Stochastic Model Predictive Control of Rocket Dynamics

Jack Hanling and Gustavo Espinosa Garcia

Georgia Institute of Technology, Atlanta, Georgia, 30313

In light of increased computational capabilities of the 21st century, Stochastic Model Predictive Control (SMPC) has become a more relevant and potentially revolutionary technique for linear and nonlinear control. This paper proposes a start-to-finish optimal control using SMPC for rocket dynamic control. It runs through the formulation of the dynamics of the system, controller optimization with consideration of uncertainties and measurement errors using a modified exterior penalty method, and control optimization with gradient descent. This framework is then applied to the rocket following a specified trajectory. This paper yields encouraging results in the application of SMPC to nonlinear control, specifically in the field of rocket control.

Variables

$g = 9.81\ m/s$ = Gravitational Constant

m = Mass of Rocket

$x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$ = Position Vector

$\dot{x}$ = Velocity Vector

$a = \ddot{x}$ = Acceleration Vector

$\sigma = \begin{bmatrix} \theta \\ \psi \\ \phi \end{bmatrix}$ = Orientation Vector

$\dot{\sigma} = \frac{d\sigma}{dt}$ = Angular Velocity

$\ddot{\sigma} = \frac{d^2\sigma}{dt^2}$ = Angular Acceleration

e_1, e_2 = Gimbal Angles

$|F_T|$ = Thrust Vector Magnitude

$u = \begin{bmatrix} e_1 \\ e_2 \\ |F_T| \end{bmatrix}$ Control Variables Vector

$\hat{x}_{\Delta t=i}$ = Position Vector Approximation Using Eulers Method at time step $\Delta t = i$

F_T = Thrust Induced Force Vector

F_D = Drag Induced Force Vector

M_{Thrust} = Thrust Induced Moment Vector

M_{Drag} = Drag Induced Moment Vector

C_M = The distance of the COM from the base of the rocket

D = Diameter of the Rocket

r = Radius of the Rocket

H = Height of the Rocket

Introduction

Until the 1960's, control theory mainly delt with the 3-parameter Proportional-Integral-Derivative (PID) controller which is still used across industries in the modern day [1]. This type of control system is suitable for relatively simple linear systems, however modern-day advancements in engineering and manufacturing often require more sophisticated methods to optimize efficiency. In engineering it is often the goal to create methods to control nonlinear systems, however controlling nonlinear systems is proven to be difficult due to their chaotic nature, requiring a highly dynamic controller.

One potential technique for controlling a system with nonlinear dynamics is the use of Model Predictive Control (MPC).

MPC uses the dynamical equation of the system to predict future states for a specified time interval in the future called the "horizon" [3]. Control parameters that force the system to follow a specified 'trajectory' are then calculated using a constrained Optimal Control Problem (OCP) [3]. The first of the series of control steps is executed, and then the process is repeated, moving the horizon up one time step, and starting from new measured state [3].

Early work on MPC resulted in the evolution of Robust Model Predictive Control (RMPC), which relies on deterministic descriptions of the system with bounded uncertainties [4]. An early form of RMPC was based on a 'min-max' principle, where the control actions were dictated by the 'worst case scenario" with regards to the constraints of the system, resulting in overly conservative control actions [4]. To deal with the more realistic case of continuous, probabilistic uncertainties, Stochastic MPC (SMPC) was developed. This method trades the bounded constraints for 'chance constraints' with specified probability level requirements (i.e. Probability of Failure metrics) [4]. This new approach allows for the control system to be violated but only by their pre-defined 'chance-constraint', allowing for a far more flexible control design [5]. Further, disturbances and uncertainties can be identified as stochastic variables and implemented into the OCP [5].

In this paper we develop a SMPC algorithm framework -- from developing the nonlinear equations describing rocket dynamics, to implementing an exterior penalty method to account for stochastic constraints, to designing the gradient descent solver for generating the necessary control steps. This framework will then be implemented on a controlled rocket trajectory problem, whereat the rocket will start at a specified altitude, and must guide itself along a defined trajectory for a specified time. The rocket uses thrust magnitude and gimbal angles for trajectory control. The gimbal angles are known to have some uncertainty, and this uncertainty is implemented into a modified exterior penalty method. Overall, this experiment was a success with numerous potential future directions for improving the method proposed here.

Theory

There are three components of theory – the nonlinear dynamics of the rocket, linearization of the nonlinear system for computing efficacy, and then stochastic model predictive control where the uncertain control variables that are being assessed are the gimbal angles e_1 and e_2. Part one of this section outlines the equations used to model the rocket dynamics. Part two outlines the linearization technique used when approximating the time horizon for the model predictive control algorithm. Part three discusses the mathematics behind the model predictive control algorithm that is implemented. Part four discusses the optimization function used, and the modified exterior penalty function used to account for variance of the true gimbal angles. A high-level view of the algorithm used in this paper for calculating SMPC optimal control values and stepping forward is shown in the state diagram in Figure 1.

The equations established in rocket dynamics subsection specify the Thrust Force and Moment in addition to the Drag Force and Moment. These are used in MATLAB functions, and then Controlled_Acceleration_Update() is used to calculate what the acceleration of displacement and orientation are, given state variables. For calculating the optimal next step, the acceleration at its current state is calculated as a function of the control variables:

$$u = \begin{bmatrix} e_1 \\ e_2 \\ |F_T| \end{bmatrix}$$

Eulers method is then used to step the function forward in time five times at .1 second intervals, giving SMPC a time horizon of .5 seconds. This is fundamental to model predictive control. Approximating the future state based on the current state, and then selecting control variables that help guide the future state to the desired state the best. To calculate this, the future states calculated via Euler Method are:

$$\hat{x} = \begin{bmatrix} \hat{x}_{\Delta t=.1} \\ \hat{x}_{\Delta t=.2} \\ \hat{x}_{\Delta t=.3} \\ \hat{x}_{\Delta t=.4} \\ \hat{x}_{\Delta t=.5} \end{bmatrix} \text{ where } \hat{x}_{\Delta t=i} = \begin{bmatrix} \hat{x} \\ \hat{y} \\ \hat{z} \end{bmatrix}_{\Delta t=i}$$

$$\hat{\sigma} = \begin{bmatrix} \hat{\sigma}_{\Delta t=.1} \\ \hat{\sigma}_{\Delta t=.2} \\ \hat{\sigma}_{\Delta t=.3} \\ \hat{\sigma}_{\Delta t=.4} \\ \hat{\sigma}_{\Delta t=.5} \end{bmatrix} \text{ where } \hat{\sigma}_{\Delta t=i} = \begin{bmatrix} \hat{\theta} \\ \hat{\psi} \\ \hat{\phi} \end{bmatrix}_{\Delta t=i}$$

