Robust Thrust Vector Control of Launch Vehicles

Simão Amaro
simaoamaro1999@tecnico.ulisboa.pt

Instituto Superior Técnico, Lisboa, Portugal

October 2022

Abstract

VIRIATO is a thrust vectored launch vehicle, whose flight mode while leaving the atmosphere ranges from subsonic to supersonic, such that it would be challenging to ensure a suitable performance throughout all its trajectory using classical controllers. To overcome this difficulty, a Robust controller is developed that ensures a fast response, robustness to mass variations, actuation delays and wind gusts. To benchmark the H_∞ controller, PD, LQG and NMPC controllers are also derived. In order to build and test these algorithms, Spin.Works developed the VIRIATO Test Platform, a small scaled prototype of VIRIATO with 1 m height. A simulation environment in *MATLAB/Simulink* is built, comprising the most significant kinematic and dynamic equations, the available sensors and their noise specification. Then the system is linearized, the transfer functions are used to construct the PD controller and the state space matrices the LQG and Robust controllers. Additionally, using the nonlinear equations, the NMPCEKF controller is formulated. The simulation results comparing the LQG, NMPCEKF and H_∞ controllers indicate that the latter is the fastest alongside the NMPCEKF and is robust to a mass variation of 20%, actuation delays of 0.1 s and wind gusts of 5 m/s or 35 N. The H_∞ was able to follow square, loop and eight shaped trajectories, while pointing at the center of the loops, with a position error below 0.5 m and a delay of 1 s. The position error increased slightly when the yaw reference changed abruptly.

Keywords: Thrust Vector Control, Robust Control, Kalman Filter, H_∞

1. Introduction

1.1. Motivation

SpaceX has sparked the interest of private companies in space exploration and launchers, having done, as of this moment, 187 launches, 149 landings and 124 total reflights [18]. The aforementioned success caused a considerable number of startups in this field of to appear over the years and more precisely, since 2017, this trend has seen an even bigger increase [20].

VIRIATO is a project that aims to develop, integrate and operate a sub-orbital vehicle to validate and test key technologies to the future development of a Portuguese microsatellite launcher, being launched preferably in the planned space port in Santa Maria, Açores [15].

In the scope of VIRIATO, Spin.Works is responsible for the development of the GNC subsystem [19]. To implement and test the algorithms, a smaller scaled VIRIATO, the VIRIATO Test Platform, with an height of 1 m was built. This thesis focus on the design and computational simulations of the GNC system in *MATLAB/Simulink*, namely the development of Robust position controllers and guidance algorithms that enable the vehicle to follow square, loop or eight shaped trajectories.

1.2. Literature review

Several small scale launch vehicles project, similar to the VIRIATO Test Platform, have been developed and some even tested, such as EAGLE. The flight controllers developed in this thesis were motivated to some extent by the aforementioned project. To decide what approaches to take, 3 publications were studied [4], [16] and [5] in years 2017, 2018 and 2019, respectively, which allowed to have a better understanding of the advantages and disadvantages of each control or estimation technique and also their limitations.

In the first year in [4], to estimate the position, velocity, Euler angles and angular velocity, a modular approach was taken, having 2 separate filters working together, in which a faster, Strapdown filter, ran at 100 Hz, performed the integration from accelerometers and gyroscopes, obtaining a state prediction and a slower Kalman filter, ran at 10 Hz, did the error correction, using measurements from GPS, altimeter and magnetometer. The control task was split into 4 categories, namely system identification, yaw control (the rotation around the vertical axis), thrust vector control and position control. The thrust vector control is responsible for the

attitude and altitude, receiving references from the outer loop position control. External to these control loops, a guidance system is responsible for generating position and velocity references. With this architecture, hand tuned PD controllers were used in the first place, but were not further discussed and replaced by sliding mode controllers. Additionally, a model-reference adaptive scheme, which feeds a nominal plant with the same inputs as the real plant, during flight, computes the difference in inputs required to compensate for the verified output variations. Both the inner and outer loops use sliding mode controllers (designed independently due to the difference in order of magnitude of the closed loop bandwidth of each loop). The altitude controller produced a response with a settling time of around 5-10 seconds and there is a constant 20 cm of error, which is expected given that the sliding mode controllers had their integral component disabled to prevent the excessive accumulation of error (windup). The position controllers have a settling time of about 10 seconds and also show some static error. The yaw controller works quite well as expected, having a fast response and very little error.

In the next year in [16], the same setup was used, with the addition of a saturation function replacing the sign function, which is a very common technique to avoid chattering in sliding mode control. Additionally, the model-reference adaptive scheme was removed. This time, it appears that the altitude simulations were the same as in [4], having changed only the position controllers, which is understandable because these had the worst results. In this work, the attitude controller was tested independently and exhibited 2° oscillations, which were attributed to inertial matrix uncertainties, wind gusts and sloshing. The results of the position controller were very similar to [4] and the authors considered them to be conservative and that more aggressive gains could be tested.

In the last studied attempt in [5], the sliding mode controllers were abandoned and replaced by traditional PID controllers. The results were again similar to the ones obtained in the previous years. The altitude controller had an offset of about 10%, which increased over time and was attributed to not having real-time mass estimation and using a constant nominal feedforward thrust input, while the mass of the vehicle was decreasing due to fuel consumption. The position controller had a settling time of about 15 s and afterwards, while hovering, a maximum deviation of 0.2 m, under mild winds speeds up to 5 m/s. In this work, a guidance system was designed, using a polynomial scheme to generate a continuous reference for velocity and position. This includes the possibility of limiting the maximum velocity and acceleration to reach the desired position. The results, including the guidance controller, prove that tracking is being done and the controller is able, although with error oscillations of 0.2 m, to track the position references.

From these 3 publications, some key ideas were derived, namely: a 2 phase prediction and correction estimator should be derived, sliding mode controllers do not exhibit better performance than simpler PID controllers and the control architecture of a guidance system, an outer and an inner loop, should be adopted. In fact, to validate the assumption that the sliding mode controllers did not lead to exceptional results, some exploratory simulations for VIRIATO were done using them, but the chattering problem was so persistent that to avoid it, the performance was severely compromised.

