QUANTITATIVE FEEDBACK DESIGN AND CONSTRUCTION OF A TWO BY TWO SYSTEM WITH LARGE DISTURBANCES

Edward Sidney Boje

Submitted in partial fulfilment of the requirements of the degree of Doctor of Philosophy, in the Department of Electrical Engineering, University of Natal

Durban, March 1989

I hereby certify that all the material incorporated in this thesis is my own unaided work except where specific acknowledgement is made by name or in the form of a reference. The work contained herein has not been submitted for a degree at any other university

Edward Buys
....

Acknowledgements

Meeting my wife Ashley is the best thing that happened to me. Completing this thesis must be about the best thing that has happened to her.[1]

My supervisor, Prof. E. Eitelberg, has been a well behaved input during the execution of this project – enough low frequency gain (ideas, supervision and encouragement) to get the thing done without saturation from amplification of high frequency noise. Prof. Eitelberg rescued me from a tedious literature survey of modern and classical control theory by providing the idea and resources for the construction of the "flying machine" as a tough practical control problem. He has also been a friend in high places.

The staff of the Mechanical and Electrical Engineering Workshops of the University of Durban–Westville provided willing assistance in the construction of the flying machine and in the numerous repairs required to it. Mr S K Moodley (research assistant to Prof E Eitelberg) has been particularly helpful in arranging, organizing, fixing and getting all sorts of time consuming tasks (including the drafting of some of the figures) done.

The financial support of the Council for Scientific and Industrial Research (1987 and 1988), the University of Natal (1987) is acknowledged.

Edward Boje

"Whatever dies, was not mixt equally"
John Donne – *The Good Morrow*

[1]The acknowledgement of what is important in life is inspired by a similar comment in the introduction of Thomas Kailath's *Linear Systems* (1980) and has been in my mind since registering for this degree.

QUANTITATIVE FEEDBACK DESIGN AND CONSTRUCTION OF A TWO BY TWO SYSTEM WITH LARGE DISTURBANCES

Abstract

The quantitative feedback theory (QFT) of Horowitz is theoretically well developed for multivariable systems but there is not sufficient knowledge on its application to practical problems. A "flying machine" consisting of an airframe with two independently controlled sets of wings has been designed and constructed as a 2–input 2–output control problem. The airframe is constrained to move vertically on guide wires and to rotate about a pivot. Air flow over the wings is provided by two 7.5kW fans operated without any attempt at providing non–turbulent flow. The arrangement of the wings is such that in open loop, the dynamic behaviour of the airframe from the rear set of wings to the height is non–minimum phase. Additionally, the airframe is unstable for some flight conditions. This uncertain, non–linear and highly disturbed plant provides an ideal practical environment in which to test controller design theory.

The construction, modelling, parameter estimation and simulation of the flying machine is described. Three different controller structures are discussed, with actual controller designs arrived at from QFT understanding. The controller designs for the flying machine take into account parameter uncertainty and trade off disturbance attenuation against rate and amplitude saturation at the wing angle inputs.

CONTENTS

CHAPTER 1 – INTRODUCTION

This thesis describes the development of a practical system – a "flying machine" – on which to test the quantitative feedback theory (QFT) of Horowitz for multivariable systems. The flying machine has been constructed in the Control laboratory of the Department of Electrical Engineering, University of Durban–Westville, Durban. It consists of an airframe with two independently controlled sets of wings. The airframe is constrained to move vertically on guide wires and to rotate about a pivot. Air flow over the wings is provided by two 7.5kW fans operated without any attempt at providing non–turbulent flow. The arrangement of the wings is such that in open loop, the dynamic behaviour of the airframe from the rear set of wings to the height is non–minimum phase. Additionally, the airframe is unstable for some flight conditions. This uncertain, non–linear and highly disturbed plant provides an ideal practical environment in which to test controller design theory. The controller designs for the flying machine take into account parameter uncertainty and trade off disturbance attenuation against rate and amplitude saturation at the wing angle inputs. This thesis is strongly oriented towards the practical aspects of controller design but also includes some new results for design with plant input constraints.

The construction, modelling, parameter estimation and simulation of the flying machine is described in Chapter 2. Chapter 3 introduces some of the ideas from previously published work on quantitative feedback theory and some new methodology for design to the plant input for multivariable plants. Three different controller structures are discussed, with actual controller designs arrived at from QFT understanding. These designs are presented in Chapters 4, 5 and 6. The controllers were implemented using analog filters and tested. Each of Chapters 4, 5 and 6 presents results for the controller designed in that chapter. Comments on the modern control theory are fashionable in the classical (quantitative) feedback control literature and these are included as part of the conclusion, Chapter 7.

All designs (loop shaping) reported in this thesis were executed with the aid of the CAD package, "DESIGN", written by Professor E Eitelberg and Mr G Sutcliffe.

Although it may seem out of context to present results in the introduction, it must be noted at the outset that feedback control has three fundamental purposes:

1) To reduce the effects of parameter uncertainty, parameter changes and non–linearity on the plant response to (quantitatively defined) acceptable levels.
2) To reduce the effect of external disturbances on the plant output to (quantitatively defined) acceptable levels.
3) To stabilize open–loop unstable plants.
Figure 1.1 shows the height and pitch angle of the flying machine to be described in this thesis when operated without control (i.e. with the wings fixed). During the test, for heights outside about [0.7m, 1.2m] and for pitch angles outside about [–20°, 20°] corrective action was taken to prevent the flying machine from crashing or flipping over by means of a hand held leash. From Figure 1.1 it is reasonable to conclude that at least some (and in fact all) of the reasons for having feedback control are satisfied, justifying what follows.

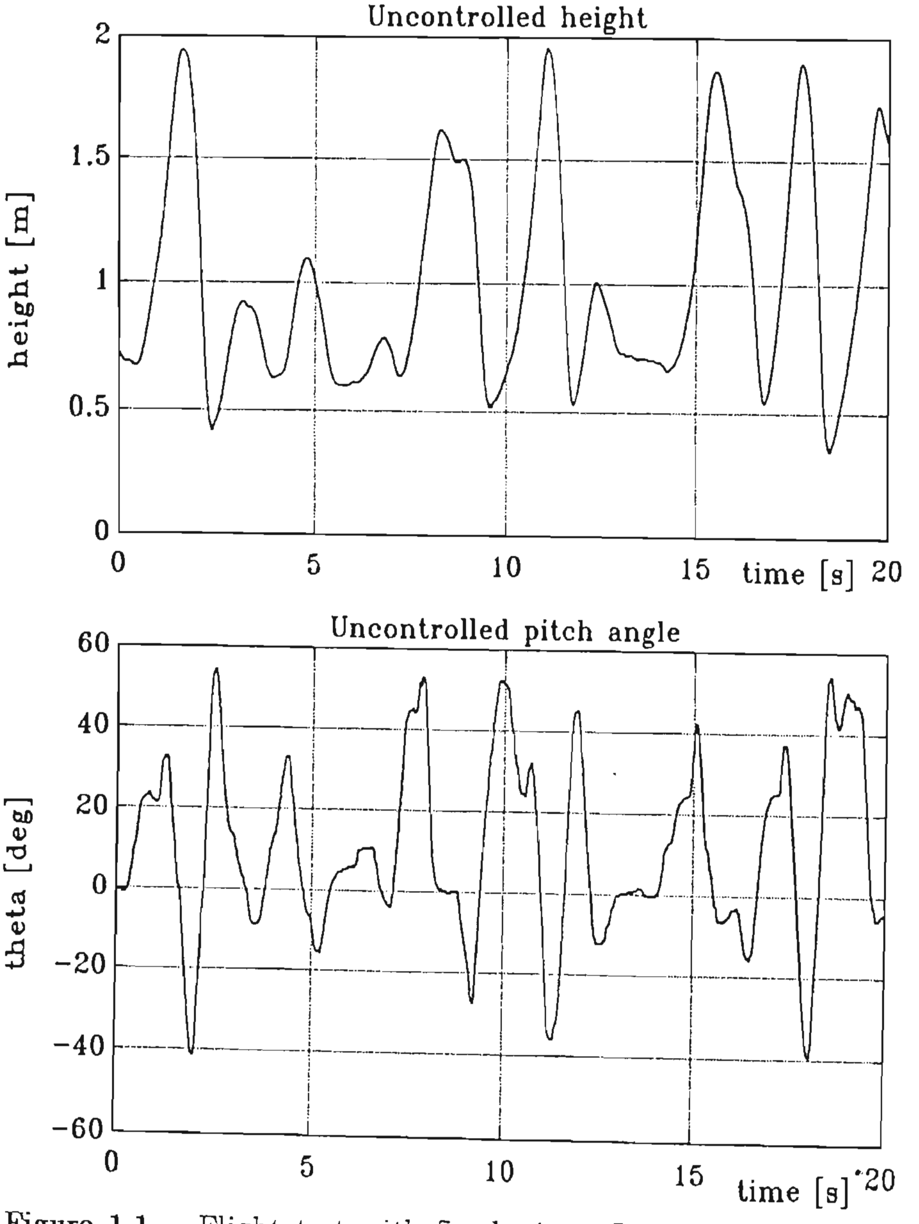


Figure 1.1 – Flight test with fixed wings. Larger excursions were prevented by means of a leash

CHAPTER 2 – CONSTRUCTION, MODELLING AND SIMULATION

Table 2.1 shows the model parameters and symbols which will be used in this and subsequent chapters.

2.1 Construction

2.1.1 Mechanical Construction

The initial design of the flying machine air frame was started by Prof E Eitelberg with the help of two University of Durban-Westville undergraduate students, Mr S Govender and Mr T Naraimsamy. The original design requirements were:

$$\left.\begin{array}{l}\text{Overall mass not to exceed 2 Kg}\\ \text{Ratio of back to front wing areas} \approx 3{:}1\\ \text{Length} \approx 1\text{ m}\\ \text{Wing span} \approx 1\text{ m}\\ \text{Total wing area} \approx 0.2\text{ m}^2\\ \text{Wing aspect ratio (length:width)} \approx 2{:}1\end{array}\right\} \tag{2.1}$$

At an early stage in the project the responsibility for the construction was taken over by the author. The flying machine was constructed by the author and the Electrical and Mechanical workshops of the University of Durban-Westville, with the co-operation of Prof E Eitelberg and his research assistant, Mr S K Moodley. Ad-hoc changes were introduced to the design whenever they were felt necessary. The "as-built" mechanical construction is shown in Figure 2.1a. Figure 2.1b shows a detail of the rear wing mount and actuator. Figure 2.1c is a photograph of the flying machine.

2.1.2 Wing Pivots

As constructed, the wings are pivoted along their geometric centres. In retrospect this design may have been improved as the aerodynamic centre of a flat, rectangular plate is theoretically located at the "quarter chord" (0.25 the wing width (chord) from the leading edge) (Bertin and Smith, 1979, p.114). Had the wings been pivoted nearer their aerodynamic centres, there would have been less lift induced moment on the wing actuators, reducing wear and the deadband which resulted. Pivoting the wings in front of the aerodynamic centres would have allowed the lift induced moment to act as an anti-deadband device, always turning the wings towards the no-lift position and allowing the wing actuators to turn the wings against this moment. This arrangement would result in a gain of

Base model data used in derivation of templates

Term	Minimum	Maximum	Nominal	Description
θ	-15°	15°	0°	pitch angle[1]
v_{wind}	15m/s	20m/s	20m/s	wind speed
$\dot{\theta}$	−1rad/s	1rad/s	0rad/s	rate of change of pitch angle
$\dot{h}$	−1m/s	1m/s	0m/s	rate of change of height

Variables / signals

$\theta(s)$	Pitch angle (angle of attack) w.r.t ground
$H(s)$	Height of pivot point (w.r.t. ground)
$\alpha(s)$	Front wing angle (w.r.t. body)
$\beta(s)$	Back wing angle (w.r.t.body)
$E_t(s)$	External torque disturbance (including static moment imbalance)
$E_\ell(s)$	External lift disturbance (including static force imbalance)

Derived model parameters

Term	Minimum	Maximum	Nominal	Description
J_p			0.14	Eq(2.22), Table 2.2 – inertia (Nm^2)
m_a			1.28	Table 2.2 – mass of airframe (kg)
m_p			0.29	Table 2.2 – mass of pivot (kg)
m_c			0.70	Table 2.2 – mass of counter weights (kg)
m			2.27	Table 2.2 – total moving mass (kg)
μ_t	0.7	1.3	1.2	Eq(2.21), (2.24) – linearised friction coefficient in turning
μ_ℓ	10.6	16.8	16.3	Eq(2.21), (2.23) – linearised friction coefficient in lift
k_t	−1.1	30.2	17.9	Eq(2.21) – linearised torque coeff. deternined by body pitch ∠, θ
k_ℓ	95.9	255.8	242.7	Eq(2.21) – lift coeff. determined by body pitch ∠, θ
k_{tf}	8.5	27.3	25.9	Eq(2.21) – torque coeff. determined by front wing ∠ to body, α
$k_{\ell f}$	19.9	63.6	60.3	Eq(2.21) – lift coeff. determined by front wing ∠ to body, α
k_{tb}	15.6	49.1	43.8	Eq(2.21) – torque coeff. determined by back wing ∠ to body, β
$k_{\ell b}$	60.1	192.2	182.4	Eq(2.21) – lift coeff. determined by back wing ∠ to body, β
A			0.0	Eq(2.21) – small coeff. resulting from linearisation
B			0.0	Eq(2.21) – small coeff. resulting from linearisation

Other design considerations

σ_ℓ^2	132.0 N^2	Section 2.3.2 and eq(2.29) for variance of lift disturbance
η_ℓ	1.9	
$\omega_{x\ell}$	70.0	
σ_t^2	9.5 N^2m^2	Section 2.3.2 and eq(2.30) for variance of torque disturbance
η_t	0.14	
xt	70.0 rad/s	
$\rho_\infty = 1.2$ kg/m^3		Density of air in Durban
$\sigma_\alpha = \sigma_\beta = 0.5$ rad		Section 2.3.3, allowable std deviations of wing positions, eq(2.32a) and of wing rates, eq(2.32b)
$\sigma_{\dot\alpha} = \sigma_{\dot\beta} = 2.0$ rad/s		

[1]Note these are values used in design. Actual values exceed these under some flying conditions.

