Research Project in Mechatronics Engineering

Final Research Report

Enhancing Classical Control Laws with Reinforcement Learning

K'vaan Valabh

Project Report ME038-2023

Co-worker: Sumukha Viswakarma

Supervisor: Dr Roberto Armellin
Co-supervisor: Dr Harry Holt

13 October 2023

ENGINEERING
DEPARTMENT OF MECHANICAL
AND MECHATRONICS ENGINEERING

ME038-2023

ENHANCING CLASSICAL CONTROL LAWS WITH REINFORCEMENT LEARNING

K'vaan Valabh

ABSTRACT

Improving control systems in the space industry is a paramount concern due to the substantial financial and safety risks associated with failures and sub-optimal performance. One avenue of research aimed at addressing these challenges involves the autonomous selection of control parameters using machine learning algorithms. Reinforcement learning, in conjunction with Lyapunov-based controllers, has emerged as a promising approach that combines stability with performance optimisation. While reinforced controllers have been developed for various spacecraft dynamics contexts, there exists a noticeable gap in research regarding their application in spacecraft formation flying.

This report presents the design, implementation, and analysis of a state-dependent reinforced Lyapunov controller, with the goal of enhancing fuel efficiency. It covers the design of a classical Lyapunov control law, a Reinforcement Learning framework, their integration, and an analysis employing a custom simulation model.

The state-dependent reinforced Lyapunov controller demonstrated its superiority by achieving up to a 72% lower ΔV compared to the classical Lyapunov controller. It is important to note, however, that the presented results represent optimal simulations. Inconsistencies and unreliability emerged during the model's training, emphasising the need for further investigation to attain consistent and reliable outcomes. While it has been demonstrated that reinforcement learning can improve classical controllers, there remains ample room for research and development to ensure consistent and robust results.

DECLARATION

Student

I ……K'vaan Valabh……. hereby declare that:

1. This report is the result of the final year project work carried out by my project partner (see cover page) and I under the guidance of our supervisor (see cover page) in the 2023 academic year at the Department of Mechanical and Mechatronics Engineering, Faculty of Engineering, University of Auckland.
2. This report is not the outcome of work done previously.
3. This report is not the outcome of work done in collaboration, except that with a project sponsor as stated in the text.
4. This report is not the same as any report, thesis, conference article or journal paper, or any other publication or unpublished work in any format.

In the case of a continuing project: State clearly what has been developed during the project and what was available from previous year(s):

Signature: 

Date: 11/10/2023

Supervisor

I confirm that the project work undertaken by this student in the 2023 academic year **is not** (*strikethrough as appropriate*) part of a continuing project, components of which have been completed previously.

Comments, if any:

Signature: 

Date: 11/10/2023

Table of Contents

List of Figures

List of Tables

Acknowledgements

Thank you Roberto Armellin and Harry Holt for your support and guidance in this research endeavour. Coming into this project with no experience or knowledge in spacecraft dynamics and control systems was frightening. However your wisdom and great explanation skills helped me through this immensely. I would also like to give thanks to my partner Sumukha Viswakarma for his excellent software skills and mathematical knowledge.

Abbreviations

RL	Reinforcement Learning
PSO	Particle Swarm Optimisation
CW	Clohessy-Wiltshire
A2C	Advantage Actor Critic
PPO	Proximal Policy Optimisation
IPCS	Inverted Pendulum Cart System
GTO-GEO	Geostationary Transfer Orbit to Geostationary Orbit
LEO-GEO	Low Earth Orbit to Geostationary Orbit

1. Introduction

Classical control systems have traditionally been implemented to govern spacecraft manoeuvres. However, these systems' performance is often limited by their inability to cater towards dynamic environments and adapt to changing states using their static control policies. These limitations are of great concern in the space industry where the consequences of unreliable or unsatisfactory control systems can have detrimental effects on human safety, monetary resources, and mission objectives. Therefore there is a growing need and interest in exploring new methods to enhance and optimise the performance of these existing systems.

To address and overcome these limitations, this research aims to explore pre-existing classical control laws, namely Lyapunov control, and utilise the benefits of optimisation algorithms, specifically reinforcement learning, to enhance performance in the context of space formation flying. Comparisons between the non-optimised and optimised systems will be investigated and used to inform future work both within the project's scope and in a broader context.

Try get some reference for first paragraph

2. Literature Review

2.1 Lyapunov Control Law

Lyapunov control is based on the Lyapunov stability theory, which assess the stability characteristics of nonlinear dynamical systems through the use of a Lyapunov function, $V(x)$. This scalar, non-unique, energy-like function is selected to guarantee stability and used to derive corresponding control efforts [3]. The Lyapunov function can be illustrated as a 'bowl shaped' function over a state space (put fig ref and and put ref in caption). If the slope of $V(x)$ is non-positive at any state within the state space, the nonlinear dynamical system is stable about the origin, in other words, there will always exist a control effort to move the state towards equilibrium.

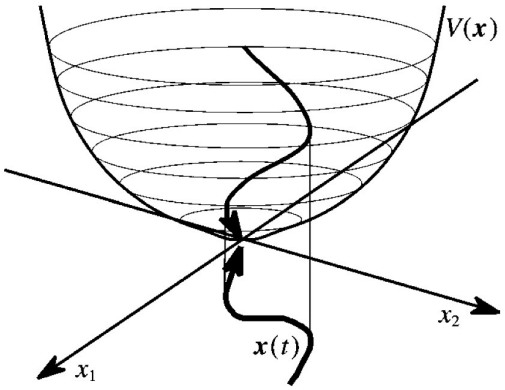

Figure 1 Illustration of a Lyapunov function taken from [1]

This can be conveyed through a set of mathematical constraints. [1]'s candidate Lyapunov function for a linear spring-mass-damper system will be used for explanatory analysis.

$$V = X^T[P]X \tag{1}$$

$$\dot{V} = -X^T[R]X \tag{2}$$

Here, X is the state and $[P]$ and $[R]$ are the matrices that serve as the weightings that determine the response of the control system, which can be tuned for optimised performance. To be considered a Lyapunov function, the candidate function must be positive definite, and its derivative must be negative semi-definite. It's worth noting that only a negative definite derivative guarantees asymptotic stability. Therefore, to meet these requirements, both $[P]$ and $[R]$ must be positive definite matrices. The idea is to select or form a candidate Lyapunov function that complies with these constraints for the given dynamical system, with the aim of ensuring stability.

2.1.1 Strengths and Limitations

Lyapunov stability guarantees that the system's state will remain bounded near the equilibrium point, providing robustness against disturbances and uncertainties, as exemplified in [4] where a Lyapunov-based controller was developed for the stabilisation of the IPCS. This attribute is of great significance in the space industry, where disturbances and uncertainties are prevalent, and risk mitigation is of utmost importance. Furthermore, it holds value in terms of design and implementation, as it eliminates the need for analytical solutions to nonlinear differential equations [1], which are often highly complex and challenging to obtain.

Nevertheless, this stability guarantee comes with some costs. For systems with complex dynamics, finding and deriving an appropriate Lyapunov function can be challenging [1]. More importantly, Lyapunov-based control laws lack robust optimal performance. Their performance depends on user-defined weighting heuristics, often resulting in sub-optimal control [3]. In the context of spacecraft formation flying, where constraints on time, fuel, and accuracy are stringent, achieving optimal performance is paramount. Therefore, when using a Lyapunov-based law in spacecraft control, addressing its suboptimality through the use of a subsidiary optimisation system is essential.