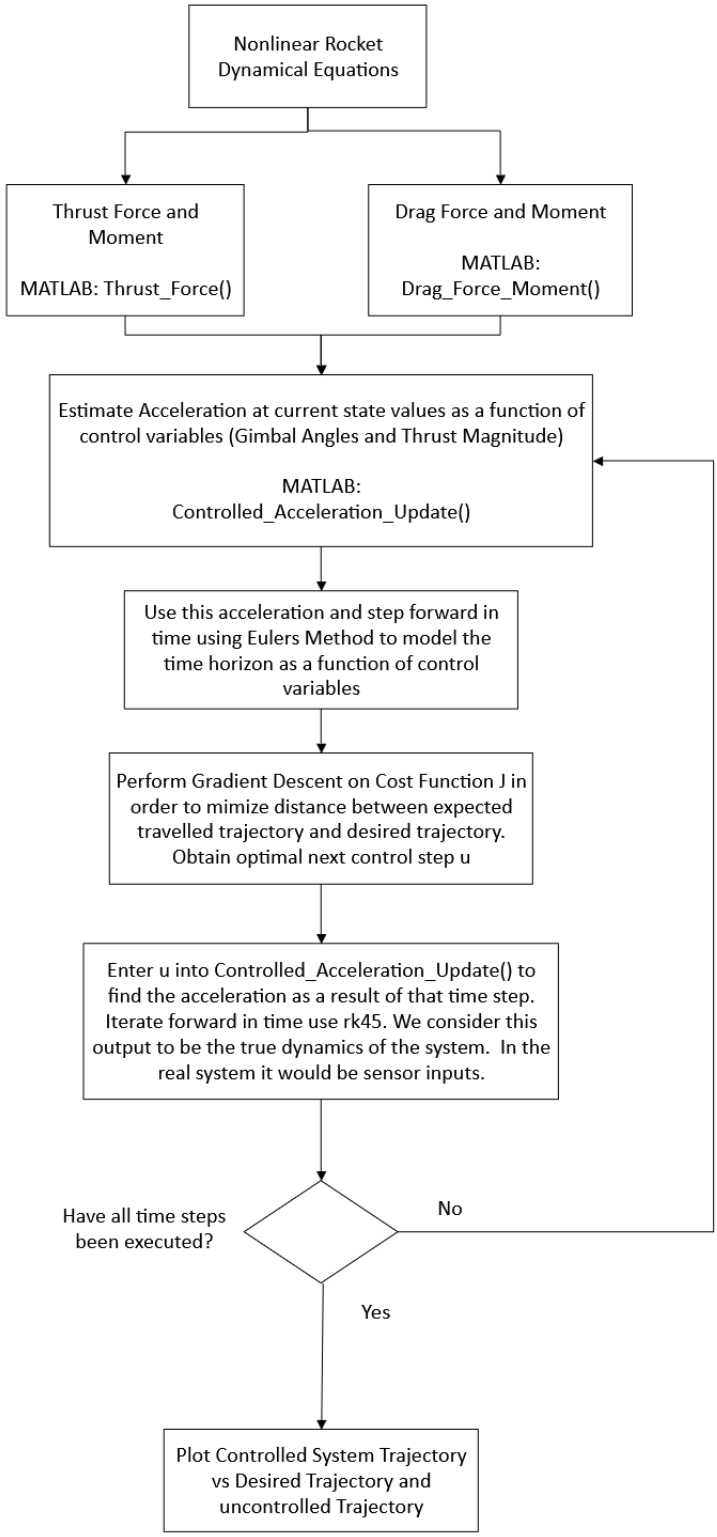


Figure 1: High Level Algorithm for How SMPC is Implemented

Accelerations are calculated using the Controlled_Acceleration_Update() at its current state variables σ, $\dot{\sigma}$, and $\dot{x}$. Symbolic variables for each time step forward $(u_1 \dots u_5)$ are substituted into the Controlled_Acceleration_Update() function which partially linearizes the system of equations and allows us to iterate them forward using Eulers method at a relatively low computational cost. There is then a cost function, J, specified in section III, which is minimized using gradient descent, and finds the optimal combination of control variables. Then, only the first control variable corresponding to $\hat{x}_{\Delta t=.1}$ is output and used by the system to control the dynamics. RK45 uses this control step, iterates .1 second forward in time, and then sends in the new state coordinates into the optimizer function Control_Step_Value(), and the system repeats until the desired number of time steps are completed.

i. Rocket Dynamics

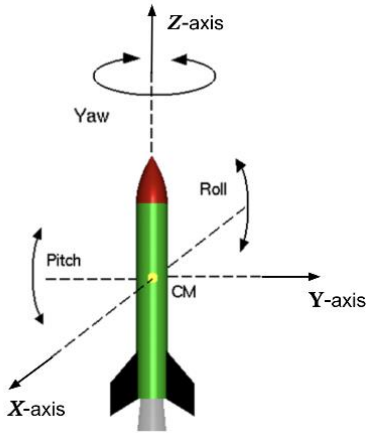


Figure 2. Global Coordinate System of Rocket Example [6] (*Image Modified)

SMPC will be used on nonlinear rocket dynamics in this paper. Rocket dynamics were modelled using the three main forces acting on the system – Force of Gravity, Drag Force, and Thrust Force. The differential equation describing the dynamics of the rocket is:

$$ma = -mg + F_D + F_T$$

The rockets orientation can be described by pitch, yaw, and roll. These are defined in Figure 1. The engine Gimbal Angles, e_1 and e_2 are collinear with Yaw and Pitch Respectively.

Drag Induced Force:

The rotation matrices for Yaw, Pitch, and Roll are as follows:

$$R_{yaw} = \begin{bmatrix} \cos(\Psi) & -\sin(\Psi) & 0 \\ \sin(\Psi) & \cos(\Psi) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$R_{pitch} = \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) \end{bmatrix}$$

$$R_{roll} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi) & -\sin(\phi) \\ 0 & \sin(\phi) & \cos(\phi) \end{bmatrix}$$

$$R = R_{yaw} R_{ptich} R_{roll}$$

Rockets longitudinal axis can be calculated via: $l = Rl_0$

Where $l_0 = [0,0,1]^T$

l_0 represents the original orientation of the rocket, as shown in Figure 2, where the vector from the base of the rocket to the top is [0;0;1]. All orientations will be taken with respect to this coordinate system to ensure consistency.

Angle between the rocket's longitudinal axis and the rockets velocity vector:

$$\cos(\alpha) = \frac{\dot{x}}{|\dot{x}|} \cdot l$$

This variable, α ,provides insight as to the direction that the rocket is being hit by air, and therefore allows for us to determine the cross-sectional area of the rocket in contact with the air. In this case, we make the simplifying assumption that the air is stagnant (i.e. there are no wind gusts, only the velocity of the rocket induces drag).

$$A = \pi DH|sin(\alpha)| + \frac{\pi}{4} D^2 |\cos(\alpha)|$$

Thrust Induced Force:

Thrust force can be calculated in a similar way, although the thrust vector also has Gimbal angles which influence the direction of thrust relative to the orientation of the rocket. Multiplying all the rotational matrices allows for the thrust direction with respect to the original orientation to be computed. It should be noted that R_{e1} and R_{e2} are of the same form as R_{Pitch} and R_{Roll} respectively, since they are in the same axes.

$$R_{e1} = \begin{bmatrix} \cos(e_1) & 0 & \sin(e_1) \\ 0 & 1 & 0 \\ -\sin(e_1) & 0 & \cos(e_1) \end{bmatrix}$$

$$R_{e2} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(e_2) & -\sin(e_2) \\ 0 & \sin(e_2) & \cos(e_2) \end{bmatrix}$$

$$R = R_{pitch} \cdot R_{e1} \cdot R_{yah} \cdot R_{roll} \cdot R_{e2}$$

$$l_{Thrust} = R \cdot \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

The thrust vector can be computed as the magnitude of the thrust multiplied by the direction of the thrust vector.