Table 2.1 – Model parameters and symbols used in this and subsequent chapters

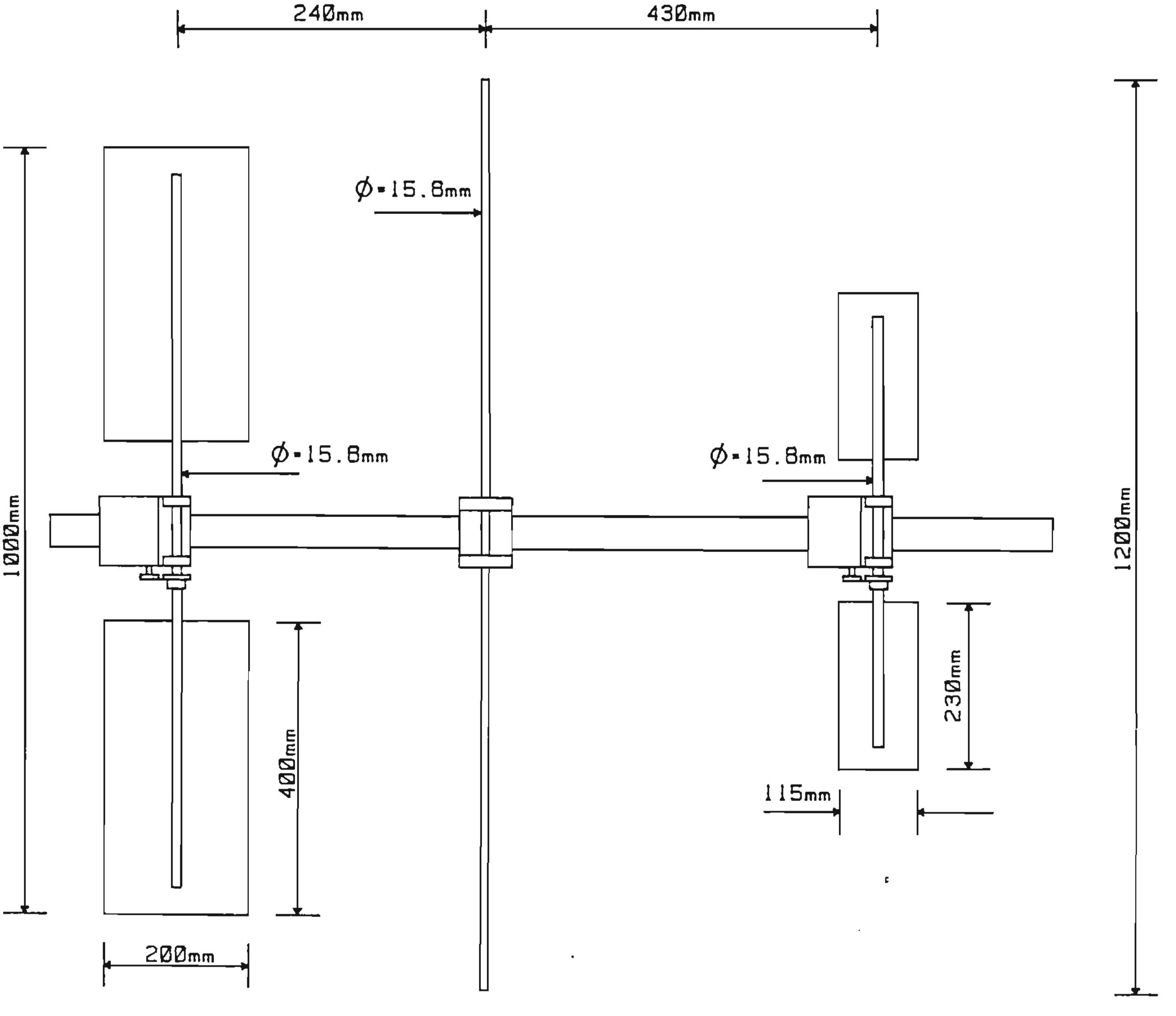


Figure 2.1a As-built mechanical construction of the flying machine

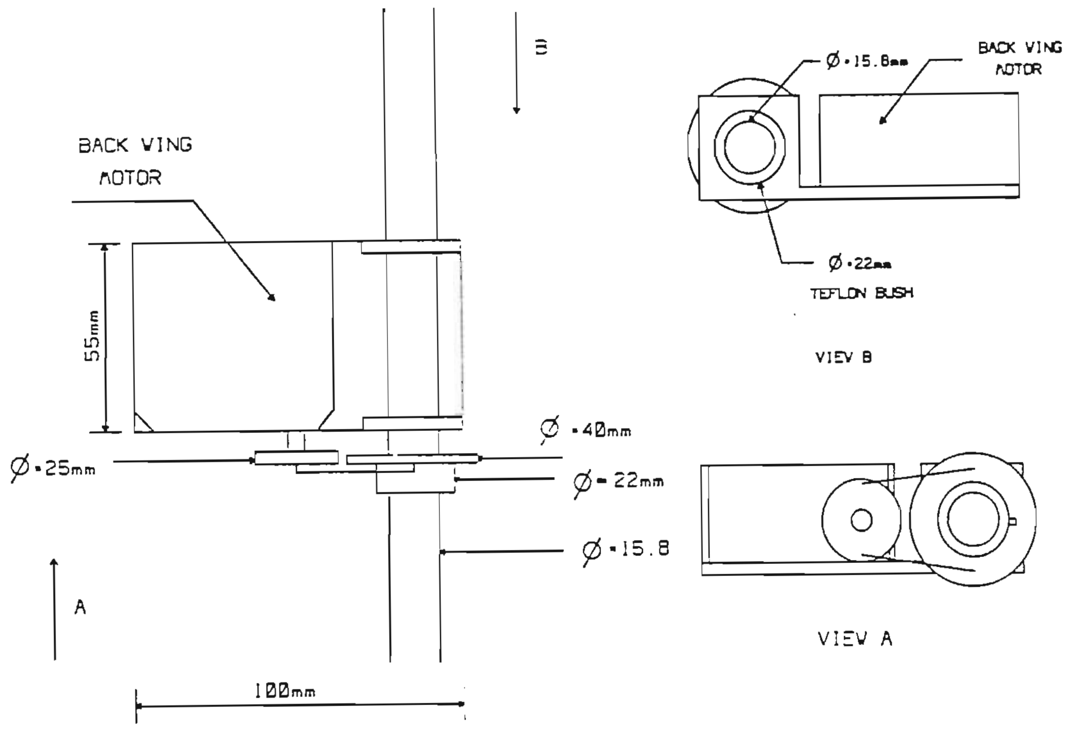
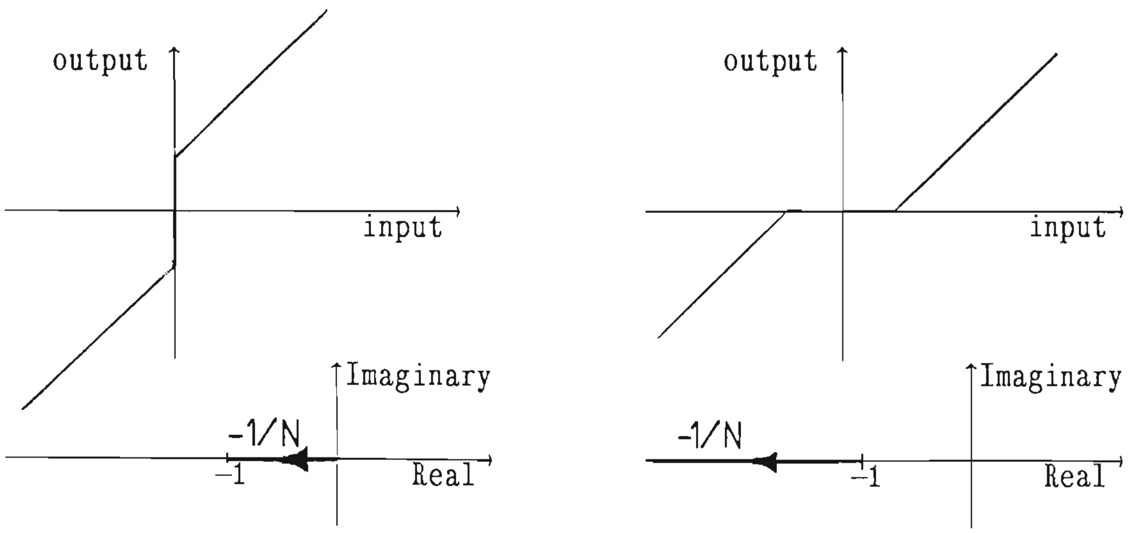


Figure 2.1b – Detail of back wing mount and actuator

Figure 2.1c – Photograph of flying machine

zero around the zero angle of attack position rather than infinite gain - the difference is shown in Figure 2.2, along with sketches of -1/N (negative inverse of nonlinear "describing function") loci indicating why linear design might result in limit cycles for the existing arrangement. Since the wings are turned about 2° to 4° in steady state, detailed describing function analysis would require sinusoidal signals with offset. For details of describing function analysis, see Atherton (1982).



a) Pivot behind c/4 line   Pivot in front of c/4 line

Figure 2.2 Deadband resulting from attachment of wings and corresponding -1/N loci on the Nyquist diagram

2.1.3 Selection and Arrangement of fans

A rule of thumb formula for the maximum lift which could be expected was originally provided by Professor O. Rubin of the University of Pretoria. It is based on Bernoulli's principle - the maximum lift is approximately the product of free stream dynamic pressure (Bertin and Smith, 1979, p.39) and the surface area of the wings,

$$L_{max} \approx q_{\infty} A \tag{2.2}$$

where,

L_{max} = maximum lift

q_{∞} = free stream dynamic pressure

$= \frac{1}{2} \rho_{\infty} v_{\infty}^{2} \approx 0.7\, p_{\infty} M^{2}$ (see Table 2.1)

A = wing area

The free stream dynamic pressure is one half the free stream density, ρ_{∞} (measured far from the wing) and the square of the relative velocity of the wing to the free stream, v_∞^2. (Dr Rubin's formula is equivalent: 0.7 × free stream pressure (p_∞) × Mach number squared, $M^2 = \left[\frac{v_{wing}}{v_{sound}}\right]^2$.)

With a total wing area of 0.2 m^2 and a design mass of 2 Kg, the wind speed required to just lift the air frame is (with L ≈ 20 N, $\rho_\infty \approx 1.225$ Kg/m^3),

$$v_{min} = \sqrt{20/(0.5\times1.22\times0.2)} = 12.8 \text{ m/s} \tag{2.3}$$

To achieve in excess of this wind speed over an area 1 m wide and 2 m high, two fans were purchased from Luft Industries with the following specifications:
a) 1000 mm ϕ fans with 7 blades (30° blade pitch angle).
b) 7.5 kW, 1400 r.p.m. 380 V, 3 phase motors.
c) Each fan is rated at 17 m^3/s in free air.

Figure 2.3 shows an approximate air flow cross section about 1 m from the fans from corrected measurements made with a hand held anemometer (which over read by a factor of 1.38 compared to an alcohol manometer in a wind tunnel - lift predicted from measurements based on this instrument would therefore be in error by a factor of nearly 2!).

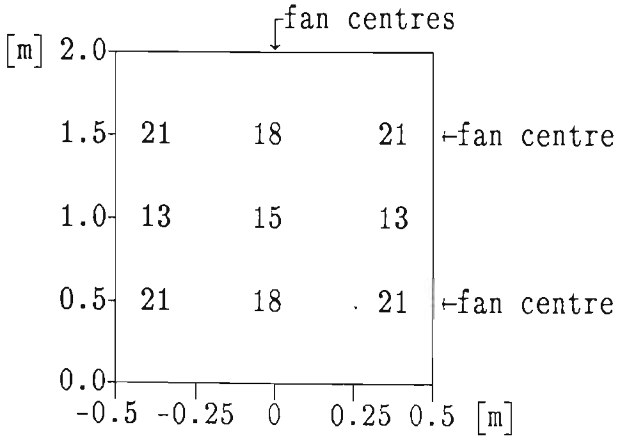


Figure 2.3 – Very approximate wind speeds - corrected from hand held anemometer readings

After discussions with Professor L. Roberts (Department of Mechanical Engineering, University of Natal) it was decided not to try guiding the air stream in any way. (The same discussion highlighted the importance of fan inlets for good airflow and high fan efficiency - standard inlet cones were purchased with the fans.) Apparently badly designed "straighteners" can worsen the free stream airflow pattern. During experimentation, a design which flew one day did not fly the next day due to changed airflow pattern caused by the removal of spare furniture in the laboratory. The presence of spectators also affected the air flow pattern. The equipment layout which resulted in the least disturbed airflow pattern is shown in Figure 2.4 below.

Figure 2.4a shows the general layout of the laboratory area. Figure 2.4b and 2.4c are photographs of the arrangement of "dirty" and "control" laboratories respectively.

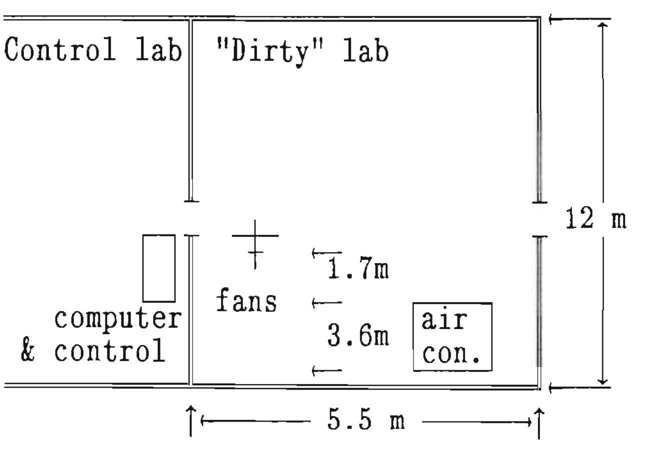


Figure 2.4a Laboratory equipment layout

Figure 2.4b – Layout of laboratory showing position of flying machine and fans

Figure 2.4c – Layout of control laboratory showing control and measurement equipment. (Flying machine is next door)

2.1.4 Actuators

Futaba FP-S128 and Sataba FP-S134 model aeroplane servos were selected for the front and back wing actuators respectively. Specification sheets were available for only the FP-S128 - it has a rated torque of 0.035 Nm which allows the front wing to be accelerated at ($J_{\text{front wing}} = 1.14\times10^{-4}$ Kg m^2),

$$\ddot{\alpha} = 0.035 \;/\; 1.14\times10^{-4} = 307 \text{ rad/s}^2, \qquad (2.4)$$

reaching a terminal velocity of 200°/s in 11 ms.

As supplied, the servos operate via a pulse width modulated input signal which can easily be interfaced to a model aeroplane radio transmitter. A voltage to pulse - width modulator circuit was built based on LM555 timers. Because the modulator circuit was not sufficiently linear and the holding torque was not adequate, the drive electronics of the servos were replaced with a push-pull amplifiers. The actuators have essentially an inner loop feedback which reduces the uncertainty of the outer loop. This inner loop was closed with large bandwidth. The wing position signals were brought out of the motor casings. Circuit diagrams are included in Appendix 1.

As a result of power supply dependence and the sharing of 0V (common) conductors (in order to save weight and keep the wiring harness joining the flying machine to the instrumentation flexible) some non-minimum phase problems appeared which limited the achievable loop gain of the local wing controller feedback circuits severely. These problems were analyzed in a paper in the Transactions of the South African Institute of Electrical Engineers (Boje and Eitelberg, 1989) and solved (by increasing the current rating of the power supplies and separating power and instrumentation supplies and their 0V conductors. The paper is included as Appendix 2.

2.1.5 The Frame

The air frame is constrained to move vertically on guide wires and to rotate about a pivot. For this purpose, a frame was constructed using mild steel angle and tubing. The air frame is pivoted and the pivot slides on guide-wires. The pivot is kept horizontal by means of flexible wires (stainless steel angling tracer wire) passed around pulleys rather like those of a drawing board rule. See Figure 2.5

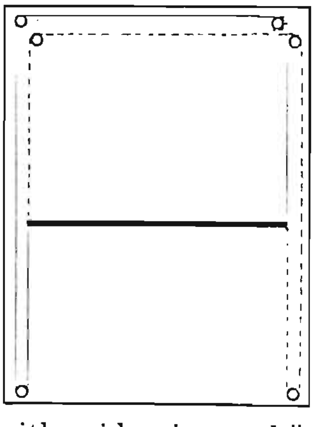


Figure 2.5 Frame with guide wires and "parallel arm"

2.2 Flying Machine - First principles model

2.2.1 Dynamic model

The dynamic model of the flying machine was obtained (after discussion with Prof. E Eitelberg) by considering the airframe as a rigid body and the pivot–counterweight system as a second rigid body, joined to the airframe at the pivot. Applying Newton's second law (D'Souza and Garg 1984, p95) to each body,

$$\frac{d\ m\underline{v}_c}{dt} = \underline{F} \qquad \text{translational momentum} \tag{2.5a}$$

$$\frac{d\ \underline{J}_c\underline{\omega}_c}{dt} = \underline{M}_c \qquad \text{rotational momentum} \tag{2.5b}$$

where,

- m is the mass of the rigid body
- $\underline{v}_c$ is the translational velocity of the centre of mass of the rigid body
- $\underline{F}$ is the the net external force acting on the rigid body.
- $\underline{J}_c$ is the inertia of the rigid body,calculated with respect to the centre of mass
- $\underline{\omega}_c$ is the angular velocity of the centre of mass of the rigid body
- $\underline{M}_c$ is the net external moment calculated with respect to the centre of mass, acting on the rigid body

As the flying machine is constrained to move on the vertical slide and to rotate about the pivot, there are only two degrees of freedom. With reference to Figure 2.6 (which has the x–axis vertically and the y–axis horizontally),

Force balance of vertical slide from movement of pivot–counterweight system:

$$(m_p+m_c)\ddot{h} = -(m_p-m_c)g - \mu_{\ell o}\dot{h} - R_x \tag{2.6a}$$

where

- m_p is the mass of the pivot beam
- m_c is the mass of the counterweight
- $\ddot{h}$, $\dot{h}$, and h are is the vertical acceleration, velocity and position of the pivot
- g is the acceleration due to gravity
- $\mu_{\ell o}$ models all the sliding friction
- R_x is vertical component of reaction force on airframe

