Performance of a Model Predictive Control Based Autonomous Rendezvous and Docking Algorithm for CubeSats using Hardware Emulation

Andrew Fear [*]
Georgia Institute of Technology, Atlanta, Georgia

E. Glenn Lightsey[†]
Georgia Institute of Technology, Atlanta, Georgia

Hardware emulation of typical CubeSat flight computers is utilized to benchmark the performance of a three-phase Model Predictive Control (MPC) algorithm for autonomous rendezvous and docking (AR&D). The length of the MPC prediction horizons affects the computational complexity and therefore the solution time of finding an optimal control sequence. This study investigates the limitations, if any, of current state-of-the-art CubeSat flight systems regarding the ability to take advantage of this type of guidance algorithm. A virtual machine with an ARM processor typical of CubeSat available hardware is used to test the performance of the algorithm. Monte Carlo simulations are run to calculate the average computation time per optimal control solution and compare these values across varying prediction horizon lengths.

I. Introduction

Autonomous Rendezvous and Docking (AR&D) is a key enabling technology for missions involving spacecraft repair, re-supply and crew exchanges, object retrieval, and on-orbit assembly of larger structures [1]. Of interest to this paper is AR&D using CubeSat class small satellites. For example, [2] explores using CubeSats for on-orbit assembly of a telescope structure. Performing on-orbit assembly using CubeSats lowers costs associated with individual launches and repairs. CubeSats are typically considered secondary payloads and are launched as rideshare opportunities, which is less expensive than buying a launch opportunity as a primary payload. As for repair, the in-space repair of a large spacecraft is not always possible and may be costly. In an assembled structure, if one elementary unit has malfunctioned, only a single replacement unit needs to be launched to take the place of the malfunctioned unit. However, this requires complex un-docking and docking maneuvers to be performed.

There are additional challenges to utilizing CubeSats for AR&D that stem from the small form factor and low production cost. There are limited commercial options available for CubeSat hardware that meets volume constraints and in effect constrains the performance available. An MPC algorithm for CubeSat AR&D that considers the restricted

[*]Ph.D. Candidate, Department of Aerospace Engineering, Student AIAA Member.
[†]Professor, Department of Aerospace Engineering, AIAA Member

available actuation that minimizes fuel was developed in [3]. While it was shown that the guidance can be performed while adhering to the actuation limits, the computational load on CubeSat flight hardware was not calculated. This paper uses hardware emulation to remain agnostic to specific CubeSat implementations while investigating the computation load and times needed to run the guidance algorithm to test its efficacy on these flight systems.

The paper is organized as follows. First a brief description of the MPC AR&D guidance algorithm is presented. Next, a discussion of tools and setup, as well as the parameters chosen for the hardware emulation. Finally, the resulting computation times for varying algorithm parameters, such as the prediction horizon length and sampling times, is investigated.

II. Problem Scneario

The problem scenario investigated involves two 6U CubeSats in Low Earth Orbit (LEO). One CubeSat, dubbed the chaser, is active and performs the AR&D maneuver using a 6-DOF propulsion system. The other Cubesat is the passive target of the maneuver. The dynamics is restricted to the motion of the center of mass (CoM) of the chaser spacecraft relative to the docking port located on the target spacecraft. The chaser is assumed to be controllable, which is not addressed further in this paper.