2. Background

2.1. System Description

The complete system is shown in figure 1.

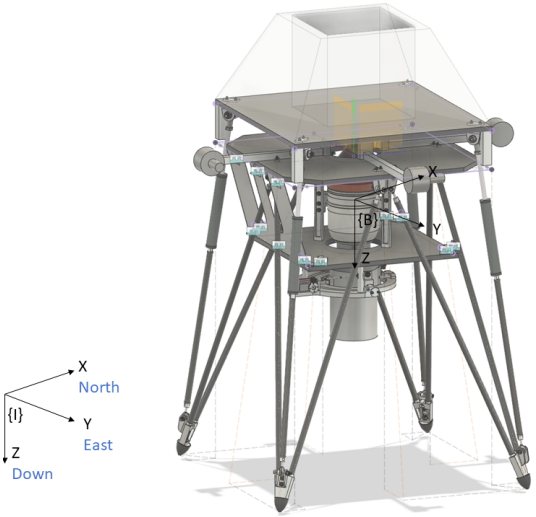


Figure 1: VIRIATO Test Platform [19].

The system has 4 actuators, thrust T from the jet engine, the outer and inner gimbals, δ and ϵ respectively and rotors that produce a torque τ_r, enabling to control 4 outputs simultaneously. In total, as it will be shown later, there are 12 states, which means it is an under-actuated system. To describe the vehicle with Newton-Euler equations, two reference frames are defined: the inertial and body frames, presented in figure 1. These allow for a complete description of the most important dynamics, without too much error given that the majority of the mass is fixed when seen from the body frame. Another more precise approach would be to use Denavit-Hartenberg parameters, defining more reference frames to capture all the dynamics (for instance, the rotation of the gimbals).

The transformation from the body to the inertial frame is given by the rotation matrix

$$R_B^I = R(\lambda) = R_z(\psi)R_y(\theta)R_x(\phi) \tag{1}$$

and the inverse transformation by $R_I^B = {R_B^I}^T$.

2.2. Gimbals and Jet Engine

The gimbals and thrust outlet are shown in figure 2.

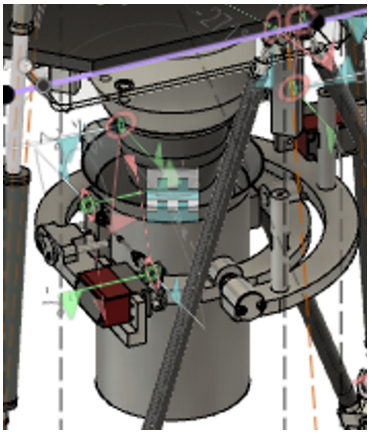


Figure 2: Gimbals and thrust outlet system [19].

In figure 3, the distance l between the centre of mass CG and the centre of thrust C_t is displayed. The rotation of the outer and inner gimbals δ, ϵ, respectively, can be seen in figure 4. The outer gimbal rotates the motor along the roll axis (x) of the body frame, producing a roll moment and a force in the transversal axis (y). The inner gimbal rotates the motor along the pitch axis (y) of the body, producing a pitching moment and a force in the longitudinal axis (x).

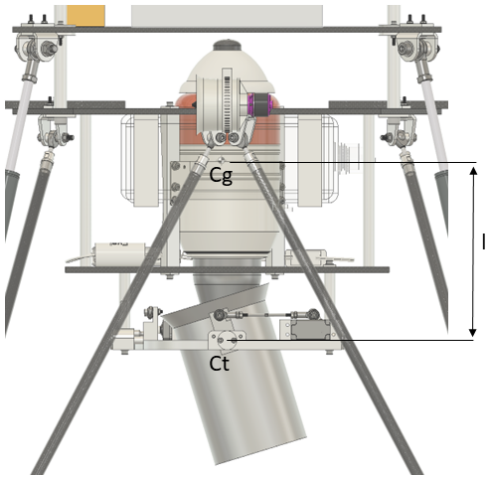
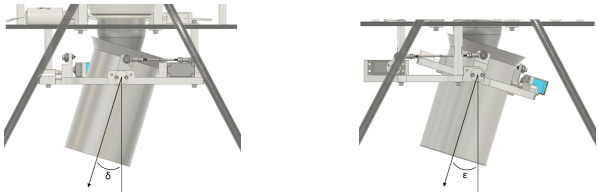


Figure 3: Gimbals and thrust outlet lateral view [19].

The rotation matrix of the outer gimbal to the body frame B is



(a) Outer gimbal δ (b) Inner gimbal ϵ

Figure 4: Outer, inner gimbals δ and ϵ [19].

$$R_\delta^B = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\delta & -\sin\delta \\ 0 & \sin\delta & \cos\delta \end{bmatrix} \tag{2}$$

and of the inner gimbal

$$R_\epsilon^B = \begin{bmatrix} \cos\epsilon & 0 & \sin\epsilon \\ 0 & 1 & 0 \\ -\sin\epsilon & 0 & \cos\epsilon \end{bmatrix}. \tag{3}$$

Thus, the force due to the jet engine expressed in the body frame is

$$\mathbf{T}_m = -R_\delta^B R_\epsilon^B T\mathbf{e}_z = \begin{bmatrix} -T\sin\epsilon \\ T\cos\epsilon\sin\delta \\ -T\cos\epsilon\cos\delta \end{bmatrix} \tag{4}$$

and the resulting motor torque τ_m

$$\tau_m = l\mathbf{e}_z \times \mathbf{T} = \begin{bmatrix} -Tl\cos\epsilon\sin\delta \\ -Tl\sin\epsilon \\ 0 \end{bmatrix}. \tag{5}$$

2.3. Rotors

One of the 4 rotors around the VIRIATO Test Platform can be seen in figure 5.

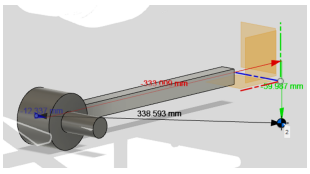


Figure 5: One of the 4 rotors of the VIRIATO Test Platform [19].