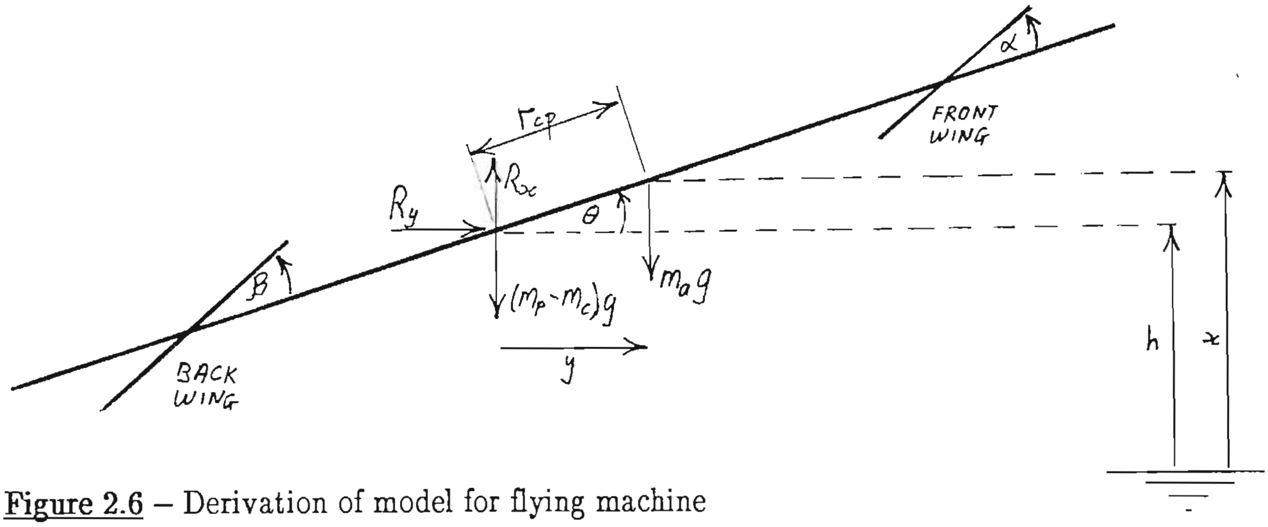


Figure 2.6 – Derivation of model for flying machine

Force balance of airframe with respect to its centre of mass:

$$m_a \ddot{x} = -m_a g + (\text{lift applied by wings}) + R_x \tag{2.6b}$$

$$m_a \ddot{y} = - (\text{drag applied by wings}) + R_y \tag{2.6c}$$

where

m_a is the mass of the airframe

$\ddot{x}$ is the vertical acceleration of the centre of mass of the airframe

$\ddot{y}$ is the horizontal acceleration of the centre of mass of the airframe

R_y is horizontal component of reaction force on airframe

Moment balance of the airframe with respect to its centre of mass:

$$J_c \ddot{\theta} = -\mu_{to} \dot{\theta} + (\text{torque applied by wings}) - (\text{reaction torque of wings}) + R_y\, r_{cp} \sin(\theta) - R_x\, r_{cp} \cos(\theta) \tag{2.6d}$$

where

J_c is the moment of inertia of the airframe along its length, calculated with respect to its centre of mass

$\ddot{\theta}$, $\dot{\theta}$ and θ are the angular acceleration, velocity and position (pitch angle) of the airframe with respect to the ground

μ_{to} is the angular friction of the pivot

r_{cp} is the distance between the pivot, P, and the centre of mass, C (r_{cp}>0 in Figure 2.6)

Because of the constraints imposed by the pivot and guide wire, the following geometry can be used to remove the dependence on x and y above,

$$x = h + r_{cp}\sin(\theta), \qquad \dot{x} = \dot{h} + r_{cp}\,\dot{\theta}\cos(\theta)$$
$$\ddot{x} = \ddot{h} + r_{cp}\,(\ddot{\theta}\cos(\theta) - \dot{\theta}^{2}\sin(\theta)) \tag{2.6e}$$

$$y = r_{cp}\cos(\theta), \qquad \dot{y} = -\,r_{cp}\,\dot{\theta}\sin(\theta)$$
$$\ddot{y} = -r_{cp}\,(\ddot{\theta}\sin(\theta) + \dot{\theta}^{2}\cos(\theta)) \tag{2.6f}$$

The equations of motion in terms of h(t) and θ(t) are therefore,

$$(m_p+m_c+m_a)\ddot{h} = -\,r_{cp}\,m_a\,(\ddot{\theta}\cos(\theta) - \dot{\theta}^{2}\sin(\theta)) - (m_a+m_p-m_c)g$$
$$-\,\mu_{\ell o}\dot{h} + (\text{lift applied by wings}) \tag{2.7a}$$

$$J_c\ddot{\theta} = -\mu_{to}\dot{\theta} + (\text{torque applied by wings}) - (\text{reaction torque of wings}) +$$
$$r_{cp}\sin(\theta)\,\{(\text{drag applied by wings}) - m_a\,r_{cp}\,(\ddot{\theta}\sin(\theta) + \dot{\theta}^{2}\cos(\theta))\} +$$
$$r_{cp}\cos(\theta)\,\{(\text{lift applied by wings}) - m_a(\ddot{h}+r_{cp}(\ddot{\theta}\cos(\theta)-\dot{\theta}^{2}\sin(\theta))) - m_a g\} \tag{2.7b}$$

Rearranging and simplifying eq(2.7b).

$$J_c\ddot{\theta} = -\mu_{to}\dot{\theta} + (\text{torque applied by wings}) - (\text{reaction torque of wings}) +$$
$$r_{cp}\{(\text{drag applied by wings})\sin(\theta) + (\text{lift applied by wings})\cos(\theta)\}$$
$$-\,r_{cp}\cos(\theta)\,m_a\,(g+\ddot{h}) - m_a\,r_{cp}^{2}\,\ddot{\theta}$$

or,

$$J_p\,\ddot{\theta} = -\mu_{to}\dot{\theta} + (\text{torque applied by wings w.r.t pivot})$$
$$-\,(\text{reaction torque of wings}) - r_{cp}\cos(\theta)\,m_a\,(g+\ddot{h}) \tag{2.7c}$$

where,

J_p is the moment of inertia of the airframe about the pivot, calculated along the length of the body, $J_p = J_c + m_a\,r_{cp}^{2}$

The parallel axis theorem (*ibid* p.88) has been used in deriving eq(2.7c). Note that the torques due to rotational friction and wing reaction do not depend on the point at which the moment balance is calculated. The system has two degrees of freedom and the dynamic model, eq(2.7) is consequently fourth order.

2.2.2 Lift, Drag and Wind vectors

Standard aerodynamic theory (Bertin and Smith, 1979) shows that the lift and drag produced by an aerofoil is proportional to the so-called free-stream dynamic pressure, $q_\infty = \frac{1}{2}\rho_\infty v_\infty^2$, the area of the wing and the wing angle of attack (when this is small). For a flat plate inclined at α radians, the theoretical section coefficient of lift, C_ℓ, is,

$$C_\ell = \frac{\ell}{0.5\ \rho_\infty\ v_\infty^2} = 2\,\pi\,\alpha \tag{2.8}$$

where ℓ is the lift per unit length per unit chord (width)

Experimental values are found in Section 2.3. The lift and drag are therefore modelled by, (coefficient) × (wing area) × (wind speed)^2. Using subscript dot ($\cdot$) as a generic subscript and subscripts f and b for the front and back wings respectively, the combined lift force of the front and back wings is given by

$$\begin{aligned} F = {} & v_f^2\, a_f\, \{\cos(\text{wind} \angle \text{ to ground})\, c_{\ell f}(\text{front wing} \angle \text{ to wind}) + \\ & \sin(\text{wind} \angle \text{ to ground})\, c_{df}(\text{front wing} \angle \text{ to wind})\, \} \\ & + v_b^2\, a_b\, \{\cos(\text{wind} \angle \text{ to ground})\, c_{\ell b}(\text{back wing} \angle \text{ to wind}) + \\ & \sin(\text{wind} \angle \text{ to ground})\, c_{db}(\text{back wing} \angle \text{ to wind})\, \} \end{aligned} \tag{2.9}$$

where,

F is the total vertical lift force due to the wings

$v_\cdot$ is the wind velocity at the respective wing

$a_\cdot$ is the wing area

$c_{\ell\cdot}$ is the section lift coefficient

$c_{d\cdot}$ is the section drag coefficient

The torque provided by the wings is,

$$\begin{aligned} T = {} & v_f^2\, a_f\, r_{fp}\, \{\cos(\text{body} \angle \text{ to wind})\, c_{\ell f}(\text{front wing} \angle \text{ to wind}) + \\ & \sin(\text{body} \angle \text{ to wind})\, c_{df}(\text{front wing} \angle \text{ to wind})\, \} \\ & - v_b^2\, a_b\, r_{bp}\, \{\cos(\text{body} \angle \text{ to wind})\, c_{\ell b}(\text{back wing} \angle \text{ to wind}) + \\ & \sin(\text{body} \angle \text{ to wind})\, c_{db}(\text{back wing} \angle \text{ to wind})\, \} \\ & - \{\text{wing reaction torque}\} \end{aligned} \tag{2.10}$$

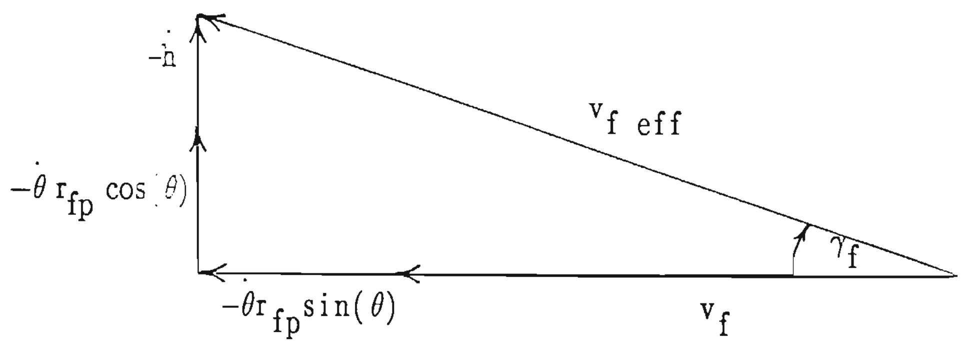
where,

$r_{\cdot p}$ is the distance between the wing aerodynamic centre and the pivot

The aerodynamic centre of a flat plate is theoretically at the "quarter chord" $\frac{1}{4}$ of width of the wing, measured from the leading edge.

The measured wind speed from the fans (at an angle of 0° to the ground) were initially used for the wind speed, $v_{\cdot}^2$, terms. It was observed however that the rotational friction increased substantially when the fans were on. This was initially attributed to increased bearing force on the Teflon bushes which is partially correct but as was pointed out by Prof. E Eitelberg, the simple model above can account for some of the additional friction if the wind vector is considered more carefully - see Figures 2.7 and 2.8. The effective wind speed is calculated as the vector sum of the actual wind speed and model states:

<u>Front wing:</u>



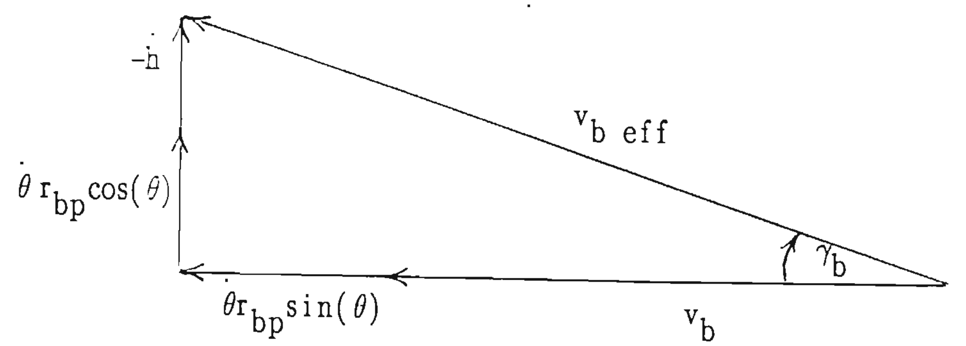
<u>Figure 2.7</u> – Front wing wind vector (state dependent terms exaggerated)

$$v_{f\ eff}^2 = (v_f - \dot{\theta}\, r_{fp}\sin(\theta))^2 + (-\dot{h} - \dot{\theta}\, r_{fp}\cos(\theta))^2 \tag{2.11}$$

at an angle (effective wind angle to the ground),

$$\gamma_f = \arctan\left[\frac{-\dot{h} - \dot{\theta}\, r_{fp}\cos(\theta)}{v_f - \dot{\theta}\, r_{fp}\sin(\theta)}\right] \tag{2.12}$$

<u>Back wing:</u>



<u>Figure 2.8</u> – Back wing wind vector (state dependent terms exaggerated)

$$v_{b\ eff}^2 = (v_b + \dot{\theta}\, r_{bp}\sin(\theta))^2 + (-\dot{h} + \dot{\theta}\, r_{bp}\cos(\theta))^2 \tag{2.13}$$

at an angle (effective wind angle to the ground),

$$\gamma_b = \arctan\left[\frac{-\dot{h} + \dot{\theta}\, r_{bp}\cos(\theta)}{v_b + \dot{\theta}\, r_{bp}\sin(\theta)}\right] \tag{2.14}$$

The model for the flying machine is therefore,

$$\begin{aligned}(m_p+m_c+m_a)\ddot{h} + \mu_{\ell o}\dot{h} + (m_a+m_p-m_c)g + m_a r_{cp}(\ddot{\theta}\cos(\theta)-\dot{\theta}^2\sin(\theta)) =\\ v_{f\ eff}^2\, a_f\, \{\cos(\gamma_f)\, c_{\ell f}(\theta+\alpha+\gamma_f) + \sin(\gamma_f)\, c_{df}(\theta+\alpha+\gamma_f)\} +\\ v_{b\ eff}^2\, a_b\, \{\cos(\gamma_b)\, c_{\ell b}(\theta+\beta+\gamma_b) + \sin(\gamma_b)\, c_{db}(\theta+\beta+\gamma_b)\}\\ = F(\theta,\alpha,\beta,h,\dot{h},\dot{\theta})\end{aligned} \tag{2.15}$$

$$\begin{aligned}J_p\ddot{\theta} + \mu_{to}\dot{\theta} + m_a r_{cp}\cos(\theta)\,(\ddot{h} + g) =\\ v_{f\ eff}^2\, a_f\, r_{fp}\{\cos(\theta+\gamma_f)\, c_{\ell f}(\theta+\alpha+\gamma_f) + \sin(\theta+\gamma_f)\, c_{df}(\theta+\alpha+\gamma_f)\} -\\ v_{b\ eff}^2\, a_b\, r_{bp}\{\cos(\theta+\gamma_b)\, c_{\ell b}(\theta+\beta+\gamma_b) + \sin(\theta+\gamma_b)\, c_{db}(\theta+\beta+\gamma_b)\} -\\ (J_\alpha\ddot{\alpha} + \mu_\alpha\dot{\alpha} + J_\beta\ddot{\beta} + \mu_\beta\dot{\beta}) = T(\theta,\alpha,\beta,h,\dot{h},\dot{\theta})\end{aligned} \tag{2.16}$$

The configuration of the flying machine used throughout this work allows the following simplifications.

2.2.3 Wing reaction torques

For the front wing, after linearisation (Section 2.2.5) the model is,

$$\text{Front Torque} = (-J_\alpha s^2 - \mu_\alpha s + k_{tf})\alpha(s) \tag{2.17}$$

The reaction torque on the body caused by the turning (friction), $\mu_\alpha s$, and acceleration (inertia), $J_\alpha s^2$, of the front wing is much smaller than the torque induced by the wind (k_{tf}). The resulting numerator is non-minimum phase but (assuming small values of μ_α) the right hand plane zero is (for the nominal plant – see Table 2.1) at

$$\sqrt{k_{tf0}/J_\alpha} = \sqrt{25.9/1.141\times10^{-4}} = 476 \text{ rad/s} \tag{2.18}$$

about one decade beyond the frequency range in which the model is expected to be accurate.

For the back wings the model is,

$$\text{Back Wing Torque} = -(J_\beta s^2+\mu_\beta s+k_{tb})\beta(s) \tag{2.19}$$

The same as the front wings except the behaviour is minimum phase and has (at the nominal gain k_{tb0}) a corner frequency,

$$\sqrt{k_{tb0}/J_\beta} = \sqrt{43.8/1.0622\times10^{-3}} = 203 \text{ rad/s} \tag{2.20}$$

with damping determined by the friction μ_β. Although the corner frequency is lower than for the front wings, these numerator terms being minimum phase, should not be a problem and will therefore be neglected.

2.2.4 Co–location of the pivot and centre of mass

All terms resulting from the distance between the centre of mass and the pivot, r_{cp} = 0.01m (Table 2.2 below) are neglected as the pivot and centre of mass can be regarded as being co–located. This simplifies the model considerably. These terms are however included in the non-linear simulation so that other airframe configurations can be examined. The pivot is also situated approximately 0.01m above the centre of mass of the airframe. This has been neglected from the start as it is insignificant.