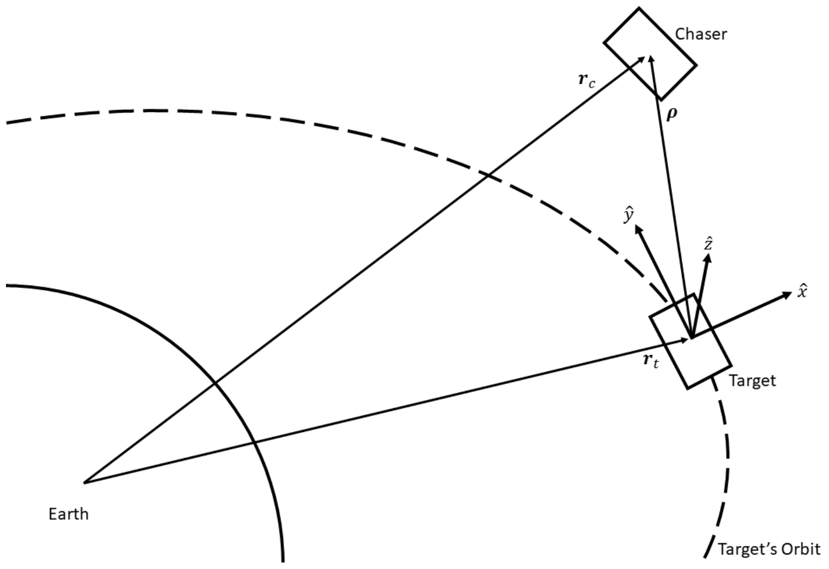

Fig. 1 LVLH Reference frame of the target

The relative motion is defined by the classic Clohessy-Wiltshire (CW) equations [4]. The reference frame, shown in Fig. 1, is centered at the target's center of mass. The $\hat{x}$-axis is radially outwards from Earth, the $\hat{y}$-axis in the velocity direction, and the $\hat{z}$-axis is in the direction of the target orbit angular velocity completing the right-hand rule.

The state vector $\boldsymbol{x}(k) \in \mathbb{R}^6$ is defined as $\boldsymbol{x}(k) = [\boldsymbol{\rho}(k) \quad \dot{\boldsymbol{\rho}}(k)]$, where $\boldsymbol{\rho}(k) \in \mathbb{R}^3$ describes the components of the relative position in the CW frame, while $\dot{\boldsymbol{\rho}}(k) \in \mathbb{R}^3$ describes the components of the relative velocity in the CW frame.

The applied control to the chaser is defined by the vector $\boldsymbol{u}(k) \in \mathbb{R}^3$. The equations of motion are linearized with a sample interval $t_s \in \mathbb{R}^+$ to obtain discrete linear CW state equations at time index $k \in \mathbb{N}_0$:

$$\boldsymbol{x}_{k+1} = \boldsymbol{A}\boldsymbol{x}_k + \boldsymbol{B}\boldsymbol{u}_k \tag{1}$$

III. AR&D MPC Algorithm

The objective of the AR&D algorithm is to guide the relative position of the chaser, $\boldsymbol{x}_k$ to the docking point located on the target body, $\boldsymbol{x}_k^d$. The strategy presented in [3] splits the scenario into 3 phases based on the relative distance of the chaser to the target's CoM: Rendezvous (> 150 m), Approach (10-150 m), and Docking (< 10 m). During each phase, a receding horizon MPC law is performed as follows. At every sample instance $t_s \in \mathbb{R}^+$, a constrained linear quadratic (LQ) optimization problem is solved for the control sequence $\boldsymbol{u}_k$ over a prediction horizon of length $N \in \mathbb{N}_0$. The first element of the control sequence, $\boldsymbol{u}_k^0$, is applied to the system. The full algorithm can be seen in Algorithm 1.

The corresponding LQ optimal control problem for the rendezvous phase MPC is given by

$$\begin{aligned}
\min_{\boldsymbol{u}_k} \quad & J(\boldsymbol{x}_k, \boldsymbol{u}_k) = \sum_{i=0}^{N-1} \left(\|\boldsymbol{x}_{i|k}\|_Q^2 + \|\boldsymbol{u}_{i|k}\|_R^2 \right) + \|\boldsymbol{x}_{N|k}\|_{Q_f}^2 + \lambda \mathbf{1}^{\mathbf{T}} \left(\boldsymbol{H}\boldsymbol{x}_{N|k} \right)_+ \\
\textbf{s.t.} \quad & \boldsymbol{x}_{0|k} = \boldsymbol{x}(t_k) \\
& \boldsymbol{x}_{i+1|k} = \boldsymbol{A}\boldsymbol{x}_{i|k} + \boldsymbol{B}\boldsymbol{u}_{i|k} \\
& \hat{\boldsymbol{n}}_k^T \boldsymbol{x}_k \geq r_1 \quad k = 0, ..., N \\
& |\boldsymbol{u}_k|_\infty \leq u_{max}
\end{aligned} \tag{2}$$

