Robust Control for Landing Reusable Rockets

Aditya Parameshwaran
Mechanical Engineering
aparame@clemson.edu

Akshit Lunia
Mechanical Engineering
alunia@clemson.edu

Shahil Shaik
Mechanical Engineering
shahils@clemson.edu

***Abstract*—The cost-effective nature of reusable rockets has led to a need to develop control algorithms capable of landing first-stage rockets at designated targets successfully. However, the task of rocket re-landing needs very high precision, where even the slightest deviations can lead to catastrophic results. The objective of this paper is to develop a control algorithm that works with the nonlinear continuous time model of the rocket to generate optimal control inputs to land with minimum effort and deviation from the target, even in the presence of unmeasured wind disturbances. The wind disturbances usually vary and can not be designed into the system model. To estimate the unmeasured wind disturbances, we implement the Nonlinear Moving Horizon Estimation technique (NMHE), which estimates the disturbance in the current time step by observing the error in the estimated state and real state measurements. The disturbance estimates are then used by Nonlinear Model Predictive Control to compute an optimal control sequence. The optimal control input is decided based on minimizing the cost function, which incorporates the goal of our problem. The CasADI solver is used to solve the Nonlinear Programming problem. To observe the robustness of implemented controller, we apply the controller to 3 different scenarios. The results are compared to discuss the behavior of the controller and eventually the report is concluded with future work.**

***Index Terms*—Nonlinear Model Predictive Control, Nonlinear Moving Horizon Estimation, Rocket lander**

I. Introduction

Re-purposing rockets for relaunch has been gaining popularity ever since SpaceX became the first company to successfully land and relaunch the Falcon 9 rocket on March 30th, 2017. By reusing rockets, SpaceX was able to reduce the launch cost by 80%. The rocket landed on an autonomous ship about 70m x 50m. Maintaining safety and reliability of repeated landings is an important aspect of this feature, as one of the purposes in the future would be to land astronauts directly on the ground rather than the traditional parachute approach. Significant cost reduction in relaunching and decreasing raw material usage are the main motivational aspects of the relaunching rockets.

In general, Vertical Take-Off and Landing (VTOL) rockets are used for launching spacecrafts and payload past the atmosphere. They are made up of 2 main components; the first one is the actual payload with smaller rockets attached to the control capsule, and the second one is the larger rocket that propels the payload out from the ground to space. We attempt to land the former back to earth using optimal control methods so that it can be reused. Landing the rockets can be a complex task due to the disturbances that are typically seen during the process. One of the major reasons of a failed landing could be the effect of wind on the surface area of the rockets. The landing pads placed on the sea are also vulnerable to tidal changes. Hence SpaceX uses drone ships that can autonomously stabilise themselves to based on the rocket trajectory. It is therefore imperative to develop robust control approaches when tackling the landing approach and take into account the various disturbances that could occur in a real life scenario.

For scope of this project, we will discuss 3 rocket landing scenarios and the optimal control policies that can be used in the respective scenarios. We are considering a 2D environment with a non-linear dynamics model for a VTOL rocket that is landing on a wide enough barge on the sea. With a focus on optimality and robustness, we always try to minimise the fuel usage in landing of the rockets in addition to following a safe trajectory subject to constraints of dynamics equations and state and control variables. The problem is transformed into a nonlinear programming problem (NLP) via time discretization. NLP is solved using the CasADi solver which uses interior point optimization (IPOPT) method to solve for an optimal solution that minimises the cost function based on the set of the constraints over a prediction horizon. Finally we compare the results between the different scenarios and identify the robustness of the controller for the worst case scenario.

The remaining of the report is divided accordingly: Section II discusses about the state dynamics and the test scenarios, Section III introduces the problem formulation, with the cost function and set of constraints based on the each scenario. Then we discuss the results of the simulations in Section IV and finally end the report with Section V where we discuss the Conclusions and Future Work.

II. Preliminaries

In this report, we focus particularly on the final stage of landing the rocket on the barge on the sea. For a more accurate description of the model, we chose a system on nonlinear Ordinary Differentials Equations (ODE's) as described in [1].