2.2.5 The linearised model

The linear feedback design is undertaken by considering small signal linearisations of the non-linear model eq(2.15) and eq(2.16) about all reasonable (not necessarily steady state) operating points. This means that the design is not strictly quantitative as it might be if design had utilized the "linear equivalent" plant set (Horowitz, 1982b). Checking the correctness of this approach to "quantitative" design by simulation is discussed in Section 2.4. After the simplifications above, the linearised model is,

$$\underbrace{\begin{bmatrix} Js^2+\mu_t s+k_t & As \\ -(Bs+k_\ell) & ms^2+\mu_\ell s \end{bmatrix}}_{\underline{D}(s)} \underbrace{\begin{bmatrix} \theta(s) \\ H(s) \end{bmatrix}}_{\underline{Y}(s)} = \underbrace{\begin{bmatrix} k_{tf} & -k_{tb} \\ k_{\ell f} & k_{\ell b} \end{bmatrix}}_{\underline{N}(s)} \underbrace{\begin{bmatrix} \alpha(s) \\ \beta(s) \end{bmatrix}}_{\underline{U}(s)} + \underbrace{\begin{bmatrix} E_t(s) \\ E_\ell(s) \end{bmatrix}}_{\underline{E}(s)} \tag{2.21}$$

with signals,

$\theta(s)$, $H(s)$	Angle of attack and Height (w.r.t.ground)
$\alpha(s)$, $\beta(s)$	Front and Back wing angles (w.r.t.body)
$E_t(s)$, $E_\ell(s)$	External Torque and Lift disturbances

and parameters,

J	inertia with respect to the pivot
m	total moving mass, $m_a + m_p + m_c$
μ_t, μ_ℓ	linearised friction coeffs (turning and lift)
k_t, k_ℓ	linearised torque and lift coefficients (determined by body $\angle$ of attack, θ)

k_{tf}, $k_{\ell f}$	linearised torque and lift coefficients (determined by front wing ∠ to body, α)
k_{tb}, $k_{\ell b}$	linearised torque and lift coefficients (determined by back wing ∠ to body, β)
A, B	coefficients with small magnitudes resulting from linearisation

The linearised coefficients at any operating point, op., are calculated via,

$$\mu_t = \mu_{to} - \left.\frac{\partial T}{\partial \dot{\theta}}\right|_{o.p.} \qquad \mu_\ell = \mu_{\ell o} - \left.\frac{\partial F}{\partial \dot{h}}\right|_{o.p.}$$

$$k_t = -\left.\frac{\partial T}{\partial \theta}\right|_{op.} \qquad k_{tf} = \left.\frac{\partial T}{\partial \alpha}\right|_{op.}$$

$$k_{tb} = \left.\frac{\partial T}{\partial \beta}\right|_{op.} \qquad A = -\left.\frac{\partial T}{\partial \dot{h}}\right|_{op.}$$

$$k_\ell = \left.\frac{\partial F}{\partial \theta}\right|_{op.} \qquad k_{\ell f} = \left.\frac{\partial F}{\partial \alpha}\right|_{op.}$$

$$k_{\ell b} = \left.\frac{\partial F}{\partial \beta}\right|_{op.} \qquad B = \left.\frac{\partial F}{\partial \dot{\theta}}\right|_{op.}$$

The equations for these partial derivatives of eq(2.9) and eq(2.10) are presented in the form of the computer program, TEMPLATE, Appendix 3.1.

The minimum, maximum and nominal values of the coefficients used in designing the controllers are given in Table 2.1. (The numerical value of the lift and torque coefficients depend on the wing's (effective) angle of attack and the square of the (effective) wind speed. The structured relationship existing between the coefficients is included in the design by constructing templates based on the independent variables.)

2.2.6 External Disturbances

Among the sources of torque and lift (external) disturbances are,

a) The mass of the airframe is not balanced by the counter–weights and there is therefore a steady–state offset in lift of about –8N (Table 2.2). There is a (smaller) torque offset.

b) wind direction is not parallel to the ground;

c) air flow over wings is not laminar;

d) the wind direction and speed is affected by the movement of the airframe. (The resulting disturbances are state dependent, invalidating linear plant models.)

The approximate measurement of the disturbances is treated in Section 2.3.2.

2.3 Parameter and disturbance identification

2.3.1 Identification of model parameters

The model parameters summarized in Table 2.1 were obtained as follows.

Mass

The mass of the airframe and counter-weights was obtained by measurement and by calculation based on the volume of each mechanical component. The centre of mass of the airframe is located at $(\sum_a m_i r_i)/(\sum_a m_i) = 0.01\text{m}$ (in front of pivot), using data from Table 2.2 below where the sum is over the components of the airframe alone.

Inertia

The inertia of the airframe about the pivot was calculated using the standard formula,

$$J = \sum_i (J_i + m_i r_i^2) \tag{2.22}$$

(Inertia about the pivot equals the sum of the inertia of each component (about the axis parallel to the pivot and through the centre of mass of that component) plus the mass of each component multiplied by the square of the distance between the centre of mass of that component and the pivot) (Fowles, 1977, p.187, D'Souza and Garg, 1984, p.88) and standard tables (Fowles, 1977, p.320) by Mr S.K. Moodley and checked by the author. Table 2.2 below summarizes the components of the airframe.

	inertia	mass	distance	
Back wings (J_β)	1.013×10^{-3}	0.4124	−0.240	air–frame
Back wing servo & bracket	1.297×10^{-4}	0.1234	−0.285	
Front Wings (J_α)	1.141×10^{-4}	0.1587	+0.430	
Front wing servo & bracket	4.287×10^{-5}	0.1121	−0.400	
Body	5.081×10^{-2}	0.4210	+0.080	
Centre bracket	1.671×10^{-5}	0.0522	−0.020	
Pivot	n.a.	0.290	0	}pivot
Counter weights	n.a.	0.698	0	}c.w.

Table 2.2 - Airframe components for calculating inertia (inertia in kg m^2, mass in kg, distance towards front positive)

Friction coefficients

The friction coefficients at zero wind speed, μ_{t0} and $\mu_{\ell 0}$, were obtained by means of model identification:

When the flying machine is dropped with the fans off, it reaches terminal velocity, $\dot{h}_\infty$, and the (assumed linear) friction coefficient for lift is determined by,

$$\mu_{\ell 0} = (m_a + m_p - m_c)\, g \,/\, \dot{h}_\infty \tag{2.23}$$

To measure the friction coefficient for turning, the airframe was allowed to swing freely as a pendulum, with the height fixed. For small angles, the equation of motion is,

$$J\,\ddot{\theta} + \mu_{t0}\,\dot{\theta} + k_1\,\theta = k_2 \tag{2.24}$$

The coefficients k_1 and k_2 depend on the exact position of the pivot relative to the centre of mass and were identified rather than analysed (J was pre–specified). The identification technique used was based on "macro–difference" expressions (Eitelberg, 1988), using software written by Prof Eitelberg. The algorithms developed by the author (Boje, 1986 and 1988) were equally applicable but were not available under the MS–DOS operating system.

Results were, $\mu_{t0} = 0.11$, with a "standard deviation" (as defined in the software) of 0.01 (k_1=0.15±0.02, k_2=0.43±0.04). The coefficient of friction due to relative movement of the airframe and the airstream is an order of magnitude larger than μ_{t0}

Lift and drag coefficients

The lift and drag coefficients for the wings were evaluated by means of wind-tunnel tests on one front wing. Lift and drag measurements were normalised (to obtain the section lift and drag coefficients) by dividing by the free steam dynamic pressure, $q_\infty = \frac{1}{2}\rho_\infty v_\infty^2$ and by the wing area. Results are presented on Figures 2.9 and 2.10 respectively. If φ is the wing angle relative to the free stream wind direction,

$$\text{lift} = c_\ell(\varphi) \times \frac{1}{2}\rho_\infty \times (\text{Wind Speed})^2 \times (\text{Wing Area}) \tag{2.25}$$

$$\text{drag} = c_d(\varphi) \times \frac{1}{2}\rho_\infty \times (\text{Wind Speed})^2 \times (\text{Wing Area}) \tag{2.26}$$

Figure 2.9 and 2.10 show constrained (to obtain necessary symmetry) second order "least–squares" fits for the data which allow easy calculation of the coefficients and their derivatives. These curves are given (for φ in radians) by,

$$\hat{c}_\ell(\varphi) = [-5.305(\varphi+0.011)^2+5.006(\varphi+0.011)]\mathrm{sgn}(\varphi+0.011) \tag{2.27}$$

$$\hat{c}_d(\varphi) = 3.363\ \varphi^2 + 0.0677 \tag{2.28}$$

Figure 2.9 also shows the theoretical section lift coefficient, $c_\ell = 2\ \pi\ (\varphi + 0.011)$, from eq(2.8). Note that the wing produces zero lift at $\varphi = -0.6°$

wind speed	wing angle 0°	5°	10°	15°	20°
10	0.30	0.77	1.38	1.65	2.07
15	0.45	1.65	2.90	3.70	4.10
18	0.52	2.30	4.10	5.20	5.70
20	0.61	2.90	5.10	6.35	–
22	0.65	3.55	6.50	–	–
23	–	–	6.80	–	–
24	0.75	4.23	–	–	–
25	0.78	4.58	–	–	–

Table 2.3a Raw lift data (lift in Newtons, wind speed in m/s)

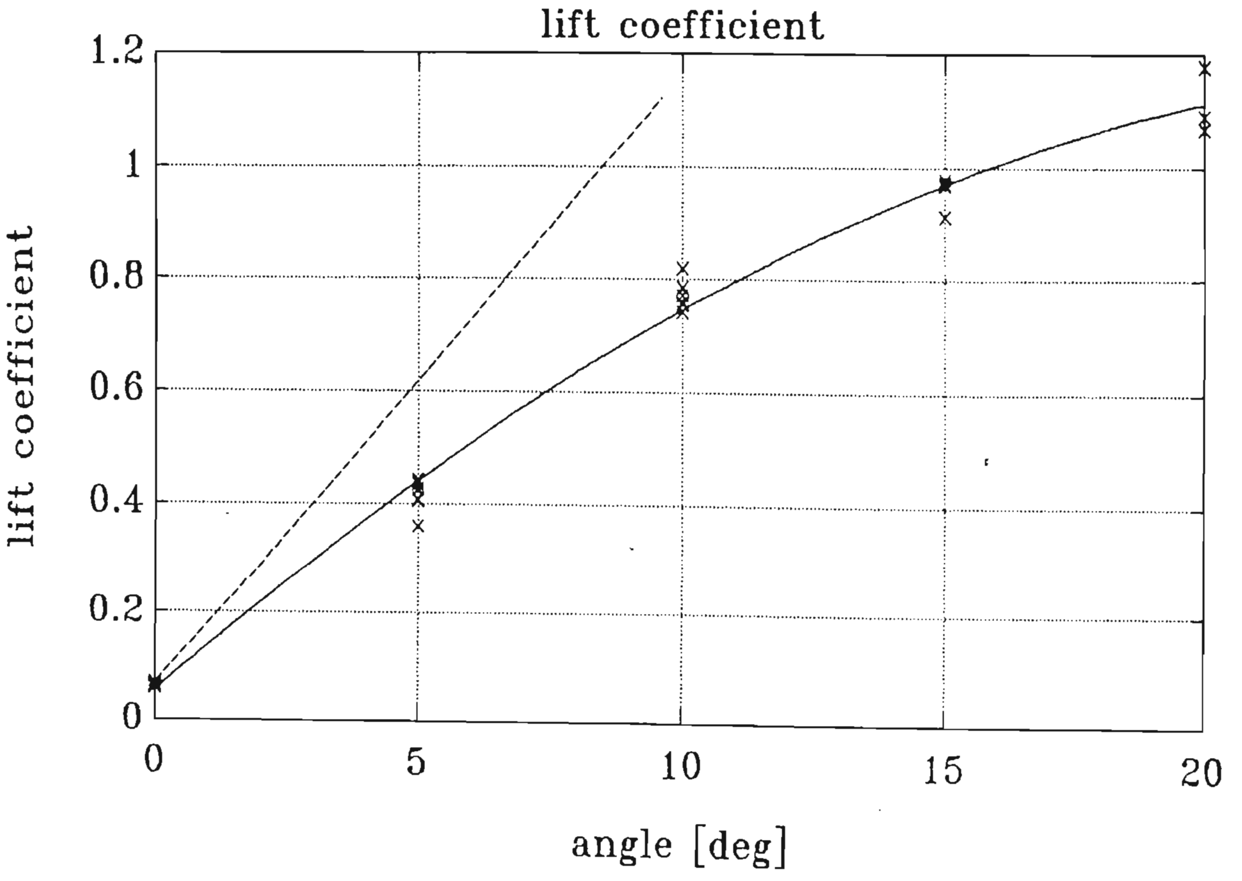


Figure 2.9 – Section lift coefficient, c_ℓ. × indicates measured data, solid line is least squares curve, eq(2.27) and dashed line is theoretical value, eq(2.8)

wind speed	wing angle 0°	5°	10°	15°	20°
10	0.15	0.20	0.40	0.55	0.85
15	0.30	0.43	0.80	1.20	1.75
18	0.38	0.60	1.08	1.70	2.40
20	0.45	0.70	1.35	2.10	–
22	0.55	0.85	1.65	–	–
23	–	–	1.75	–	–
24	0.60	0.95	–	–	–
25	0.68	1.05	–	–	–

Table 2.3b Raw drag data (drag in Newtons, wind speed in m/s)

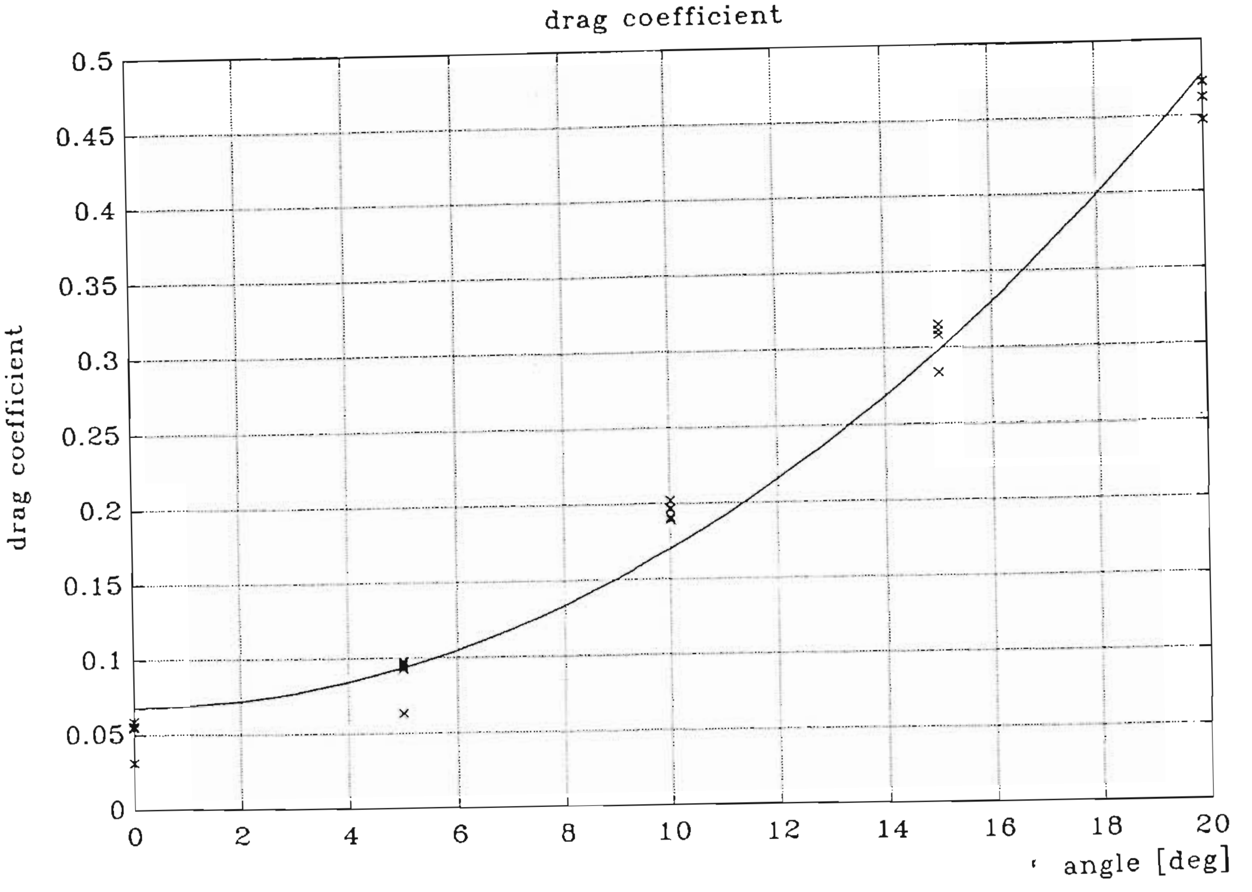


Figure 2.10 – Section drag coefficient, c_d. × indicates measured data, solid line is least squares curve, eq(2.28)

2.3.2 Disturbance Measurement

The equipment used for wind tunnel tests consists of a fulcrum and beam balance. The whole apparatus is damped in a viscous oil as it designed to give steady-state values. The same holds for the wind speed measurement device, which has a small orifice to low-pass filter fluctuations in wind speed. The real experiment has large ventilation fans operating in a confined space and complicated flow patterns are developed. To get a feel for the dynamic behaviour of the lift and drag, a "live" experiment was designed.

To measure the lift and drag, the airframe was fixed rigidly in airstream of one fan and the strain in the attachments was measured. It was hoped to measure the lift and drag in such a manner as to determine the effect one wing had on the other. This experiment proved to be too complicated for the equipment and time allocated to it and expectations had to be lowered to measuring the lift and drag produced by a single wing.

