IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

Machine Learning Based Framework For Worst Case Performance Analysis

[1]Lekshmi I S, [2]Nivya B Higgins, [3]Rifa Shereef, [4]Sampoorna Varshinii M K S, [5]Jishy Samuel, [6]Prof. Rehna R S

[1,2.3,4]Student, [5]Division Head PMSD, [6]Assistant Professor

[1,2,3,4,6]Department of Computer Science Engineering, [5]MSSG/CGSE

[1,2,3,4,6]LBS Institute of Technology for Women, Thiruvananthapuram, India

[5]Vikram Sarabhai Space Centre, Thiruvananthapuram, India

Abstract: This work investigates the development and optimisation of reinforcement learning for predicting worst-case scenarios in launch vehicle simulations. Simulations take into account various environmental factors that can affect the launch, including wind conditions, temperature, atmospheric pressure and other parameters. Here we are trying to identify potential failure modes and anomalies that can occur during a rocket launch. Reinforcement learning models are trained using an objective function designed to accurately predict worst-case scenarios during a rocket launch. It also provides valuable insights into the factors contributing to worst-case scenarios, enabling targeted strategies for risk mitigation and system improvement. This approach aims to quantify the impact of individual parameters or their combinations on the predicted worst-case outcome. This paper demonstrates the potential of reinforcement learning in predicting the worst-case scenarios accurately and thereby launch vehicle simulations can be used for verifying the algorithm's robustness. The developed models can inform decision-making and improve the overall resilience and efficiency of space missions by predicting and mitigating worst-case scenarios.

Keywords - Worst-Case Scenarios, Reinforcement Learning, Launch Vehicle Simulations, Environmental factors, Anomalies, Failure modes, Risk Mitigation, Space missions

I. INTRODUCTION

Rocket launches are intricate and costly endeavors fraught with a high rate of failure, making it imperative to enhance efficiency and safety in rocket travel. Simulations play a pivotal role in mitigating costs and identifying potential issues. Specifically, worst-case analysis simulations are crucial for pinpointing risks and anomalies associated with rocket launches. In this context, reinforcement learning emerges as a promising approach to augment simulations for improved guidance and risk assessment.

II.BACKGROUND

This paper explores using reinforcement learning to analyze worst-case scenarios in rocket launches. The idea is to build models that can predict potential failures or performance problems during a rocket's performance from lift-off till satellite separation. These models are trained with an "objective function" that helps them focus on identifying the absolute worst situations. Reinforcement learning offers a significant advantage by providing early warnings of potential issues. This allows launch operators to make informed decisions and take corrective actions during critical phases of the launch, ultimately increasing the mission's success rate. Since rocket launches are expensive and complex, simulations are crucial for testing the system before real-world flight. These simulations involve detailed computer models that consider everything from the rocket itself to environmental factors like wind. The goal of this research is to use reinforcement learning to streamline these simulations, making them faster, cheaper, and more efficient. This would also allow for the development of robust control and guidance for rockets during launch.

This paper goes on to outline the key steps involved in conducting a worst-case analysis for a spacecraft. This includes collecting and preparing data, simulating those scenarios, defining potential worst-case scenarios, and then analyzing the data to develop a reinforcement learning model. Once the model is built, it's important to set thresholds for what constitutes a critical event and then thoroughly test and validate the entire system. The overall goal of this research is to make space missions safer and more reliable by proactively identifying and addressing potential problems using real-time or near-real-time analysis of spacecraft data.

III. OBJECTIVE

This project proposes using reinforcement learning to predict worst-case scenarios during launch vehicle simulations. By formulating a specific objective function, the models will be trained to identify these critical situations. Additionally, this project also aims to identify how individual factors contribute to worst-case outcomes. This will allow engineers to develop targeted strategies to mitigate risks and optimize the launch system for improved performance, safety, and overall mission success.

IV. LITERATURE REVIEW

This review examines recent research advancements in aerospace and technology, focusing on methodologies that enhance predictability, control, and efficiency in various systems. The papers explored here encompass diverse domains including air traffic control [1], rocketry [2], avionics software analysis [3], neural network applications [4], worst-case analysis [5], satellite anomaly detection [6], and reinforcement learning [7, 8].

Crisostomo et al. (2008) propose a methodology that combines worst-case and Monte Carlo methods for accurate aircraft trajectory prediction, addressing uncertainties in real-world scenarios [1]. This approach utilizes a full, non-linear aircraft model and refines predictions with each radar observation. While this method offers advantages like improved accuracy and adaptability to wind effects, challenges remain in estimating aircraft mass during descent.