A. Nonlinear Moving Horizon Estimation

The general application of a Nonlinear Moving Horizon Estimation (NMHE) technique is to estimate the unmeasured states and other unknown parameters at current time step. The NMHE observes the past and current system measurements and inputs at each discrete time step. The past measurements/input window is of a fixed dimension which moves and updates at every time step (Refer Figure 1). Assume that we are at time t_k with corresponding k time index, with the

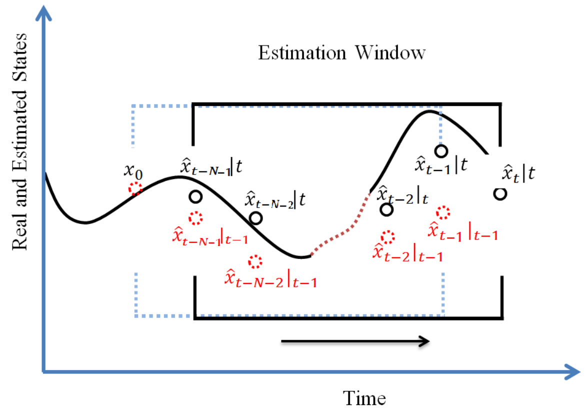

Fig. 1. Moving Horizon Estimation Scheme [4].

moving measurement window of $N + 1$ sampling instants. We discretize a continuous system to make the system computationally tractable. Assume the discretized system model with state estimation and disturbances as,

$$\begin{aligned} \dot{\hat{x}}(k) &= f(\hat{x}(k), u(k), \widehat{\omega}(k)) \\ y(k) &= h(\hat{x}(k), u(k)) + \hat{v}(k) \end{aligned} \tag{1}$$

Where, $\hat{\omega}(k)$ and $\hat{v}(k)$ are the estimated unmeasured disturbances in the process and measurements of the system. Let the unmeasured disturbances for the NMHE window be,

$$W_k = \begin{bmatrix} \widehat{\omega}_{k-N} \\ \widehat{\omega}_{k-N+1} \\ \vdots \\ \widehat{\omega}_k \end{bmatrix} \tag{2}$$

$$V_k = \begin{bmatrix} \hat{v}_{k-N} \\ \hat{v}_{k-N+1} \\ \vdots \\ \hat{v}_k \end{bmatrix} \tag{3}$$

with the convenient notation $\widehat{\omega}_k = \widehat{\omega}\,(t_k)$, $\widehat{v}_k = \widehat{v}(t_k)$, $\widehat{x}_k = \widehat{x}(t_k)$, and $\widehat{y}_k = \widehat{y}(t_k)$. The NMHE solves to find the current estimated state $\widehat{x}(t_k)$ and compare it to $x(t_k)$ based on the measured output $y(k)$, and the known control input $u(k)$. The estimator is designed as an optimal control problem [5],

$$\begin{aligned} &\min_{x_{k-N}, w_k} J\left(x(k-N), \dots, x(k), W_k, V_k\right) \\ &= T_s \cdot \sum_{i=k-N}^{k} \|\hat{v}_i\|^2_{R^{-1}} + T_s \cdot \sum_{k=0}^{i-1} \|\widehat{\omega}_i\|^2_{Q^{-1}} - \|\hat{x}_0 - \bar{x}_0\|^2_{P^{-1}} \end{aligned} \tag{4}$$

where, $\bar{x}_0$ is the initial state guess and $T_s = t_(k-1) - t_k$. The minimization must occur under the model constraints as well as additional constraints to assure stability. The solved optimization produces the estimated process disturbance $\widehat{\omega}(k)$ which is used to update the plant dynamics. The updated plant dynamics is further used by Nonlinear MPC to formulate optimal inputs to reach target.

B. Nonlinear MPC

Model Predictive Control is a very important tool that has been in use in the process industry since the 1980s. It is formulated as the repeated solution to an open-loop finite horizon optimal control problem subject to some state and input constraints along with the system dynamics. The basic principle of MPC is depicted in Figure 1. The algorithm takes in the measurements at time t and the controller predicts the system dynamics behaviour over the prediction horizon T_p. The control inputs are then determined such that the predetermined open-loop cost function is minimized. Provided there were no disturbance in the system and the optimization problem could be solved over an infinite horizon, the control inputs calculated at $t = 0$ could be applied to the system open loop for all $t \geq 0$. However, in practice, this cannot be done because of measurement noises and disturbance in the system. Hence, in order to incorporate feedback, the algorithm only implements the first control input from the optimal control input sequence and re-evaluates the optimal control problem for the new sampling instant $t+\delta$.