The total force generated by these 4 rotors when paired is null, resulting only in a torque $\tau_r\mathbf{e}_z$. Therefore, the total force of the inputs expressed in the body frame is $\mathbf{F}_B = \mathbf{T}$ and the torque in the body frame τ^B is

$$\tau^B = \tau_m + \tau_r\mathbf{e}_z = \begin{bmatrix} -Tl\cos\epsilon\sin\delta \\ -Tl\sin\epsilon \\ \tau_r \end{bmatrix} \tag{6}$$

Finally, the dynamics of the actuators are approximated by first order systems. This approximation

is common for gimbals (controlled by servos) or rotors; however, the engine could benefit from proper system identification. Nonetheless, choosing a first order system is an usual first approach.

$$h(s) = \frac{1}{st_u + 1}. \tag{7}$$

2.4. Newton-Euler Equations of Motion

The equations of motion are derived using the Newton-Euler equations, which combine the translational and rotational dynamics of a rigid body [2].

$$\frac{d}{dt}\begin{bmatrix}\mathbf{p}^B \\ \mathbf{v}^B \\ \lambda \\ \mathbf{\Omega}\end{bmatrix} = \begin{bmatrix}-\mathbf{\Omega}\times\mathbf{p} + \mathbf{v} \\ -\mathbf{\Omega}\times\mathbf{v} + \frac{\mathbf{F}_B}{m} + R_I^B g\mathbf{e}_z \\ Q(\lambda)\mathbf{\Omega} \\ J^{-1}(\tau_B - \mathbf{\Omega}\times J\mathbf{\Omega})\end{bmatrix} \tag{8}$$

with $\boldsymbol{\lambda}$ the Euler angles and $Q(\boldsymbol{\lambda})$ the transformation from angular velocities to Euler rates

$$Q(\lambda) = \begin{bmatrix}1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi\sec\theta & \cos\phi\sec\theta\end{bmatrix} \tag{9}$$

2.5. State Space Horizontal Equations

The linearized model is

$$f(x) = A\mathbf{x} + B\mathbf{u}. \tag{10}$$

The matrix can be decoupled into longitudinal, lateral and vertical movements. The longitudinal movement A_x matrix is

$$A_x = \begin{array}{c} \\ \dot{p_x^B} \\ \dot{v_x^B} \\ \dot{\theta} \\ \dot{w_y} \end{array} \begin{array}{c} \begin{array}{cccc} p_x^B & v_x^B & \theta & w_y \end{array} \\ \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & -g & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \end{array} \tag{11}$$

and the B_x matrix

$$B_x = \begin{array}{c} \\ \dot{p_x^B} \\ \dot{v_x^B} \\ \dot{\theta} \\ \dot{w_y} \end{array} \begin{array}{c} \begin{array}{c} \epsilon \end{array} \\ \begin{bmatrix} 0 \\ -g \\ 0 \\ \frac{-mgl}{J_{yy}}. \end{bmatrix} \end{array} \tag{12}$$

The state space equations for the lateral movement is analogous.

2.6. Height Transfer Function

Combining the position by velocity in z,

$$\dot{p}_z^B = v_z^B \leftrightarrow p_z^B = \frac{v_z^B}{s}. \tag{13}$$

with the velocity in z by the thrust T,

$$\dot{v}_z^B = -\frac{1}{m}T \leftrightarrow v_z^B = -\frac{1}{m}\frac{1}{s}T, \tag{14}$$

one gets

$$p_z^B = -\frac{1}{m}\frac{1}{s^2}T. \tag{15}$$

2.7. Yaw Transfer Function

Combining the yaw by angular velocity in z,

$$\dot{\psi} = w_z \leftrightarrow \psi = \frac{w_z}{s} \tag{16}$$

with the angular velocity in z by the torque of the rotors τ_r,

$$\dot{w_z} = \frac{1}{J_{zz}}\tau_r \leftrightarrow w_z = \frac{1}{J_{zz}}\frac{1}{s}\tau_r \tag{17}$$

one gets

$$\psi = \frac{1}{J_{zz}}\frac{1}{s^2}\tau_r. \tag{18}$$

2.8. Sensor Specifications

To observe the position, Euler angles and angular velocity states, the Xsens mti-680 was selected [22]. Although this unit can provide velocity measurements, in this work it is considered that they are not available, so their noise is not going to be quantified. From the datasheet [21], the sensor specifications are shown in table 1.

Table 1: Sensor specification of the Xsens mti-680 in SI units.

	Horizontal Pos.	Vertical Pos.
RMS	1	2

Roll/Pitch	Yaw	Angular Vel.	Acceleration
0.0035	0.0175	0.0012*	0.0059*

*Obtained from the PSD of the noise at 100 Hz.

2.9. Disk Margins

Disk margins are a more comprehensive approach to stability margins than classical gain and phase margins, namely in the domain of multiple input multiple output *MIMO* systems. Contrary to classical margins, disk margins compare variations in gain and phase simultaneously, leading to a disk in a plane having as y axis the phase margin and x axis the gain margin. Additionally, it takes into account multi loop variations (in the system in equation 11, there are 4 loops, so it would vary the gain and phase of them individually simultaneously) at the inputs and outputs (it can be shown that perturbations at the inputs can affect stability differently than at the outputs) [17].

An example disk $D(\alpha, \sigma)$, having $\sigma = 0.2$ and $\alpha = 0.75$ is displayed in figure 6. From these two values, a minimum and maximum gain of γ_{min} and γ_{max}, respectively, can be multiplied to the system, assuming a null variation in phase. On the contrary, a maximum phase of ϕ_{max} can be added, assuming a null variation in gain. The function *diskmargin* of MATLAB was used to perform the calculations [12].

