burlington, MA, 2004.

[74] Komendera, E., Scheeres, D., and Bradley, E., "Intelligent Computation of Reachability Sets for Space Missions," *IAAI-12 (Proceedings of the 24th Conference on Innovative Applications of Artificial Intelligence)*, Toronto; July 2012.

[75] Xue, D., Li, J., Baoyin, H., and Jiang, F., "Reachable Domain for Spacecraft with a Single Impulse", *Journal of Guidance, Control, and Dynamics*, Vol. 33, No. 3 (2010), pp. 934-942.

[76] Co, T.C., Black, J.T., and Zagaris, C., "Responsive Satellites Through Ground Track Manipulation Using Existing Technology", *Journal of Spacecraft and Rockets*, Vol. 50, No. 1 (2013), pp. 206-216.

[77] Z.J. Hall and P. Singla. "Higher Order Polynomial Expansion for Calculating Satellite Reachability Sets." Proceedings of the AAS/AIAA Space Flight Mechanics Meeting, AAS Paper 19-437, Ka'anapali, Maui, Hawaii, Jan. 2019.

[78] Adurthi, N., Singla, P., and Singh, T. "Conjugate unscented transform rules for uniform probability density functions." Proceedings of the American Control Conference, June 2013, pp. 2454-2459.

[79] Adurthi, N., and Singla, P. "A conjugate unscented transformation based approach for accurate conjunction analysis." AIAA Journal of Guidance, Control and Dynamics, Vol. 38, No. 9 (2015), pp. 1642-1658.

[80] Julier, S.J., Uhlmann, J.K., and Durrant-Whyte, H.F. "A new method for the nonlinear transformation of means and covariances in filters and estimators." IEEE Transactions on Automatic Control, Vol. 45, No.3 (2000), pp. 477-482.

[81] Tedrake, R. "Underactuated Robotics: Algorithms for Walking, Running, Swimming, Flying, and Manipulation." Online Course Notes. MIT Computer Science and Artificial Intelligence Lab

[82] "Find minimum of constrained nonlinear multivariable function," fmincon, Software Package, Ver. R2018a,Mathworks,Natick, MA, March 2018.

[83] "Choosing the algorithm," Software Package, Ver. R2018a,Mathworks,Natick, MA, March 2018.

[84] Nocedal, J., Wright, S.J. Numerical Optimization. Springer, 2006.

[85] Reiter, J.A., Spencer, D.B., and Linares, R. "Spacecraft Detection Avoidance Maneuver Optimization Using Reinforcement Learning," AAS 19-506, Advances in the Astronautical Sciences, Vol. 168 (2019).

[86] Reiter, J.A., Spencer, D.B., and Linares, R. "Spacecraft Maneuver Strategy Optimization for Detection Avoidance Using Reinforcement Learning," AAS 19-751, AAS/AIAA Astrodynamics Specialist Conference, Portland, ME August 11-15, 2019.

[87] Reiter, J.A., Spencer, D.B., and Linares, R. "Spacecraft Stealth Through Orbit-Perturbing Maneuvers Using Reinforcement Learning," Pre-Print, AIAA/AAS Space Flight Mechanics Meeting, Orlando, FL, January 6-10, 2020.

[88] Reiter, J.A. and Spencer, D.B. "Optimization in Space-Based Pursuit-Evasion Games Through Competitive Coevolution," AAS 19-750, AAS/AIAA Astrodynamics Specialist Conference, Portland, ME August 11-15, 2019.

[89] Coady, P., "Proximal Policy Optimization with Generalized Advantage Estimation." URL https://github.com/pat-coady/trpo

Vita

Jason A. Reiter

Jason Alexander Reiter was born in Hinsdale, Illinois in 1991 but moved a number of times before settling down in Camas, Washington in 2003. After graduating from Camas High School in 2009, he earned a Bachelor of Science degree in Aerospace Engineering from the California Polytechnic State University (Cal Poly) in San Luis Obispo, California, with a minor in Physics, in 2014. After graduating from Cal Poly, Jason worked for NASA's Jet Propulsion Lab, where his work on optimizing solar electric propulsion transfers inspired his first research paper. The following fall, he began his Master's degree work at Penn State under Dr. Spencer, funded by a research assistantship position with the Penn State Applied Research Lab under Dr. Reichard.

The following two summers, Jason worked for the Aerospace Corporation's Mission Analysis and Operations Division in Chantilly, Virginia and then for Applied Defense Solutions in Columbia, Maryland. In August 2016, he began his PhD research, funded by the Department of Defense's Science Math and Research for Transformation (SMART) fellowship. Since then, his summers have been spent working alongside his sponsoring facility in Albuquerque, New Mexico and other Air Force offices. This led to a part-time job with the Air Force Research Lab as well.

Meanwhile, Jason has served as the student program manager of the Astrodynamics Research Group of Penn State (ARGoPS) where he led multiples team projects and competitions, including the Global Trajectory Optimization Competition, and hosted multiple research symposia. His research interests include space domain awareness, trajectory optimization, game theory, and operations analysis.