with prediction horizon N_1 and control intervals of $t_{s,1}$. The constraint $\hat{\boldsymbol{n}}_k^T \boldsymbol{x}_k \geq r_1 \quad k = 0, ..., N$ describes a convexified "keep-out zone" (KOZ) to avoid solutions where the chaser would fly close by the target to reach the docking point. The end condition for this phase is when the chaser position has reached the approach cone, approximated as an n-dimensional polyhedron defined by $\mathcal{P} = \left\{ \boldsymbol{x} \in \mathbb{R}^3 : \boldsymbol{H}\boldsymbol{x} \leq 0 \right\}$, where $\boldsymbol{H} \in \mathbb{R}^{n \times 3}$.

Upon reaching the approach cone, the approach phase finds the optimal control sequence by solving Eq. 3. The control response is quicker with a shorter prediction horizon N_2 and sample time $t_{s,2}$. The approach phase ends once chaser reaches the $r_2 = 10$ m separation distance within the docking cone.

$$
\begin{aligned}
\min_{\boldsymbol{u}_k} \quad & J(\boldsymbol{x}_k, \boldsymbol{u}_k) = \sum_{i=0}^{N-1} \left(\|\boldsymbol{x}_{i|k}\|_Q^2 + \|\boldsymbol{u}_{i|k}\|_R^2 \right) + \|\boldsymbol{x}_{N|k}\|_{Q_f}^2 + \lambda \sum_{i=1}^{N} \mathbf{1}^T \left(\boldsymbol{H}\boldsymbol{x}_{i|k} \right)_+ \\
\text{s.t.} \quad & \boldsymbol{x}_{0|k} = \boldsymbol{x}(t_k) \\
& \boldsymbol{x}_{i+1|k} = \boldsymbol{A}\boldsymbol{x}_{i|k} + \boldsymbol{B}\boldsymbol{u}_{i|k} \\
& \hat{\boldsymbol{n}}_{i|k}^T \boldsymbol{x}_{i|k} \geq r_2 \quad i = 0, ..., N \\
& |\boldsymbol{u}_k|_\infty \leq u_{max}
\end{aligned}
\tag{3}
$$

The docking phase solves Eq. 4 for the optimal control sequence with prediction horizon N_3 at sample interval $t_{s,3}$.

$$
\begin{aligned}
\min_{\boldsymbol{u}_k} \quad & J(\boldsymbol{x}_k, \boldsymbol{u}_k) = \sum_{i=0}^{N-1} \left(\left\|\boldsymbol{x}_{i|k} - \boldsymbol{x}_{i|k}^d\right\|_Q^2 + \|\boldsymbol{u}_{i|k}\|_R^2 \right) + \left\|\boldsymbol{x}_{N|k} - \boldsymbol{x}_{N|k}^d\right\|_{Q_f}^2 + \lambda \sum_{i=1}^{N} \mathbf{1}^T \boldsymbol{H} \left(\boldsymbol{x}_{i|k} - \boldsymbol{x}_{N|k}^d \right)_+ \\
\text{s.t.} \quad & \boldsymbol{x}_{0|k} = \boldsymbol{x}(t_k) \\
& x_{i+1|k} = \boldsymbol{A}\boldsymbol{x}_{i|k} + \boldsymbol{B}\boldsymbol{u}_{i|k} \\
& - \hat{\boldsymbol{n}}_{i|k} \cdot \boldsymbol{x}_{i|k} \leq 0 \quad i = 0, ..., N \\
& \left|\boldsymbol{u}_{i|k}\right|_\infty \leq u_{max}
\end{aligned}
\tag{4}
$$

IV. Simulation Strategy

A. Convex Optimization Problem Solver

The convex LQ optimal control problems are solved using the CVXGEN [5] code generation tool. The mathematical descriptions of the optimal control problems Eqs. 2, 3, and 4 are input into the online solver to obtain an optimized C code solver.