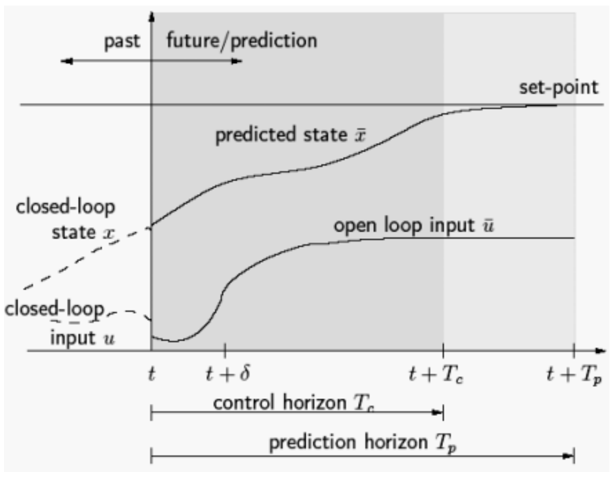

Fig. 2. Principle of model predictive control.

The optimal control inputs are determined based on predicted system behavior which allows the direct inclusion of state and input constraints as well as the minimization of the cost function. However, since only a finite prediction horizon is chosen, the predicted system behavior will in general differ from the closed loop one. Hence, precautions must be taken in order to achieve closed-loop stability and reasonable closed-loop performance. The overall process of NMPC can be summarized as follows:

1) The system states are estimated for the prediction horizon.
2) Solve the optimal control problem to determine the optimal control input which minimizes the cost function over the prediction horizon T_p.
3) Implement the first control input to evolve the system to the next sampling instant.

4) repeat from step (2).

C. Mathematical formulation of NMPC

Consider the class of continuous time systems described by the following nonlinear differential equation

$$\dot{x}(t) = f(x(t), u(t)), \quad x(0) = x_0 \tag{5}$$

subject to input and state constraints of the form:

$$\begin{aligned} u(t) &\in U, \forall t \geq 0, \\ x(t) &\in X, \forall t \geq 0. \end{aligned} \tag{6}$$

where $x(t) \in \mathbb{R}^n$ and $u(t) \in \mathbb{R}^m$ denote vectors of state and inputs, respectively.The input and state constraint sets are given by:

$$\begin{aligned} U &:= \{u \in \mathbf{R}^m \mid u_{\min} \leq u \leq u_{\max}\} \\ X &:= \{x \in \mathbf{R}^n \mid x_{\min} \leq x \leq x_{\max}\} \end{aligned} \tag{7}$$

with the constant vectors u_{min}, u_{max} and x_{min}, x_{max}.

As discussed in the previous section, the optimal control input is computed at every time step by solving the following optimal control problem:

$$\min_{\bar{u}(\cdot)} J(x(t), \bar{u}(\cdot)) \tag{8}$$

subject to:

$$\begin{aligned} &\dot{\bar{x}}(\tau) = f(\bar{x}(\tau), \bar{u}(\tau)), \bar{x}(t) = x(t) \\ &\bar{u}(\tau) \in U, \tau \in [t, t+T_c] \\ &\bar{u}(\tau) = \bar{u}(t+T_c), \tau \in [t+T_c, t+T_p] \\ &\bar{x}(\tau) \in X, \tau \in [t, t+T_p] \end{aligned} \tag{9}$$

with the cost functional

$$J(x(t), \bar{u}(\cdot)) := H(x(t+T_p)) + \int_t^{t+T_p} F(\bar{x}(\tau), \bar{u}(\tau)) \mathrm{d}\tau \tag{10}$$

Here T_p and T_c are the prediction and control horizon with $T_c \leq T_p$. The bar denotes internal controller variables and $\bar{x}(\cdot)$ is the solution of $Eq.(5)$ driven by control input signal $\bar{u}(\cdot) : [t, t+T_p] \to U$. It is important to distinguish the variables in the controller from real system variables because even in normal cases the predicted values will not be the same as the actual closed-loop values.

The cost function J comprises the terminal cost H and the stage cost F which often takes the quadratic form given by:

$$\begin{aligned} H(x(t_h)) &= (x(t_h) - x_s(t_h))^T P(x(t_h) - x_s(t_h)) \\ F(x, u) &= (x - x_s)^T Q(x - x_s) + (u - u_s)^T R(u - u_s) \end{aligned} \tag{11}$$

Here, $t_h = t+T_p$ which is basically the end time of the time horizon. Also, x_s and u_s denote a desired reference trajectory which can be a constant or time-varying. For the purpose of this project, we assume the stage and terminal costs to be constant, i.e., the system tries to reach the target at the end of every time horizon. Hence the stage and terminal cost transform as follows:

$$\begin{aligned} H(x(t_h)) &= (x(t_h) - x_{target}(t_h))^T P(x(t_h) - x_{target}(t_h)) \\ F(x, u) &= (x - x_{target})^T Q(x - x_{target}) + (u)^T R(u) \end{aligned} \tag{12}$$

