Learning Based Model Predictive Control for Vertical Take-off And Landing Aircraft

Thesis submitted

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE AWARD OF THE DEGREE OF

Bachelor of Technology

in

Electronics and Communication Engineering

Submitted by

Prajwal Thakur

(17JE002847)

Under the supervision of

Prof. Mrinal Sen IIT(ISM) Dhanbad)
Prof. Harikumar Kandath (IIIT Hyderabad)

Department of Electronics Engineering

Indian Institute of Technology (Indian School of Mines),Dhanbad-826004

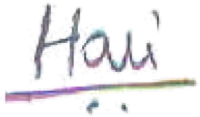
May 2021

इलेक्ट्रॉनिक्स अभियांत्रिकी विभाग
DEPARTMENT OF ELECTRONICS ENGINEERING
भारतीय प्रौद्योगिकी संस्थान (भारतीय खनि विद्यापीठ), धनबाद
INDIAN INSTITUTE OF TECHNOLOGY (INDIAN SCHOOL OF MINES), DHANBAD
धनबाद-826004, झारखण्ड, भारत / DHANBAD-826004, JHARKHAND, INDIA

CERTIFICATE

This is to certify that the **Prajwal Thakur** (Roll No. 17JE002847), student of B.Tech. (ECE), Department of Electronics Engineering Indian Institute of Technology (Indian School of Mines),Dhanbad has worked under my supervision and completed his Project work entitled "**Learning Based Model Predictive Control for Vertical Take-off And Landing Aircraft** " in partial fulfilment of requirements for the award of degree of **Bachelor of Technology** in **Electronics and Communication Engineering** from Indian Institute of Technology (Indian School of Mines)Dhanbad.

This work has not been submitted for any other degree ,award, or distinction else whereto the best of my knowledge and belief. He is solely responsible for the technical data and information provided in this work..



Prof. Harikumar Kandath
External Supervisor ,
International Institute of Information Technology
Hyderabad
India -500032

Prof. Mrinal Sen
Internal Supervisor,
Indian Institute of Technology
(Indian School of Mines),**Dhanbad**
India - 826004

FORWARDED BY:

Head of the Department,
Department of Electronics Engineering
Indian Institute of Technology
(Indian School of Mines), Dhanbad

Contents

List of Figures

Chapter 1

Introduction

1.1 Over-View

There exist many Control Algorithms today , and It depends on the application what control algorithm to use. PID controller is unarguably is widely used, but also possess many challenges in designing. To overcome these challenges and design Controllers for aviation application, optimization-based control techniques are most preferred, like MPC controller. Vertical Takeoff and Landing Aircraft possess the quality of both of a helicopter fixed-wing aircraft dynamics. It is challenging to design a controller for such aircraft because it has hybrid dynamics consisting of hover dynamics, transitional dynamics, and fixed-wing dynamics.Designing a PID controller in such cases is far more difficult than a MPC controller , and therefore ofter MPC controller is used in such cases. MPC controller have few issues as well , Its performance rely on the availability of an accurate predictive model and design choices like weights and cost function . Un accurate or poor design choice can degrade the performance of the controller. Therefore To increase the performance of the controller We investigate the ways to combine MPC controller with online/offline learning based technique to design a better controller. This project investigate how

to design such Learning-Based MPC controller in fixed wing Dynamics region of a VTOL.

1.2 Literature Survey

There is five flight profile for VTOL aircraft, Each one of them has different dynamics (Yuksek et al., 2016) , namely hover dynamics like a helicopter, fixed-wing dynamics(Yuksek et al., 2016), and transitional dynamics(Yuksek et al., 2016). Aerodynamic forces that will act on these three dynamics are also different, and there are different ways to calculate them. One way to model the vehicle in CAD and then apply and observe its aerodynamic effects over a simulation.Ozdemir et al. (2014). Another Way is to model the forces acting from first principle (ie from newton law of physics , fluid dynamics)(Govdeli et al., 2019). In this Project, I have used the first principle, previous data to model the Forces torques and Dynamics for VTOL in MATLAB. There are so many ways to design controllers for VTOL; unarguably PID controller is widely used, and we can also use the PID controller here. (ref:pid-wikipedia). Designing the PID controller is tiresome and requires a lot of effort when designing the controller for a large system like VTOL. There is a method known as successive loop control (uavBook chapter 6), widely used in aerial vehicle control. Nevertheless, despite the success of PID controller, we have a lot of drawbacks also in it some of them are: 1.When we have to design the controller with multiple constraints; designing becomes tedious. 2.Tuning the hyper parameters to match the desired performance is tedious, and required a lot of knowledge from Vehicle dynamics. To overcome these challenges and outperform the PID controller's performance, we have used the Linear MPC controller (ref:Stanislaw H. Żak -PID lecture). This controller is based on optimization. In this type of controller, we aimed to minimize some cost. The designing of this cost is crucial as its hyper parameters affects the performance. MPC controller is very robust to noises , but has some drawbacks too, This controller depends on the Prediction Model of the System

. It is very Important to correctly design this prediction model . Learning Based Controller (Hewing et al., 2020) overcome this challenge by learning the Prediction Model online , while controlling the aircraft ,of-course we have to provide initial guess of the prediction model. This type of controller is also some times also known as Model Based Reinforcement learning.

Chapter 2

VTOL dynamics

2.1 VTOL flight profile

Following figure shows different flight profile of Figure 2.1 VTOL :

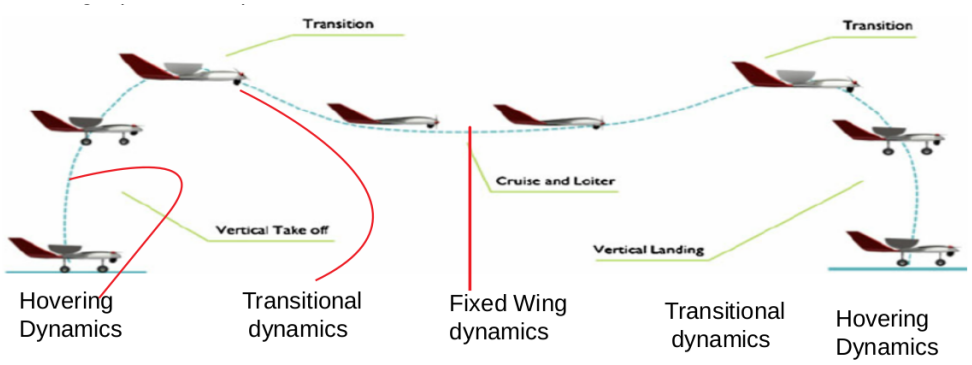
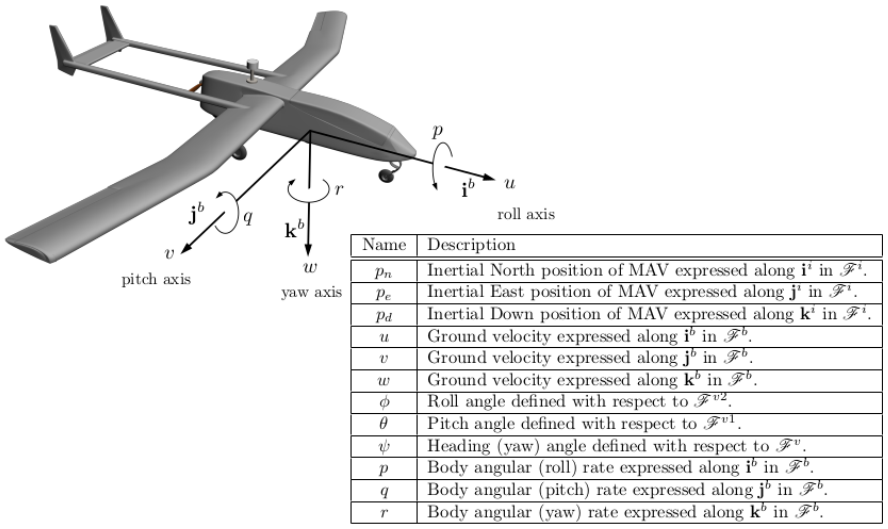


FIGURE 2.1: flight profile (Hewing et al., 2020)

- Hovering ; Behave like a Helicopter
- Transition
- Fixed wing Dynamics Behave as a Fixed wing UAV

2.2 Kinematics Equation

Aircraft State Variables There is 12 states variable for an aircraft that has been taken into consideration in this Thesis .Following Figure Figure 2.2 taken from "Beard and McLain,Small Unmanned Aircraft,Chap 3" shows these 12 states Variable. Out of these 12, 5 states Variables has been taken into consideration as a