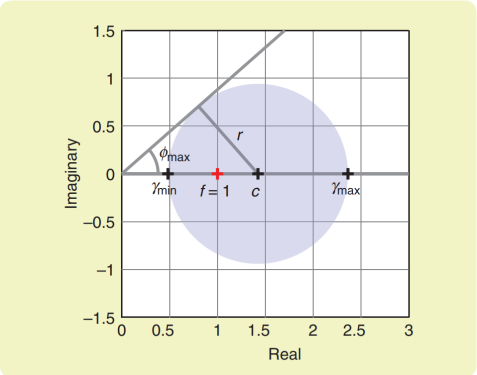


Figure 6: The set of variations $D(\alpha, \sigma)$ having $\sigma = 0.2$ and $\alpha = 0.75$ [17].

3. Implementation

3.1. Pre Kalman Filtering Position Measurements

To overcome the noisy position signals in table 1, a Kalman filter is derived, combining the inertial positions and accelerations from the Xsens mti-680 unit using kinematic laws. The equations are

$$\begin{aligned} \dot{\boldsymbol{x}} &= A\mathbf{x} + B\mathbf{u} \\ \boldsymbol{y} &= C\mathbf{x}, \end{aligned} \tag{19}$$

with $\mathbf{x}$ the states, $\mathbf{u}$ the known inputs and $\mathbf{y}$ the outputs. For a steady state Kalman filter, the gain can be computed once by solving the associated Riccatti equation [3] with the aforementioned matrices and the costs Q and R.

The cost Q refers to the process noise, so the bigger it is, the less we trust the model. On the contrary, the R matrix corresponds to the measurement noise, being usual to make experimental tests to quantify sensor noise variance to populate the elements of the R matrix. Just as the Q matrix, the bigger the R, the less we trust the measurements and so their noise is reduced.

The state space matrices for the position kalman filter KFP are

$$A_{KFP} = \begin{matrix} \\ \dot{p} \\ \dot{v} \end{matrix} \begin{matrix} \begin{matrix} p & v \end{matrix} \\ \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \end{matrix}, \quad B_{KFP} = \begin{matrix} \\ \dot{p} \\ \dot{v} \end{matrix} \begin{matrix} \begin{matrix} acc \end{matrix} \\ \begin{bmatrix} 0 \\ 1 \end{bmatrix} \end{matrix} \tag{20}$$

and

$$C_{KFP} = \begin{matrix} \\ y \end{matrix} \begin{matrix} \begin{matrix} p & v \end{matrix} \\ \begin{bmatrix} 1 & 0 \end{bmatrix} \end{matrix} \tag{21}$$

with p the position, v the velocity and acc the known acceleration. Computing the Kalman filter with the function *kalman* [14] having

$$Q = 1 \times 10^{-8} and\, R = 1 \tag{22}$$

leads to the gain

$$L = \begin{bmatrix} 0.0811 & 0.0033 \\ 0.0033 & 0.0003 \end{bmatrix}. \tag{23}$$

The gains are very small, which is expected because the position measurements are very noisy and by having a small Q compared to R there is a small trust in them.

With these specifications, several tests were conducted, coming to the conclusion that the cost Q should not be increased because that would lead to a higher reliance in the position measurements and thus more noise; however, it could also not be decreased due to the increase in drift.

The position and velocity errors average around 10^{-3} m and 10^{-5} m/s in the beginning of the simulation, but could increase up to 0.035 m and 10^{-4} m/s due to drift, respectively.

3.2. Kalman Filtering Horizontal Movement LTR Technique

Using the pre filtered position and velocity measurements from the previous subsection 3.1 and the state space equations 11 and 12, a Kalman filter can be derived.

In this work, the loop transfer recovery (LTR) technique is used, which aims to recover the properties of a full feedback controller, even when not having access to some states [8].

The algorithm is split into two phases, prediction and correction, and can be described by the following equation

$$\dot{\mathbf{x}} = A\mathbf{x} + B\mathbf{u} + L[\mathbf{y} - C\mathbf{x}]. \tag{24}$$

According to the LTR technique [8], in the design of the Kalman filter, the perturbations are introduced as

$$\dot{x} = A\mathbf{x} + B\mathbf{u} + B\mathbf{d} \tag{25}$$

which means that the perturbations are weighted by the input matrix. To validate these results, the loop gain for different values of σ is shown in figure 7.

The results having the different values of σ were very similar, so in the end $\sigma = 10$ was chosen because it is enough to give a clear estimate of the states and would lead to the most noise reduction. Following the LTR technique, the R matrix was chosen as the identity matrix.

3.3. Height PD Controller

The transfer function between the input thrust T and the z position including the time constant of the motor t_m of 0.1 s is

$$p_z^B = -\frac{1}{m}\frac{T}{s^2(t_m s + 1)} = -\frac{0.0727}{s^2(0.1s + 1)} \tag{26}$$

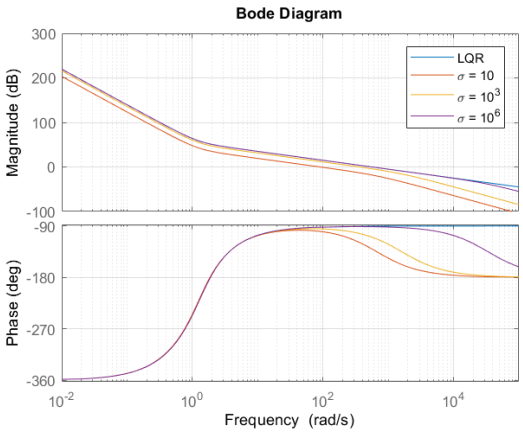


Figure 7: Loop gain using several σ values.

The pole at $s = -10$ is due to the time constant of the motor, the s^2 is the double integration from vertical force to position and the negative gain is due to the upward direction of the thrust along the negative z axis (the z axis points downwards). Using the function *pidTuner* of *MATLAB* it is possible to define the gains according to the given specifications.

The PD controller structure is:

$$PD_z = Kp_z + \frac{Kd_z s}{a_z s + 1} \tag{27}$$

with $Kp_z = -30, a_z = 0.00139, Kd_z = -50$.

