Projeto de Graduação

DEPARTAMENTO DE ENGENHARIA ELÉTRICA

05 de julho de 2017

MATHEMATICAL MODELING OF A TILT-ROTOR DRONE

Rodrigo Simões Pessoa

PUC RIO

www.ele.puc-rio.br

Mathematical Modeling of a Tilt-Rotor Drone

Aluno: Rodrigo Simões Pessoa

Orientador(es): Msc. William de Souza Barbosa

Phd. Mauro Speranza Neto

Course completion work determined as a partial requirement for obtaining a Bachelor's Degree in Control and Automation Engineering from the Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, Brasil.

DEDICATION

I dedicate this work to my late uncle Mario D. P. Simões; an exemplar scholar and student throughout all his life. His attention to detail and excellence in all manners of work, no matter how hard it seemed to be, serve both as inspiration and as a cherished memory. It is said that one lives on through their work and the memories left behind. Let this work be a tiny piece of good memory, one through which he can live onwards.

ACKNOWLEDGEMENTS

To my main supervisor Msc. William de Souza Barbosa: I will miss our interesting chats and long discussions when trying to figure out what happened in unclearly explained theses. I will always remember how simplicity and humility will never mask knowledge or intelligence, but rather add to them. Naturally, I owe him many thanks for his assistance throughout this project and for his knowledge in nonlinear control systems and dynamic systems modelling, as they proved to be invaluable.

To my secondary supervisor Phd. Mauro Speranza Neto: I will miss his confidence when stating, matter-of-factly, that I would be able to solve any problem I came across, even when I did not know entirely how to go about it. His extensive experience with dynamic systems modelling was, without a doubt, a crucial help throughout this project.

I would also like to thank the Pontifical Catholic University of Rio de Janeiro for providing the knowledge, skillset, tools and workspace for the creation of this project, which would be bound to fail otherwise.

I would like to extend my thanks to Phd. Ana Maria Beltran Pavani for help and dedication as coordinator of the Control and Automation Engineering course and Phd. Professor Thiago Roux de Oliveira for his insight and guidance of nonlinear control techniques.

A very special gratitude to Bárbara Gomes Dantas, who provided me with unwavering emotional support, for always encouraging and being supportive. It was she who made me strive for excellence, especially through the struggles, always certain that I could triumph over any obstacle.

I am grateful to my mother, Lilia Domingues de Paula Simões, and my father, Marcio Nunes Pessoa, who have provided me through moral, emotional and economical support in my life. I am also grateful to my other family members and friends who have supported me along the way.

ABSTRACT

The objective of this course completion work was to develop a realistic and sophisticated mathematical model of a tilt rotor drone in modular form. The mathematical formulation used, as well as its implications, simplifications and assumptions were exposed in a detailed manner, to allow for future research. A discrete, code line simulator using MATLAB script and a continuous simulator using MATLAB’s Simulink tool were created. The PID control was used as a classical control technique in an attempt to stabilize and control the system. The LQR control was used as a modern control technique and discussions were made about the use of a feedback control loop stabilization applied to this multiple inputs multiple outputs (MIMO) nonlinear non affine system. The aerodynamic forces and drag were discussed and a mathematical implementation of such forces was suggested as a possible addition to future models.

Keywords: tilt rotor drone, mathematical modeling, MIMO, non-affine, nonlinear systems

DEPARTAMENTO DE ENGENHARIA ELÉTRICA

Modelagem Matemática de um Drone com Rotores Inclináveis

RESUMO

O objetivo deste trabalho de conclusão de curso foi desenvolver um modelo matemático realista e sofisticado de um drone com rotor inclináveis, em forma modular. A formulação matemática utilizada, bem como suas implicações, simplificações e premissas forma expostas de forma detalhada para permitir o seu uso em pesquisas futuras. Um simulador de linha de código, discreto, usando o script do MATLAB e um simulador contínuo utilizando a ferramenta Simulink do MATLAB foram desenvolvidos. O controle PID foi usado como técnica de controle na tentativa de estabilizar e controlar o sistema. O controlador LQR foi usado como técnica de controle moderno e foram feitas discussões sobre o uso da estabilização por realimentação de estados aplicado a este sistema de múltiplas entradas e múltiplas saídas (MIMO), não-linear e não-afim. As forças aerodinâmicas e o arrasto foram discutidos e a implementação matemática de tais forças foi sugerida como uma possível adição aos futuros modelos.

Palavras-chaves: drone com rotores inclináveis, modelagem matemática, MIMO, não-afim, sistemas não-lineares

FIGURE LIST

Figure 1 (Wing-arms and motors counter rotations) .... page 03

Figure 2 (Alteration of flaps to back propeller) .... page 03

Figure 3 (Coordinate system fixed to the drone’s body $O_F x_F y_F z_F$) .... page 04

Figure 4 (Figure depicting the tilting of a propeller) .... page 06

Figure 5 (Figure depicting the linearization of $f(t)$ with a frequency of $3.33\ Hz$) .... page 12

Figure 6 (Figure depicting the linearization of $f(t)$ with a frequency of $16.67\ Hz$)........ page 12

Figure 7 (Figure depicting the linearization of $f(t)$ with a frequency of $33.33\ Hz$) .... page 13

Figure 8 (Figure depicting the linearization of $f(t)$ when the frequency approaches infinity) ...... page 13

Figure 9 (PID controller diagram) .... page 17

Figure 10 (State-space feedback controller diagram) .... page 18

ABREVIATIONS AND ACRONYMS LIST

VTOL	– Vertical Takeoff and Landing
BC	– Before Christ
CAC	- California Aero Components
UAV	- Unmanned Aerial Vehicles
MIMO	– Multiple Inputs, Multiple Outputs
SISO	– Single Input, Single Output
ZOH	– Zero Order Hold
PID	– Proportional Integral Derivative controller
LQR	– Linear Quadratic Regulator
EAS	– Equivalent Airspeed
MAC	– Mean Aerodynamic Chord
AR	– Aspect Ration
ISA	– International Standard Atmosphere

SYMBOL LIST

$O_F x_F y_F z_F$ – Non-inertial coordinate system fixed to the drone's body and free to rotate with it.
$O_I x_I y_I z_I$ – Inertial coordinate system fixed to a given point in space.
ϕ – Angular position in relation to the X-axis (it is also a state).
θ – Angular position in relation to the Y-axis (it is also a state).
Ψ – Angular position in relation to the Z-axis (it is also a state).
$R(\phi, \theta, \psi)$ – Three dimensional rotation matrix.
$R_x(\phi)$ – Rotational matrix along the X-axis. It is called the roll matrix.
$R_y(\theta)$ – Rotational matrix along the Y-axis. It is called the pitch matrix.
$R_z(\psi)$ – Rotational matrix along the Z-axis. It is called the yaw matrix.
c_α – Abbreviation for the cosine function applied to the variable $(\cos(a))$.
s_α – Abbreviation for the sine function applied to the variable $(\sin(a))$.
$\vec{X}$ – State vector X.
$\vec{\mathrm{P}}$ – Position vector P.
$\vec{V}$ – Velocity vector V.
$\vec{\Xi}$ – Attitude vector Ξ.
$\vec{\Omega}$ – Angular velocity vector Ω.
x – Position along the X-axis (it is also a state).
y – Position along the Y-axis (it is also a state).
z – Position along the Z-axis (it is also a state).
v_x – Component of the velocity along the X-axis (it is also a state).
v_y – Component of the velocity along the Y-axis (it is also a state).
v_z – Component of the velocity along the Z-axis (it is also a state).
ω_ϕ – Component of the angular velocity along the X-axis (it is also a state).
ω_θ – Component of the angular velocity along the Y-axis (it is also a state).
ω_ψ – Component of the angular velocity along the Z-axis (it is also a state).
$\vec{F}$ – Net force vector.
m – Mass.
$\dot{\vec{V}}$ – Acceleration vector.
$\vec{M}$ – Net moment vector.
J – Inertia matrix (generally called inertia tensor).
$\dot{\vec{\Omega}}$ – Angular acceleration vector.
$\vec{F}_{prop}$ – Net propeller force vector.
$\vec{F}_{drag}$ – Net aerodynamic force vector (often called drag).
$\vec{F}_g$ – Gravity force vector.
$\vec{M}_{prop}$ – Net propeller moment vector.
$\vec{M}_{drag}$ – Net aerodynamic moment vector.
$\vec{M}_{gyro}$ – Net gyroscopic moment vector.
$F_{propeller}$ – Propeller force (absolute value for it is not a vector).
$k_{propeller}$ – Propeller angular velocity to lift (force) constant.
$\vec{\omega}_{propeller}$ – Propeller angular velocity vector.
$\vec{F}_{lp}^{\ T}$ – Transposed left propeller force vector.
$\vec{F}_{rp}^{\ T}$ – Transposed right propeller force vector.
δ_{lp} – Left propeller tilt angle.
δ_{rp} – Right propeller tilt angle.
k_{lp} – Left propeller angular velocity to lift (force) constant.
k_{rp} – Right propeller angular velocity to lift (force) constant.
$\vec{\omega}_{lp}$ – Left propeller angular velocity vector.
$\vec{\omega}_{rp}$ – Right propeller angular velocity vector.
$sign(\omega)$ – Sign function applied to variable ω.
$sgn(\omega)$ – Alternative symbol for sign function applied to variable ω.
$\vec{F}_{bp}^{\ T}$ – Transposed back propeller force vector.
k_{bp} – Back propeller angular velocity to lift (force) constant.
ω_{bp} – Back propeller angular velocity vector.
g – Absolute force of gravity (approximately $9.81\ {}^{m}/_{s}$).
$\left(\vec{F}_g^{\ T}\right)_F$ – Transposed force of gravity vector written in the fixed (non-interial) coordinate system.

$\left(\vec{F_g}^T\right)_I$ – Transposed force of gravity vector written in the inertial coordinate system.
$\vec{\tau}$ – Torque or moment of force.
$\vec{r}$ – Position vector from a given point to the point where a given force acts upon (different from position vector $\vec{\mathrm{P}}$).
$\vec{r}_{cm\to rp}$ – Position vector from the center of mass to the right propeller.
$\vec{r}_{cm\to lp}$ – Position vector from the center of mass to the left propeller.
$\vec{r}_{cm\to bp}$ – Position vector from the center of mass to the back propeller.
$x_{cm\to lp}$ – X-axis component of vector $\vec{r}_{cm\to lp}$.
$y_{cm\to lp}$ – Y-axis component of vector $\vec{r}_{cm\to lp}$.
$z_{cm\to lp}$ – Z-axis component of vector $\vec{r}_{cm\to lp}$.
$x_{cm\to rp}$ – X-axis component of vector $\vec{r}_{cm\to rp}$.
$y_{cm\to rp}$ – Y-axis component of vector $\vec{r}_{cm\to rp}$.
$z_{cm\to rp}$ – Z-axis component of vector $\vec{r}_{cm\to rp}$.
$x_{cm\to bp}$ – X-axis component of vector $\vec{r}_{cm\to bp}$.
$y_{cm\to bp}$ – Y-axis component of vector $\vec{r}_{cm\to bp}$.
$z_{cm\to bp}$ – Z-axis component of vector $\vec{r}_{cm\to bp}$.
$\dot{\vec{\delta}}$ – Angular velocity vector of propeller's tilt.
$|\vec{F}|$ – Norm of the net force vector $\vec{F}$.
$|\vec{\omega}|$ – Norm of the angular velocity vector $\vec{\omega}$
J_{xx} – Component of the inertia matrix (inertia tensor) J.
J_{xy}– Component of the inertia matrix (inertia tensor) J.
J_{xz}– Component of the inertia matrix (inertia tensor) J.
J_{yx} – Component of the inertia matrix (inertia tensor) J.
J_{yy}– Component of the inertia matrix (inertia tensor) J.
J_{yz}– Component of the inertia matrix (inertia tensor) J.
J_{zx} – Component of the inertia matrix (inertia tensor) J.
J_{zy}– Component of the inertia matrix (inertia tensor) J.
J_{zz}– Component of the inertia matrix (inertia tensor) J.
$\dot{\vec{X}}$ – First derivative with respect to time of the state vector $\vec{X}$.
$A(t)$ – Time variant matrix.
$B(t)$ – Time variant matrix.
$\vec{U}$ – Input or control vector.
$C(t)$ – Time variant matrix.
$D(t)$ – Time variant matrix.
$A_1, \dots, A_\mathrm{n}$ – Time invariant matrices.
$B_1, \dots, B_\mathrm{n}$ – Time invariant matrices.
$C_1, \dots, C_\mathrm{n}$ – Time invariant matrices.
$D_1, \dots, D_\mathrm{n}$ – Time invariant matrices.
t – Time.
$f(t)$ – Single variable function.
$\vec{X}(k)$ – Discrete time state vector.
$f(\vec{X},\vec{U})$ – Function with multiple vectorial inputs.
$h(\vec{X},\vec{U})$ – Function with multiple vectorial inputs.
$\vec{Y}$ – Output vector.
$f^{(n)}$ – Nth derivative of function $f(t)$.
t^* – Chosen local point for function $f(t)$.
$A(k)$ – Jacobian matrix.
$B(k)$ – Jacobian matrix.
$C(k)$ – Jacobian matrix.
$D(k)$ – Jacobian matrix.
T – Sampling frequency (rate).
A_D – Zero order hold equivalent matrix.
B_D – Zero order hold equivalent matrix.
$e(t)$ – Error function.
K_p – Proportional gain (or constant).
K_i – Integral gain (or constant).
K_d – Derivative gain (or constant).
$\vec{X}_d$ – Desired values for the state vector $\vec{X}$.
Q_1 – State cost matrix

Q_2 – Controller cost matrix
N – Positively defined matrix
$J(\vec{U})$ – Linear quadratic cost equation
$F_1(\vec{X})$ – Vectorial function obtained from combining other state functions
$h_2(\vec{U})$ – Input vectorial function
$f_3(\vec{X})$ – State vectorial function
$h_4(\vec{U})$ – Input vectorial function
K_1 – Gain matrix
K_2 – Gain matrix
w_v – Noise and / or perturbation matrix related to vector $\dot{\vec{V}}$
w_Ω– Noise and / or perturbation matrix related to vector $\dot{\vec{\Omega}}$
$\pi_{xy}: A_{xy}\bar{n}_{xy}$ – Finite plane π generated by normal vector $\bar{n}$ with area A
$\overrightarrow{drag_{lift}}$ – Lift induced drag force
$\overrightarrow{F_L}$ – Lift force vector
V_e – Equivalent airspeed (EAS)
ρ – Air density at current altitude
ρ_0 – Air density at sea level under ISA conditions
v – Relative air velocity
S – Total wing area
e – Wing efficiency
(AR) – Aspect ratio
A_s – Surface area
$\overrightarrow{drag_{parasitic}}$ – Parasitic drag force vector (if there is no accent then it is the absolute value of the vector)
$\overrightarrow{drag_{surface}}$– Surface drag force vector (if there is no accent then it is the absolute value of the vector)

$\overrightarrow{drag_{shape}}$– Shape drag force vector (if there is no accent then it is the absolute value of the vector)
$\overrightarrow{drag_{interference}}$– Interference drag force vector (if there is no accent then it is the absolute value of the vector)
C_{shape} – An aerodynamical coefficient that depends on the shape of the object
$C_{surface}$ – An aerodynamical coefficient that depends on the Reynold number
$\overrightarrow{C_1}$ – Vectorial coefficient related to the shape drag
$\overrightarrow{C_2}$ – Vectorial coefficient related to the surface drag
$\overrightarrow{C_3}$ – Vectorial coefficient related to the interference drag
$Proj_{\bar{v}}\bar{u}$ – Orthogonal projection of vector $\vec{u}$ in vector $\vec{v}$
$\overrightarrow{F_D}$ – Net drag force vector

SUMMARY

INTRODUCTION .... page 01

1. DESCRIPTION OF MATHEMATICAL MODELLING .... page 03

1.1 Description of the v-22 osprey dynamics and the proposed model dynamics .... page 03

1.2 Definition of norms used, coordinate systems and states .... page 04

1.3 Application of newton-euler equations .... page 06

1.4 System analisys and nonlinear simulators .... page 11

1.5 Proposed mathematical modelling of aerodynamic forces .... page 11

2. POSSIBLE IMPLEMENTATIONS OF SOME CONTROLLERS .... page 17

2.1 PID controller.. page 17

2.2 State-space feedback controller .... page 18

2.3 Linear Quadratic Regulator controller (LQR).. page 18

2.4 Discussion of hover control through feedback loop stabilization .... page 19

3 CONCLUSION AND FUTURE IMPROVEMENTS .... page 22

4 REFERENCES.. page 23

5 APPENDICES .... page 24

5.1 Appendix 1 – 3d model of the tilt rotor drone .... page 25

5.2 Appendix 2 – simulation results without control .... page 26

INTRODUCTION

The helicopter is an aircraft created with the ability to take off and land vertically (VTOL), to hover and maneuver in all directions in restricted airspaces where fixed-winged aircrafts would not be able to perform. It is, therefore, an agile aircraft, capable of following targets in constrained spaces, which led, naturally, military applications.