B. 42 Spacecraft Simulator

The 42 Spacecraft Simulation tool [6] is an open-source framework written in C for testing spacecraft controllers and proximity flight operations. A GUI is included in the 42 framework as a visual aid. This work uses 42 to propagate the chaser and target state dynamics with Earth as a central body including perturbing forces from aerodynamic drag and Earth's oblateness. The simulation also contains a thruster model that is used to perform the control efforts calculated.

42 can be compiled to run as a single simulation executable as well as separate client-server instances. In this configuration, the server instance handles the environment and dynamics updates while passing off necessary information to the client connection which runs only the controller instance. This separation allows the testing of the MPC guidance

Algorithm 1: AR&D Guidance

```
Initialize $t_k = t(0), \boldsymbol{x}_k = \boldsymbol{x}(0)$
begin phase 1 Rendezvous
    Set rendezvous prediction horizon parameters
    $N \leftarrow N_1$
    $t_s \leftarrow t_{s,1}$
    repeat
        Solve (2) for optimal control sequence $\boldsymbol{u}_k$
        Apply first control element, $u_k^0$
        $k \leftarrow k + 1$
        Update state estimate, $\boldsymbol{x}_k$
    until $\boldsymbol{x}_k <= H\boldsymbol{x}_k$ and $\|\boldsymbol{x}_k\| \leq r_1$            // End Rendezvous if inside approach cone
end
begin phase 2 Approach
    Set approach prediction horizon parameters
    $N \leftarrow N_2$
    $t_s \leftarrow t_{s,2}$
    repeat
        Solve (3) for optimal control sequence $\boldsymbol{u}_k$
        Apply first control element, $u_k^0$
        $k \leftarrow k + 1$
        Update state estimate, $\boldsymbol{x}_k$
    until $\boldsymbol{x}_k \leq r_2$                                      // End Approach Phase
end
begin phase 3 Docking
    Set docking prediction horizon parameters
    $N \leftarrow N_3$
    $t_s \leftarrow t_{s,3}$
    repeat
        Solve (4) for optimal control sequence $\boldsymbol{u}_k$
        Apply first control element, $u_k^0$
        $k \leftarrow k + 1$
        Update state estimate, $\boldsymbol{x}_k$
    until $\boldsymbol{x}_k \leq r_3$                                      // End Maneuver
end
```

algorithm separate from the environment simulation as if it was running on a spacecraft.

C. ARM Emulation

In order to verify the control algorithm's efficiency in a flight-like setting, a virtual machine is be used to run the 42 client portion of the simulation. The open-source machine virtualizer, QEMU [7], is used to create the target platform. A virtual machine is chosen instead of performing a real hardware-in-the-loop test to remain agnostic to the hardware chosen. This allows for different board specification testing as needed. As most CubeSat available flight computers contain ARM processors, this type was chosen as the generic processor to use for emulation.

As shown in Fig. 2, the 42 simulation server runs on the host machine, while the 42 client runs on the virtualized ARM machine. A socket connection passes necessary state and actuation effort between the server-client programs.

The limitations of QEMU are that it may not be entirely cycle-accurate. However, it provides a preliminary evaluation of running the developed algorithm on an appropriate system.

