UNIVERSITÉ D'EVRY-VAL-D'ESSONNE

Escola d'Enginyeria de Telecomunicació i Aeroespacial de Castelldefels

UNIVERSITAT POLITÈCNICA DE CATALUNYA

BACHELOR'S FINAL PROJECT

Title: Longitudinal stability Control System design for the UAV Ultra Stick 25e

Author: Josep Llobera Capllonch

Director: Yasmina Bestaoui Sebbane

Date: January, 27th 2015

Title: Longitudinal stability Control System design for the UAV Ultra Stick 25e

Author: Josep Llobera Capllonch

Director: Yasmina Bestaoui Sebbane

Date: January, 27th 2015

Overview

The purpose of this study is to design a system control in order to obtain the best longitudinal stability of the UAV Ultra Stick 25e. The aircraft is shown in Figure 0.

Figure 0. Ultra Stick 25e

The state-space model has been taken from a previous study of the University of Minnesota which its tittle is *System Identification for Small, Low-Cost, Fixed-Wing Unmanned Aircraft.* This study will be explained briefly as background but it is not the aim of the project to go in depth in this matter. This project focuses on the comparison of Classical Control, Optimal Control and Robust Control methods in order to find the best solution for the longitudinal stability of the Ultra Stick 25e. To design the controllers and to study the responses of the control systems I have used Matlab's Control System and Robust Control Toolboxes.
Only continuous time systems have been treated.

INDEX

1. INTRODUCTION

1.1. Longitudinal stability

The stability of an aircraft is defined as the aircraft's ability to sustain a specific, prescribed flight condition. The concept of stability is closely related to the equilibrium of the aircraft. If the net forces and moments exerted on the aircraft is zero, the aircraft is in equilibrium, in that flight condition: the lift equals the weight, the thrust equals the drag, and no moment of force acting on the aircraft.

In this case, the longitudinal stability of an aircraft refers to the aircraft's stability in the pitching plane, which describes the position of the aircraft's nose in relation to its tail and the horizon.

1.1.1. Static stability

When an aircraft undergoes some turbulence (or some form of static imbalance) when in equilibrium flight, the nose tilts slightly up or down (an increase or decrease in the angle of attack), or there will be a slight change in flight attitude. There are additional forces acting on the aircraft, and it is no longer in the equilibrium condition.

If the aircraft continues to increase the orientation after disturbance, the aircraft is said to be statically unstable. If there are no further changes in flight attitude and if the aircraft retains the position, which means there are no net forces or moments acting on the aircraft in the new orientation too, then the aircraft is said to be statically neutral. If forces are generated on the aircraft in a way such that forces causing the disturbance are countered, and the aircraft attains its original position, then the aircraft is said to be statically stable. In the figure below (Fig. 1.1) you can see a schematic explanation.

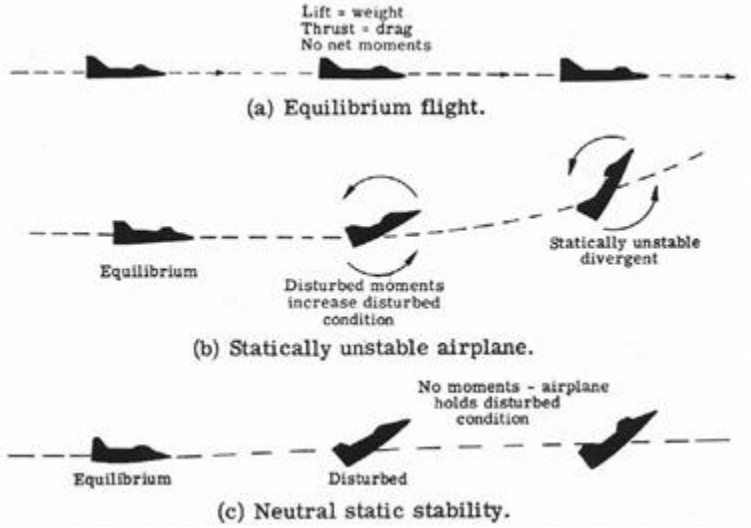


Fig. 1.1 Static stability scheme

1.1.2. Dynamic stability

If an aircraft is statically stable, it may undergo three types of oscillatory motion during flight. When imbalance occurs the airplane attempts to retain its position, and it reaches the equilibrium position through a series of decaying oscillations, and the aircraft is said to be dynamically stable. If the aircraft continues the oscillatory motion without decay in the magnitude, then the aircraft is said to be on dynamically neutral. If the magnitude oscillatory motion increases and the aircraft orientation start to change rapidly, then the aircraft is said to be dynamically unstable.

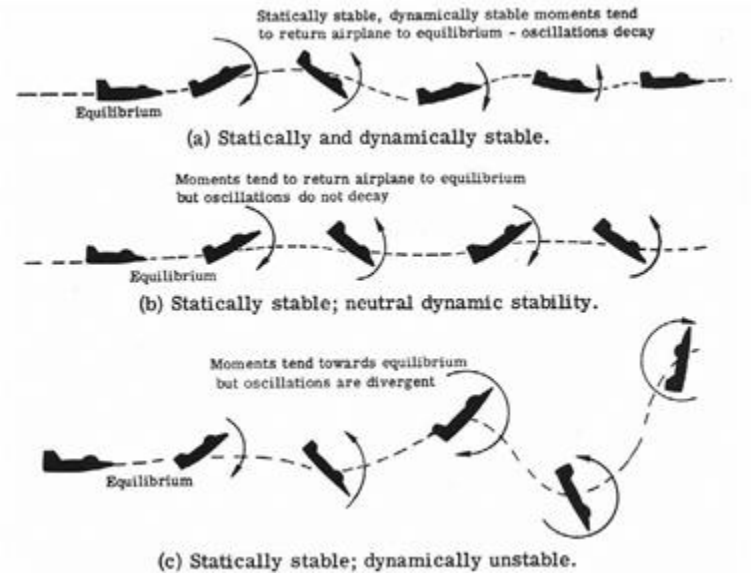


Fig. 1.2 Dynamic stability scheme

An aircraft that is both statically and dynamically stable can be flown hands off, unless the pilot desires to change the equilibrium condition of the aircraft.

1.1.3. Aircraft longitudinal modes

When the aircraft is not perturbed about the roll or yaw axis, only the longitudinal modes are required to describe the motion.

Oscillating motions can be described by two parameters, the period of time required for one complete oscillation, and the time required to damp to half-amplitude, or the time to double the amplitude for a dynamically unstable motion. The longitudinal motion consists of two distinct oscillations, a long-period oscillation called a phugoid mode and a short-period oscillation referred to as the short-period mode.

The long-period of phugoid mode involves a trade between kinetic and potential energy. In this mode, the aircraft, at nearly constant angle of attack, climbs and slows, then dives, losing altitude while picking up speed. The motion is usually of such a long period (about 93 seconds for a 747) that it need not be highly damped for piloted aircraft.

The short period mode involves rapid changes to the angle of attack and pitch attitude at roughly constant airspeed. This mode is usually highly damped; its

frequency and damping are very important in the assessment of aircraft handling. For a 747, the frequency of the short-period mode is about 7 seconds, while the time to halve the amplitude of a disturbance is only 1.86 seconds.

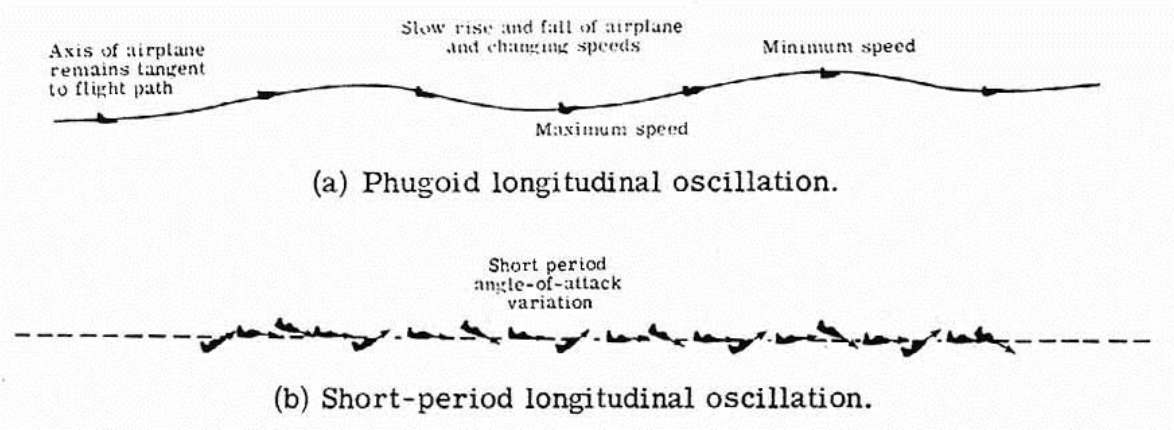


Fig. 1.3 Longitudinal modes

1.2. Aircraft Dynamics

The flight dynamics of an aircraft are described by its equations of motion (EOM). We are going to derive these equations in this section.

Conventional aircraft are subject to external forces and moments due to gravity, propulsion, and aerodynamics. The central modelling task is to determine expressions for these external forces and moments. A simple nonlinear model is obtained when the equations of motion are written in the vehicle body axis. Standard nomenclature is used for the following states: *x-y-z* body axis velocities (*u*, *v*, *w*); *x-y-z* body axis angular rates (*p*, *q*, *r*); and a standard 3-2-1 ordered rotation sequence of Euler angles (ϕ, θ, ψ). The *x-y-z* body-axis aerodynamic forces are denoted *X*, *Y*, and *Z*, and the corresponding moments are denoted *L*, *M*, and *N*.

For simplicity, gyroscopic effects of the rotating mass of the motor are assumed to be insignificant and the thrust T is assumed to act through the center of gravity and coincide with the body x axis.

To derive the equations of motion of an aircraft, we start by examining forces.

1.2.1. Force equations

Our starting point in this is Newton's second law:

$$F = m\frac{dV}{dt} = mA \tag{1.1}$$

There is one slight problem. The above equation holds for the R_i (inertial reference frame). But we usually work in the R_b(body fixed frame). So we need to convert it. To do this, we can use the relation:

$$A = \frac{dV}{dt}\Big|_i = \frac{dV}{dt}\Big|_b + \Omega_{bi} \times V \tag{1.2}$$

Substituting by $V = [u, v, w]^T$ and $\Omega_{bi} = [p, q, r]^T$ in the above equation we obtain:

$$F = mA = m\begin{bmatrix} \dot{u} + qw - rv \\ \dot{v} + ru - pw \\ \dot{w} + pv - qu \end{bmatrix} \tag{1.3}$$

We are going to continue with the forces our aircraft is subject to. There are two important kinds of forces: gravity and aerodynamic forces. The gravitational force F_g is in fact quite simple. It is given by:

$$F_g\big|_i = \begin{bmatrix} 0 \\ 0 \\ mg \end{bmatrix} \tag{1.4}$$

Where g is the gravitational acceleration. However we want the force in the $R_b$reference frame, for this reason we need to apply the rotation matrix between both reference frames. Following there is a schematic explanation about how to obtain the rotation matrix:

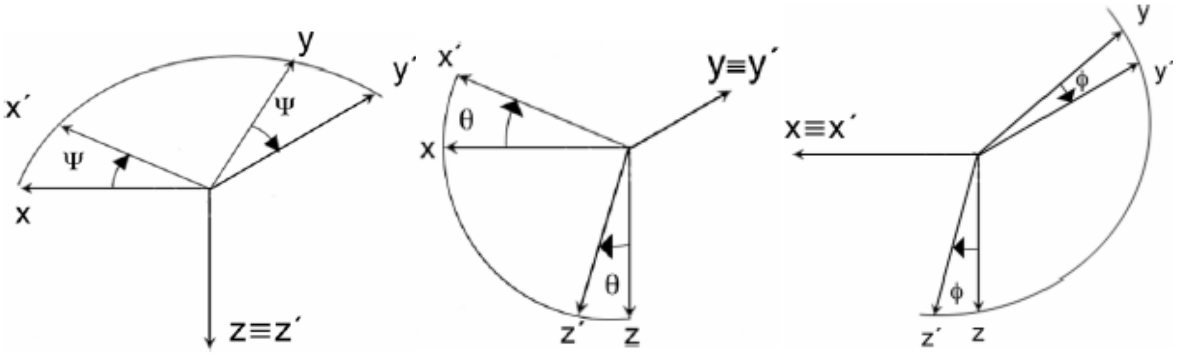


Fig. 1.4 Rotation axis scheme

And here we have the resulting rotation matrices:

$$R_3(\psi)=\begin{bmatrix}\cos\psi & \sin\psi & 0\\ -\sin\psi & \cos\psi & 0\\ 0 & 0 & 1\end{bmatrix} \quad R_2(\theta)=\begin{bmatrix}\cos\theta & 0 & -\sin\theta\\ 0 & 1 & 0\\ \sin\theta & 0 & \cos\theta\end{bmatrix} \quad R_1(\phi)=\begin{bmatrix}1 & 0 & 0\\ 0 & \cos\phi & \sin\phi\\ 0 & -\sin\phi & \cos\phi\end{bmatrix}$$

$$L_{bi}(\psi,\theta,\phi)=R_1(\phi)R_2(\theta)R_3(\psi) \tag{1.5}$$

$$L_{bi}=\begin{bmatrix}\cos\theta\cos\psi & \cos\theta\sin\psi & -\sin\theta\\ \sin\phi\sin\theta\cos\psi-\cos\phi\sin\psi & \sin\phi\sin\theta\sin\psi+\cos\phi\cos\psi & \sin\phi\cos\theta\\ \cos\phi\sin\theta\cos\psi+\sin\phi\sin\psi & \cos\phi\sin\theta\sin\psi-\sin\phi\cos\psi & \cos\phi\cos\theta\end{bmatrix} \tag{1.6}$$

Now that we have the rotation matrix to go from R_i to R_b, we will apply it:

$$F_g\big|_b=L_{bi}F_g\big|_i=mg\begin{bmatrix}-sin\theta\\ sin\phi cos\theta\\ cos\phi cos\theta\end{bmatrix} \tag{1.7}$$

For the aerodynamic forces F_{aero} we will just say that:

$$F_{aero}|_b=\begin{bmatrix}X\\ Y\\ Z\end{bmatrix} \tag{1.8}$$

Also we don't have to forget the thrust propulsion, which in this case is assumed to act through the center of gravity and coincide with the body axis:

$$F_{thrust}|_b=\begin{bmatrix}T\\ 0\\ 0\end{bmatrix} \tag{1.9}$$

By combining this knowledge with the equation of motion for forces, we finally find that:

$$m\begin{bmatrix}\dot{u}+qw-rv\\ \dot{v}+ru-pw\\ \dot{w}+pv-qu\end{bmatrix}=mg\begin{bmatrix}-sin\theta\\ sin\phi cos\theta\\ cos\phi cos\theta\end{bmatrix}+\begin{bmatrix}X\\ Y\\ Z\end{bmatrix}+\begin{bmatrix}T\\ 0\\ 0\end{bmatrix} \tag{1.10}$$

1.2.2. Moment equations

First we will examine angular momentum. The angular momentum of an aircraft B_G(with respect to the CG) is defined as:

$$B_G = I_G \Omega_{bi} \tag{1.11}$$

The parameter I_G is the inertia tensor, with respect to the CG. It is defined as:

$$I_G = \begin{bmatrix} I_x & -I_{xy} & -I_{xz} \\ -I_{xy} & I_y & -I_{yz} \\ -I_{xz} & -I_{yz} & I_z \end{bmatrix} \tag{1.12}$$

Now let's look at the moments. The moment acting on our aircraft, with respect to its CG, is given by:

$$M_G = \frac{dV}{dt}\Big|_i \tag{1.13}$$

The above relation only holds for inertial reference frames. However, we want to have the above relation in R_b. So we rewrite it to:

$$M_G = \frac{dB_G}{dt}\Big|_b + \Omega_{bi} \times B_G \tag{1.14}$$

Which can be rewritten as:

$$M_G = I_G \frac{d\Omega_{bi}}{dt}\Big|_b + \Omega_{bi} \times I_G \Omega_{bi} \tag{1.15}$$

Resulting in the following matrix:

$$M_G = \begin{bmatrix} I_x \dot{p} + (I_z - I_y)qr - I_{xz}(pq + \dot{r}) \\ I_y \dot{q} + (I_x - I_z)pr - I_{xz}(p^2 - r^2) \\ I_z \dot{r} + (I_y - I_z)pq - I_{xz}(qr - \dot{p}) \end{bmatrix} \tag{1.16}$$

Again, we can distinguish two types of moments, acting on our aircraft. There are moments caused by gravity, and moments caused by aerodynamic forces. Luckily, the moments caused by gravity are zero due to that the resultant gravitational force acts in the CG. So we only need to consider the moments caused by aerodynamic forces. We denote those as:

$$M_{G,aero}\big|_b = \begin{bmatrix} L \\ M \\ N \end{bmatrix} \quad \rightarrow \quad \begin{bmatrix} I_x \dot{p} + (I_z - I_y)qr - I_{xz}(pq + \dot{r}) \\ I_y \dot{q} + (I_x - I_z)pr - I_{xz}(p^2 - r^2) \\ I_z \dot{r} + (I_y - I_z)pq - I_{xz}(qr - \dot{p}) \end{bmatrix} = \begin{bmatrix} L \\ M \\ N \end{bmatrix} \tag{1.17}$$

Notice that I have assumed that the XZ-plane of the aircraft is a plane of symmetry. For this reason, $I_{xy} = I_{yz} = 0$.

1.2.3. Kinematic equations

The relationship between the body fixed angular velocity vector $\Omega_{bi} = [p, q, r]^T$ and the rate of change of the Euler angles $\dot{\eta} = \left[\dot{\phi}, \dot{\theta}, \dot{\psi}\right]^T$ can be determined by resolving the Euler rates into the body fixed frame:

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} \dot{\phi} \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\phi & sin\phi \\ 0 & -sin\phi & cos\phi \end{bmatrix} \begin{bmatrix} 0 \\ \dot{\theta} \\ 0 \end{bmatrix} + \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\phi & sin\phi \\ 0 & -sin\phi & cos\phi \end{bmatrix} \begin{bmatrix} cos\theta & 0 & -sin\theta \\ 0 & 1 & 0 \\ sin\theta & 0 & cos\theta \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ \dot{\psi} \end{bmatrix} \quad \textbf{(1.18)}$$

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & sin\phi tan\theta & cos\phi tan\theta \\ 0 & cos\phi & -sin\phi \\ 0 & sec\theta sin\phi & sec\theta cos\phi \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \quad \textbf{(1.19)}$$

1.2.4. Equations brief

In the following table we can see the resulting equations of motion:

Table 1.1. Equations of motion

Force Equations (extracted from Equation 1.10)	$\dot{u} = (rv - qw) + \frac{X}{m} - gsin\theta + T/m$	(1.20)
	$\dot{v} = (pw - ru) + \frac{Y}{m} + gcos\theta sin\phi$	**(1.21)**
	$\dot{w} = (qu - pv) + \frac{Z}{m} + gcos\theta cos\phi$	**(1.22)**
Moment Equations (extracted from Equation 1.17)	$\dot{p} - \left(\frac{I_{xz}}{I_x}\right)\dot{r} = -\frac{qr(I_z - I_y)}{I_x} + qp\left(\frac{I_{xz}}{I_x}\right) + L/I_x$	**(1.23)**

	$\dot{q} = -pr\frac{(I_x - I_z)}{I_y} - (p^2 - r^2)\frac{I_{xz}}{I_y} + M/I_y$	**(1.24)**
	$\dot{r} - \left(\frac{I_{xz}}{I_z}\right)\dot{p} = -pq\frac{(I_y - I_z)}{I_z} - qr\frac{I_{xz}}{I_z} + N/I_z$	**(1.25)**
Kinematic Equations (extracted from Equation 1.19)	$\dot{\phi} = p + tan\theta(qsin\phi + rcos\phi)$	**(1.26)**
	$\dot{\theta} = qcos\phi + rsin\phi$	**(1.27)**
	$\dot{\psi} = sec\theta(qsin\phi + rcos\phi)$	**(1.28)**

1.3. State-space modeling

State-space modeling is a mathematical characterization of the [coupled] aircraft dynamics in terms of ordinary linear differential equations with constant coefficients. The coefficients of the equations are the force and moment (stability and control) derivatives of the equations of motion.

For the problem of parameter identification of a linear system, the process is assumed to be governed by the linear matrix differential equations (state and observation equations) presented in equation 1.29 and equation 1.30. Using this system structure, the problem then becomes one of estimating the parameter values of the coefficient matrix A (the stability derivatives) and the coefficient matrix B (the control derivatives) that describe the aircraft's aerodynamic response to changes in the state variables, x, and the control variables, u.

$$\dot{x} = Ax + Bu \quad \textbf{(1.29)}$$

$$y = Cx \quad \textbf{(1.30)}$$

To simplify, the nonlinear model is linearized by assuming small perturbations from a steady, level trim condition and the thrust is assumed to be constant. Then, considering that the airplane is left/right symmetric, longitudinal dynamics can be decoupled from the lateral/directional dynamics and both can be studied separately.