Guo (2023) presents a simulation program that leverages machine learning to train an AI for rocket control [2]. This approach offers cost-effective testing and optimizes AI control for different rocket phases. However, limitations include the lack of consideration for real-world factors like air resistance and limited training scenario variance.

The Federal Aviation Administration (2023) introduces a "learn-and-extrapolate" methodology that utilizes machine learning to estimate the Worst-Case Execution Time (WCET) of avionics software [3]. This method offers versatility and adaptability; however, its performance can vary across different programs.

Kumar (2021) explores a Deep Neural Network (DNN) approach for early WCET estimation, enabling early insights during system development [4]. While this method offers promise, the resulting WCET predictions can be inaccurate and require further refinement.

Cheng et al. (2021) propose an LSTM-based method for anomaly detection in satellite power systems using telemetry data [5]. This approach demonstrates effectiveness in real-time anomaly detection but would benefit from a comparative analysis with other methods.

Moltafet et al. (2019) examine the timeliness of information in wireless sensor networks using the Age of Information (AoI) metric [6]. This study analyzes worst-case scenarios to understand how various parameters impact AoI. However, the paper would benefit from real-world validation and incorporating more realistic network conditions.

The papers by Kumar et al. [7] and Lillicrap et al. [8] delve into advancements in offline reinforcement learning (RL). Kumar et al. introduce Conservative Q-Learning (CQL) that addresses challenges associated with learning from pre-collected data [7]. CQL exhibits robustness and superior performance compared to existing methods but necessitates further theoretical analyses, particularly with deep neural networks. Lillicrap et al. propose the Deep Deterministic Policy Gradient (DDPG) algorithm that demonstrates effectiveness in learning policies across various environments [8]. However, DDPG requires a large number of training episodes and can be computationally expensive.

The reviewed papers showcase significant advancements in applying innovative techniques to enhance predictability, control, and efficiency in aerospace and technology domains. These methodologies leverage machine learning, worst-case analysis, and non-linear modeling to address complexities in various systems. While challenges and limitations remain in areas like real-world applicability, training requirements, and model accuracy, the research presented here paves the way for further advancements and real-world applications.

V. RESEARCH METHODOLOGY

The project begins with a file perturbation step, where a Python script is employed to modify specific parameter values. These values are changed within a specified range with a particular step size, resulting in the generation of multiple new files. Following the file perturbation, a batch program is used to automate the generation of additional files required for analysis using the launch vehicle trajectory simulator, SITARA. Software for Integrated Trajectory Analysis with Real-Time Applications (SITARA) is a 6D trajectory simulation software that serves as the core foundation for both real-time and non-real-time trajectory simulations for all ISRO launch vehicles, facilitating mission synthesis and analysis. This tool is essential for mission design, as well as the validation of subsystems and the comprehensive verification of avionics systems in these vehicles. The next step involves extracting the required data from the simulation files using a Python script and storing it into a comma-separated values (CSV) file. Finally, a deep Q-learning approach is applied to analyze the data and enhance the model's capabilities for worst-case scenarios. An environment is created to represent the data. An agent interacts with this environment, learning to predict the worst case of the parameter. The agent's Q-values are updated using a neural network model, which is trained over multiple episodes using an epsilon-greedy policy for exploration and exploitation. This trained agent can then be used to predict the values for the worst case of the parameter, providing valuable insights into the system's behavior under extreme conditions.

The objective is to construct a predictive model capable of determining the proportional or percentage influence of individual parameters or their combinations on the identified worst-case outcome, as shown in Fig. 1. The steps involved in this project are as follows:

- perturb the input parameters in the direction of worst and generate the files
- conduct 6-DOF simulation
- acquire the simulation results
- generate appropriate CSV files
- identify the worst-case performance
- analyze and visualize the results

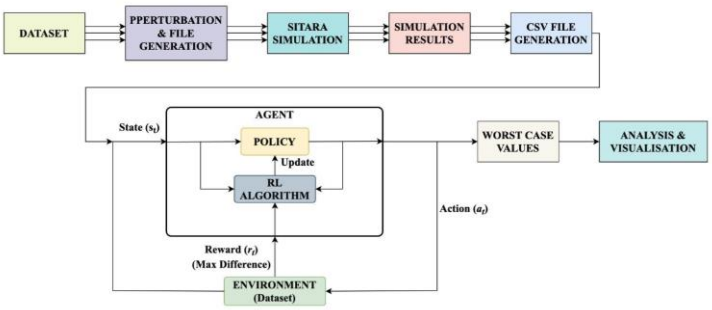

Fig. 1 Flow chart for methodology

5.1 Perturbation

Perturbation of input parameters refers to the deliberate modification or alteration of the values, conditions, or variables used as inputs in a simulation, model, or system. This perturbation is done to observe how the changes in these input parameters affect the outcomes, behavior, or performance of the system being studied. The main goal of perturbation is to assess the system's behavior under different conditions and to analyze its sensitivity to variations in specific parameters.