To measure the forces, strain gauges were mounted on steel from a tape measure, shown diagrammatically in Figure 2.11. These strain gauge bridges were connected to high gain, dc coupled differential amplifiers (built with LM308 operational amplifiers) followed by an amplifier to set offset and span (gain). The bridges worked acceptably during calibration, providing linear, repeatable output with acceptably low noise. The availability of cheap, high precision operational amplifiers with large gain–bandwidth products makes this direct measurement approach reasonable. In application it was found that the gauges were very much more sensitive to bending than pure tension and that it was not possible to operate them in pure tension in the airstream. As a result, new tension only gauges were constructed out of nichrome wire, two strands being used as active gauges and two as dummy gauges to reduce the effect of temperature dependence in the nichrome wire. These gauges provided acceptable performance at low wind speeds (up to about 15m/s) but the lift produced at high wind speed was larger than expected, causing the nichrome wires to yield, giving meaningless results.

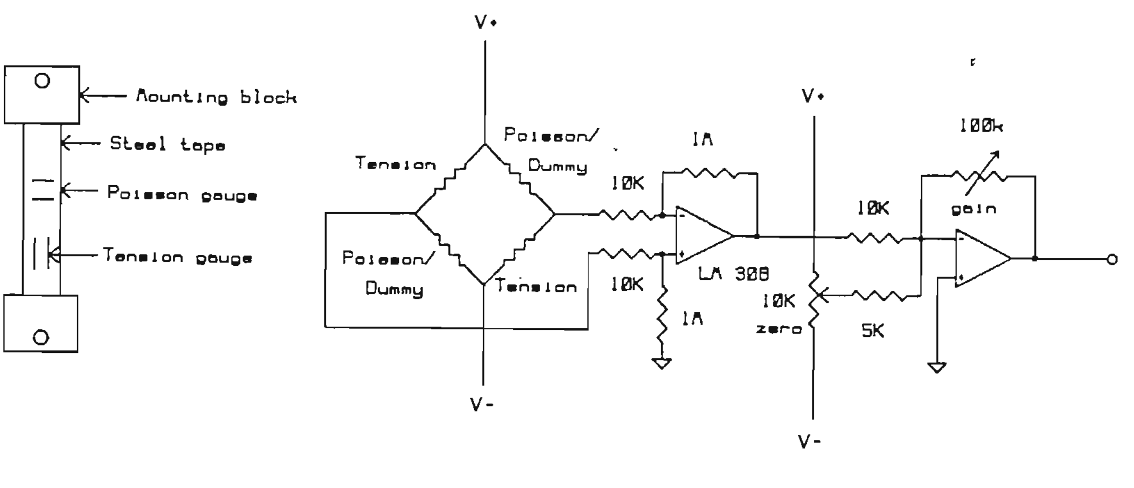


Figure 2.11 – Strain measuring devices

The results of two tests are presented in Figure 2.12 (Test R0 – 15m/s wind speed, wing angle ≈ 2°) and Figure 2.13 (Test R22 – 15m/s wind speed, wing angle ≈ 20°). At the end of each run, the fan was stopped and a 5N calibration weight was applied to the drag and lift axes – the test data has been scaled to show results in Newtons. The second test result, R22, has been used as the disturbance input in most simulation runs and design calculations which follow (it could be argued that the low wind speed is very roughly compensated by the high wind angle). For front wing data, the test data was multiplied the ratio of the front to back wing areas and reversed in time.

Further analysis of the data in Figure 2.13 (R22) is presented. Figure 2.14 and 2.15 show the magnitude spectra of the lift and torque components of the data respectively, calculated using θ=0° and nominal gains from Table 2.1. In Section 3.3 it is shown that some useful design considerations can be obtained if the disturbance can be modelled as broad band noise and Figures 2.14 and 2.15 present possible broad band disturbance models with the following parameters,

$$|E_\ell(\omega)| = \begin{cases} \sqrt{\eta_\ell} & \text{for } \omega = [0, \omega_{x\ell}] \\ 0 & \text{for } \omega > \omega_{x\ell} \end{cases} \tag{2.29}$$

$\sigma_\ell^2 = 132.0\ \mathrm{N}^2$	variance of lift disturbance (N^2)
$\eta_\ell = 1.9\ \mathrm{N}^2/(\mathrm{rad/s})$	lift disturbance model power per rad/s
$\omega_{x\ell} = 70.0$ rad/s	lift disturbance model bandwidth

$$|E_t(\omega)| = \begin{cases} \sqrt{\eta_t} & \text{for } \omega = [0, \omega_{xt}] \\ 0 & \text{for } \omega > \omega_{xt} \end{cases} \tag{2.30}$$

$\sigma_t^2 = 9.5\ \mathrm{N}^2\mathrm{m}^2$	variance of torque disturbance ($\mathrm{N}^2\mathrm{m}^2$)
$\eta_t = 0.14\ \mathrm{N}^2\mathrm{m}^2/(\mathrm{rad/s})$	torque disturbance model power per rad/s
$\omega_{xt} = 70.0$ rad/s	torque disturbance model bandwidth

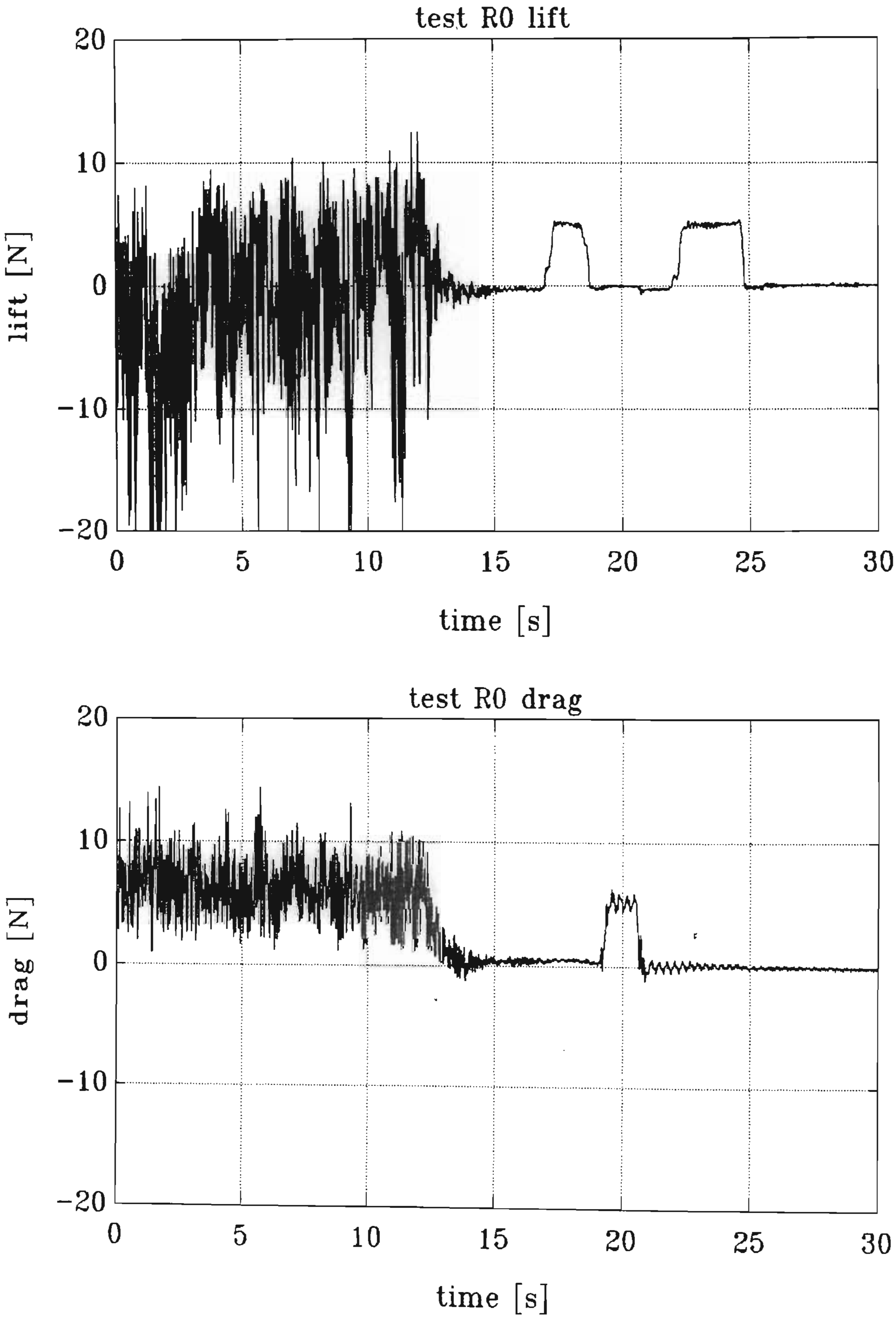


Figure 2.12 – Test R0 – Lift and drag measurement, 15m/s wind speed, wing angle ≈ 2°

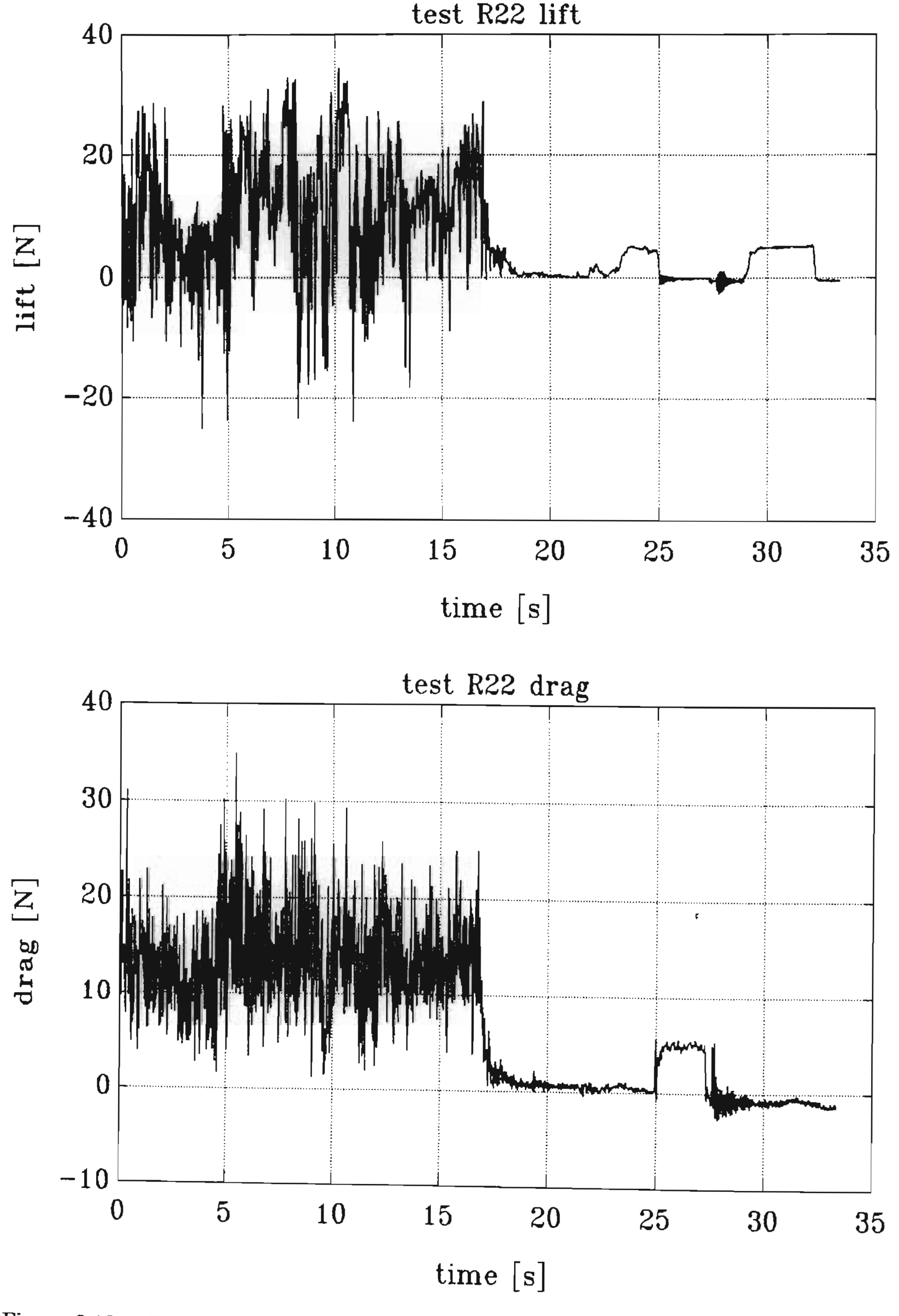


Figure 2.13 – Test R22 – Lift and drag measurement, 15m/s wind speed, wing angle ≈ 20°

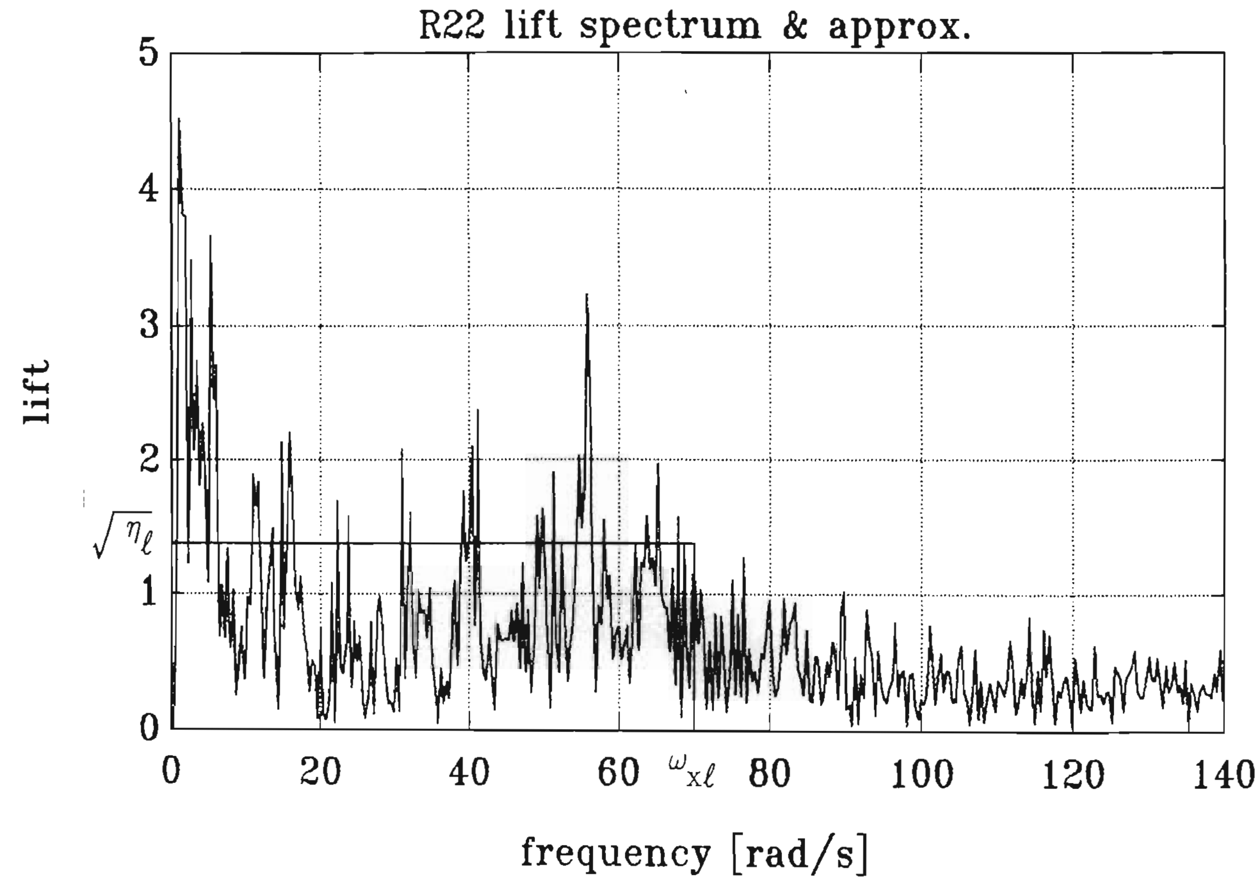


Figure 2.14 – Magnitude spectrum for total lift disturbance from test R22 and equivalent broad band model