The gain and phase margins are 45.5 dB and 60.7^{o}, respectively. The closed loop bandwidth is 5.56 rad/s. The rise, settling times are 0.35 and 3.64 seconds, respectively and the overshoot is 15.2%.

3.4. Yaw PD Controller

The transfer function between the input torque τ_r and yaw including the time constant of 0.005 seconds of the rotors is

$$\psi = \frac{1}{J_{zz}} \frac{\tau_r}{(t_r s + 1)s^2} = \frac{1.601}{s^2(0.005s + 1)} \tau_r \tag{28}$$

The PD controller is:

$$PD_\psi = Kp_\psi + \frac{Kd_\psi s}{a_\psi s + 1} \tag{29}$$

with $Kp_\psi = 113$, $Kd_\psi = 21.2$ and $a_\psi = 0.00269$.

The resulting gain and phase margins are 22.6 dB and 70.1^{o}, respectively. The closed loop bandwidth is 61.8 rad/s. The settling time is 0.1 seconds and the overshoot is 3.45%.

3.5. LQG Position Controller

The dynamics can be described by

$$\begin{aligned} \dot{\boldsymbol{x}} &= A\mathbf{x} + B\mathbf{u} \\ \boldsymbol{y} &= C\mathbf{x}, \end{aligned} \tag{30}$$

with $\mathbf{x}$ the states, $\mathbf{u}$ the inputs and $\mathbf{y}$ the outputs. The cost function of the system is

$$J = \int_0^\infty \mathbf{x}^T Q \mathbf{x} + \mathbf{u}^T R \mathbf{u} \quad dt, \tag{31}$$

and the algorithm attempts to minimize this cost by solving the corresponding Riccati equation. It can also be proven that the resulting gain from this solution leads to an asymptotically stable closed loop [10].

The real task in LQR design is to choose suitable Q and R matrices such that the system has the required performance in terms of response and robustness. A starting point can be settled using the Bryson's rule [10],

$$\mathrm{Q} = \begin{bmatrix} \frac{1}{0.1^2} & 0 & 0 & 0 \\ 0 & \frac{1}{0.1^2} & 0 & 0 \\ 0 & 0 & \frac{1}{(1.2\pi/180)^2} & 0 \\ 0 & 0 & 0 & \frac{1}{(1\pi/180)^2} \end{bmatrix} \tag{32}$$

and

$$\text{and } \mathrm{R} = \frac{1}{12 \times \pi/180} = 11.36. \tag{33}$$

The gains were computed using the *lqr* function of *MATLAB*. With these specifications, the rise and settling times to a step input are 2.54 and 4.39 seconds, respectively. The closed loop bandwidth is 0.856 rad/s. The gain, phase and disk margins (using the *diskmargin* function of MATLAB) of the system with the LQG controller are 4.6 dB, 23.8° and 0.16, respectively.

3.6. NMPCEKF controller

In addition to equation 8, the mass of the vehicle is added as a state and the EKF updates it at each time step. The dynamic equation of the mass is $\frac{d\delta_m}{dt} = 0$, assuming that in the prediction horizon of the NMPC the mass does not change (which is a reasonable approximation assuming a short time of a few seconds).

The chosen prediction and control horizons are

$$\begin{aligned} P_h &= 2s \\ C_h &= 2s \end{aligned} \tag{34}$$

accounting for 20 time steps, as the sample time is

$$h_s = 0.1s \tag{35}$$

The cost function is similar to the linear MPC

$$J = \sum_{k=1}^{P_{h_n}} \mathbf{y}(k)^T Q \mathbf{y}(k) + \mathbf{u}(k)^T R \mathbf{u}(k) \tag{36}$$

The constraints are:

$$\begin{aligned} -17^{\text{o}} <= \delta <= 17^{\text{o}} \\ -17^{\text{o}} <= \epsilon <= 17^{\text{o}} \\ 0N <= T <= 220N \\ -1.1Nm <= \tau_r <= 1.1Nm \end{aligned} \tag{37}$$

To formulate the optimization problem, the multiple shooting technique was used, leading to the constraint

$$\sum_{k=1}^{P_{h_n}} \mathbf{x}(k+1) - f(\mathbf{x}(k), \mathbf{u}(k)) = 0. \tag{38}$$

The solver tries not only to find the optimal inputs but also the states. The dimensionality of the problem increases; however it is sparse and more linear, reducing the calculation speed [9]. Lastly, the chosen solver was CasADi [1] using an interior point method [6], which proved to be significantly faster than the solver of MATLAB.

Then, the EKF extended Kalman filter is derived. The objective is to estimate the mass of the VIRIATO Test Platform to use in the NMPC model. The algorithm can be summarized in 2 steps, prediction and correction, as is summarized in equations 39 to 44.

Prediction of state, covariance and output estimates

$$\hat{\mathbf{x}}_{k|k-1} = f(\hat{\mathbf{x}}_{k-1|k-1}, \mathbf{u}(k)) \tag{39}$$

$$\mathbf{P}_{k|k-1} = \mathbf{\Phi}_k \mathbf{P}_{k-1|k-1} \mathbf{\Phi}_k^T + \mathbf{Q}_k \tag{40}$$

$$\mathbf{y}_k = h(\hat{\mathbf{x}}_{k|k-1}) \tag{41}$$

gain computation

$$\mathbf{K}_k = \mathbf{P}_{k|k-1} \mathbf{H}_k^T \left[\mathbf{H}_k \mathbf{P}_{k|k-1} \mathbf{H}_k^T + \mathbf{R}_k \right]^{-1} \tag{42}$$

correction of state and covariance estimates

$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k (\mathbf{z}_k - \mathbf{y}_k) \tag{43}$$

$$\begin{aligned} \mathbf{P}_{k|k} = (\boldsymbol{I}_n - \mathbf{K}_k \mathbf{H}_k) \mathbf{P}_{k|k-1} (\boldsymbol{I}_n - \mathbf{K}_k \mathbf{H}_k)^T + \\ + \mathbf{K}_k \mathbf{R}_k \mathbf{K}_k^T \end{aligned} \tag{44}$$

The details of the matrices can be studied in [7].