Name	Description
p_n	Inertial North position of MAV expressed along $\mathbf{i}^i$ in $\mathcal{F}^i$.
p_e	Inertial East position of MAV expressed along $\mathbf{j}^i$ in $\mathcal{F}^i$.
p_d	Inertial Down position of MAV expressed along $\mathbf{k}^i$ in $\mathcal{F}^i$.
u	Ground velocity expressed along $\mathbf{i}^b$ in $\mathcal{F}^b$.
v	Ground velocity expressed along $\mathbf{j}^b$ in $\mathcal{F}^b$.
w	Ground velocity expressed along $\mathbf{k}^b$ in $\mathcal{F}^b$.
ϕ	Roll angle defined with respect to $\mathcal{F}^{v2}$.
θ	Pitch angle defined with respect to $\mathcal{F}^{v1}$.
ψ	Heading (yaw) angle defined with respect to $\mathcal{F}^v$.
p	Body angular (roll) rate expressed along $\mathbf{i}^b$ in $\mathcal{F}^b$.
q	Body angular (pitch) rate expressed along $\mathbf{j}^b$ in $\mathcal{F}^b$.
r	Body angular (yaw) rate expressed along $\mathbf{k}^b$ in $\mathcal{F}^b$.

Beard & McLain, "Small Unmanned Aircraft," *Princeton University Press*, 2012, Chapter 3: Slide 2

FIGURE 2.2: Aircraft State Variables

longitudinal State Variables , These are (u, w, q, θ, p_d)

Equation of Motion The six-degree-of-freedom , 12 state model for the UAV kinematics and dynamics are summarized as follows Figure 2.3, and detailed explanation and derivation is given in "Beard McLain,Small Unmanned Aircraft,Chap 3".

In these Equations (u,v,w) are the speed of UAV in Body frame of refrence , (p_n, p_e, p_d) are defined in Inertial frame of reference (f_x, f_y, f_z) are the forces experience by UAV in body frame of reference , (l,m,n) are the torque experienced by UAV in (x^b, y^b, z^b) body axis respectively .

The equations of motion are a system of 12 first order ODE's:

$$\begin{pmatrix}\dot{p}_n\\ \dot{p}_e\\ \dot{p}_d\end{pmatrix} = \begin{pmatrix}\mathrm{c}_\theta\mathrm{c}_\psi & \mathrm{s}_\phi\mathrm{s}_\theta\mathrm{c}_\psi - \mathrm{c}_\phi\mathrm{s}_\psi & \mathrm{c}_\phi\mathrm{s}_\theta\mathrm{c}_\psi + \mathrm{s}_\phi\mathrm{s}_\psi\\ \mathrm{c}_\theta\mathrm{s}_\psi & \mathrm{s}_\phi\mathrm{s}_\theta\mathrm{s}_\psi + \mathrm{c}_\phi\mathrm{c}_\psi & \mathrm{c}_\phi\mathrm{s}_\theta\mathrm{s}_\psi - \mathrm{s}_\phi\mathrm{c}_\psi\\ -\mathrm{s}_\theta & \mathrm{s}_\phi\mathrm{c}_\theta & \mathrm{c}_\phi\mathrm{c}_\theta\end{pmatrix}\begin{pmatrix}u\\ v\\ w\end{pmatrix}$$

$$\begin{pmatrix}\dot{u}\\ \dot{v}\\ \dot{w}\end{pmatrix} = \begin{pmatrix}rv - qw\\ pw - ru\\ qu - pv\end{pmatrix} + \frac{1}{\mathsf{m}}\begin{pmatrix}f_x\\ f_y\\ f_z\end{pmatrix},$$

$$\begin{pmatrix}\dot{\phi}\\ \dot{\theta}\\ \dot{\psi}\end{pmatrix} = \begin{pmatrix}1 & \sin\phi\tan\theta & \cos\phi\tan\theta\\ 0 & \cos\phi & -\sin\phi\\ 0 & \frac{\sin\phi}{\cos\theta} & \frac{\cos\phi}{\cos\theta}\end{pmatrix}\begin{pmatrix}p\\ q\\ r\end{pmatrix}$$

$$\begin{pmatrix}\dot{p}\\ \dot{q}\\ \dot{r}\end{pmatrix} = \begin{pmatrix}\Gamma_1 pq - \Gamma_2 qr\\ \Gamma_5 pr - \Gamma_6(p^2 - r^2)\\ \Gamma_7 pq - \Gamma_1 qr\end{pmatrix} + \begin{pmatrix}\Gamma_3 l + \Gamma_4 n\\ \frac{1}{J_y}m\\ \Gamma_4 l + \Gamma_8 n\end{pmatrix}$$

where

$$\Gamma_1 = \frac{J_{xz}(J_x - J_y + J_z)}{\Gamma}, \quad \Gamma_2 = \frac{J_z(J_z - J_y) + J_{xz}^2}{\Gamma}, \quad \Gamma_3 = \frac{J_z}{\Gamma},$$
$$\Gamma_4 = \frac{J_{xz}}{\Gamma}, \quad \Gamma_5 = \frac{J_z - J_x}{J_y}, \quad \Gamma_6 = \frac{J_{xz}}{J_y},$$
$$\Gamma_7 = \frac{(J_x - J_y)J_x + J_{xz}^2}{\Gamma}, \quad \Gamma_8 = \frac{J_x}{\Gamma}, \quad \Gamma = J_x J_z - J_{xz}^2.$$

Beard & McLain, "Small Unmanned Aircraft," *Princeton University Press*, 2012, Chapter 3: Slide 15

FIGURE 2.3: Kinematics Equation

2.3 Forces And Moments

In the Figure 2.3 we have mentioned (f_x, f_y, f_z) and (l,m,n) . In this section we will compute the these force and moment vector for a particular type of VTOL.

TURAC VTOL UAV In this Thesis I chose TURAC VTOL Figure 2.4 for designing the controller and further study because of the available detailed study of its forces,moments, aerodynamic properties.

Forces on TURAC The Figure 2.5 shows the Total forces and Moments on TURAC VTOL

Forces along x-axis:

$F_x = F_T^X + F_{Ax} - mg\sin(\theta)$

$F_T^x = (T_{f1} + T_{f2})\cos i_t$

where i_t corresponds to the tilt angle of two front rotors

$F_{Ax} = F_{Ax}^{fs} + F_{Ax}^{wake}$

$F_{Ax}^{fs} = C_x^{fs}\tilde{q}^{fs}(A - 2 * A_s)$

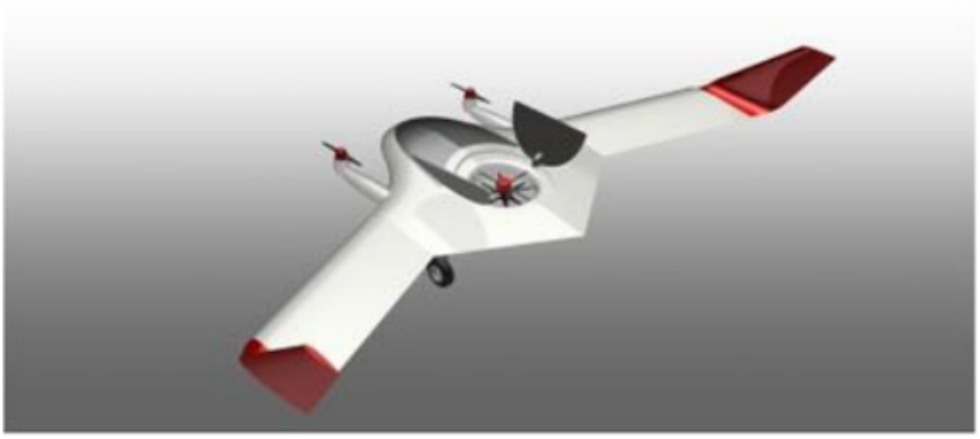


FIGURE 2.4: TURAC VTOL (Ozdemir et al., 2014)