2.2 Reinforcement Learning

Reinforcement learning (RL) is a branch of machine learning, where an agent learns to make decisions based on its feedback of interaction with the environment through actions (as illustrated in Figure 2). The feedback is in terms of rewards for desirable behaviour and penalties for undesirable behaviour. The goal of RL is to find an optimal policy that maximises the received rewards over one episode [5]. The policy consists of a path taken through a neural network containing state-action and action-action neurons. Each neuron is parameterized by a trainable weight [5]. These weights are dynamically adjusted through feedback, where the weights are strengthened for rewards, and weakened for penalties, resulting in an enhanced policy. An optimal policy results in optimal action being taken in a given state. RL is distinct from other branches of Machine learning as the agent receives

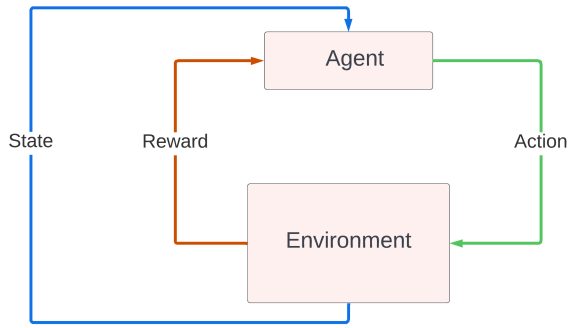

Figure 2 Illustration of a RL process. Based on [2]

.

feedback from the environment, unlike Supervised and Unsupervised Learning where set input data is given to learn off before application.

RL has been widely applied to optimise control parameters in various control systems. For example, [6] improved the robustness of an LQR controller using policy gradient methods. Similarly, in [7], RL was employed to improve a stable, yet sub-optimal, Lyapunov-based controller for the speed control of an ultrasonic motor. The combination of Lyapunov control with RL is a common approach, aiming to balance stability and control performance effectively.

2.3 Previous Work

More specific and relevant research has been undertaken, such as [3]. This research looks into spacecraft trajectory design using Lyapunov control laws and RL to develop a reinforced Lyapunov controller. It aims to assess the performance of the reinforced controller in comparison with classical controllers in terms of robustness, elapsed time, and propellant mass used. With similar motivations as [6] and [7], it aims to merge Lyapunov control laws with RL to combine the benefits and eliminate the drawbacks of both. It aims to achieve this via the implementation of state-dependency. The Reinforced controller was tested and assessed against previous literature in the context of Geostationary Transfer Orbit to Geostationary Orbit (GTO-GEO) and Low Earth Orbit to Geostationary Orbit (LEO-GEO) using Keplerian dynamics. An initial comparison between Particle Swarm Optimisation (PSO) for a state-independent system, and PSO used for a state-dependent system, demonstrated varying weights throughout the transfer can improve optimality. RL was then used to optimise the weights of the control law throughout the transfer. It was found that RL enhanced the controller's performance greatly. For example, for the mass-optimal simulations, the optimal solution can save 31.05 kg (14.0%) at a cost of 5.65 days (3.9%). [3] clearly illustrates the effectiveness of the implementation of RL into Lyapunov control systems in the context of space dynamics. It achieves a pairing of stability and high performance which is a necessity in the space industry.

Overall, the field of control systems with RL implementation has been widely studied. RL has been used to combat the limitations of classical control systems [7], including in the context of space dynamics [9]. LQR and Lyapunov-based controllers have also

been implemented as space control systems [7]. However, gaps, specific to this project, exist in existing literature where implementations and concepts have not been conducted thoroughly. One of these gaps is the application of reinforced Lyapunov controllers on space formation flying. The comparison of numerical simulations of classical controllers and reinforced controllers both undertaken in the same research project has not been demonstrated significantly either. Applying a reinforced Lyapunov controller to a unique context and comparing an LQR and Lyapunov-based controller with RL on the same problem will hopefully lead to new questions and avenues of further research.

3. Objectives and Scope

Overall, the objectives of this research are:

- Explore the principles and applications of Lyapunov control law in control systems engineering.
- Design and Implement a Lyapunov-based controller on a simple relative spacecraft dynamics model.
- Develop and integrate an RL framework to enhance the Lyapunov-based controller's performance.
- Compare state-independent and state-dependent simulation results, specifically the performance of the classical controller compared to the reinforced controller in terms of the given metrics.

This research will remain focused on the context of spacecraft formation flying, with fuel consumption efficiency and time constraints as the primary performance metrics. The scope of this study also includes the implementation of a simulation model in which the controllers can be assessed. It is also worth noting that the dynamics, control theory, and RL implementation required in this project have not been taught in the undergraduate curriculum at the University of Auckland. Therefore, much of the students' efforts and time went into background learning and research, which may not be reflected in this report.

4. Control Law Design

4.1 Spacecraft Dynamics

Although this research aims to investigate the enhancement of Lyapunov-based control law using reinforcement learning within the domain of spacecraft formation flying, as discussed earlier, further context needs to be established to provide a clear and comprehensive framework for the study. This additional context will serve as the basis for defining the scenario that will be simulated and optimised, enabling meaningful comparisons between the classical and reinforced controllers.

4.1.1 Spacecraft Formation Flying

Spacecraft formation flying is described as the tracking or maintenance of a desired relative separation or position between spacecraft [8]. In the context of this project, we will focus on spacecraft rendezvous, where the desired relative separation between two spacecraft is zero. This can be depicted in Figure 3, which makes use of the chaser-target notation. In a rendezvous, the chaser spacecraft approaches the target spacecraft until there is no longer relative separation.

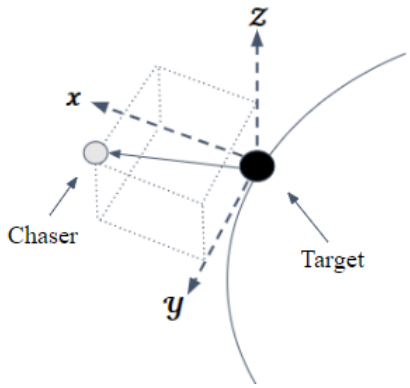

Figure 3 Illustration of a Chaser-Target System

4.1.2 Clohessy-Wiltshire Equations

The Clohessy-Wiltshire equations, commonly referred to as the CW equations, were selected due to their inherent simplicity and their suitability for modeling relative dynamics. As the project focuses on formation-flying scenarios (i.e. rendezvous), it is essential to consider relative dynamics exclusively. The advantage of dealing with a circular orbit lies in the fact that complex dynamics are unnecessary for this specific comparison project. The equations are as follows:

$$\ddot{x} = 3n^2x + 2n\dot{y} \tag{3}$$

$$\ddot{y} = -2n\dot{x} \tag{4}$$

$$\ddot{x} = -n^2z \tag{5}$$

With n being orbital mean motion: Where the x-axis is along the radial vector and the z-axis is along the angular momentum vector of the target spacecraft, with the y-axis completing the right-handed system. [9]. With n being orbital mean motion:

$$n = \sqrt{\frac{\mu}{a^3}} \tag{6}$$

4.1.3 Spacecraft Parameters and Initial Conditions

The values for gravitational acceleration and the gravitational parameter are considered universally accepted constants in the field. On the other hand, the selection of the semi-major axis and the spacecraft's initial conditions were selected from a high-fidelity formation flying simulation in [8], thus ensuring suitability and relevancy for this projects simulations.