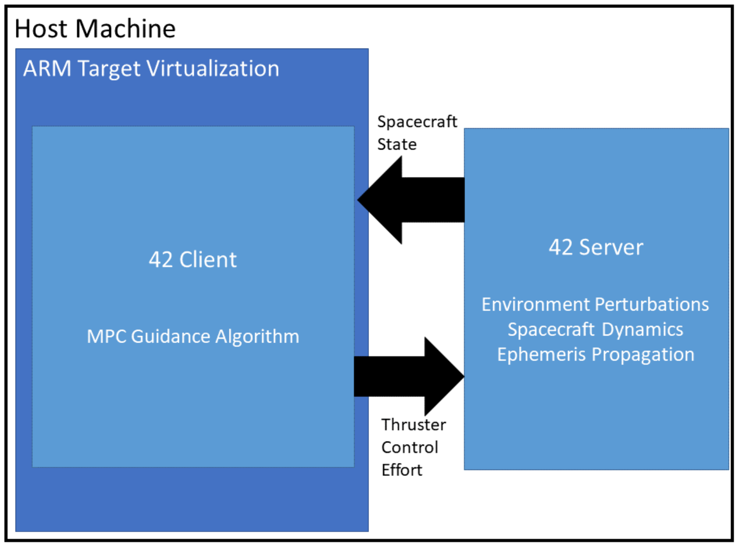

Fig. 2 Diagram showing the interprocess communication between the environment simulation server and the flight software client running on a virtual machine.

V. Algorithm Benchmarking

The length of the prediction horizon, N, in the optimal control problem affects the response and performance of the guidance algorithm. However, increasing the prediction horizon also increases the complexity of the solver and therefore the computation time needed to find a solution. As the guidance algorithm is intended to be flown in onboard a spacecraft in real-time, this computation time must be much less than the MPC sample time. Additionally, a faster solution time equates to less delay in the applied actuation effort.

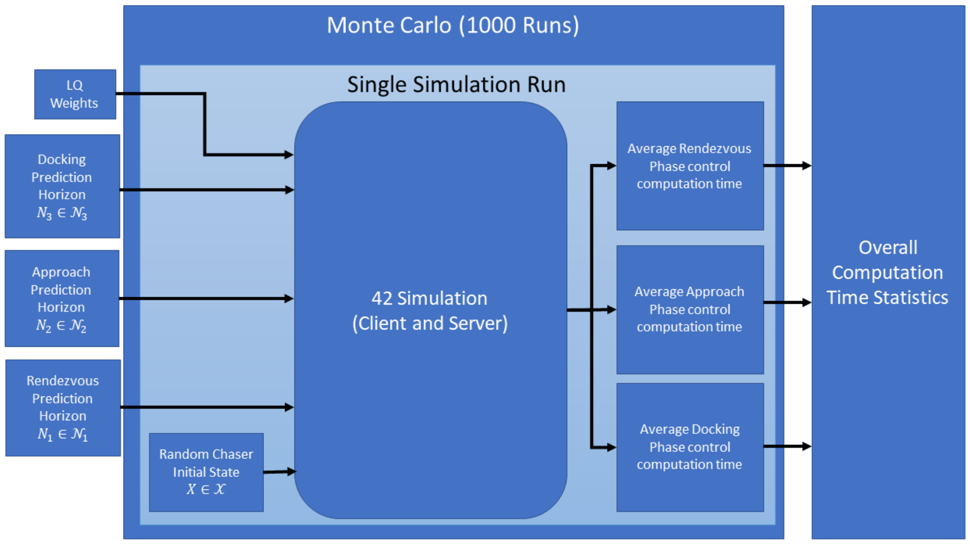
Monte Carlo (1000 Runs)
Single Simulation Run
LQ Weights
Docking Prediction Horizon $N_3 \in \mathcal{N}_3$
Approach Prediction Horizon $N_2 \in \mathcal{N}_2$
Rendezvous Prediction Horizon $N_1 \in \mathcal{N}_1$
Random Chaser Initial State $X \in \mathcal{X}$
42 Simulation (Client and Server)
Average Rendezvous Phase control computation time
Average Approach Phase control computation time
Average Docking Phase control computation time
Overall Computation Time Statistics

Fig. 3 Algorithm computation time test plan.