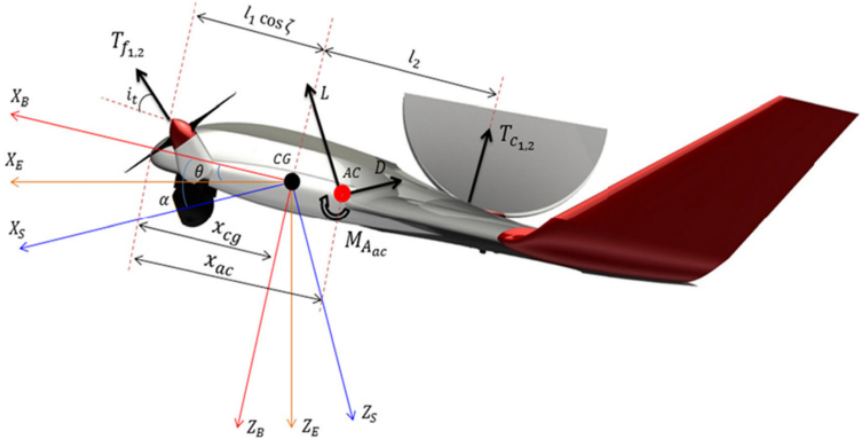


FIGURE 2.5: Forces and Moments on TURAC VTOL (Yuksek et al., 2016)

$$C_x^{fs} = -C_D^{fs}\cos\alpha + C_L^{fs}\sin\alpha$$

$$C_L^{fs} = [C_{L_o} + C_{L\alpha}\alpha + C_{Lq}c * q/(2 * V_a) + C_{L\delta_e}\delta_e]$$

$$C_D^{fs} = [C_{D_o} + C_{D\alpha}\alpha + C_{Dq}c * q/(2 * V_a) + C_{D\delta_e}\delta_e]$$

$$F_{A_x}^{wake} = 1/2 * \rho * V_T^2 * A_s * C_x^{wake}$$

$$C_x^{wake} = -C_D^{wake}\cos\alpha_{eff} + C_L^{wake}\sin\alpha_{eff}$$

α_{eff} is the effective angle of attack

$$C_L^{wake} = [C_{L_o} + C_{L\alpha_{eff}}\alpha_{eff} + C_{Lq}c * q/(2 * V_a) + C_{L\delta_e}\delta_e]$$

$$C_D^{wake} = [C_{D_o} + C_{D\alpha_{eff}}\alpha_{eff} + C_{Dq}c * q/(2 * V_a) + C_{D\delta_e}\delta_e]$$

Forces along Y- axis

$$F_y = F_T^y + F_{Ay}$$

$F_T^y = 0$

$F_y = F_{Ay}^{fs} + F_{Ay}^{wake}$

$F_{Ay}^{fs} = C_y^{fs} \tilde{q}^{fs} (A - 2 * A_s)$

$F_{A_Y}^{wake} = 1/2 * \rho * V_T^2 * A_s * C_Y^{wake}$

$where C_Y^* = C_{Y_\beta}^* \beta + C_{Y_{\delta_a}}^* \delta_a + C_{Y_{\delta_r}}^* \delta_r + b^*/(2 * V_\infty) * (C_{Y_p}^* p + C_{Y_r}^* r) + C_{Yo}$

Forces along z-axis

$F_z = F_T^z + F_{Az} + (F_{Gz}))$

$F_T^z = -(T_{f1} + T_{f1}) sin(i_t) - (T_{c1} + T_{c2})$

$F_{Az} = F_{Az}^{fs} + F_{Az}^{wake}$

$F_{Az}^{fs} = C_z^{fs} \tilde{q}^{fs} (A - 2 * A_s)$

$F_{A_z}^{wake} = 1/2 * \rho * V_T^2 * A_s * C_z^{wake}$

$C_x^{fs} = -C_D^{fs} \cos\alpha + C_L^{fs} \sin\alpha$

$C_x^{wake} = -C_D^{wake} \cos\alpha_{eff} + C_L^{wake} \sin\alpha_{eff}$

$\tilde{q}^{fs} = 1/2 * \rho * V_\infty^2$

where F_* denote forces . The subscript A and T stand for aerodynamic and thrust components, respectively. Aerodynamic forces are composed of free-stream forces/-moments and ones generated on the propeller wake regions. C_x^* is defined as aerodynamic force coefficients . T_{f_*} denotes the thrust force by Two front rotors. T_{c_*} denotes thrust force by coaxial rotors . C_D^*, C_L^* denotes the lift coefficient and drag coefficient respectively

Moments The Figure 2.6 shows the Total Moments and geometric dimension on TURAC VTOL

roll moment $L = L_T + L_A$

$L_A = L_B^{fs} + L_B^{wake}$

L_T denotes moment due to thrust forces: $= (T_{f1} - T_{f2}) \cos it \sin\tau$

$L_b^{fs} = C_l^{fs} \tilde{q}^{fs} (A - 2 * A_s) * b$

$L_B^{wake} = 1/2 * \rho * V_T^2 * A_s * C_l^{wake}$

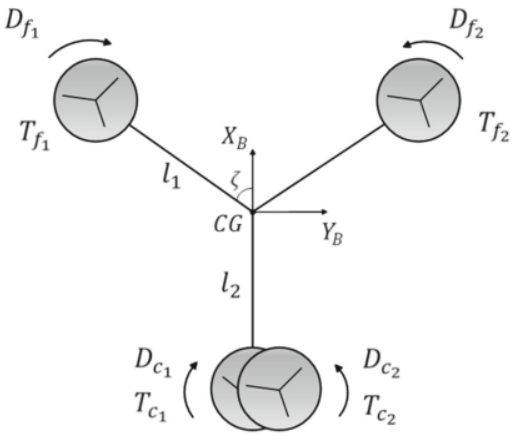


FIGURE 2.6: Moment and geometric dimension (Yuksek et al., 2016)

$where C_l^{fs} = C_{l_\beta}^{fs}\beta + C_{l_{\delta_a}}^{fs}\delta_a + C_{l_{\delta_r}}^{fs}\delta_r + b^{fs}/(2 * V_\infty) * (C_{l_p}^{fs}p + C_{l_r}^{fs}r) + C_{lo}$

yaw moment $L = N_T + N_A$

$N_A = N_B^{fs} + N_B^{wake}$

N_T denotes moment due to thrust forces: $= (-D_{f1} + D_{f2} - D_{c1} + D_{c2})$

D_* Denotes the Drag moment

$N_b^{fs} = C_n^{fs}\tilde{q}^{fs}(A - 2 * A_s) * b$

$N_B^{wake} = 1/2 * \rho * V_T^2 * A_s * C_n^{wake}$

$where C_n^{fs} = C_{n_\beta}^{fs}\beta + C_{n_{\delta_a}}^{fs}\delta_a + C_{n_{\delta_r}}^{fs}\delta_r + b^{fs}/(2 * V_\infty) * (C_{n_p}^{fs}p + C_{n_r}^{fs}r) + C_{no}$

pitch moment: $M = M_T + M_A$

$M_A = M_B^{fs} + M_B^{wake}$

M_T denotes moment due to thrust forces: $= (T_{f1} + T_{f2}) \sin(it) l_1 \cos\tau - (T_{c1} + T_{c2}) l_2$

$M_b^{fs} = C_m^{fs}\tilde{q}^{fs}(A - 2 * A_s) * \tilde{c}$

$M_B^{wake} = 1/2 * \rho * V_T^2 * A_s * C_m^{wake}$

$where C_m^{fs} = C_{m_\alpha}^{fs}\alpha + C_{m_{\delta_a}}^{fs}\delta_a + C_{m_{\delta_r}}^{fs}\delta_r + c^{fs}/(2 * V_\infty) * (C_{m_q}^{fs}q) + C_{mo}$

Aerodynamic forces and moments exerted on the UAV are function of total airflow vector, V_T .V_T is composed of V_∞ which is free Airstream velocity generated by translation motion . Second component is V_{out} is the propeller induced airflow. These are related as :