Here, P and Q matrices should be positive semi-definite and R should be a positive definite matrix. The optimal solution of the problem defined above is denoted by $\bar{u}^*(\cdot; x(t)) : [t, t+T_p] \to U$. The optimal control problem is solved repeatedly at the sampling instants $t_j = j\delta$, $j = 0, 1, ...$ and the fist value of the optimal control sequence is fed to the system. Thus the nominal closed-loop system is given by:

$$\dot{x}(t) = f(x(t), \bar{u}^*(t; x(t_j))) \tag{13}$$

III. Problem Formulation

In this section, we focus on developing the system dynamics and defining the inputs and their constraints. We also define the noise acting on the system.

A. System Modelling

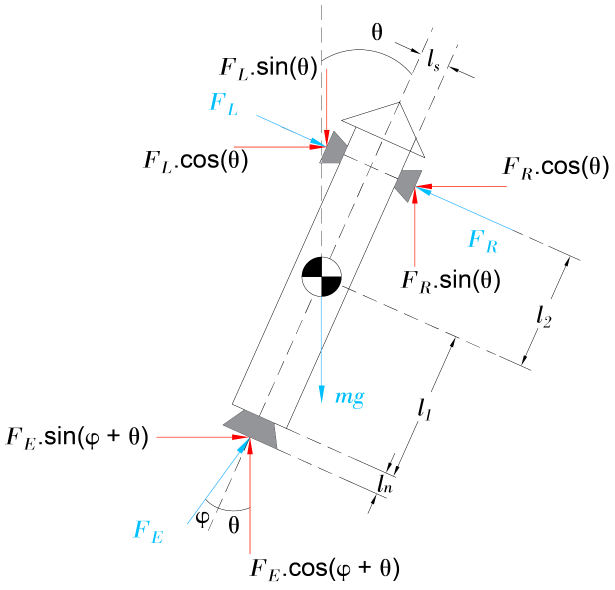

Fig. 3. Free Body Diagram of a Rocket.

Consider the rocket in Figure 3., illustrating the free-body diagram of the rocket when landing. Let

F_E = Main Engine Thruster Force
F_R = Right Thruster Force
F_L = Left Thruster Force
$F_S = F_L - F_R$
θ = Angle between the vertical and the longitudinal axis of the rocket
ψ = Angle between the Nozzle and the longitudinal axis of the rocket
l_1 = longitudinal length between the Center of Gravity (COG) and F_E

l_2 = longitudinal length between the COG and F_R, F_L
l_n = Nozzle length
m = Rocket dry mass + fuel mass
x = horizontal position of the rocket
z = vertical position of the rocket

We consider three force inputs in our system,

1) Main Engine Thrust, F_E
2) Side Thruster force, $F_S = F_L - F_R$
3) Nozzle Angle, ψ

The input limits we consider for the system are [1],

$$\begin{aligned} 0N < &F_E < 6486N \\ -130N < &F_S < 130N \\ -15^\circ < &\psi < 15^\circ \end{aligned} \tag{14}$$

To design appropriate controllers, we now make use of the system ODE [1],

$$m\ddot{x} = F_E \sin(\theta + \varphi) + F_S \cos(\theta)$$

$$\ddot{x} = \frac{F_E \cos(\varphi)\sin(\theta) + F_E \cos(\theta)\sin(\varphi) + F_S \cos(\theta)}{m} \tag{15}$$

$$m\ddot{z} = F_E \cos(\theta + \varphi) - F_s \sin(\theta) - mg$$

$$\ddot{z} = \frac{F_E \cos(\varphi)\cos(\theta) - F_E \sin(\varphi)\sin(\theta) - F_S \sin(\theta) - mg}{m} \tag{16}$$

$$J\ddot{\theta} = -F_E \sin(\varphi)\,(l_1 + l_n \cos(\varphi)) + l_2 F_S \tag{17}$$