In my case I'm going to study the longitudinal dynamics and they are described by the state $x_{lon} = (u, w, q, \theta)^T$, which corresponds to equations (1.20), (1.22), (1.24)

and (1.27). The forces X and Z, and the moment M, are assumed to be linear functions of u, w, q, and the elevator deflection δ_{elev}. Now, by applying:

$$A = \frac{\partial f}{\partial X}\Big|_{X = X_e;\ U = U_e} \tag{1.31}$$

$$B = \frac{\partial f}{\partial U}\Big|_{X = X_e;\ U = U_e} \tag{1.32}$$

Where the subscript "e" refers to the equilibrium points of the system which lead to steady, level trim condition, which means $u = \dot{v} = \dot{w} = \dot{p} = \dot{q} = \dot{r} = p = q = r = \dot{\phi} = \dot{\theta} = \dot{\psi} = 0$. It results the following system:

$$\dot{x}_{lon} = A_{lon}x_{lon} + B_{lon}\delta_{elev} \tag{1.33}$$

$$A_{lon} = \begin{pmatrix} X_u & X_w & X_q - W_e & -g\ cos\theta_e \\ Z_u & Z_w & Z_q + U_e & -g\ sin\theta_e \\ M_u & M_w & M_q & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \quad B_{lon} = \begin{pmatrix} X_{\delta_{elev}} \\ Z_{\delta_{elev}} \\ M_{\delta_{elev}} \\ 0 \end{pmatrix} \tag{1.34}$$

The next step is to identify the terms of these matrices. To do it, the Department of Aerospace Engineering and Mechanics has applied the Frequency Domain System Identification process. This process is explained briefly in the following section.

1.4. Frequency domain System identification

First of all a baseline model should be created in order to have an idea of the system so we could design appropriate flight experiments to apply the method. In this case a baseline model of the Ultra Stick 25e flight dynamics is generated using aerodynamic data from two similar airframes:

- Ultra Stick Mini. Control derivatives and stability derivatives associated with the body velocities are estimated from wind-tunnel tests performed with an Ultra Stick Mini. This airframe is smaller than the 25e and fits in the wind tunnel available at the University of Minnesota (where the system identification process has been). The 25e and the Mini have similar aerodynamics but are not exact geometric scales of each other.

- Ultra Stick 120. Stability derivatives associated with the angular rates are taken from an aerodynamic model for the Ultra Stick 120. This airframe is larger than the 25e, has similar aerodynamics, but is not an exact geometric scale.

Although these approximations in the aerodynamics, to simplify the longitudinal dynamics are decoupled further into the phugoid and the short-period modes. The baseline parameters of the phugoid mode are kept and the frequency domain system identification process is focused on the research of the short-period mode parameters. The reason of this step is that the phugoid mode is typically very slow and lightly damped, and dominates the response in u, θ, and a_x. For control applications accurate knowledge of the phugoid mode is not crucial due to the low frequency of the oscillation, which is compensated for with feedback control. On the contrary, the short-period mode is typically fast, moderately damped, and dominates the response in w, q, and a_z. Moreover, stability and performance characteristics also depend primarily on the short-period mode. So, for this reason, system identification is applied to the short-period model shown in the following system, where the state vector is $x_{lon} = [w, q]^T$:

$$A_{lon} = \begin{bmatrix} Z_w & Z_q + U_e \\ M_w & M_q \end{bmatrix} \quad B_{lon} = \begin{bmatrix} Z_{\delta_{elev}} \\ M_{\delta_{elev}} \end{bmatrix} \tag{1.35}$$

Then, the baseline model for the Ultra Stick 25e is used to characterize definitely the phugoid mode and also as a guide to design flight experiments. So the following step is to do these flight experiments and obtain input-output flight data in order to start the frequency domain system identification process, which is comprised of two steps:

1. To extract frequency responses using spectral quantities computed from input-output flight data.
2. To fit the linear state-space models to the extracted frequency responses. Parameters in the linear models are identified through a nonlinear optimization that minimizes the fitting error in the frequency domain.

1.5. Control systems basis

1.5.1. Main components

A block diagram of a basic feedback loop is shown in Figure 1.5. The system loop is composed of two components, the process *P* and the controller. The controller has two blocks: the feedback block *C* and the feedforward block *F*. There are two disturbances acting on the process, the load disturbance *d* and the measurement noise *n*. The load disturbance represents disturbances that drive the process away from its desired behavior. The process variable *x* is the real physical variable that we want to control. Control is based on the measured signal *y*, where the measurements are corrupted by measurement noise *n*. Information about the process variable *x* is thus distorted by the measurement noise. The process is influenced by the controller via the control variable *u*. The process is thus a system

with three inputs and one output. The inputs are: the control variable *u*, the load disturbance *d* and the measurement noise *n*. The output is the measured signal. The controller is a system with two inputs and one output. The inputs are the measured signal *y* and the reference signal *r* and the output is the control signal *u*. Note that the control signal *u* is an input to the process and the output of the controller and that the measured signal is the output of the process and an input to the controller. In Figure 1.5 the load disturbance was assumed to act on the process input. This is a simplification, in reality the disturbance can enter the process in many different ways. To avoid making the presentation unnecessarily complicated we will use the simple representation in Figure 1.9. This captures the essence and it can easily be modified if it is known precisely how disturbances enter the system.

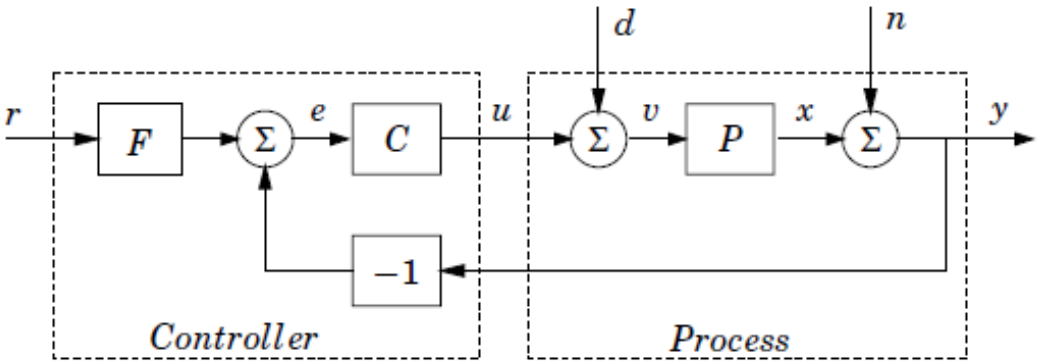


Fig. 1.5 Block diagram of a basic feedback loop

1.5.2. Design issues

Many issues have to be considered in analysis and design of control systems. Basic requirements are:

- Stability
- Ability to follow reference signals
- Reduction of effects of load disturbances
- Reduction of effects of measurement noise
- Reduction of effects of model uncertainties

The possibility of instabilities is the primary drawback of feedback. Avoiding instability is thus a primary goal. It is also desirable that the process variable follows the reference signal faithfully. The system should also be able to reduce the effect of load disturbances. Measurement noise is injected into the system by the feedback. This is unavoidable but it is essential that not too much noise is injected. It must also be considered that the models used to design the control systems are

inaccurate. The properties of the process may also change. The control system should be able to cope with moderate changes.

The goal of the control system varies with the application. For example, in process control the major emphasis is often on attenuation of load disturbances, while the ability to follow reference signals is the primary concern in motion control systems.

2. PROBLEM FORMULATION

2.1. Control Systems Design

Control is used to modify the behaviour of a system so it behaves in a specific desirable way over time. In this case, for example, we want the aircraft to follow a desired altitude, heading, and velocity profile independent of wind gusts. This is being accomplished today by automatic control systems designed to act without human intervention.

To design a controller that makes a system behave in a desirable manner, we need a way to predict the behaviour of the quantities of interest over time, specifically how they change in response to different inputs. The role of control theory is to help us gain insight on how and why feedback control systems work and how to systematically deal with various design and analysis issues. Specifically, the following issues are of both practical importance and theoretical interest:

1. Stability and stability margins of closed-loop systems.
2. How fast and smooth the error between the output and the set point is driven to zero.
3. How well the control system handles unexpected external disturbances, sensor noises, and internal dynamic changes.

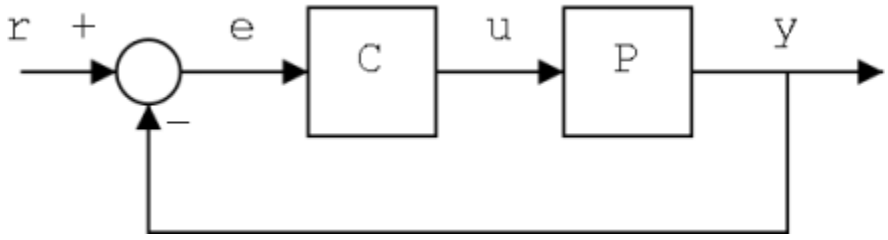


Fig. 2.1 Basic feedback structure

In order to evaluate the control system performance we will study the step response, focusing on its transient response. To characterize the transient response we will analyze the settling time which is the time it takes for the output to settle within 2 percent of its final value; the percent overshoot, which is how much the output exceeds the set-point *r* percentage wise during the period that *y* converges to *r*; and the steady-state error, which refers to the difference, if any, between *y* and *r* as *y* reaches its steady-state value.

Then, our objective will be to find the best controller for the Ultra Stick 25e in order that the system could supply the fastest and smoothest signal with the minimum steady-state error while it is robust enough to resist unexpected disturbances. To

do this, first we will find the best solution applying different methods: classical control, optimal control, fuzzy control, robust control. And after, we will compare them and choose the best option.

2.2. Autopilot types

An autopilot is a system used to control the trajectory of a vehicle without constant 'hands-on' control by a human operator being required. Autopilots do not replace a human operator, but assist them in controlling the vehicle, allowing them to focus on broader aspects of operation, such as monitoring the trajectory, weather and systems. Autopilots have evolved significantly over time, from early autopilots that merely held an attitude to modern autopilots capable of performing automated landings under the supervision of a pilot.

In order to ensure that there is no doubt about how an autopilot works we will exemplify its function. If during a cruise flight the aircraft flies in Mach hold mode, this means that the aircraft flies at constant Mach speed though automatic control of pitch angle by the elevator. When the aircraft flies, fuel is burned and its weight decreases by the time, so its speed tends to increase. Then the work of the autopilot is to detect the speed increase and to correct it by the elevator. The effect will be that the plane will rise slowly as the fuel is burning, resulting in a more efficient flight.

In this project we will only study the basic configurations for the longitudinal autopilot:

- Maintain longitudinal velocity constant
- Maintain vertical velocity constant
- Maintain pitch rate constant
- Maintain pitch constant

2.3. The plant: Ultra Stick 25e State-space model

Let’s take a look to our plant, the Ultra Stick 25e state-space model obtained by a frequency domain system identification process (see section 1.4.) realized in the University of Minnesota:

$$A_{lon} = \begin{pmatrix} -0.38 & 0.60 & -0.36 & -9.80 \\ -0.98 & -10.65 & 16.74 & -0.21 \\ 0.18 & -5.39 & -16.55 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \quad B_{lon} = \begin{pmatrix} -0.36 \\ -3.62 \\ -141.57 \\ 0 \end{pmatrix} \qquad \textbf{(2.1)}$$

First we will check the stability of the plant by examining the eigenvalues of the matrix A_{lon}. The eigenvalues are the values λ for which the following equation is true:

$$|(\lambda I - A)| = 0 \qquad \textbf{(2.2)}$$

And a system is stable as long as all the eigenvalues of A have negative real part. These are our eigenvalues:

Table 2.1. Eigenvalues of $\boldsymbol{A_{lon}}$

-13.5925 + 9.0398i
-13.5925 - 9.0398i
-0.1975 + 0.4735i
-0.1975 - 0.4735i

As we can observe in the Table 2.1. above, all the eigenvalues have negative real part, then the system is stable.

The following step is to ensure that the plant is fully controllable and fully observable, which means if it is possible to steer the states from any initial value to any final value within some finite time (controllable), and if by observing the output and the input over a finite period of time it is possible to deduce the value of the state vector of the system (observable). In other words, as controllability establishes that an input is capable to bring any initial state to any desired final state, observability determines that knowing an output trajectory gives enough information to predict the initial state of the system. So they are important characteristics that we should know about our plant.

The procedure is simple:

- A system (or a pair (A, B)) is controllable if and only if the controllability matrix V = [B, AB,…, A^{n-1}B] has full (row) rank n.
- A system (or a pair (A, C)) is observable if and only if the observability matrix O = $[C, CA, \dots, CA^{n-1}]^T$ has full (column) rank n.

In this case we just had to see if both V and O matrices have rank 4, and by an easy matlab code (see Appendix A.1.) I have checked that the system is fully controllable and observable. I have to say also that I have supposed that the output matrix (C) was an identity matrix of 4x4, which means that there is no correlation between channels.

3. CLASSICAL CONTROL

3.1. Classical control and PID definition

The most common classical control method is to add a PID controller. A PID controller calculates an error value as the difference between a measured process variable and a desired setpoint. The controller attempts to minimize the error by adjusting the process through use of a manipulated variable.

Basically it consists in a proportional gain (K_P), an integral gain (K_I) and a derivative gain (K_D). Hence the name PID. The proportional gain is a pure gain adjustment acting on the error signal. The error signal is the difference between the desired position and the actual position of the plant. The integral gain adjusts the accuracy of the plant, and the derivative gain adjusts the damping of the plant.

Definig *m(t)* as the controller output and *e(t)* as the error signal (difference between input and output signals), the final form of the PID algorithm is:

$$m(t) = K_p e(t) + K_I \int_0^t e(\tau) d\tau + K_D \frac{de(t)}{dt} \qquad \textbf{(3.1)}$$

And if we apply the Laplace transform to the equation above, we obtain the PID transfer function:

$$PID(s) = K_P + \frac{K_I}{s} + K_D s \qquad \textbf{(3.2)}$$

These are the basis of the PID controller and in the following section we will apply them to our plant.

3.2. Solution and results

This is the scheme of my proposal for the Classical Control solution:

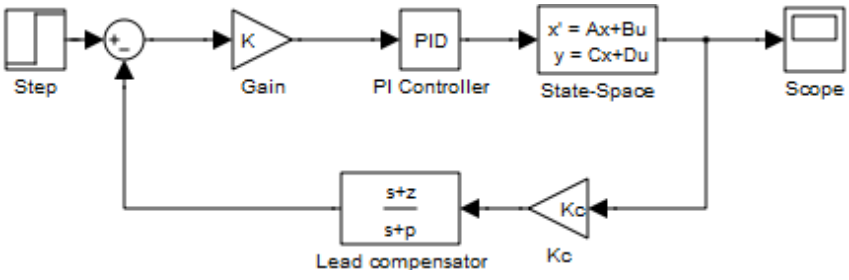


Fig. 3.1. Classical Control solution

As you can see above in *Fig 3.1.* I have used a PI controller. The proportional part (K_P) of the controller increases the speed of the control system response and also decreases the steady-state error. On the other hand the integral part $(\frac{K_I}{s})$ eliminates the steady-state error. I have omitted the derivative part of the controller because it is not feasible in practice, so instead I have add a lead compensator $\left(\frac{S+Z}{S+P}K_C, with\ p > z\right)$, which has the same effect on the plant: it makes the system faster, smoother and with less overshoot.

Also I have add a gain (K), it is just a tool to help me to regulate the gains of the PI controller and also to inverse the signal when need it.

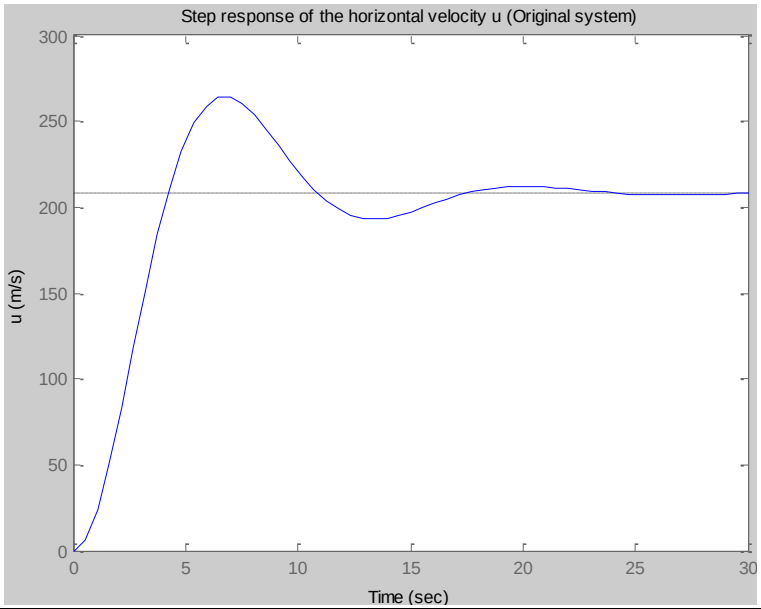
These parameters $(K, K_P, K_I, K_C, z \text{ and } p)$ are dependent on each other and they have been selected by doing some trial and error until I have obtained the best response, a trade-off of these qualities: low overshoot, smooth and fast signal behaviour, and low steady-state error. These are the selected values:

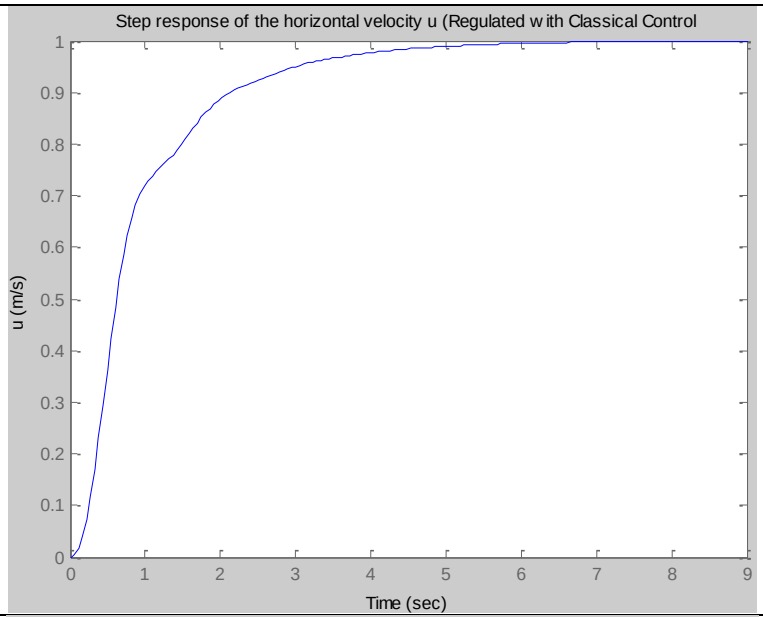
Table 3.1. Numerical values of the proposed Classical Control solution

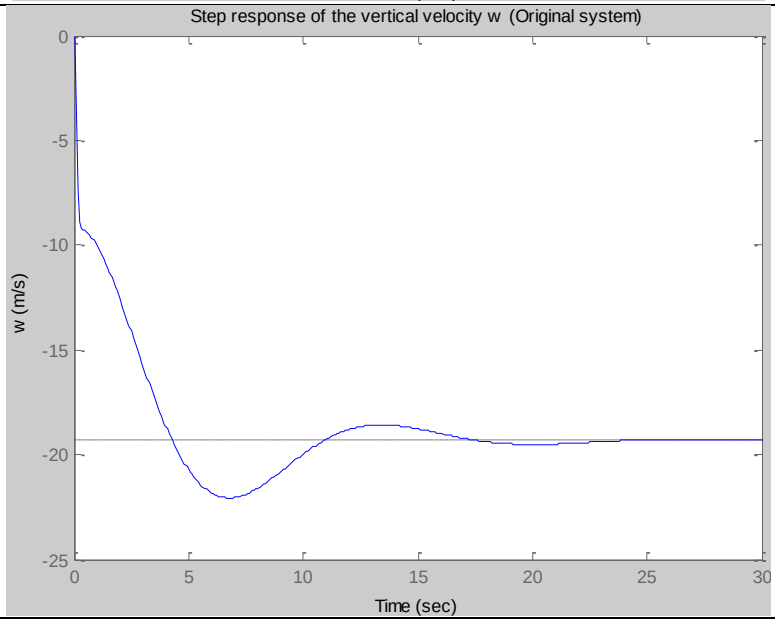
State Variables	K	K_P	K_I	K_C	z	p
u	0.1	1	1	14	1	14
w	-2	5	3	7	1	7
q	-30	8	5	7	1	7
θ	-3	8	7	20	1	20

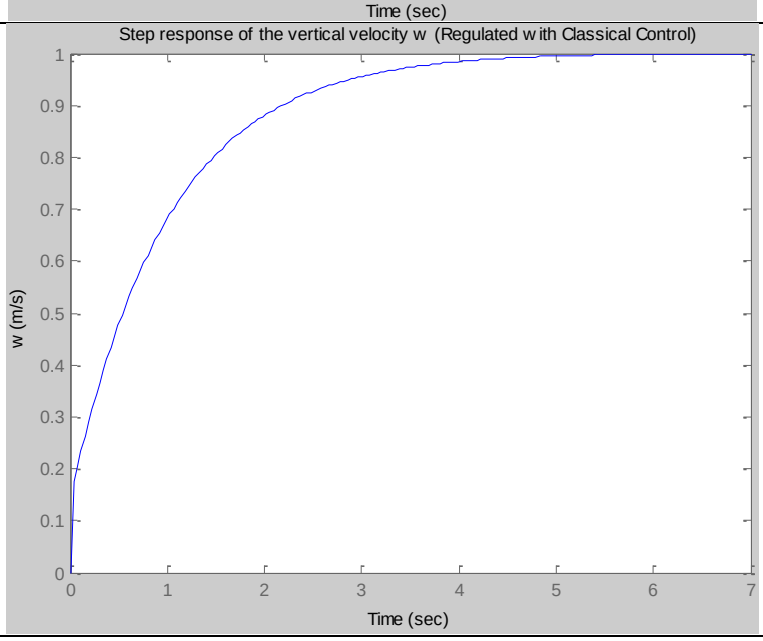
In the graphs of the Table 3.2. we can see the improvement achieved with the Classical Control application comparing the step responses of the original system against the step responses of the regulated one.