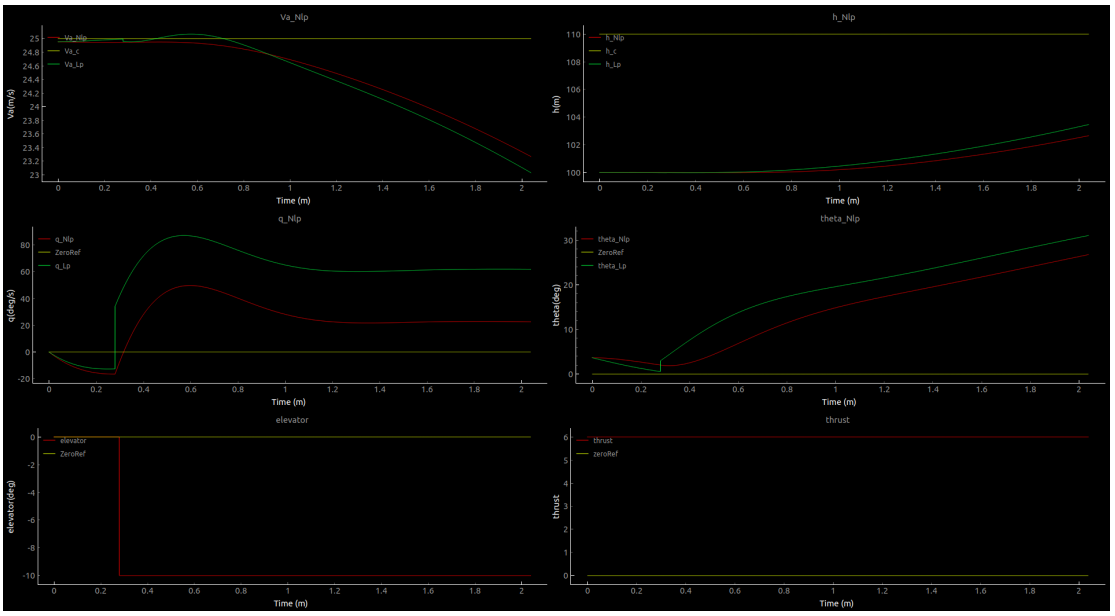


FIGURE 2.7: Open loop Simulation

$V_T = \sqrt{(V_{out}\xi \sin(\alpha + i_t))^2 + (V_{out}\xi \sin(\alpha + i_t) + V_\infty)^2}$

$V_{in} = (V_\infty * \cos\alpha \cos(it) + V_{out})/2$

where V_{in} is the intake airflow velocity

$V_{out} = \sqrt{(V_\infty \cos(\alpha)\cos(i_t))^2 + ((T_{f1} + T_{f2})/(\rho * \pi * R_p^2))}$

where R_p is the radius of the propeller.

Once we derived the dynamics and Equation of Motions we will assume this as **True non linear Model** of the VTOL. To check the correctness of the derived equation , few Open loop simulations of the VTOL has been run in the python. The Figure 2.7 shows one of the such response of the aircraft when elevator angle is changed from 0^{deg} to -10^{deg}. In graph we can observe that aircraft response represented by green color is making an ascent when elevator is changing from 0 to -10 . Thus we can say Aircraft is behaving correctly for change in elevator inputs.

2.4 Trim Condition and Linearized Dynamics

Trim point is simply a operating point (system states, control) at which a system is operating,For example A straight level flight. A Equilibrium point is sometime confused with trim point.A equilibrium point is the system operating point at which the net force on the system is zero.
Equilibrium point $\subset$ Trim point
A straight level flight which means a Aircaft flying at a Constant height with a constant horizontal speed is also a Equilibrium point.
In this Thesis I have used MATLAB trim function to get the system states at trim condition .
To obtain the states and input at trim point , first a Simulink file has been designed as shown in Figure A.1 .This file shows the relationship between the change in inputs (elevator anglw,aileron ,rudder , thrust force) to the change in output (V_a,α,β). Force and Moments is a functional Block which represent the forces and moments on TURAC as discussed in section 2.3. After calculating the Force and Moments on VTOL the Vector [F,M] is passed to the dynamics block which represent VTOL kinematics Equation to get the next states.

This file is then sent to **trim** function of MATLAB to obtain the states and inputs at trim condition. A script has is designed to set the desired condition for Trim and then pass it to the Trim function as shown in Figure A.2

Trim condition for level flight (flying at a constant Altitude)

- V_a desired Airspeed =25m/s
- γ (gamma) desired flight path angle =0
- R (desired Radius) = ∞ for a level flight

```
>> x_trim

x_trim =

   -0.0000
    0.0000
 -100.0000
   24.9489
    0.0000
    1.5973
    0.0000
    0.0639
   -0.0000
   -0.0000
    0.0000
    0.0000

>> u_trim

u_trim =

     0
     0
     0
     6
     6

>> A

A =

   -0.0475    0.2383   -1.5973   -9.7900         0
   -0.5908   -2.6983   24.9489   -0.6268         0
   -0.0884   -0.5145   -0.8796         0         0
         0         0    1.0000         0         0
    0.0639   -0.9980         0   25.0000         0

>> B

B =

   -0.3062    0.0735    0.0735
    4.7830   -0.0087   -0.0087
  -49.8341   -0.0001   -0.0001
         0         0         0
         0         0         0

>> eig(A)

ans =

   0.0000 + 0.0000i
  -1.8080 + 3.4693i
  -1.8080 - 3.4693i
  -0.0047 + 0.1999i
  -0.0047 - 0.1999i
```

FIGURE 2.8: States and control Input at Trim point

- -$p_d = h_o =$ 100m desired Altitude

Figure 2.8 shows the obtained Trim states $(p_n, p_d, u, v, w, \psi, \theta, \phi, p, q, r)$ and input (aileron, elevator,rudder,front rotor-1 Thrust force,front rotor-2 Thrust force,coaxial rotor-1 Thrust force, coaxial rotor-2 Thrust force) A function has been designed in MATLAB to get the system **Continuous State Space model** at obtained Trim state . This state space model has been further divided to get Longitudinal and Transitional State space model. Further Eigen Values of system MATRIX has been obtained to verify the correctness of obtained state Space Model.

In thesis I will focus on designing controller to control the Longitudinal States therefore And lateral dynamics is being controlled by general PID controller.

Relation Between continuous Non linear model Vs State Space model

$\dot{x} = f_{true-model}(x, u)$

$(x^*, u^*) \longleftarrow$ trim states and inputs

Linearizing Around Trim point

$\dot{x} - \dot{x}^* = \frac{\mathrm{d}f(x^*, u^*)}{\mathrm{d}x}(x - x^*) + \frac{\mathrm{d}f(x^*, u^*)}{\mathrm{d}u}(u - u^*)$

x^* =0 ie We are not changing Trim conditions therefore Constant Trim State

then we define ,

$\frac{\mathrm{d}f(x^*, u^*)}{\mathrm{d}x} = A$ and

$\frac{\mathrm{d}f(x^*, u^*)}{\mathrm{d}u} = B$ Therefore We get,

$\dot{x}$ = A(x - x^*) + B(u - u^*) as a Linearized system in a state space format. The obtained State Space Model will act like a predictive Model in Our MPC controller designed for Longitudinal dynamics controller [Chapter-3].

Chapter 3

Model Predictive Control

3.1 Introduction

Model Predictive Control which is also known by Receding Horizon Control (RHC), Generalized Predictive Control (GPC)..

(ref:wikipedia-MPC) MPC is based on iterative, finite-horizon optimization of a plant model. At time t the current plant state is sampled and a cost minimizing control strategy is computed (via a numerical minimization algorithm) for a relatively short time horizon in the future: $[t, t+T]$. Specifically, an online or on-the-fly calculation is used to explore state trajectories that emanate from the current state and find a cost-minimizing control strategy until time $[t+T]$. Only the first step of

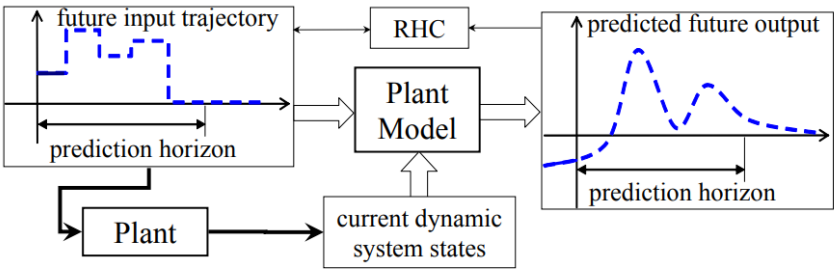


FIGURE 3.1: Receding Horizon Control Concept (ref:EE392m - Spring 2005 mpc lecture)

the control strategy is implemented, then the plant state is sampled again and the calculations are repeated starting from the new current state, yielding a new control and new predicted state path.
The prediction horizon keeps being shifted forward and for this reason MPC is also called receding horizon control Figure 3.1.