The information about the launch vehicle is given in a file using which the appropriate target parameters are identified here as thrust misalignment and gyroscope drift parameter, each of them is perturbed from one simulation to another thereby generating the synthetic data that will be used for training.

5.1.1 Thrust Misalignment (TM) - Angle, Value

Variation in angle by 10 in the range of [0, 350] and value by 0.5 in the range of [-2, 2].

5.1.2 Gyroscope Drift in 3 axes - G0X, G0Y, G0Z

These values are changed over the nominal range of 30 to [29.7, 30.3] with a step size of 0.1. Each file represents a unique combination of the perturbed values, while the other two values remain constant.

It is a method to continuously improve previously obtained approximate solutions and handles the problem of nonlinear equations for which exact solutions cannot be obtained. It is used for demonstrating, predicting, and describing phenomena. Hence after perturbation, we get combinatorial perturbed files as the output generating 324 files for parameter one and 21 files for parameter two respectively.

5.2 Simulation

Simulation programs like 6 degrees of freedom (DOF) trajectory are performed on the data that is generated from standalone tests,empirical methods,results of design techniques and the measurements from all the physical systems. The method involves generating a large number of samples from specified ranges for input parameters, running simulations, and using the aggregated results to estimate outcomes or behaviors of interest. Further here we propose to do it intelligently by extending the limits thereby reducing the repeatability.

5.2.1 Thrust Misalignment

Simulation of the files generated in perturbation (run1.1 - run324.1) using SITARA software see Fig. 2. The simulator input files are generated for each of the perturbed files by batch processing the files in the command prompt to generate them for all the perturbed files in a parallel manner to increase the efficiency. Here we have divided the perturbed files into 4 batches and executed the SITARA simulation for all 324 files.

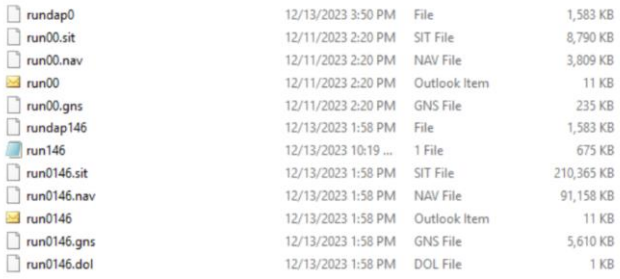

Fig. 2 SITARA files generated for TM (nominal file and file)

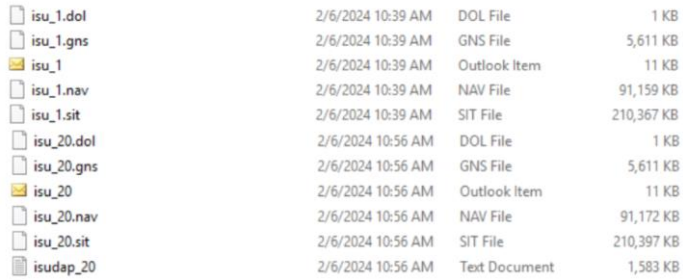

Fig. 3 SITARA files generated for gyroscope drift (1st file and 20th file)

5.2.2 Gyroscope Drift

Following the file perturbation, a batch file is used to automate the generation of additional files required for analysis. We conduct the Sitara files generation of i_1.1 to i_21.1 files and then generate the files, named isu_1.msg to isu_21.msg, as shown in Fig. 3, that are essential for calculating the difference between apogee and perigee values for each perturbed file and the variation in inclination.

5.3 File Generation

In this step we convert the obtained input files from the previous step to their corresponding CSV files, applying the filters to include only the essential range of data from them. This is done by utilizing the functions and modules from pandas.

5.2.1 Thrust Misalignment

The input files [rundap1.txt - rundap324.txt] were processed to extract relevant data within the time range of 120s to 260s, discarding data outside this range. These processed data were then converted to CSV format, resulting in files labeled as "r1.csv" to "r324.csv," each containing approximately 7000 rows representing the time, C1, C2 and parameter values. Then the difference between the C1, C2 of the corresponding r files is calculated and the maximum difference from each of the files is then stored in a CSV file along with its corresponding values of the input parameters.

5.2.2 Gyroscope Drift

This step involves extracting the perigee and apogee values from the “isudap” files [isudap_1.txt - isudap_21.txt] using a Python script. For each file, the difference between the apogee and perigee values is calculated and stored in a CSV file along with the corresponding perturbed values of G0X, G0Y, and G0Z.