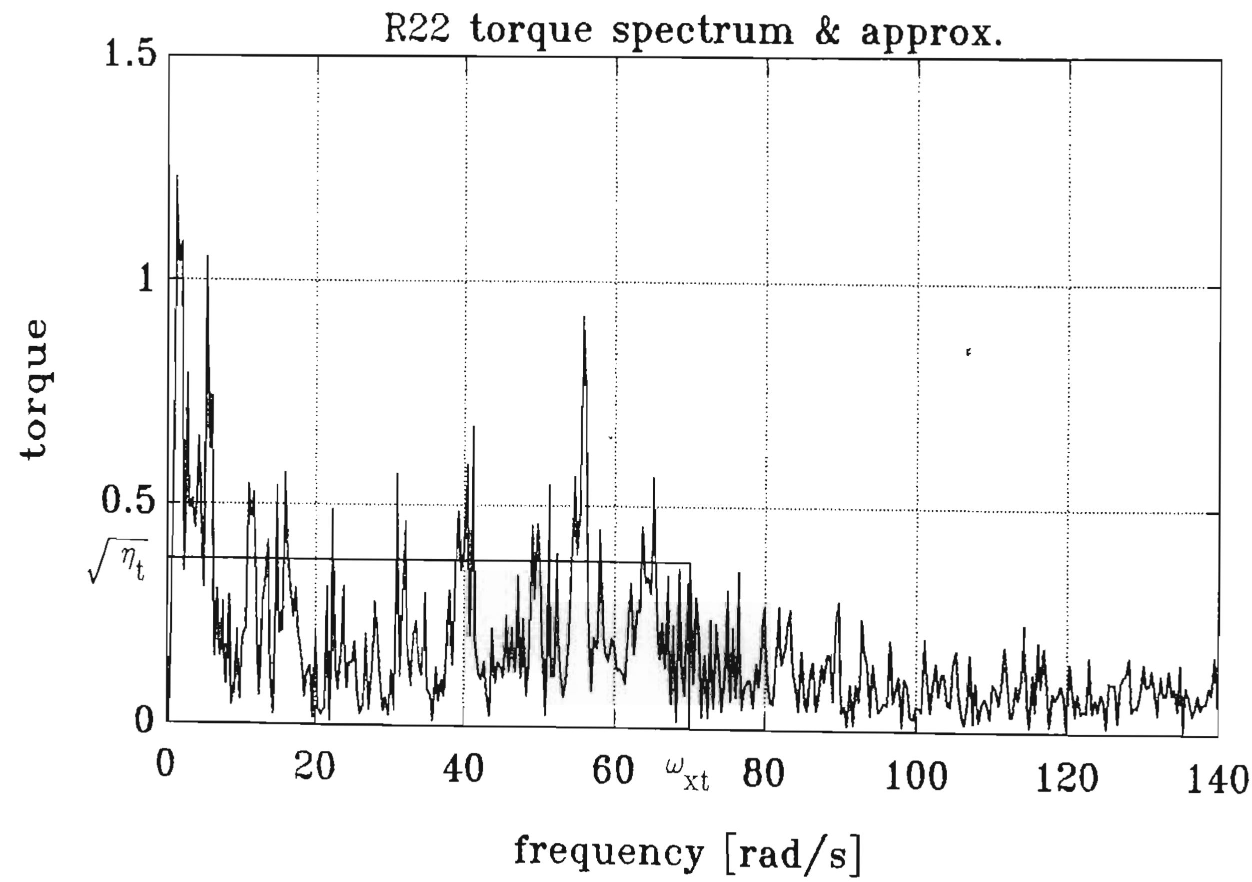


Figure 2.15 – Magnitude spectrum for total torque disturbance from test R22 and equivalent broad band model

2.3.3 Other Design Constraints

The following constraints are used in the subsequent Chapters

$$-50^\circ \le \alpha \le 50.0^\circ, \qquad |\dot{\alpha}| \le 200^\circ/\mathrm{s} \tag{2.31a}$$

$$-35^\circ \le \beta \le 40^\circ, \qquad |\dot{\beta}| \le 200^\circ/\mathrm{s} \tag{2.31b}$$

These values were obtained by direct measurement. As a result the following rough allowable input standard deviations were defined,

$$\sigma_\alpha = \sigma_\beta = 0.5 \text{ rad} \tag{2.32a}$$

$$\sigma_{\dot{\alpha}} = \sigma_{\dot{\beta}} = 2.0 \text{ rad/s} \tag{2.32b}$$

The wing actuator dynamics were modelled by,

$$\frac{\alpha(s)}{\hat{\alpha}(s)} = \frac{1}{(s/\omega_{n\alpha})^2 + 2\zeta_\alpha(s/\omega_{n\alpha}) + 1}, \quad \omega_{n\alpha}=170 \text{ rad/s}, \ \zeta_\alpha=0.4 \tag{2.33}$$

$$\frac{\beta(s)}{\hat{\beta}(s)} = \frac{1}{(s/\omega_{n\beta})^2 + 2\zeta_\beta(s/\omega_{n\beta}) + 1}, \quad \omega_{n\beta}=170 \text{ rad/s}, \ \zeta_\beta=0.4 \tag{2.34}$$

This model for wing actuator dynamics stems from the fact that each actuator is essentially an integrator (from applied voltage to the actuator motor to position), with some high frequency dynamics (due principally to the load and motor inertias and the inductance of the motor armature). The wing actuator controllers were closed with proportional controllers, with the gain set at the value which resulted in the fastest response without too poorly damped oscillations. Thereafter the wing actuators were considered as part of the plant. In some cases design was executed using $\omega_{n\cdot} = 100$ rad/s to account for the effect of actuator rate saturation but in each design, the wing actuator dynamics were regarded as fixed (i.e. with no uncertainty).

2.4 Simulating the plant and controllers in closed loop using modular simulation and Modula 2

Simulation is a fast, inexpensive method of:
a) Testing the first principles model of the plant.
b) Obtaining understanding of the behaviour of the plant
c) Gaining confidence that a controller design will provide satisfactory results when applied to the non–linear plant. Although the quantitative feedback design can be based on a "linear equivalent" plant set for the non–linear plant, this is only possible if every required plant output vector and the unique input vector corresponding to that output vector is known and is Laplace transformable. In this practical problem, the quantitative feedback design is based on the linearised plant set resulting from linearisations about all reasonable operating points. As a result, the plant uncertainty is state–dependent and the design does not guarantee performance.

This section will outline the simulation of the flying machine and its feedback controllers via an existing (modular) simulation method (Eitelberg, 1982), implemented using the recently developed Modula 2 programming language (Wirth, 1983; King, 1988). The modular features of Modula 2 allowed the structure of simulation software to mimic the real plant–controller structure – As the real plant and real controllers are separate it makes good sense to keep the simulation of the plant and controllers separate. In this way, once the plant (which is non–linear and uncertain but fixed) has been correctly simulated, no alteration of the simulation code for the plant is required when the controller structure is altered.

In explicit simulation of a system the step size must be of the same order as the shortest time constant in the system. In control problems, there are often far–off poles in the open–loop system which are (for example) necessary to make the controller realizable and for it to have desirable high frequency properties. These poles are of little interest when simulating the response of the closed loop system to disturbances or inputs but must not make the simulation of a stable system unstable. It is well known (Eitelberg, 1983) that implicit methods can be applied to stiff simulation problems where explicit methods fail.

The simplest implicit method is the implicit Euler:
Given the continuous time system,

$$\dot{\underline{x}} = \underline{f}(\underline{x},t); \qquad \underline{x}(t_0) = \underline{x}_0 \tag{2.35}$$

The implicit Euler solution is given by:

$$\underline{x}_{k+1} = \underline{x}_k + \Delta t\ \underline{f}(\underline{x}_{k+1}, t_{k+1})$$

$$= \underline{x}_k + \Delta t\ (\underline{f}(\underline{x}_k, t_k) + \left.\frac{\partial \underline{f}}{\partial \underline{x}}\right|_{\underline{x}_k, t_k} (\underline{x}_{k+1} - \underline{x}_k))$$

$$= \underline{x}_k + \Delta t\ (\underline{I} - \Delta t\ \underline{J}_k)^{-1}\ \underline{f}_k \tag{2.36}$$

($\underline{J}_k$ is the model Jacobian evaluated at time t_k.)
For numerical reasons this is actually programmed as:

a) Solve : $(\underline{I}/\Delta t - \underline{J}_k)\ \Delta \underline{x} = \underline{f}_k$

b) Add $\underline{x}_{k+i} = \underline{x}_k + \Delta \underline{x}$ (2.37)

For the modular simulation using implicit Euler the following are defined (directly from Eitelberg, 1982):

"module state differential equation"

$$\dot{\underline{x}}_i = \underline{f}_i(\underline{x}_i, \underline{u}_i),\ i=1,\ldots,m \tag{2.38}$$

"module output equation"

$$\underline{y}_i = \underline{g}_i(\underline{x}_i, \underline{u}_i),\ i=1,\ldots,m \tag{2.39}$$

"connection and external input equation"

$$\underline{u}_i = \underline{h}_i(\underline{y}_1, \ldots, \underline{y}_m, t),\ i=1,\ldots,m \tag{2.40}$$

"module Jacobian"

$$\underline{J}_i = \frac{\partial \underline{f}_i(\underline{x}_i, \underline{u}_i)}{\partial \underline{x}_i} \tag{2.41}$$

"module input sensitivity"

$$\underline{F}_{u\,i} = \frac{\partial \underline{f}_i(\underline{x}_i, \underline{u}_i)}{\partial \underline{u}_i} \tag{2.42}$$

"coupling sensitivity" (provided there are no algebraic loops)

$$\frac{\partial \underline{u}_i}{\partial \underline{x}_j} = \frac{\partial \underline{h}_i}{\partial \underline{y}_j} \left[\frac{\partial \underline{g}_j}{\partial \underline{x}_j} + \frac{\partial \underline{g}_j}{\partial \underline{u}_j} \frac{\partial \underline{u}_j}{\partial \underline{x}_j} \right] \tag{2.43}$$

The language features available in Modula 2 (and not in older languages such as FORTRAN) which can enhance the development of such modular simulation programs are:

a) Dynamic variable definition – allowing very large problems to be solved on small machines.

b) Separate compilation of Modules – an improvement on Pascal.

c) PROCEDURE TYPES – allowing modules to be altered at run–time

d) Concurrency – allowing modules to be solved in parallel

e) Highly structured code with strong variable types (and modern language support environment) – enhances the development and maintenance of software.

The structure of the Modula 2 simulation program is conceptually:

```
(*-----------------------------------------------------------------------*)

DEFINITION  MODULE  MatrixUtilities;
TYPE  Vector;
TYPE  Matrix;
     (* All  matrix  operations  are  defined  here  *)
END  MatrixUtilities.
(*-----------------------------------------------------------------------*)

DEFINITION  MODULE  LocalModule1;
(* Same  structure  for  each  i = 1..m  *)
FROM  MatrixUtilities   IMPORT  Vector,  Matrix;

PROCEDURE  SolveAugmentedSystem    (
                  Input                : Vector;
              VAR AugmentedState       : Matrix);

(*  Solve  : Φi [δxi,Xi] = [fi, Fu,i,k] for AugmentedState, [δxi,Xi] *)

(* Note  that  the  IMPLEMENTATION  MODULE  must  make  its  own  local
   copy  of  the  augmented  state  vector  for  use  in
   UpdateStateVector.   This  means  that  the  AugmentedState   above
   can  be  modified   /  discarded   by  the  global  program  *)

PROCEDURE  UpdateStateVector(
                  InputIncrement       : Vector;
              VAR Output               : Vector);
(*  Evaluate  : Δxi = δxi + Xi δui
                xi(k+1) = xi(k)+ Δxi  *)

END  LocalModule1.
(*-----------------------------------------------------------------------*)
```

```
MODULE  Global;

FROM  MatrixUtilities   IMPORT  Vector,  Matrix
                                      (*  and  matrix  routines  *);

IMPORT  LocalModule1   (*,......,  LocalModule  [NumberModules]   *);

CONST
    NumberModules   = 2;  (*  In  this  problem  *)

(*--------------------------------------------------------------*)

PROCEDURE  SolveGlobalEquation
       (AugState                    : ARRAY  OF Matrix;
        VAR  ModuleInputIncrements    : ARRAY  OF Vector);
(*  In  this  PROCEDURE,  construct  coupling  sensitivities   and  solve
   for  module  input  increment  vectors.
```

$$\delta\underline{u}_i - \sum_{j=1}^{m} \frac{\partial\underline{u}_{i,k}}{\partial\underline{x}_{j,k}} \underline{X}_j\, \delta\underline{u}_j = \sum_{j=1}^{m} \frac{\partial\underline{u}_{i,k}}{\partial\underline{x}_{j,k}}\, \delta\underline{x}_j \qquad i=1..m \quad *)$$

```
END  SolveGlobalEquation;

(*--------------------------------------------------------------*)

VAR  SolveAugSystem    : ARRAY[1..NumberModules]     OF
                           PROCEDURE   (    (*  Input   *)  Vector,
                                        VAR (*  Aug.  State   *)  Matrix);

VAR  UpdateState    : ARRAY[1..NumberModules]     OF
                        PROCEDURE   (     (*  Input  increment   *)  Vector,
                                     VAR  (*  Output   *)  Vector);

VAR
   Input        : ARRAY[1..NumberModules]     OF  Vector;
   Output       : ARRAY[1..NumberModules]     OF  Vector;
   InputIncr    : ARRAY[1..NumberModules]     OF  Vector;
   AugState     : ARRAY[1..NumberModules]     OF  Matrix;

VAR
   Module,    Step              : CARDINAL;
   StepSize,   Time,   StopTime   : REAL;

(*--------------------------------------------------------------*)
BEGIN
(*  Initialise   local   module   procedures,   variables   etc  *)

    SolveAugSystem[1]     := LocalModule1.SolveAugmentedSystem;
    UpdateState[1]     := LocalModule1.UpdateStateVector;
    (*         .
               .  etc   *)
```

```
(* Do the required  simulation  - adding  step  size  control  etc.  if
   required  *)
    Time  := 0.0;
    Step  := 0;
    REPEAT
        Step  := Step  + 1;
        Time  := Time  + StepSize;      (*Denoted  k=1,2,..*)

        FOR Module  := 1 TO NumberModules  DO
                                          (* IN PARALLEL  IF POSSIBLE  *)
                                       (* OR AS CONCURRENT  PROCESSES  *)
            SolveAugSystem   [Module]
                                 (Input[Module],   AugState[Module]    );
        END;

        SolveGlobalEquation    (AugState,  InputIncr);

        FOR Module  := 1 TO NumberModules  DO
                                          (* IN PARALLEL  IF POSSIBLE  *)
                                       (* OR AS CONCURRENT  PROCESSES  *)

           UpdateState   [Module]
                               (InputIncr[Module],    Output[Module]    );
        END;
    UNTIL  Time  >= StopTime;

END Global.
(*----------------------------------------------------------------------*)
```

Figure 2.19 Modula–2 modular program structure

The actual programs are included in Appendix 3.

CHAPTER 3 – QUANTITATIVE FEEDBACK DESIGN – THEORETICAL CONSIDERATIONS

3.1 Introduction to the principal ideas of Quantitative feedback design for SISO systems

A two–degree–of–freedom control system is shown in Figure 3.1. The plant is uncertain and given by the set, $\mathscr{P} = \{P(s)\}$. There are unknown external disturbances, $\mathscr{E} = \{E(s)\}$ affecting the plant output via $\mathscr{P}^* = \{P^*(s)\}$. Feedback signals are contaminated by sensor noise, $\mathscr{N} = \{N(s)\}$. Feedback (via the controller, G(s)) is only required in order to reduce the effect of plant uncertainty (and non–linearity) on the plant response, to reduce the effect of disturbance, E, or to stabilize open–loop unstable systems. The command response of the plant can be modified by means of the pre–filter, F(s).