$$F_T = |F| \cdot R \cdot \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

Thrust Induced Moment:

We must implement equations describing how the Thrust and Drag Force Vectors induce moments on the rocket. The location of the COM with respect to the base of the rocket is distance C_M

For thrust, the moment will be:

$$M_{Thrust} = \ lC_M \ sin(\cos^{-1}(l \cdot l_{Thrust})) \times T$$

Recall that l = $l_0 R$, thereby giving the position of the rocket with respect to l_0

Drag Induced Moment:

For simplicity, the center of pressure is at the center of the rocket. The center position of the rocket is 70/2 = 35 m above the base of the rocket. Therefore, the induced moment will be:

$$F_D = \ -.5 \ \rho C_D \dot{x}^2 \left[\pi DH|\sin(\alpha)| + \frac{\pi}{4} D^2 |\cos(\alpha)|\right]$$

$$M_{Drag} = l(35 - C_M)\sin(\alpha) \times F_D$$

where $\cos(\alpha) = \frac{\dot{x}}{|\dot{x}|} \cdot l$

Displacement Accelerations:

Displacement Acceleration can be calculated as follows:

$$a = \begin{bmatrix} \ddot{x}_1 \\ \ddot{x}_2 \\ \ddot{x}_3 \end{bmatrix} = \frac{-mg + \ F_D + F_T}{m}$$

Orientation (Angular) Accelerations:

Angular Acceleration can be computed as follows:

$$RI \begin{bmatrix} \ddot{\theta} \\ \ddot{\psi} \\ \ddot{\phi} \end{bmatrix} = M_{Thrust} - M_{Drag}$$

$$\ddot{\sigma} = \begin{bmatrix} \ddot{\theta} \\ \ddot{\psi} \\ \ddot{\phi} \end{bmatrix} = (RI)^{-1}(M_{Thrust} - M_{Drag})$$

Where $I = \begin{bmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{yx} & I_{yy} & I_{yz} \\ I_{zx} & I_{zy} & I_{zz} \end{bmatrix}$

The above nonlinear differential equation will be linearized for the MPC algorithm to reduce computational cost and iterated forward in time using Euler Method to create the 'Horizon'. The true dynamics of the system will be modeled by solving this nonlinear differential equation one step to the next using RK45.

ii. Linearization

This section concerns the state diagram in Appendix 1.A. Control step variables for each time step along with the current state conditions $(\sigma, \dot{\sigma}, \dot{x})$ are used to compute the accelerations $\ddot{x}(u_i)$ and $\ddot{\sigma}(u_i)$ which are a function of the control variables. Calculating the acceleration once for the state variables and then substituting in the control variables at each time step results in a linearization of the nonlinear equations, and therefore also a high likelihood of error. For this reason, we only look five-time steps into the future, as any further is likely to yield incorrect approximations of the future, which could be more detrimental than helpful to our optimizer. Euler method is then used to step the method forward in time.

Euler Method

$$\dot{\hat{x}}_{k+1} = \dot{\hat{x}}_k + \Delta t \cdot \ddot{\hat{x}}_k$$

$$\hat{x}_{k+1} = \hat{x}_k + \Delta t \cdot \dot{\hat{x}}_k$$

$$\dot{\hat{\sigma}}_{k+1} = \dot{\hat{\sigma}}_k + \Delta t \cdot \ddot{\hat{\sigma}}_k$$

$$\dot{\hat{\sigma}}_{k+1} = \hat{\sigma}_k + \Delta \mathrm{t} \cdot \dot{\hat{\sigma}}_k$$

Where $\ddot{\hat{x}}$ and $\ddot{\hat{\sigma}}_k$ are computed using the current state variables $(\sigma, \dot{\sigma}, \dot{x})$ and are function of the control variables. When five iterations of the Eulers Method is computed, mammoth equations describing the variables $\hat{x}_1 \ldots \hat{x}_5$ as functions of the control variables $[u_1 \ldots u_5]$ are generated.

iii. Model Predictive Control

Model Predictive Control Formula is of the general format from [7]:

$$J(k) = \sum_{j=1}^{N} \left|\left|r(k+j|k) - y(k+j|k)\right|\right|_{Q_j}^2 + \sum_{j=0}^{N_u-1} \left|\left|\Delta u(k+j|k)\right|\right|_{R_j}^2$$

Where

$r(k+j,k)$ = reference vector

$y(k+j|k)$ = predicted output vector

$\Delta u(k+j|k)$ = Input Increment vector

The goal is to minimize the difference between the approximated trajectory, x_i, for a given time horizon N, and similarly set up a cost function for changes in the control step u, which punishes large variance in control steps, which allows for a smoother sequence of control steps.

The control formula is inspired by this model.

Define: $u_i = \begin{bmatrix} e_{1,i} \\ e_{2,i} \\ |F_T|_i \end{bmatrix}$

As the control vector at time horizon i. The cost function we implement is:

$$J = \sum_{i=1}^{N} \left[w_{Traj} \left(\hat{x}_i - x_{des_i} \right)^2 + w_{F_{var}} (|F_T|_i - |F_T|_{i-1})^2 + w_{e_{var}} \left(\left(e_{1,i} - e_{1,i-1} \right)^2 + \left(e_{2,i} - e_{2,i-1} \right)^2 \right) \right]$$

Where $|F_T|_0,\ e_{1,0},\ e_{2,0}$ = The previous respective control steps taken by the RK45 method. This way the variance does not deviate too much from the previous control step as well. The weights were calibrated to be as follows:

$w_{Traj} = 500$ → Large Weight to Ensure it stays close to trajectory

$w_{F_{var}} = 10^{-7}$ → Thrust is a large value. Variance should be larger to allow for larger control steps needed to converge.

$w_{e_{var}} = .1$ → Gimbal Angle have small magnitude and therefore their variance must be amplified in J function

These weights were experimentally derived. Future work can focus on finding an empirical way of determining optimal weights for this problem.

iv. Optimization and Stochastic Constraints of Loss Function (J)

A large outline of the gradient descent method used in this paper can be found in Appendix 1.B. The first control step is based on the past executed control step – this is the initial value for u when computing the gradient of J. J is a function of all of the control variables u_i, a total of 15 independent variables. The u vector is reformatted as follows:

$$u = \left[|F_T|_1, |F_T|_2, |F_T|_3, |F_T|_4, |F_T|_5, e_{1,1}, e_{1,2}, e_{1,3}, e_{1,4}, e_{1,5}, e_{2,1}, e_{2,2}, e_{2,3}, e_{2,4}, e_{2,5} \right]$$

Where u_i now refers to index i of this vector.

J's derivative is taken with respect to each of these control variables to find its gradient.

$$Grad = \nabla J = \frac{\partial J}{\partial u} = d$$

Where d is our search direction.