Using the aforementioned simulation setup, Monte Carlo analysis was performed with randomized chaser initial conditions. During the individual simulation runs, the computation time of the CVXGEN calls made to solve for the optimal control at each time step was logged. The average solution computation time for the phases is calculated after completion of the run. Overall computation time statistics are compiled from the 100 simulation runs. The test plan, shown in Fig. 3, involves Monte Carlo sets for varying values of prediction horizon lengths for the rendezvous and approach phases to investigate how this parameter affects the computation time. Another parameter of interest are the MPC control interval times. From the resulting data, an informed decision on the efficacy of running the guidance algorithm on currently available hardware, and the amount of control delay to expect is inferred.

A. Rendezvous Phase

For the rendezvous phase, the prediction horizon varies between $N_1 = 30, 35, 40$, while the approach and docking phase prediction horizons were kept constant at $N_2 = 15$ and $N_3 = 12$, respectively.

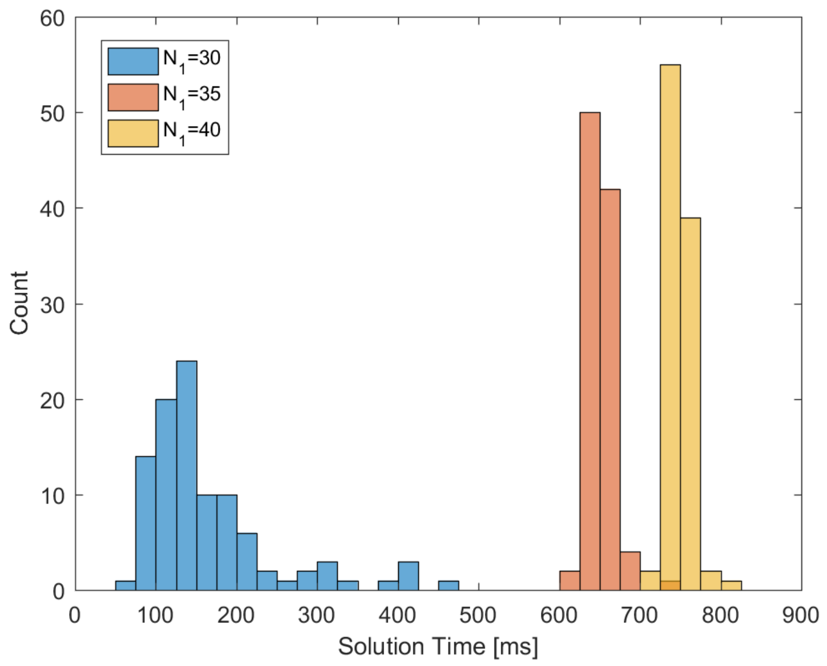

Fig. 4 Solution time counts for Rendezvous phase with varying prediction lengths.

In Fig. 4, the bin width of the rendezvous optimal control problem is 25 ms. As expected, the computation time is increased alongside the prediction horizon length. However, there is more spread and unpredictability with the lower end of the prediction horizon tests. The average computation time for the rendezvous phase prediction horizons of 30, 35, and 40 was 151.75 ms, 650.21 ms, and 748.25 ms, respectively. It should be noted that even with the longer prediction horizons the computation times are all less than a second, which gives ample time between control updates during the rendezvous phase.

Additionally, the rendezvous phase control intervals were tested to determine if there was an unexpected contribution to the computation time. A set of 100 scenarios were run for rendezvous control times of $T_s = 30, 45, 60$ s with a prediction length of $N_1 = 30$. Fig. 5 confirms that the distribution of computation times is similar for all control interval times.