These parameters, as well as the spacecraft's initial position, velocity, and mass are defined below:

Orbital Parameter	Value
Gravitational Acceleration, g_0	9.81 $\mathrm{m\,s^{-2}}$
Gravitational Parameter, μ	3.986 x 10^5 $\mathrm{km^3\,s^{-2}}$
Semi-Major Axis of Orbit, a	6900 km

Table 1 Selected Orbital Parameters for Simulation.

$$X_0 = \begin{bmatrix} x \\ y \\ z \\ \dot{x} \\ \dot{y} \\ \dot{z} \\ m_0 \end{bmatrix} = \begin{bmatrix} 250 \ \mathrm{m} \\ 0 \ \mathrm{m} \\ 500 \ \mathrm{m} \\ 0 \ \mathrm{m\,s^{-1}} \\ -0.403 \ \mathrm{m\,s^{-1}} \\ 0 \ \mathrm{m\,s^{-1}} \\ 900 \ \mathrm{kg} \end{bmatrix} \tag{7}$$

Due to spacecraft propellant's high cost resulting from its finite availability, fuel consumption has been selected to be optimised. It will be indicated by ΔV, a metric proportional to the thrust per unit mass required by spacecraft for manoeuvres [10]. Once control effort is implemented, the spacecraft will experience mass loss throughout the manoeuvre. This rate of change can be realised through (8). From the calculation of this mass change over the rendezvous period, ΔV (9) can be found.

$$\dot{m} = -\frac{u\,m}{Isp\,g_0} \tag{8}$$

$$\Delta V = Isp\,g_0 \ln \frac{m_0}{m} \tag{9}$$

Where Isp is the spacecraft-specific parameter, specific impulse. The success of a meaningful simulation is dependent on its verisimilitude. Therefore the selection of realistic parameters and constraints is crucial. Thus the Isp was chosen to be 1000s. In conjunction with these requirements, thrusters have restricted abilities, thus a maximum control effort (U_{MAX}) of $0.0022\,\mathrm{m\,s^{-2}}$ was selected from the simulation constraints in [8] for a $900\,\mathrm{kg}$ spacecraft. If control effort is calculated to be more than U_{MAX}, the effort would be made to saturate at this maximum value.

Using these newly defined initial conditions and parameters, numerical methods can be used to obtain the uncontrolled and natural relative position and velocity of the spacecraft over time. Here, the ODE45 solver from MATLAB was used to numerically integrate the CW equations to illustrate the natural dynamics over 2 orbital periods.
It is evident that the motion is unbounded and oscillatory with a period equal to that of the orbit, meaning that without control effort, the chase spacecraft will become more distant from the target spacecraft as time goes by. As such, control effort must be made to push the chase spacecraft towards the target in order to achieve the rendezvous.

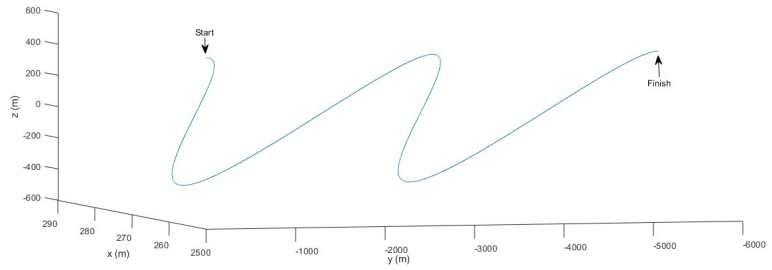

Figure 4 Natural CW Dynamics of an Uncontrolled Simulation

4.2 Lyapunov Control

The process of deriving a Lyapunov control law is inherently iterative, as it necessitates a thorough evaluation of candidate Lyapunov functions to ensure compliance with stability criteria. While multiple iterations were undertaken in this project, this report will exclusively present the selected Lyapunov function.

4.2.1 Lyapunov Function

The stability of a system is dependent on the chosen Lyapunov function. Different Lyapunov functions capture different dynamics for different systems, therefore the selection of an appropriate function is vital in ensuring a system's stability. The chosen Lyapunov function was adapted from [11], where Lyapunov-based control of CW spacecraft dynamics and docking was developed and assessed. As [11] accounted for both translational and rotational control, the provided function and it's derivative had to be modified to exclude the rotational part.

$$V = \frac{1}{2} X^T [K] X \tag{10}$$

$$\dot{V} = -X^T [P] X \tag{11}$$

Where X is the state vector, such as that of the first 6 elements of Equation 7, and $[K]$ and $[P]$ are 6x6 weighting matrices. These are the heuristics that affect the controller's performance and will be tuned, and eventually trained using reinforcement learning.

$$[K] = \begin{bmatrix} A & 0 & 0 & 0 & 0 & 0 \\ 0 & B & 0 & 0 & 0 & 0 \\ 0 & 0 & C & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad [P] = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & D & 0 & 0 \\ 0 & 0 & 0 & 0 & E & 0 \\ 0 & 0 & 0 & 0 & 0 & F \end{bmatrix} \tag{12}$$

Where A:F are tunable element weights. It can be seen that only the top left diagonal for $[K]$ and the bottom right diagonal for $[P]$ are heuristics. $[K]$ and $[P]$ Both must be positive definite, meaning $A : F$ must be positive values. This will ensure that V and $\dot{V}$ are positive semi-definite and negative semi-definite, respectively, thereby guaranteeing the system's stability.

4.2.2 *Lyapunov Control Law*

With the appropriate Lyapunov function now defined, a corresponding control law can be identified. Like the Lyapunov function, the derived control law was also adapted from [11], via the removal of the rotational terms. This control law satisfies the stability requirement for a true lyapunov function.

$$U = -[P]X - [K]X - [A]X \tag{13}$$

Where U is the control effort i.e., the acceleration applied to the spacecraft at each simulation step. As mentioned earlier, this value will be saturated to U_{MAX} if its calculated value exceeds its limit (U_{MAX}). This control effort can then be applied via:

$$\dot{X} = [A]X + [B]U \tag{14}$$

Where X is the current state, $\dot{X}$ is the derivative of the current state, $[A]$ is the CW equations in a 6x6 matrix form, and $[B]$ is a zero-filled 6x6 matrix with a 3x3 identity matrix in the bottom right corner.

$$[A] = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 3n^2 & 0 & 0 & 0 & 2n & 0 \\ 0 & 0 & 0 & -2n & 0 & 0 \\ 0 & 0 & -n^2 & 0 & 0 & 0 \end{bmatrix} \quad [B] = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{15}$$

5. State-independent Simulation

In state-independent (or non-state-dependent) simulations, weights are initially selected based on the system's initial conditions and remain constant throughout the simulation. This section explores the implementation and outcomes of these controlled-dynamics simulations, considering both non-optimised and optimised weight selection.

5.1 Numerical Simulation

The spacecraft's nonlinear controlled-dynamics can be numerically integrated over time to simulate the chase spacecraft's trajectory. This was implemented in Python using the open-source software library, SciPy. The **solve_ivp** function, an initial value problem solver from the SciPy library, was utilised for this numerical simulation. An event function was added to prevent effect of jittering and to minimise unnecessary computational effort after the system has converged to specific thresholds. These thresholds were chosen to be $1\,\mathrm{m}$ for position and $0.1\,\mathrm{ms}^{-1}$ for velocity. Figure 5 illustrates the process of the simulation. Initial conditions and parameters are fed into the numerical solver, which then applies the calculated control effort and integrates the state derivative to find the next state. The process is reiterated for the newly calculated state until either the predetermined number of iterations is reached or the state falls within the specified thresholds, as determined by the events function. This process and implementation serve as the foundation for all subsequent simulations in this report.