The origins of the helicopter date to 400 BC in China where children would play with a flying bamboo toy typically referred to as a Chinese top [10]. The toy was spun and, as a result, the spinning would create lift, enabling the top to fly once it was released. Later contributions to the creation of the helicopter came from Leonard Da Vinci during the renaissance who envisioned the helicopter as a screw and from Mikhail Lomonosov (July 1754) [10] who adapted Da Vinci's model with a wound-up spring device and demonstrated it to the Russian Academy of Science.

Many continued to contribute and work on the helicopter's design until the helicopters reached the design it is known for today, with a main rotor and a tail rotor. An earlier model worth mentioning is the German Focke-Wulf Fw 61 which, according to the company Heli-Mart (a subsidiary of California Aero Components – CAC), *"broke all of the existing 1937 world records for helicopter flight, and became a benchmark of the age" [10]*. Another important model the American R-4 helicopter, also called Sikorsky R-4, created by Igor Sikorsky, which was one of the first mass produced helicopters.

The aircrafts with tilt rotors come as an attempt to create aircrafts, which could fill the niche between fixed-wing aircrafts and helicopters. The objective would be to synergistically unite the advantages of both aircrafts, as the helicopter cannot achieve high cruising speeds when compared to the fixed-wing aircrafts. A high cruise speed is desirable, especially when large travel distances are required, for example, from a safe zone to a zone of confrontation or danger. The tilt rotors aircraft seeks, therefore, to combine the following characteristics:

1. Vertical Takeoff and Landing (VTOL)
2. Precise movement in spatially restricted airspaces
3. High cruising speeds
4. Greater stability and control

The first characteristic eliminates the need for a landing strip requiring, instead, a sufficiently big circular space, for example a forest clearing, as in the case of the helipad. The second characteristic, inherent to helicopters, is desired for the new aircraft to be agile enough, enabling it to adequately position itself even in spatially restricted airspaces. This is an important characteristic for military applications as it guarantees that the aircraft will be able to chase ground-based mobile targets with ease, maintaining positional and aerial superiority. Furthermore, the aircraft's agility would facilitate difficult maneuvers, common in rescue situations at high-risk environments (for example, in the rescue of survivors during a flood with torrential rains and strong winds).

The third characteristic is important because it enables the aircraft to reach far destinations rapidly. The reduction in time lost during flight can be critical in rescue missions and can be profitable on cargo supplying and transportation missions. The high cruise speeds also create the possibility for mid-flight refueling missions of fixed-winged aircrafts. The fourth feature is sought after in any aircraft (and most systems) because it reduces the risk of compromising the aircraft's stability, which, in the worst-case scenario, can lead to loss of the aircraft and, potentially, human lives.

The exponential growth and miniaturization of technology enabled the creation of drones and UAVs (unmanned aerial vehicles) as automated or remote controlled unmanned aerial vehicles capable of performing several tasks. Drones are capable of spying, collecting data and have been employed in geographical mapping, terrain recognition and other applications, such as aerial supervision of crop fields. The American Army concluded that the loss of soldier lives in manned aircraft missions and, consequently, the loss of investment was so considerable that many missions replaced manned aircrafts with UAVs.

The advent of drones and UAV brought many challenges and innovations to the field of control and automation. The mathematical modelling and control of flying aircrafts continue to pose challenges to this day, because the variety of aircraft types and models lead to various systems, which, typically, have multiple inputs and multiple outputs (MIMO).

These systems may also be sub actuated, nonlinear, non-affine and exhibit partial controllability and stability. The construction of tilt rotor drones, made possible by the new technologies, would require a suitable and fine-tuned mathematical model and control in order to allow the drones to be autonomous or remote controlled.

The objective of this course completion work is to develop a realistic and sophisticated mathematical model of a tilt rotor drone in modular form. This allows it to be used in the simulation and control of a tilt rotor drone (and aircraft) while allowing other models to be incorporated into it with the intent of creating larger, more complex models if required. Classical control methods and other nonlinear methods will be discussed and suggested as means to control the aircraft.

1. DESCRIPTION OF MATHEMATICAL MODELLING

The following section will describe in details the process through which the mathematical modeling was obtained. Descriptions of the dynamical behavior of the real V-22 Osprey aircraft will be discussed as well as their implementation, in mathematical terms, in the model.

1.1 Description of the v-22 osprey dynamics and the proposed model dynamics

The tilt rotor aircraft model will be based on the V-22 Osprey military aircraft because it is an example of a 'real' aircraft built by Bell Helicopter and Boeing Helicopters at the request of the US Army. Its dynamics serve as a basis for describing the behavior of an aircraft with generic tilting rotors. Assumptions and hypothesis will be made to simplify the state-space model and to accommodate the large-scale difference.

The yaw control in the V-22 Osprey is obtained through control of the differential longitudinal cyclic pitch, which displaces the pressure angle resulting in a yaw moment around the center of mass while maintaining a net vertical thrust. This method will be simplified in the mathematical model by counter-rotations of the wing-arms and motors as shown in Fig. 1 During cruise flight the motors, typically vertically aligned due to takeoff, need to be able to tilt until they are horizontal and, as a result, they will be designed to be able to rotate slightly more than 90 °.

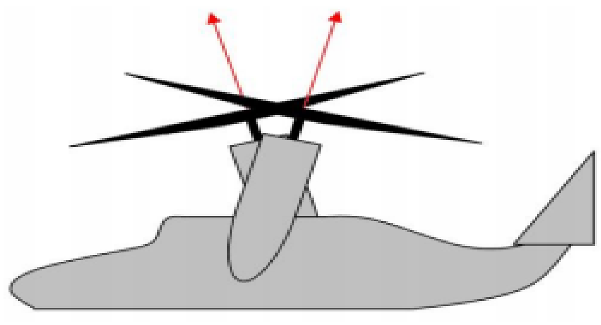


Fig. 1 – Wing-arms and motors counter-rotations.

The pitch control, in practice, is accomplished by the use of various complex techniques, such as pendulum balancing and longitudinal cyclic step control. The mathematical model will add a vertically aligned tail rotor instead of utilizing flaps as this will provide a control method dissociated from the yaw control to simplify the pitch control (figure 2). This method is much simpler to utilize than those used in the V-22 Osprey and it mainly compromises the aesthetic appeal of the original aircraft.

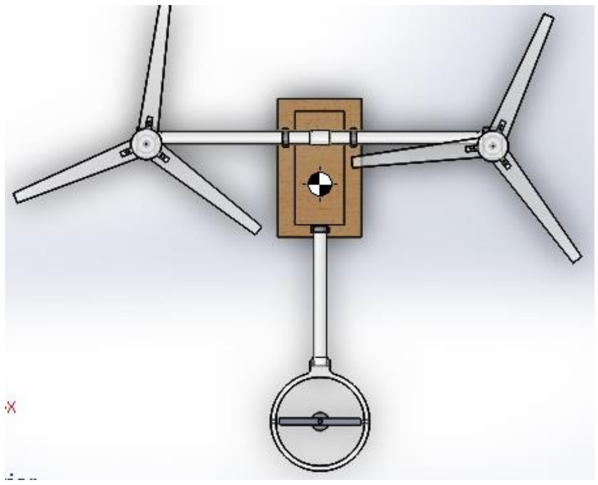


Fig. 2 – Alteration of flaps to back propeller

The roll control is produced by creating a pressure differential on each propeller. This differential impulse creates a moment around the center of mass of the aircraft, causing a rotation about its longitudinal axis. The V-22 Osprey creates this pressure differential by adjusting the angular velocity of each propeller since the overall speed control is slower due to the big inertia of the rotating propeller blades.

The model will use two separate engines and will create the differential boost simply by differences in the angular velocities of each engine. An important observation is the lack of considerable moment of inertia of the propeller blade on a drone since it has small and light propellers. Their physical dimensions and properties mean that the moment of inertia does not prevent fast rates of change of the motors angular velocity.

The vertical translation control is obtained, in the V-22 Osprey, by the collective change of the impulse in each propeller that is consequence of the collective alteration of the angular velocities of the motors. The V-22 Osprey is able to adjust the longitudinal angle between the wing and the rotors by tilting them laterally. This feature will be neglected in the mathematical model due to the associated mechanical, technical and control complexities. The horizontal translation control is obtained, therefore, through the natural coupling with the pitch and roll.

1.2 Definition of norms used, coordinate systems and states

The coordinate systems chosen will be defined according to the DIM 9300 norm as this is commonly used norm to describe aircrafts. The coordinate system will have the vertical axis defined as the Z-axis with downwards being the positive direction. The longitudinal axis is defined as the X-axis and aligned with the drone. The front of the drone is the positive direction. The transversal axis is defined as the Y-axis. If observed in the YZ plane, the left is defined as the positive Y-axis direction. The resulting coordinate system is shown in Fig 2.

The equations of motion will be derived using the coordinate system fixed to the drone and will be called $O_F x_F y_F z_F$ (Fig. 3). The origin of this coordinate system O_F is chosen to coincide with the center of mass of the drone. It is noted that this coordinate system is free to rotate with the drone. Another coordinate system is required in order to fully derive the equations of motion. This second coordinate system will be an inertial coordinate system $O_I x_I y_I z_I$ whose axis are parallel to that of the fixed to the drone's frame coordinate system $O_F x_F y_F z_F$ and with the same directions. The origin of this system can be placed anywhere in space.

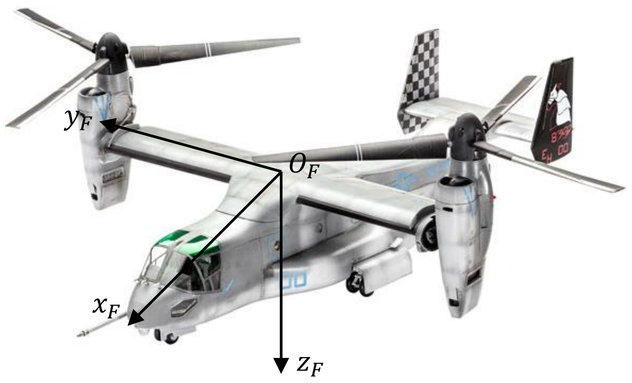


Fig. 3 – Coordinate system fixed to the drone's body $O_F x_F y_F z_F$.

A state can be expressed in either the inertial reference frame using the inertial coordinate system $O_I x_I y_I z_I$ or in the non-inertial reference frame using the fixed to the drone coordinate system $O_F x_F y_F z_F$, referred to as 'fixed coordinate system'. Since the same state can be written using two different reference frames, said frames share a basis change relationship between them. The linear transformation matrix that brings a state from the fixed coordinate system to the inertial coordinate system is the three dimensions rotation matrix.

The rotation matrix $R(\phi,\theta,\psi) \in \mathbb{R}^3$ can be decomposed into three matrices, each corresponding to a rotation along an axis. These matrices represent the yaw, pitch and roll movement. The convention chosen for the rotation will be the $z-y-x$ intrinsic rotations convention and the yaw, pitch and roll angles will be, respectively, ϕ,θ,ψ such that:

$$R(\phi,\theta,\psi) = R_z(\phi)R_y(\theta)R_x(\psi) \qquad \textbf{[1]}$$

Where:

$$R_x(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi) & \sin(\phi) \\ 0 & -\sin(\phi) & \cos(\phi) \end{bmatrix} \textit{ is the roll matrix} \qquad \textbf{[2]}$$

$$R_y(\theta) = \begin{bmatrix} \cos(\theta) & 0 & -\sin(\theta) \\ 0 & 1 & 0 \\ \sin(\theta) & 0 & \cos(\theta) \end{bmatrix} \textit{ is the pitch matrix} \qquad \textbf{[3]}$$

$$R_z(\psi) = \begin{bmatrix} \cos(\psi) & \sin(\psi) & 0 \\ -\sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \textit{ is the yaw matrix} \qquad \textbf{[4]}$$

The orientation and rotation of the drone can, therefore, be describe through the matrix $R(\phi,\theta,\psi)$ assuming the rotations are always performed in the order shown above (roll, pitch and yaw). The matrix $R(\phi,\theta,\psi)$ can be obtained by the product of each intrinsic rotation:

$$\therefore R(\phi,\theta,\psi) = R_z(\phi)R_y(\theta)R_x(\psi) =$$

$$\begin{bmatrix} \cos(\psi) & \sin(\psi) & 0 \\ -\sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos(\theta) & 0 & -\sin(\theta) \\ 0 & 1 & 0 \\ \sin(\theta) & 0 & \cos(\theta) \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi) & \sin(\phi) \\ 0 & -\sin(\phi) & \cos(\phi) \end{bmatrix} = \qquad \textbf{[5]}$$

$$\begin{bmatrix} \cos(\psi)\cos(\theta) & -\sin(\psi)\cos(\phi) + \cos(\psi)\sin(\theta)\sin(\phi) & \sin(\psi)\sin(\phi) + \cos(\psi)\sin(\theta)\cos(\phi) \\ \sin(\psi)\cos(\theta) & \cos(\psi)\cos(\phi) + \sin(\psi)\sin(\theta)\sin(\phi) & -\cos(\psi)\sin(\phi) + \sin(\psi)\sin(\theta)\cos(\phi) \\ -\sin(\theta) & \cos(\theta)\sin(\phi) & \cos(\theta)\cos(\phi) \end{bmatrix}$$

In order to reduce the size of a matrix to a more comprehensible and compact form, let $\cos(\alpha) = c_\alpha$ and $\sin(\alpha) = s_\alpha$. The complete rotation matrix $R(\phi,\theta,\psi)$ can be rewritten as:

$$R(\phi,\theta,\psi) = \begin{bmatrix} c_\psi c_\theta & -s_\psi c_\phi + c_\psi s_\theta s_\phi & s_\psi s_\phi + c_\psi s_\theta c_\phi \\ s_\psi c_\theta & c_\psi c_\phi + s_\psi s_\theta s_\phi & -c_\psi s_\phi + s_\psi s_\theta c_\phi \\ -s_\theta & c_\theta s_\phi & c_\theta c_\phi \end{bmatrix} \qquad \textbf{[6]}$$

The complete movement of a rigid body can be described through its position, orientation, velocity and angular velocity. The states are chosen as to describe the complete movement of the drone and, as a result, the choice of position, velocity, orientation and angular velocity is natural. The complete state is:

$$\vec{X} = \begin{bmatrix} \vec{\mathrm{P}} \\ \vec{V} \\ \vec{\Xi} \\ \vec{\Omega} \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \\ v_x \\ v_y \\ v_z \\ \phi \\ \theta \\ \psi \\ \omega_\phi \\ \omega_\theta \\ \omega_\psi \end{bmatrix} \qquad \textbf{[7]}$$

Where $\vec{\mathrm{P}}$ is the position vector, $\vec{V}$ is the velocity vector, $\vec{\Xi}$ is the angle vector (more commonly called orientation vector or attitude vector) and $\vec{\Omega}$ is the angular velocity vector.

1.3 APPLICATION OF NEWTON-EULER EQUATIONS

The mathematical model will use the Newton-Euler formulation to obtain the dynamic equations for the motion of the drone in the fixed reference frame $O_F x_F y_F z_F$. The Newton-Euler formulation states that the motion of a rigid body can be describe through two equations, one related to the summation of forces and one related to the summation of moments, both of which are all acting upon the rigid body:

$$\begin{cases} \vec{F} = m\dot{\vec{V}} + \vec{\Omega} \times (m\vec{V}) \\ \vec{M} = J\dot{\vec{\Omega}} + \vec{\Omega} \times (J\vec{\Omega}) \end{cases} \qquad [\mathbf{8}]$$

In the above equation, J is an inertia matrix, $\vec{F}$ is the summation of forces and $\vec{M}$ is the summation of moments. This means that:

$$\begin{cases} \vec{F} = \vec{F}_{prop} + \vec{F}_{drag} + \vec{F}_g \\ \vec{M} = \vec{M}_{prop} + \vec{M}_{drag} + \vec{M}_{gyro} \end{cases} \qquad [\mathbf{9}]$$

Where $\vec{F}_{prop}$ and $\vec{M}_{prop}$ are the net propulsion force and moment respectively, $\vec{F}_{drag}$ and $\vec{M}_{drag}$ are the net aerodynamic drag force and moment respectively, $\vec{F}_g$ is the force of gravity and $\vec{M}_{gyro}$ is the gyroscopic moment of the tilting rotors.