Since the gimbal angles are assumed to have some stochastic uncertainty with respect to what their actual angle is, a Monte Carlo Simulation (MCS) is used to find the probability of failure (P_f). Probability of failure that exceeds 20% is punished severely, where the e1 and e2 control steps are raised to a large power and manually added into the gradient. The algorithm is as follows

1. Run 1000 monte Carlo simulations.
2. Using the average e1 and e2 values given by the sensors, influence them with the noise, and then determine the percent that fails (exceed pi/2 radians)

If $P_f > .20$:

$$d_i = d_i + u_i^{19}$$

This excessively large cost function will force the gimbal angles to return within the limits. In order to ensure that these gimbal angles do not 'overshoot', normalization is used on the gradient for the gimbal angles.

$$d_i = \frac{d_i}{||d_{6:10}||} \quad for \quad i \in [6,10]$$

$$d_i = \frac{d_i}{||d_{11:15}||} \quad for \quad i \in [11,15]$$

Finally, the gradient is multiplied by a weighted step function alpha:

$$\alpha = [10^6, 10^6, 10^6, 10^6, 10^6, 1,1,1,1,1,1,1,1,1,1]$$

This ensures that the thrust values are take large steps, as the thrust value is often in the millions and in order to get close to convergence in 10000 steps, must take aggressive steps. These values of α were experimentally determined.

To update the u values every step, the following is executed:

$$u = u + \alpha * d$$

This gradient descent is the run a maximum of 10000 times, or when the norm of the gradient reaches less than 1. This allows a decently optimal value to be converged too without having the computer spend too much time computing it.

Example Problem

In recent years the consumer space, satellite, and defense industries have fostered a growing demand for the refined design and development of rocket systems. This has led to the aerospace applications of MPC based control methods becoming a major subject of interest in recent years. The ability to control rocket dynamics effectively is a difficult task and as such rocket examples are often used to benchmark the performance of a sophisticated control system. Control models often rely on commonly accepted approximations of the general dynamics as well as frequently collected sensor data for feedback. The exact values of the respective sensor measurements seldom are reflected in the raw data captured by the sensor as there exists unavoidable uncertainty resulting from environmental noise. SMPC controllers have shown promise in developing a method of implementing an MPC in a system subject to stochastic uncertainty. SMPC controllers optimize the input from the present time to the time horizon H, minimizing the difference between actual and desired trajectory while enforcing hard limits using probability of failure to ensure the likelihood of a catastrophic disaster remains low. In this example an SMPC controller will be implemented for the purpose of optimally controlling a theoretical rocket system.

The example problem this paper will address is as follows. Given a rocket with the following characteristics:

Rocket Characteristics

Mass: m = 100,000 kg

Initial Velocity: $v_0 = \begin{bmatrix} 1 \\ 0 \\ 10 \end{bmatrix}$ m/s

Air Density: $\rho = 1.2 \frac{kg}{m^3}$

Coefficient of Drag of Rocket: $C_D = .75$

Diameter of Rocket: D = 3.667 m

Height of Rocket: H = 70 m

Center of Mass of Rocket (Measured from base of the rocket): C_M = 30 m

Initial Position (w.r.t Ground): $x = \begin{bmatrix} 0 \\ 0 \\ 1000 \end{bmatrix} m$

Initial Orientation (Angles): $\sigma = \begin{bmatrix} .01 \\ .01 \\ .01 \end{bmatrix} radians$

Previous Thrust Magnitude: $F_{T_{prev}} = 1{,}000{,}000\ N$

Previous Gimbal Angles: $e_{1_{prev}} = 0\ radians$ & $e_{2_{prev}} = 0\ radians$

Time Step: $dt = .1\ second$

Period of Execution: 2.5 seconds (25 iterations)

We want the rocket to follow the trajectory:

Desired Trajectory: $\begin{Bmatrix} x(t) \\ y(t) \\ z(t) \end{Bmatrix} = \begin{bmatrix} .1 * t \\ .1 * t \\ 1000 + 1 * t \end{bmatrix} meters$

Given the following constraint of the Gimbal Angles: $e_1 < \frac{\pi}{2}$ and $e_2 < \frac{\pi}{2}$

During preliminary testing of the rocket, the engineers discovered that at any given Gimbal angle the actuators set it to be, there is an approximately normal distribution about that value with a standard deviation of .1 radians. In other words:

$$e_{1_{True}} = N\left(e_{1_{set}}, .1\right) radians$$

$$e_{2_{True}} = N\left(e_{2_{set}}, .1\right) radians$$

Because of this error, engineers are fearful that when using Gimbal angles near its limit, an error could result in breaking the rocket. For this reason, you will need to account for the probability of failure (i.e. $e_1 > \frac{\pi}{2}$ *or* $e_2 > \frac{\pi}{2}$), such that any e values with a probability of failure greater than 20% must be eliminated to avoid failure of the rocket. There are sensors on board that will update you every .1 seconds of what the new state variables are ($[\sigma, \dot{\sigma}, x, \dot{x}]$).

Using the theory established in the previous section, we developed state diagrams (as shown in appendix) and respective MATLAB code to simulate this. Since we do not have a real rocket to obtain the "Updated State Variables" ($[\sigma, \dot{\sigma}, x, \dot{x}]$) every .1 seconds from our sensors, we use an RK45 method in MATLAB in order to create a better approximation of what the true dynamics of the rocket look like in MATLAB, and consider those dynamics to be the true trajectory travelled. These values deviate from Stochastic Model Predictive Control approximations of the state of the system at future time slightly as Euler Method is less accurate but faster than RK45. A time horizon of 5 steps of .1 second length will be used. This is to compromise between the need to see farther in the future to use MPC more effectively, and with the computational cost of doing so with a nonlinear system along with the increasing error from Eulers Method.

Results

i. Gradient Descent Regression Verification

To test the gradients descents regression ability to minimize the error between the desired trajectory and controlled trajectory, one iteration of control step was run. A maximum of 10,000 gradient descent steps were taken, and the algorithm had the capability of stopping early if the norm of the gradient was less than 1. Adaptive step size was used where $\alpha = 10^6$ for thrust gradient and $\alpha = 1$ for gimbal angle gradient. The graph shows the convergence of the controlled trajectory with the desired trajectory for one step. Note that the time stepping algorithm used here is Eulers Method, which is less accurate than RK45. We therefore used RK45 as a baseline of what the 'true' dynamics of the system in replacement for experimental data that would be obtained by sensors on board.