5.4 Training

The complete dataset is divided into train and test datasets. The training dataset is used to fit and tune your models. The test dataset is used to evaluate your models. The synthetic data used to create the feed-forward neural network is trained to produce the results for the worst case that is the amount of contribution of the parameters to cause it and identify the causal parameter.

To enhance the model's prediction capabilities for worst-case scenarios, we implemented a reinforcement learning (RL) approach. An RL agent was trained to predict the values that resulted in the maximum difference in system behavior. The agent was trained using the maximum difference as the reward, aiming to maximize this difference through its actions. An environment is created to represent the data, where the state is the time step and the action is the perturbed values of TM in the case of parameter one and G0X, G0Y, and G0Z in the case of parameter two. The agent interacts with this environment, learning to predict the maximum difference along with the corresponding perturbed values.

Here, a training loop is implemented for a reinforcement learning (RL) agent using a Deep Q-network (DQN) to learn in an environment. Within each episode, the agent iterates through a fixed number of steps, where it selects actions based on an epsilon-greedy policy, balancing exploration and exploitation. After choosing an action, the agent observes the resulting next state and reward from the environment. Using these observations, it updates its Q-value estimates via temporal difference learning, aiming to minimize the mean squared error between predicted and target Q-values. The agent gradually improves its policy over episodes, adjusting its exploration rate through epsilon decay.

A Q-learning algorithm is used to train a neural network model to learn an optimal policy for an environment. The training process involves iterating over multiple episodes, where each episode starts with resetting the environment to its initial state. Within each episode, the agent interacts with the environment for a fixed number of steps, selecting actions based on an epsilon-greedy policy. This policy balances exploration and exploitation by choosing random actions with probability epsilon or selecting the action with the highest predicted Q-value otherwise. After taking each action, the agent receives feedback from the environment in the form of the next state and the associated reward. The Q-values of the current state-action pairs are then updated using the Q-learning update rule, which combines the immediate reward with the discounted maximum Q-value of the next state. The variation in the Q values during the training are shown in Fig. 4, Fig. 5 and Fig. 6.

```
Episode: 1
Episode: 1, Step: 1, Q-Values: [ 3.045322   -0.20789585]
Episode: 1, Step: 2, Q-Values: [0.30490535 2.0905337 ]
Episode: 1, Step: 3, Q-Values: [ 1.1002444  -0.11728764]
Episode: 1, Step: 4, Q-Values: [ 0.00531591 -0.06037834]
Episode: 1, Step: 5, Q-Values: [1.1141495e+00 8.1385439e-04]
Episode: 1, Step: 6, Q-Values: [0.12273315 2.0702875 ]
Episode: 1, Step: 7, Q-Values: [0.22273184 3.0909324 ]
Episode: 1, Step: 8, Q-Values: [ 4.067758   -0.14759757]
Episode: 1, Step: 9, Q-Values: [ 4.733206   -0.19090801]
Episode: 1, Step: 10, Q-Values: [ 3.5965018  -0.22516152]

Episode: 10, Step: 90, Q-Values: [34.093266 71.670235]
Episode: 10, Step: 91, Q-Values: [38.20626  57.428745]
Episode: 10, Step: 92, Q-Values: [30.683622 43.106926]
Episode: 10, Step: 93, Q-Values: [23.15209  29.151571]
Episode: 10, Step: 94, Q-Values: [15.832651 14.431946]
Episode: 10, Step: 95, Q-Values: [ 8.120939 29.036913]
Episode: 10, Step: 96, Q-Values: [15.781073 42.893974]
Episode: 10, Step: 97, Q-Values: [23.057455 57.088783]
Episode: 10, Step: 98, Q-Values: [30.519163 71.352036]
Episode: 10, Step: 99, Q-Values: [38.022793 77.5261  ]
Episode: 10, Step: 100, Q-Values: [41.25957  61.873142]
Value of the maximum difference in the training data: Max_Diff     5.138854
TM          50.000000
Value       -2.000000
Name: 45, dtype: float64
```

Fig. 4 Prediction of the worst-case output of parameter 1

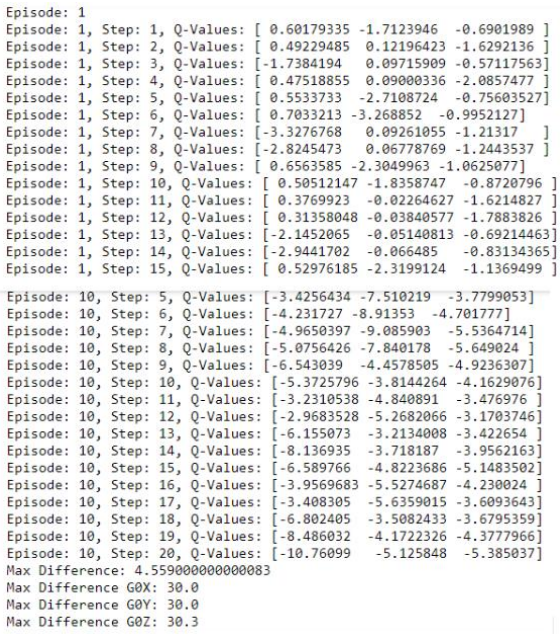

Fig. 5 Prediction of the worst-case output of parameter 2 (Case 1 - Apogee, Perigee)