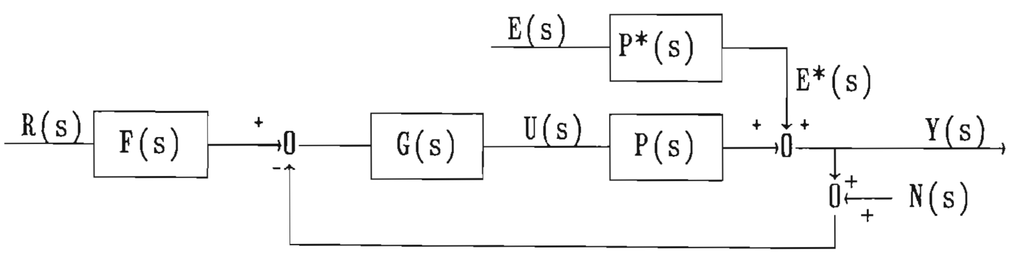


Figure 3.1 Two degree of freedom feedback structure

Quantitative feedback design for SISO and MIMO systems, given quantitative specifications on the plant output, Y(s), is well established (see Horowitz, 1982, for a list of references). The paper by Horowitz and Sidi (1972) probably gives the clearest introduction to the design technique. SISO quantitative feedback design proceeds as follows:

a) Specifications to achieve desired command following

$$T_{Y/R}(s) = Y(s)/R(s) = \frac{F(s)G(s)P(s)}{1 + G(s)P(s)} \tag{3.1}$$

and modified disturbance, E*(s), rejection

$$T_{Y/E^*}(s) = Y(s)/E^*(s) = \frac{1}{1+G(s)P(s)} \tag{3.2}$$

of the form,

$$A(\omega) \leq |T_{Y/R}(j\omega)| \leq B(\omega) \tag{3.3}$$

and

$$|T_{Y/E^*}| \leq C(\omega) \tag{3.4}$$

are assumed given. $L(s) = P(s)\ G(s)$ is the open loop transfer function. Horowitz (1963, Chapter 9) and many others have treated the problem of obtaining such specifications from step response specifications, knowledge of the stochastic properties of the disturbance (see Section 3.3) etc. Magnitude specifications eq(3.3) and eq(3.4) alone are sufficient for minimum phase systems to achieve desired time domain behaviour but phase specifications must be included for non–minimum phase systems.

b) Select a number of discrete frequencies, ω_i, at which design boundaries will be checked and an arbitrary nominal plant P_0. Note that the choice of the nominal plant does not affect the outcome of the design. The free choice of nominal plant is a very attractive feature of the design as it is a "handle" (by which the plant set is held) rather than representing a most likely (nominal) operating point. The freedom of choice of the nominal even allows one to chose a nominal outside the plant set if the extra conceptual difficulty can be justified by other considerations. (The idea of a "handle" originated during the writing of the CAD package, "DESIGN" by Prof E Eitelberg and Mr G Sutcliffe.)

c) Construct plant templates at these frequencies ($\mathcal{T}_i = \{|P(j\omega_i)|_{dB}, Arg(p(j\omega_i))\}$). At least since Horowitz and Sidi's 1972 paper, quantitative feedback design has been most easily executed on the Nichols and inverse Nichols charts where, $|L|_{dB} = |P|_{dB} + |G|_{dB}$ and $Arg(L) = Arg(P) + Arg(G)$ (i.e. G can shift but neither rotate nor change the shape of the templates).

d) From the command response bounds, eq(3.3), construct corresponding bounds (at each ω_i) on the nominal open loop transfer function, $L_0 = P_0\ G$, for the nominal plant, P_0, by manipulating $\mathcal{T}_i$ on the Nichols chart to ensure that the closed loop uncertainty is within acceptable bounds, $\Delta|L/(1+L)| \leq B(\omega)-A(\omega)$. (The final specification, eq(3.3) is achieved using pre–filter, F.) Call these bounds,

$$\mathcal{B}_{Y/R\,i} = \{L_0 : \Delta|L/(1+L)| \leq B(\omega_i)-A(\omega_i)\} \tag{3.5}$$

e) From the disturbance rejection bounds, eq(3.4), construct corresponding bounds (at each ω_i) on the open loop transfer function, L_0, for the nominal plant, P_0, manipulating the plant template on the inverse Nichols chart. Call these bounds,

$$\mathcal{B}_{Y/E^*\,i} = \{L_0 : |Y(j\omega_i)/E^*(j\omega_i)| \leq C(\omega_i)\} \tag{3.6}$$

f) Find a controller, G, so that $L_0 = G\ P_0$ satisfies both its bounds, $\mathcal{B}_{Y/R\,i}$ and

$B_{Y/E^* i}$ at each frequency, ω_i. In the QFT literature usually L_0 is shaped (found) and then $G=L_0/P_0$ is obtained. This can lead to inconvenient controller realizations. The package, "DESIGN" allows L_0 to be shaped by defining P_0 and selecting and adjusting G.

3.2 Some additions to SISO QFT – Disturbance entering through part of the plant and input specifications

The quantitative feedback theory (QFT) presented in the literature has concentrated on achieving specifications on the plant output. QFT has been criticized (by Doyle, 1986) for not "explicitly including" input specifications during the design. Horowitz's reply (1987) points out that the QFT does provide a "price list" of the cost of feedback in terms of bandwidth. Quantitative specifications on the plant input can be included in QFT but there are some implications of having to meet both input and output specifications (especially in multivariable control problems). Plant uncertainty reduction and disturbance rejection are typically achieved at the expense of high loop gain ($|L(s)|=|G(s)P(s)|\gg 1$), over a larger frequency range than the frequency range over which feedback benefits are required. The amplification (to the plant input) of disturbances, command inputs and sensor noise resulting from high loop gain may not be acceptable in physical plants due to saturation or wear at the plant input. As the performance of any control system design is limited by the power which can be delivered by the plant which is to be controlled, input specifications should be included in the design of practical controllers. (Plant input saturation is not always catastrophic if care is taken during the design.)

The plant input is often ignored in practical plants – the ubiquitous PID controller (or the one mentioned in the Introduction) is expected to provide large gain at high frequency which will invariably saturate the plant input. Plant maintenance engineers find that actuators must constantly be replaced unless they "detune" the derivative gain.

For a fixed plant, fixing the disturbance to output behaviour obviously also fixes the disturbance to input behaviour, but in uncertain design, if the specifications allow some design freedom, this can be used to select a controller to satisfy both input and output specifications. This section deals only with disturbance – to – plant input specifications from which other input specifications follow easily.

SISO plants have disturbance specifications of the form,
Output,

$$|Y/E| = \left|\frac{P^*(j\omega)}{1+L(j\omega)}\right| = |P^*|\left|\frac{1}{1+L}\right| = \left|\frac{1}{G}\right|\left|\frac{P^*}{P}\right|\left|\frac{L}{1+L}\right| \leq a(\omega) \tag{3.7}$$

Input,

$$|U/E| = \left|\frac{G(j\omega)P^*(j\omega)}{1+L(j\omega)}\right| = |G|\,|P^*|\left|\frac{1}{1+L}\right| = \left|\frac{P^*}{P}\right|\left|\frac{L}{1+L}\right| \le b(\omega) \tag{3.8}$$

Unlike the loci of constant $|L/(1+L)|$ (Nichols chart) and loci of constant $|1/(1+L)|$ (Inverse Nichols chart), loci of constant $|U/E|$ and $|Y/E|$ cannot be plotted on the rectangular grid of log–magnitude and angle of the open loop transfer function, L(s). The two forms used for each of the eq(3.7) and eq(3.8) above highlight the fact that standard QFT design using templates can only be carried out on the Nichols $\left[\frac{L}{1+L}\right]$ or inverse Nichols $\left[\frac{1}{1+L}\right]$ charts. To design quantitatively requires using the worst case magnitudes and the second form may then be more economical than the first because of possible cancellation between P and P*. For example, using eq(3.8) for input specifications, the constraints are, $|G|\left|\frac{1}{1+L}\right| \le b/\left[\sup_{P^*\epsilon\mathcal{P}^*}|P^*|\right]$ and, $\left|\frac{L}{1+L}\right| \le b/\left[\sup_{(P,P^*)\epsilon(\mathcal{P},\mathcal{P}^*)}\left|\frac{P^*}{P}\right|\right]$ respectively. Neither are able to utilize possible cancellation between P* and (1+L). In most plants, at least at low frequency $(P,P^*)\epsilon(\mathcal{P},\mathcal{P}^*)$ is an ordered pair since the uncertainty is structured. The design should if possible make maximum use of the structure of the plant. If suitable CAD tools exist to draw bounds on $L_0(j\omega_i)$ using $\left|\frac{P^*}{1+L}\right| \le a(\omega_i)$ or $\left|\frac{GP^*}{1+L}\right| \le b(\omega_i)$ directly, over design is avoided in the drawing of the bounds. One method would be to evaluate the left hand sides and test the inequalities for each frequency ω_i of interest, for each (P,P*) of the plant set (or a representative subset) and over a sufficiently fine grid of $L_0=(|L_0|, \mathrm{Arg}(L_0))$. This is the 'Afghan method', because "its effrontery evokes the Soviet Union's approach to Afghanistan" (Golubev, 1983).

Note that if it is required that $\Delta|L/(1+L)| \approx 0$dB then $|GP| \approx |1+GP|$ and from eq(3.8),

$$|U/E| \approx \left|\frac{P^*}{P}\right| \le \sup_{(P,P^*)\epsilon(\mathcal{P},\mathcal{P}^*)}\left|\frac{P^*}{P}\right| \le b(\omega) \tag{3.9}$$

showing that the plant input is unavoidable (and determined by the plant design alone) at those frequencies where high loop gain is required for other purposes (sensitivity reduction, disturbance rejection and stability).

Finding bounds on transfer functions like $|G|\left|\frac{1}{1+L}\right|$ (plant output (measurement) noise to plant input) and $\left|\frac{1}{G}\right|\left|\frac{L}{1+L}\right|$ (plant input disturbance to plant output) is easy using standard Nichols and inverse Nichols charts as there is usually no uncertainty in G and inequalities can be tested along lines of constant $|G| = |L_0|/|P_0|$ since $|P_0(j\omega_i)|$ is fixed for ω_i. For a fixed value of $|G|$ the template is moved horizontally across the Nichols or inverse Nichols Chart (i.e. keeping $|G|$ constant) and the region in which the respective inequality is valid are marked. This process is repeated at as many values of $|G|$ as are required to construct an accurate bound.

When input specifications are included, it should be clear that it may not be possible to solve a given problem – the upper and lower bounds on $L_0(j\omega_i)$ can have no intersection. In this sense, specifications resulting from input limits are like performance specifications on non–minimum phase plants – they must be compatible with the plant capability.

If the power at the input is the only limitation, the available input power must be divided between sensor noise and disturbance unless the relative phases of sensor noise and disturbance signals are known.

The design for sensor noise is very similar to that for disturbance and $|U/R|$ specifications constrain the design of the pre–filter, F(s), after the closed loop has been designed. In some feedback problems, the transfer of the external reference to the plant input may be severely limited by the disturbance reduction requirements of the closed loop. In Horowitz and Liao (1986), non–linear compensation is proposed for a saturating, unstable plant in order to block the command signal to plant input when all the input power is required to reject disturbances so that closed loop stability can be maintained.

3.3 Obtaining Plant Input Specifications for Quantitative Feedback Design

Obtaining frequency domain input specifications from the time domain or from knowledge of stochastic behaviour of the plant signals is treated in the literature (Horowitz 1963, Chapter 9). In the case of the flying machine it will be seen that external disturbances enter the plant such that $P^*(s) = \frac{1}{k} P(s)$ and input specifications will be on $\frac{1}{k} |L/(1+L)|$, with wide band noise modelling the disturbances.

Consider a dominantly second order closed–loop system, with the model,

$$T_{U/E}(s) = \frac{G(s)P^*(s)}{1 + L(s)} = \frac{1}{k\ [(s/\omega_n)^2 + 2\zeta s/\omega_n + 1]} \tag{3.10}$$

(The impulse response of disturbance–to–plant input is $t_{u/e}(t) = \mathscr{L}^{-1}\{T_{U/E}(s)\}$.)

If the external disturbance is stationary, zero mean, band limited white noise and ergodicity is assumed,

$$\mathscr{E}\{e(t)\} = 0 \tag{3.11}$$

$$\begin{aligned} |E(j\omega)|^2 &= \eta \quad \omega = [0, \omega_x],\ \omega_x \gg \omega_n \\ &= 0 \quad \omega > \omega_x \end{aligned} \tag{3.12}$$

(One sided spectra will be used.) The noise bandwidth is much larger than the cut–off frequency of $T_{U/E}$. The external disturbance has a variance of $\sigma_E^2 = \eta\ \omega_x$.

The resulting plant input is,

$$u(t) = \int_0^t t_{u/e}(\tau)\ e(t-\tau)\ d\tau \tag{3.13}$$

and expected value is,

$$\begin{aligned} \mathscr{E}\{u(t)\} &= \mathscr{E}\left\{ \int_0^t t_{u/e}(\tau)\ e(t-\tau)\ d\tau \right\} \\ &= \int_0^t t_{u/e}(\tau)\ \mathscr{E}\{e(t-\tau)\}\ d\tau \\ &= 0 \end{aligned} \tag{3.14}$$

(A pre–condition for exchanging the order of integration and expectation is that the integral must remain bounded (Cooper and McGillem, 1979, p.179) which excludes the possibility of a non proper system (i.e. a system with an excess of zeros over poles).)

The power spectrum of the plant input is given by,

$$|U_E(j\omega)|^2 = \frac{1}{k^2\ [(1-(\omega/\omega_n)^2)^2\ +\ (2\zeta\omega/\omega_n)^2]}\ \eta \qquad \omega = [0,\omega_x]$$

$$= \quad 0 \qquad\qquad \omega > \omega_x$$

(3.15)

The variance of the plant input resulting from the zero mean disturbance is given by the auto–correlation of the power spectral density at zero time separation,

$$\mathscr{E}\{u(t)^2\} = R_{uu}(0) = \int_0^{\infty} U^2(j\omega)\, d\omega$$

$$= \int_0^{\omega_x} \frac{1}{k^2\ [(1-(\omega/\omega_n)^2)^2\ +\ (2\zeta\omega/\omega_n)^2]}\ \eta\ d\omega$$

$$\approx \frac{\eta}{k^2}\int_0^{\infty} \frac{1}{(1-(\omega/\omega_n)^2)^2\ +\ (2\zeta\omega/\omega_n)^2}\ d\omega \qquad (3.16)$$

(since $\int_{\omega_x}^{\infty} \cdot\ d\omega \approx 0$ if $\omega_x \gg \omega_n$)

Substituting $x=\omega/\omega_n$,

$$\mathscr{E}\{u(t)^2\} = \omega_n \frac{\eta}{k^2}\int_0^{\infty} \frac{1}{x^4\ +\ (4\zeta^2-2)\ x^2\ +\ 1}\ dx$$

$$= \omega_n \frac{\eta}{k^2}\frac{\pi}{4\zeta} \qquad (3.17)$$

If the standard deviation of the input must be small, keep ω_n small, k large and ζ large. Eq(3.17) shows explicitly what Horowitz has always said that large bandwidth (and the resulting amplification of sensor noise to the plant input) is the price paid for feedback benefits.

Consider the variance of the plant input derivative (denoted, subscript dot). The power spectrum of the plant input derivative is given by,

$$|U_{dot\ E}(j\omega)|^2 = \frac{\omega^2}{k^2\ [(1-(\omega/\omega_n)^2)^2\ +\ (2\zeta\omega/\omega_n)^2]}\ \eta \qquad \omega=[0,\omega_x]$$

$$= \quad 0 \qquad\qquad \omega>\omega_x$$

(3.18)

The variance of the plant input derivative resulting from the zero mean disturbance is given by the auto–correlation of the power spectral density at zero time separation,

$$\mathscr{E}\left\{\left[\frac{du(t)}{dt}\right]^2\right\} = R_{\dot{u}\dot{u}}(0) = \int_0^{\infty} (j\omega\, U(j\omega))^2\, d\omega$$

$$= \int_0^{\omega_x} \frac{\omega^2}{k^2\ [(1-(\omega/\omega_n)^2)^2 + (2\zeta\omega/\omega_n)^2]}\eta\ d\omega$$

$$\approx \frac{\eta}{k^2} \int_0^{\infty} \frac{\omega^2}{(1-(\omega/\omega_n)^2)^2 + (2\zeta\omega/\omega_n)^2}\ d\omega \qquad (3.19)$$

Substituting $x=\omega/\omega_n$,

$$\mathscr{E}\{u(t)^2\} = \omega_n^3 \frac{\eta}{k^2} \int_0^{\infty} \frac{x^2}{x^4 + (4\zeta^2-2)\ x^2 + 1}\ dx$$

$$= \omega_n^3 \frac{\eta}{k^2} f(\zeta) \qquad (3.20)$$

The integrals, $\int_0^{\infty} \frac{1}{x^4+(4\zeta^2-2)x^2+1}\,dx = \frac{\pi}{4\zeta}$ and $\int_0^{\infty} \frac{x^2}{x^4+(4\zeta^2-2)x^2+1}\,dx = f(\zeta)$ (evaluated numerically) are shown as a function of ζ in Figure 3.2.

The disturbances acting on the flying machine can be regarded as roughly band–limited white noise (this motivates Section 2.3.2) and the disturbance to plant input transfer functions will be seen to have the form of eq(3.10). Equations (3.17) and (3.20) will therefore be useful in the flying machine controller designs, described in Chapters 4, 5 and 6.