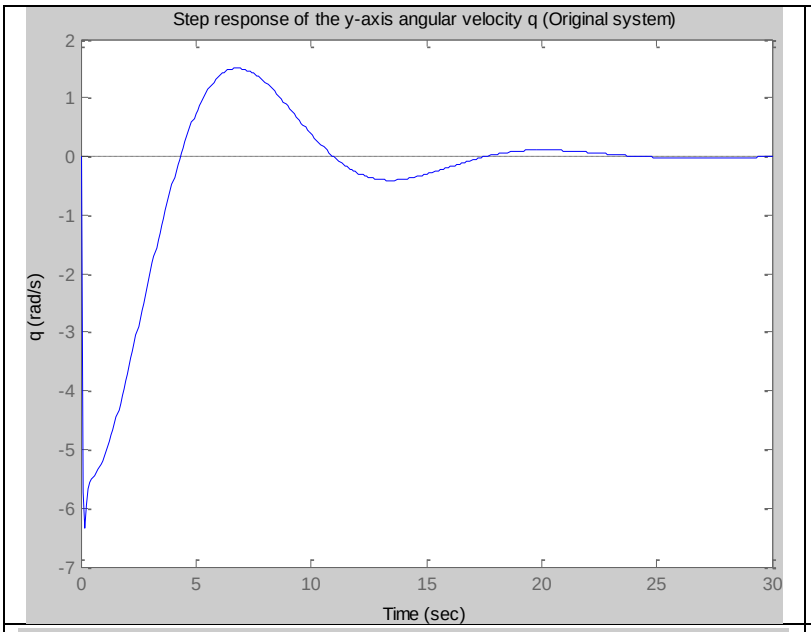
Table 3.2. Step response comparison between original and regulated system

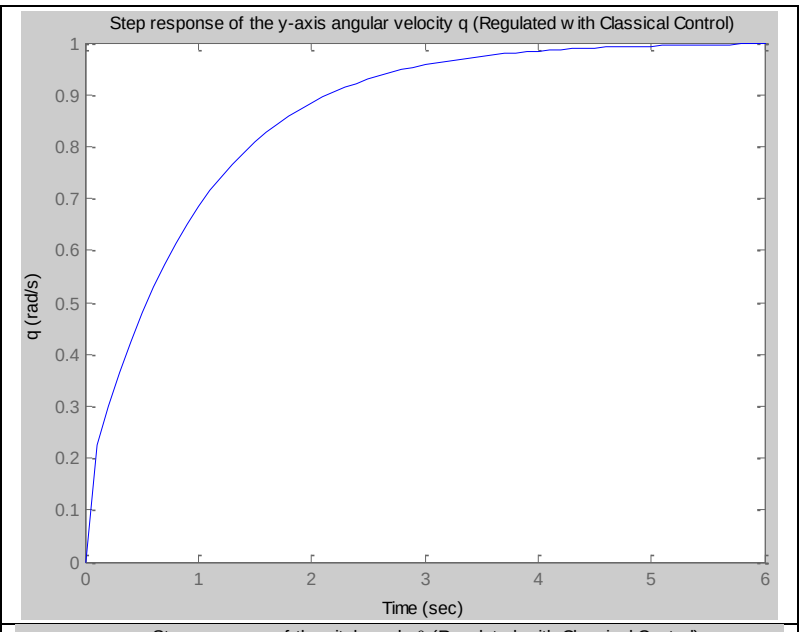

Step response of the horizontal velocity u (Original system)
u (m/s)
Time (sec)

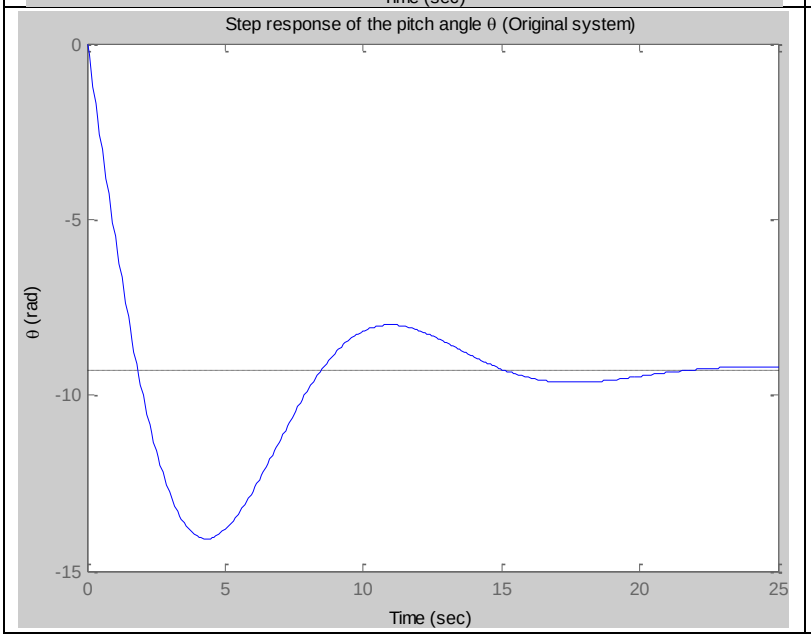

Step response of the horizontal velocity u (Regulated with Classical Control
u (m/s)
Time (sec)

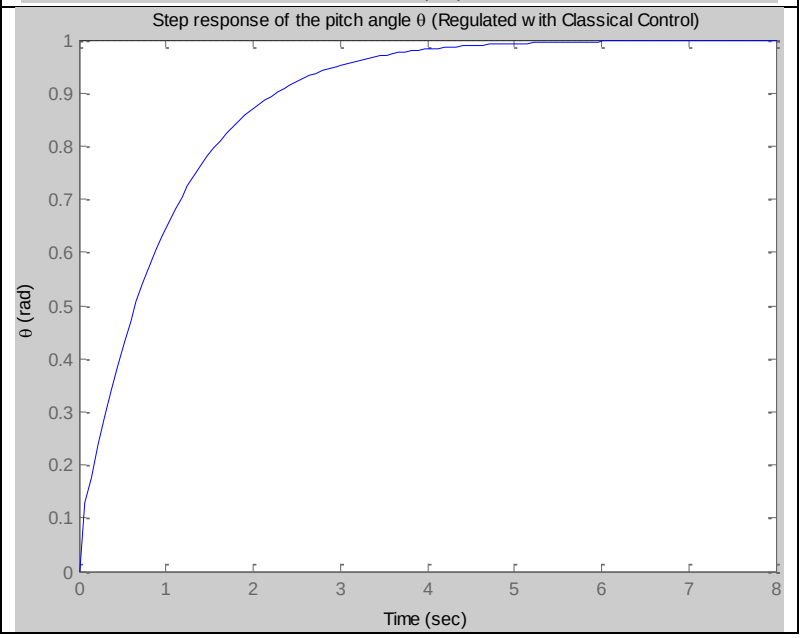

Step response of the vertical velocity w (Original system)
w (m/s)
Time (sec)


Step response of the vertical velocity w (Regulated with Classical Control)
w (m/s)
Time (sec)









And to remark how good is the controller design I join also the numerical description of the transient responses.

Table 3.3. Steady state error and settling time comparison

State Variables	Steady state error (%)		Settling time (sec)	
	Original System	Regulated System	Original System	Regulated System
u	-20680.1578	0.0837	16.47	4.18
w	2026.2929	0.1102	15.72	3.77

q	102.5332	0.3984	16.44	3.74
θ	1019.8033	0.2843	19.88	3.87

The steady state error has been almost eliminated and the settling time has decrease an average of 75%. Moreover we can see in the table below that we have achieved also a good robustness. In classical controller design methods, plant uncertainties are taken into account by stability margins relating to amplitude and phase, and these margins are really broad except for the state variable *u*.

Table 3.4. Gain and phase margins of the classical control design

State Variables	**Robustness (Classic Control)**	
	Gain Margin (dB)	Phase Margin (º)
u	12.9 at 6.57 rad/s	-180 at 0 rad/s
w	Inf	-180 at 0 rad/s
q	Inf	Inf
θ	Inf	-180

4. OPTIMAL CONTROL

4.1. Optimal control and LQR definitions

Optimal control deals with the problem of finding a control law for a given system such that a certain optimality criterion is achieved. A control problem includes a cost function that is a function of state and control variables. An optimal control is a set of differential equations describing the paths of the control variables that minimize this cost function.

A particular Optimal Control method is the Linear Quadratic Regulator (LQR). LQR is a control scheme that provides the best possible performance with respect to some given measure of performance. The LQR design problem is to design a state feedback controller K such that the objective function *J* (Eq. 4.2) is minimized. In this method a feedback gain matrix is designed which minimizes the objective function in order to achieve some compromise between the use of control effort, the magnitude, and the speed of response that will guarantee a stable system.

For a continuous-time linear system described by

$$\dot{x} = Ax + Bu \tag{4.1}$$

With a cost function defined as

$$J = \int_0^\infty (x^T Q x + u^T R u) dt \tag{4.2}$$

Where Q and R are the weight matrices, Q is related to the state constraints and it is required to be positive definite or positive semi-definite symmetry matrix; R is related to the control constraints and it is required to be positive definite symmetry matrix. One practical method is to Q and R to be diagonal matrix. The value of the elements in Q and R is related to its contribution to the cost function J.

The feedback control law that minimizes the value of the cost is

$$u = -Kx \tag{4.3}$$

K is given by

$$K = R^{-1} B^T P \tag{4.4}$$

And P can be found by solving the continuous time algebraic Riccati equation:

$$A^T P + PA - PNR^{-1} B^T P + Q = 0 \tag{4.5}$$

4.2. Solution and results

The rudimentary work explained above to compute the optimal gain K will be done by a Matlab programme. Its code is available in the *Appendix A.3.*

So our main task here is to define the cost function, which is defined by the weight matrices Q and R. We will place constraints on all the state variables (u, w, q, θ), and also to the elevator deflection (δ_{elev}), following the particular performance of the Ultra Stick 25e.

The unique information about the aircraft performance is that with the throttle set around 70% its airspeed is closed to 19m/s, and that all control surfaces are actuated via electric servos with a maximum deflection of 25deg in each direction. With this information I have considered the following cost function:

$$J = \int_0^\infty \left[\left(\frac{\Delta u}{\Delta u_{max}}\right)^2 + \left(\frac{\Delta w}{\Delta w_{max}}\right)^2 + \left(\frac{\Delta q}{\Delta q_{max}}\right)^2 + \left(\frac{\Delta \theta}{\Delta \theta_{max}}\right)^2 + \left(\frac{\Delta \delta_{elev}}{\Delta \delta_{elev_{max}}}\right)^2 \right] dt \qquad \textbf{(4.6)}$$

With

$$\begin{cases} \Delta u_{max} = 25 m/s \\ \Delta w_{max} = 12 m/s \\ \Delta q_{max} = 30 deg/s \\ \Delta \theta_{max} = 30 deg \\ \Delta \delta_{elev_{max}} = 25 deg \end{cases}$$

Thus, the Q and R matrices are

$$Q = \begin{bmatrix} \frac{1}{\Delta u_{max}^2} & 0 & 0 & 0 \\ 0 & \frac{1}{\Delta w_{max}^2} & 0 & 0 \\ 0 & 0 & \frac{1}{\Delta q_{max}^2} & 0 \\ 0 & 0 & 0 & \frac{1}{\Delta \theta_{max}^2} \end{bmatrix} \qquad R = \frac{1}{\Delta \delta_{elev_{max}}^2} \qquad \textbf{(4.7)}$$

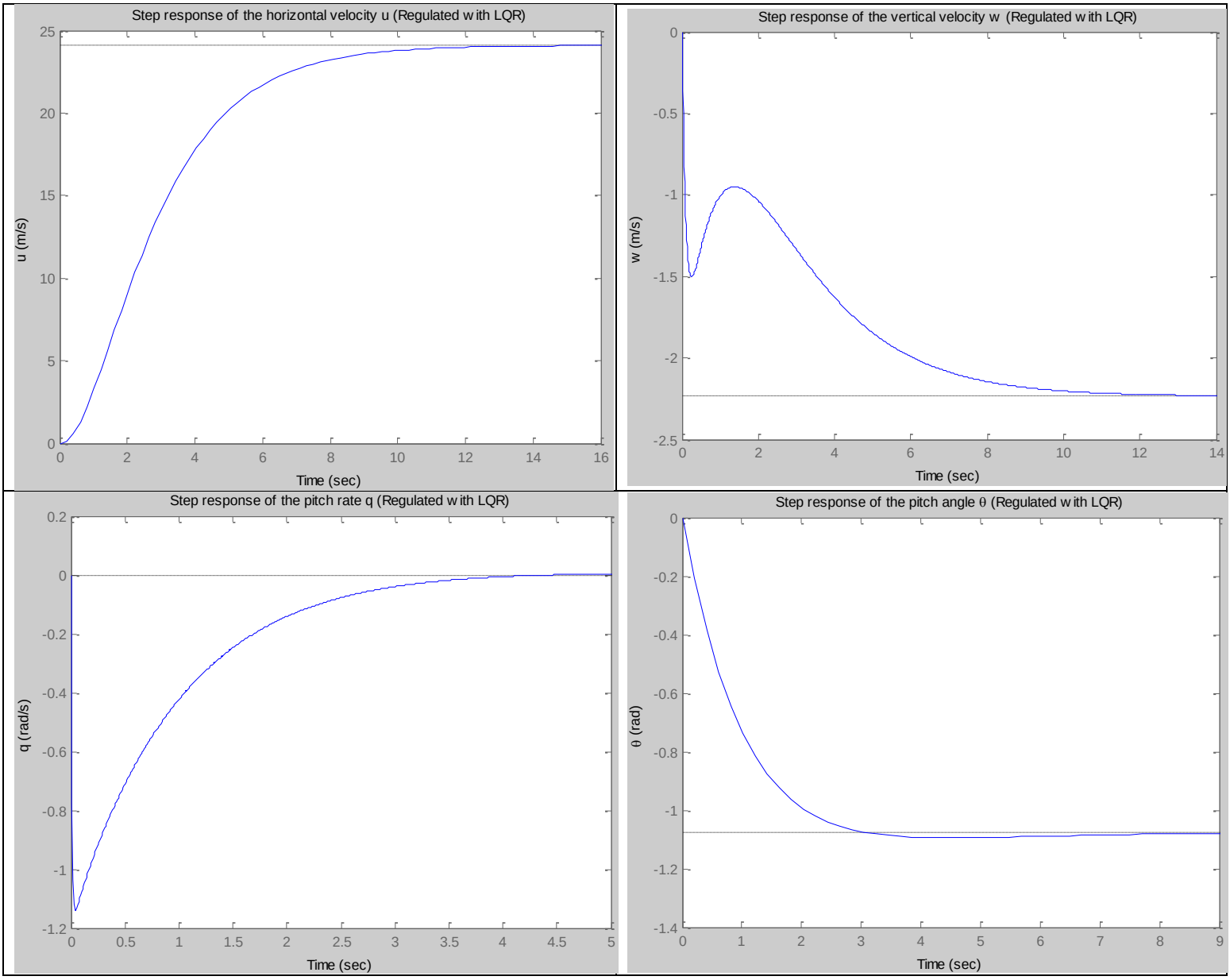
Now by applying the linear quadratic regulator method we obtain the gain:

$$K = [0.0011 \;\; 0.0287 \;\; -0.7276 \;\; -0.8574]$$

The new control law is therefore:

$$\Delta \delta_e = 0.0011 \Delta u(t) + 0.0287 \, \Delta w(t) - 0.7276 \, \Delta q(t) - 0.8574 \, \Delta \theta(t)$$

Table 4.1. Step response of the system with LQR



Examination of the graphs above demonstrates that there is a large steady-state error. One way to correct this is by introducing a precompensator (Nbar) to scale the overall output.

Unlike other design methods, the full-state feedback system does not compare the output to the reference; instead, it compares all states multiplied by the control matrix (**K x**) to the reference (see Figure 4.1.).

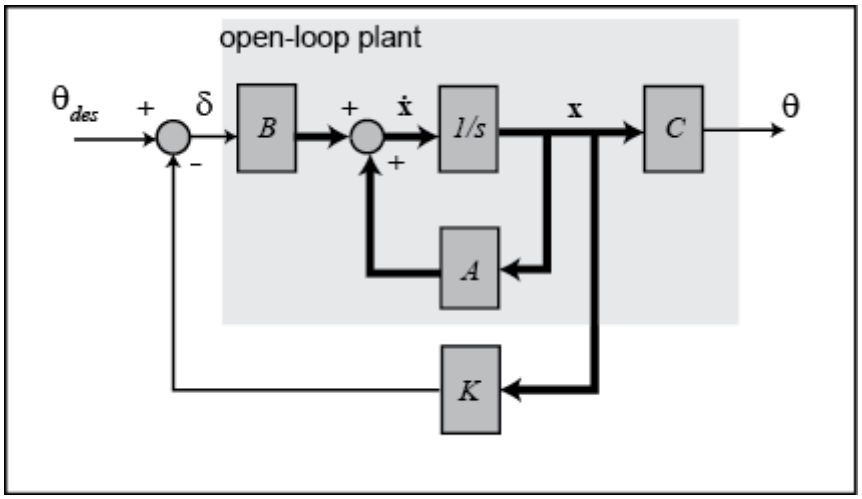


Fig. 4.1. Schematic of a full-state feedback control system (with D = 0)

Thus, we should not expect the output to equal the commanded reference. To obtain the desired output, we can scale the reference input so that the output equals the reference in steady state. This can be done by introducing a precompensator scaling factor called Nbar. The basic schematic of our state-feedback system with scaling factor (Nbar) is shown in Figure 4.2.

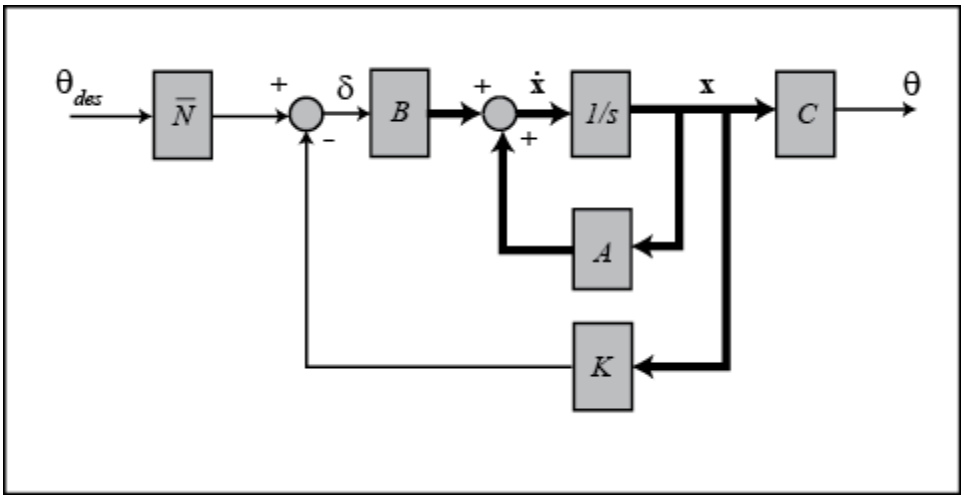


Fig. 4.2. Basic schematic of our state-feedback system with scaling factor (Nbar)

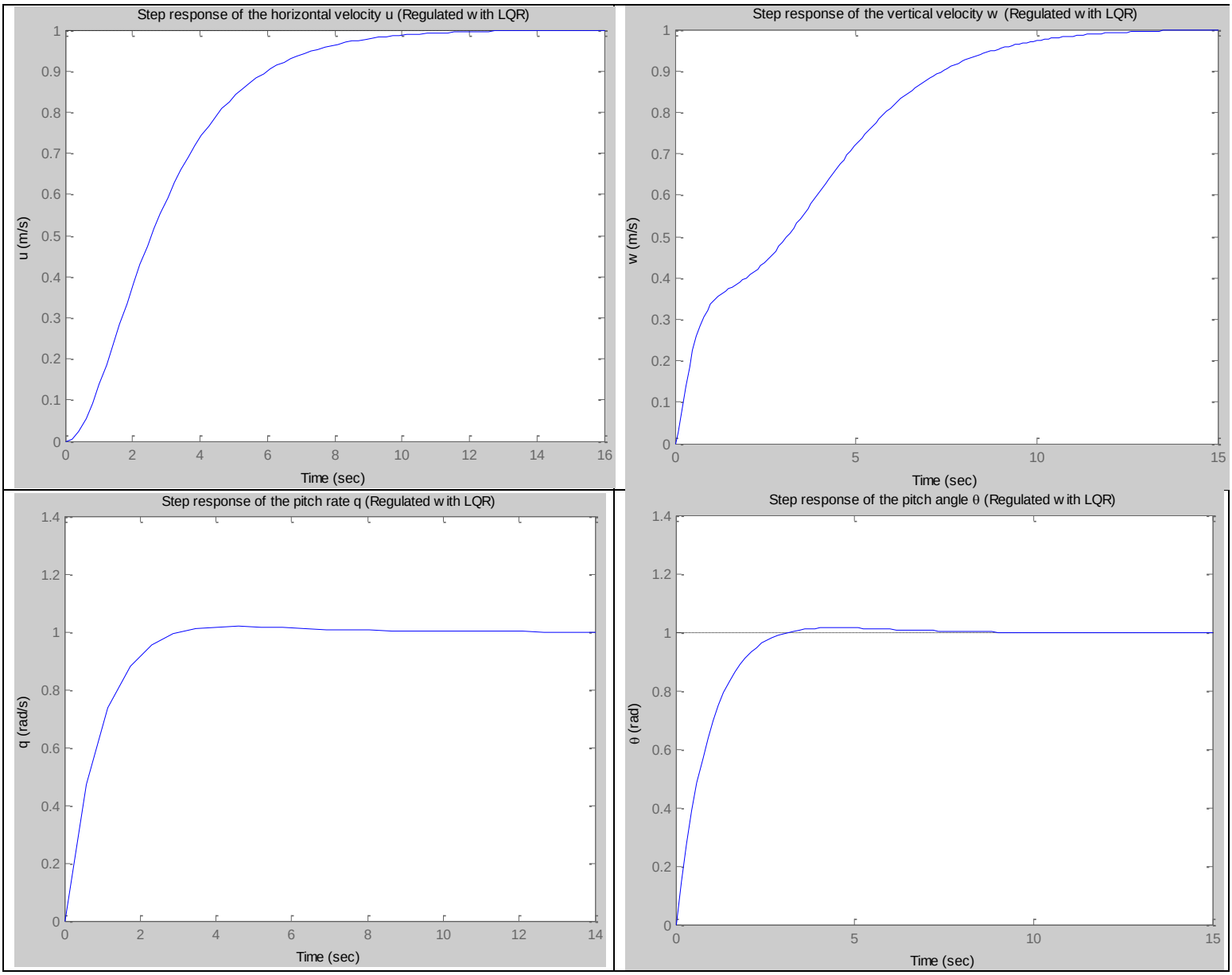
We can find Nbar from the given MATLAB function `rscale.m`. This function will find the scale factor for a full-state feedback system to eliminate the steady-state error. Your code will need to be modified as follows:

```
Nbar = rscale(A,B,C,D,K);  and sys_cl = ss(A-B*K,B*Nbar,C,D);
```

Being `sys_cl` the closed loop system.