To determine $\vec{F}_{prop}$, the propulsion force generated by one propeller will be modelled as:

$$F_{propeller} = k_{propeller}\vec{\omega}_{propeller}{}^2 \qquad [\mathbf{10}]$$

Where $k_{propeller}$ is a constant relating the angular velocity of the rotor and the lift generated and $\vec{\omega}_{propeller}$ is the angular velocity of said rotor. Both frontal propellers can be tilted by an angle δ as shown in Fig. 4. These tilt angles are unique for each propeller, that is, they are not coupled and are independently controlled. The vectorial decomposition of propeller forces is the same for each of the frontal propellers:

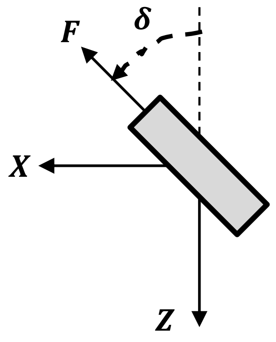


Fig. 4 – Figure depicting the tilting of a propeller.

$$\begin{cases} \vec{F}_{lp}{}^T = \begin{bmatrix} \sin(\delta_{lp}) & 0 & -\cos(\delta_{lp}) \end{bmatrix} k_{lp}\vec{\omega}_{lp}{}^2 \ \ (left\, propeller) \\ \vec{F}_{rp}{}^T = \begin{bmatrix} \sin(\delta_{rp}) & 0 & -\cos(\delta_{rp}) \end{bmatrix} k_{rp}\vec{\omega}_{rp}{}^2 \ \ (right\, propeller) \end{cases} \qquad [\mathbf{11}]$$

In order to ensure the model is generic, propellers can either have opposing directions for the angular velocities or the same direction for them (typically they have an opposing direction to compensate for the gyroscopic effect). To allow this to be changed by a given controller or by a parameter in case of a simulation, the '$sign$' (or 'sgn') function is used. The '$sign$' function is given by the following mathematical expression:

$$sign(\omega) = \begin{cases} -1, \omega < 0 \\ 1, \omega \geq 0 \end{cases} \qquad [12]$$

$$sgn(\omega) \equiv sign(\omega) \qquad [13]$$

The resulting equation for the frontal propeller forces becomes:

$$\begin{cases} \vec{F}_{lp}{}^{T} = [\sin(\delta_{lp}) \quad 0 \quad -\cos(\delta_{lp})] sgn(\omega_{lp}) k_{lp} \omega_{lp}{}^{2} \ \ (left\,propeller) \\ \vec{F}_{rp}{}^{T} = [\sin(\delta_{rp}) \quad 0 \quad -\cos(\delta_{rp})] sgn(\omega_{rp}) k_{rp} \omega_{rp}{}^{2} \ \ (right\,propeller) \end{cases} \qquad [14]$$

The back propeller (or tail propeller) is fixed parallel to the Z-axis such that:

$$\vec{F}_{bp}{}^{T} = [0 \quad 0 \quad -1] sgn(\omega_{bp}) k_{bp} \omega_{bp}{}^{2} \ \ (back\,propeller) \qquad [15]$$

The resulting propeller forces $\vec{F}_{prop}$ can, therefore, be written in matrix form as:

$$\vec{F}_{prop} = \begin{bmatrix} \sin(\delta_{lp}) & \sin(\delta_{rp}) & 0 \\ 0 & 0 & 0 \\ -\cos(\delta_{lp}) & -\cos(\delta_{rp}) & -1 \end{bmatrix} \begin{bmatrix} sgn(\omega_{lp}) k_{lp} \omega_{lp}{}^{2} \\ sgn(\omega_{rp}) k_{rp} \omega_{rp}{}^{2} \\ sgn(\omega_{bp}) k_{bp} \omega_{bp}{}^{2} \end{bmatrix} \qquad [16]$$

$$\therefore \vec{F}_{prop} = \begin{bmatrix} s_{\delta_{lp}} & s_{\delta_{rp}} & 0 \\ 0 & 0 & 0 \\ -c_{\delta_{lp}} & -c_{\delta_{rp}} & -1 \end{bmatrix} \begin{bmatrix} sgn(\omega_{lp}) k_{lp} \omega_{lp}{}^{2} \\ sgn(\omega_{rp}) k_{rp} \omega_{rp}{}^{2} \\ sgn(\omega_{bp}) k_{bp} \omega_{bp}{}^{2} \end{bmatrix} \qquad [17]$$

The next force to determine is the force of gravity $\vec{F}_g$ which is given, in the inertial reference frame, by the expression:

$$\vec{F}_g{}^{T} = [0 \quad 0 \quad 1] mg \qquad [18]$$

In order to write $\vec{F}_g$ in the fixed (and non-inertial) reference frame, it is necessary to apply a basis change transformation matrix:

$$\left(\vec{F}_g{}^{T}\right)_F = R(\phi, \theta, \psi) \left(\vec{F}_g{}^{T}\right)_I \qquad [19]$$

Where $\left(\vec{F}_g{}^{T}\right)_F$ and $\left(\vec{F}_g{}^{T}\right)_I$ is the force of gravity written in the fixed basis and inertial basis respectively. Since the Newton-Euler formulation is being applied on the fixed reference frame, the force of gravity can be written as:

$$\vec{F}_g = R(\phi, \theta, \psi) \left(\vec{F}_g{}^{T}\right)_I = \begin{bmatrix} c_\psi c_\theta & -s_\psi c_\phi + c_\psi s_\theta s_\phi & s_\psi s_\phi + c_\psi s_\theta c_\phi \\ s_\psi c_\theta & c_\psi c_\phi + s_\psi s_\theta s_\phi & -c_\psi s_\phi + s_\psi s_\theta c_\phi \\ -s_\theta & c_\theta s_\phi & c_\theta c_\phi \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} mg \qquad [20]$$

$$\vec{F}_g{}^{T} = \begin{bmatrix} s_\psi s_\phi + c_\psi s_\theta c_\phi \\ -c_\psi s_\phi + s_\psi s_\theta c_\phi \\ c_\theta c_\phi \end{bmatrix} mg \qquad [21]$$

Initially, the aerodynamic drag force and moment will be consider to have negligible effect on the drone's dynamics and, as a result, $\vec{F}_{drag} = \vec{M}_{drag} = \vec{0}$. To determine $\vec{M}_{prop}$, it is necessary to determine all components of each torque, created by each force acting upon the drone. The relationship between force and torque, or moment of force, is given by the following equation:

$$\vec{\tau} = \vec{r} \times \vec{F} \qquad [22]$$

Where $\vec{r}$ is the position vector. Let $\vec{r}_{cm\to rp}$,$\vec{r}_{cm\to lp}$ and $\vec{r}_{cm\to bp}$ be position vectors between the center of mass of the drone and the respective point of actuation of the lift force for each rotor. The respective torques are, therefore:

$$\vec{\tau}_{lp} = \vec{r}_{cm\to lp} \times \vec{F}_{lp} = \begin{bmatrix} x_{cm\to lp} \\ y_{cm\to lp} \\ z_{cm\to lp} \end{bmatrix} \times \begin{bmatrix} s_{\delta_{lp}} \\ 0 \\ -c_{\delta_{lp}} \end{bmatrix} k_{lp}\omega_{lp}{}^2 = \begin{bmatrix} -y_{cm\to lp}c_{\delta_{lp}} \\ z_{cm\to lp}s_{\delta_{lp}} + x_{cm\to lp}c_{\delta_{lp}} \\ -y_{cm\to lp}s_{\delta_{lp}} \end{bmatrix} k_{lp}\omega_{lp}{}^2 \qquad [23]$$

$$\vec{\tau}_{rp} = \vec{r}_{cm\to rp} \times \vec{F}_{rp} = \begin{bmatrix} x_{cm\to rp} \\ y_{cm\to rp} \\ z_{cm\to rp} \end{bmatrix} \times \begin{bmatrix} s_{\delta_{rp}} \\ 0 \\ -c_{\delta_{rp}} \end{bmatrix} k_{rp}\omega_{rp}{}^2 = \begin{bmatrix} -y_{cm\to rp}c_{\delta_{rp}} \\ z_{cm\to rp}s_{\delta_{rp}} + x_{cm\to rp}c_{\delta_{rp}} \\ -y_{cm\to rp}s_{\delta_{rp}} \end{bmatrix} k_{rp}\omega_{rp}{}^2 \qquad [24]$$

$$\vec{\tau}_{bp} = \vec{r}_{cm\to bp} \times \vec{F}_{bp} = \begin{bmatrix} x_{cm\to bp} \\ y_{cm\to bp} \\ z_{cm\to bp} \end{bmatrix} \times \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} k_{bp}\omega_{bp}{}^2 = \begin{bmatrix} y_{cm\to bp} \\ -x_{cm\to bp} \\ 0 \end{bmatrix} k_{bp}\omega_{bp}{}^2 \qquad [25]$$

Adding the '$sign$' function and writing the compact matrix form for all propeller torques $\vec{\tau}_{prop}$:

$$\vec{\tau}_{prop} = \begin{bmatrix} -y_{cm\to lp}c_{\delta_{lp}} & -y_{cm\to rp}c_{\delta_{rp}} & y_{cm\to bp} \\ z_{cm\to lp}s_{\delta_{lp}} + x_{cm\to lp}c_{\delta_{lp}} & z_{cm\to rp}s_{\delta_{rp}} + x_{cm\to rp}c_{\delta_{rp}} & -x_{cm\to bp} \\ -y_{cm\to lp}s_{\delta_{lp}} & -y_{cm\to rp}s_{\delta_{rp}} & 0 \end{bmatrix} \begin{bmatrix} sgn(\omega_{lp})k_{lp}\omega_{lp}{}^2 \\ sgn(\omega_{rp})k_{rp}\omega_{rp}{}^2 \\ sgn(\omega_{bp})k_{bp}\omega_{bp}{}^2 \end{bmatrix} \qquad [26]$$

The gravitational force is treated as a net force acting on the drone's center of mass and, as a result, produces no net torque. Since two rotors are also being rotated, the gyroscopic effect must be considered in this model. The total gyroscopic moment is the sum of each gyroscopic moment given by the formula (KENDOUL, FANTONI, and LOZANO, "Modeling and Control of a Small Autonomous Aircraft Having Two Tilting Rotors"[12]):

$$\vec{M}_{gyro} = J\left(\vec{\dot{\delta}} \times \frac{\vec{F}}{|\vec{F}|}\right)\omega \qquad [27]$$

Each frontal rotor are able to tilt around the Y-axis with angular velocity $\dot{\delta}$. This angular velocity will be defined by the servomotor's nominal speed (assuming a servomotor is used to produce the tilting of the propellers). Applying the above equation to each propeller:

$$\vec{M}_{gyro,lp} = J\left(\vec{\dot{\delta}} \times \frac{\vec{F}_{lp}}{|\vec{F}_{lp}|}\right)|\vec{\omega}_{lp}| = \begin{bmatrix} J_{xx} & J_{xy} & J_{xz} \\ J_{yx} & J_{yy} & J_{yz} \\ J_{zx} & J_{zy} & J_{zz} \end{bmatrix} \left(\dot{\delta}\begin{bmatrix} 0 \\ -1 \\ 0 \end{bmatrix} \times \begin{bmatrix} s_{\delta_{lp}} \\ 0 \\ -c_{\delta_{lp}} \end{bmatrix}\right) k_{lp}\omega_{lp}{}^2 \qquad [28]$$

$$\vec{M}_{gyro,lp} = \begin{bmatrix} J_{xx} & J_{xy} & J_{xz} \\ J_{yx} & J_{yy} & J_{yz} \\ J_{zx} & J_{zy} & J_{zz} \end{bmatrix} \begin{bmatrix} c_{\delta_{lp}} \\ 0 \\ s_{\delta_{lp}} \end{bmatrix} k_{lp}\dot{\delta}\omega_{lp}{}^2 = \begin{bmatrix} J_{xx}c_{\delta_{lp}} + J_{xz}s_{\delta_{lp}} \\ J_{yx}c_{\delta_{lp}} + J_{yz}s_{\delta_{lp}} \\ J_{zx}c_{\delta_{lp}} + J_{zz}s_{\delta_{lp}} \end{bmatrix} k_{lp}\dot{\delta}\omega_{lp}{}^2 \qquad [29]$$

$$\vec{M}_{gyro,rp} = J\left(\vec{\dot{\delta}} \times \frac{\vec{F}_{rp}}{|\vec{F}_{rp}|}\right)|\vec{\omega}_{rp}| = \begin{bmatrix} J_{xx} & J_{xy} & J_{xz} \\ J_{yx} & J_{yy} & J_{yz} \\ J_{zx} & J_{zy} & J_{zz} \end{bmatrix} \left(\dot{\delta}\begin{bmatrix} 0 \\ -1 \\ 0 \end{bmatrix} \times \begin{bmatrix} s_{\delta_{rp}} \\ 0 \\ -c_{\delta_{rp}} \end{bmatrix}\right) k_{rp}\omega_{rp}{}^2 \qquad [30]$$

$$\vec{M}_{gyro,rp} = \begin{bmatrix} J_{xx} & J_{xy} & J_{xz} \\ J_{yx} & J_{yy} & J_{yz} \\ J_{zx} & J_{zy} & J_{zz} \end{bmatrix} \begin{bmatrix} c_{\delta_{rp}} \\ 0 \\ s_{\delta_{rp}} \end{bmatrix} k_{rp}\dot{\delta}\omega_{rp}{}^2 = \begin{bmatrix} J_{xx}c_{\delta_{rp}} + J_{xz}s_{\delta_{rp}} \\ J_{yx}c_{\delta_{rp}} + J_{yz}s_{\delta_{rp}} \\ J_{zx}c_{\delta_{rp}} + J_{zz}s_{\delta_{rp}} \end{bmatrix} k_{rp}\dot{\delta}\omega_{rp}{}^2 \qquad [31]$$

$$\vec{M}_{gyro,bp} = J\left(\dot{\vec{\delta}} \times \frac{\vec{F}_{bp}}{|\vec{F}_{bp}|}\right)|\vec{\omega}_{bp}| = \begin{bmatrix} J_{xx} & J_{xy} & J_{xz} \\ J_{yx} & J_{yy} & J_{yz} \\ J_{zx} & J_{zy} & J_{zz} \end{bmatrix} \left(\dot{\delta}\begin{bmatrix} 0 \\ -1 \\ 0 \end{bmatrix} \times \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}\right) k_{bp}\omega_{bp}{}^2 \qquad [32]$$

$$\vec{M}_{gyro,bp} = \begin{bmatrix} J_{xx} & J_{xy} & J_{xz} \\ J_{yx} & J_{yy} & J_{yz} \\ J_{zx} & J_{zy} & J_{zz} \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} k_{bp}\dot{\delta}\omega_{bp}{}^2 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} k_{rp}\dot{\delta}\omega_{rp}{}^2 = \vec{0} \qquad [33]$$

The back propeller does not produce a gyroscopic effect, naturally, because it is fixed to the body of the drone and it cannot tilt like the frontal propellers. Combining the results in matrix form $\vec{M}_{gyro,prop}$ and adding the '$sign$' function:

$$\vec{M}_{gyro,prop} = \begin{bmatrix} J_{xx}c_{\delta_{lp}} + J_{xz}s_{\delta_{lp}} & J_{xx}c_{\delta_{rp}} + J_{xz}s_{\delta_{rp}} & 0 \\ J_{yx}c_{\delta_{lp}} + J_{yz}s_{\delta_{lp}} & J_{yx}c_{\delta_{rp}} + J_{yz}s_{\delta_{rp}} & 0 \\ J_{zx}c_{\delta_{lp}} + J_{zz}s_{\delta_{lp}} & J_{zx}c_{\delta_{rp}} + J_{zz}s_{\delta_{rp}} & 0 \end{bmatrix} \begin{bmatrix} sgn(\omega_{lp})k_{lp}\omega_{lp}{}^2 \\ sgn(\omega_{rp})k_{rp}\omega_{rp}{}^2 \\ sgn(\omega_{bp})k_{bp}\omega_{bp}{}^2 \end{bmatrix} \qquad [34]$$

To obtain the differential equations of the system it is necessary to apply the obtained relationships to the Newton-Euler formulation [3, 7, 12]. The expressions will be given in their reduced form as expanding them would result in impractically big equations.

$$\begin{cases} \vec{F} = \vec{F}_{prop} + \vec{F}_{drag} + \vec{F}_g \cong \vec{F} = \vec{F}_{prop} + \vec{F}_g \\ \vec{F} = m\dot{\vec{V}} + \vec{\Omega} \times (m\vec{V}) \end{cases} \qquad [35]$$

$$\therefore \dot{\vec{V}} = \tfrac{1}{m}\left[\vec{F} - \vec{\Omega} \times (m\vec{V})\right] \qquad [36]$$

$$\begin{cases} \vec{M} = \vec{M}_{prop} + \vec{M}_{drag} + \vec{M}_{gyro} \cong \vec{M}_{prop} + \vec{M}_{gyro} \\ \vec{M} = J\dot{\vec{\Omega}} + \vec{\Omega} \times (J\vec{\Omega}) \end{cases} \qquad [37]$$

$$\therefore \dot{\vec{\Omega}} = J^{-1}\left[\vec{M} - \vec{\Omega} \times (J\vec{\Omega})\right] \qquad [38]$$