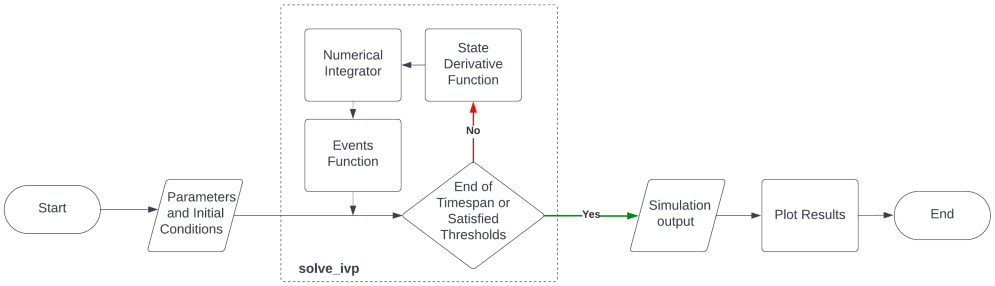

Figure 5 Python Simulation Implementation Flowchart

Simulations for the described initial conditions and mission parameters were undertaken, with both $[K]$ and $[P]$ as non-optimised identity matrices. The results are as follows:

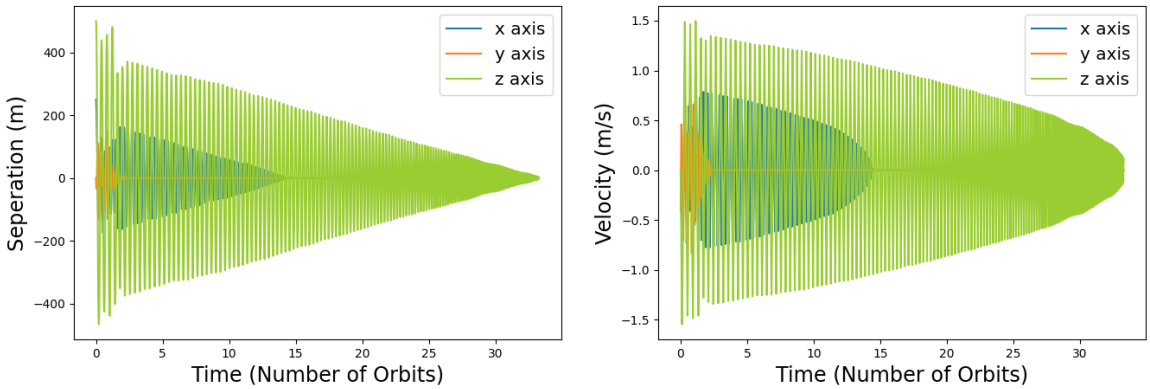

Figure 6 Position (Left) and Velocity (Right) Response of the Non-optimised Simulation

Analysing the results, it's evident that the system converges and stabilises. This convergence is clearly illustrated in Figure 6, where the relative dynamics progressively approach 0. Additionally, in Figure 7, it's notable that the Lyapunov function consistently maintains a positive semi-definite state, while it's derivative consistently remains in a negative semi-definite state.

Although the rendezvous mission was successfully completed, the performance of the control system is sub-par, with a ΔV of 506.58 $\mathrm{m\,s^{-1}}$ and a control time nearly spanning 35 orbits. To enhance efficiency, optimisation of the weighting matrices ($[K]$ and $[P]$) is imperative.

5.2 Particle Swarm Optimisation

In order to obtain a meaningful comparison between the state-independent and state-dependent control systems, it's essential to thoroughly optimise the state-independent controller. This optimisation serves as a baseline, enabling a precise assessment of the improvements attributed primarily to the introduction of state-dependency through reinforcement learning.

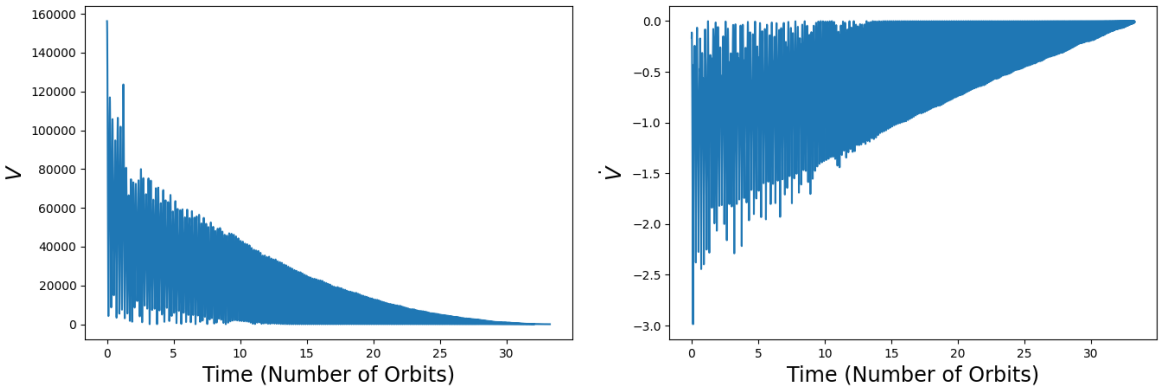

Figure 7 Lyapunov function (Left) and it's Derivative (Right) over Time

Particle Swarm Optimisation (PSO) was chosen to optimise the weighting matrices due to its simplicity and ease of implementation [12], distinguishing it from more complex alternative like Genetic Algorithms and Ant Colony Optimisation. PSO, a computational method for optimisation, works through iterative improvements to candidate solutions based on a given measure of quality [13]. It involves a population of particles moving across a search space, with their movements guided by the best-known local solutions. This collaborative approach drives the swarm towards an optimal solution.

The typical quality measure employed is a cost function, with the algorithm aimed at minimising this function to achieve optimal weighting. The cost function was initially designed with the objective of minimising ΔV. However, considering the project's scope, a successful rendezvous is essential. As a result, provisions were made to discourage non-converging simulation. When simulations did converge, the cost function was straightforward, focusing on the spacecraft's ΔV. However, in cases where convergence did not occur within the specified simulation, a penalty proportional to the separation was introduced. This penalty was deliberately set to be greater than any realistic ΔV, ensuring that the optimal ΔV solution would always drive convergence in the dynamics.

The PySwarms Library was used to implement the ΔV optimisation. PSO was configured to optimise a swarm of 20 particles over 1000 iterations.

A	B	C	D	E	F	ΔV ($\mathrm{m\,s^{-1}}$)	T (orbits)
0.121^2	0.717^2	0.044^2	0.494^2	0.894^2	0.732^2	36.98	2.39

Table 2 Summary of Weightings and Results for PSO

It is evident that PSO alone has a great effect on performance. By examining the cost history displayed in Figure 8, it becomes evident that PSO consistently and iteratively reduces the cost function to its minimum. The fact that this minimum was reached at just over 400 iterations and showed no further improvement afterward serves as an indicator that an excellent optimal solution has been achieved. Comparing Figure 9 and Table 2 to the non-optimised simulation results, PSO decreased ΔV by 92.7% and control time by 91%. While these optimisations are commendable, there is a desire to enhance the system

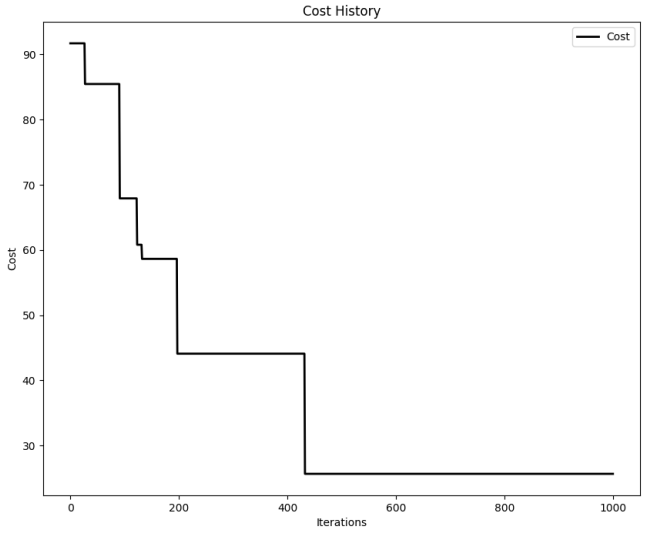

Figure 8 Position (Left) and Velocity (Right) Response of the Non-optimised Simulation