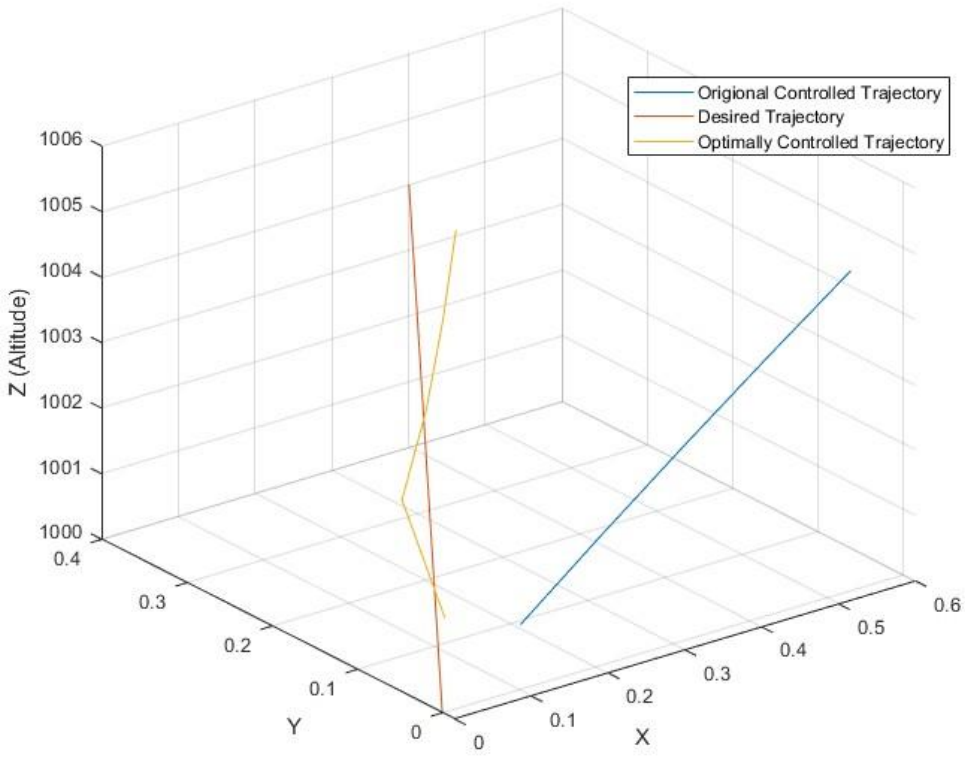


Figure 3. Estimated System Response for One Time Horizon

The algorithm takes in information from the previous control step and uses Euler's method to determine the next five position vectors as a function of the five input control vectors u. The resulting estimated position of the system at each time step within the horizon is represented by the $x_1, \dots, x_5$ vectors. Since the control input is determined using an approximation of the dynamics, the effectiveness of the resulting optimized input is highly dependent on the ability of the approximation method to provide a valid representation of the system response. In order to confirm the validity of our gradient descent method to select optimal u values that follow the desired trajectory, we refer to Figure 3. This figure shows the trajectory using the pervious control steps (before Gradient Descent) and the trajectory using the optimal control steps (after Gradient Descent). The optimally controlled trajectory clearly results in a closer alignment to the desired trajectory. The Mean Squared Error (MSE) of the originally controlled trajectory is 1.104, compared to MSE of the optimally controlled trajectory .3230. The reduction in MSE demonstrates Gradient Descents Capability of selecting optimal control values u. It is worth noting that the error still existing between desired and optimally controlled trajectory can arise from two places. First, the gradient descent has a limit of 10,000 iterations. If it is unable to converge to an optimal solution, larger errors may occur. Further, the function that is being optimized by gradient descent is the MPC loss function J, which is composed of more than just the error between desired and controlled trajectories, meaning that the optimal trajectory isn't necessarily exactly the desired trajectory.

ii. Justification of Simulation Results

For the full simulation of the system is expanded to explore the performance of the SMPC. The system was run for 25 iterations of .1 s time steps representing 2.5 seconds of the rocket response. During reach .1 second time step, gradient descent was used to find an optimal next control value, while the stochastic exterior penalty method was used to ensure adheres to the Gimbal Angle Limits.

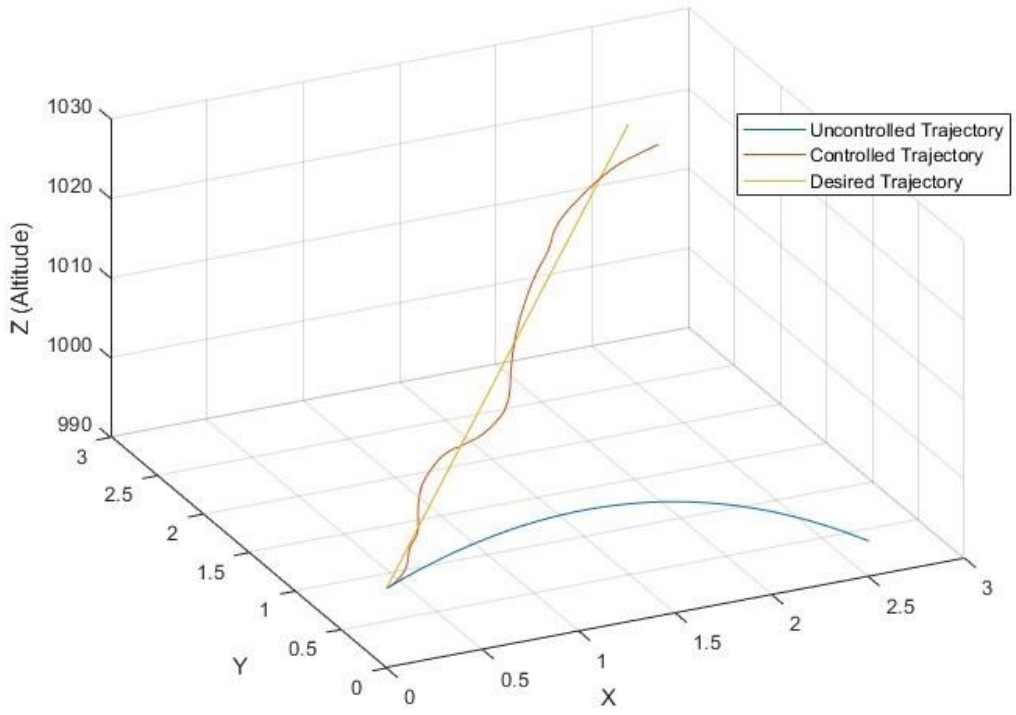


Figure 4. System Response of 25 time-steps

Figure 4 shows the result of iterating forward 2.5 seconds with the controlled trajectory, desired trajectory, and uncontrolled trajectory. As can be seen in the figure, the controlled trajectory follows the desired trajectory significantly more accurately than the uncontrolled trajectory. This is further exemplified with the MSE errors with the controlled trajectory MSE calculated to be 1.8371×10^3 and the uncontrolled trajectory MSE calculated to be 1.9622×10^5. This represents a massive improvement and a 99% reduction in total MSE further demonstrating the system's capabilities of producing a valid control scheme.

iii. Comparison Between Stochastic and Deterministic Optimums

The stochastic method used in this paper was designed to account for uncertainty of the position of the gimbal angles and restrict the angle if it approaches its angle limit in this case set to be $\left|\frac{\pi}{2}\right|$. If the optimum position angle was achievable by the rocket's actuators, it is possible to treat the gimble angles as a deterministic variable allowing the restriction of the values to be achieved by an additional constraint on the loss function. However, in reality, the rocket's actuators are subject to uncertainty therefore maintaining the angles within the acceptable bounds requires the use of the modified Stochastic Exterior Penalty Method. The method results in optimal gimbal angle outputs which demonstrate a failure probability of less than 20% estimated using MCS. The figure below shows the gimbal's angle steps vs time when the Stochastic Exterior Penalty Method when active as well as inactive.