The objective is to land the rocket at the target with minimum deviation and minimum effort in the presence of wind disturbance. We build upon (10),(11), and (12) to create our cost function. The cost function penalizes the time taken to reach the steady state and the inputs, reducing effort. Where the penalty matrices are,

$$P = 10^4 \begin{bmatrix} 1\,0\,0\,0\,0\,0 \\ 0\,1\,0\,0\,0\,0 \\ 0\,0\,1\,0\,0\,0 \\ 0\,0\,0\,1\,0\,0 \\ 0\,0\,0\,0\,1\,0 \\ 0\,0\,0\,0\,0\,1 \end{bmatrix}$$

$$Q = 10^4 \begin{bmatrix} 10\,0\,0\,0\,0\,0\,0 \\ 0\,1\,0\,0\,0\,0 \\ 0\,0\,1\,0\,0\,0 \\ 0\,0\,0\,1\,0\,0 \\ 0\,0\,0\,0\,1\,0 \\ 0\,0\,0\,0\,0\,1 \end{bmatrix}$$

$$R = \begin{bmatrix} 1\,0\,0 \\ 0\,1\,0 \\ 0\,0\,1 \end{bmatrix}$$

1) Wind Disturbances in X-Direction: We consider a Gaussian disturbance ($\omega(t)$) acting on the system due to the effect of wind acting on the rocket. The mean wind velocity of The wind disturbance is an unmeasured state disturbance that directly affects the x – position of the rocket. We consider a wind velocity of 7 m/s acting perpendicular to the rocket's surface. This is the average wind velocity observed around Cape Canaveral, Florida, where the Falcon 9 rockets land. The wind velocity of 7 m/s applies a wind load of approximately 300 N. This wind load is translated to the acceleration of 7 m/s^2 in the x – direction. To estimate an unmeasured disturbance, we use the NMHE technique.

B. Setting up the NMPC + NMHE solvers

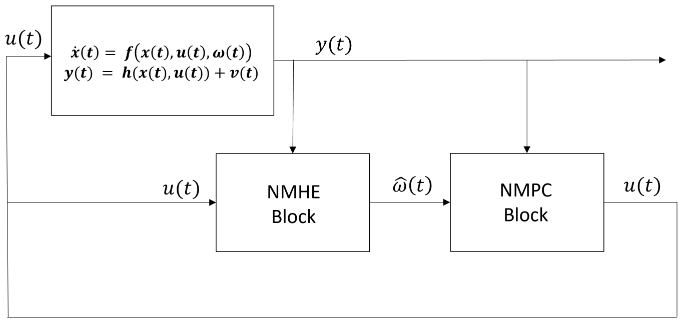
$u(t)$
$\dot{x}(t) = f(x(t), u(t), \omega(t))$
$y(t) = h(x(t), u(t)) + v(t)$
$y(t)$
$u(t)$
NMHE Block
$\hat{\omega}(t)$
NMPC Block
$u(t)$

Fig. 4. System Model Design.

The control algorithm which is necessary to achieve our objective needs to combine NMHE and NMPC to work in tandem (Refer to Fig. 4.). And since our work is based on simulation, we create a virtual environment mimicking the disturbance that the rocket will be subjected to. Though we know the disturbance acting on the rocket, we only use the disturbance to simulate a realistic environment and do not send this information to either NMHE or NMPC. The disturbance is only added to the system process, which provides only the state measurements to the NMHE and NMPC solver. As discussed in Section II, the NMHE solver finds the error in the system process and updates the NMPC system model with the estimated disturbance. Now, the NMPC solver holds the current measurements, updated system model, and cost function for which we find optimal control inputs. The optimal control input is then sent to the simulated system and the NMHE to complete the loop. The process is repeated till the target is achieved (Refer to Algorithm 1).

IV. RESULTS

The problem defined in the previous section was solved on MATLAB using the open-source CasADI solver [2] for Nonlinear Programming (NLP) problems. The CasADI solver is written in Python and C++. Hence to implement it in MATLAB, we used a control and estimation tool called MPCTools for linear and non-linear dynamics models. This solver is particularly faster than the native MATLAB solver *fmincon*. The optimization method used within CasADi is the IPOPT (Interior Point OPTimizer) which uses an interior point

Algorithm 1 NMHE-NMPC Algorithm

```
1: Define the system dynamics parameters
2: Initialize the start location of the rocket
3: Define the Target location for the rocket
4: Define the ODE
5: Redefine the ODE using Casadi Integrator for
   simulations
6: Redefine the ODE using Casadi Function for
   solvers
7: Define the stage cost, terminal cost using
   Casadi Function
8: Initialize the Gaussian noise
9: Initialize NMHE solver with parameters
10: Initialize NMPC solver with parameters
11: for $t$ in $1 : N_{SIM}$ do
12:     Compute $y(t)$ using $x(t)$
13:     Save $y(t)$ and $u(t-1)$ as new measurments for
        NMHE
14:     Solve NMHE
15:     Save $\bar{w}$ from NMHE solver
16:     Predict states $\bar{x}$ in $N_t$ using $\bar{w}$
17:     Save the predicted states to NMPC solver
18:     Solve NMPC for $u$
19:     Find $x(t+1)$ using $u$ from NMPC and actual $w$
20:     if $x(3, t+1) < 0.003$ then
            Break
21:     end if
22: end for
```