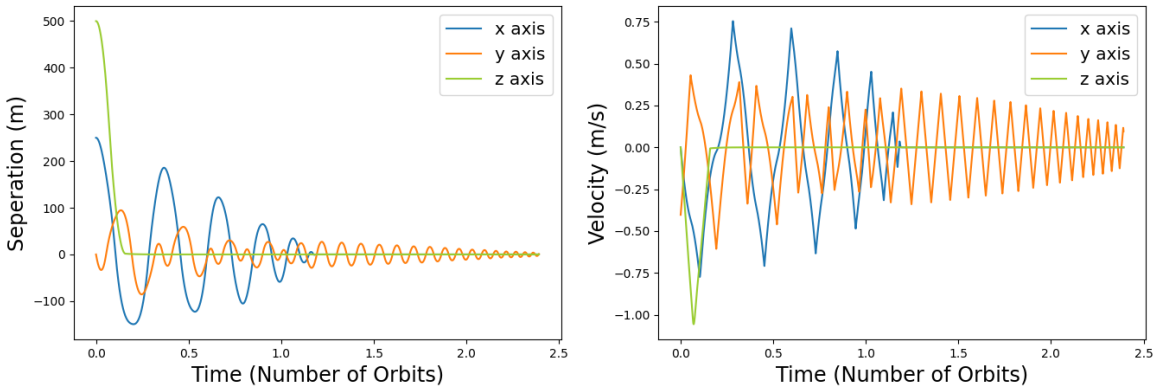

Figure 9 Position (Left) and Velocity (Right) Response of the PSO-optimised Simulation

further by introducing state-dependency.

6. State-dependent Simulation using Reinforcement Learning

In state-dependent simulations. the weights are subject to multiple reselections and updates throughout the simulation, each time based on the current state at the moment of reselection. This section explores the implementation and outcomes of a reinforced Lyapunov controller, a system designed to introduce state dependency into the control system. This entails utilising reinforcement learning to leverage a trained model for selecting optimal weights corresponding to various states during the simulation.

6.1 Reinforcement Learning Framework

To achieve state dependency using reinforcement learning, a context-specific RL framework must be designed. Figure 10 illustrates the overall process of reinforcement learning in this project.

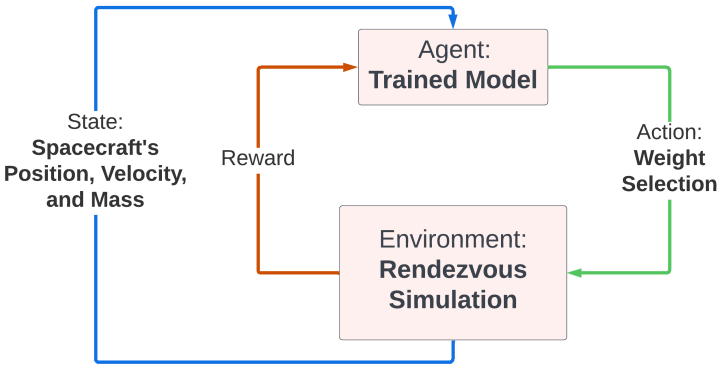

Figure 10 Reinforcement Learning Flowchart

In summary, The trained model uses feedback from the recent weight change and the spacecraft's new current state to select new weights. Figure 11 visualises this concept in greater detail and also provides an overview of the software implementation of the framework. This framework serves as the foundation for training the learning model and conducting the subsequent state-dependent simulations.

Commencing from an initial state, the controlled dynamics are numerically integrated over a designated time period, T_{STEP}. This period represents the frequency at which weight updates occur. The final state resulting from this simulation period is then assessed to determine if the spacecraft has successfully completed the rendezvous, based on thresholds for position and velocity defined earlier. If the spacecraft has achieved convergence, the entire simulation is concluded, and the environment is reset. However, if convergence has not been reached, a reward is computed based on the final state. Additionally, a check is performed to ascertain if the maximum allowable number of episodes has been reached. Here, an episode is defined as the count of times the step function has been called. If the maximum episode limit had been attained, the partially completed rendezvous is documented, and the environment is reset. In cases where the limit has not been reached, the calculated reward, in conjunction with the pre-trained model, is used to update the weights. This iterative process persists until either the chase spacecraft successfully rendezvous with the chief, or the maximum number of episodes is achieved.

This was implemented in Python using **Stable Baselines3 (SB3)**. SB3 is a set of credible implementations of reinforcement learning algorithms in PyTorch [14]. It was selected due to its custom environment implementation simplicity, unified environment structure, and algorithm library. The environment structure comprises three main functions: `init()`, `step()`, and `reset()` (as shown in Figure 11). `init()` initialises the environment, encompassing tasks such as setting the initial state, defining matrices, specifying the action space, and configuring the observation space. `step()` is responsible for conducting

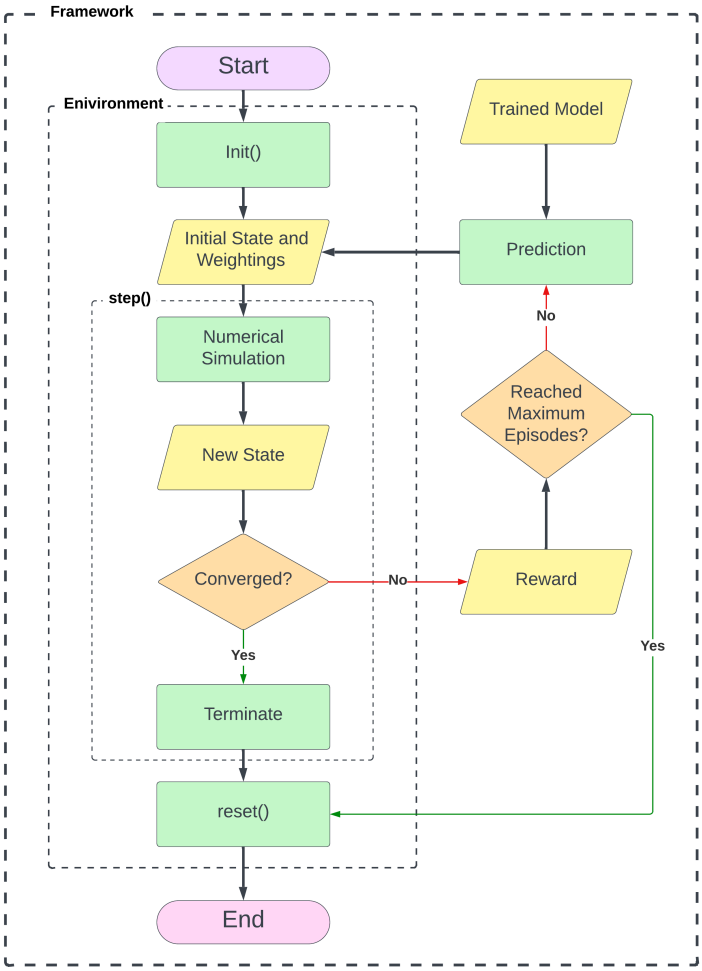

Figure 11 Reinforcement Learning Framework

simulations and conducting rewards and is invoked multiple times throughout the process. `reset()` restores the environment to its initial state.

6.2 Reward Function Design

As the learning of the model, and thus the overall performance of the system is based on the cumulative rewards received through the environment interaction, it is key that the reward function is designed carefully and effectively. The reward function should encourage the desired behaviour, making the agent strive for improvement, and thus optimally. It should also discourage undesired behaviours, helping the model avoid actions that lead to non-optimal outcomes.

For this algorithm, a similar design and theory were applied as in the PSO cost function discussed in Section 5.2. The reward function incorporates three key components: ΔV, position error, and velocity error. If the reward function relied solely on ΔV, it would lack the necessary incentive for convergence. Therefore, we take into account state error if the rendezvous has not been fully completed. This is mathematically depicted in 16 and

17.

$$Reward = -\Delta V \tag{16}$$

$$Reward = -\Delta V - ||Position|| - ||Velocity|| \tag{17}$$

Where the position and velocity components are the norm of the state vector. As the agent's objective is to maximise reward while the aim is to minimise ΔV, the metric has been inverted to represent it as negative. If, after `step()` has simulated for T_{STEP} and the dynamics have not yet converged, the reward is calculated based on (17). However, if the rendezvous is achieved within the T_{STEP} period, the reward is determined by (16), and the simulation concludes. These reward systems are illustrated in Figure 12.