3.7. H_∞ Position Controller

In H_2 or H_∞ control, the problem is formulated by the following specifications

$$\min_{\mathbf{K}} ||\mathbf{N}(k)||_{2 \text{ or } \infty}, \quad \mathbf{N} = \begin{bmatrix} \mathbf{W}_u \mathbf{KS} \\ \mathbf{W}_T \\ \mathbf{W}_P \mathbf{S} \end{bmatrix} \tag{45}$$

with $\mathbf{S}$ being the sensitivity function (for performance), $\mathbf{T}$ the complementary sensitivity function (for robustness and noise attenuation) and the loop transfer $\mathbf{KS}$ (to penalize large inputs).

The state space system in equation 11 comprises 4 poles at the origin, which means that the system must be shifted by an arbitrary value α, the H_∞ gains computed and then shifted back to the original plant [23]. In this case, to achieve a fast performance, an α of 2 was chosen, so the poles move to $s = -2$.

Then, the plant is augmented with weighting functions W_1, W_2 and W_3 [11], corresponding to $\mathbf{W}_P$, $\mathbf{W_u}$ and $\mathbf{W}_T$, respectively. W_1 should be high in the control bandwidth to ensure good reference tracking and disturbance rejection (obtaining small sensitivity S), whereas W_3 should be the opposite and is responsible for robustness (in the case of multiplicative disturbance) and noise attenuation (to achieve small complementary sensitivity T). In order to limit control effort in a particular band, W_2 should be large to obtain small $\mathbf{KS}$.

Given that the weighting functions W_1,W_2 and W_3 shape N, in order to minimize its norm, these functions should be chosen accordingly, by following the aforementioned guidelines and with some tuning. In this case, the weighting functions are the same for the 3 algorithms, where W_1 was designed by setting a pole close to the desired bandwidth and a zero a decade faster, with unity static gain

$$W_1 = \frac{0.1(s + 10w)}{s + w}, \tag{46}$$

with w the frequency of the pole so that the transfer function has a bandwidth close to its location. The weighting functions are

$$\mathbf{W}_1 = \begin{bmatrix} \frac{0.1(s+62.8)}{s+6.28} \\ \frac{0.1(s+628)}{s+62.8} \\ \frac{0.1(s+1885)}{s+188.5} \\ \frac{0.1(s+4398)}{s+439.8} \end{bmatrix}, \mathbf{W}_3 = \begin{bmatrix} \frac{10(s+6.28)}{s+62.8} \\ \frac{10(s+62.8)}{s+628} \\ \frac{10(s+188.5)}{s+1885} \\ \frac{10(s+439.8)}{s+4398} \end{bmatrix}. \tag{47}$$

Note that the weighting function $\mathbf{W}_3$ is the inverse of $\mathbf{W}_1$ to ensure robustness, due to the fact that for higher frequencies $\mathbf{W}_3$ should be large to attenuate undesired dynamics and noise. Lastly,

$$W_2 = 1/(17 \times pi/180), \tag{48}$$

sets actuation cost.

The *Hinfsyn* function of MATLAB was used to find the controller that minimizes the peak singular value [13].

With these specifications, the rise and settling times to a step input are 1.24, 2.3 seconds, respectively. The closed loop bandwidth is 1.81 rad/s or 0.29 Hz. The gain, phase and disk margins (using the *diskmargin* function of MATLAB) of the system with the H_∞ controller are 12.2 dB, 75.9° and 0.22, respectively.

4. Results

4.1. Control Performance

After testing the controllers in nonlinear simulations, some conclusions can be taken.. Firstly, the height and yaw are successfully controlled using a PID and a PD only, so more complex techniques are not necessary to ensure good tracking performance, disturbance rejection and robustness.

Conversely, the position loop could benefit from more sophisticated controllers such as the NMPCEKF or the H_∞. Both offer different benefits, namely the NMPCEKF is better suited for complex trajectories which have simultaneous varying references, whereas the H_∞ controller excels in simple trajectories, ensuring the highest level of robustness and disturbance rejection.

A summary of the results is present in table 2. All controllers are capable of dealing with the expected mass variation of -2.5 Kg, the NMPCEKF and H_∞ controllers are the fastest and the latter has the best robustness and disturbance rejection properties, as mentioned in the previous paragraph, moving only 0.6 m to a wind gust disturbance of 5 m/s or 35 N.

Lastly, the LQG controller has worse performance in almost all parameters, which confirms that the added complexity of the NMPCEKF or H_∞ could compensate.

Table 2: Simulation results of the 3 controllers.

	LQG	NMPCEKF	Hinf
Settling time [s]	5	4	4
Robust to mass variation	yes		
Max. actuation delay [s]	0.1	0.1	0.5
Wind gusts max. pos. error [m]	4.2	1.2*	0.6

*In the previous simulations the settling times of the LQG and H_∞ are affected by the varying yaw. If the yaw was kept constant, the settling times would be as indicated in the table.
**There was also an error in the Down position and after 10 seconds
the NMPCEKF was able to return to its initial position.

4.2. Estimation Errors

Here the estimation errors are presented for a simulation having step references of 1 m in North, East, Down and a yaw of 150°.

Using the linear Kalman filter, the errors of the position, velocity and angular velocity estimates average a mean error of 10^{-3} m, 10^{-4} m/s and 10^{-3} °/s, respectively, showing a good performance; however, the Euler angles have significant errors, reaching up to 2°. This is mostly due to the fact that when the yaw is varying, the Euler angles and the angular velocities are coupled (see equation 8) which leads to model mismatch. To prevent this issue, the Q matrix of the filter could be increased to represent the lack of trust in the model, but this would in turn increase noise.

The extended Kalman filter EKF did not show the limitation of increased error when the yaw varies, due to its nonlinear nature. It averaged 10^{-3} position (m), velocity (m/s) and angular velocity (°/s) mean errors, and Euler angle errors of 0.05°.

4.3. Guidance

A trajectory is fed to the controller, having a constant 2 m/s wind perturbation pushing the VIRIATO Test Platform North.