optimization method to solve for the optimal solution within the state and input constraints. We use MATLAB *2022.b* version on a *i9, 11th* gen Intel processor with 8 cores running at 3.4GHz of clock speed. The average run time per simulation is around 6.5 seconds, of which 0.5 secs is taken to build the NMHE solver, 0.3 seconds to build the NMPC controller, and 4.7secs on average to run each simulation.

A. *Unbounded with No Disturbances*

To test the soundness of our MATLAB model, we first ran the NMPC block simply with a no disturbances, unbounded state dynamics model. The target for our system was to reach the goal coordinates at the origin $(0, 0)$ with minimum control effort u. This is a very idealistic scenario, as the bounds on the control inputs have not been set yet. This allows the controller to find extreme control inputs that are not possible in a real-life scenario. As we can see in Figure 2, the control inputs are in the range of [0,6000N] for the main thruster F_e and [0,4000N] for the side thrusters F_s. This is a relatively large value for the side thrusters considering the design limits of our rocket. The difference we see in this scenario is that the rocket lands from the start location of [10,30] to the origin within 2.5 seconds.

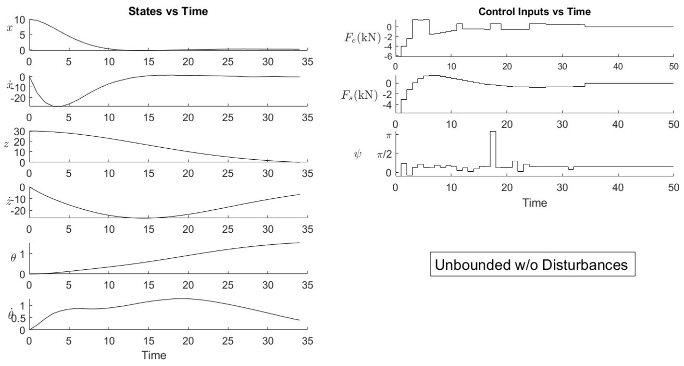

Fig. 5. System Information for Unbounded states and control inputs

B. *Bounded with No Disturbances*

For the next scenario, we bounded our states and control inputs according to [1] as follows:

$$[-2, -\infty, -0.1, -\infty, -pi, -\infty] \leq x \\ \leq [\infty, \infty, \infty, \infty, pi, \infty] \\ [0N, -130N, -pi/12] \leq u \leq [6486N, 130N, pi/12]$$

The control bounds are shown in Equation 14 as well. With the same start location $(10m, 30m)$ as before, we get the simulation plot as shown in Figure 6 and the respective control input data shown in Figure 7. As noted, the side thruster force F_s drops from a maximum value of 4 kN for the unbounded scenario, to the range of -130N to 130N for the bounded control inputs scenario. There is a significant difference in the simulation time as the bounded inputs scenario takes about 7 seconds for the rocket to land. Figure 7 shows the time step simulation of a rocket landing on the barge. The target location is the origin, although we have an acceptable range of $[-2m, 2m]$ on which the rocket can land. Moving forward for the last scenario, this will be our baseline simulation to which we apply the wind disturbances.