Throughout the training, the epsilon value gradually decays to shift the agent's focus from exploration to exploitation. This decay ensures that the agent initially explores the environment more broadly but gradually exploits the learned policy as training progresses. Finally, after all episodes are completed, the predictions can be made. The only difference in the architecture of both the parameters is that the dimension of the output parameter is 2 for parameter one and 3 for parameter two as shown in Fig. 7 and Fig. 8 respectively.

5.5 Prediction

The models designed here are then used to predict the unknown results, thereby obtaining the combination of parameters that cause the Launch Vehicle's worst-case scenarios. Assessing the results of a simulation is a crucial step to ensure the reliability, accuracy, and usefulness of the simulation. The assessment process involves analyzing the simulation outputs, comparing them with expected or known outcomes, and nominal values, and interpreting the implications of the results. Here's a systematic approach to assess the results of a simulation:

- Verification
- Visualization
- Documentation

Prediction screenshots for both parameters are given in Fig. 4, Fig. 5 and Fig. 6

5.6 Analysis and Visualization

Finally, the predicted output is analyzed by comparing the parameter value for the maximum difference condition with the parameter value for the nominal file of parameter one. This is done by visualizing them graphically by plotting their corresponding rundap file plots as shown in Fig. 9 and Fig. 10 respectively. The two conditions for the worst case of the second parameter are also analyzed.

```
Episode: 1
Episode: 1, Step: 1, Q-Values: [-0.00262263  0.00149094 -0.00913356]
Episode: 1, Step: 2, Q-Values: [-0.01321035  0.00048096 -0.00219598]
Episode: 1, Step: 3, Q-Values: [-0.00728818 -0.01274954 -0.00387515]
Episode: 1, Step: 4, Q-Values: [-0.00796852 -0.00142268 -0.01818221]
Episode: 1, Step: 5, Q-Values: [-0.00906999 -0.00426396 -0.02418608]
Episode: 1, Step: 6, Q-Values: [-0.01102697 -0.00593751 -0.02813355]
Episode: 1, Step: 7, Q-Values: [-0.01766771 -0.00735369 -0.01181917]
Episode: 1, Step: 8, Q-Values: [-0.02082357 -0.00650924 -0.01250745]
Episode: 1, Step: 9, Q-Values: [-0.01587223 -0.00751237 -0.0226281 ]
Episode: 1, Step: 10, Q-Values: [-0.01802798 -0.02320066 -0.0161149 ]
Episode: 1, Step: 11, Q-Values: [-0.01959869 -0.00969149 -0.02578755]
Episode: 1, Step: 12, Q-Values: [-0.02129823 -0.02821897 -0.01918532]
Episode: 1, Step: 13, Q-Values: [-0.03136252 -0.01329604 -0.02061987]
Episode: 1, Step: 14, Q-Values: [-0.02430721 -0.01515685 -0.0366115 ]
Episode: 1, Step: 15, Q-Values: [-0.02637013 -0.03571239 -0.02394418]

Episode: 10, Step: 10, Q-Values: [-0.7161838 -0.7068496 -0.7134363]
Episode: 10, Step: 11, Q-Values: [-0.7234353  -0.7139064  -0.72203237]
Episode: 10, Step: 12, Q-Values: [-0.7315234 -0.7221007 -0.7206465]
Episode: 10, Step: 13, Q-Values: [-0.7393339  -0.729657   -0.74034035]
Episode: 10, Step: 14, Q-Values: [-0.7470699  -0.7544257  -0.73739445]
Episode: 10, Step: 15, Q-Values: [-0.7576673  -0.7491942  -0.75734377]
Episode: 10, Step: 16, Q-Values: [-0.7621857  -0.75465995 -0.76064116]
Episode: 10, Step: 17, Q-Values: [-0.7665523 -0.7597444 -0.7640486]
Episode: 10, Step: 18, Q-Values: [-0.76722115 -0.7643733  -0.7643138 ]
Episode: 10, Step: 19, Q-Values: [-0.77343106 -0.7689355  -0.76864475]
Episode: 10, Step: 20, Q-Values: [-0.77574277 -0.77243966 -0.77236533]
Max Inclination Difference: 0.02200000000000024
Max Difference G0X: 30.3
Max Difference G0Y: 30.0
Max Difference G0Z: 30.0
```