The values of the Nbar precompensators are 0.0415 for *u*, -0.4474 for *w* and -0.9299 for θ. For the pitch rate *q* I have had to use another method because of its tendency to go to zero. Instead of the Nbar I have added an integrator and a gain before the closed loop system. As gain I've just take the final value of the step response in order to stabilize the step response of the pitch rate at 1rad/s. Also, I have added a low pass filter for *w* in order to have a step response without oscillations. See Table 4.2. for the corresponding step responses.

Table 4.2. Step response of the system with corrected LQR



4.3. Classical control Vs Optimal control

For our plant to use classical control is best than to use optimal control. Optimal control provides much slower responses in *u* and *w*, and slightly faster responses in *q* and *θ*. The unique improvement that seems to have against the classical control is the lower steady state error, which is due to the precompensator Nbar. However, this precompensator is calculated based on the model of the plant and it is located outside of the feedback loop. Therefore, if there are errors in the model (or unknown disturbances) the precompensator will not correct for them and there will be steady-state error.

In the Table 4.3. and Table 4.4. it can be seen the numerical values of the performance comparison between both results.

Table 4.3. Steady state error and settling time comparison

State Variables	Steady state error (%)		Settling time (sec)	
	Classical Control	LQR	Classical Control	LQR
u	0.0837	0	4.18	9.15
w	0.1102	0	3.77	10.74
q	0.3984	0	3.74	2.67
θ	0.1333	0	3.89	2.62

Table 4.4. Gain and phase margins comparison

State Variables	Gain Margin (dB)		Phase Margin (º)	
	Classical control	Optimal control	Classical control	Optimal control
u	12.9 at 6.57 rad/s	34 at 4.52 rad/s	-180 at 0 rad/s	-180 at 0 rad/s
w	Inf	47.1 at 43.6 rad/s	-180 at 0 rad/s	-180 at 0 rad/s
q	Inf	Inf	Inf	168 at 0.255 rad/s
θ	Inf	Inf	-180 at 0 rad/s	168 at 0.265 rad/s

5. ROBUST CONTROL

5.1. Robust control introduction

Robust control is a branch of control theory which explicitly deals with uncertainty. Robust control methods are designed to function properly provided that uncertain parameters or disturbances are found within some (typically compact) set. Robust methods aim to achieve robust performance and stability in the presence of bounded modelling errors.

Although there are several robust control methods, in this section we will just focus on two of them: Linear-Quadratic Gaussian (LQG) control and H-infinity loop shaping.

5.2. Linear-Quadratic-Gaussian (LQG) control

5.2.1. Definition

Linear-quadratic-Gaussian (LQG) control is a modern state-space technique for designing optimal dynamic regulators and servo controllers with integral action (also known as set point trackers). This technique allows you to trade off regulation/tracker performance and control effort, and to take into account process disturbances and measurement noise.

To design LQG regulators and set point trackers, you perform the following steps:

1. Construct the LQ-optimal gain.
2. Construct a Kalman filter (state estimator), which is an algorithm that estimates the state of a system from measured data.
3. Form the LQG design by connecting the LQ-optimal gain and the Kalman filter.

To clarify the concepts of regulator and servo controller I have to say that the main difference between them is that a regulator only filters the noise of the system and regulates its output *y* around zero, while a servo controller, in addition, sets the output *y* to a reference value. So what we want is a servo controller, a system that ensures that the output *y* tracks the reference command while rejecting process disturbances and measurement noise.

The Kalman filter works in a two-step process. In the prediction step, the Kalman filter produces estimates of the current state variables, along with their uncertainties. Once the outcome of the next measurement (corrupted with some amount of error, including random noise) is observed, these estimates are updated using a weighted average, with more weight being given to estimates with higher

certainty. Because of the algorithm's recursive nature, it can run in real time using only the present input measurements and the previously calculated state and its uncertainty matrix; no additional past information is required.

5.2.2. Solution and results

This is the proposed solution, a servo controller with integral action:

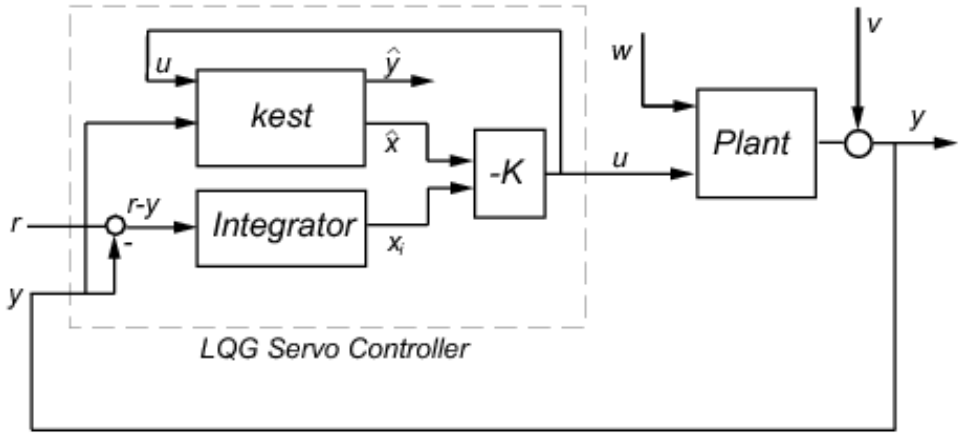


Fig. 5.1. LQG Design of Servo Controller with Integral Action

The plant in the previous figure is subject to disturbances or process noise *w* and is driven by controls *u*. The servo controller relies on the noisy measurements *y* to generate these controls; where *v* is the measurement noise. The plant state and measurement equations are of the form:

$$\dot{x} = Ax + Bu + Gw \quad \textbf{(5.1)}$$

$$y = Cx + Hw + v \quad \textbf{(5.2)}$$

and both *w* and *v* are modeled as white noise.

To adapt our plant to this scheme we need to define the matrices G and H. For the matrix G, which is a vector in this case, we will make it with the gains of the plant. To obtain such gains we just have to take the final values of the step responses of the system in open loop (see first column of Table 3.2.). And to simplify we set H with a null value. The resulting matrices are:

$$G = \begin{pmatrix} 200 \\ -20 \\ 0 \\ -10 \end{pmatrix} \quad \textbf{(5.3)}$$

$$H = 0 \quad \textbf{(5.4)}$$

Our aircraft, the Ultra Stick 25e, is instrumented with an IMU that provides measurements of angular rates and translational accelerations. Three gyroscopes form the angular rate sensor, and three accelerometers form the acceleration sensor. A ground test, with the throttle set to around 70%, resulted in a noise amplitude approximately 2 deg/s in each angular rate channel, and 0.5 m/s in each acceleration channel. So we will consider these values as measurement noise and, to make it easy, we will consider unitary values for the process noise. Moreover, as an assumption, in order to simplify the Kalman filter construction, we will let the noise be uncorrelated between the different states. With this knowledge we will assume the following process noise (Qn) and measurement noise (Rn):

Table 5.1. Noise data for the Kalman filter

State Variables	Qn	Rn
u	1 m/s	0.5 m/s
w	1 m/s	0.5 m/s
q	1 rad/s	2 deg/s = 0.035 rad/s
θ	1 rad	2 deg = 0.035 rad

Now that we know all the data needed we just have to follow the following steps:

1. Construct the LQ-optimal gain with the same weight matrices we used in the LQR controller (see Equation 4.7).
2. Construct a Kalman filter (state estimator) with the Qn and Rn values above.
3. Form the LQG design by connecting the LQ-optimal gain and the Kalman filter.
4. And finally, add the LQG controller to the plant and close the system with an unity feedback.

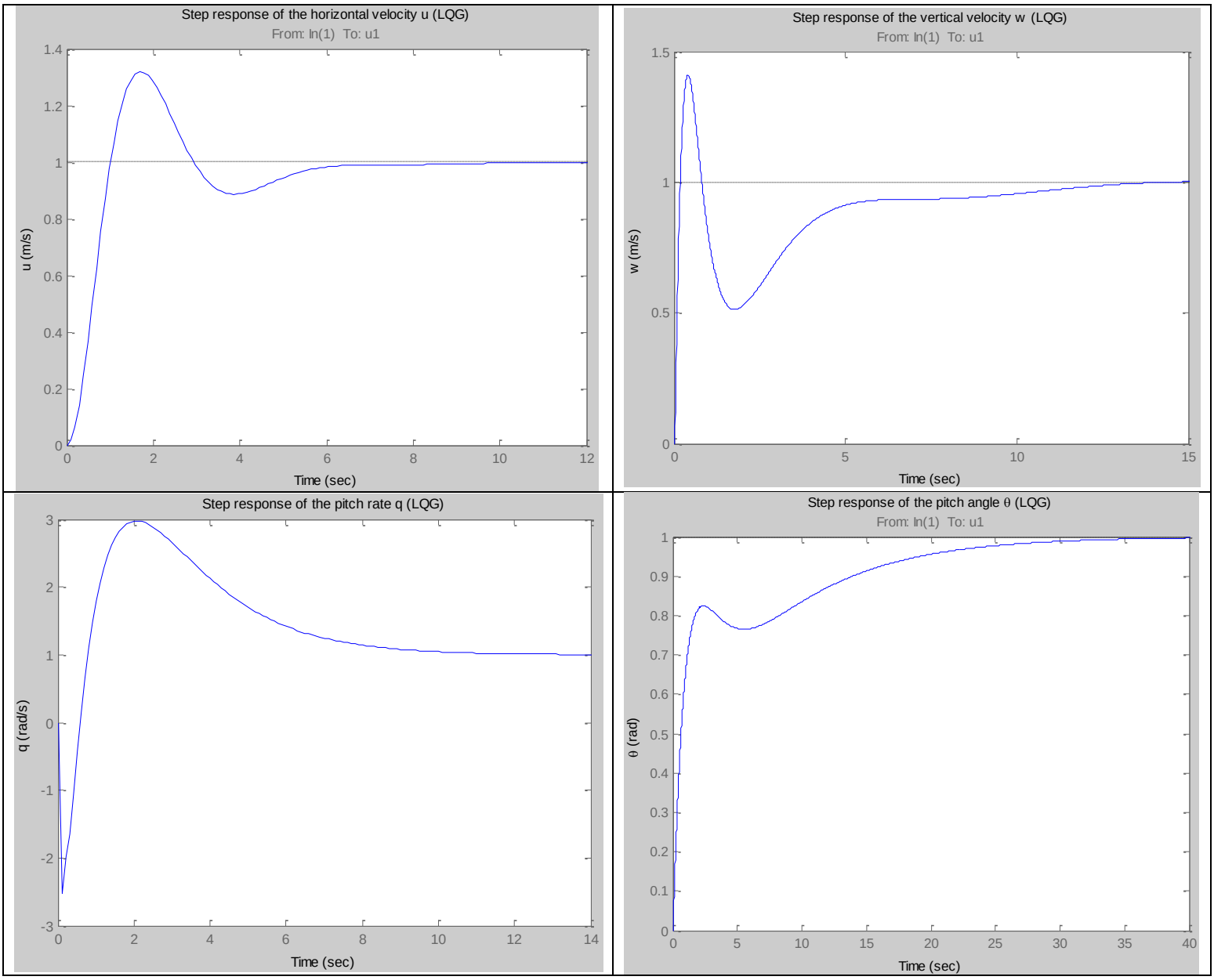
This is what I've obtained:

Table 5.2. LQG Servo Controllers

State Variables	LQG Servo Controller	
	Kest	**K**
u	a = x1_e x2_e x3_e x4_e x1_e -283.3 0.6 -0.36 -9.8 x2_e 27.17 -10.65 16.74 -0.21 x3_e -0.4969 -5.39 -16.55 0 x4_e 14.14 0 1 0 b = u1 y1 x1_e -0.36 282.9 x2_e -3.62 -28.15 x3_e -141.6 0.6769 x4_e 0 -14.14 c = x1_e x2_e x3_e x4_e y1_e 1 0 0 0 x1_e 1 0 0 0 x2_e 0 1 0 0 x3_e 0 0 1 0 x4_e 0 0 0 1 d = u1 y1 y1_e 0 0 x1_e 0 0 x2_e 0 0 x3_e 0 0 x4_e 0 0	$K=\begin{pmatrix} 0.4220 \\ 0.0536 \\ -0.7448 \\ -3.0214 \\ -0.4363 \end{pmatrix}^T$
w	a = x1_e x2_e x3_e x4_e x1_e -0.38 283.9 -0.36 -9.8 x2_e -0.98 -36.37 16.74 -0.21 x3_e 0.18 -1.216 -16.55 0 x4_e 0 -13.95 1 0 b = u1 y1 x1_e -0.36 -283.3 x2_e -3.62 25.72 x3_e -141.6 -4.174 x4_e 0 13.95 c = x1_e x2_e x3_e x4_e y1_e 0 1 0 0 x1_e 1 0 0 0 x2_e 0 1 0 0 x3_e 0 0 1 0 x4_e 0 0 0 1 d = u1 y1 y1_e 0 0 x1_e 0 0 x2_e 0 0 x3_e 0 0 x4_e 0 0	$K=\begin{pmatrix} 0.0441 \\ -0.0085 \\ -0.7310 \\ -0.7061 \\ 0.4363 \end{pmatrix}^T$

q	a = 		x1_e	x2_e	x3_e	x4_e
---	---	---	---	---
x1_e	-0.38	0.6	-1072	-9.8
x2_e	-0.98	-10.65	107.3	-0.21
x3_e	0.18	-5.39	-40.44	0
x4_e	0	0	53.46	0 | b = |	u1	y1
x1_e	-0.36	1072
x2_e	-3.62	-90.53
x3_e	-141.6	23.89
x4_e	0	-52.46 | c = |	x1_e	x2_e	x3_e	x4_e
---	---	---	---	---
y1_e	0	0	1	0
x1_e	1	0	0	0
x2_e	0	1	0	0
x3_e	0	0	1	0
x4_e	0	0	0	1 | d = |	u1	y1
y1_e	0	0
x1_e	0	0
x2_e	0	0
x3_e	0	0
x4_e	0	0		$K=\begin{pmatrix} 0.0009 \\ 0.0288 \\ -0.7285 \\ -0.7747 \\ 0.1854 \end{pmatrix}^T$
θ	a = 		x1_e	x2_e	x3_e	x4_e
---	---	---	---	---
x1_e	-0.38	0.6	-0.36	545
x2_e	-0.98	-10.65	16.74	-75.83
x3_e	0.18	-5.39	-16.55	-3.83
x4_e	0	0	1	-53.52 | b = |	u1	y1
x1_e	-0.36	-554.8
x2_e	-3.62	75.62
x3_e	-141.6	3.83
x4_e	0	53.52 | c = |	x1_e	x2_e	x3_e	x4_e
---	---	---	---	---
y1_e	0	0	0	1
x1_e	1	0	0	0
x2_e	0	1	0	0
x3_e	0	0	1	0
x4_e	0	0	0	1 | d = |	u1	y1
y1_e	0	0
x1_e	0	0
x2_e	0	0
x3_e	0	0
x4_e	0	0		$K=\begin{pmatrix} -0.0006 \\ 0.0290 \\ -0.7305 \\ -1.2075 \\ 0.4363 \end{pmatrix}^T$

Table 5.3. Step response of the system regulated with LQG



As you can see in the table above the step responses have some oscillations with overshoot (as well as undershoot for the pitch rate response) before stabilizing to the reference value. Moreover, for the pitch rate autopilot I have had to add a precompensator in order to reduce a large steady-state error. In fact, with just one look, if we compare them to the step responses obtained with LQR and Classical Control, we can conclude that LQG is not the best solution for our problem.

*For the detailed LQG design procedure you can find the MATLAB code used in the *Appendix A.4.*

5.3. H-infinity loopshaping

5.3.1. Definition

This section formalizes the notion of loopshaping for linear control system design. The loop shaping approach is inherently two-fold. First, we shape the open-loop transfer function (or matrix) P(s)C(s), to meet performance and robustness specifications. Once this is done, then the compensator must be computed, from knowing the nominal product P(s)C(s) and the nominal plant P(s). This means that if we rename the designed loop transfer function as L = PC (target loop shape), it suffices to just pick C=L/P.

This simple step involves a plant inversion: the idea is to first shape L as a stable transfer function meeting the requirements of stability and robustness, and then divide through by the plant transfer function.

As the hard mathematical work is made by the computer, our main goal is to find an adequate target loop shape. There are several guidelines for choosing a target loop shape L:

- Stability Robustness: L should have a gain of less than 0dB at high frequencies, where the plant model is so poor that the phase error may approach 180 degrees.
- Performance: L should have high gain where you want good control accuracy and good disturbance rejection.
- Crossover and Rolloff: L should cross the 0dB line between these two frequency regions, and roll off with a slope of -20 to -40 dB/decade past the crossover frequency Wc.

Just to remember: The frequency Wc, where the gain crosses the 0dB line, is called the crossover frequency and marks the transition between performance and robustness requirements.

5.3.2. Solution and results

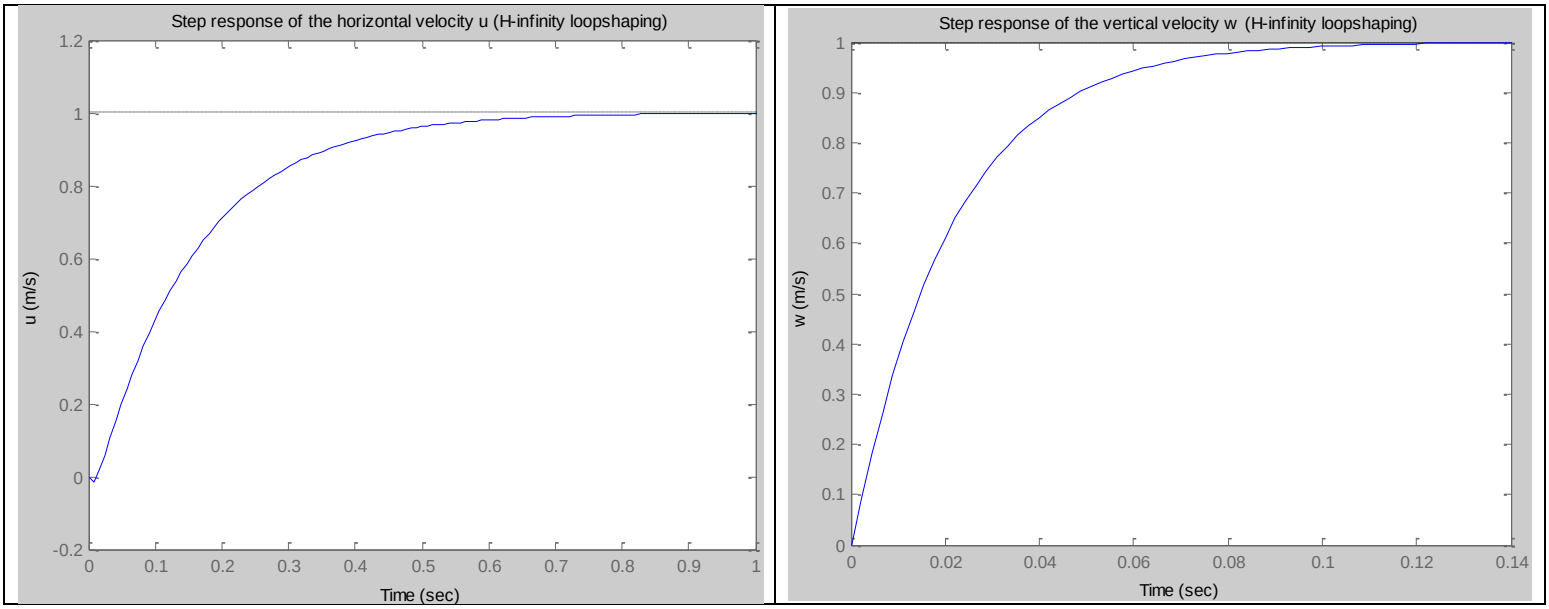
Open-loop shaping methods tune the controller so that the open-loop transfer function L(s) = P(s)C(s) has an appropriate shape. Normally high magnitudes at low frequencies are required for load disturbance attenuation. A -20dB/decade slope is desired near the crossover frequency for appropriate robustness. Finally, small magnitudes at high frequencies are advantageous for a good attenuation of measurement noise or unmodeled dynamics perturbations. Practically, a pure integrator is a recommended shape satisfying these specifications. Then, for its functionality and simplicity, I have chosen L= Wc/s as our target loop shape.