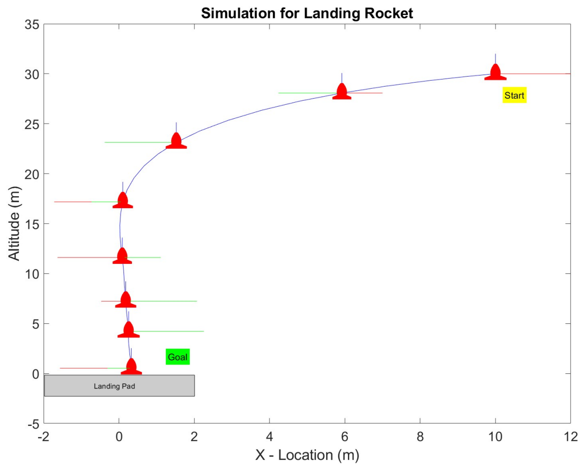

Fig. 6. Bounded States and Control Inputs Simulation

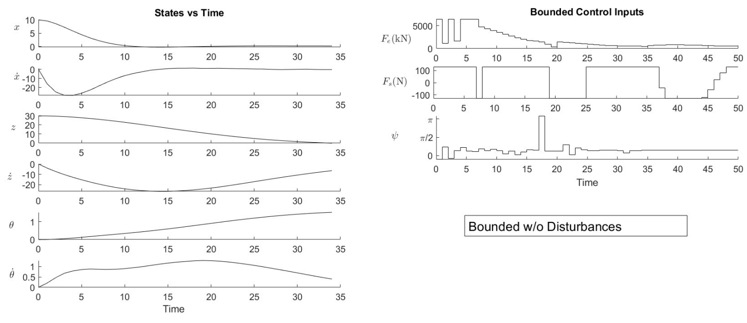

Fig. 7. Bounded States and Control Inputs Simulation

C. Bounded with Disturbances

The bounded state and control inputs with wind disturbances scenario are the closest to the real-life scenario we focus on. As discussed in Section III-A1, the wind velocity in the X direction acting on the x state is a bounded Gaussian noise $\sim N(\mu, \sigma) = N(7,3)m/s^2$. In a real-time application, the bounded disturbances are unknown to the plant model. Hence we are the NMHE solver to estimate the states using a moving horizon. The moving horizon estimates future x state, updates that as the predicted states for the NMPC solver, which in turn solves for an optimal control input $u*$. The results As we can

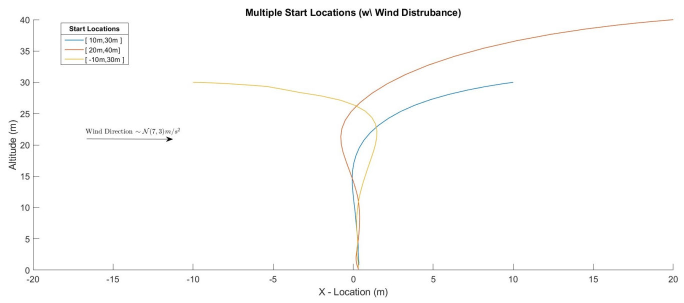

Fig. 8. Multiple Start Locations

note in Figure 8, the controller works accurately in bringing the rocket from multiple start locations to the origin. The wind disturbance acting on the trajectories can be clearly identified at around the $20m$ height. Even though the yellow and blue runs start from the same height, the trajectories made due the positive effect of the wind tend to push the rocket further away from the origin which the controller has to compensate for eventually.

We also compared multiple run simulations to understand the bounds of our disturbance. Even though from the perspective of this simulation, we have provided the noise values, in a real-life scenario we would expect the disturbances to be unknown to the plant model. Hence we use the NMHE solver to estimate the states as well as the wind disturbance to update the states $\hat{x}$ and disturbance $\hat{w}$. From figure 10, we created a bounded tube plot to understand the bounds of our simulation for multiple runs.