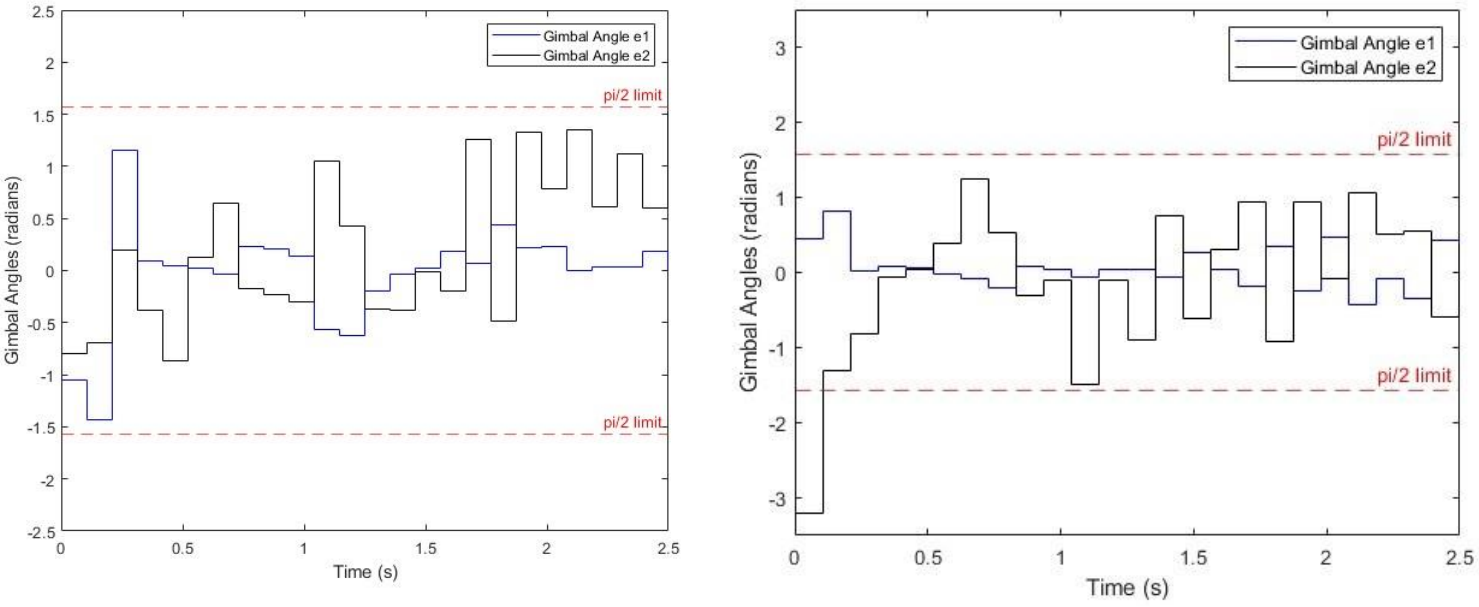


Figure 5. Gimbal angles over time. *Left; Stochastic Exterior Penalty Active. Right; No intervention*

Comparatively, when the SEPM is not applied, the Gimbal angle exceeds $-\frac{\pi}{2}$ in pursuit of reducing thrust. This angle request violates the failure criteria and represents an unrealistic application of thrust. This demonstrates that the SEPM can determine when the stochastic variables of gimble angle violate the permitted probability of failure and effectively apply an additional penalty within the loss function in order to keep all angles in the simulation within the limit bounds.

Discussion

The use of the SMPC in the application of the rocket example problem shows encouraging performance of the controlling algorithm. The gradient method worked effectively to converge the solution to a relatively optimal control input values u. Exemplified in Figure 3, the optimized controlled trajectory follows the desired trajectory substantially better than the non-optimized controlled trajectory. The effectiveness of gradient descent was further

demonstrated by the reduction of the MSE from 1.104 to .3230 when using gradient descent to optimize the control step for the five-step time horizon. This was effective in the use of our problem, although better algorithms could be implemented. Variable step size α could have been implemented into the gradient descent to enhance convergence. Optimal α values could have been computed using Golden Method. Likewise, Conjugate gradient is known to have quicker convergence than Gradient descent, which could be implemented in the future to find a more optimal u value and potentially speed up the gradient descent section of the code. Newtons method could also have some potential use in this optimization problem, although computation of the Hessian may prove to be computationally costly.

The rocket's trajectory was shown to follow the desired trajectory impressively well, however deviation from the path remains evident throughout the flight. It was apparent from Figure 4 that there was periodic overshooting as the rocket attempted to get closer to the desired trajectory. This could be a symptom of multiple possible causes, although several potential solutions could be approached in future work. First, smaller time step could allow for more refined control, potentially allowing it to converge more smoothly and prevent overshooting. Another method of reducing this overshooting would be to extend the length of the time horizon, and potentially use a more accurate method than Eulers method. A small-time horizon and Eulers method were used in this paper simply as a means to reduce the time it takes to compute the optimal control step. If access to a more powerful computer or a more effective linearization technique were implemented, these suggestions may be possible to implement. A stricter adherence to the desired path could be accomplished through additional modifications of the weighting factors utilized in the loss function. The weighting factors were obtained through trial-and-error analysis until a combination of weights that resulted in acceptable performance were found. Other methods to determine weighting factors in optimization problems have been proposed and the implementation of a more robust technique, such as the Taguchi Method [8], would most likely result in better system performance.

SMPC used in this paper used gradient descent, Euler Method, and RK45 all to find an optimal control step and one .1 second time step. These algorithms add additional layers of computation complexity for each time step and therefore significantly affect the total computation time for the simulation. In our simulation of 25 time-steps (or a total simulation time of 2.5 seconds), the computation time was about 5-7 minutes with additional time steps increasing the total time accordingly. This significantly slowed down the weight refinement process in our model and is unacceptable in terms of real-life implementation. For the SMPC to be implemented in a real system, the computation time must be sufficiently less than the time-step size. This problem has been likewise noted throughout the current literature. Future work looking into methods of reducing computational cost of computing the time horizon would be admirable, although computing nonlinear equations forward in time is inherently a costly method, hence why the use of SMPC has been mostly limited to linear systems.