The procedure will be the following:

1. Find the crossover frequency, Wc, of the system with the MATLAB command `sigma`, which plots the minimum and maximum I/O gain as a function of frequency. And after, define the desired loop shape as L= Wc/s.
2. With the function `[K,CL,GAM] = loopsyn(P,Gd)` we compute an H-infinity controller K, such that the gains of the open-loop response P(s)*K(s) match the target-loop shape Gd as well as possible (while stabilizing the aircraft dynamics). The value GAM indicates the loop-shaping accuracy (GAM ≥ 1, with GAM=1 being perfect fit), in other words: it compares the singular values of the open-loop L=G*K with the target-loop shape Gd. The parameter CL contains the closed-loop system (CL= G*K/(I+GK)).

**A further explanation of the procedure can be seen in the Appendix A.5. where there are the corresponding MATLAB codes used in this section.*

Table 5.4. Step response of the system regulated with H-infinity loopshaping



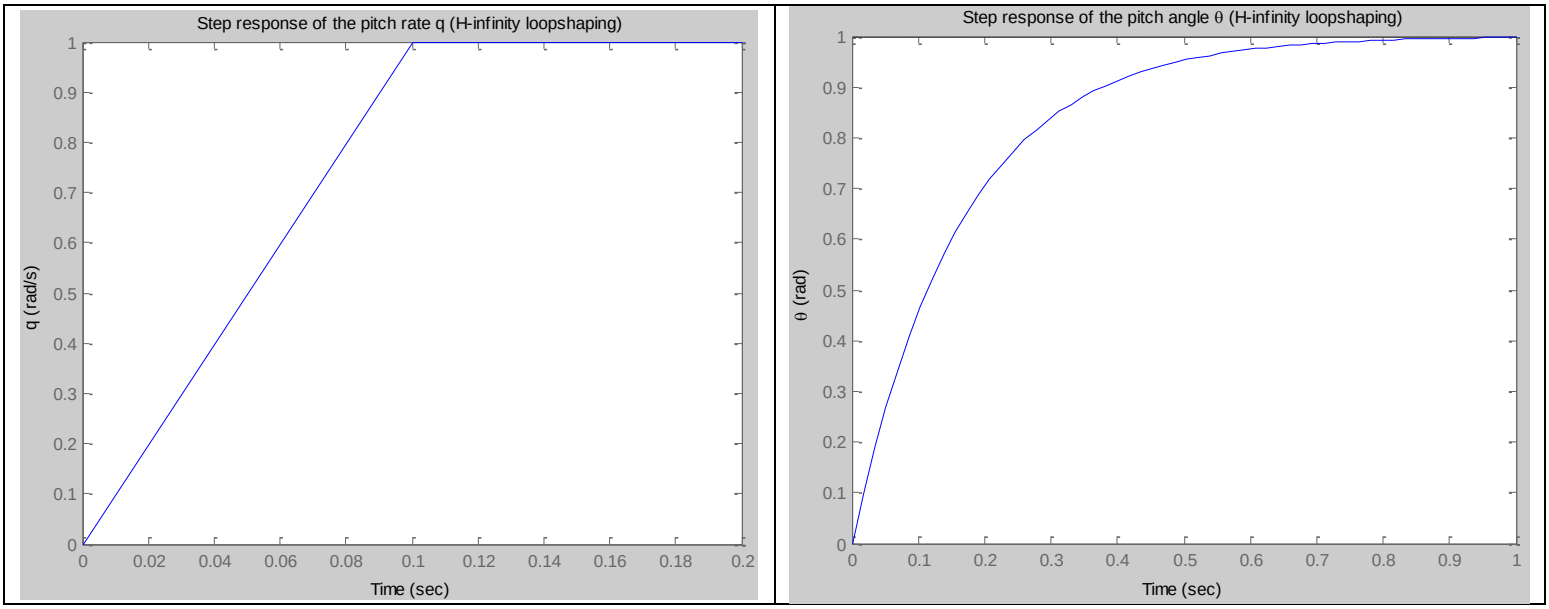


Table 5.5. Crossover frequencies and GAM values of the controllers obtained

State Variables	Wc (rad/s)	GAM	20*log10(GAM) (dB)
u	6.73	1.4946	3.49
w	48.12	1.4266	3.09
q	141.22	1.4506	3.23
θ	6.11	1.4169	3.03

In the *Table 5.5.* above you can see that our approximation to the target loop is really good in all the cases (±3.49dB in the worst case). And in the *Table 5.4.* you can appreciate how good is the solution proposed. With the H-infinity loopshaping controller we obtain a fast and smooth step response without oscillations nor overshoot.

5.3.3. *Controller simplification*

Model reduction techniques presented here are based on the Hankel singular values of a system. They can achieve a reduced-order model that preserves the majority of the system characteristics.

In control theory, Hankel singular values, named after Hermann Hankel, provide a measure of energy for each state in a system. They are the basis for balanced model reduction, in which high energy states are retained while low energy states are discarded. The reduced model retains the important features of the original model.

Therefore, as eigenvalues define a system stability, Hankel singular values define the "energy" of each state in the system. Keeping larger energy states of a system preserves most of its characteristics in terms of stability, frequency, and time responses.

Mathematically, given a stable state-space system (A,B,C,D), its Hankel singular values are defined as:

$$\sigma_H = \sqrt{\lambda_i(VO)} \quad \textbf{(5.4)}$$

Where σ_H corresponds to the Hankel singular value of each state; λ_i, to the eigenvalues of each state; and *V* and *O*, to the controllability and observability matrices.

Our steps to reduce the controllers obtained will be the following:

1. First we compute the Hankel singular values of K to understand how many controller states effectively contribute to the control law.
2. After, depending of the energy of the states, we will select an appropriate order to be reduced.
3. Finally, we will check that our reduced controller Kr is a good approximation of K.

**See in the Appendix A.5. for further explanation. There you will find the MATLAB code.*

In the *Table 5.6.* below are shown the H-infinity loopshaping controllers already reduced, which will be the ones that will be implemented. Notice the state-space form of the controllers.

Table 5.6. H-infinity loopshaping reduced controllers (Kr)

Kr (u)

```
a =
        x1      x2      x3      x4      x5
 x1    -4225    725.2   242.7   38.26       0
 x2    -725.2  -0.09794 -0.7972  -0.2565      0
 x3    -242.7  -0.7972  -11.34   -19.34       0
 x4     38.26   0.2565   19.34   -0.7807      0
 x5       0       0       0       0  4.746e-011

b =
      u1
 x1 -287.3
 x2 -1.345
 x3 -6.619
 x4  1.49
 x5  6.352

c =
       x1     x2     x3     x4     x5
 y1  -287.3   1.345   6.619   1.49  0.004582

d =
    u1
 y1  0
```

Kr (w)

```
a =
       x1     x2     x3     x4     x5
 x1   -4862   1677   35.88  -37.19      0
 x2   -1677  -0.3355 -1.974  0.08993     0
 x3   35.88   1.974 -0.2732  0.6954      0
 x4   37.19  0.08993 -0.6954 -0.02595     0
 x5     0      0      0      0  -5e-013

b =
      u1
 x1    234
 x2  1.898
 x3  -1.47
 x4 -0.4015
 x5  -512.5

c =
       x1     x2     x3     x4     x5
 y1    -234   1.898   1.47  -0.4015 0.004778

d =
    u1
 y1  0
```

Kr (q)

```
a =
        x1       x2       x3       x4       x5
 x1  -0.0002441  1.196e-005 -0.0001288  8.754e-007       0
 x2 -1.196e-005   -0.4457    9.486    -0.06578        0
 x3   0.0001288    9.486    -4379      211          0
 x4  8.754e-007   0.06578    -211     -0.2818         0
 x5       0        0        0        0  1.047e-015

b =
       u1
 x1    243.4
 x2    5.964
 x3   -64.21
 x4   -0.4364
 x5 -1.24e+007

c =
       x1     x2     x3     x4     x5
 y1   243.4  -5.964   64.21  -0.4364 0.004778

d =
    u1
 y1  0
```

Kr (θ)

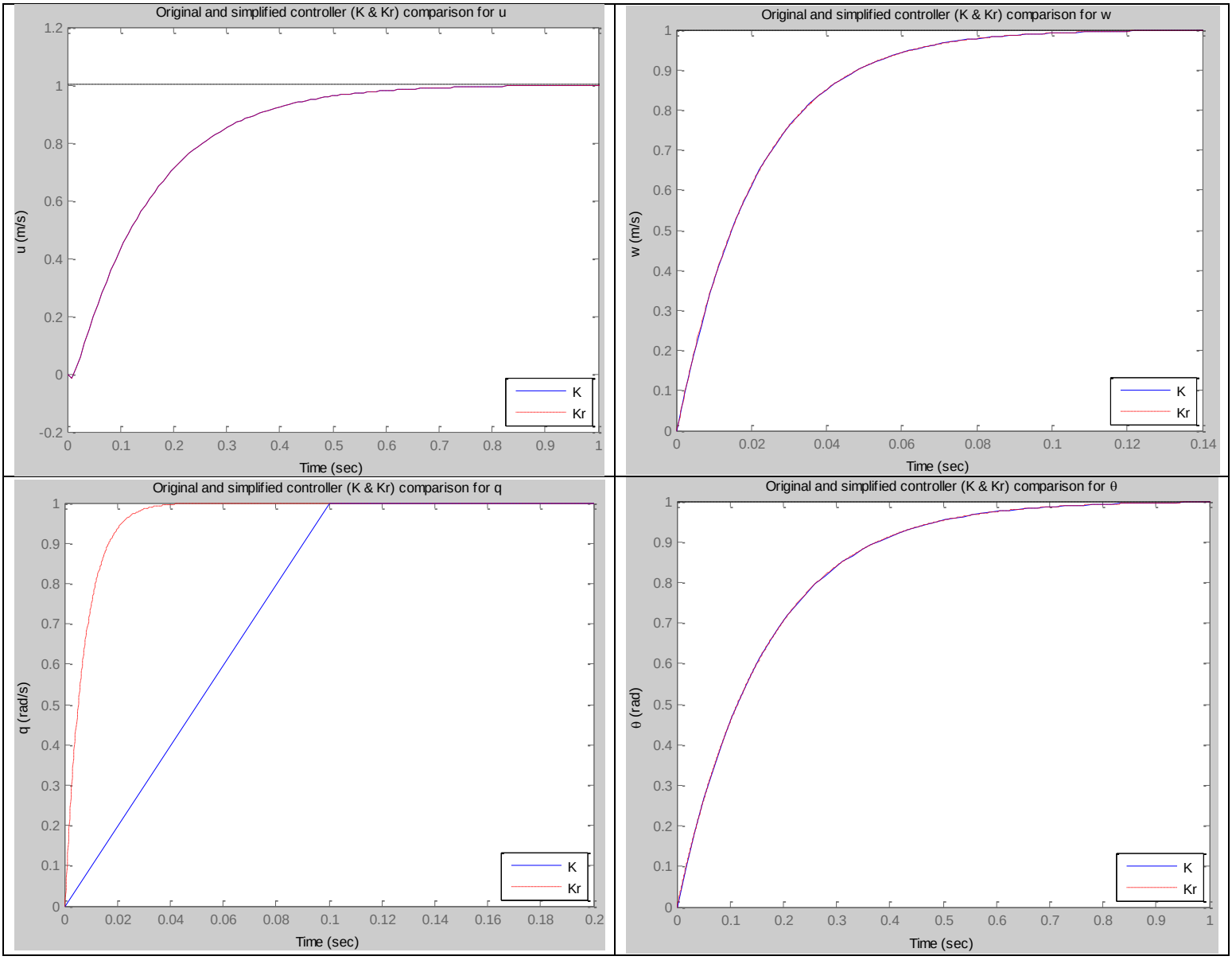
```
a =
        x1      x2      x3      x4      x5
 x1    -8204    4108    10.44   -35.83       0
 x2    -4108   -0.03454 -0.02076  0.07436      0
 x3    -10.44  -0.02076 -0.1831   1.626        0
 x4    -35.83  -0.07436 -1.626   -10.71        0
 x5       0       0       0       0  -1.506e-012

b =
      u1
 x1  851.9
 x2  1.741
 x3 0.5324
 x4  1.867
 x5 -137.1

c =
       x1     x2     x3     x4     x5
 y1  -851.9   1.741  0.5324  -1.867 0.004778

d =
    u1
 y1  0
```

Table 5.7. Step response comparison of the system regulated with the original H-infinity loopshaping controller (K) and the reduced one (Kr)



5.4. LQG Vs H-infinity loopshaping

Although with just one look it was evident that the LQG solution was not the best, here, in Table 5.8. and Table 5.9., I expose a numerical value comparison of both robust control proposed solutions, LQG and H-infinity loopshaping. As you can see there's not too much difference in the steady state error but the settling time is very much faster for the H-infinity solution. Moreover, while the LQG step responses

have oscillations and overshoot the H-infinity loopshaping provides to the system a smooth not undulated response as well as larger stability margins.

Table 5.8. Steady state error and settling time comparison

State Variables	Steady state error (%)		Settling time (sec)	
	LQG	H-infinity loopshaping	LQG	H-infinity loopshaping
u	0.2211	0.1418	5.85	0.5959
w	-0.1132	0.2978	11.83	0.0820
q	-1.2255	0.0002	9.21	0.0980
θ	0.5377	0.2952	25.94	0.6413

Table 5.9. Gain and phase margins comparison

State Variables	Gain Margin (dB)		Phase Margin (º)	
	LQG	H-infinity loopshaping	LQG	H-infinity loopshaping
u	19.9 at 6.82 rad/s	25.2 at 127 rad/s	70.2 at 2.32 rad/s	-180 at 0 rad/s
w	27 at 83.6 rad/s	Inf	104 at 9.18 rad/s	-180 at 0 rad/s
q	-10.2 at 6.78 rad/s (UNSTABLE)	Inf	-112 at 104 rad/s	Inf
θ	43.2 at 87.3 rad/s	62.6 at 4.11e+3 rad/s	-180 at 0 rad/s	-180 at 0 rad/s

6. ROBUSTNESS TEST

From the points above we have seen that Classical Control is the best solution in terms of stability margins while the H-infinity loopshaping is the best in terms of performance. However, gain and phase margins do not guarantee stability for simultaneous process changes and it is one of our goals to ensure the controller performance within some well-defined bounds of disturbances, plant uncertainties and noise. For this reason, in this point, I the controllers will be tested.

In *Figure 1.5.* you can see how the disturbances and noise (after been modelled) will be introduced into the autopilot circuit. As disturbances we will only take into account the wind, which is the most important physical disturbance that an aircraft could experiment during a flight. And as noise we will use the measurement noise that I have already defined in *Table 5.1.* to build the Kalman filter of the LQG controller (page 30).

Due to my version of MATLAB I have had to test the plant uncertainties separately from the disturbances and the noise. So first, I have checked that the uncertainties weren't a problem, and after I have proceeded to test the controllers with the wind conditions and the measurement noise.

*In order to not make the document too much extensive and to speed up its reading, the graphs that will appear from here on will be just the ones related to the fixed pitch angle autopilot. The tests of the other controllers are included in the *Appendix C*.

6.1. Wind conditions modelling

To model the wind conditions I have made use of the Dryden Wind Turbulence Model which is a mathematical model of continuous gusts accepted for use by the United States Department of Defense in certain aircraft design and simulation applications. The Dryden model treats the linear and angular velocity components of continuous gusts as spatially varying stochastic processes and specifies each component's power spectral density.

Specifically, the tool that I have employed is the SIMULINK Dryden Wind Turbulence Model (Continuous) block, which uses the Dryden spectral representation to add turbulence to the aerospace model by passing band-limited white noise through appropriate forming filters.

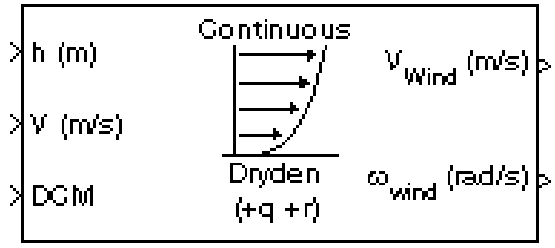


Fig. 6.1. SIMULINK Dryden Wind Turbulence Model block

This block above has as inputs: the altitude of the flight (*h*), the aircraft speed (*V*) and the Direction Cosine Matrix (*DCM*), which is the rotation matrix from R_i (inertial reference frame) to R_b(body fixed frame).The inputs used are the followings:

- The selected flight altitude will be the maximum operational altitude allowed according to *Ultra Stick Aircraft Operations and Maintenance Plan (UAVLAB-OMP-001)*, which is 400ft or 120m.
- The selected aircraft speed will be the one resulting from the throttle set to around 70%, which is 19 m/s.
- And finally, as we just care of the longitudinal dynamics, we just have to use the rotation matrix around the y-axis, $R_2(\theta)$ from Equation 1.5 (page 6). In order to simplify we will assume that the plane keeps a level flight when the pitch angle is nearly zero degrees. Thus, the DCM in this case will be a 3x3 identity matrix.

Moreover, in the block properties, we have to select the wind velocity and the sign of the angular rates due to the gust. I decided to introduce 10 m/s as wind speed, which is just a little bit more of the half aircraft speed. Also, as the airfoil is not symmetrical but is semi-symmetrical, I supposed that it wouldn't be a bad approximation if we say that the sign of the angular rate will have more or less the same disturbance effect. So, in this case, I have only used the positive angular rate.

Then, after configuring the block and select its inputs, it will be ready to provide, as outputs, a three-element signal that contains the turbulence velocities (V_{wind}) and another three-element signal that contains the turbulence angular rates (w_{wind}). The first and the third element of V_{wind} correspond to the disturbances in *u* and *w*; the third element of w_{wind} corresponds to the disturbance in *q*, which will be also used as an approximation for the pitch angle (θ) disturbance.

6.2. Plant uncertainties

As I have explained before in the point *1.4. Frequency domain System identification* (page 10), the plant has uncertainties. Most of them come from the

baseline model of the phugoid dynamics, which were kept against the short-period mode elements, which were obtained experimentally and thus they are more accurate. For this reason I have given 10% of uncertainty to the phugoid mode dynamics and 5% of uncertainty to the short-period dynamics.

What I have obtained is that all the controllers done are robust enough to keep their performance against this kind of uncertainty. Below you can see the insignificant roll of plant uncertainty in this case.

Table 6.1. Step responses with plant uncertainties

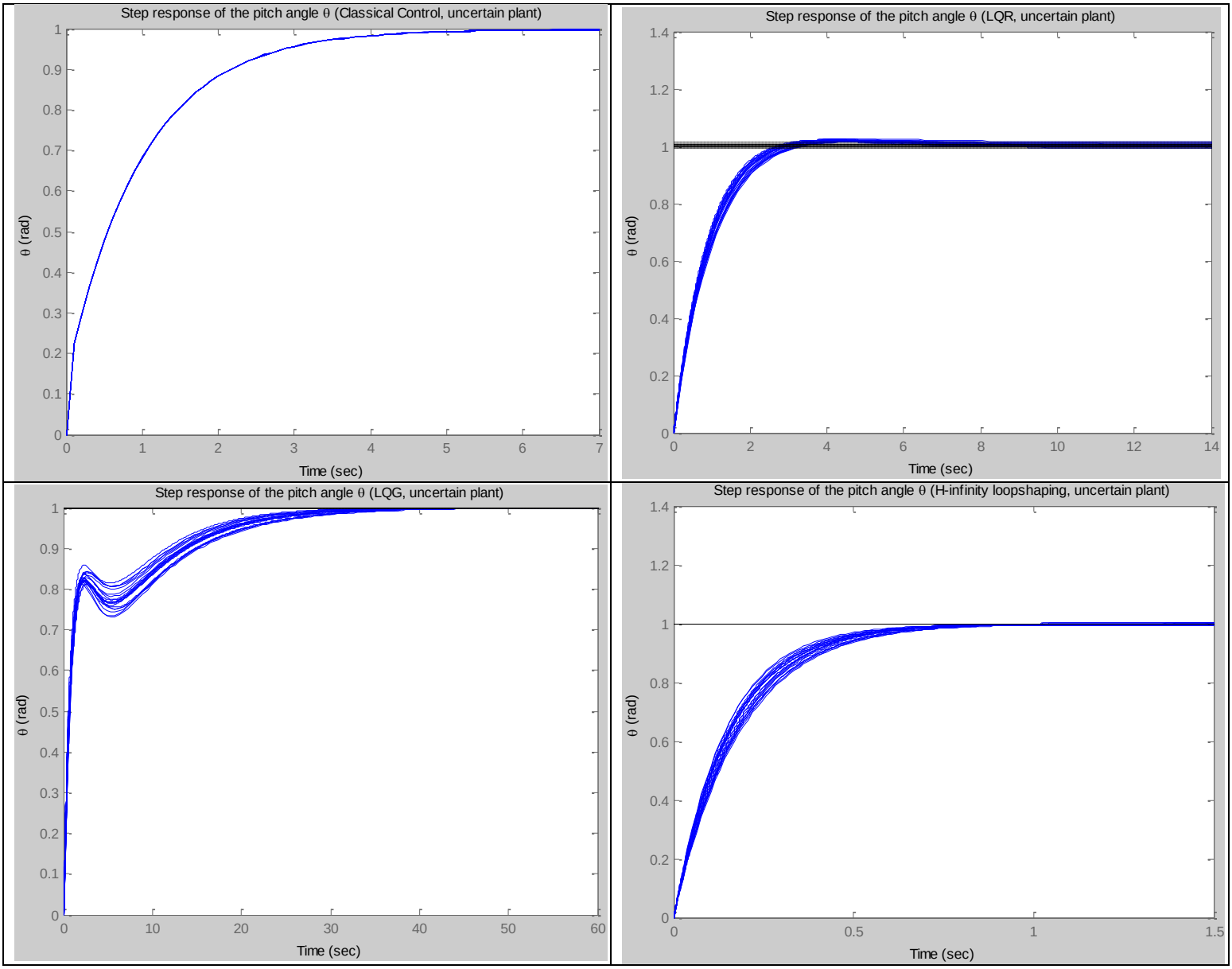


Table 6.2. Maximum error of pitch angle due to plant uncertainties

Error (%)			
Classical control	**LQR**	**LQG**	**H-infinity loopshaping**
0.3	1.1	0.2	0.6

Plant uncertainties have less effect on the system regulated with Classical control. Nevertheless, the other controllers are also robust enough to face the plant uncertainties. The LQG step response has a large settling time but, as you can see in *Table 5.3.* and *Table 5.8.*, this is a prior problem that doesn't come from plant uncertainties.