Fig. 6 Prediction of the worst-case output of parameter 2 (Case 2 - Inclination

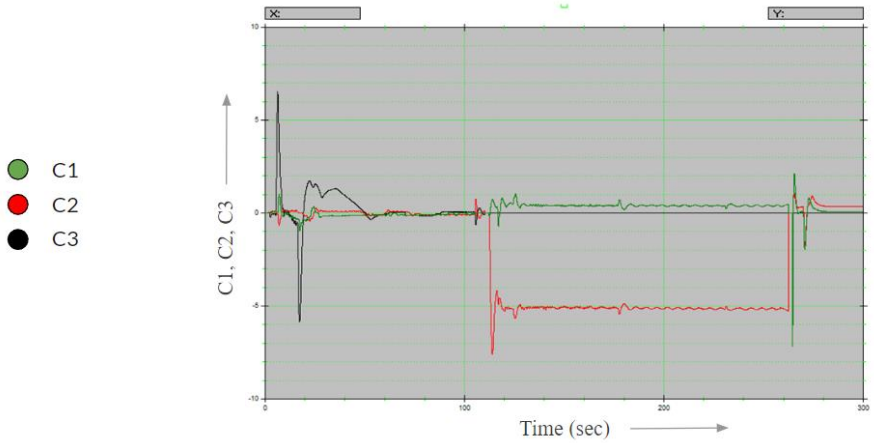

Fig. 9 Graph of parameter 1 for worst-case condition

VI. Results and Discussion

Prediction of the worst-case output for thrust misalignment based on the updated Q values are done as shown in Fig. 4. We can analyze the output based on the worst-case Q-value predicted by the DQN, which in turn predicts the maximum thrust misalignment value of -2.000000 with an asymmetry of at 50.000000 degrees. Assume that the Q-values represent the expected cumulative reward of taking a particular action in a given state, a negative Q-value indicates a negative reward, which could correspond to a failure or a suboptimal outcome thus pointing to a worst-case scenario where maximum fuel usage occurs.

The graph of thrust misalignment for the nominal condition is given in Fig. 10. From Fig. 9 and Fig. 10, the predicted data point at an angle of 50 degrees with -2 inclination and the nominal file (180 degrees with 0 inclination) graphs are compared where we can observe a significant deviation between graphs from the time 120 to 260 which points toward the maximum deviation which infers that the predicted point points to a maximum worst case scenario. Graph of comparison of the C1 value of thrust misalignment between nominal condition and worst case condition and graph of the comparison of C2 value of thrust misalignment between the nominal condition and worst case condition is given in Fig. 11 and Fig. 12 respectively.

The output in Fig. 4 shows the Q-values for each action at each step of each episode. The Q-values represent the expected cumulative reward for taking each possible action in the current state. The agent chooses the action with the highest Q-value at each step. Here in the first episode, the initial state has a Q-value of [0.60179335, -1.7123946, -0.6901989] for each of the three possible actions. The agent chooses the first action, which has a Q-value of 0.60179335 and observes the new state and reward. The new state has a Q-value of [0.49229485, 0.12196423, -1.6292136], and so on for each subsequent step. The final Q-values are shown at the end of each episode. For example, at the end of the first episode, the final Q-values are [0.42719033, -3.9943874, -1.2111145]. These Q-values represent the expected cumulative reward for taking each possible action in the final state of the episode.

At the end of the training process for the prediction of the worst-case output for parameter 2, the maximum difference and the corresponding G0X, G0Y, and G0Z values are printed. These values represent the largest deviation from the desired trajectory, which the agent aims to minimize. In this case, the final maximum difference value is 4.559. The corresponding G0X, G0Y, and G0Z values of 30.0, 30.0, and 30.3 respectively, indicate the specific deviations in each dimension that led to the maximum difference.