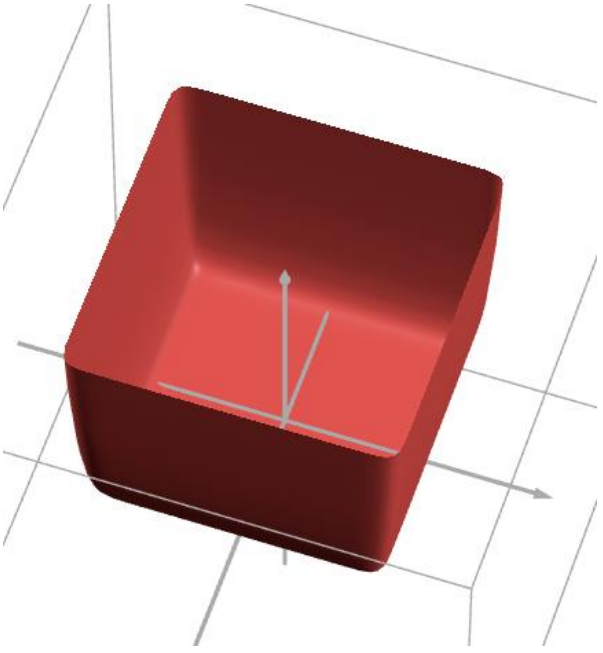


Figure 6. Function $z = x^{20} + y^{20}$

Finally, the modified Stochastic Exterior Penalty Method, which is the stochastic component of our SMPC, appears to have performed its duty with the data we gather in this run according to Figure 5. The first step without the Stochastic Exterior Penalty Method resulted in the Gimbal angle going beyond the threshold, likely in attempt to apply a reverse thrust, which a rocket is not capable of doing. The Gimbal angles with the stochastic exterior penalty method resulted in the Gimbal angle staying marginally below the limit, likely due to the requirement set by the P_f limit. Typically, in the Exterior Penalty Method, the penalty is multiplied by a values such as 10, and then that value is raised to a power every time the limit function is violated, thereby eventually forcing the optimization algorithm to follow the limit function requirement again. This requires adding the value to the gradient and computing the gradient every time a gradient descent is performed, which would be extremely computationally costly for our algorithm to do. Our algorithm computes the gradient with respect to the control variables, and then converts that gradient into a MATLAB function, which the u values can be substituted into in order to increase speed of computation. Consequently, we used a modified approach, whereat we calculate the gradient of u^{20}, which we approximate to u^{19}. When the P_f value is violated, the u function, which for the gimbal angles will be $\frac{\pi}{2} > 1$, will be raised to the

19^{th} power resulting in an exorbitantly, likely to outweigh any other gradient magnitude. This gradient function is graphed in Figure 6 for reference. This is then added to the gradient of the Gimbal Angle control value that violated its constraint. The Gimbal Angles are then renormalized, which ensures that their direction vectors fall within the values of 0 and 1, ensuring that overshooting does not occur because of this. This method effectively acts in the same way that the traditional Exterior Penalty Method would, only with more aggressive punishment for exceeding the P_f limit, and has the added benefit that the gradient function does not need to be remade, thereby reducing computational cost of computing the gradient every time.

Conclusion

In this paper we used SMPC to effectively control a rocket's trajectory while accounting for a stochastic limit. The method has shown ability to account for stochastic uncertainty in control variables, utilize an approximation of system behavior, and be adapted to provide optimal control inputs within the bounds of the probability of failure. The SMPC control algorithm in this paper produced very encouraging results as the desired trajectory was followed by the rocket while keeping the stochastic variables within the acceptable range of values. Despite the necessary linear approximation and lower accuracy Euler method used in computing the time horizon, the optimized controls could still produce a significant decrease in MSE compared to the undesired trajectory. The successful implementation of SMPC in this example, as well as the adaptability of this method, emphasizes the strength of SMPC based control methods. There are several potential suggestions we offer in the discussion for future work that can be done to improve our method in terms of optimization techniques, method of linearization and computing forward in time, extending the time horizon and using smaller time segments. This is an exciting field bringing together optimization and stochastic limits, with numerous potential applications.

Contributions

Jack & Gus: Worked concurrently on generating the matlab code, debugging of code, running simulations, and worked on Abstract, Introduction, Theory, Example Problem, Result, Discussion, Conclusion, and edits to the paper. This was a very difficult problem we set out to do so we were constantly working together to overcome challenges.

References

[1] Holkar, K. S., and Waghmare, L. M., "An Overview of Model Predictive Control" International Journal of Control and Automation, Vol. 3, No. 4, 2010, pp. 47–64.

[2] Kaiser, E., Kutz, J. N., and Brunton, S. L., "Sparse identification of nonlinear dynamics for model predictive control in the low-data limit," *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 474, Nov. 2018, p. 20180335.

[3] González, E., Sanchis, J., García-Nieto, S., and Salcedo, J., "A comparative study of stochastic model predictive controllers," Electronics, vol. 9, Dec. 2020, p. 2078.

[4] Mesbah, A., "Stochastic model predictive control: An overview and perspectives for future research," *IEEE Control Systems*, vol. 36, Dec. 2016, pp. 30–44.

[5] Farina, M., Giulioni, L., and Scattolini, R., "Stochastic linear model predictive control with chance constraints – a review," *Journal of Process Control*, vol. 44, Aug. 2016, pp. 53–67.

[6] Benson, T., "Practical rocketry," NASA Available: https://www.grc.nasa.gov/www/k-12/rocket/TRCRocket/practical_rocketry.html.

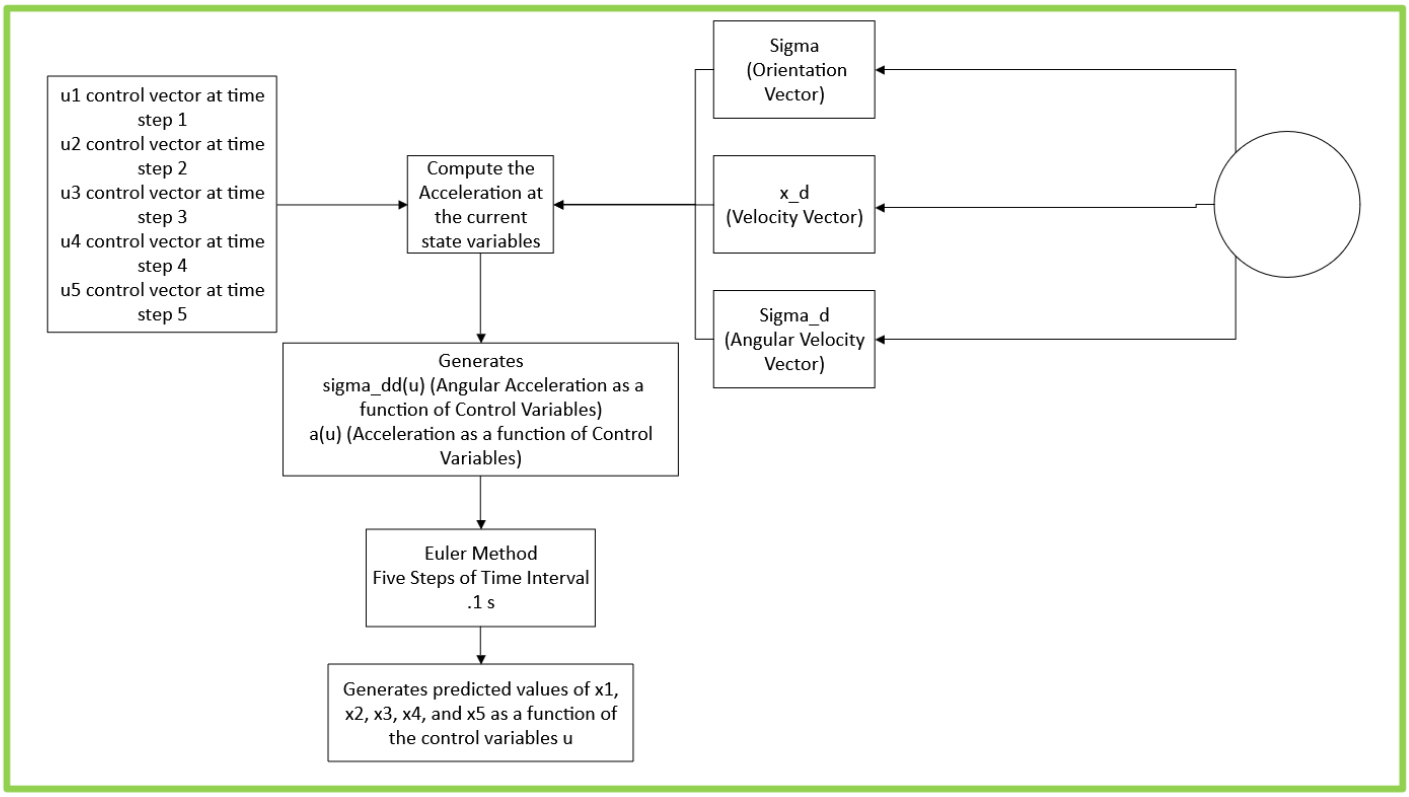
[7] Yan, Z., and Wang, J., "Model predictive control of nonlinear systems with unmodeled dynamics based on feedforward and recurrent neural networks," IEEE Transactions on Industrial Informatics, vol. 8, Nov. 2012, pp. 746–756.