6.3. Wind and noise test for the pitch controller

For the pitch controller it corresponds 0.035 rad (2 deg) as measurement noise (*see Table 5.1.*). In this test the value of the reference pitch will be very important; noise and disturbance are introduced into the circuit and they will have more effect when the reference value is low. At least I consider than an aircraft has to be able to maintain a 3 degree pitch, which is a common ascent/descent gradient, so the following step responses have 0.05 rad (2.86 deg) as reference value.

Note: In the next graphs the y-axis corresponds to the value of the pitch angle (rad) and the x-axis corresponds to the time (sec).

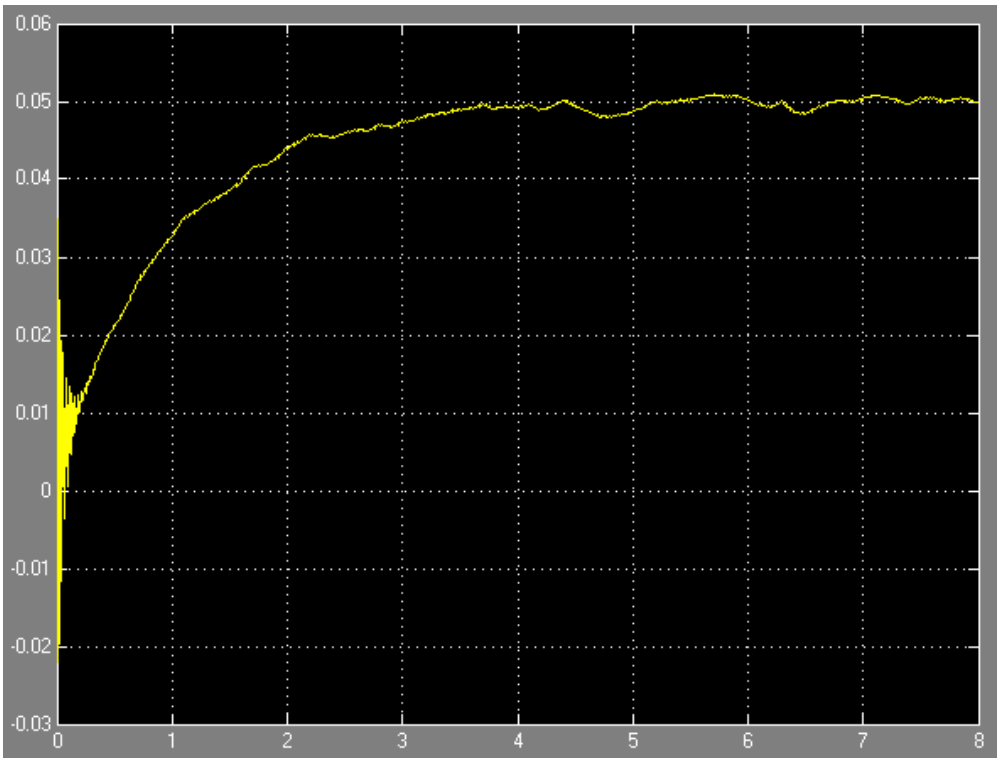


Fig. 6.2. Step response with noise and wind conditions (Classical control)

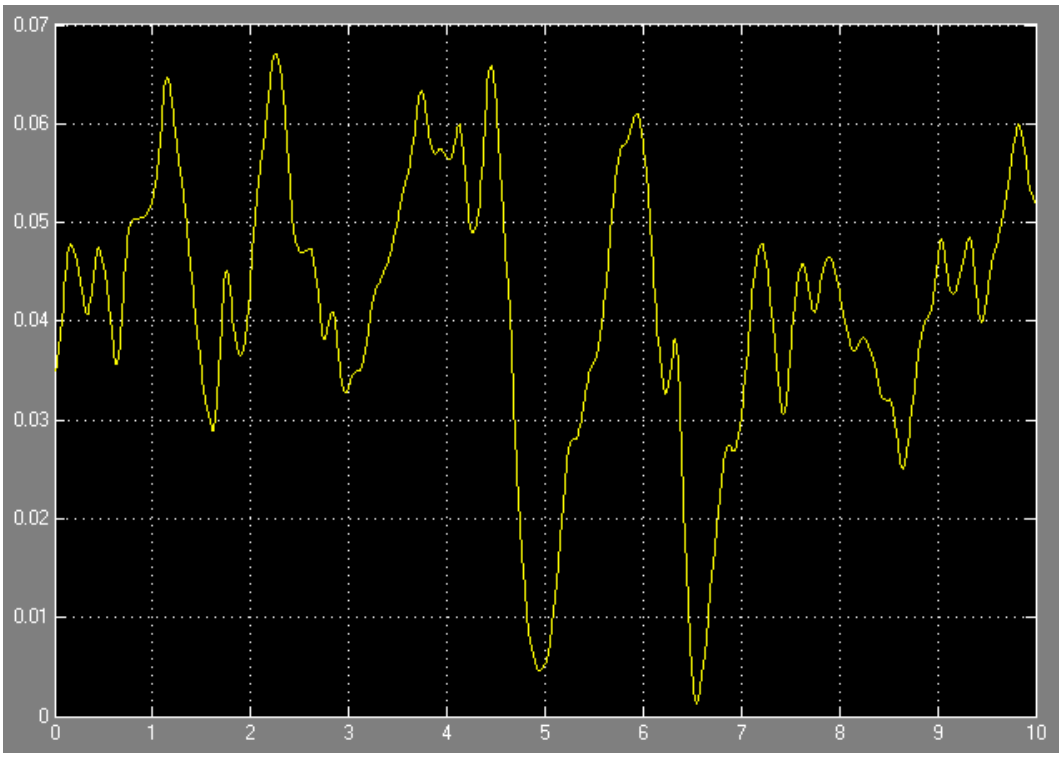


Fig. 6.3. Step response with noise and wind conditions (LQR)

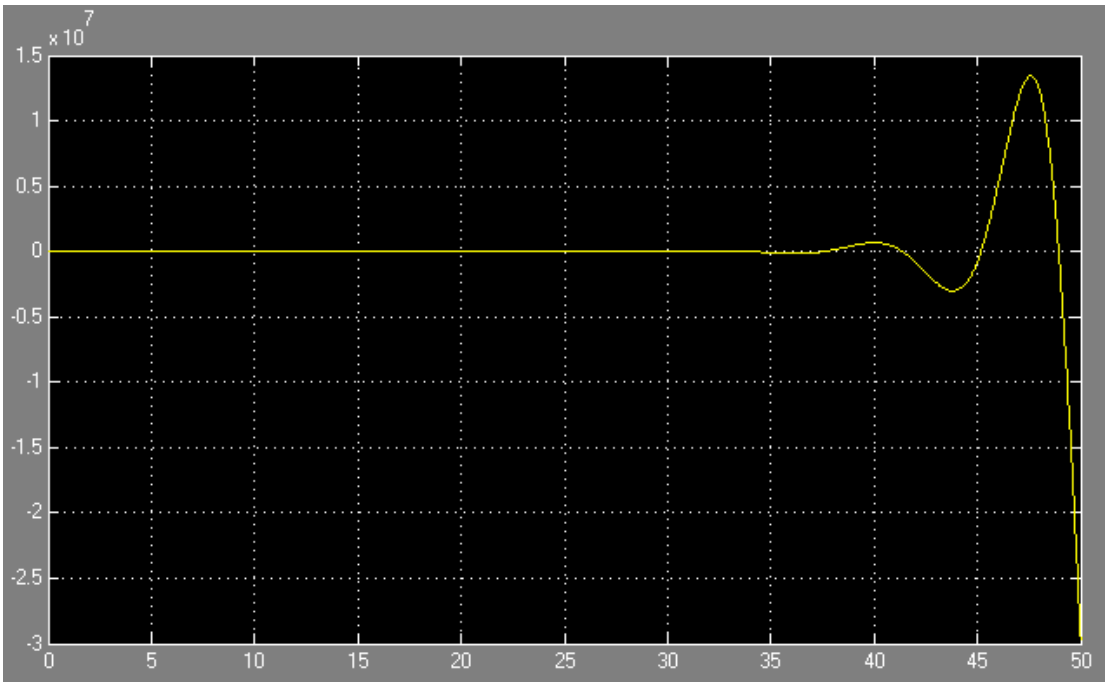


Fig. 6.4. Step response with noise and wind conditions (LQG)

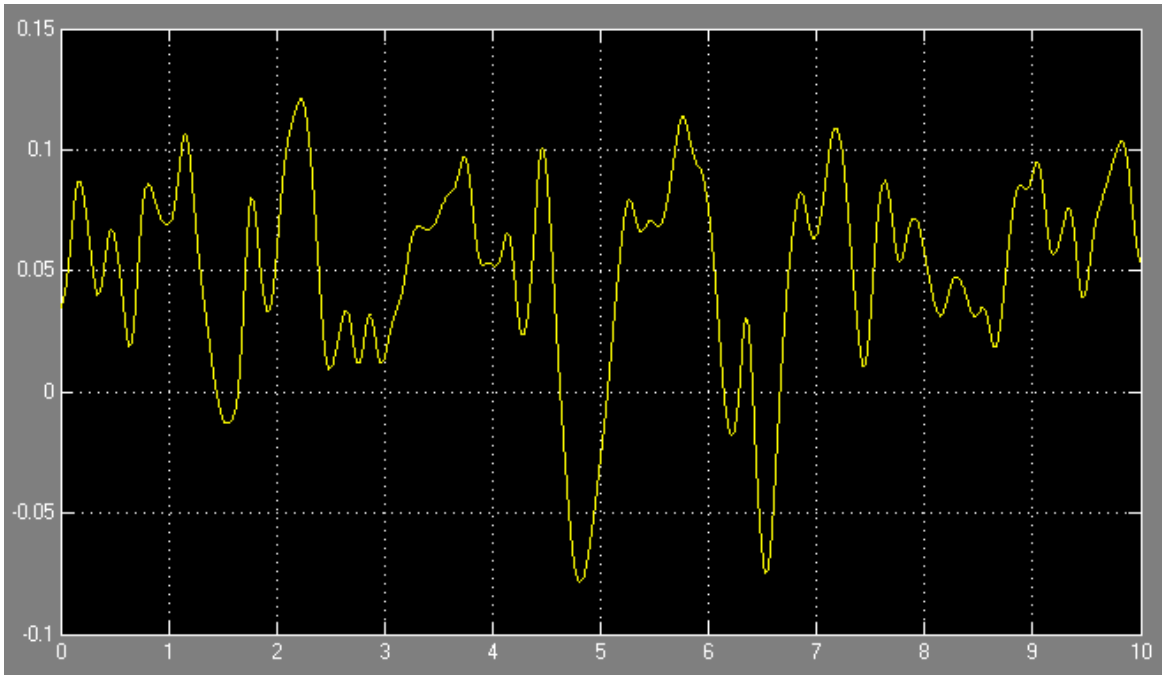


Fig. 6.5. Step response with noise and wind conditions (H-infinity loopshaping)

No doubt, although all the controllers resist the plant uncertainties pretty well, only the Classical control solution proposed passes our wind and noise test. It is, definitely, the best choice. In the *Figure 6.6.* below we can see that it has less than 4% of error in the worst case. The unique negative it has are the high frequency oscillations at the first of the transient response (see *Figure 6.2.*). However, they are damped in less than half second, so they can be considered as acceptable.

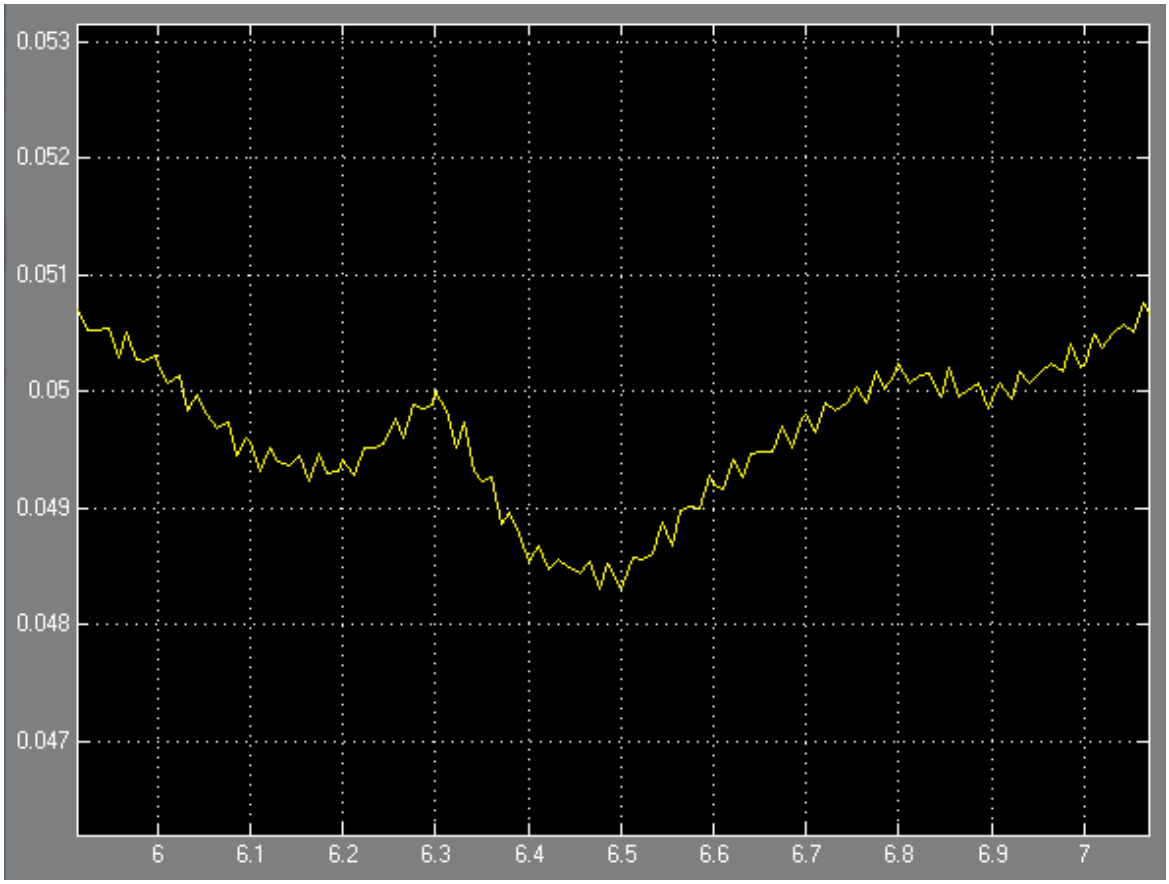


Fig. 6.6. Zoom of *Figure 6.2.* from 6 to 7 seconds (Classical control)

7. CONCLUSION

7.1. Best solution

After all, the Classical Control method (implemented with a simple trial and error approach) has been the best solution. It has not the best settling time because the H-infinity loopshaping controller makes the system more than 3 seconds faster than the classical controller, and their step responses look quite similar in terms of steady-state error and smoothness, but the Classical Control solution is definetly the most robust. In the case of the pitch controller, while the LQR and H-infinity loopshaping lead the disturbed model of the system to a noisy step responses with high amplitude oscillations, and even with LQG the system becomes unstable when trying to maintain a constant pitch angle, the Classical controller is able to stabilise the signal to the desired output with just some oscillations, due to the noise effect, of no more than 4% of the reference value as amplitude.

7.2. Difficulties

Control systems is a broad and tedious theme with a lot of mathematics behind and, with just a 6 ECTS subject as background, the real engineering work it has been to synthesise all the information related found on internet, to understand it and to know how the different control methods function without having a very deep knowledge of them. For this reason, in most cases, and thanks to the MATLAB tools, it has been easier to find the solution than to explain it.

Another inconvenient that I had it was that due to my MATLAB version, 7.7.0 (r2008b), I have not been able to use more powerful and modern tools such as automatic PID tuner.

I would like to express my gratitude in this sense to my tutor, Mme Bestaoui, to soften these difficulties along my stage in Université d'Evry Val d'Essone.

7.3. What is next?

These have been unexpected results for me. It's true that LQR doesn't take into account robustness and LQG takes into account the noise rejection but not the disturbances, but the H-infinity loopshaping was supposed to deal with both disturbance and noise, and after seeing its performances I would bet that it would be the most robust solution. Nevertheless, it wasn't. This fact could be due to that I should select another target loop shape or that the H-infinity loopshaping controller needs some kind of reinforcement. If I had to continue with this project for more time I would focus on the H-infinity investment and improving the original controllers described here.

Furthermore, the next steps, once we are able to maintain constant our four longitudinal state variables (u,w,q θ) with just the effect of the elevator deflection, will be:

1. To study the elevator deflection behaviour to assure a correct functioning.
2. To introduce the thrust as another controller input as well as the elevator deflection is.
3. To make more longitudinal complex controllers that could maintain an ascend/descend rate.
4. To combine the longitudinal autopilot with the lateral/directional one and make a controller that enables the aircraft to do a steady coordinated turn.

8. BIBLIOGRAPHY

1. Mecánica del vuelo, 2ª edición, M.A. Gómez Tierno, M. Pérez Cortés, C. Puentes Márquez, Ibergarceta Publicaciones, 2012

2. Dorobantu, A., Murch, A., Mettler, B., and Balas, G, “Frequency Domain System Identification for a Small, Low-Cost, Fixed-Wing UAV”, AIAA Journal of Aircraft, Vol. 50, No. 4, pp. 1117-1130, July, 2013

3. Dorobantu, A., Murch, A., Mettler, B., and Balas, G, “Frequency Domain System Identification for a Small, Low-Cost, Fixed-Wing UAV”, AIAA Atmospheric Flight Mechanics Conference and Exhibit, AIAA Paper 2011-6719, Portland, OR, 2011

4. Hall, J.A., “Imaging tubes”, Cap. 13 en The Infrared handbook, Wolfe, W.W., Zissis, G.J., Eds., pp. 132-176, ERIM, Ann Aror, MI (1978).

5. Åström, K.J., “Control System Design”, University of California, Santa Barbara, 2002

6. Ultra Stick Aircraft Operations and Maintenance Plan (UAVLAB-OMP-001). Web. <http://trac.umnaem.webfactional.com/export/1011/trunk/Documentation/Operations%20and%20Maintenance%20Plans/Ultra%20Stick%20Operations%20and%20Maintenance%20Plan%20UAVLAB-OMP-001.pdf>

7. Galdos, G., Karimi, A. and Longchamp, R., “Robust Loop Shaping Controller Design for Spectral Models by Quadratic Programming”, Laboratoire d’Automatique of Ecole Polytechnique Fédérale de Lausanne, Lausanne (Switzerland).Web. <http://la.epfl.ch/files/content/sites/la/files/shared/import/migration/cdc_07.pdf>

8. Toivonen, H., “Lecture notes on Robust Control by state-space methods”, Department of Chemical Engineering at Åbo Akademi University, Rurku (Finland). Web. < http://users.abo.fi/htoivone/courses/robust/hsem.pdf>

9. MIT OpenCourseWare, “Control Systems - Loopshaping”, 2009. Web. <http://ocw.mit.edu/courses/mechanical-engineering/2-017j-design-of-tromechanical-robotic-systems-fall-2009/course-text/MIT2_017JF09_ch12.pdf>

10. Sreeraj, P.V., “Design and Implementation of PID Controller with Lead Compensator for Thermal Process”, International Journal of Computer Applications (0975 – 8887), Vol. 67, No 1, April, 2013

11. Blakelock, J., “Automatic Control of Aircraft and Missiles”, 2nd Ed., John Wiley & Sons, 1991

12. Anderson, D. F. and Eberhardt, S., “Understanding Flight”, 2nd Ed., McGraw Hill, 2010

13. G.F Franklin, J.D. Powell, A. Emani-Naeini, “Feedback Control of Dynamic Systems”, 4a Edición, Prentice-Hall, 2002

14. R. Nelson, “Flight Stability and Automatic Control”, 2nd Ed., Mc Graw-Hill, 1998

15. D. Arzelier, D. Peaucelle, “Représentation et analyse des systèmes linéaires”, Tomes 1 et 2, Version 1, ENSICA, 1999

16. P. Lewis, Sistemas de Control en Ingeniería, Prentice Hall, 1999

17. W. Bolton, “Control Engineering”, 2nd Ed., Longman, 1998

18. P.J. Antsaklis and Z. Gao, “Control System Design,” The Electronics Engineers' Handbook, 5th Edition, McGrawHill, Section 19, pp. 19.1-19.30, 2005

19. “Loop Shaping of HIMAT Pitch Axis Controller”, Matlab Robust Control Toolbox User’s Guide. Web. <http://www.mathworks.com/help/robust/examples/loop-shaping-of-himat-pitch-axis-controller.html>

20. “Functions for Compensator Design”, Matlab Control System Toolbox User’s Guide. Web. <http://www.mathworks.com/help/control/getstart/functions-for-compensator-design.html#brs1ke4>

21. “Dryden Wind Turbulence Model (Continuous)”, Simulink Aerospace Blockset User’s guide. Web. <http://www.mathworks.com/help/aeroblks/drydenwindturbulencemodelcontinuous.html>

22. <http://en.wikipedia.org/wiki/Kalman_filter>

APPENDIX A – MATLAB codes (θ)

Note: Apart from the code used to determine the stability, controllability and observability of the plant (A.1.), only codes for the pitch controllers are exposed here. The controllers of the other variables follow exactly the same structure.