The above differential equations can be written as two separate functions $f(\vec{X})$ and $h(\vec{U})$. Considering the first Newton-Euler equation:

$$\dot{\vec{V}} = \frac{1}{m}\left[\vec{F} - \vec{\Omega} \times (m\vec{V})\right] = \frac{\vec{F}}{m} - \frac{1}{m}\left[\vec{\Omega} \times (m\vec{V})\right] \qquad [39]$$

Let:

$$g_1(\vec{X}, \vec{U}) = -\tfrac{1}{m}\left[\vec{\Omega} \times (m\vec{V})\right] \qquad [40]$$

$$g_2(\vec{X}, \vec{U}) = \tfrac{\vec{F}}{m} = \tfrac{1}{m}\left[\vec{F}_{prop}(\vec{U}) + \vec{F}_g(\vec{X})\right] \qquad [41]$$

The above functions can both be dissociated into a sum of $f(\vec{X})$ and $h(\vec{U})$ functions:

$$f_1(\vec{X}) = -\tfrac{1}{m}\left[\vec{\Omega} \times (m\vec{V})\right] \qquad [42]$$

$$h_1(\vec{U}) = 0 \qquad [43]$$

$$\therefore g_1(\vec{X}, \vec{U}) = f_1(\vec{X}) + h_1(\vec{U}) = f_1(\vec{X}) \qquad [44]$$

$$f_2(\vec{X}) = \tfrac{\vec{F}_g(\vec{X})}{m} \qquad [45]$$

$$h_2(\vec{U}) = \frac{\vec{F}_{prop}(\vec{U})}{m} \quad [46]$$

$$\therefore g_2(\vec{X},\vec{U}) = f_2(\vec{X}) + h_2(\vec{U}) \quad [47]$$

$$\therefore \dot{\vec{V}} = g_1(\vec{X},\vec{U}) + g_2(\vec{X},\vec{U}) = f_1(\vec{X}) + f_2(\vec{X}) + h_2(\vec{U}) \quad [48]$$

Combining $f_1(\vec{X})$ and $f_2(\vec{X})$ into a new function $F_1(\vec{X})$:

$$F_1(\vec{X}) = f_1(\vec{X}) + f_2(\vec{X}) \quad [49]$$

$$\therefore \dot{\vec{V}} = f_1(\vec{X}) + f_2(\vec{X}) + h_2(\vec{U}) = F_1(\vec{X}) + h_2(\vec{U}) \quad [50]$$

Applying the same procedure to the second Newton-Euler equation:

$$\dot{\vec{\Omega}} = J^{-1}[\vec{M} - \vec{\Omega} \times (J\vec{\Omega})] = J^{-1}\vec{M} + J^{-1}[-\vec{\Omega} \times (J\vec{\Omega})] \quad [51]$$

Let:

$$g_3(\vec{X},\vec{U}) = J^{-1}[-\vec{\Omega} \times (J\vec{\Omega})] \quad [52]$$

$$g_2(\vec{X},\vec{U}) = J^{-1}\vec{M} = J^{-1}[\vec{M}_{prop}(\vec{U}) + \vec{M}_{gyro}(\vec{U})] \quad [53]$$

Similarly to what was done previously, the above functions can also be dissociated into a sum of $f(\vec{X})$ and $h(\vec{U})$ functions:

$$f_3(\vec{X}) = J^{-1}[-\vec{\Omega} \times (J\vec{\Omega})] \quad [54]$$

$$h_3(\vec{U}) = 0 \quad [55]$$

$$\therefore g_3(\vec{X},\vec{U}) = f_3(\vec{X}) + h_3(\vec{U}) = f_3(\vec{X}) \quad [56]$$

$$f_4(\vec{X}) = 0 \quad [57]$$

$$h_4(\vec{U}) = J^{-1}[\vec{M}_{prop}(\vec{U}) + \vec{M}_{gyro}(\vec{U})] \quad [58]$$

$$\therefore g_4(\vec{X},\vec{U}) = f_4(\vec{X}) + h_4(\vec{U}) = h_4(\vec{U}) \quad [59]$$

$$\dot{\vec{\Omega}} = g_3(\vec{X},\vec{U}) + g_4(\vec{X},\vec{U}) = f_3(\vec{X}) + h_4(\vec{U}) \quad [60]$$

1.4 System analysis nonlinear simulators

Analyzing equations $[54] - [60]$ it was possible to observe some nonlinearities particularly in, but not restricted to, the inputs. This type of system is called non-affine in input system, that is, a system in which the input does not appear linearly (or in a linear form) [4, 5, 6]. Non-affine systems are, usually, harder to control than affine systems. Typical control strategies for affine systems, such as feedback loop stabilization and other techniques often require complex adaptations for non-affine systems, with mixed results.

Computer simulations are attempts to represent behaviors of, typically, real systems or phenomena. They reproduce such behaviors, generally, though the use of mathematical models and have become a powerful and useful tool, capable of providing insight as to how systems behave and work, even if the 'why' is not clear (STROGATZ, "The End of Insight") [20].

A computer simulation is the execution of an actual computer model which, because it is composed of equations, relationships and algorithms, that describe a certain behavior, typically assumes the form of a mathematical model. One of the most common uses for computer simulation, in science, is to solve differential equations that cannot be solved analytically or that would be to ineffective and inefficient to be solved analytically. It is also used extensively in the control and automation field to simulate space state models.

There are many ways to analyze and simulate dynamic systems but three common approaches are solving differential equations, space state model analysis and transfer function analysis. Solving differential equations are often expensive in terms of computer power and time. Differential equations algorithms, however, are often used by mathematical programs when performing certain types of simulations, especially if such simulation is defined "continuously" (defined and modelled in continuous time although, strictly speaking, since it is a computer simulation, it is necessarily discrete). The transfer function approach has many drawbacks, however, which render it unable to adequately analyze the system studied here. These drawbacks include:

- definition under zero initial conditions (which will not be the case for certain types of movements)
- can only be applied to linear time invariant systems (the system studied here is both nonlinear and time variant)
- it does not give any information about the internal states of the system
- it cannot be applied to MIMO systems (there are techniques which attempt to circumvent this restriction by creating matrices of transfer functions)

The space state model [17] is comparatively more effective because it has as advantages many of the transfer function's disadvantages. It can be applied to nonlinear, time variant MIMO systems and gives information about the behavior of the systems states. A nonlinear, time variant MIMO space state model is defined by a system dynamic equation and an output equation:

$$\begin{cases} \dot{\vec{X}} = A(t)\vec{X} + B(t)\vec{U} \\ \vec{Y} = C(t)\vec{X} + D(t)\vec{U} \end{cases} \qquad [\mathbf{61}]$$

Using the mathematical computation program MATLAB two simulators were created, one modelled in continuous time and the other one modelled in discrete time [16]. The continuous time simulator was created using MATLAB's simulation tool "Simulink" and consists of the implementation, in block diagrams, of the Newton-Euler equations. The discrete time simulator was implemented as a MATLAB script and, thusly, it is a code line simulator.

A strategy to implement a nonlinear system simulation is to perform successive linearization around different points and treat the nonlinear system as linear in each region [16]. A time variant system, written in the form of state space, has matrices A, B, C, D that vary with time as shown by the equation above. It is necessary, then, to introduce a linearization frequency, that is, a period in which the system is assumed linear around a previously defined point. It can be said, using this assumption, that the matrices of the system are, in fact, constant for a given time interval:

$$A(t) = \begin{cases} A_1, t \in [0 \quad t_1[\\ A_2, t \in [t_1 \quad t_2[\\ \vdots \\ A_n, t \in [t_{n-1} \quad t_n[\\ \vdots \end{cases} \quad B(t) = \begin{cases} B_1, t \in [0 \quad t_1[\\ B_2, t \in [t_1 \quad t_2[\\ \vdots \\ B_n, t \in [t_{n-1} \quad t_n[\\ \vdots \end{cases}$$
$$C(t) = \begin{cases} C_1, t \in [0 \quad t_1[\\ C_2, t \in [t_1 \quad t_2[\\ \vdots \\ C_n, t \in [t_{n-1} \quad t_n[\\ \vdots \end{cases} \quad D(t) = \begin{cases} D_1, t \in [0 \quad t_1[\\ D_2, t \in [t_1 \quad t_2[\\ \vdots \\ D_n, t \in [t_{n-1} \quad t_n[\\ \vdots \end{cases} \qquad [\mathbf{62}]$$

The linearization frequency, naturally, must be adequately small for the system to avoid being represented with aliasing. Consider a SISO system with a behavior described by $f(t)$ and given by the following expression:

$$f(t) = e^t \sin(5t)\,, t \in [0 \quad 3] \qquad [63]$$

The following sequence of figures (Fig 5, Fig 6, Fig 7 and Fig 8) show that the effects of linearization are negligible provided the frequency is big enough in relation the frequency of the system's behavior. For the example SISO system, a frequency of approximately $33\,Hz$ already yields reasonable answers.

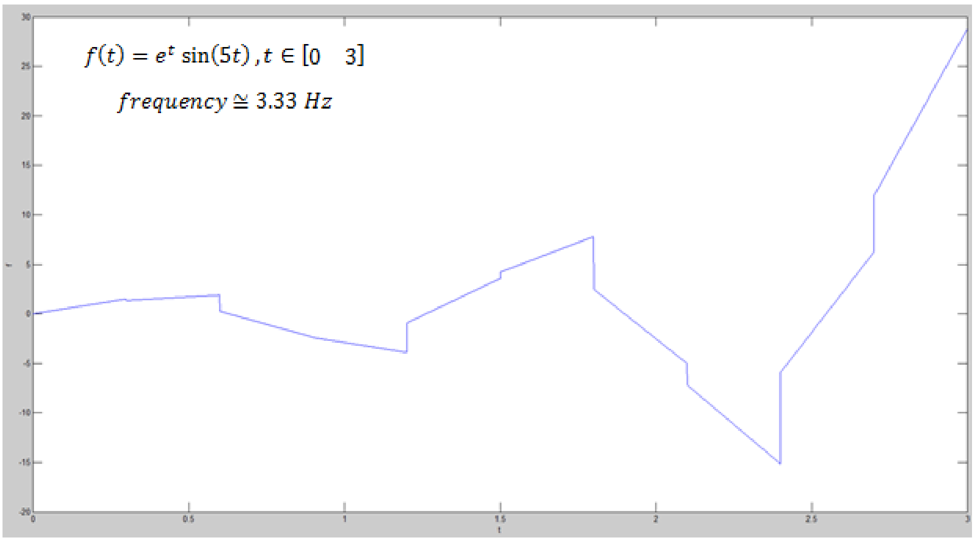


Fig. 5 - Figure depicting the linearization of $f(t)$ with a frequency of $3.33\,Hz$.

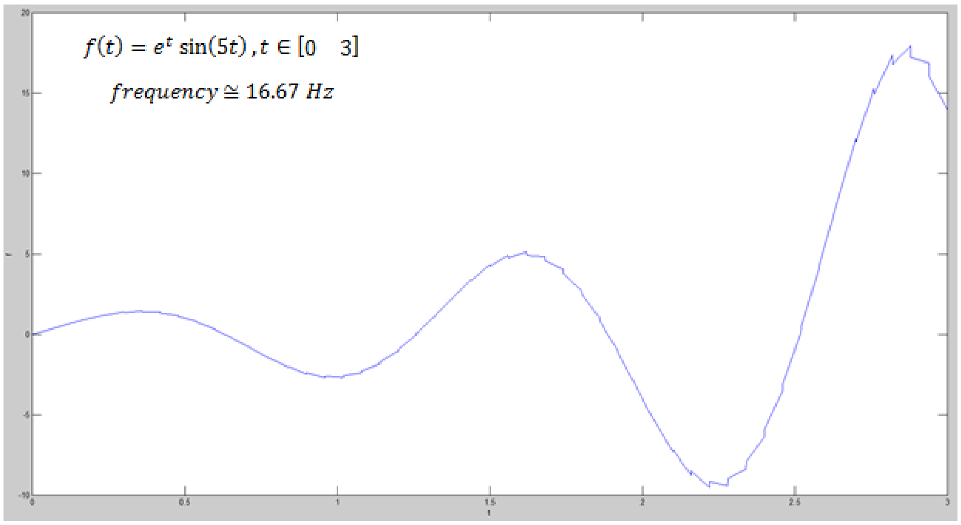


Fig 6 – Figure depicting the linearization of $f(t)$ with a frequency of $16.67\,Hz$

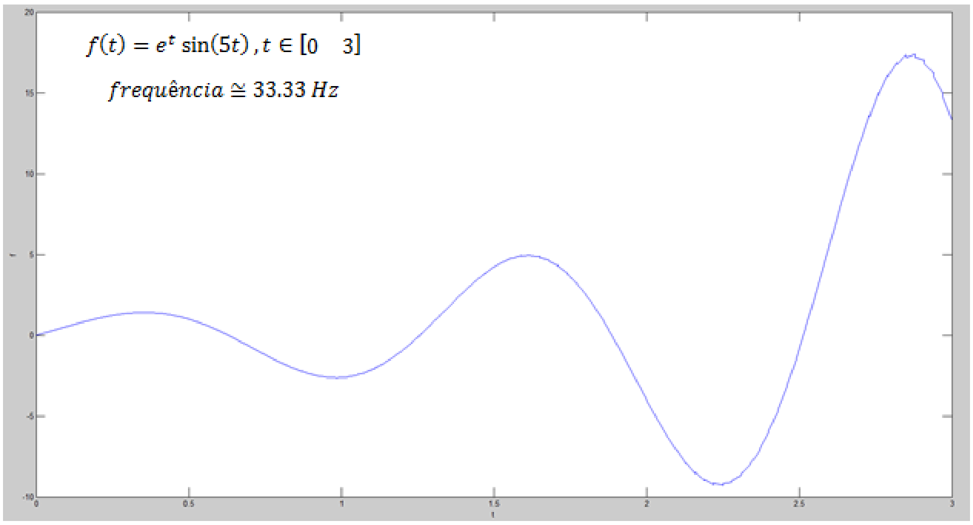


Fig 7– Figure depicting the linearization of $f(t)$ with a frequency of $33.33\ Hz$.

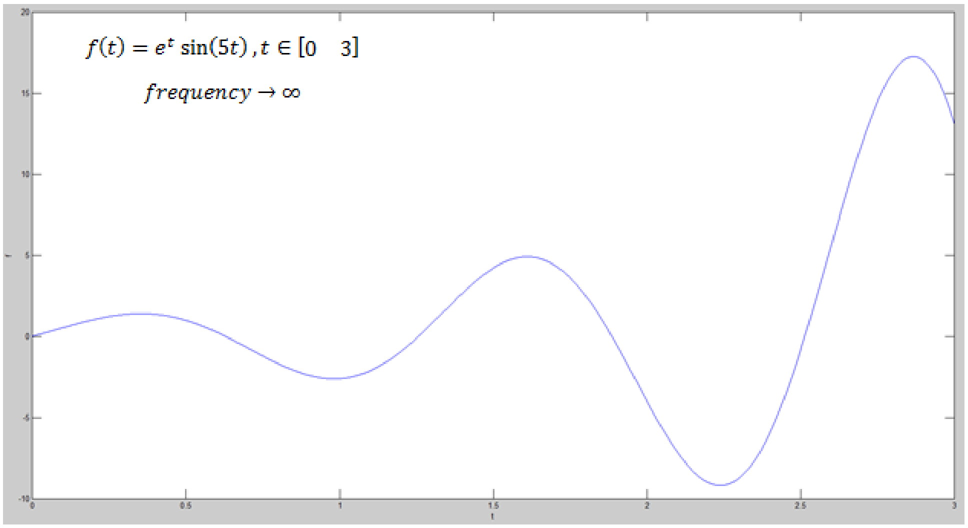


Fig 8 – Figure depicting the linearization of $f(t)$ when the frequency approaches infinity.