- At each time step, compute control by solving an openloop optimization problem for the prediction horizon
- Apply the first value of the computed control sequence
- At the next time step, get the system state and re-compute

Simple Algorithm can be written as :

$$
\begin{aligned}
J^*(x(t)) = & \underset{\{u_k\}_{k=0}^{N-1}}{\text{minimize}} \sum_{k=0}^{N-1} (l(x_k, u_k)) \\
& \text{subject to } x_0 = x(t) \\
& \text{for k=0,1.....,N-1:} \\
& x_{k+1} = f(x_k, u_k) \\
& x_k \in \mathcal{X} \\
& u_k \in \mathcal{U}
\end{aligned}
$$

In above formulation state X is 5 dimensional longitudinal state space where
x^1 = u (x-axis Air speed)
x^2 = w(z-axis Air speed)
x^3 = q (rate of change of pitch)
x^4 = θ (pitch Angle in radian)
x^5 = h (Height of the Aircraft)

And U is 3 dimensional control space where

u^1 = elevator Angle in radian

$u^2 = T_{f1}$

$u^3 = T_{f2}$

where T_{f*} represent the Thrust force generated by front rotors.

Cost function $l(x_k, u_k))$ is known as a cost function ,In this thesis I chose quadratic cost in the form of

$W_x(X_{desired} - X)^2 + W_u * (U)$

The first term $W_x(X_{desired} - X)^2$ forces the system to reach to the desired States , and second term $W_u * (U)$ make sure that the control effort should be small .

W_x and W_u are the Weights and typically a design Parameter and indicates how much importance we give to each of the chosen performance matrix .

These weights are found by various methods , In this thesis initially they are found by hit and trial method and later in chapter-4 I have explored the automated methods to find these weights.

Predictive Model of the System $x_{k+1} = f(x_k, u_k)$

$f(x_k, u_k)$ is the Approximate Model of the System that we get through Analysis or through Any other Measurement technique. In this thesis I have used **Linear** longitudinal State Space Model (discussed in Previous chapter) around the **Trim States** as a Approximate longitudinal dynamics of VTOL aircraft.

We know from chapter-2 that ,linear continuous State space around the Trim condition x^* and u^* has the following form

$\dot{x}$ = A(x - x^*) + B(u - u^*)

and $(y - y^*)$ = C(x - x^*)

C= I^5x (I is identity Matrix of Oder 5)

Above two equation is in Continuous time , to discretize it and to get the equation in the form of

$x_{k+1} = \tilde{A}(x_k - x^*) + \tilde{B}(u_k - u^*)$, where x_{k+1} is the the next state around the trim (x^*) state, I will use **Runge Kutta** 4^{th} order discretization method .

3.2 Runge-kutta Integration Method :

Overview Following figure shows the discretizing the continous time model through RK4 method:

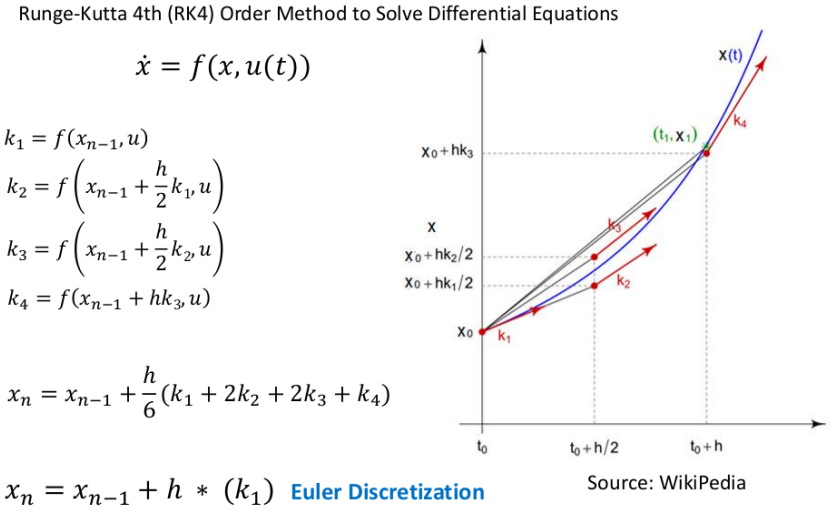


FIGURE 3.2: RK4 discretization (Worthmann et al., 2016)

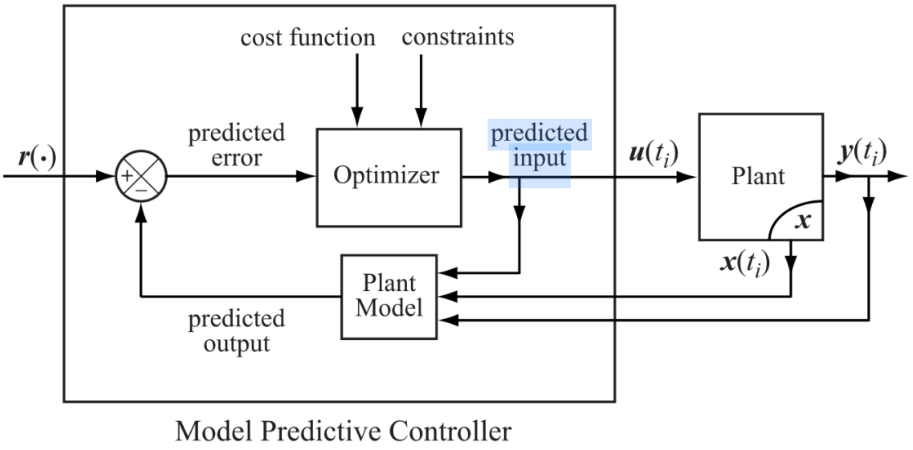


FIGURE 3.3: Linear MPC control (ref:Stanislaw H. Żak -PID lecture)

Discretizing State Space in Python Code Following code shown in Figure A.3 has been developed in Python to discretize the Continuous State Space Model through RK4 :

3.3 Linear MPC control

Introduction Linear Model Predictive Control is a variant of model predictive control (MPC) that is characterized by the use of linear system models in the prediction.

In last section I mentioned that I have used the longitudinal linear-state space system as a predictive model of the VTOL . This linearized model has been found out around the trim condition which in this case is the condition of aircraft in fixed wing region and flying at a constant Height. I have developed the Linear MPC controller to track the desired Height.

CASADI CASADi is an open-source tool for nonlinear optimization and algorithmic differentiation. It facilitates rapid and efficient — implementation of different

methods for numerical optimal control. I have used CASADI as an optimizer to solve the minimization problem shown in section 3.1.

The Following Figure A.4 shows the code snippet developed in python to solve the minimization problem shown in section 3.1 as a non linear Optimization problem. Here f_{xdot} is a function representing $\dot{x}$ = A(x) + B(u)

Result Following Figure 3.4 is a graph from a simulation showing the Aircraft (red color) is able to track desired height. In this case suitable weight has been found by hit and trial method.

weights = ([1.0, 0.01, 0.5, 2.001, 5.0, 4.00, 0.001, 0.001])

First five weights elements are for state error minimization and next third weights elements are of small control effort.

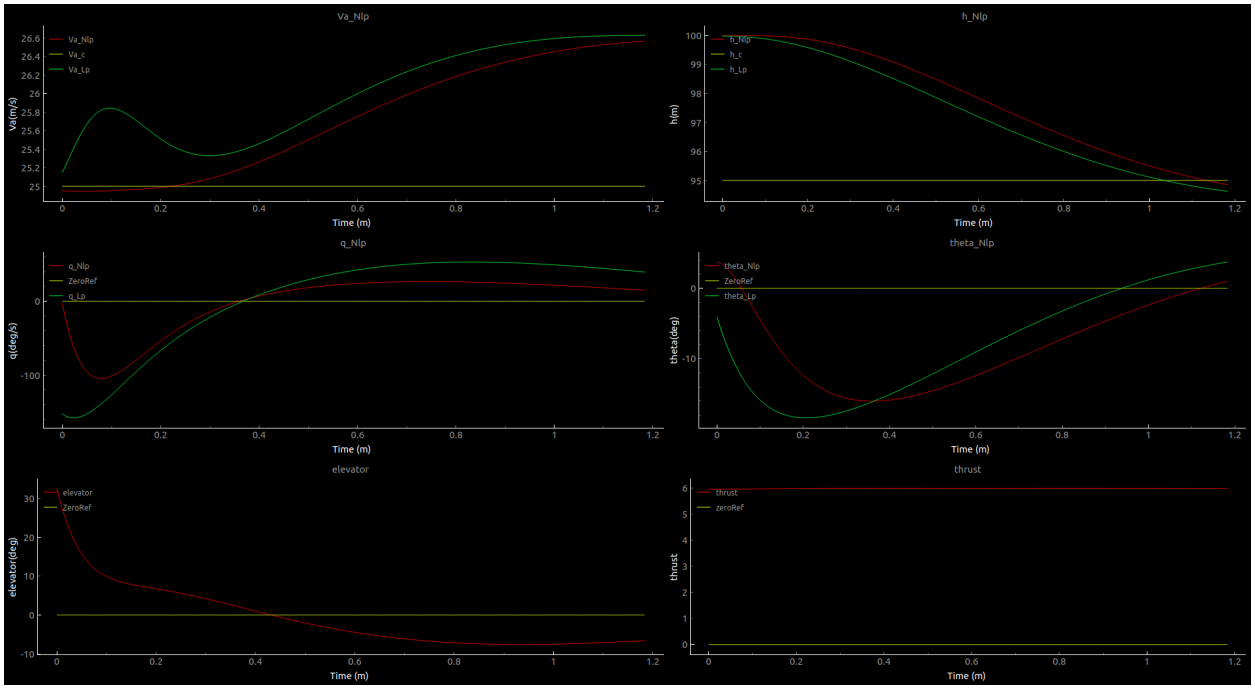


FIGURE 3.4: Linear MPC control showing that air Craft is tracking the desired Height