A.1. Stability, controllability and observability study

```
clear all;
close all;

%% State-space model
A=[-0.38 0.60 -0.36 -9.80;
   -0.98 -10.65 16.74 -0.21;
   0.18 -5.39 -16.55 0;
   0 0 1 0];

B=[-0.36;
   -3.62;
   -141.57;
   0];

C= eye(4);

%% Eigenvalues of A

e=eig(A); %eigenvalues of A

%% Controllability
V=[B, A*B, (A^2)*B, (A^3)*B];%Controllability matrix
rank_V=rank(V);

%% Observability
O=obsv(A,C); %Observability matrix
rank_O=rank(O);
```

A.2. PI and lead-compensator implementation

```
clear all;
close all;

%% Define system
A=[-0.38 0.60 -0.36 -9.80;
   -0.98 -10.65 16.74 -0.21;
   0.18 -5.39 -16.55 0;
   0 0 1 0];

B=[-0.36;
   -3.62;
```

```
   -141.57;
   0];

C=[0 0 0 1];

D=0;

sys=ss(A,B,C,D);

%% Define system with uncertainties
a11=ureal('a11',1,'Percentage',10);
a12=ureal('a12',1,'Percentage',10);
a13=ureal('a13',1,'Percentage',10);
a14=ureal('a14',1,'Percentage',10);
a21=ureal('a21',1,'Percentage',10);
a22=ureal('a22',1,'Percentage',5);
a23=ureal('a23',1,'Percentage',5);
a24=ureal('a24',1,'Percentage',10);
a31=ureal('a31',1,'Percentage',10);
a32=ureal('a32',1,'Percentage',5);
a33=ureal('a33',1,'Percentage',5);
a34=ureal('a34',1,'Percentage',10);
a41=ureal('a41',1,'Percentage',10);
a42=ureal('a42',1,'Percentage',10);
a43=ureal('a43',1,'Percentage',10);
a44=ureal('a44',1,'Percentage',10);

uA=[-0.38*a11 0.60*a12 -0.36*a13 -9.80*a14;
   -0.98*a21 -10.65*a22 16.74*a23 -0.21*a24;
   0.18*a31 -5.39*a32 -16.55*a33 0;
   0 0 1*a43 0];

b1=ureal('b1',1,'Percentage',10);
b2=ureal('b2',1,'Percentage',5);
b3=ureal('b3',1,'Percentage',5);
b4=ureal('b4',1,'Percentage',10);
uB=[-0.36*b1;
   -3.62*b2;
   -141.57*b3;
   0];

usys=ss(uA,uB,C,D);

%% Original system
%Plot step response
figure(1);
step(sys);
title('Step response of the pitch angle \theta (Original system)');
ylabel('\theta (rad)');

%System information
S_original = stepinfo(sys)
[y,t]=step(100*sys);
sserror_original=100-y(end); %steady-state error(%)
```

```
%% Regulated system
s=tf('s'); %Define s as the Laplace variable
k=-3; %Gain
H=8+7/s; %PI controller
LeadComp=20*((s+1)/(s+20)); %Lead compensator
sys_close=feedback(k*H*sys,LeadComp); %Closed loop system
%Plot step response
figure(2);
step(sys_close,7);
str=sprintf('Step response of the pitch angle \\theta (Regulated with
Classical Control)');
title(str);
ylabel('\theta (rad)');

%Plot step response with uncertainties
usys_close=feedback(k*H*usys,LeadComp); %Closed loop system with
uncertainties
figure(3);
step(usys_close);
str=sprintf('Step response of the pitch angle \\theta (Classical Control,
uncertain plant)');
title(str);
ylabel('\theta (rad)');

%System information
S_final = stepinfo(sys_close)
[y,t]=step(100*sys_close);
sserror_final=100-y(end);

%Display gain and phase margins
figure(4)
margin(sys_close)
```

A.3. LQR implementation

```
clear all;
close all;

%% Define matrices
A=[-0.38 0.60 -0.36 -9.80;
   -0.98 -10.65 16.74 -0.21;
   0.18 -5.39 -16.55 0;
   0 0 1 0];

B=[-0.36;
   -3.62;
   -141.57;
   0];

C4=[0 0 0 1];

D=0;
```

```
%% Define system with uncertainties
a11=ureal('a11',1,'Percentage',10);
a12=ureal('a12',1,'Percentage',10);
a13=ureal('a13',1,'Percentage',10);
a14=ureal('a14',1,'Percentage',10);
a21=ureal('a21',1,'Percentage',10);
a22=ureal('a22',1,'Percentage',5);
a23=ureal('a23',1,'Percentage',5);
a24=ureal('a24',1,'Percentage',10);
a31=ureal('a31',1,'Percentage',10);
a32=ureal('a32',1,'Percentage',5);
a33=ureal('a33',1,'Percentage',5);
a34=ureal('a34',1,'Percentage',10);
a41=ureal('a41',1,'Percentage',10);
a42=ureal('a42',1,'Percentage',10);
a43=ureal('a43',1,'Percentage',10);
a44=ureal('a44',1,'Percentage',10);

uA=[-0.38*a11 0.60*a12 -0.36*a13 -9.80*a14;
   -0.98*a21 -10.65*a22 16.74*a23 -0.21*a24;
   0.18*a31 -5.39*a32 -16.55*a33 0;
   0 0 1*a43 0];

b1=ureal('b1',1,'Percentage',10);
b2=ureal('b2',1,'Percentage',5);
b3=ureal('b3',1,'Percentage',5);
b4=ureal('b4',1,'Percentage',10);

uB=[-0.36*b1;
   -3.62*b2;
   -141.57*b3;
   0];

usys=ss(uA,uB,C4,D);
%% Linear Quadratic Regulator
Q = [(1/25)^2,0,0,0;...
    0,(1/12)^2,0,0;...
    0,0,(1/(30*pi/180))^2,0;...
    0,0,0,(1/(30*pi/180))^2];
R = 1/(25*pi/180)^2;
[K,S,E] = lqr(A,B,Q,R);
A2 = A - (B*K);

%% pitch (O)
Nbar_O = rscale(A,B,C4,0,K); %Compute precompensator gain Nbar
cl_sys = ss(A2,B*Nbar_O,C4,D); %Closed loop system
%Plot O
figure(1);
step(cl_sys);
title('Step response of the pitch angle \theta (Regulated with LQR)');
ylabel('\theta (rad)');
%Compute steady state error in %
[y,t]=step(cl_sys);
sserror_o_percent=(1-y(end))/100;
%Display system information
o_info = stepinfo(cl_sys)
```

```
%Display gain and phase margins
figure(2)
margin(cl_sys);

%Plot step response with uncertainties
A2 = uA - (uB*K);
usys_close=ss(A2,uB*Nbar_O,C4,D); %Closed loop system with uncertainties
figure(3);
step(usys_close);
str=sprintf('Step response of the pitch angle \\theta (LQR, uncertain
plant)');
title(str);
ylabel('\theta (rad)');
```

A.4. LQG implementation

```
clear all;
close all;

%% State-space system
A=[-0.38 0.60 -0.36 -9.80;
   -0.98 -10.65 16.74 -0.21;
   0.18 -5.39 -16.55 0;
   0 0 1 0];

B=[-0.36;
   -3.62;
   -141.57;
   0];

G = [200; -20; 0; -10];
C4=[0 0 0 1];

D = 0;
H = 0;
sys = ss(A,[B G],C4,0);

%% Define system with uncertainties
a11=ureal('a11',1,'Percentage',10);
a12=ureal('a12',1,'Percentage',10);
a13=ureal('a13',1,'Percentage',10);
a14=ureal('a14',1,'Percentage',10);
a21=ureal('a21',1,'Percentage',10);
a22=ureal('a22',1,'Percentage',5);
a23=ureal('a23',1,'Percentage',5);
a24=ureal('a24',1,'Percentage',10);
a31=ureal('a31',1,'Percentage',10);
a32=ureal('a32',1,'Percentage',5);
a33=ureal('a33',1,'Percentage',5);
a34=ureal('a34',1,'Percentage',10);
a41=ureal('a41',1,'Percentage',10);
a42=ureal('a42',1,'Percentage',10);
```

```
a43=ureal('a43',1,'Percentage',10);
a44=ureal('a44',1,'Percentage',10);

uA=[-0.38*a11 0.60*a12 -0.36*a13 -9.80*a14;
   -0.98*a21 -10.65*a22 16.74*a23 -0.21*a24;
   0.18*a31 -5.39*a32 -16.55*a33 0;
   0 0 1*a43 0];

b1=ureal('b1',1,'Percentage',10);
b2=ureal('b2',1,'Percentage',5);
b3=ureal('b3',1,'Percentage',5);
b4=ureal('b4',1,'Percentage',10);

uB=[-0.36*b1;
   -3.62*b2;
   -141.57*b3;
   0];


usys=ss(uA,uB,C4,D);

%% LQG gain
nx = 4;    %Number of states
ny = 1;    %Number of outputs
Q = [(1/25)^2,0,0,0;...
    0,(1/12)^2,0,0;...
    0,0,(1/(30*pi/180))^2,0;...
    0,0,0,(1/(30*pi/180))^2];
R = 1/(25*pi/180)^2;
Q = blkdiag(Q,1);
%R = 1;
K = lqi(ss(A,B,C4,D),Q,R);


%% Kalman Filter
%Assumption: noise uncorrelated between the different states --> Qn, Rn
are diagonal
Qn = 1; %process noise
Rn = 0.035; %noise signal from the sensors
kest = kalman(sys,Qn,Rn);
s = tf('s');
trksys = lqgtrack(kest,K,'1dof'); %connect LQ-optimal gain with the
Kalman filter
sys=ss(A,B,C4,D);


%% Add Kalman filter to the system and plot step response
% Closed loop
clsys = feedback(trksys*sys,1)
figure(1)
step(clsys)
str=sprintf('Step response of the pitch angle \\theta (LQG)');
title(str);
ylabel('\theta (rad)');
```

```
%Plot step response with uncertainties
uclsys=feedback(trksys*usys,1); %Closed loop system with uncertainties
figure(2);
step(uclsys);
str=sprintf('Step response of the pitch angle \\theta (LQG, uncertain
plant)');
title(str);
ylabel('\theta (rad)');

%System information
S_final = stepinfo(clsys)
[y,t]=step(100*clsys);
sserror_final=100-y(end);

%Display gain and phase margins
figure(3)
margin(clsys);
```

A.5. H-infinity loopshaping implementation

```
clear all;
close all;

%% State-space model
A=[-0.38 0.60 -0.36 -9.80;
   -0.98 -10.65 16.74 -0.21;
   0.18 -5.39 -16.55 0;
   0 0 1 0];

B=[-0.36;
   -3.62;
   -141.57;
   0];

C4=[0 0 0 1]; %longitudinal speed u

D=0;

sys=ss(A,B,C4,D);


%% Define system with uncertainties
a11=ureal('a11',1,'Percentage',10);
a12=ureal('a12',1,'Percentage',10);
a13=ureal('a13',1,'Percentage',10);
a14=ureal('a14',1,'Percentage',10);
a21=ureal('a21',1,'Percentage',10);
a22=ureal('a22',1,'Percentage',5);
a23=ureal('a23',1,'Percentage',5);
a24=ureal('a24',1,'Percentage',10);
a31=ureal('a31',1,'Percentage',10);
a32=ureal('a32',1,'Percentage',5);
```

```
a33=ureal('a33',1,'Percentage',5);
a34=ureal('a34',1,'Percentage',10);
a41=ureal('a41',1,'Percentage',10);
a42=ureal('a42',1,'Percentage',10);
a43=ureal('a43',1,'Percentage',10);
a44=ureal('a44',1,'Percentage',10);

uA=[-0.38*a11 0.60*a12 -0.36*a13 -9.80*a14;
   -0.98*a21 -10.65*a22 16.74*a23 -0.21*a24;
   0.18*a31 -5.39*a32 -16.55*a33 0;
   0 0 1*a43 0];

b1=ureal('b1',1,'Percentage',10);
b2=ureal('b2',1,'Percentage',5);
b3=ureal('b3',1,'Percentage',5);
b4=ureal('b4',1,'Percentage',10);

uB=[-0.36*b1;
   -3.62*b2;
   -141.57*b3;
   0];


usys=ss(uA,uB,C4,D);


%% H-infinity loop shaping
s=tf('s'); %Define s as the Laplace variable
%Singular value diagram to obtain the crossover frequency (Wc)
figure(1)
sigma(sys) %Wc=6.11 rad/s (Value seen on figure 1)
Wc=6.11;
G=Wc/(s); %desired loopshape
[K,CL,GAM] = loopsyn(sys,G);
%% Closed-loop system
L=K*sys; %Adding disturbance to the system
L_close=feedback(L,1); %Closed-loop
figure(2)
step(L_close);
str=sprintf('Step response of the pitch angle \\theta (H-infinity
loopshaping)');
title(str);
ylabel('\theta (rad)');

%System information
S_final = stepinfo(L_close)
[y,t]=step(100*L_close);
sserror_final=100-y(end);

%Display gain and phase margins
figure(3)
margin(L_close);

%% Controller simplification
%Hankel singular values of K
```

```
figure(4)
hsv = hankelsv(K);
semilogy(hsv,'*--'), grid
title('Hankel singular values of K (\theta)'), xlabel('Order')

%Reduce controller from 7 to 5 states
Kr = reduce(K,5);
order(Kr) %check simplification

%Closed-loop responses comparison (K and Kr)
Lr=Kr*sys;
Lr_close=feedback(Lr,1);
figure(5)
step(L_close,'b',Lr_close,'r-.');
str=sprintf('Original and simplified controller (K & Kr) comparison for
\\theta');
title(str);
ylabel('\theta (rad)');
legend('K','Kr','Location','Southeast')

%% Plot step response with uncertainties
uLr=Kr*usys;
uLr_close=feedback(uLr,1); %Closed loop system with uncertainties
figure(6);
step(uLr_close);
str=sprintf('Step response of the pitch angle \\theta (H-infinity
loopshaping, uncertain plant)');
title(str);
ylabel('\theta (rad)');
```

APPENDIX B – Simulink models (θ)

Note: Only codes for the pitch controllers are exposed here. The controllers of the other variables follow exactly the same structure.

B.1. PI and Lead-compensator scheme

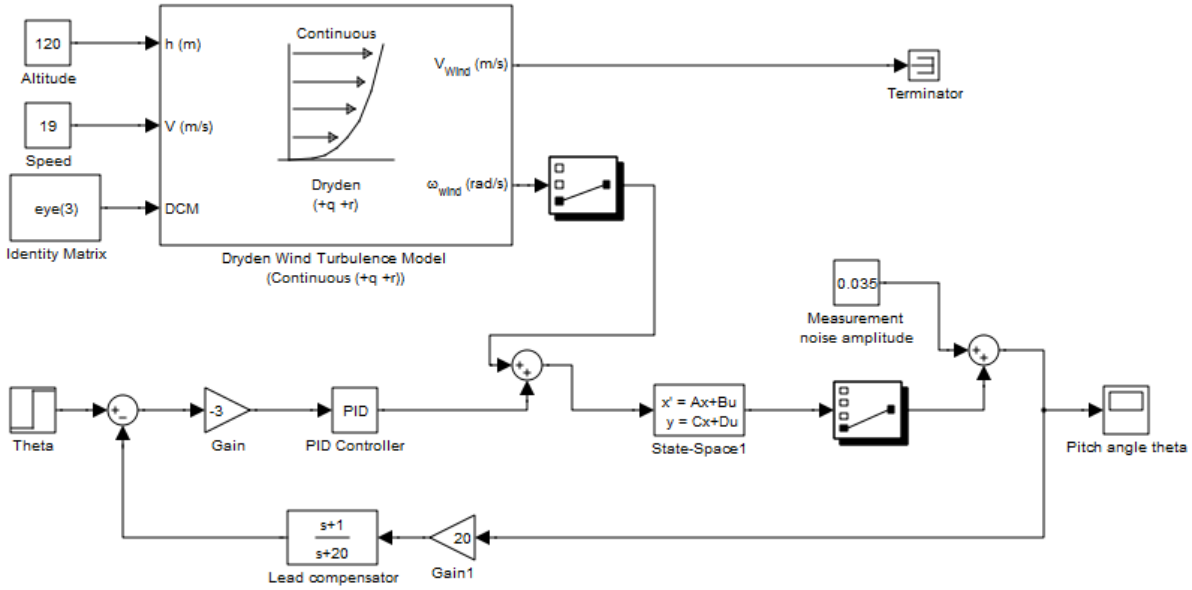


Fig. B.1. PI and Lead compensator Simulink block scheme

B.2. LQR scheme

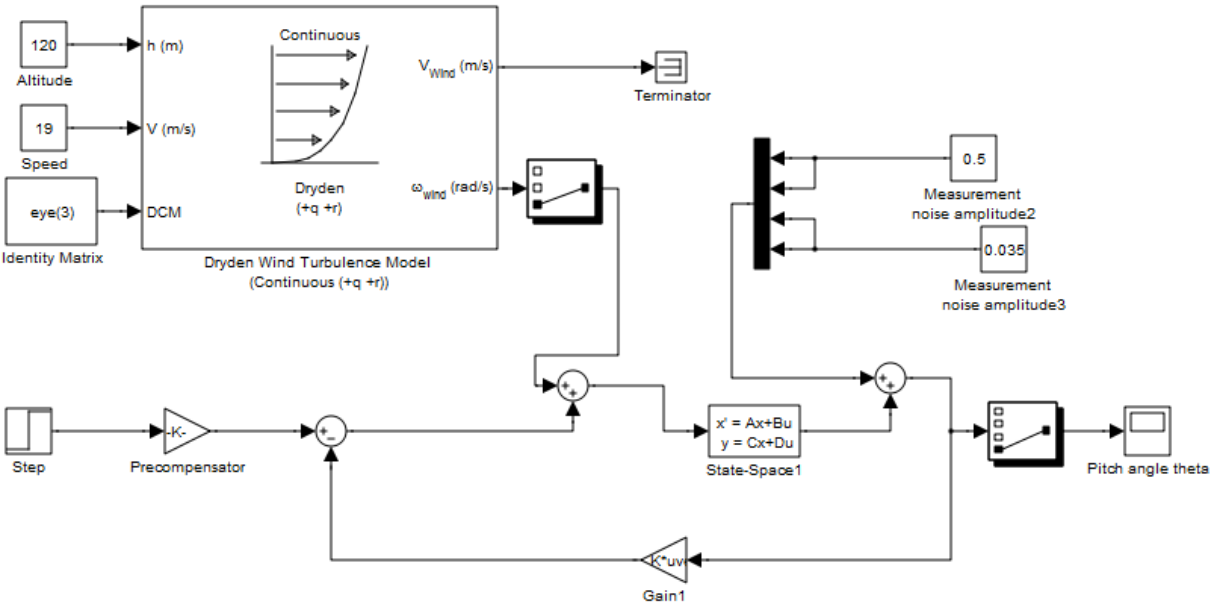


Fig. B.2. LQR Simulink block scheme

B.3. LQG scheme

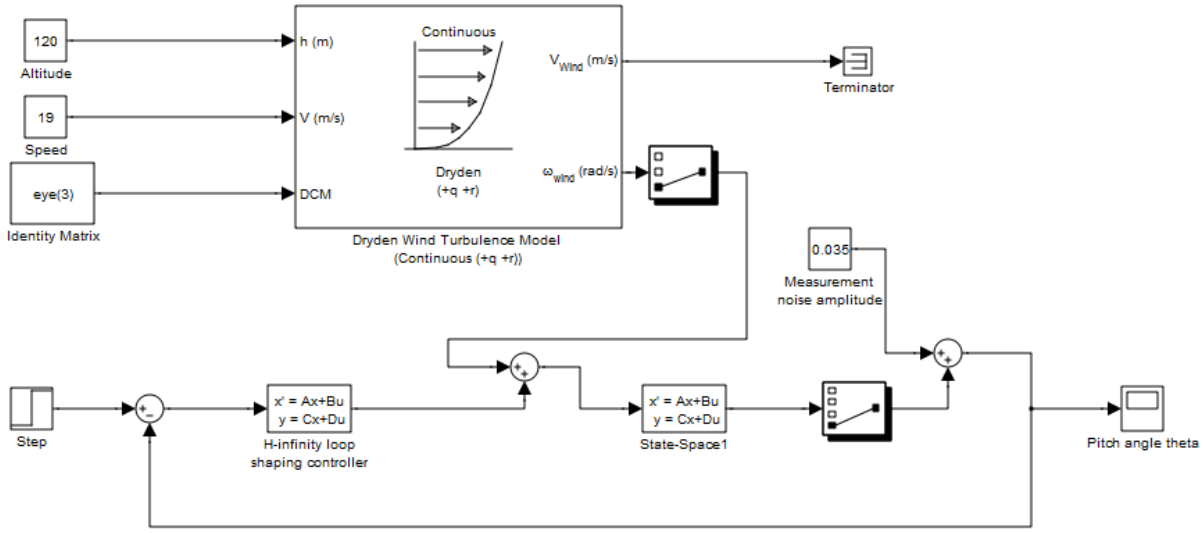


Fig. B.3. LQG Simulink block scheme

B.4. H-infinity loopshaping scheme

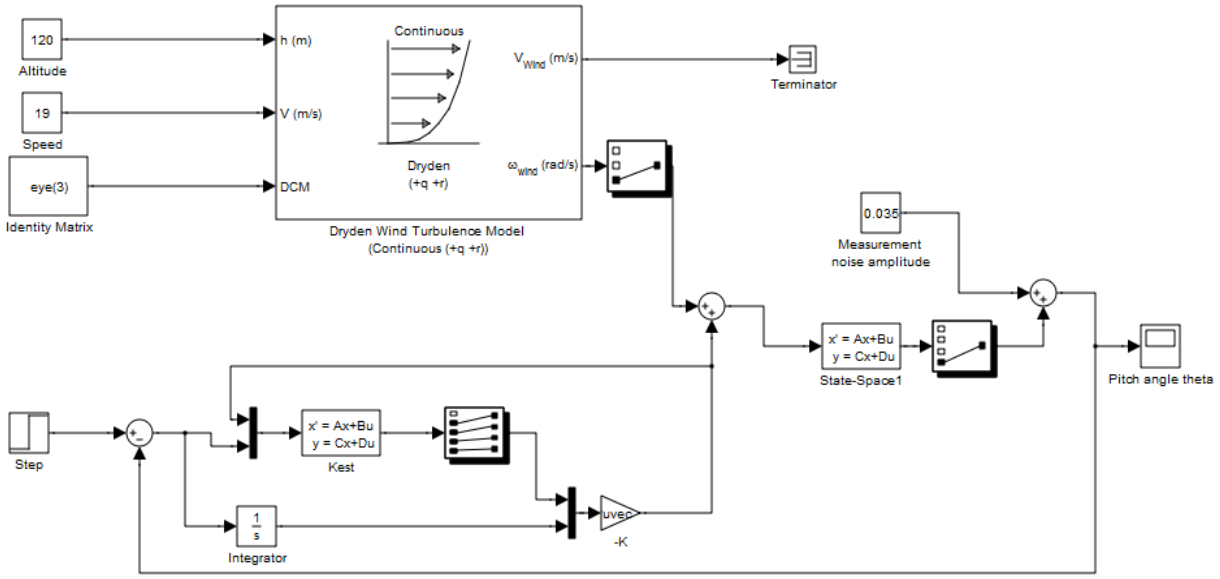


Fig. B.4. H-infinity loopshaping Simulink block scheme

APPENDIX C – *u, w, q* step responses with wind and noise conditions

Note: The units of the y-axis are the International System units: m/s for u and w, and rad/s for q. The x-axis corresponds to the time which is represented in seconds.