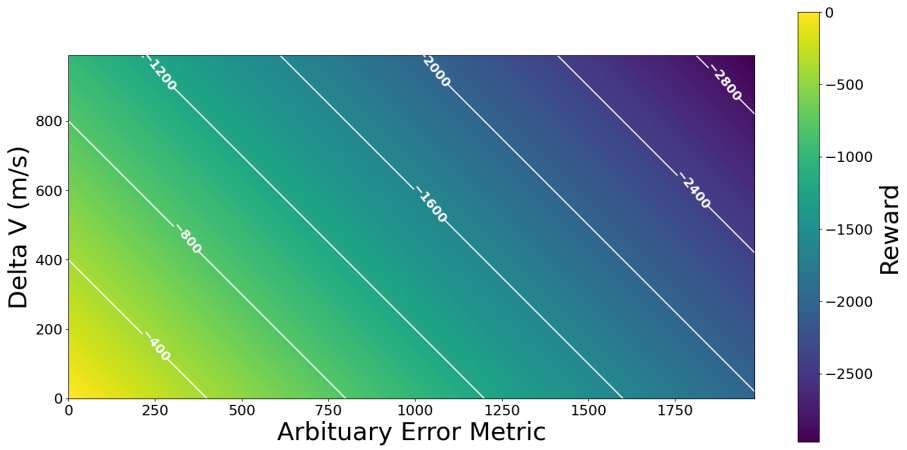

Figure 12 Reward Function Heatmap

The presented reward function demonstrates its effectiveness. A clear positive gradient towards achieving rendezvous with zero fuel consumption is evident and consistent across the reward. Consequently, as long as ΔV and error are non-zero, the agent will continuously have the incentive to reduce both error and ΔV, thus consistently enhancing its optimality.

6.3 Model Training

Now that the RL framework has been designed, the next step is to train the model/agent. In this project, the SB3's **A2C** (Advantage Actor-Critic) algorithm is utilised to train the agent and create a policy for the agent's decision-making process. A2C was chosen for its compatibility with box-type action and observation spaces. PPO (Proximal Policy Optimisation) was another candidate algorithm due to its similar properties. The hyperparameters were kept at default, namely the learning rate = 0.0007 and the discount rate = 0.99.

The training process involves the agent taking action and receiving feedback in the form of rewards. The agent's objective is to maximise cumulative rewards. Through a process of trial and error, the agent gradually learns to seek better rewards and further refines its

policy. This policy will then determine the agent's decisions when using the trained model for prediction.

During the training process, both successes and failures were encountered. An example illustrating a success is presented in Figure 13.

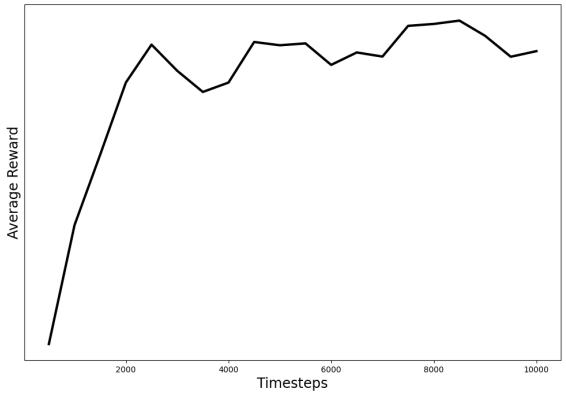

Figure 13 Average Reward Throughout the Learning Period

The training process consisted of 10,000 iterations, with a time step of T_{STEP} = 10000 s. This training can be considered a success, given the evident increase in cumulative rewards over time. The initial phase of the training progressed smoothly, with no declines in rewards. After approximately 2,000 iterations, the model incurred penalties, as indicated by the negative slopes in the graph, and it seemed to oscillate within this high-reward range, searching for ways to further increase its rewards.

However, it's important to note that this outcome was not consistent across all training runs. In some cases, the model experienced an oscillatory pattern at the lower end of the graph, where the rewards were extremely low. In certain instances, the learning curve exhibited chaotic behavior, bouncing between high and low rewards without meaningful learning. Therefore, it is crucial to exercise caution when interpreting these results, as the focus will be on the outcomes of well-performing simulations while those that do not exhibit meaningful learning will be neglected.

get tensor flow working - and train model. hyper parameters

6.4 State-dependent Simulation

The trained models were subsequently employed, together with the rest of the RL framework, to conduct multiple simulations at specified weight update frequencies: 20000 s, 5000 s, 1000 s, and 500 s. A variety of responses were observed for each given frequency. Therefore, only the most ΔV-optimal simulation for each frequency will be presented. The outcomes of these simulations are displayed and summarised. Dashed vertical lines indicate instances of weight changes, with "Weight Change: 0" representing the initial weighting matrices. While the T_{STEP} = 20000 s simulation doesn't strictly fall into the category of state-dependent simulations due to its absence of weight updates, it will serve

as a baseline for the purpose of comparing performance between PSO optimisation and state-independent RL optimisation.

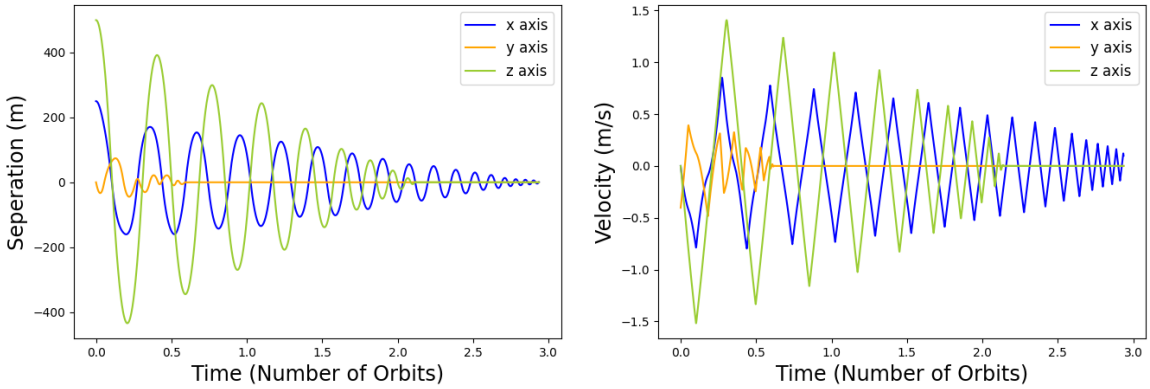

Figure 14 Position (Left) and Velocity (Right) Response of the State-dependent Simulation: T_{STEP} = 20000 s.

Weight Change	A	B	C	D	E	F	ΔV ($\mathrm{m\,s^{-1}}$)	T (Orbits)
0	0.1340^2	0.082^2	0.047^2	0.278^2	0.247^2	0.201^2	51.26	0
Total ΔV							$51.26\ \mathrm{m\,s^{-1}}$	
Total ΔT							2.94 Orbits	

Table 3 Summary of Weightings and Results: T_{STEP} = 20000 s.