Chapter 4

Learning Based MPC

4.1 Introduction :

Learning-based MPC addresses the automated and data-driven generation or adaptation of elements of the MPC formulation such that the control performance with respect to the desired closed-loop system behavior is improved.(Hewing et al., 2020) The setup in which this learning takes place can be diverse. For instance, offline learning considers the adaptation of the controller between different trials or episodes of a control task, during which data are collected. In methods that learn online, on the other hand, the controller is adjusted during closed-loop operation (while performing repetitive tasks) or using the data collected during one task execution. While much of the research in learning-based MPC is focusing on automatically improving the model quality, which is the most obvious component affecting MPC performance, several research efforts are addressing the formulation of the MPC problem directly or utilizing the MPC concept to satisfy constraints during learning-based control. In this Thesis I have focused on second Category for learning and improving the Performance:

- Learning the system dynamics that we can use as a better predictive model

- Learning the controller design,learning better Weights,constraints ,cost function

4.2 Learning MPC weights

Introduction To learn the weights I have designed the algorithm by modifying the Deep deterministic policy Gradient method (Lillicrap et al., 2019) which is a Reinforcement learning Method to learn the policy. Here policy simply refers to the what action agent should take at a particular state.

Algorithm Algorithm 1 shown below has been developed for learning Actor function μ. After the learning process the optimal weights can be obtained simply by $\arg\max \mu()$.

The developed algorithm consider MPC weight as an continuous-action a_w which is then explored by the DDPG algorithm. Actor function $\mu()$ defines what action should take at a particular state. Critic function $Q()$ defines how good is it to take the action a_w in a state.

Actor first issues the action a_w given initial state s_1 and current parameters θ, In this thesis I have fixed the initial state for all the episode to the trim state obtained in chapter-2. Obtained a_w used as an Weight for MPC to obtain set of optimal actions which then applied to the system to get the next-state reward and if terminal state is reached. This process is repeated till terminal state has been reached and finally a tuple of (initial state, Weights,total Reward ,Terminal State) is stored in a buffer for training of Critic newt work.

Next two Steps are Similar to the Original DDPG algorithm where Critic and Actor network is updated.

Algorithm 1 Weight Search through Deep Deterministic Policy Gradient Algorithm

Randomly initialize critic network $Q(s, a_w|\theta^Q)$ and actor $\mu(s|\theta^\mu)$ with weights θ^Q and θ^μ.
Initialize target network Q' and μ' with weights $\theta^{Q'} \leftarrow \theta^Q$,$\theta^{\mu'} \leftarrow \theta^\mu$
Initialize replay buffer $\mathcal{B}$
for $episode = 1, M$ **do**
  Initialize a random process $\mathcal{N}$ for action exploration
  Receive initial observation state s_1
  Select $a_w = \mu(s_1|\theta^\mu) + \mathcal{N}$ according to the current policy and exploration noise
  $\mathcal{W} \leftarrow a_w$
  do
    Use Weight Matrix $\mathcal{W}$ to estimate optimal action sequence $U_t^{(H)}$
    execute first action u_t from selected sequence $U_t^{(H)}$
      $(s_{t+1}, r_{t+1}, done) \leftarrow \mathcal{F}_{true_model}(s_t, u_t)$
  $\mathcal{R} \leftarrow r_t + \gamma r_{t+1}$
  While($\sim done$)
  $\mathcal{R} \leftarrow \mathcal{R}/T$(normalizing the total Reward by the total no of time steps in a episode)
  $Store(s_1^i, a_w^i, \mathcal{R}^i, s_T^i)$ in $\mathcal{B}$
  Sample a random mini-batch of N transitions $(s_1^i, a_w^i, \mathcal{R}, s_T^i)$ from $\mathcal{B}$
  Set $y^i = \mathcal{R}^i$
  Update the critic by minimizing the Loss :

$$L = \frac{1}{N}\sum_{i=1}^{N}(y^i - Q(s_1^i, a_w^i|\theta^Q))^2$$

  Update the actor policy using the sampled policy gradient:

$$\nabla_{\theta^\mu} J \approx \frac{1}{N}\sum_{i=1}^{N} \nabla_{a_w} Q(s, a_w|\theta^Q)|_{s=s_1^i, a_w=\mu(s_1^i)} \nabla_{\theta^\mu}\mu(s|\theta^\mu)|_{s_1^i}$$

  Update the target networks: $\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'}; \theta^{\mu'} \leftarrow \tau\theta^u + (1-\tau)\theta^{\mu'}$
end for $= 0$

Terminal States Two conditions are assessed to determine if a state is terminal:

- If time steps remaining $t_R = 0$, the state is terminal
- if the magnitude of the altitude error exceeds a critical value, $|e_h|$ ¿ $e_{h,max}$, the state is terminal

The maximum permitted altitude deviation was set to $e_{h,max}$ = 20-30 meter.

Reward function A reward function will be constructed that encodes the following behaviour:

- Track the aircraft altitude from current height h to the target altitude h_T , such that $e_h = h\text{-}h_T = 0$

A simple reward function the encodes the desired mentioned behaviour can designed as $r_t = (1 - |e_h|)$ such that when $|e_h|$ is zero reward obtained become maximum .
However it has been found that it is advantageous to provide agents rewards normalised in e.g. in range [0,1] to improve the stability of neural network convergence (**?**).
Therefore Normalised error terms has been used, The normalised error $\bar{e}$ is calculated by:

$$\bar{e} = \frac{\frac{|\bar{e}|}{k}}{1 + \frac{|\bar{e}|}{k}}$$

Scaling factor k can be considered as at what absolute error value do I consider agent's work to be half complete.
Reward at time t is therefore

$$r_t = 1 - \bar{e_t}$$

which is always bounded in (0,1).

Simulation Result The following graphs is obtained for the following parameters

- M =1000 episodes
- learning rates = 0.0005 for both actor and critic
- S_1 = (h=100meter,Va = 25 m/s)

- $h_{desired}$ = 105 meter

The weights and total Reward obtained corresponding to the graph Figure 4.1 is $[2.55460801, 2.49933617, 2.52168748, 2.69877644, 2.59287848, 2.58079338, 2.50337014, 2.54272376]$" reward =0.49121906413479727