Table C.1. Step responses with noise and wind conditions (u) for 19 m/s as reference value

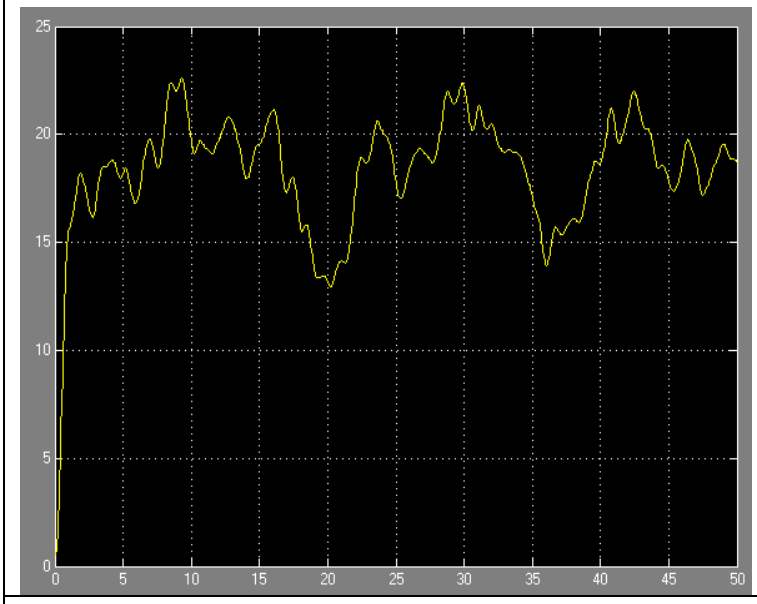
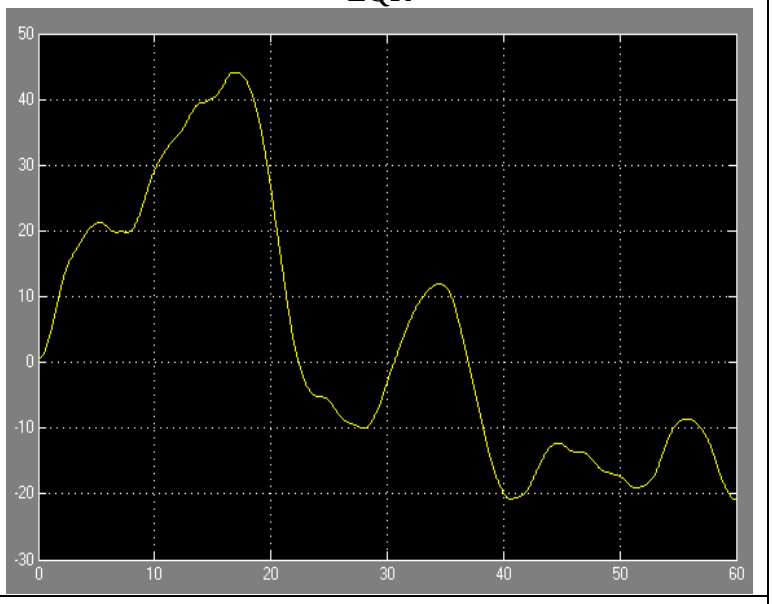
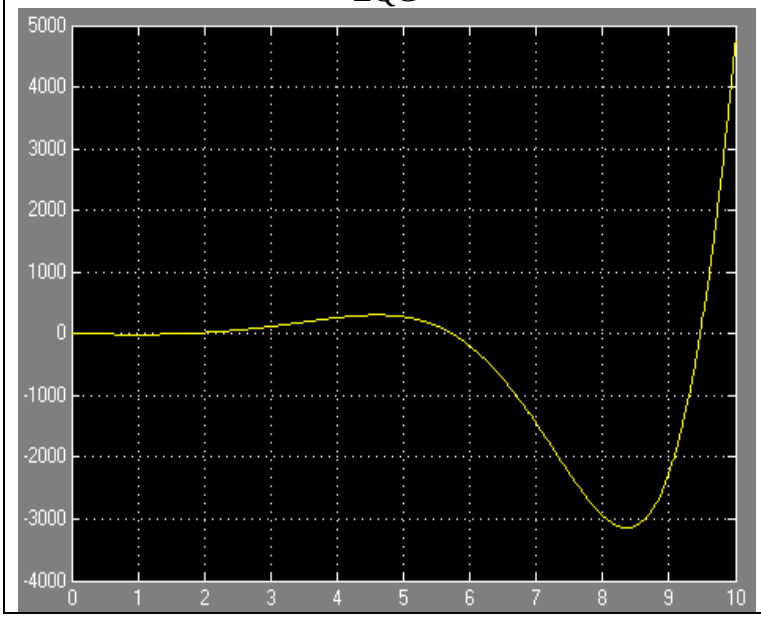
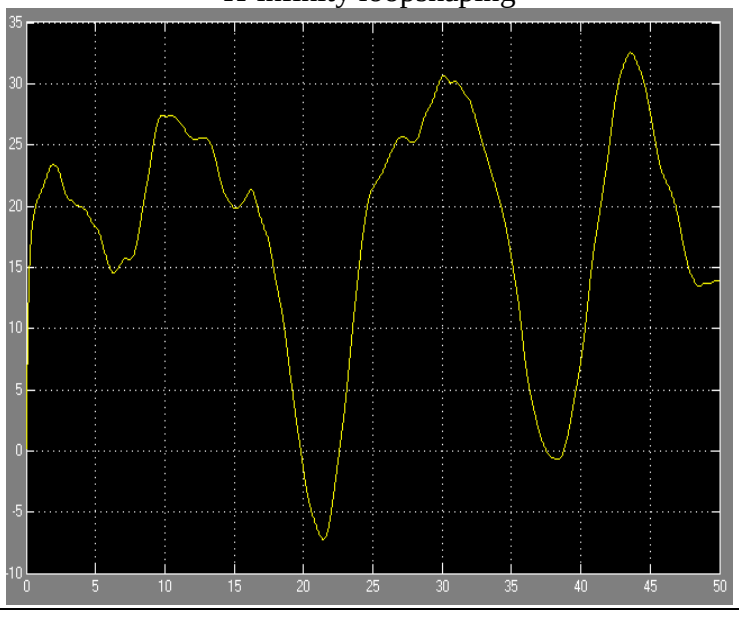
Classical control	LQR
	
LQG	H-infinity loopshaping
	

Table C.2. Step responses with noise and wind conditions (w) for 3 m/s as reference value

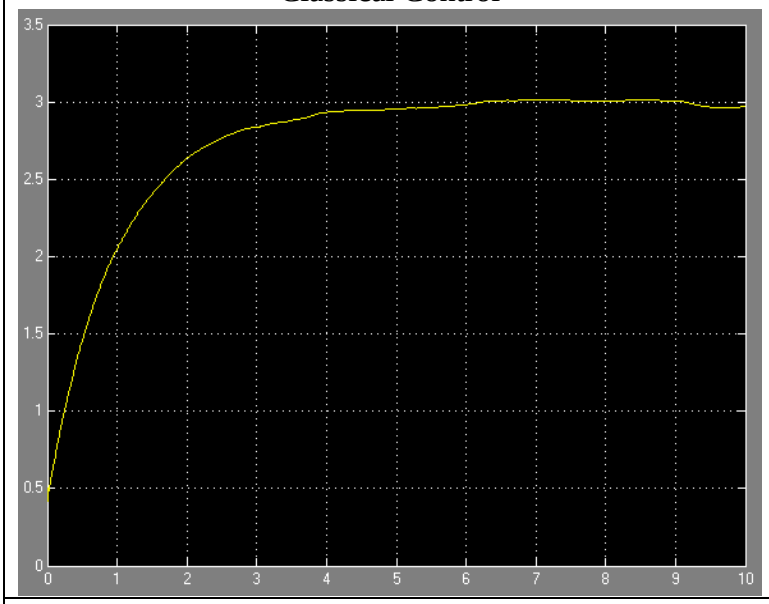
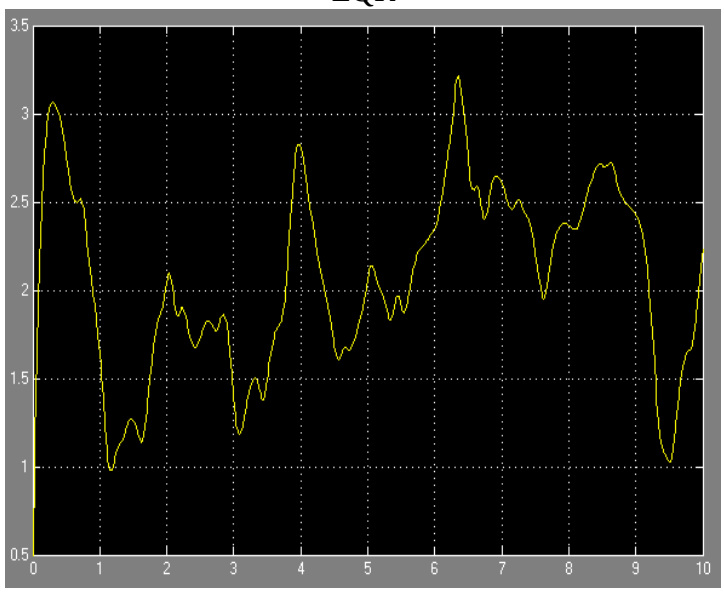
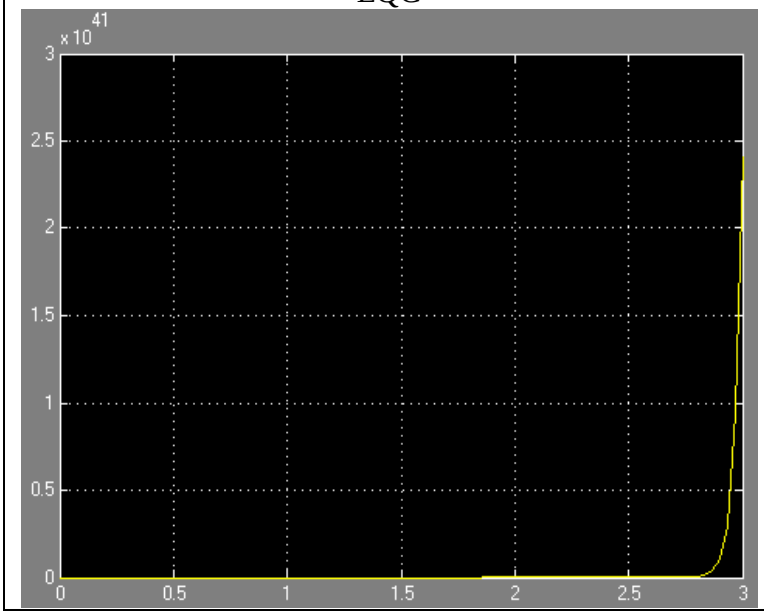
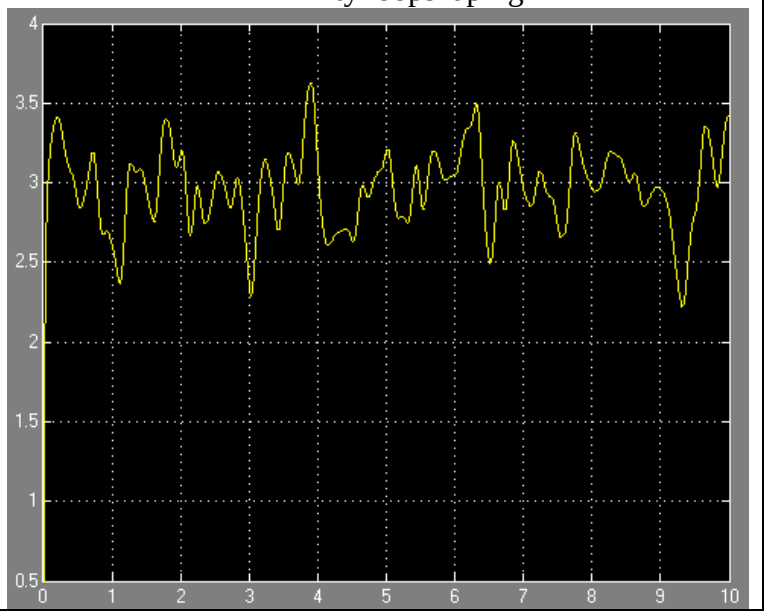
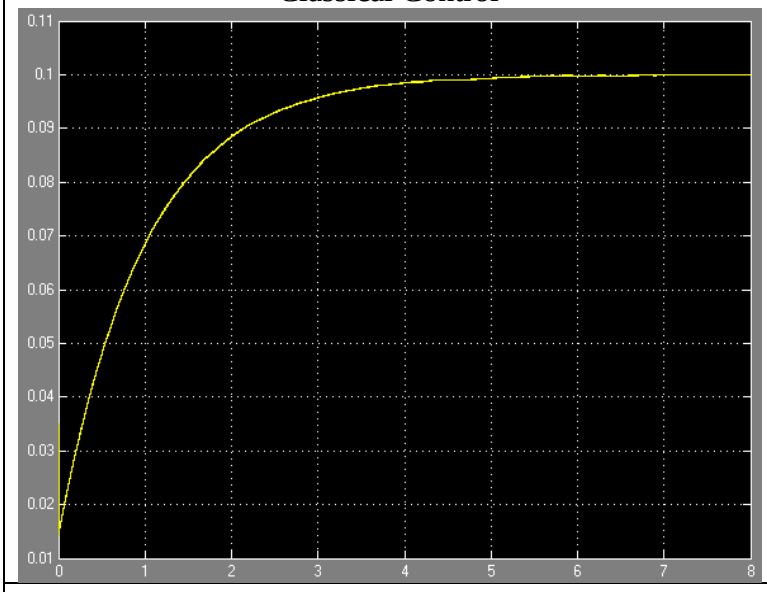
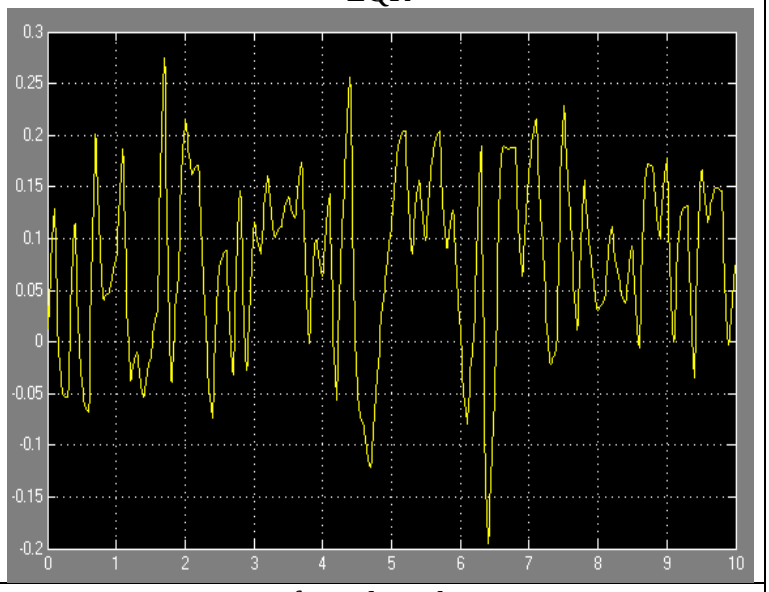
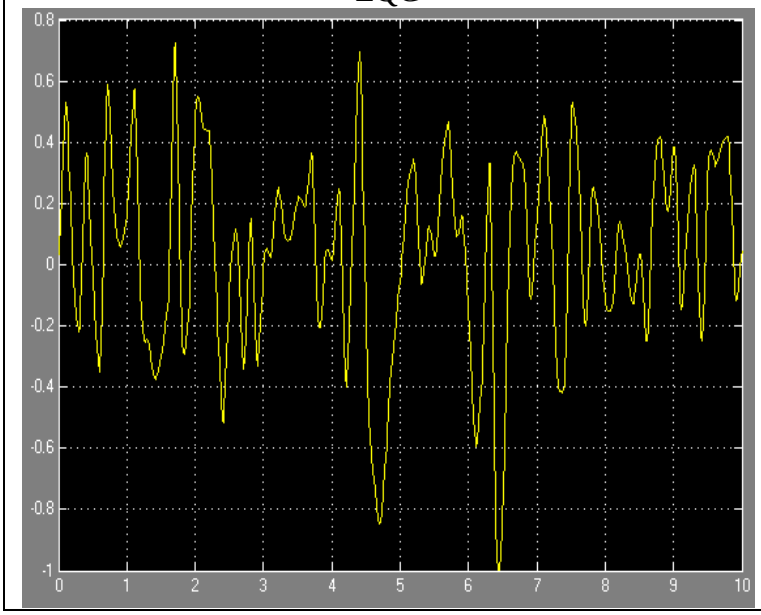
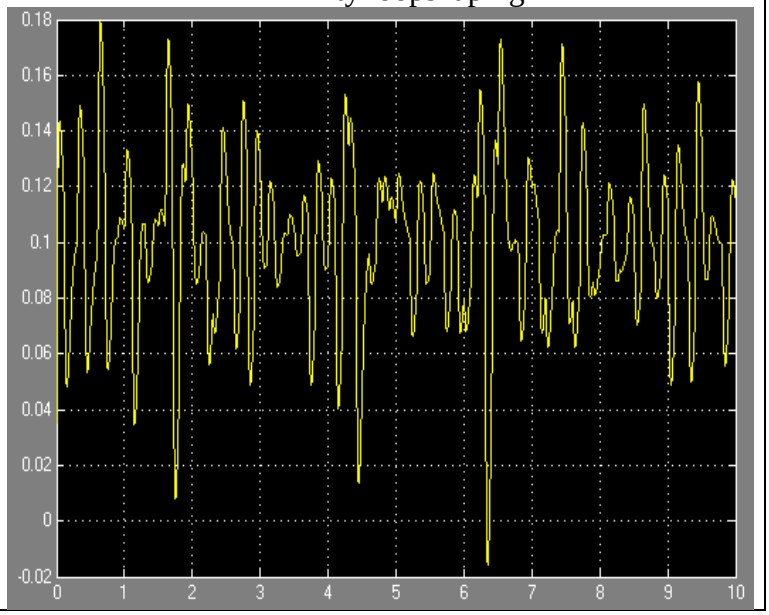
Classical Control	LQR
	
LQG	H-infinity loopshaping
	

Table C.3. Step responses with noise and wind conditions (q) for 0.1 rad/s as reference value

Classical Control	LQR
	
LQG	**H-infinity loopshaping**
	

As we can see in the tables above, the Classical Control scheme proposed is also the best solution for the other autopilots (constant longitudinal speed, constant vertical speed and constant pitch rate) as well it is if we want to maintain a constant pitch angle. The worst case is for the longitudinal speed, this is because we have modelled the wind as headwind, so the effect of the wind disturbance on this variable is more important than for the others.