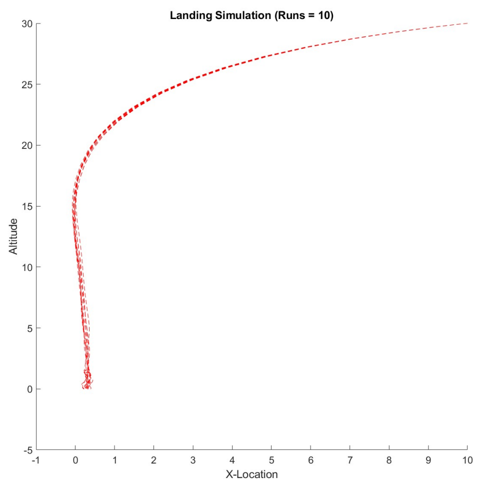

Fig. 9. 10 Simulation runs with Disturbance

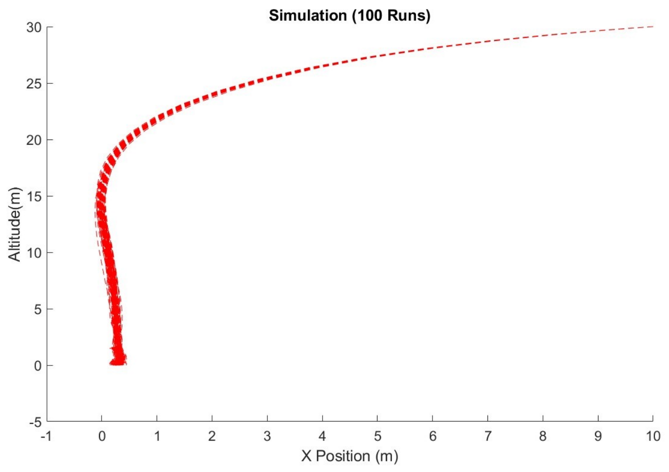

Fig. 10. 100 Simulation Runs with Disturbance

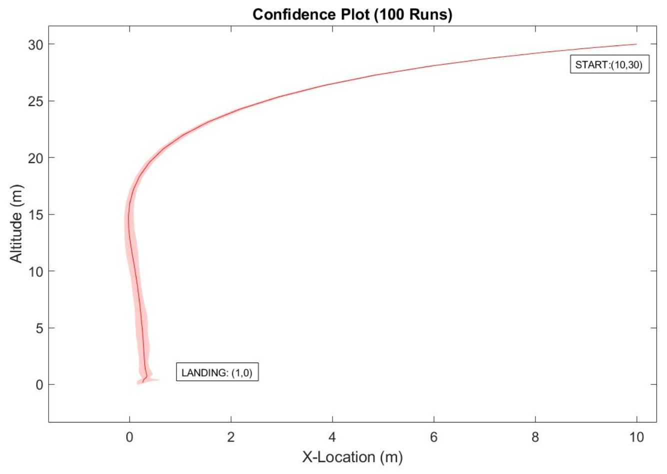

Fig. 11. Confidence Plots (For 100 Runs)

In Figure 11, we can see that the end coordinates of the tube plot still lie in the maximum landing range of the barge $(-2, 2)m$. This shows that the controller thus designed works optimally and robustly as well.

V. Conclusions and Future Work

In this project, we were successfully able to mode a 6 state, 3 input rocket lander system with non-linear ODE's and solved the task of landing the rocket safely onto a barge. We employ Non-linear Model Predictive Control methods to solve for the non-linear dynamics and minimize a set cost function. The cost function minimises the control input to the system (i.e. the main engine, side thrusters and the engine angle) as well as ensures that the rocket reaches the target location. We have applied our control system to 3 scenarios as described above that test the stability, optimality, and robustness of the system. In the first 2 scenarios, we compare a no wind disturbance model for bounded and unbounded state and control inputs. In the final scenario, we try to assume a Gaussian wind disturbance in the positive X direction acting on the state x as the constant load on the side of the rocket. To predict the states and the disturbances, we use a Non-linear Moving Horizon Estimate that estimates and updates the $\hat{x}$ states and $\hat{w}$ disturbances and considers that as the predicted prior states for the NMPC solver. The controller can handle the disturbances well and land the rocket even in multiple run simulations. Finally, we compare the results between the 3 scenarios and discuss more about each one of them.

There is quite a lot of scope for future work on this project. We have considered employing a 3D dynamics model with a higher fidelity simulation environment for this case. Another focus area could be to develop a model with higher dependencies within each state and focus on the stability of the system considering we are using a non-linear MPC solver. Optimization using Interior point methods can be a challenging task for 3D polytopes when solving for an optimal solution and working on that can be a task when we employ 3D dynamics.

References

[1] Ferrante, Reuben. "A robust control approach for rocket landing." Master's thesis (2017).

[2] CasADi solver "https://web.casadi.org/"

[3] Allgower, Frank, Rolf Findeisen, and Zoltan K. Nagy. "Nonlinear model predictive control: From theory to application." Journal-Chinese Institute Of Chemical Engineers 35.3 (2004): 299-316.

[4] Moving horizon estimation scheme.png. (2022, November 30). Wikimedia Commons, the free media repository. Retrieved 15:56, December 12, 2022 from "https://commons.wikimedia.org/w/index.php/title=File: Moving_horizon_estimation_scheme.png oldid=710987118."

[5] Johansen, T. A. (2011). Introduction to nonlinear model predictive control and moving horizon estimation. Selected topics on constrained and nonlinear control, 1, 1-53.

Acknowledgment

Aditya Parameshwaran contributed by conducting literature review for the rocket lander model, setting up the CasADi solver in MATLAB with the state dynamics, cost functions and solving the baseline unbounded problem. He also worked on developing all the plots for the 3 scenarios and has written the Introduction, Results and Conclusions for the report. Shahil Shaik has contributed by applying the state and control bounds to model along with integrating the NMPC and NMHE codes. He has written the theory for NMPC in the report and developed the pseudo code for our system. Akshit Lunia has contributed by developing the NMHE solver and writing the theory for it, and also written and designed the diagrams for the Problem Formulation with Sections III-B and III-A and the Abstract for the report.