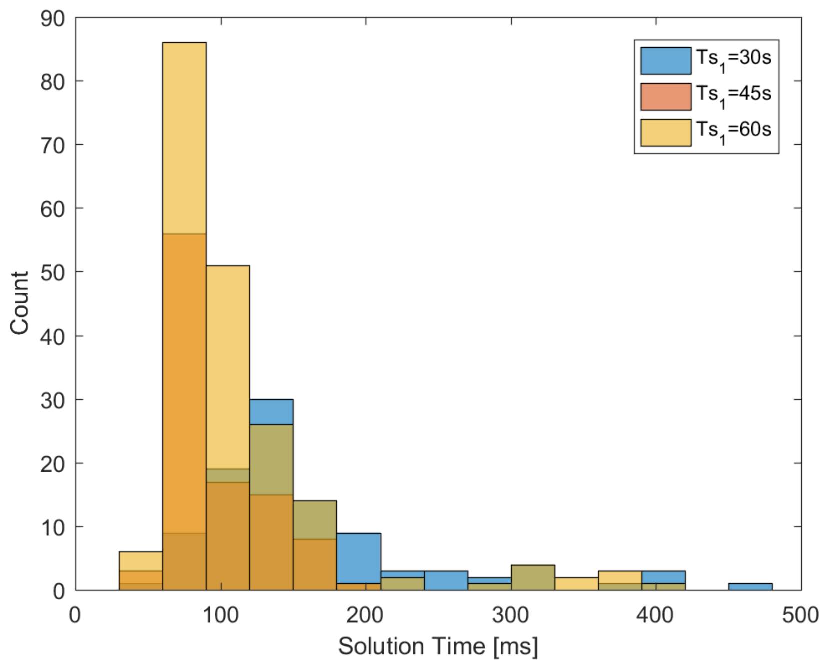

Fig. 5 Solution time counts for Rendezvous phase with varying control intervals.

B. Approach Phase

The prediction horizon values tested for the approach phase were $N_2 = 10, 15, 25, 30$. For the analysis, the rendezvous phase prediction length was kept to $N_1 = 30$.

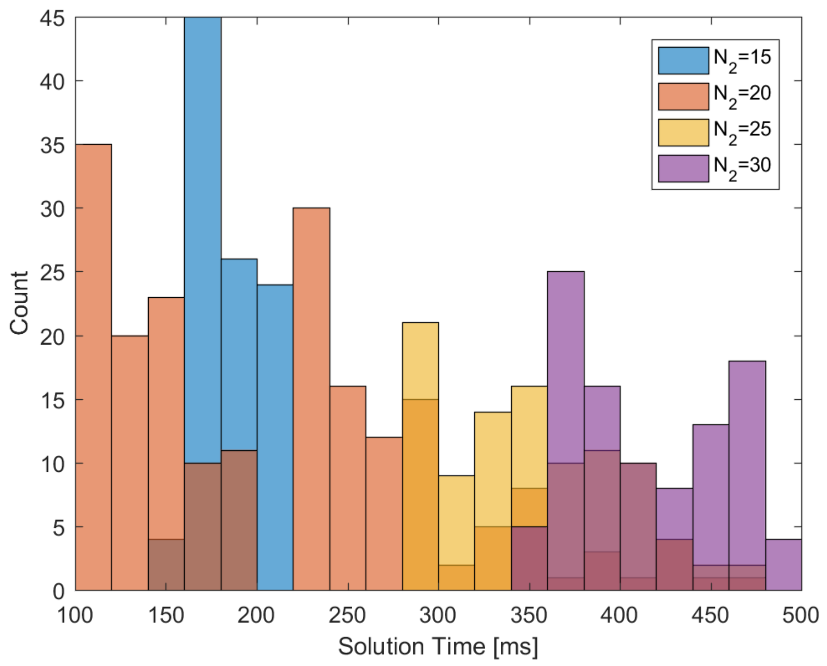

Fig. 6 Solution time counts for Approach phase with varying prediction lengths.

The results of the approach analysis are much less spread out than for the rendezvous phase. However, it does still show the general trend of greater computation with longer prediction length. The exception to this rule is $N_2 = 20$, which has greater spread. Similar to the rendezvous analysis, the computation times for the approach phase optimal control problem are all less than a second, again a good performance for an onboard system.