The discrete time simulator was implemented by locally linearizing the space state system on a $X(k)$ neibourghood. Since the studied system is composed of nonlinear equations, consider that those equations are written in the following form:

$$\begin{cases} \dot{\vec{X}} = f(\vec{X}, \vec{U}) \\ Y = h(\vec{X}, \vec{U}) \end{cases} \qquad [64]$$

A first order Taylor series can be used to linearize the nonlinear system above. The Taylor series is given by:

$$f(t) \cong \sum_{n=0}^{\infty} \frac{f^{(n)}(t^*)}{n!}(t - t^*)^n \qquad [65]$$

Where $f^{(n)}$ is the nth derivative of $f(t)$ and t^* is the local point where the function was linearized. The first order Taylor series is a reduced form of the general equation and is given by:

$$f(t) \cong \frac{f^0(t^*)}{0!}(t - t^*)^0 + \frac{f^1(t^*)}{1!}(t - t^*)^1 \qquad [66]$$

$$\therefore f(t) \cong f(t^*) + \left.\frac{df(t)}{dt}\right|_{t=t^*} (t - t^*) \qquad [67]$$

Applying the first order Taylor series transformation to the nonlinear system on a local point $X(k)$, the following is obtained:

$$\begin{cases} \dot{\vec{X}} \cong \dot{\vec{X}}(k) + A(k)\left[\vec{X} - \vec{X}(k)\right] + B(k)\left[\vec{U} - \vec{U}(k)\right] \\ \vec{Y} \cong \vec{Y}(k) + C(k)\left[\vec{X} - \vec{X}(k)\right] + D(k)\left[\vec{U} - \vec{U}(k)\right] \end{cases} \qquad [68]$$

Where F, G, H, J are Jacobian matrices given by:

$$\begin{aligned} A &= \frac{df(\vec{X},\vec{U})}{d\vec{X}} \quad B = \frac{df(\vec{X},\vec{U})}{d\vec{U}} \\ C &= \frac{dh(\vec{X},\vec{U})}{d\vec{X}} \quad D = \frac{dh(\vec{X},\vec{U})}{d\vec{U}} \end{aligned} \qquad [69]$$

Assuming the control of the system is done through the use of a digital medium, it is reasonable to assume that the actuation will be 'discrete' and, as a result, a suitable representation of such phenomena is the use of the zero order holder (ZOH). Considering that the sampling frequency is T, the new matrices A_D and B_D are given by:

$$A_D = e^{AT} \qquad [70]$$

$$B_D = \int_0^T e^{B\tau} d\tau \qquad [71]$$

The next discrete value of $\dot{\vec{X}}$ is given by:

$$\vec{X}(k+1) = A_D \vec{X}(k) + B_D\left[f\left(\vec{X}(k), \vec{U}(k)\right) - F(k)\vec{X}(k)\right] \qquad [72]$$

The next output values is simply given by:

$$\vec{Y}(k) = h\left(\vec{X}(k), \vec{U}(k)\right) \qquad [73]$$

Most parameters used in the simulator, such as inertia matrix J, the position vectors $\vec{r}_{cm\to rp}$, $\vec{r}_{cm\to lp}$ and $\vec{r}_{cm\to bp}$, etc, were obtained from a 3D model, based upon the original V-22 Osprey aircraft, created using the program *SolidWorks*. Other experimentally and heuristically obtained parameters were conjectured for the purpose of the simulation. The 3D drone model can be seen in APPENDIX 1 and some preliminary simulation results (without any controller) can be seen in APPENDIX 2.

1.5 Proposed mathematical modelling of aerodynamic forces

There are several mathematical models for the aerodynamic forces [1, 8, 9, 19, 21], yet, most of them are based of experiments and are, in essence, heuristically obtained. These mathematical model propositions tend to be relationships with a strict working boundary, one that, if violated, ensures that the model will fail.

Initially, the drone's volume is divided into control areas or planes. The planes are chosen in such a way that each normal vector generating each plane is parallel to the unit canonical vectors $\vec{\imath}, \vec{\jmath}, \vec{k}$. This means, in other words, that each unit canonical vector of the coordinate system fixed to the drone are normal vectors that generate the planes. For each plane, a control area is defined with the same area (and shape) as the drone's respective projection on that plane:

$$\begin{array}{ll} \pi_{xy}: \bar{n}_{xy} = \bar{k} & \pi_{xy}: A_{xy}\bar{n}_{xy} = A_{xy}\bar{k} \\ \pi_{yx}: \bar{n}_{yx} = -\bar{k} & \pi_{yx}: A_{yx}\bar{n}_{yx} = -A_{yx}\bar{k} \\ \pi_{yz}: \bar{n}_{yz} = \bar{\imath} & \pi_{yz}: A_{yz}\bar{n}_{yz} = A_{yz}\bar{\imath} \\ \pi_{zy}: \bar{n}_{zy} = -\bar{\imath} & \pi_{zy}: A_{zy}\bar{n}_{zy} = -A_{zy}\bar{\imath} \\ \pi_{zx}: \bar{n}_{zx} = \bar{\jmath} & \pi_{zx}: A_{zx}\bar{n}_{zx} = A_{zx}\bar{\jmath} \\ \pi_{xz}: \bar{n}_{xz} = -\bar{\jmath} & \pi_{xz}: A_{xz}\bar{n}_{zx} = -A_{xz}\bar{\jmath} \end{array} \qquad [\mathbf{74}]$$

In most aircrafts, drones, UAVs, etc, there are symmetries between opposing control areas. This means that it is possible to reduce the number of planes, as each opposing plane will have, approximately, the same drag force acting upon it.

The aerodynamic drag is divided into three parts:

1. Parasitic drag - composed as a linear combination of shape drag, surface friction and interference.

2. Lift-induced drag - a component that occurs whenever a moving object redirects the flow of air coming in its direction.

3. Wave drag - a component of aerodynamic drag on wings of airplanes and fuselage, on propeller blades and projectiles moving at supersonic transonic speeds and due to the presence of shock waves.

Wave drag can be neglected in this approach since the real V-22 Osprey aircraft (the one used to base the drone upon) cannot reach supersonic speeds because the maximum speed of this aircraft is approximately $565\ km/h$ and supersonic speed (called Mach 1) is roughly $1224\ km/h$ (which is almost double the maximum speed of the aircraft).

The drag induced by lift can be defined as the component of the lift force that opposes the direction of movement due to the vector decomposition of the lift force. For a planar wing with an elliptical distribution for the lift force, induced drag is often calculated as follows:

$$\begin{cases} \overrightarrow{drag_{lift}} = \dfrac{\overrightarrow{F_L}^2}{\frac{1}{2}\rho_0 {V_e}^2 S\pi e(AR)} \\ V_e = \sqrt{\dfrac{\rho v^2}{\rho_0}} = v\sqrt{\dfrac{\rho}{\rho_0}} \end{cases} \qquad [\mathbf{75}]$$

Where $\overrightarrow{F_L}$ is the lift force, ρ_0 is the air density at sea level under ISA conditions ($1225\ km/m^3$), V_e is the equivalent airspeed (EAS), S is the total area of the wing (given by the product of the reach of the wing and the average linear dimension called the 'chord' or mean aerodynamic chord – MAC), e is the efficiency of the wing given its geometric shape and distribution of the sustaining force (typically between 0.85 and 0.95 for an elliptic distribution of the lift force) and (AR) is a product of geometric dimensions of the wing called aspect ratio.

An important observation to make is that these equations make the induced drag depend on the square of the lift force. They also take into account the (AR) and surface area. This is only an approximation and is not valid for high angles of attack and for very high (AR) values. High angles of attack, nonetheless, tend not to be a problem as a high angle of attack would compromise the flight stability of drone regardless.

The parasitic drag is composed of three drag subtypes: the surface drag (induced when a fluid drags on a non-ideal surface), the drag shape (induced when the shape of an object strikes and redirects a fluid) and the interference drag. This approach will consider that the interference drag is a milder manifestation of the wave drag and, therefore, only assumes expressive values when the velocity reaches values close to Mach 1 and, as a result, the net parasitic drag is:

$$\begin{gathered} drag_{parasitic} = drag_{surface} + drag_{shape} + drag_{interference} \\ drag_{surface} + drag_{shape} \gg drag_{interference} \\ \therefore drag_{parasitic} \approx drag_{surface} + drag_{shape} \end{gathered} \quad [\mathbf{76}]$$

The remaining forms of drag follow the general drag equation but have different drag coefficients for each drag type:

$$\begin{cases} drag_{shape} = \dfrac{1}{2}\rho V^2 A_s C_{shape} \\ C_{shape} \rightarrow depends\ on\ the\ shape \end{cases} \quad [\mathbf{77}]$$

$$\begin{cases} drag_{surface} = \frac{1}{2}\rho V^2 A_s C_{surface} \\ C_{surface} \approx \frac{0.074}{Re^{0.2}} \end{cases} \quad [\mathbf{78}]$$

The net parasitic drag is given as a weighted linear combination of the different drag subtypes. These weighting coefficients vary with the relationship between the velocity vector and the control surfaces. Consider, for example, a square with almost null length. If it is parallel to the velocity vector, it will only have surface drag but, on the other hand, if it is perpendicular to the velocity vector it will only have shape drag. If it is not perpendicular or parallel to the velocity vector it will have the two forms of drag. The total drag is defined as:

$$drag_{parasitic} = C_1 drag_{shape} + C_2 drag_{surface} + C_3 drag_{interference} \quad [\mathbf{79}]$$

$$C_1 = C_1(\vec{V}, A, \vec{\Xi}), C_2 = C_2(\vec{V}, A, \vec{\Xi}), C_3 \cong 0 \quad [\mathbf{80}]$$

It is necessary to decompose the velocity into components parallel to each normal vector of each control area, in order to use them. This can be done by utilizing the orthogonal projection and by defining the coefficients C_1 and C_2 as the result of these projections ($C_1 \in \mathbb{R}^3, C_2 \in \mathbb{R}^3$). The orthogonal projection is given by:

$$Proj_{\vec{v}}\vec{u} = \left(\frac{\vec{v}\cdot\vec{u}}{|\vec{v}|^2}\right)\vec{v} \quad [\mathbf{81}]$$

Regardless of the choice of control area, the other control areas will always be orthogonal to the chosen area because the unit canonic vectors that generate each control area are orthogonal, thus:

$$\overrightarrow{C_1} = \begin{bmatrix} Proj_{\vec{\imath}}\vec{V} \\ Proj_{\vec{\jmath}}\vec{V} \\ Proj_{\vec{k}}\vec{V} \end{bmatrix} = \begin{bmatrix} (\vec{\imath}\cdot\vec{V})\vec{\imath} \\ (\vec{\jmath}\cdot\vec{V})\vec{\jmath} \\ (\vec{k}\cdot\vec{V})\vec{k} \end{bmatrix} \quad [\mathbf{82}]$$

$$\overrightarrow{C_2} = \begin{bmatrix} Proj_{\vec{\jmath}}\vec{V} + Proj_{\vec{k}}\vec{V} \\ Proj_{\vec{\imath}}\vec{V} + Proj_{\vec{k}}\vec{V} \\ Proj_{\vec{\imath}}\vec{V} + Proj_{\vec{\jmath}}\vec{V} \end{bmatrix} = \begin{bmatrix} (\vec{\jmath}\cdot\vec{V})\vec{\jmath} + (\vec{k}\cdot\vec{V})\vec{k} \\ (\vec{\imath}\cdot\vec{V})\vec{\imath} + (\vec{k}\cdot\vec{V})\vec{k} \\ (\vec{\imath}\cdot\vec{V})\vec{\imath} + (\vec{\jmath}\cdot\vec{V})\vec{\jmath} \end{bmatrix} \quad [\mathbf{83}]$$

The total net drag $\overrightarrow{F_D}$ is simply the sum of both types of drags:

$$\overrightarrow{F_D} = \overrightarrow{drag_{lift}} + \overrightarrow{drag_{parasitic}} \quad [\mathbf{84}]$$

$$\overrightarrow{F_D} = \overrightarrow{drag_{lift}} + \vec{C}_1 drag_{shape} + \vec{C}_2 drag_{surface} \quad [\mathbf{85}]$$

$$\overrightarrow{F_D} = \overrightarrow{drag_{lift}} + \begin{bmatrix} (\bar{\imath}\cdot\bar{V})\bar{\imath} \\ (\bar{\jmath}\cdot\bar{V})\bar{\jmath} \\ (\bar{k}\cdot\bar{V})\bar{k} \end{bmatrix} drag_{shape} + \begin{bmatrix} (\bar{\jmath}\cdot\bar{V})\bar{\jmath} + (\bar{k}\cdot\bar{V})\bar{k} \\ (\bar{\imath}\cdot\bar{V})\bar{\imath} + (\bar{k}\cdot\bar{V})\bar{k} \\ (\bar{\imath}\cdot\bar{V})\bar{\imath} + (\bar{\jmath}\cdot\bar{V})\bar{\jmath} \end{bmatrix} drag_{surface} \quad [\mathbf{86}]$$

2. POSSIBLE IMPLEMENTATIONS OF SOME CONTROLLERS

This section will discuss the implementations and results of some controllers, stating their overall effective and limitations.

2.1 PID controller

The PID controller is one of the oldest and most widely used controller in industrial applications [17]. Its success and overall acceptability stems from its simplicity in terms of implementation and effectiveness, as well as its low cost (in terms of both purchase and installation). It has been proven to be robust and adequate in various different types of control applications and systems, failing only in very niche applications, generally involving extremely complex, typically nonlinear, systems.

The PID controller has three terms or constants, which give its namesake. These are the proportional, integral and derivative constants (hence "PID") (Figure 9). The controller seeks to minimize an error function $e(t)$, which is, commonly, the difference between the current states (obtained through a sensor or group of sensors) and the desired states. It then proceeds to minimize error over time, allowing by adjusting the inputs $\vec{U}(t)$. Mathematically, it is a weighted sum where each coefficient is positive:

$$\begin{cases} \vec{U}(t) = K_p e(t) + K_i \int_0^t e(\tau)d\tau + K_d \frac{de(t)}{dt} \\ e(t) = \vec{X}_d - \vec{Y} \end{cases} \qquad [\mathbf{87}]$$

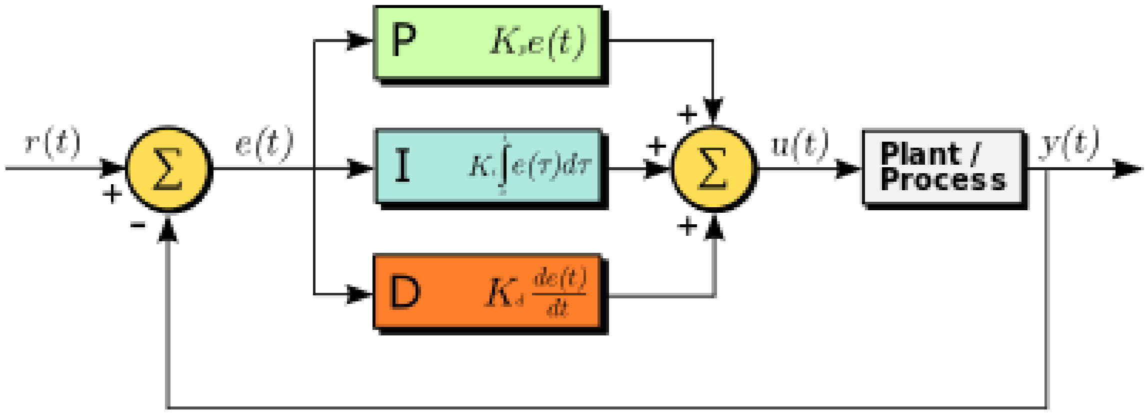


Fig. 9 – PID controller diagram

Where $\vec{X}_d$ is the desired value for the states and $\vec{Y}$ is the sensor readings of the states. The proportional coefficient K_p acts on the present error. Increasing it results in a reduction of the time constant of an output, its error in steady state and the output's general stability. Its increase also increases the amount of overshoot and does not affect the stabilization time. The integral coefficient K_i acts on the past error. Increasing it has similar results to increasing K_p in term of what is reduced (time constant reduction, steady state error reduction and stabilization reduction).

Increasing K_i, however, increases the amount of overshoot and stabilization time. Finally, the derivative coefficient K_d acts on the future error. Increasing it has no effects on the output's time constant or error in steady state; nevertheless, it reduces overshoot and stabilization time while increasing the overall stability.

These gains (or coefficients or constants) are obtained heuristically and experimentally. There are several techniques to provide an initial estimate for these gains including Ziegler-Nichols, Cohen-Coon, Tyreus Luyben, etc, all with its advantages and disadvantages. In practice, these techniques are used once (if at all) and the rest of the calibration is done manually in an experimental manner.

The PID controllers, however, may fail if the system parameters cannot be precisely estimated or obtained and the resulting designed PID gains may not resist certain uncertainties and disturbances. Even though the PID gains can be well designed, this controller still is not as robust as other robust controllers are and often fail when the system is too complex (and nonlinear) or encounters multiple challenges from the operating environment of the system (XIN, Ming). PID controllers often are used in SISO systems because its implementation becomes significantly harder to use in MIMO systems, and its results tend to be underwhelming.

The system presented in this work is nonlinear, time-variant and non-affine in inputs, therefore, conjecturing values for the PID gains at every control period would not only be unfeasible, but also unrealistic. The system also has varying poles and zeros which may or may not be unstable for certain cases and this stops the successful control of the system by the PID controller.