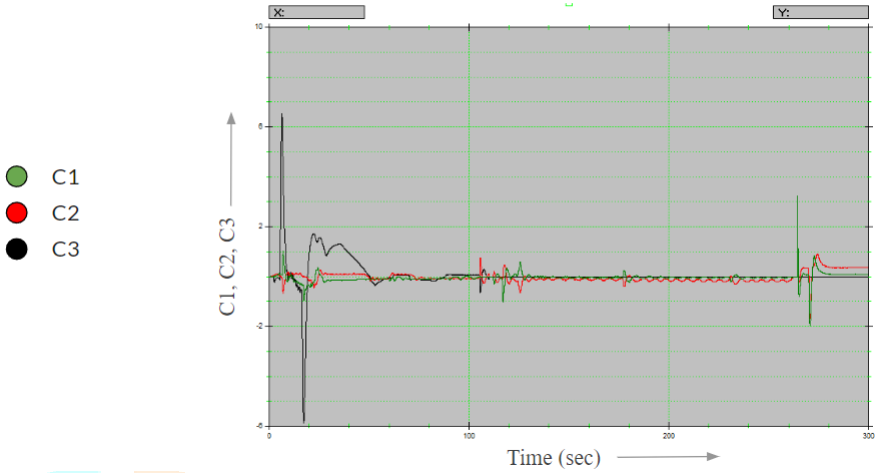

Fig. 10 Graph of parameter 1 for nominal condition

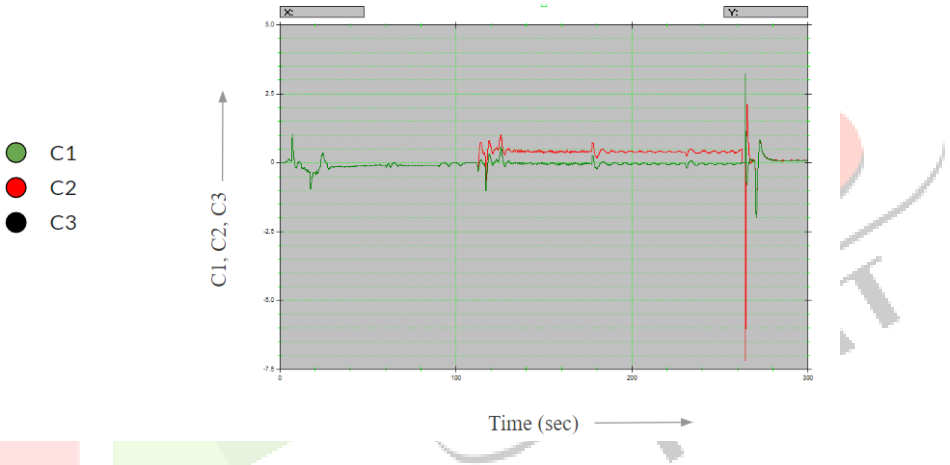

Fig. 11 Graph of comparison of C1 value of parameter 1 between nominal and worst case condition

The output in Fig. 6 at the end of the provided code shows the maximum inclination difference and the corresponding G0X, G0Y, and G0Z values, which are calculated based on the predicted Q-values. The maximum inclination difference is the highest inclination difference observed during the training process. In this case, the maximum inclination difference is 0.02200000000000024, which is a small value. However, it is important to note that the inclination difference is a relative measure, and a small value may still be significant in the context of the problem being solved. These values provide insight into the specific actions that the agent took to achieve the maximum inclination difference. A high maximum inclination difference and reasonable G0X, G0Y, and G0Z values of 30.3, 30.0, and 30.0 respectively, indicate that the DQN model has learned a policy that leads to a large inclination difference, which is the desired outcome.

Here, for the second parameter, the G0X, G0Y, and GOZ values predicted at both the apogee perigee difference and the inclination difference are taken into consideration for finding the translational and rotational aspects of the motion.

Overall in our prediction, there are high chances of failure of the Launch vehicle due to the over-exhaustion of energy when the thrust misalignment at an angle of 50 degrees at -2° during gyroscope drift of both translational point (G0X, G0Y, G0Z) = (30.0, 30.0, 30.3) and rotational point (G0X, G0Y, G0Z) = (30.3, 30.0, 30.0). The overall results are tabulated and presented in Table 1.

Table 1: Results

Parameter	Maximum Difference	Worst Case Outputs	
Thrust Misalignment	5.138854	Angle	50.000000
		Value	-2.000000
Gyroscope Drift (Apogee - Perigee)	4.559000000000083	G0X	30.0
		G0Y	30.0
		G0Z	30.3
Gyroscope Drift (Inclination)	0.022000000000000024	G0X	30.3
		G0Y	30.0
		G0Z	30.0