The weights and total Reward obtained corresponding to the graph Figure 4.2 is

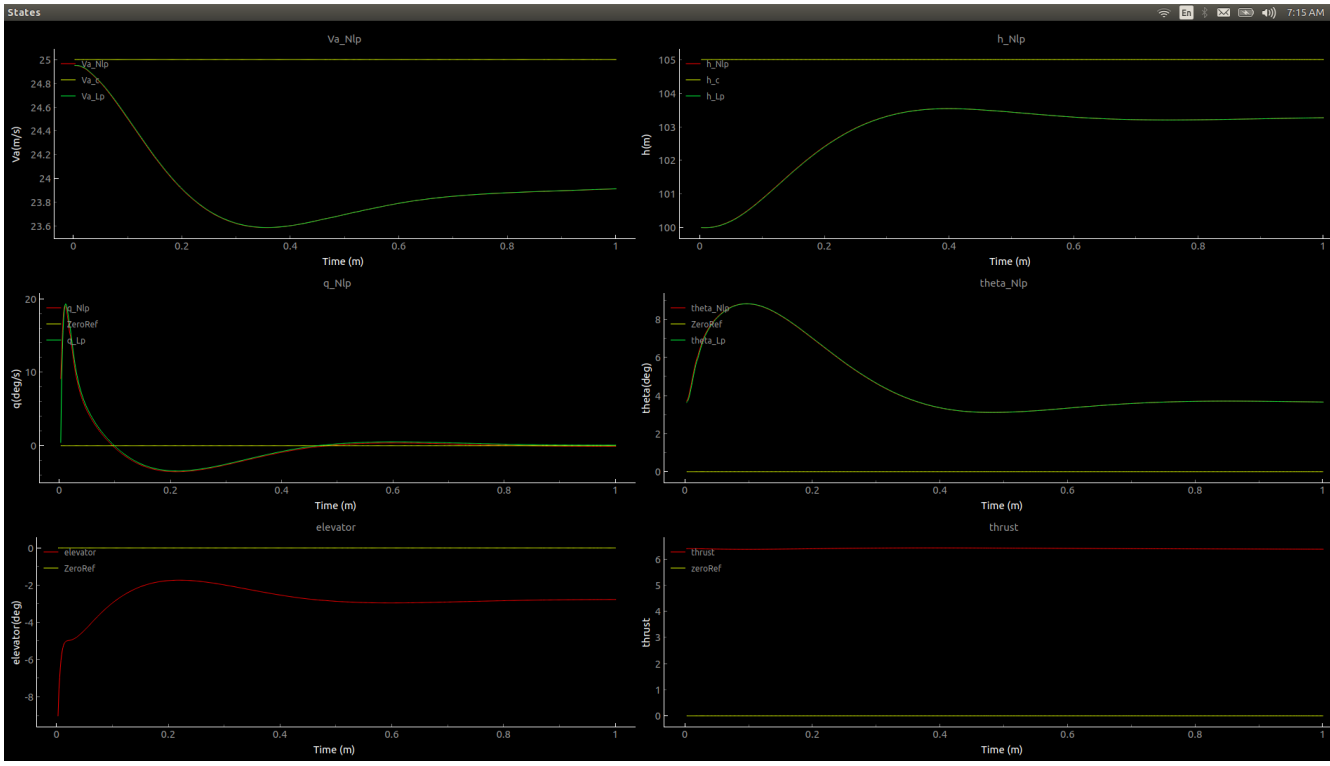


FIGURE 4.1: Simulation Result Corresponding to the Weights obtained on 1st episode run

$[0.30611667, 5.02722512, 4.83456607, 4.81023379,$
$4.95270451, -0.11439833, 5.15594659, 4.95798248]$ reward = 0.617883149112560

On a closer look between 0.2 to 0.4 in first graph It is noticeable that the Aircraft is only able to reach little more than 103 meter. In second graph we observe that Aircraft is able to reach little more than 104 meter.

Average reward has also been plotted in Figure 4.3 which shows that average score/reward is increasing as agent traverses through more number of episodes , which clearly

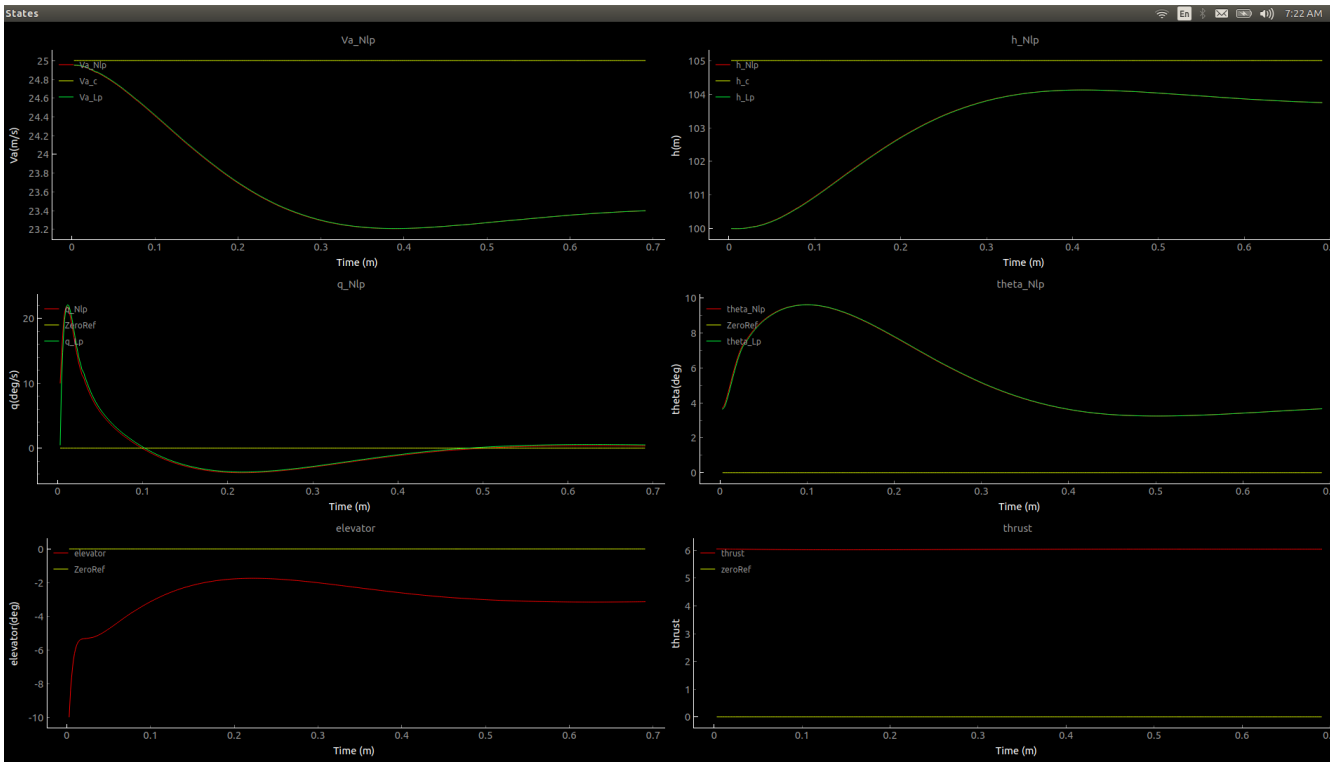


FIGURE 4.2: Simulation Result Corresponding to the best reward

shows that agent is learning and parameters of Actor network is shifting towards the region which guarantee better results and Reward.

Scope for further Improvement The learning can be further improved by taking into consideration the following points:

- Very Naive Reward function has been used . The simulation is behaving according to the defined Reward function which is to minimize the ($|h_{current} - h_{desired}|$) error.Further Reward term corresponding to fast tracking , minimum control effort , small steady state error can be added to improve the learning and hence to obtain the better Weights.

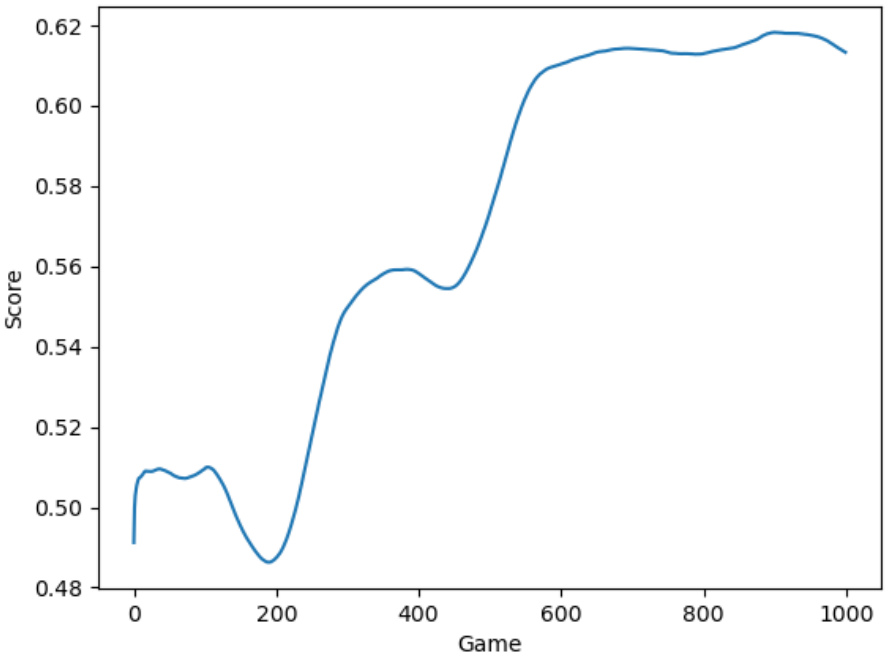


FIGURE 4.3: Average Reward for past 100 episodes

- In this Thesis I have used the Same value of hyper parameter like Number of episodes learning rate , γ and similar NN architecture as defined in Original DDPG paper. These hyper parameter can be better tuned to for our application.