2.2 State-space feedback controller

A State-space feedback controller [17] was implemented using MATLAB and its 'place' function. This function implements a closed-loop pole assignment using state feedback and, to that effect, *"computes a state-feedback matrix K such that the eigenvalues of $A - BK$ are those specified in vector P"* (MATLAB 'place' function help description) (figure 10). The function, however, failed to compute such a matrix and listed as reasons a probable proximity to a singularity or because of the lack of (or extremely reduced) controllability of the system.

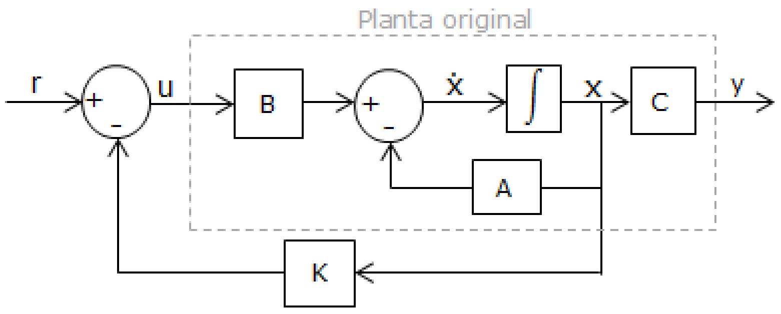


Fig. 10 - State-space feedback controller diagram

Verifying the rank of the controllability matrix, it is clear that the system has a reduced controllability, with the result being a rank of 4 on a 12 dimension matrix. The controller was able to find the gain matrix K for a reduced order system obtained through pole-zeros cancellation and state dynamic elimination; however, this reduced order system does not represent reality in terms of the physical relationships regarding the drone's flight and is, mainly, a mathematical construct.

2.3 Linear Quadratic Regulator controller (LQR)

In optimal control theory, the linear quadratic regulator (LQR) control was developed with the intent of finding the minimum cost of an operating dynamical system. The LQR controller attempts to minimize a cost function through means of a mathematical algorithm, taking into account weighting factors supplied by the user. The LQR controller, analogue to the PID controller, uses a state cost matrix $Q_1$and a controller cost matrix Q_2 which act similar to the PID gains and are obtained heuristically. Both Q_1 and Q_2 matrices are positively semi-defined. The LQR controller then attempts to minimize a linear quadratic cost function given by the following expression:

$$J(\vec{U}) = \frac{1}{2}\int_0^t \left(\vec{X}^T Q_1 \vec{X} + \vec{U}^T Q_2 \vec{U} + 2\vec{X}^T N \vec{U}\right) dt \qquad [\mathbf{88}]$$

Subject to the system dynamics:

$$\dot{\vec{X}} = A\vec{X} + B\vec{U} \qquad [\mathbf{89}]$$

Where N is a positively defined matrix. Typically, this function has an infinite horizon $t \rightarrow \infty$ but it is not mandatory and, in practice, the matrix N is chosen to be the same as the identity matrix I. The optimal gain matrix K is given by:

$$K = {Q_2}^{-1}(B^T S + N^T) \quad [90]$$

Where S is the solution of the Riccati equation given by:

$$A^T S + SA - (SB + N){Q_2}^{-1}(B^T S + N^T) + Q_1 = 0 \quad [91]$$

The control law implemented by this controller is simply an optimized proportional controller on a closed loop state feedback and is given by the equation:

$$\vec{U} = -K\vec{X} \quad [92]$$

A LQR controller was implemented using MATLAB and its ‘dlqr’ function. As expected, since the control law is fundamentally an optimization of a closed loop state feedback proportional control, it failed to find the gain matrix K and listed as reasons a probable proximity to a singularity or because of the lack of (or extremely reduced) controllability of the system. One of the limitations of this technique is that the matrix pair A and B must be stabilizable which could be the reason why it failed.

The extremely reduced controllability is also another strong possibility as to why the controller failed to stabilize the plant. Much like the PID controller, it was able to find the gain matrix K for a reduced order system, nevertheless, since the physical meaning of the system is lost when its order is reduced mathematically, it is of little relevance in terms of applied control.

2.4 Discussion of hover control through feedback loop linearization

A possible nonlinear control technique to control the hover movement is the feedback loop linearization technique [4, 5, 6], in which the nonlinear components are cancelled (or have their effects extremely diminished) through the feedback loop of an adequately chosen control law. An ideal control law would create a resulting system with a proportional control. Considering the studied system:

$$\dot{\vec{V}} = F_1(\vec{X}) + h_2(\vec{U}) \quad [93]$$

$$\dot{\vec{\Omega}} = f_3(\vec{X}) + h_4(\vec{U}) \quad [94]$$

The first derivative of the states with respect to time would be given by the following mathematical expression:

$$\dot{\vec{X}} = \begin{bmatrix} \int \dot{\vec{V}} \\ \dot{\vec{V}} \\ \int \dot{\vec{\Omega}} \\ \dot{\vec{\Omega}} \end{bmatrix} \quad [95]$$

Let $\vec{U}^*$ be an ideal control law that would result in:

$$\dot{\vec{V}} = F_1(\vec{X}) + h_2(\vec{U}^*) \cong -K\vec{X} + F_1(\vec{X}) \cong -K_1\vec{X} + w_v \quad [96]$$

$$\dot{\vec{\Omega}} = f_3(\vec{X}) + h_4(\vec{U}^*) \cong -K\vec{X} + f_3(\vec{X}) \cong -K_2\vec{X} + w_\Omega \quad [97]$$

Where w_v and w_Ω are nonlinear noises and disturbances acting upon the system. If possible, the noises and disturbances should be such that:

$$\left|-K_1\vec{X}\right| \gg w_v \qquad [98]$$

$$\left|-K_2\vec{X}\right| \gg w_\Omega \qquad [99]$$

Considering equations [46] and [58]:

$$h_2(\vec{U}) = \frac{\vec{F}_{prop}(\vec{U})}{m} \qquad [100]$$

$$h_4(\vec{U}) = J^{-1}\left[\vec{M}_{prop}(\vec{U}) + \vec{M}_{gyro}(\vec{U})\right] \qquad [101]$$

Expanding both of the equations above:

$$h_2(\vec{U}) = \frac{1}{m}\begin{bmatrix} s_{\delta_{lp}} & s_{\delta_{rp}} & 0 \\ 0 & 0 & 0 \\ -c_{\delta_{lp}} & -c_{\delta_{rp}} & -1 \end{bmatrix}\begin{bmatrix} sgn(\omega_{lp})k_{lp}{\omega_{lp}}^2 \\ sgn(\omega_{rp})k_{rp}{\omega_{rp}}^2 \\ sgn(\omega_{bp})k_{bp}{\omega_{bp}}^2 \end{bmatrix} \qquad [102]$$

$$h_4(\vec{U}) = J^{-1}\left[\begin{bmatrix} J_{xx}c_{\delta_{lp}} + J_{xz}s_{\delta_{lp}} & J_{xx}c_{\delta_{rp}} + J_{xz}s_{\delta_{rp}} & 0 \\ J_{yx}c_{\delta_{lp}} + J_{yz}s_{\delta_{lp}} & J_{yx}c_{\delta_{rp}} + J_{yz}s_{\delta_{rp}} & 0 \\ J_{zx}c_{\delta_{lp}} + J_{zz}s_{\delta_{lp}} & J_{zx}c_{\delta_{rp}} + J_{zz}s_{\delta_{rp}} & 0 \end{bmatrix}\begin{bmatrix} sgn(\omega_{lp})k_{lp}{\omega_{lp}}^2 \\ sgn(\omega_{rp})k_{rp}{\omega_{rp}}^2 \\ sgn(\omega_{bp})k_{bp}{\omega_{bp}}^2 \end{bmatrix}\right] + \qquad [103]$$

$$J^{-1}\left[\begin{bmatrix} -y_{cm\to lp}c_{\delta_{lp}} & -y_{cm\to rp}c_{\delta_{rp}} & y_{cm\to bp} \\ z_{cm\to lp}s_{\delta_{lp}} + x_{cm\to lp}c_{\delta_{lp}} & z_{cm\to rp}s_{\delta_{rp}} + x_{cm\to rp}c_{\delta_{rp}} & -x_{cm\to bp} \\ -y_{cm\to lp}s_{\delta_{lp}} & -y_{cm\to rp}s_{\delta_{rp}} & 0 \end{bmatrix}\begin{bmatrix} sgn(\omega_{lp})k_{lp}{\omega_{lp}}^2 \\ sgn(\omega_{rp})k_{rp}{\omega_{rp}}^2 \\ sgn(\omega_{bp})k_{bp}{\omega_{bp}}^2 \end{bmatrix}\right]$$

It is assumed for the hover movement that the drone has both propellers in vertical position, that is, with zero tilt. These assumptions reduce the equations above to:

$$\begin{cases} \delta_{lp} = \delta_{rp} = 0 \\ \cos(0) = 1 \\ \sin(0) = 0 \end{cases} \qquad [104]$$

$$h_2(\vec{U}) = \frac{1}{m}\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}\begin{bmatrix} sgn(\omega_{lp})k_{lp}{\omega_{lp}}^2 \\ sgn(\omega_{rp})k_{rp}{\omega_{rp}}^2 \\ sgn(\omega_{bp})k_{bp}{\omega_{bp}}^2 \end{bmatrix} \qquad [105]$$

$$h_4(\vec{U}) = J^{-1}\left[\begin{bmatrix} J_{xx} & J_{xx} & 0 \\ J_{yx} & J_{yx} & 0 \\ J_{zx} & J_{zx} & 0 \end{bmatrix}\begin{bmatrix} sgn(\omega_{lp})k_{lp}{\omega_{lp}}^2 \\ sgn(\omega_{rp})k_{rp}{\omega_{rp}}^2 \\ sgn(\omega_{bp})k_{bp}{\omega_{bp}}^2 \end{bmatrix}\right] + \qquad [106]$$

$$J^{-1}\left[\begin{bmatrix} -y_{cm\to lp} & -y_{cm\to rp} & y_{cm\to bp} \\ x_{cm\to lp} & x_{cm\to rp} & -x_{cm\to bp} \\ 0 & 0 & 0 \end{bmatrix}\begin{bmatrix} sgn(\omega_{lp})k_{lp}{\omega_{lp}}^2 \\ sgn(\omega_{rp})k_{rp}{\omega_{rp}}^2 \\ sgn(\omega_{bp})k_{bp}{\omega_{bp}}^2 \end{bmatrix}\right]$$

Choosing as an initial design for the control law:

$$\vec{U} = \begin{bmatrix} \sqrt{\omega_{lp}} \\ \sqrt{\omega_{rp}} \\ \sqrt{\omega_{bp}} \\ 0 \\ 0 \end{bmatrix} - K\vec{X} \qquad [\mathbf{107}]$$

$$h_2(\vec{U}) = \frac{1}{m}\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}\begin{bmatrix} sgn(\omega_{lp})k_{lp}\omega_{lp} \\ sgn(\omega_{rp})k_{rp}\omega_{rp} \\ sgn(\omega_{bp})k_{bp}\omega_{bp} \end{bmatrix} \qquad [\mathbf{108}]$$

$$h_4(\vec{U}) = J^{-1}\left[\begin{bmatrix} J_{xx} & J_{xx} & 0 \\ J_{yx} & J_{yx} & 0 \\ J_{zx} & J_{zx} & 0 \end{bmatrix}\begin{bmatrix} sgn(\omega_{lp})k_{lp}\omega_{lp} \\ sgn(\omega_{rp})k_{rp}\omega_{rp} \\ sgn(\omega_{bp})k_{bp}\omega_{bp} \end{bmatrix}\right] + J^{-1}\left[\begin{bmatrix} -y_{cm\to lp} & -y_{cm\to rp} & y_{cm\to bp} \\ x_{cm\to lp} & x_{cm\to rp} & -x_{cm\to bp} \\ 0 & 0 & 0 \end{bmatrix}\begin{bmatrix} sgn(\omega_{lp})k_{lp}\omega_{lp} \\ sgn(\omega_{rp})k_{rp}\omega_{rp} \\ sgn(\omega_{bp})k_{bp}\omega_{bp} \end{bmatrix}\right] \qquad [\mathbf{109}]$$

This suggestion for a possible control law still failed to stabilize the system since MATLAB's function 'place' and 'dlqr' where unable to find suitable gain matrices K. This suggests that a more robust and complex control technique is required. An alternative possibility to explore is the further use of linearization, including the linearization of the input. By applying the Taylor series to the definition of the propeller force, it is possible to transform it into a piece wise equation, such that, given a certain boundary, it is linear within that boundary:

$$\begin{cases} F_{propeller}(\vec{\omega}_{propeller})\big|_{\vec{\omega}^*} \cong \alpha + \beta(\vec{\omega}_{propeller} - \vec{\omega}^*) \\ \alpha = k_{propeller}\vec{\omega}^{*2} \\ \beta = 2k_{propeller}\vec{\omega}^* \end{cases} \qquad [\mathbf{110}]$$

The aforementioned alternative does not solve, however, the nonlinearities present the tilting of the propellers and the coupling of the states. An alternative possible solution is to implement a sliding mode output-feedback linearization as proposed by Giorgio Bartolini and Elisabetta Punta on their article "Sliding mode output-feedback stabilization of uncertain nonlinear nonaffine systems" [6]. The implementation is complex and the article does not cover certain important aspects such as strategies to define an adequate sliding manifold.

3. CONCLUSIONS AND FUTURE IMPROVEMENTS

The objective of this final course work was achieved, as a realistic, modular, mathematical model was created for the tilt rotor drone. Difficulties were encountered when attempting to control the system, however, due to is complex, nonlinear, non-affine in input nature. Classical techniques such as the PID and state-space feedback controller failed, as well as modern control techniques such as LQR controller and the feedback loop stabilization technique, although promising, needs to be further refined so that control of model can be achieved. A discrete time, code line simulator and a continuous time, Simulink simulator were created successfully to demonstrate the system's behavior.

A mathematical approach to the modelling of the aerodynamic forces was proposed. It consisted of control surfaces to map the behavior of the different forms of drag, offering an alternative 'exact' model as opposed to the commonly found experimental or heuristic models present in literature. It can be easily implemented due to the model's modular form, allowing the user to choose to incorporate it into their model or neglect its effects altogether.

The next suggested step would be the application and implementation of robust control laws and techniques such as adaptive control [11, 14], feedback stabilization [13], sliding mode control [5, 6, 11, 14], extremum seeking control [15], etc, with the objective of controlling the system without further simplifications and specific assumptions.

It is clear that this model requires sophisticated control techniques, many of which are new to field of control and automation. The lack of adequate mathematical models for this aircraft concept corroborates to the idea that this aircraft is still a challenge to control, requiring innovations and discoveries of new control methods. It is a complex system that poses a challenge from its dynamic model to the control of its behavior.