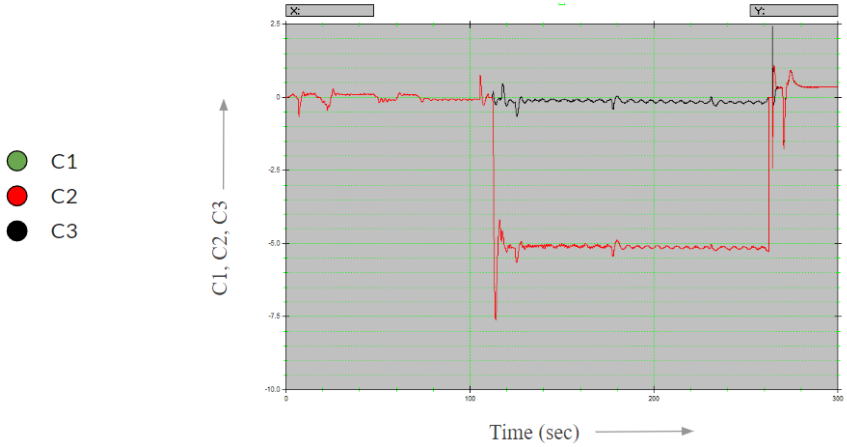

Fig. 12 Graph of comparison of C2 value of parameter 1 between nominal and worst case condition

This study sought to improve worst-case scenario analysis during rocket launches by leveraging advanced machine learning techniques, specifically deep Q-learning. Conventional approaches frequently struggle to encompass the entire spectrum of potential situations, highlighting the potential of machine learning's adaptable and data-centric approach. By accurately predicting worst-case scenarios, this approach could significantly improve mission safety and success rates. Despite the importance, there's a notable gap in applying advanced machine learning to this area.

The study successfully developed a deep Q-learning framework for this purpose, demonstrating its effectiveness in predicting worst-case scenarios. However, further research could explore advanced reinforcement learning techniques, different neural network architectures, and handling multi-agent systems to enhance the model's performance and applicability. In summary, this study marks a notable progression in utilizing machine learning for worst-case scenario analysis in rocket launches. It establishes a groundwork for forthcoming research endeavors and enhancements aimed at bolstering space mission safety and enhancing success rates.

VII. CONCLUSION

The project's goal is to enhance rocket launch safety and success by devising a machine learning framework to simulate and predict worst-case scenarios during launch. This proactive approach enables preemptive measures to address extreme situations, promoting safer and more successful missions and furthering space exploration. This project signifies a notable milestone in the development of space exploration technologies.

In the project's future scope, there lies the potential for integrating advanced deep reinforcement learning methods like Double DQN, Dueling DQN, Prioritized Experience Replay, or Rainbow DQN. These enhancements aim to amplify the agent's capacity for learning and adaptation within intricate environments. Furthermore, extending the project to encompass multi-agent systems or cooperative/competitive setups could

unveil fresh avenues for research and confrontations. Collectively, these forthcoming endeavors hold the promise of cultivating a more resilient and adaptable RL agent, equipped to navigate diverse tasks and surroundings.

VIII. ACKNOWLEDGMENT

The authors express their gratitude to their principal, Dr. Jayamohan J, their head of department, Prof. Anithakumari S, their project coordinator, Prof. Sandeep Chandran, and guide, Prof. Rehna R S, for providing them with necessary facilities and infrastructure for their final year main project. They also thank the employees at VSSC, Trivandrum, especially Jishy Samuel, for her invaluable support and guidance.

REFERENCES

[1] Crisostomi Emanuele, A Lecchini-Visintini and Jan Maciejowski, Combining Monte Carlo and worst-case methods for trajectory prediction in air traffic control, Jan. 2008.

[2] Zhenrui Guo, A Simulation Program that Controls Rockets Using AI Trained with Machine Learning, 247-262., 10.5121/csit.2023.131319., 2023.

[3] Bjorn Andersson, Dionisio de Niz, Gabriel Moreno, Jeffery Hansen, and Mark Klein, Assessing the Use of Machine Learning to Find the Worst-Case Execution Time of Avionics Software, Carnegie Mellon University, Software Engineering Institute, United States. Department of Transportation. Federal Aviation Administration. William J. Hughes Technical Center, May 2023.

[4] V. Kumar, Deep Neural Network Approach to Estimate Early Worst-Case Execution Time, 2021 IEEE/AIAA 40th Digital Avionics Systems Conference (DASC), San Antonio, TX, USA, 2021.

[5] Mohammad Moltafet, Markus Leinonen and Marian Codreanu, Worst Case Analysis of Age of Information in a Shared-Access Channel, 16th International Symposium on Wireless Communication Systems (ISWCS), Oulu, Finland, 2019.

[6] Fuqiang Cheng, Xiaohong Guo, Yingying Qi, Jingwen Xu, Wan Qiu, Zhibao Zhang, Weitao Zhang, Ningning Qi, Research on Satellite Power Anomaly Detection Method Based on LSTM, China, IEEE (ICPECA), 2021.

[7] Aviral Kumar, Aurick Zhou, George Tucker and Sergey Levine, Conservative Q-Learning for Offline Reinforcement Learning, UC Berkeley, Aug. 2020.

[8] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver and Daan Wierstra, Continuous Control With Deep Reinforcement Learning, July 2019.