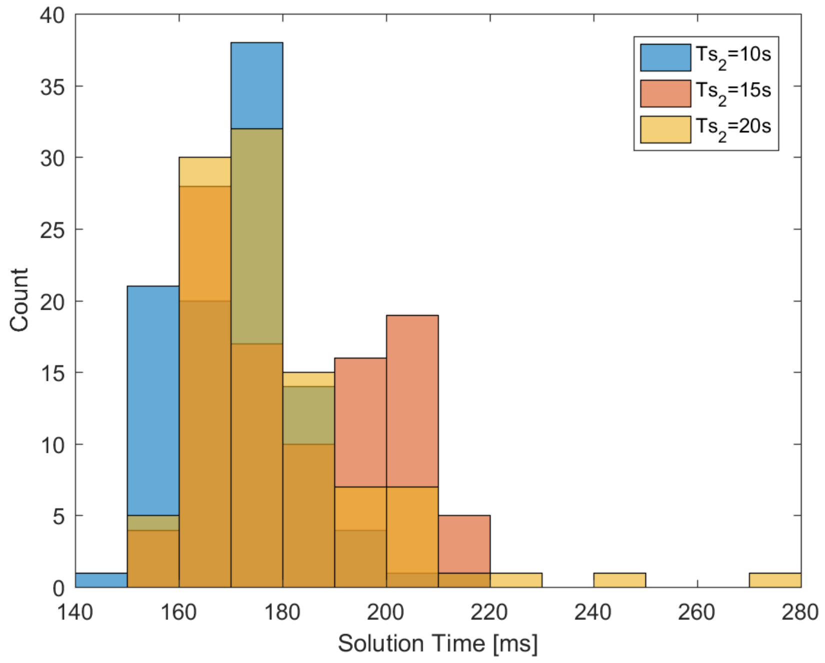

Fig. 7 Solution time counts for Approach phase with varying prediction lengths.

Additionally, the approach phase also shows the same insensitivity to control interval time as the rendezvous phase.

VI. Conclusion

The performance of a MPC guidance algorithm for the autonomous rendezvous and docking for small satellites was evaluated. The algorithm was run on emulated hardware similar in capabilities to those available for use on small satellites, specifically ARM architecture processors running a 64-bit Linux OS.

The analysis showed that although longer prediction horizon lengths in the optimal control problems in the rendezvous and approach phases of the tested algorithm resulted in longer computation times, the times remained under a second. These quick computations times are promising for the possibility of running such a guidance algorithm onboard. Future work will consist of finding improved emulation techniques that are cycle-accurate to confirm the findings of this report.

References

[1] Fehse, W., *Automated rendezvous and docking of spacecraft*, Cambridge University Press, Cambridge; New York, 2003.

[2] Pirat, C., "Toward the Autonomous Assembly of Large Telescopes Using CubeSat Rendezvous and Docking," *Journal of Spacecraft and Rockets*, Vol. 59, No. 2, 2021. https://doi.org/10.2514/1.A34945.

[3] Fear, A., and Lightsey, E. G., "Implementation of Small Satellite Autonomous Rendezvous Using Model Predictive Control," *AIAA*

SCITECH 2022 Forum, American Institute of Aeronautics and Astronautics, 2021, p. 0. https://doi.org/10.2514/6.2022-0838, URL https://arc.aiaa.org/doi/10.2514/6.2022-0838.

[4] Clohessy, W. H., and Wiltshire, R. S., “Terminal Guidance System for Satellite Rendezvous,” *Journal of the Aerospace Sciences*, Vol. 27, No. 9, 1960, pp. 653–658. https://doi.org/10.2514/8.8704.

[5] Mattingley, J., and Boyd, S., “CVXGEN: a code generator for embedded convex optimization,” *Optimization and Engineering*, Vol. 13, No. 1, 2012. https://doi.org/10.1007/s11081-011-9176-9.

[6] Stoneking, E. T., “42 - Spacecraft Simulation,” , Dec. 2021. URL https://github.com/ericstoneking/42.

[7] “QEMU,” , Sep. 2022. URL https://www.qemu.org/.