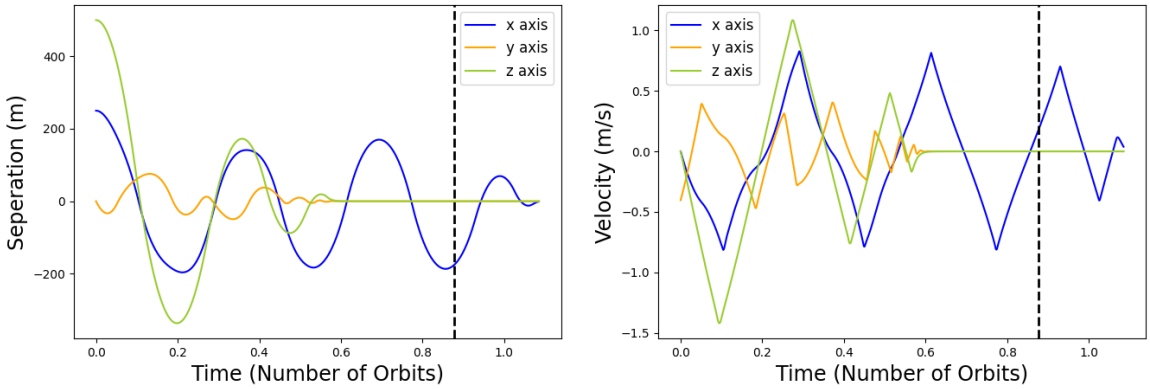

Figure 15 Position (Left) and Velocity (Right) Response of the State-dependent Simulation: T_{STEP} = 5000 s.

Weight Change	A	B	C	D	E	F	ΔV ($\mathrm{m\,s^{-1}}$)	T (Orbits)
0	0.167^2	0.068^2	0.034^2	0.288^2	0.201^2	0.286^2	16.58	0
1	0.024^2	0.226^2	0.244^2	0.195^2	0.177^2	0.177^2	1.08	0.88
Total ΔV							$19.19\ \mathrm{m\,s^{-1}}$	
Total ΔT							1.08 Orbits	

Table 4 Summary of Weightings and Results: T_{STEP} = 5000 s.

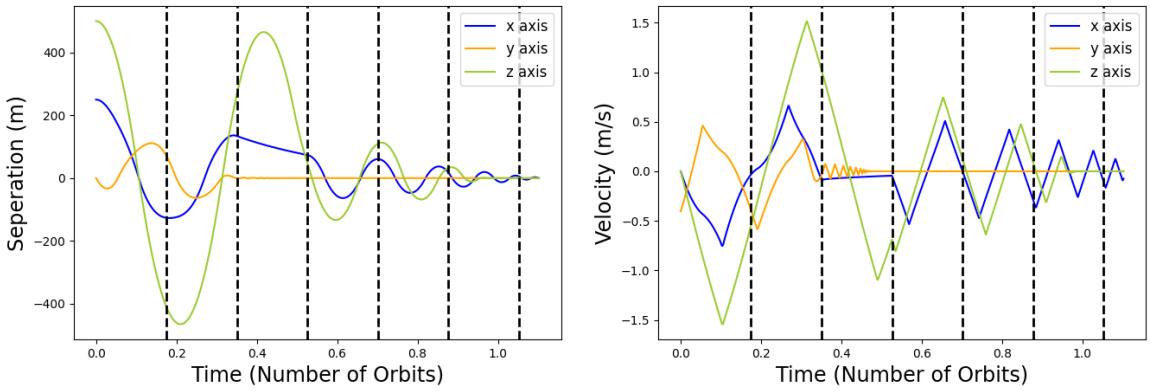

Figure 16 Position (Left) and Velocity (Right) Response of the State-dependent Simulation: T_{STEP} = 1000 s.

Weight Change	A	B	C	D	E	F	ΔV ($\mathrm{m\,s^{-1}}$)	T (Orbits)
0	0.066^2	0.279^2	0.066^2	0.093^2	0.283^2	0.301^2	3.81	0
1	0.203^2	0.0198^2	0.184^2	0.311^2	0.1314^2	0.077^2	3.71	0.18
2	0.003^2	0.206^2	0.016^2	0.134^2	0.216^2	0.231^2	2.76	0.35
3	0.068^2	0.152^2	0.048^2	0.156^2	0.172^2	0.180^2	3.17	0.53
4	0.042^2	0.169^2	0.032^2	0.148^2	0.174^2	0.196^2	3.14	0.70
5	0.041^2	0.167^2	0.031^2	0.144^2	0.170^2	0.193^2	2.59	0.88
6	0.052^2	0.150^2	0.050^2	0.149^2	0.178^2	0.181^2	0.60	1.05
Total ΔV					$19.78\ \mathrm{m\,s^{-1}}$			
Total ΔT					1.10 Orbits			

Table 5 Summary of Weightings and Results: T_{STEP} = 1000 s.

For the T_{STEP} = 500 s simulation, it's worth noting that there were a total of 100 weight changes throughout the rendezvous manoeuvre. However, due to consideration of clarity and the absence of substantial weight changes for the majority of the simulation, only a select few significant weight changes were chosen to be included in the summary.

Weight Change	A	B	C	D	E	F	ΔV ($\mathrm{m\,s^{-1}}$)	T (Orbits)
0	0.232^2	0.201^2	0.219^2	0.083^2	0.244^2	0.193^2	1.90	0
2	0.005^2	0.034^2	0.037^2	0.254^2	0.160^2	0.246^2	1.62	0.26
4	0.024^2	0.006^2	0.002^2	0.269^2	0.149^2	0.243^2	0.37	0.44
6	0.025^2	0.007^2	0.003^2	0.271^2	0.151^2	0.246^2	0.07	0.61
19	0.023^2	0.007^2	0.001^2	0.276^2	0.169^2	0.262^2	0.03	1.75
Total ΔV					$10.19\ \mathrm{m\,s^{-1}}$			
Total ΔT					8.77 Orbits			

Table 6 Summary of Weightings and Results: T_{STEP} = 500 s.

From these results, it is evident that the implementation of the RL framework and the training of the model effectively render the system state-dependent. The degree of weight

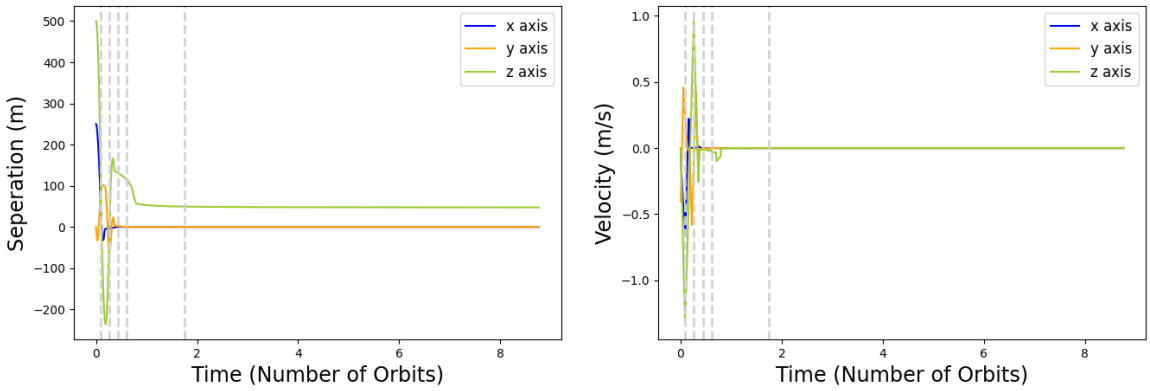

Figure 17 Position (Left) and Velocity (Right) Response of the State-dependent Simulation: T_{STEP} = 500 s.

change appears to be influenced by the frequency of weight updates and thus the dissimilarity between the previous weight change state the new one. This observation becomes evident when comparing Table 4 and 6. For T_{STEP} = 5000 s, there is a significant change in weights, whereas for T_{STEP} = 500 s, the changes are very minimal.It is also evident that different optimisations are achieved for varying numbers of weight updates. From this sample of simulations, T_{STEP} = 500 s result in the lowest ΔV, but the highest ΔT. It is important to note that drawing a meaningful conclusion of which number of updates result in optimal result may not be straightforward. What becomes apparent, however, is that weight updates do lead to improvements from state-independent RL simulations. This observation may suggest that their exists an optimal number of weight changes, although this would also be contingent on other factors such as time constraints, computational costs, and limitations of a real control system.