Appendix A

Appendix A

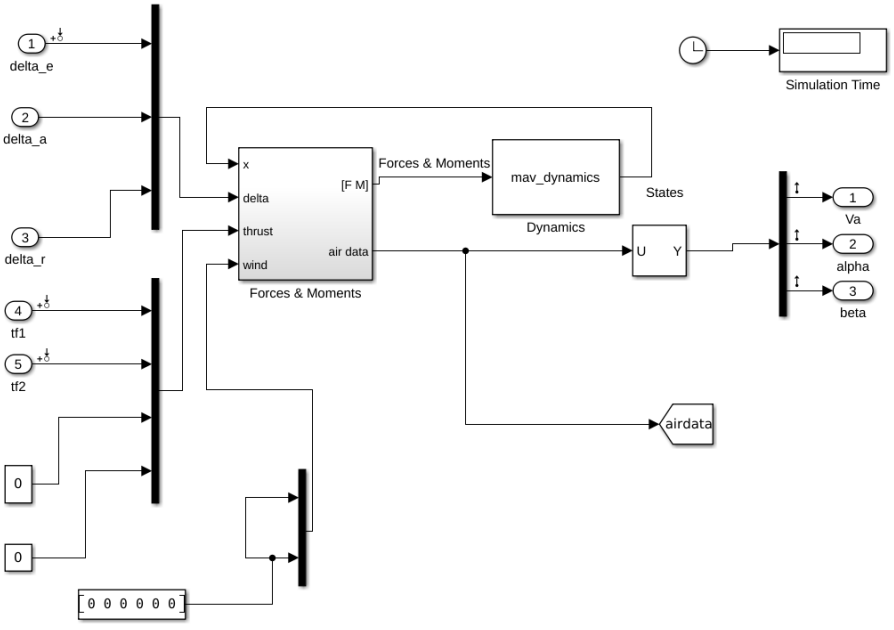


FIGURE A.1: Simulink file for Calculating Trim states

```
x0 = [0;...     %1 -  Pn
    0;...       %2 -  Pe
    -100;...    %3 -  pd
    Va;...      %4 -  u
    0;...       %5 -  v
    0;...       %6 -  w
    0;...       %7 -  phi
    gamma;...   %8 -  theta
    0;...       %9 -  psi
    0;...       %10-  p
    0;...       %11-  q
    0;...       %12-  r
    ];
% define which states to hold equal to the initial conditions
ix = [3,8];
% specify initial inputs
u0 = [0;...     %1-  e
    0;...       %2-  a
    0;...       %3-  r
    6;...       %4-  delta_t as tf1 =tf2
    6;...
    ];
% specify which inputs to hold constant
iu = [];% define constant outputs
y0 = [Va;...    %1-  Va
    0;...       %2-  alpha
    0;...       %3-  beta
    ];% specify which outputs to hold constant
iy = [1,2,3];% define constant deerivatives
dx0 = [...
    0;...               %1 - Pndot
    0;...               %2 - Pedot
    -Va*sin(gamma);...  %3 - hdot
    0;...               %4 - udot
    0;...               %5 - vdot
    0;...               %6 - wdot
    0;...               %7 - phidot
    0;...               %8 - thetadot
    0;...               %9 - psidot
    0;...               %10- pdot
    0;...               %11- qdot
    0;...               %12- rdot
    ];
if R~= Inf, dx0(9) = Va/R; end
%orignal if R~= 0, dx0(9) = Va/R; end% specify which derivatives to hold constant in trim algrithm
idx = [3;4;5;6;7;8;9;10;11;12];
%x0,u0,y0,ix,iu,iy,dx0,idx
% compute trim conditions
[x_trim,u_trim,y_trim,dx_trim] = trim('trim_vtol_horizontal_flightnon',x0,u0,y0,ix,iu,iy,dx0,idx);
%y_trim
% check to make sure that the linearization worked (should be small)
norm(dx_trim(3:end)-dx0(3:end))
end
```

FIGURE A.2: MATLAB function to Calculate Trim states

```python
import numpy as np
import pdb
from casadi import*

A_lon_cont = np.array([[-0.0475, 0.2383, -1.6000, -9.7900, 0],
                       [-0.6016, -2.6981, 24.9000, -0.6264, 0.],
                       [-0.0884, -0.5145, -0.8796, 0.0, 0.],
                       [0., 0.0, 1.0, 0.0, 0.],
                       [0.0639, -0.9980, 0.0, 25.000, 0.]])

B_lon_cont = np.array([[-0.3062, 0.0735, 0.0735],
                       [4.7643, -0.0087, -0.0087],
                       [-49.8341, -0.000, -0.000],
                       [0.0, 0.0, 0.0],
                       [0.0, 0.0, 0.0]])
dt = 0.1

fxdot = lambda x, u: np.matmul(A_lon_cont, x) + np.matmul(B_lon_cont, u)  # x_dot=AX+BU


def next_step( state_current,input_current):
    #pdb.set_trace()
    # Integrate state space using Runge-Kutta RK4 algorithm
    k1 = fxdot(state_current, input_current)
    k2 = fxdot(state_current + dt / 2 * k1, input_current)
    k3 = fxdot(state_current + dt / 2 * k2, input_current)
    k4 = fxdot(state_current + dt * k3, input_current)
    x_next = state_current + dt / 6 * (k1 + 2 * k2 + 2 * k3 + k4)

    return x_next
```

FIGURE A.3: State Space discretization in Python

```python
for k in range(1, self.N, 1):
    con = self.U[:, k]

    k1 = self.fxdot(self.X[:, k - 1], self.U[:, k])
    k2 = self.fxdot(self.X[:, k - 1] + self.dt / 2 * k1, self.U[:, k])
    k3 = self.fxdot(self.X[:, k - 1] + self.dt / 2 * k2, self.U[:, k])
    k4 = self.fxdot(self.X[:, k - 1] + self.dt * k3, self.U[:, k])
    x_next = self.X[:, k - 1] + self.dt / 6 * (k1 + 2 * k2 + 2 * k3 + k4)
    self.X[:, k] = x_next
    self.obj = self.obj + mtimes(mtimes((self.X[:, k - 1] - self.y_ref).T, self.Q),
                                 (self.X[:, k - 1] - self.y_ref)) + \
               mtimes(mtimes(con.T, self.R), con)

self.opti.subject_to(self.X[:, 0] == self.state_current[0:5])
self.opti.subject_to(self.U[:, 0] == self.state_current[5:])
```

FIGURE A.4: Formulating minimization Problem in CASADI

Bibliography

Govdeli, Y., Tran, A. T., and Kayacan, E. (2019). Multiple Modeling and Fuzzy Switching Control of Fixed-Wing VTOL Tilt-Rotor UAV. In Kearfott, R. B., Batyrshin, I., Reformat, M., Ceberio, M., and Kreinovich, V., editors, *Fuzzy Techniques: Theory and Applications*, volume 1000, pages 270–284. Springer International Publishing, Cham.

Hewing, L., Wabersich, K. P., Menner, M., and Zeilinger, M. N. (2020). Learning-Based Model Predictive Control: Toward Safe Learning in Control. *Annual Review of Control, Robotics, and Autonomous Systems*, 3(1):269–296.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. (2019). Continuous control with deep reinforcement learning. *arXiv:1509.02971 [cs, stat]*. arXiv: 1509.02971.

Ozdemir, U., Aktas, Y. O., Vuruskan, A., Dereli, Y., Tarhan, A. F., Demirbag, K., Erdem, A., Kalaycioglu, G. D., Ozkol, I., and Inalhan, G. (2014). Design of a Commercial Hybrid VTOL UAV System. *Journal of Intelligent & Robotic Systems*, 74(1-2):371–393.

Worthmann, K., Mehrez, M. W., Zanon, M., Mann, G. K. I., Gosine, R. G., and Diehl, M. (2016). Model Predictive Control of Nonholonomic Mobile Robots Without Stabilizing Constraints and Costs. *IEEE Transactions on Control Systems Technology*, 24(4):1394–1406.

Yuksek, B., Vuruskan, A., Ozdemir, U., Yukselen, M. A., and Inalhan, G. (2016). Transition Flight Modeling of a Fixed-Wing VTOL UAV. *Journal of Intelligent & Robotic Systems*, 84(1-4):83–105.