Given that the H_∞ controller is able to control the horizontal position directly, the guidance algorithm only has to limit the position reference and generate trajectories for the VIRIATO Test Platform to follow.

To limit the references, a simple solution was found,

$$\begin{aligned} e &= r - x \\ e_l &= min(2, max(-2, e)) \\ r_l &= x + e_l \end{aligned} \tag{49}$$

with e, r, x, e_l, r_l the error, reference, current state, limited error and reference, respectively. This way, the error is bounded to -2 and 2, which ensures that the inputs stay within a reasonable value.

The VIRIATO Test Platform has to follow an eight shaped trajectory, shown in the red line in figure 9, while pointing at the closest circumference center.

The tracking results of the position and velocity states in figure 8 exhibit a smooth trajectory with a delay of 1 second. When the VIRIATO Test Platform crosses the 4 m North mark, the yaw reference changes 180°, behaving as a step reference, and the response takes approximately 3 seconds to settle again, with little overshoot.

In figure 9, the path is closely followed, except in the segment where yaw has to change 180°, consequently affecting the position of the VIRIATO Test Platform. This behaviour exists because the position controller is not able to react as quickly as the yaw controller, thus suffering position errors. The wind perturbation shifts the trajectory to the right by 0.3 meters.

The gimbals move abruptly in the beginning to tilt the VIRIATO Test Platform in the right direction and then do small changes to keep it in the desired trajectory. Notice that in seconds 9 and 21 the inner gimbal shows again an abrupt response, which corresponds to the crossing of the 4 m North (figure 8 and a change of 180° of yaw, so the position controller has to compensate the yaw changes. The thrust also varies significantly at the 9 and 21 seconds mark, for the same reason as described previously and it is enough to keep the vertical (Down) error below 0.7 m (figure 8).

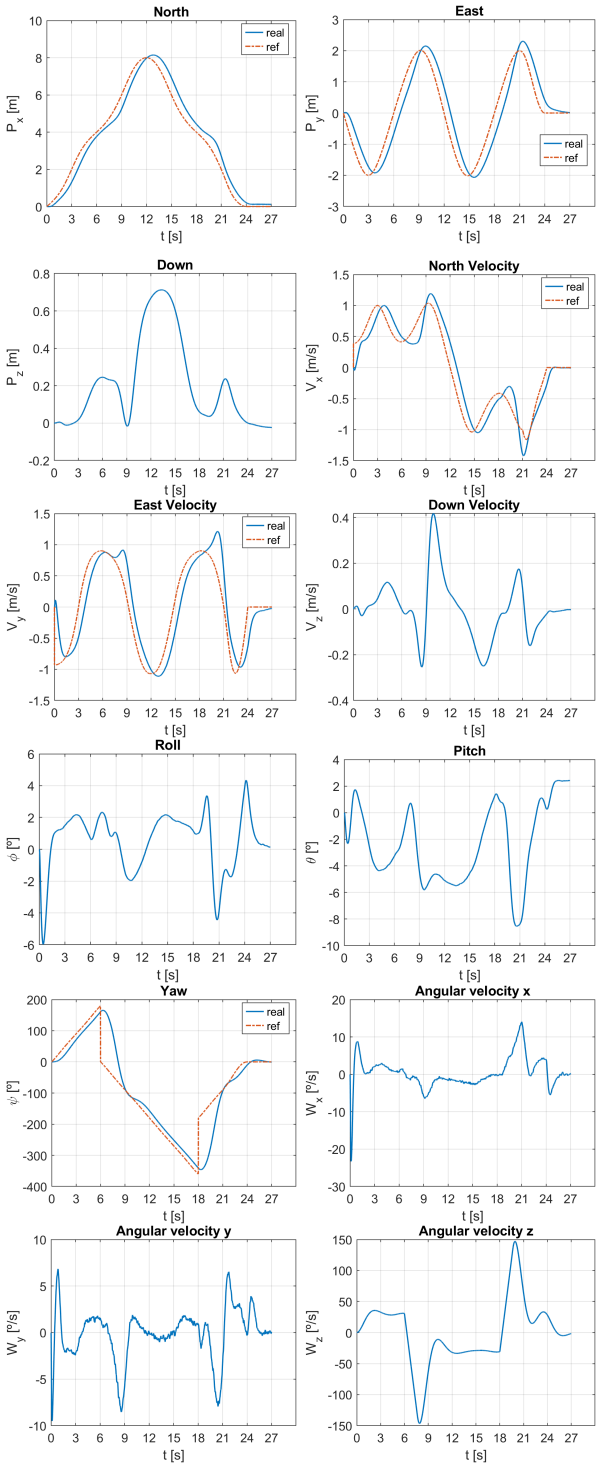
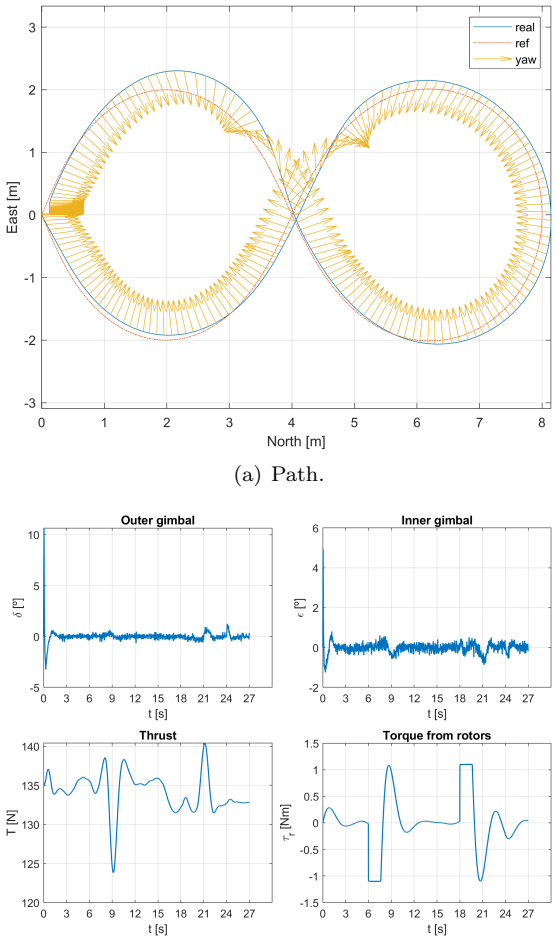


Figure 8: States of the VIRIATO Test Platform. throughout the loop trajectory.