4. REFERENCES

[1] ABBOTT, Ira H; VON DOENHOFF, Albert E. “Theory of Wing Sections” 1st edition (1959)
[2] ANDERSON JR, John. “Introduction to Flight” 7th edition (2011)
[3] BARKAI, S. M.; RAND, O.; PEYRAN; R. J.; CARLSON, R. M. “Modeling and analysis of tilt-rotor aeromechanical phenomena” (1997)
[4] BARTOLINI, G; PYDYNOWSKI, P. “Approximate linearization of uncertain nonlinear systems by means of continuous control” (1991)
[5] BARTOLINI, Giorgio; PUNTA, Elisabetta. “Multi-input sliding mode control of nonlinear uncertain non-affine systems with mono-directional actuation” (2015)
[6] BARTOLINI, Giorgio; PUNTA, Elisabetta. “Sliding mode output-feedback stabilization of uncertain nonlinear non-affine systems” (2012)
[7] ÇAKICI, Ferit. “Modelling, stability analysis and control system design of a small-sized tiltrotor UAV” (2009)
[8] CLANCY, L.J. “Aerodynamics” 1st edition (1978)
[9] DEMASI, Luciano; DIPACE, Antonio; MONEGATO, Giovanni; CAVALLARO, Rauno. “An invariant formulation for the minimum induced drag conditions of non-planar wing systems” (2014)
[10] HELIMART. “How helicopters have helped better the world”. Available in: http://helimart.com/helicopter-resources/history-in-helicopters.html. Accessed in: 11 April. (2017)
[11] IOANNOU, Petrus; SUN, Jing. “Robust adaptive control”, 1st edition (1996)
[12] KENDOUL, Farid; FANTONI, Isabelle; LOZANO, Rogelio. “Modeling and control of a small autonomous aircraft having two tilting rotors”, (2005)
[13] KHALIL, Hassan K. “Nonlinear control systems”, 1st edition (2000)
[14] KRSTIĆ, Miroslav. “Nonlinear and adaptive control design”, 1st edition (1995)
[15] KRSTIĆ, Miroslav. “Real-time optimization by extremum–seeking control”, 1st edition (2003)
[16] MEGGIOLARO, Marco Antonio. “Controle discreto do acrobot”, (2014)
[17] OGATA, Katsuhiko. “Modern Control Engineering”, 1st edition (1970)
[18] OLFATI-SABER, Reza. “Nonlinear control of underactuated mechanical systems with application to robotics and aerospace vehicles” (2001)
[19] ÖNER, Kaan Taha; ÇETİNSOY, Ertuğrul; SIRIMOĞLU, Efe; HANÇER, Cevdet; ÜNEL, Mutafa; AKŞİT, Mahmut Faruk; GÜLEZ, Kayhan; KANDEMİR, İlyas. “Mathematical modelling and vertical flight control of a tilt-wing UAV” (2010)
[20] STROGATZ, Steven. “The End of Insight” article obtained from “Edge” website, available at <https://www.edge.org/response-detail/11385> accessed at 18 June (2017)
[21] WHITE, Frank M. “Fluid Mechanics” 5th edition (2003)

FIGURES:

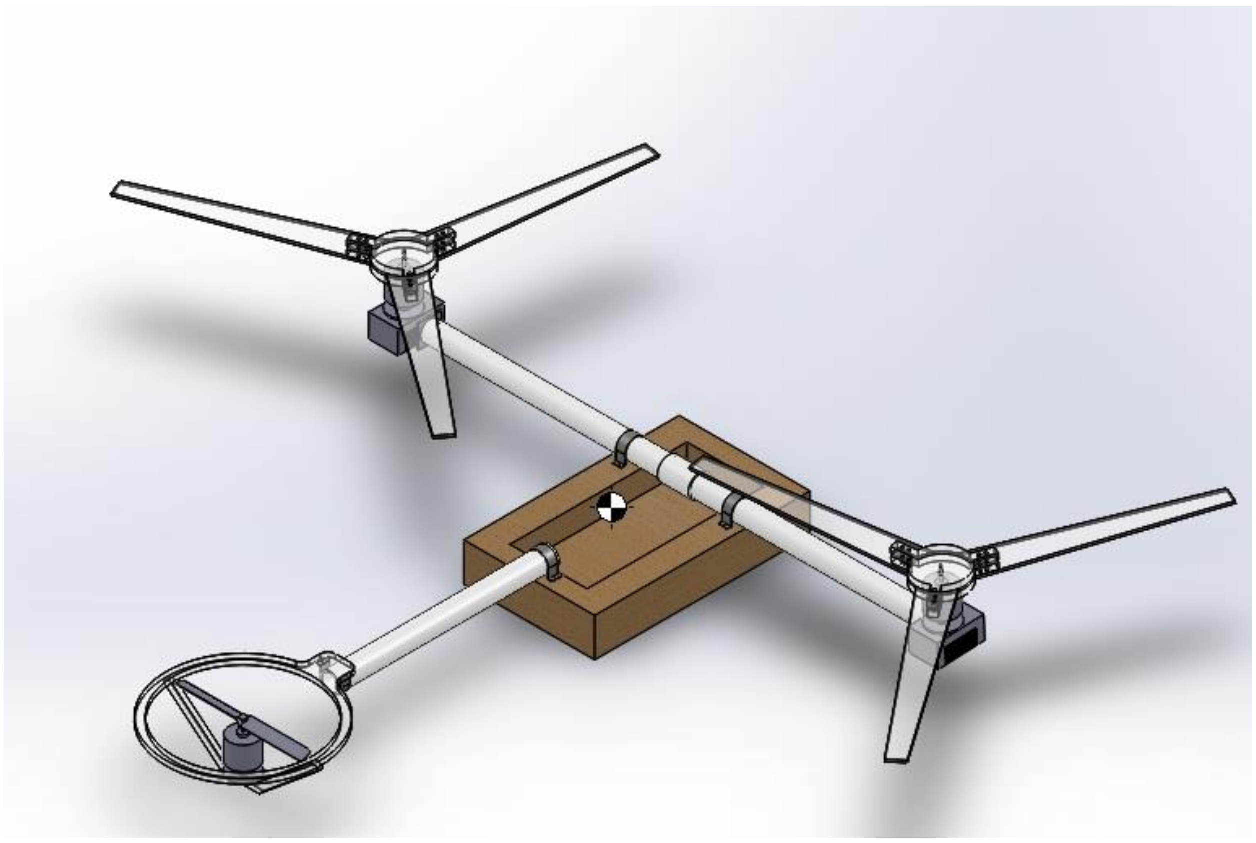
Fig 1 – Obtained from materials provided by supervisor William de Souza Barbosa at 10 May (2017).

Fig 3 – Obtained and edited from “Scale Modelling Now” website, available at <https://www.scalemodellingnow.com/hnaircraftkits-revell-mv22-osprey> and accessed at 10 May (2017).
Fig 9 – Obtained from “PID Controller” website, available at <https://en.wikipedia.org/wiki/PID_controller > and accessed at 10 May (2017).

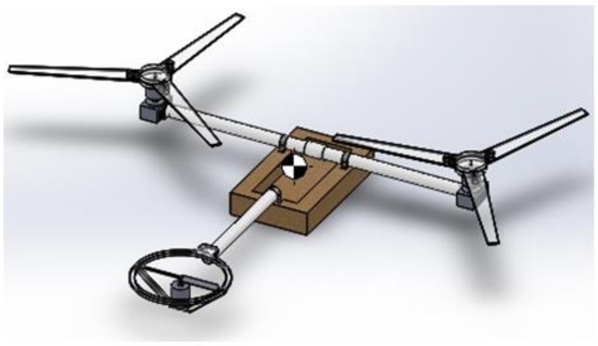
Fig 10 – Obtained from “Maxwell – Simulações em engenharia elétrica – Motor DC” website, available at < https://www.maxwell.vrac.puc-rio.br/25077/25077.PHP> and accessed at 18 June (2017).

5. APPENDICES

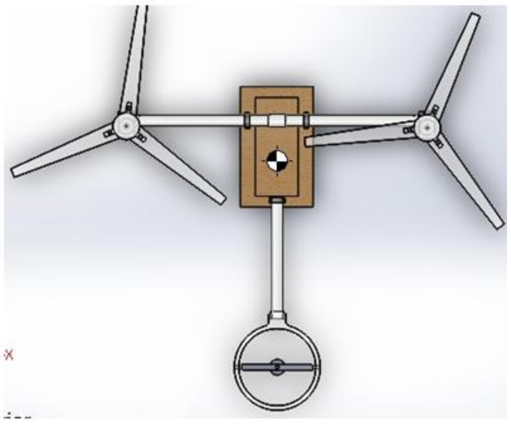
5.1 Appendix 1 - 3D MODEL OF THE TILT ROTOR DRONE



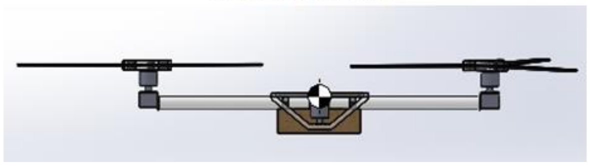
Isometric view (XYZ)



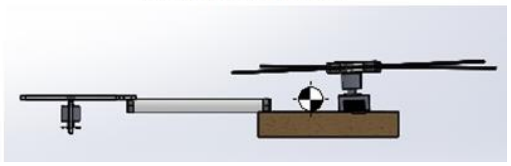
Superior view $(XY\ plane)$



Back view $(YZ\ plane)$

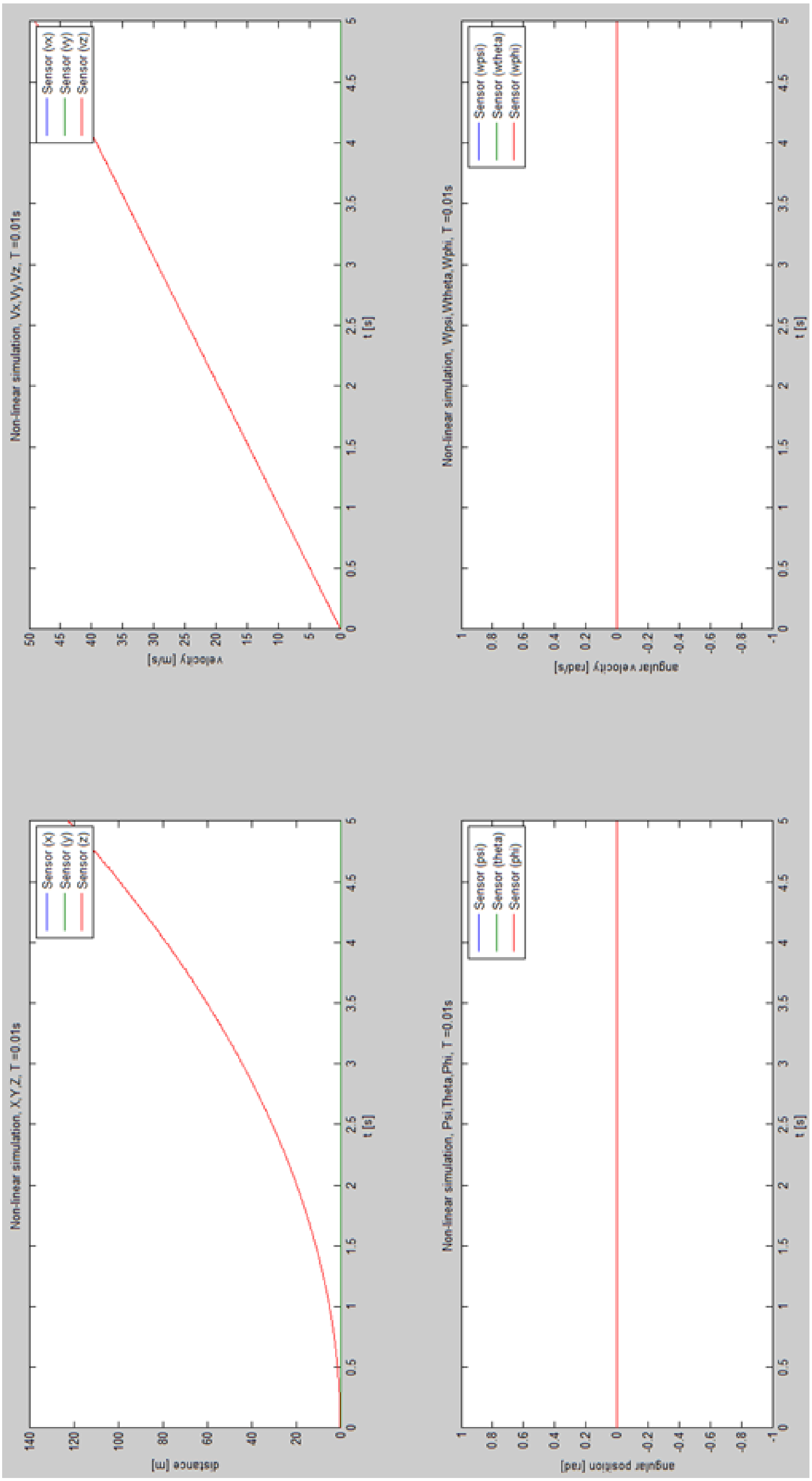


Lateral view $(XZ\ plane)$

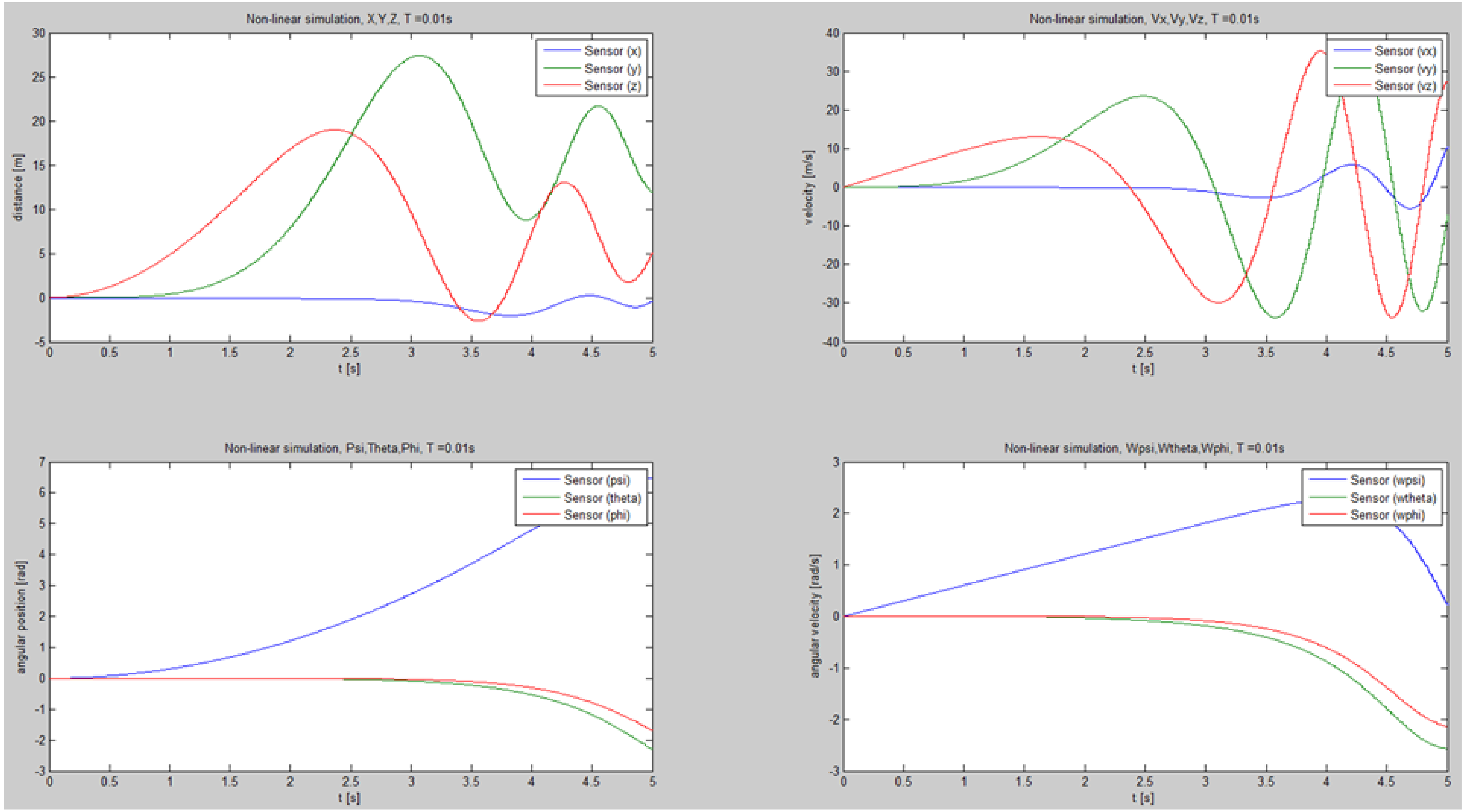


5.2 Appendix 2 - SIMULATION RESULTS WITHOUT CONTROL

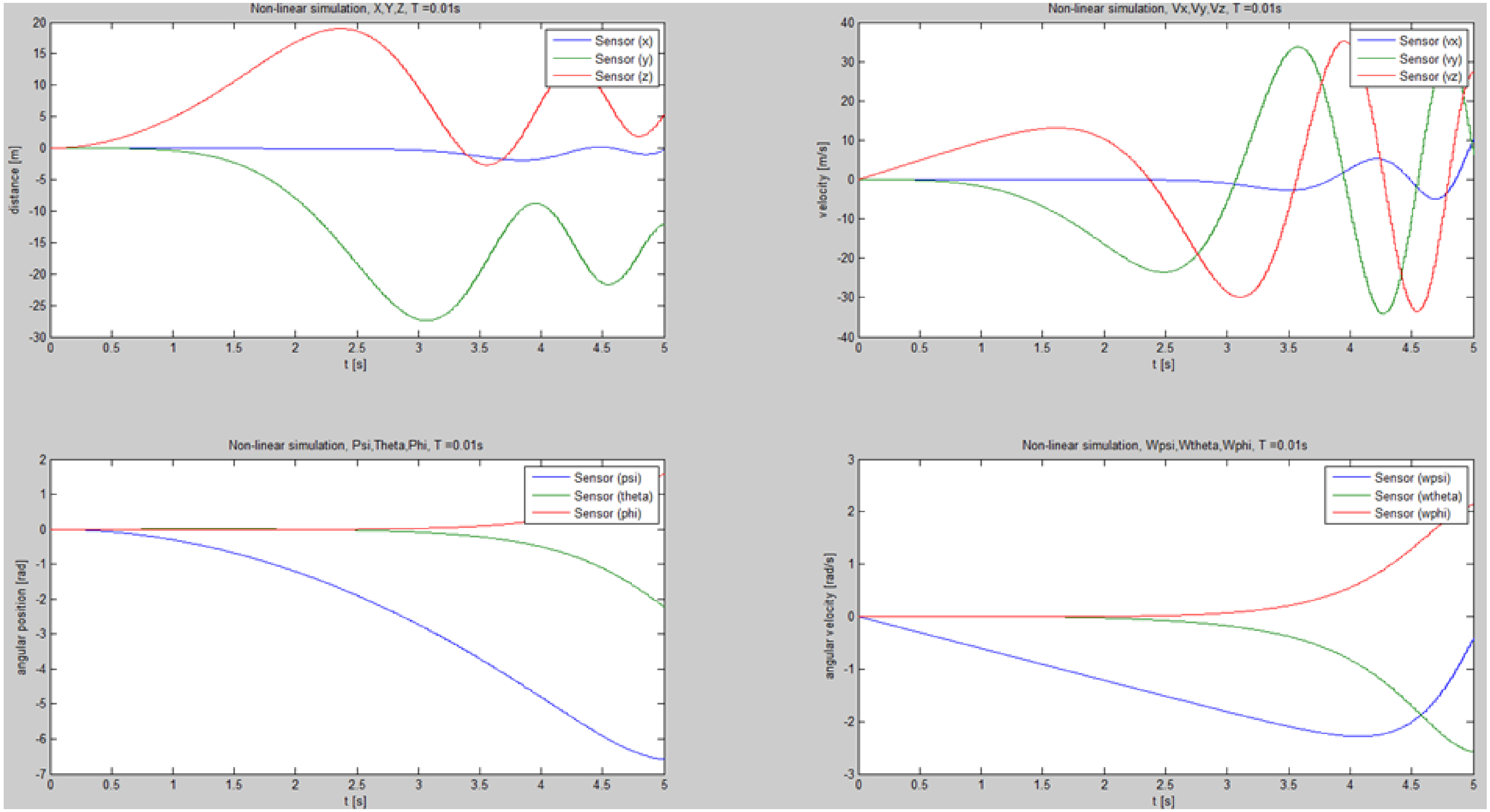
Nonlinear simulation with sampling period $T = 0.01\ [s]$, $\vec{U} = \vec{0}$ (free falling motion) and a $5\ [s]$ simulation time.