However, it is essential to exercise caution with analysis because these displayed results were selected based on their optimality. This level of optimality was not found to be guaranteed every time and the trained models were found to be inconsistent and unreliable.

7. Discussion and Comparisons

Optimiser	Number of Weight Changes	ΔV ($\mathrm{m\,s^{-1}}$)	T (orbits)
N/A	0	506.58	35
PSO	0	36.98	2.39
RL	0	51.26	2.94
	1	19.19	1.08
	6	19.78	1.10
	100	10.19	8.77

Table 7 Summary of Results for State-independent and State-dependent Simulations

Comparing the results of the three state-independent simulations, it's not surprising that PSO outperforms the RL model, with a 28% lower ΔV. This difference can be attributed to

how the PSO and Rl simulations were conducted. In this context, PSO can be considered as having 20 agents (20 particles), while RL only has one. This means that for every RL iteration or episode, PSO has 20 different ones. This aligns with similar findings in the comparison for the CartPole problem [15]. It's worth noting that both PSO and RL significantly outperform non-optimised control, reducing ΔV by 93% and 90% respectively.

Shifting the focus to the state-dependent simulations, it becomes evident that state-dependent systems are high-performing. As discussed earlier, drawing meaningful conclusions from the specific weight change frequencies' effect on performance is challenging with such a small data set. Exploring a wider range of frequencies and conducting more thorough simulations would provide more insight into how the number of weight changes is related to optimal performance. With all of this considered, all state-dependent simulations will be viewed holistically.

When comparing classical state-independent systems with state-dependent systems, examination of the summarised results reveals the superiority of state-dependency. ΔV is 46% to 72% lower in the state-dependent systems, and in some cases, ΔT decreases by approximately 1.9 orbits. These findings highlight that the enhancement of classical control laws via reinforcement learning is indeed feasible.

However, to gain more meaningful insights, considering the reliability and consistency of the controller and optimisation techniques is necessary. PSO consistently produces excellent results, with the majority of attempts resulting in weights corresponding to ΔV values under $50\,\mathrm{m\,s^{-1}}$. The same can't be said for the RL-based controller. While RL did produce optimal results, they were selected from a range of non-optimal results that would sometimes exceed a ΔV of $200\,\mathrm{m\,s^{-1}}$. These inconsistencies may be attributed to flaws in the model learning and design process. Specifically, the reward function might have issues in edge cases where the agent's progress gets stuck and cannot learn efficiently. It's also possible that the agent lacks adequate incentive to strive for greater rewards and take penalty-ridden risks. Furthermore, the learning time is a crucial factor to consider. Due to time and resource (CPU) constraints, training runs over 100,000 steps are highly time-consuming and inconvenient. It's suggested that meaningful learning only occurs after 1,000,000 steps [14]. This implies that the policy and agent may not receive enough training to genuinely learn the optimal actions and may rely on luck, as indicated in the results. Lastly, hyperparaameter tuning was not conducted due to time and resource constraints related to the learning process. Parameters such as the learning rate, discount rate, and exploration strategy could be fine-tuned to help the model learn more efficiently and effectively. A longer learning period would reveal the impact of these adjustments.

As a result of limitations in result quality and challenges encountered during the implementation of the reinforced LQR controller (refer to my partner Sumukha Viswakarma's report), it is currently not feasible to draw a meaningful comparison between the performances of the two reinforced controllers.

8. Conclusions

This research designed, simulated, and analysed the effectiveness of a reinforced Lyapunov-based Controller in enhancing the control performance in the context of spacecraft formation flying.

Overall, the conclusions drawn from this research are:

- PSO is superior in producing optimal performance results for state-independent control systems when compared to RL state-independent optimisation.
- State-dependent Lyapunov control systems are superior to state-independent Lyapunov control systems.
- Reinforcement Learning can be used to enhance the performance of classical Lyapunov control law.
- Reward function design is critical in the success of an RL model.
- The training of an RL model is only effective if hyperparameters are tuned correctly, and the model is given a sufficient number of iterations to train on.

9. Suggestion for Future Work

The failures and limitations identified in this research offer numerous avenues for future work, many of which revolve around the continuation and refinement of this research's reinforcement learning framework and RL model. Firstly, it is crucial to reevaluate and optimise the reward function to provide the agent with even stronger incentives to attain optimal rewards. Access to high-performance computing resources, such as NeSI, is recommended to facilitate proper training of the RL model. This entails fine-tuning hyperparameters, especially the learning rate, the discount function, and the exploration strategy, thereby enabling the model to broaden its learning experiences and seek optimality. Extensive learning simulations should be conducted, with iterations reaching the order of 10^6. The comprehensive approach is expected to yield a well-trained model characterised by consistent prediction, learning curves, and the production of optimal results. Moreover, there is an opportunity to delve deeper into the concept of state dependency, specifically in terms of determining the optimal number of weight changes. This can be achieved by running simulations at various frequencies multiple times to identify reliable trends. Once all of this has been refined, exploration into new initial conditions, dynamics, and other research fields can be undertaken.

References

[1] H. Schaub and J. Junkins, *Analytical mechanics of space systems (2nd ed.)*. American Institute of Aeronautics and Astronautics, 2009.

[2] Ayik, F., "Utilizing deep reinforcement learning in control: Unleashing the power of intelligent systems," 2020.

[3] H. Holt, "Trajectory design using lyapunov control laws and reinforcement learning," Ph.D. dissertation, University of Surrey, 2022.

[4] C. Ibanez, O. Frias, and M. Castanon, "Lyapunov-based controller for the inverted pendulum cart system," *Nonlinear Dynamics*, no. 40, pp. 367–374, 2005.

[5] . U. U. Hovell, K., "Deep reinforcement learning for spacecraft proximity operations guidance," *Journal of Spacecraft and Rockets*, no. 2, 2021.

[6] P. . S. T. Gravell, B. Mohajerin Esfahani, "Learning robust controllers for linear quadratic systems with multiplicative noise via policy gradient," 2020.

[7] A. Mustafa, T. Sasamura, and T. Morita, "Robust speed control of ultrasonic motors based on deep reinforcement learning of a lyapunov function," *IEEE Access*, vol. 10, pp. 46 895–46 910, 2022.

[8] K. T. Alfriend, S. R. Vadali, P. Gurfil, J. P. How, and L. S. Breger, *Spacecraft Formation Flying*. Oxford; Massachusetts: Butterworth-Heinemann, 2010.

[9] The University of Texas at Austin, "The clohessy wiltshire model."

[10] Wikipedia contributors, "Delta-v — Wikipedia, the free encyclopedia," 2023, [Online; accessed 19-July-2023]. [Online]. Available: https://en.wikipedia.org/wiki/Delta-v

[11] C. Wenn, "Lyapunov-based control of coupled translational-rotational close-proximity spacecraft dynamics and docking," Master's thesis, The University of Arizona, 2017.

[12] A. Tharwat and W. Schenck, "A conceptual and practical comparison of pso-style optimization algorithms," *Expert Systems with Applications*, vol. 167, p. 114430, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417420310939

[13] Wikipedia contributors, "Particle swarm optimization — Wikipedia, the free encyclopedia," 2023, [Online; accessed 12-October-2023]. [Online]. Available: https://en.wikipedia.org/wiki/Particle_swarm_optimizationv

[14] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann, "Stable-baselines3: Reliable reinforcement learning implementations," *Journal of Machine Learning Research*, vol. 22, no. 268, pp. 1–8, 2021. [Online]. Available: http://jmlr.org/papers/v22/20-1364.html

[15] Price, A., "The cartpole problem - competetive performance with particle swarm optimisation," 2020.