[8] Freddi, A., and Salmon, M., "Introduction to the taguchi method," *Springer Tracts in Mechanical Engineering*, Jul. 2018, pp. 159–180.

Appendix

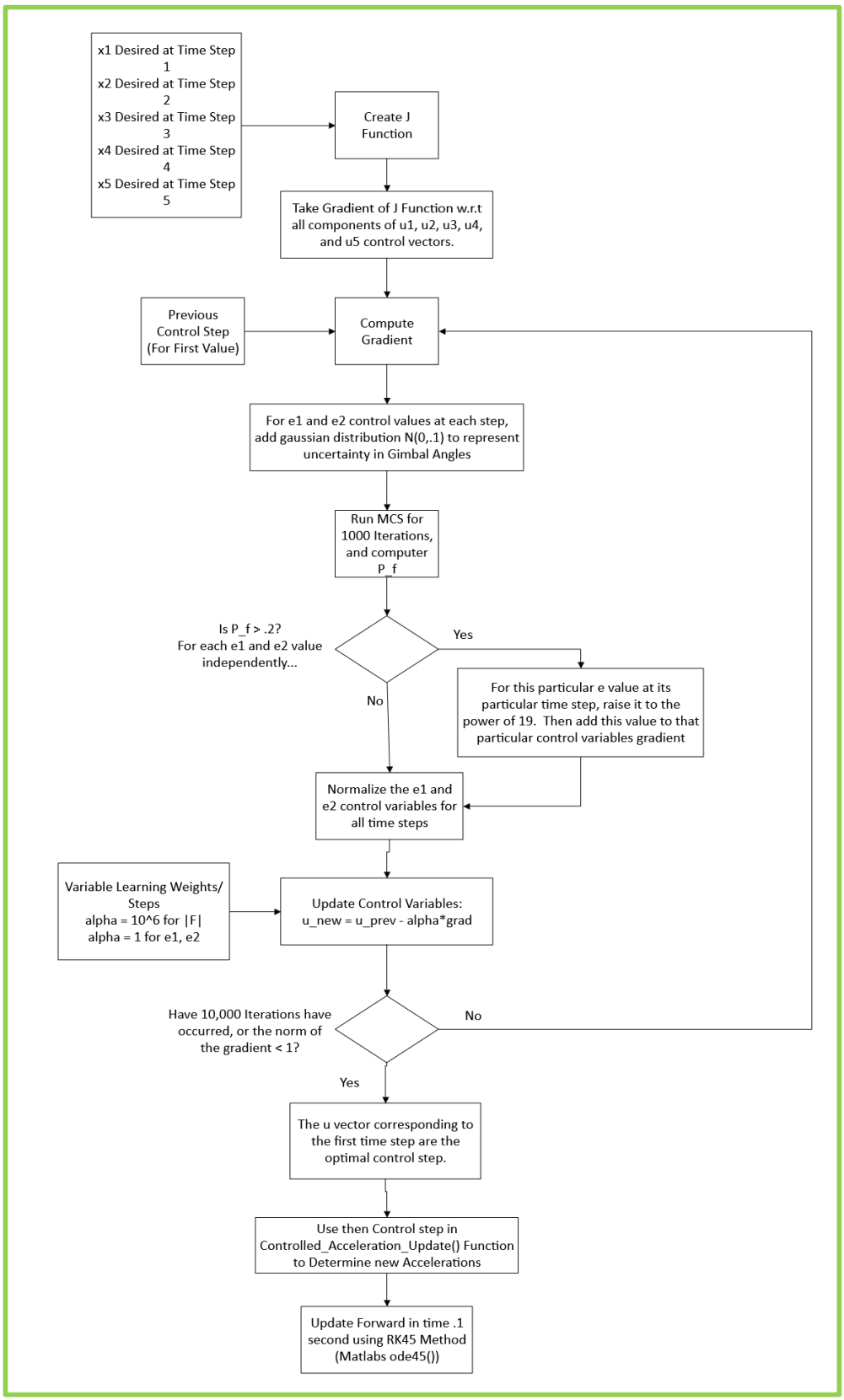
1.A

Linearization and Eulers Method



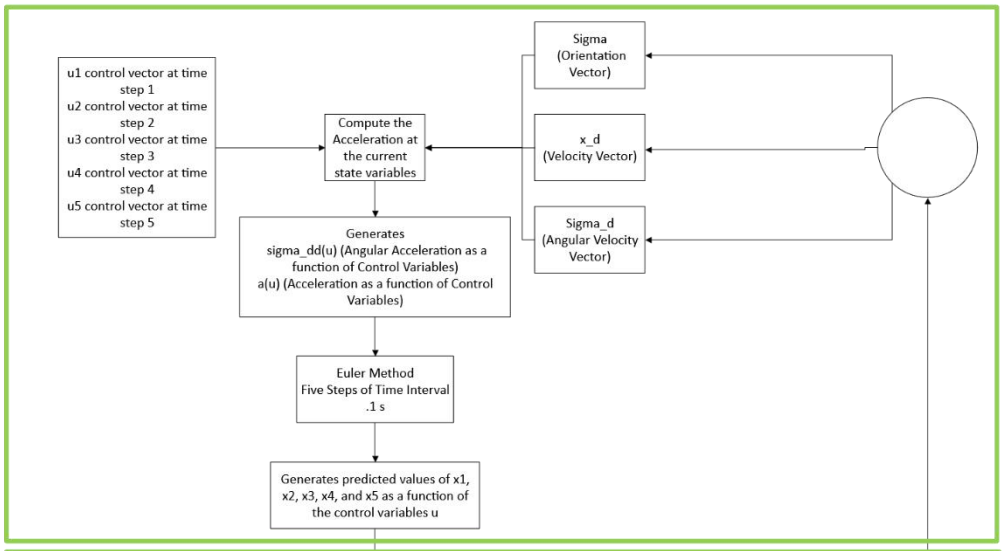
1.B

Model Predictice Control using Gradient Descent and a Stochastic Exterior Penalty Function for Gimbal Angles



1.C

Linearization and Eulers Method



Model Predictice Control using Gradient Descent and a Stochastic Exterior Penalty Function for Gimbal Angles