(a) Path.

(b) Inputs.

Figure 9: Infinity path in the North-East plane and inputs of the VIRIATO Test Platform.

5. Achievements

Firstly, the nonlinear and linear dynamics, both in transfer function and state space form were derived. Additionally, the noise specifications of the sensors were presented and the position and acceleration measurements were combined to produce cleaner position and velocity estimates.

Classical PD controllers were developed for height and yaw. The LQG and NMPCEKF algorithms were derived to benchmark the H_∞ controller. The latter was tuned using weighting functions, which proved to be challenging.

The 3 controllers were compared against the maximum expected mass variation of -2.5 Kg or 20%, actuation delays of 0.1 s in the gimbals and wind gusts of 5 m/s equalling 35 N. All were able to deal with the expected mass variation with little effort, but the H_∞ showed the best response to wind gust perturbations of 5 m/s, moving a maximum of 0.6 m and robustness against actuation delays up to 0.5 s.

An outer guidance loop was developed that generates the trajectories for the H_∞ position controller to follow and limits the references at each time step not to exceed a certain threshold, chosen as 2 m. This setup was able to follow the desired eight shaped path with a maximum error of 0.5 m, mostly due to the wind perturbation of 2 m/s and the drastic variation of the yaw reference of 180°.

All things considered, a guidance and control system was developed that enables the VIRIATO Test Platform to follow complex trajectories such as the eight shaped presented in this work, having little position and yaw errors. As can be proved, the proposed final controller is faster than the usual PD or LQG algorithms and exhibits robustness to mass variations of -2.5 Kg or 20%, 0.1 s of actuation delay and wind gusts of 5 m/s or 35 N.

Acknowledgements
I would like to firstly thank Spin.Works, namely Francisco Câmara and David Esteves, for the opportunity to work in the VIRIATO project and for providing me with all the necessary tools and insightful advice.

Additionally, I would like to express my gratitude for the guidance given by Professor Rita Cunha, who I had met in another course and promptly accepted my request to be my supervisor.

Likewise, I thank Técnico Solar Boat (TSB) for letting me pursue my passion in control theory and for all the amazing team members.

References

[1] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl. CasADi – A software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation*, 11(1):1–36, 2019.

[2] Z. Benić, P. Piljek, and D. Kotarski. Mathematical modelling of unmanned aerial vehicles with four rotors. *Interdisciplinary Description of Complex Systems*, 14:88–100, 01 2016.

[3] M. I. L. de Faria Ribeiro. Kalman and extended kalman filters: Concept, derivation and properties. *Robotics WEBook, http://www.euron.org*, 2004.

[4] M. Dumke, M. Sagliano, P. Saranrittichai, G. F. Trigo, and S. Theil. Eagle - environment for autonomous gnc landing experiments. In *10th International ESA Conference on Guidance, Navigation and Control Systems*, June 2017.

[5] M. Dumke, G. F. Trigo, M. Sagliano, P. Saranrittichai, and S. Theil. Design, development, and flight testing of the vertical take-off and landing GNC testbed EAGLE. *CEAS Space Journal*, 12(1):97–113, Jan. 2020.

[6] M. Fazlyab, S. Paternain, V. M. Preciado, and A. Ribeiro. Interior point method for dynamic constrained optimization in continuous time, 2015.

[7] FDNunes. V - gps. DEEC, IST, 2020.

[8] J. P. Hespanha. *Linear Systems Theory*. 2018.

[9] M. Kelly. Transcription methods for trajectory optimization: a beginners tutorial. 07 2017.

[10] C. Ma. Application of lqr and h2-optimal control for a quadrotor system. Master's thesis, Huazhong University of Science and Technology, 2020.

[11] MATLAB. augw. `https://www.mathworks.com/help/robust/ref/lti.augw.html`. Accessed: 2022-05-29.

[12] MATLAB. diskmargin. `https://www.mathworks.com/help/robust/ref/diskmargin.html`. Accessed: 2022-05-25.

[13] MATLAB. hinfsyn. `https://www.mathworks.com/help/robust/ref/lti.hinfsyn.html`. Accessed: 2022-05-29.

[14] MATLAB. kalman. `https://www.mathworks.com/help/control/ref/ss.kalman.html`. Accessed: 2022-06-22.

[15] Omnidea. Viriato. `https://www.omnidea.net/viriato.html`. Accessed: 2022-10-30.

[16] M. Sagliano, M. Dumke, and S. Theil. Simulations and flight tests of a new nonlinear controller for the eagle lander. *Journal of Spacecraft and Rockets*, 56(1):259–272, 2019.

[17] P. Seiler, A. Packard, and P. Gahinet. An introduction to disk margins [lecture notes]. *IEEE Control Systems Magazine*, 40(5):78–95, 2020.

[18] SpaceX. Launches. `https://www.spacex.com/launches/`. Accessed: 2022-10-30.

[19] Spin.Works. Viriato – gnc subsystem. `https://www.spinworks.pt/portfolio/viriato-gnc-subsystem/`. Accessed: 2022-10-30.

[20] VentureRadar. Top launcher start-ups. `https://www.ventureradar.com/startup/launcher`. Accessed: 2022-10-30.

[21] Xsens. Mti 600-series performance specifications. `https://mtidocs.xsens.com/sensor-specifications-2`. Accessed: 2022-09-29.

[22] Xsens. Xsens mti 600-series. `https://www.xsens.com/products/mti-600-series`. Accessed: 2022-09-29.

[23] K. Zhou, J. C. Doyle, and K. Glover. *ROBUST AND OPTIMAL CONTROL*. Prentice Hall, 1^{st} edition, 1996.