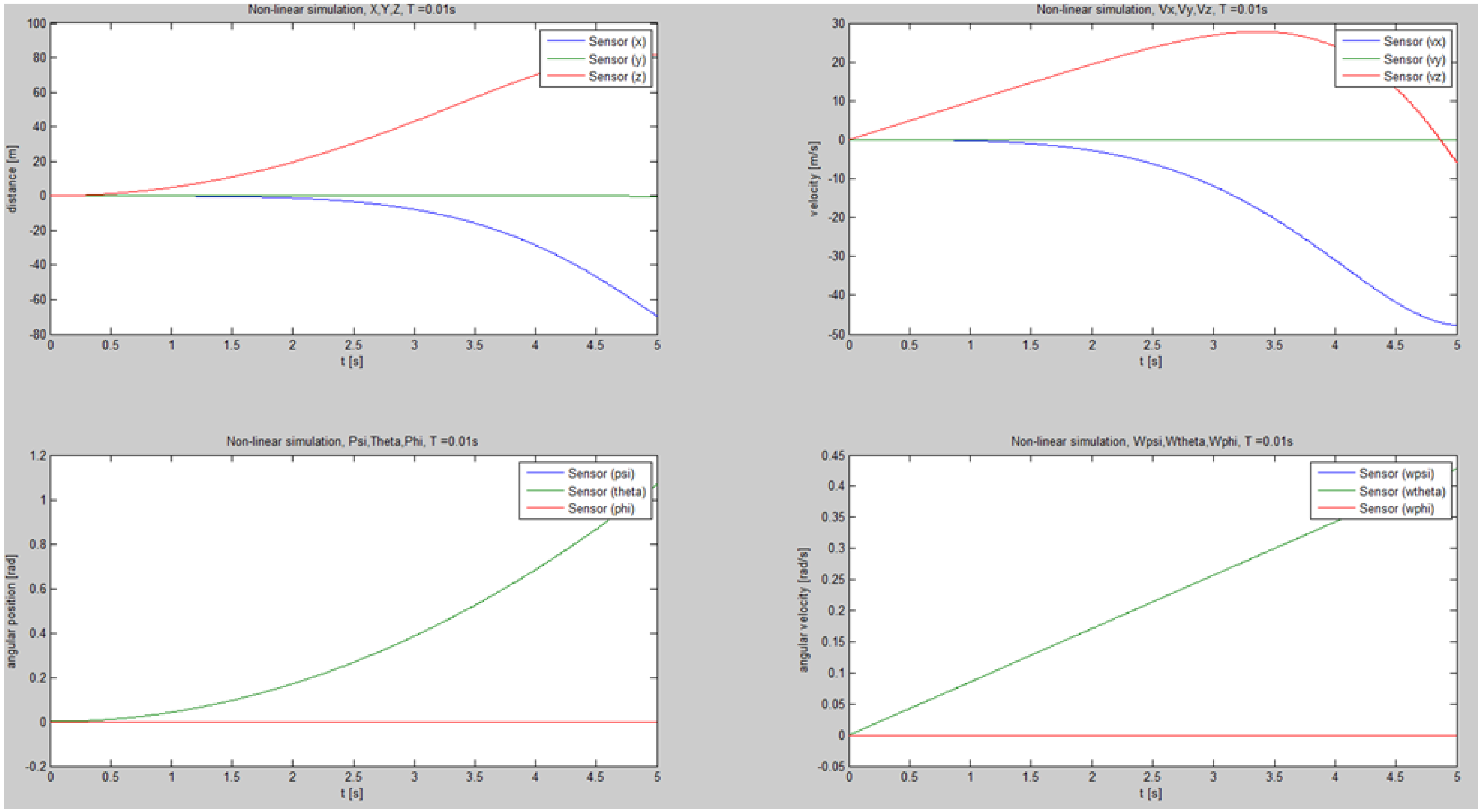
Nonlinear simulation with sampling period $T = 0.01\ [s]$, $\vec{U}^T = [0.1 \quad 0 \quad 0 \quad 0 \quad 0]$ (step input) and a 5 $[s]$ simulation time.



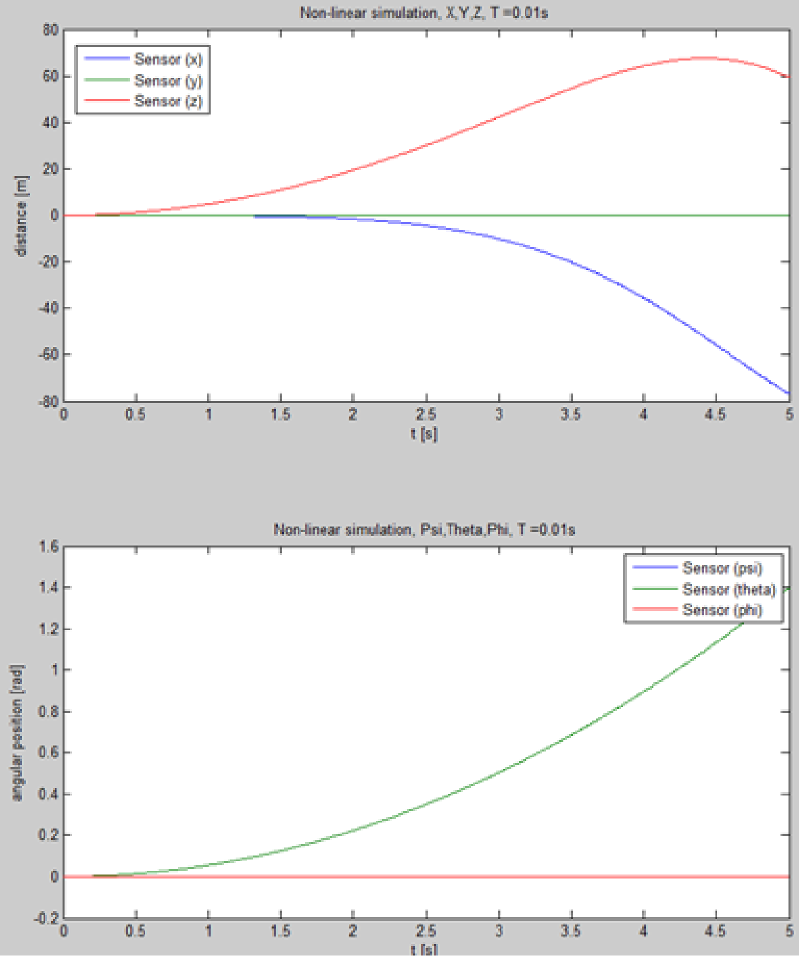
Nonlinear simulation with sampling period $T = 0.01\ [s]$, $\vec{U}^T = [0 \quad -0.1 \quad 0 \quad 0 \quad 0]$ (step input) and a 5 $[s]$ simulation time.

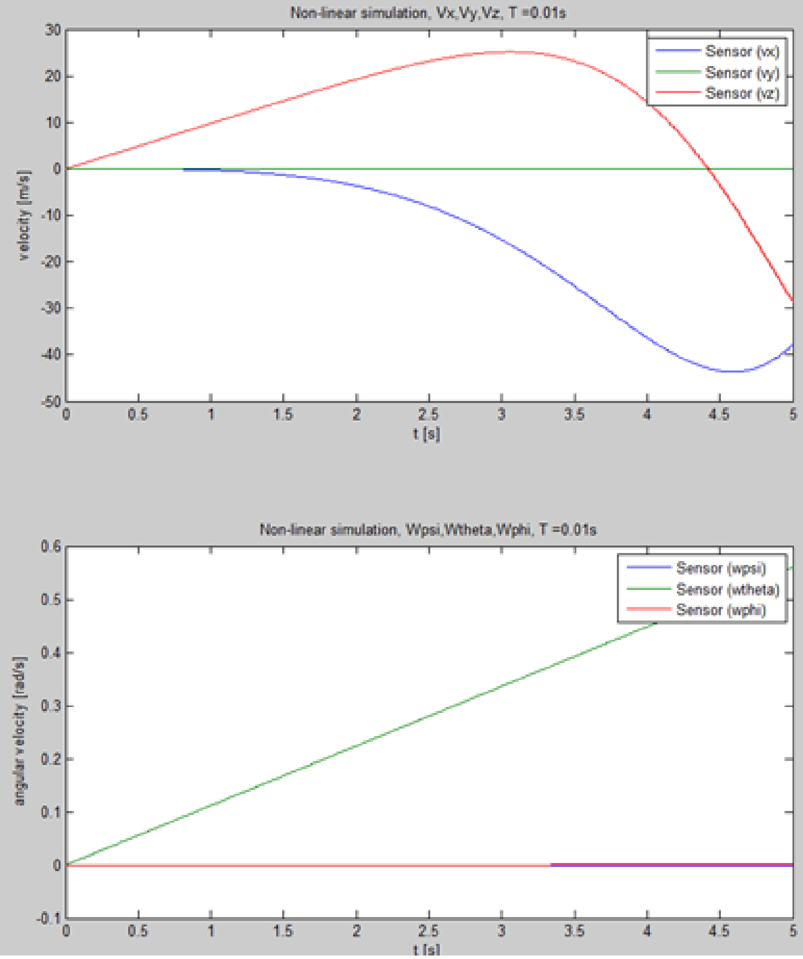


Nonlinear simulation with sampling period $T = 0.01\ [s]$, $\vec{U}^T = [0 \quad 0 \quad 0.1 \quad 0 \quad 0]$ (step input) and a $5\ [s]$ simulation time.

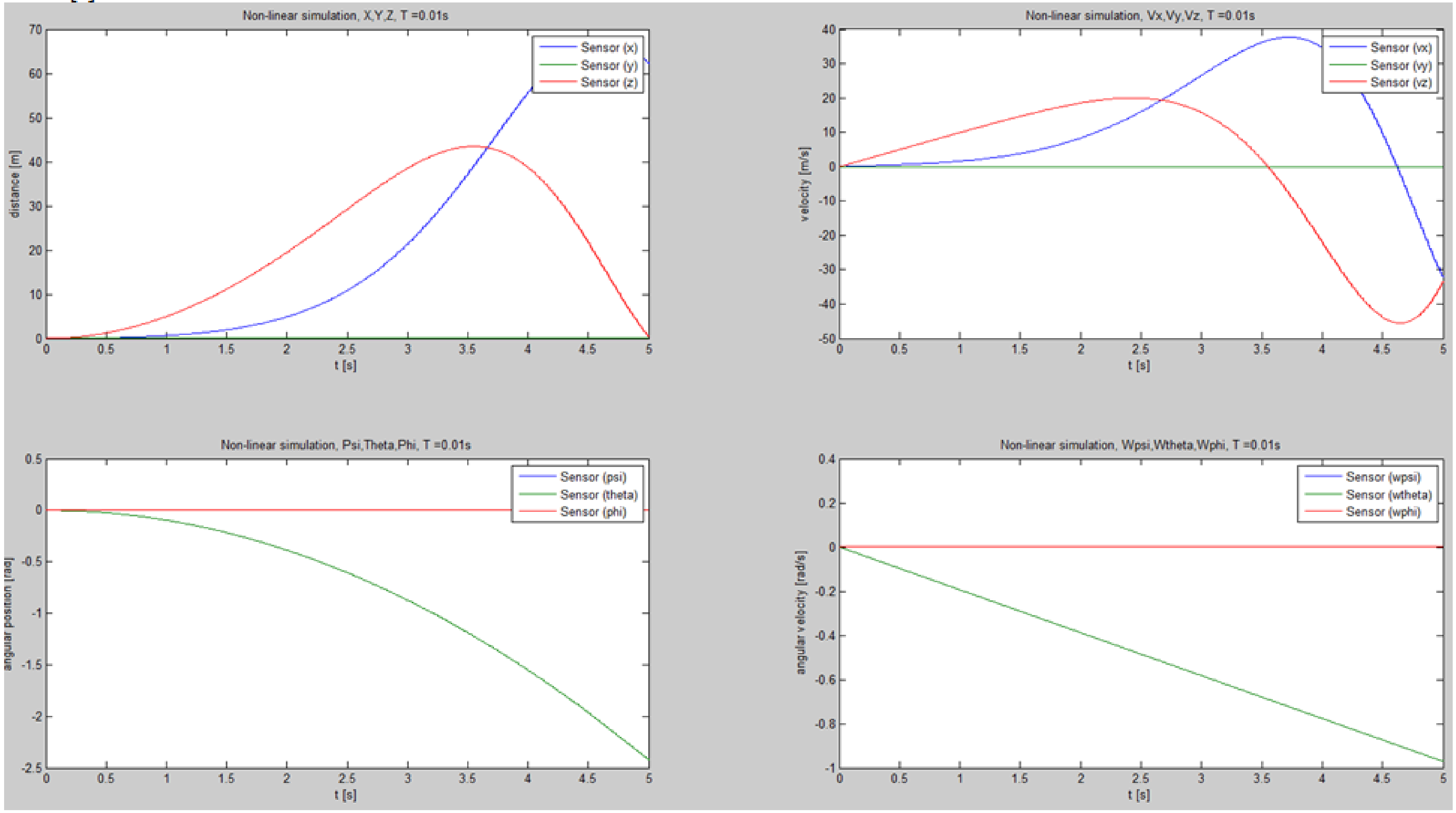


Nonlinear simulation with sampling period $T = 0.01\ [s]$, $\vec{U}^T = [0.1 \quad -0.1 \quad 0.1 \quad 0 \quad 0]$ (multiple step input – hover movement) and a $5\ [s]$ simulation time.





Nonlinear simulation with sampling period $T = 0.01\ [s]$, $\vec{U}^T = \begin{bmatrix} 1 & -1 & 0.6 & \pi/2 & \pi/2 \end{bmatrix}$ (multiple step input – cruise movement) and a $5\ [s]$ simulation time.


Non-linear simulation, X,Y,Z, T =0.01s
distance [m]
t [s]
Sensor (x)
Sensor (y)
Sensor (z)
Non-linear simulation, Vx,Vy,Vz, T =0.01s
velocity [m/s]
t [s]
Sensor (vx)
Sensor (vy)
Sensor (vz)
Non-linear simulation, Psi,Theta,Phi, T =0.01s
angular position [rad]
t [s]
Sensor (psi)
Sensor (theta)
Sensor (phi)
Non-linear simulation, Wpsi,Wtheta,Wphi, T =0.01s
angular velocity [rad/s]
t [s]
Sensor (wpsi)
Sensor (wtheta)
Sensor (wphi)