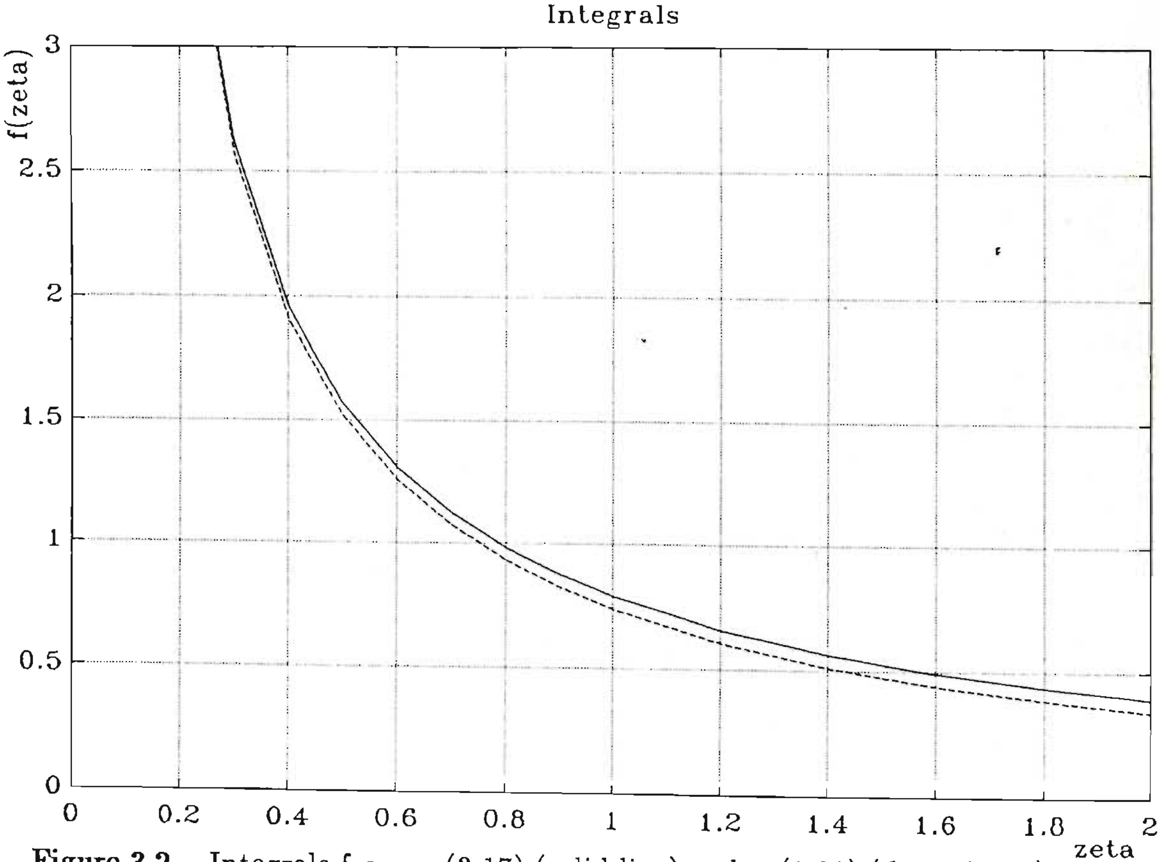


Figure 3.2 – Integrals from eq(3.17) (solid line) and eq(3.20) (dashed line)

3.4 MIMO QFT design with disturbances and plant input specifications

3.4.1 General equations for disturbance to input design

Following on what was shown in Section 3.2 for SISO plants, this section shows how disturbance to output and disturbance to input specifications can be included in MIMO design.

Consider the n–input, n–output multivariable problem shown in Figure 3.3, with disturbance to the plant output, $\underline{T}_{Y/E}(s)$

$$(\underline{I} + \underline{P}\,\underline{G})\,\underline{T}_{Y/E} = \underline{P}^* \tag{3.21}$$

and disturbance to the plant input, $\underline{V}_{U/E}(s)$,

$$(\underline{I} + \underline{G}\,\underline{P})\,\underline{V}_{U/E} = -\,\underline{G}\,\underline{P}^* \tag{3.22}$$

($\underline{I}$ is n×n identity matrix).

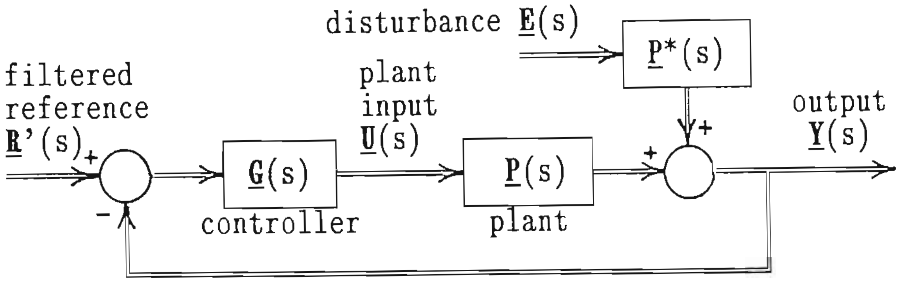


Figure 3.3 – Feedback loop with significant disturbances

The (disturbance) specifications are assumed to have the following form,

$$|\underline{T}_{Y/E}(j\omega)|_{ij} \le [\underline{A}(\omega)]_{ij} = a_{ij}(\omega) \quad \text{(output)} \tag{3.23}$$

$$|\underline{V}_{U/E}(j\omega)|_{ij} \le [\underline{B}(\omega)]_{ij} = b_{ij}(\omega) \quad \text{(input)} \tag{3.24}$$

The absolute values are taken element by element.

There may be other specifications, for example on command response and on sensor noise to plant input, but these can be incorporated in an obvious way. As is the case with output specifications, if one attempts to obtain $\underline{V}_{U/E}$ explicitly, all the entries of $\underline{G}$ and of $\underline{P}$ affect each element of $\underline{V}_{U/E}$. Following the ideas of Horowitz (1982a, 1982b), for a diagonal controller, $\underline{G} = \text{diag}\{g_1..g_n\}$, $\underline{T}_{Y/E}$ and $\underline{V}_{U/E}$ are written implicitly,

$$(\underline{P}^{-1} + \underline{G})\,\underline{T}_{Y/E} = \underline{P}^{-1}\underline{P}^*$$

$$\underline{T}_{Y/E} = (\underline{G} + \underline{Q}_d)^{-1}\,(\underline{Q}_d\underline{Q}^* - \underline{Q}_n\,\underline{T}_{Y/E}) \tag{3.25}$$

and,

$$(\underline{G}^{-1} + \underline{P})\,\underline{V}_{U/E} = -\,\underline{P}^*$$

$$\underline{V}_{U/E} = -\,(\underline{G}^{-1} + \underline{P}_d)^{-1}\,(\underline{P}^* + \underline{P}_n\,\underline{V}_{U/E}) \tag{3.26}$$

(where the plant has been expressed as the sum of matrices formed by its diagonal ($\underline{P}_d$) and non–diagonal ($\underline{P}_n$) elements, $\underline{P} = \underline{P}_d + \underline{P}_n$, $\underline{Q} = \underline{P}^{-1} = \underline{Q}_d + \underline{Q}_n$, $\underline{Q}^* = \underline{Q}_d^{-1}\, \underline{P}^{-1}\underline{P}^*$ and $[\underline{Q}]_{ij} = \frac{1}{q_{ij}}$ – the inverse of the elements of the plant inverse).

Using eq(3.26) above, quantitative design to the plant input can be undertaken by substituting the bound, $\underline{B}$ for $\underline{V}_{D/E}$ when the latter appears on the right hand side of the equation and designing to one of the following inequalities,

$$|\underline{V}_{U/E}|_{ij} \leq [\ |(\underline{G}^{-1}+\underline{P}_d)^{-1}|(|\underline{P}^*|+|\underline{P}_n|\ \underline{B})\ ]_{ij} \leq b_{ij} \tag{3.27}$$

or,

$$|\underline{V}_{U/E}|_{ij} \leq |\ (\underline{G}^{-1}+\underline{P}_d)^{-1}\underline{P}^*\ |_{ij} + [\ |(\underline{G}^{-1}+\underline{P}_d)^{-1}\underline{P}_n|\ \underline{B}\ ]_{ij} \leq b_{ij} \tag{3.28}$$

(depending on whether the design will be undertaken as a G/(1+L) or GP*/(1+L) type design). The advantage of eq(3.25) and eq(3.26) for design is that only diagonal matrices are inverted (after the plant inverse has been calculated).

A very significant difference between eq(3.25) and eq(3.26) above is that the usual (output) QFT design equation, eq(3.25), makes use of the inverse of the elements of the plant inverse, while the input design makes use of entries of the plant matrix directly. As a result, if design is to be attempted for both input and output specifications, the plant set is not the same for the two sets of constraints (unless the plant is diagonal, in which case the problem is n – SISO designs). There appear to be two possibilities:

i) Draw two sets of boundaries for the input– and output– specifications using a nominal plant and the inverse of the nominal plant respectively. When the controller is designed, two nominal open loop transfer functions corresponding to the two nominal plants must be shaped simultaneously for their respective boundaries.

ii) Select a single nominal plant, not necessarily an element of either the plant set or the set of plant inverses, and draw boundaries for that nominal, based on the input and output specifications. The loop–shaping then uses the chosen nominal. This second approach will be especially attractive in a plant with small off–diagonal elements. This approach has been employed by the author in tackling the flying machine designs of Chapters 4, 5 and 6.

The QFT improvements of Horowitz (1982b) are incorporated into the design: As a controller is designed for a loop, the loop is closed and the problem re–formulated to make use of the additional information which becomes available.

3.4.2 The case of a 2–by–2 plant

To clarify the ideas above, consider a 2–by–2 plant, $\underline{P}$, with disturbance–to–output $\underline{P}^*$ and closed loop disturbance to output and input specifications, eq(3.23) and eq(3.24) respectively.

If the controller is diagonal and the first loop (g_1) is closed first, the design problem for the first loop is as follows.

The output specification is,

$$\left[\begin{bmatrix} \frac{1}{q_{11}} & \frac{1}{q_{12}} \\ \frac{1}{q_{21}} & \frac{1}{q_{22}} \end{bmatrix} + \begin{bmatrix} g_1 & 0 \\ 0 & g_2 \end{bmatrix}\right] \begin{bmatrix} t_{11} & t_{12} \\ t_{21} & t_{22} \end{bmatrix} = \begin{bmatrix} \frac{q^*_{11}}{q_{11}} & \frac{q^*_{12}}{q_{11}} \\ \frac{q^*_{21}}{q_{22}} & \frac{q^*_{22}}{q_{22}} \end{bmatrix} \tag{3.29}$$

$$|t_{11}| = \left|\frac{q^*_{11}}{1+q_{11}g_1} - \frac{1}{1+q_{11}g_1}\frac{q_{11}}{q_{12}}t_{21}\right| \le \left|\frac{1}{1+q_{11}g_1}\right|\left[|q^*_{11}| + \left|\frac{q_{11}}{q_{12}}\right|a_{21}\right] \le a_{11}$$

or,

$$|t_{11}| \le \left|\frac{1}{g_1}\right|\left|\frac{q_{11}g_1}{1+q_{11}g_1}\right|\left[\left|\frac{q^*_{11}}{q_{11}}\right| + \left|\frac{1}{q_{12}}\right|a_{21}\right] \le a_{11} \tag{3.30}$$

and,

$$|t_{12}| = \left|\frac{q^*_{12}}{1+q_{11}g_1} - \frac{1}{1+q_{11}g_1}\frac{q_{11}}{q_{12}}t_{22}\right| \le \left|\frac{1}{1+q_{11}g_1}\right|\left[|q^*_{12}| + \left|\frac{q_{11}}{q_{12}}\right|a_{22}\right] \le a_{12}$$

or,

$$|t_{12}| \le \left|\frac{1}{g_1}\right|\left|\frac{q_{11}g_1}{1+q_{11}g_1}\right|\left[\left|\frac{q^*_{12}}{q_{11}}\right| + \left|\frac{1}{q_{12}}\right|a_{22}\right] \le a_{12} \tag{3.31}$$

Notice here that although over–design can be reduced by selecting one of the two design equations, the use of the Schwartz inequality and the use of specification, a_{ij}, instead of the actual transfer function value, t_{ij}, results in over design.

The input specification is,

$$\left[\begin{bmatrix} \frac{1}{g_1} & 0 \\ 0 & \frac{1}{g_2} \end{bmatrix} + \begin{bmatrix} p_{11} & p_{12} \\ p_{21} & p_{22} \end{bmatrix}\right] \begin{bmatrix} v_{11} & v_{12} \\ v_{21} & v_{22} \end{bmatrix} = - \begin{bmatrix} p^*_{11} & p^*_{12} \\ p^*_{21} & p^*_{22} \end{bmatrix} \tag{3.32}$$

$$|v_{11}| = \left|\frac{p^*_{11}g_1}{1+p_{11}g_1} + \frac{p_{12}g_1}{1+p_{11}g_1}v_{21}\right| \le \left|\frac{g_1}{1+p_{11}g_1}\right|\left[|p^*_{11}| + |p_{12}|b_{21}\right] \le b_{11}$$

or,

$$|v_{11}| \leq \left|\frac{p_{11}g_1}{1+p_{11}g_1}\right| \left[\left|\frac{p_{11}^*}{p_{11}}\right| + \left|\frac{p_{12}}{p_{11}}\right| b_{21}\right] \leq b_{11} \tag{3.33}$$

and,

$$|v_{12}| = \left|\frac{p_{12}^*g_1}{1+p_{11}g_1} + \frac{p_{12}g_1}{1+p_{11}g_1} v_{22}\right| \leq \left|\frac{g_1}{1+p_{11}g_1}\right| \left[|p_{12}^*| + |p_{12}| b_{22}\right] \leq b_{12}$$

or,

$$|v_{12}| \leq \left|\frac{p_{11}g_1}{1+p_{11}g_1}\right| \left[\left|\frac{p_{12}^*}{p_{11}}\right| + \left|\frac{p_{12}}{p_{11}}\right| b_{22}\right] \leq b_{12} \tag{3.34}$$

The first loop is designed to satisfy equations (3.31) to (3.34). This loop is closed and the design equations become,

$$|t_{2i}| = \left|\frac{q_{2i}^{*\prime}}{1+q_2'g_2}\right| \leq a_{2i} \quad (i=1,2) - \text{output specification} \tag{3.35}$$

$$|t_{2i}| = \left|\frac{q_{2i}^{*\prime}}{1+q_2'g_2}\right| \leq a_{2i} \quad (i=1,2) - \text{output specification} \tag{3.35}$$

with,

$$\frac{1}{q_2'} = \frac{1}{q_{22}} - \frac{q_{11}}{q_{12}q_{21}} \frac{1}{1+q_{11}g_1}$$

$$q_{2i}^{*\prime} = q_2' \left[\frac{q_{2i}^*}{q_{22}} - \frac{q_{1i}^*}{q_{21}} \frac{1}{1+q_{11}g_1}\right]$$

and,

$$|v_{2i}| = \left|\frac{g_2 p_{2i}^{*\prime}}{1+p_2'g_2}\right| \leq b_{2i} \quad (i=1,2) - \text{input specification} \tag{3.36}$$

with,

$$p_2' = p_{22} - \frac{p_{12}p_{21}}{p_{11}} \frac{p_{11}g_1}{1+p_{11}g_1}$$

$$p_{2i}^* = p_{2i}^* - \frac{p_{21}}{p_{11}} \frac{p_{11}g_1}{1+p_{11}g_1} p_{1i}^*$$

Now, a little algebra or thought indicates that $p_2' = q_2'$ and $p_{2i}^{*\prime} = q_{2i}^{*\prime}$ (since the whole design has been reduced to an exact SISO problem). In the second stage of the design, there is a single plant set and the design is very much more simple.

3.4.3 Stability

The poles of $\det((\underline{I} + \underline{G}\ \underline{P})^{-1})$ and of $\det((\underline{I} + \underline{P}\ \underline{G})^{-1}\underline{P}\ \underline{G})$ are identical so that stability need only be assured using input or output design. (This is analogous to the SISO case where $\frac{1}{1+L}$ and $\frac{L}{1+L}$ have the same poles).

PROOF

Let, $\underline{G}(s) = \underline{X}(s)\,\underline{Y}(s)^{-1}$ and $\underline{P}(s) = \underline{D}(s)^{-1}\,\underline{N}(s)$
with $\underline{X}(s)$, $\underline{Y}(s)$, $\underline{D}(s)$ and $\underline{N}(s)$ non–singular polynomial matrices (Kailath, 1980).
Then,

$$\det((\underline{I} + \underline{G}\,\underline{P})^{-1}) = \det((\underline{I} + \underline{X}\,\underline{Y}^{-1}\,\underline{D}^{-1}\,\underline{N})^{-1})$$

$$= \det((\underline{N}^{-1}\,(\underline{D}\,\underline{Y} + \underline{N}\,\underline{X})\,\underline{Y}^{-1}\,\underline{D}^{-1}\,\underline{N})^{-1})$$

$$= \frac{\det(\underline{D})\ \det(\underline{Y})\ \det(\underline{N})}{\det(\underline{N})\ \det(\underline{D}\,\underline{Y} + \underline{N}\,\underline{X})}$$

$$= \frac{\det(\underline{D})\ \det(\underline{Y})}{\det(\underline{D}\,\underline{Y} + \underline{N}\,\underline{X})} \qquad (3.37)$$

and,

$$\det((\underline{I} + \underline{P}\,\underline{G})^{-1}\,\underline{P}\,\underline{G}) = \det((\underline{I} + \underline{D}^{-1}\,\underline{N}\,\underline{X}\,\underline{Y}^{-1})^{-1}\,\underline{D}^{-1}\,\underline{N}\,\underline{X}\,\underline{Y}^{-1})$$

$$= \det((\underline{D}^{-1}(\underline{D}\,\underline{Y} + \underline{N}\,\underline{X})\,\underline{X}^{-1}\,\underline{N}^{-1}\,\underline{D})^{-1})$$

$$= \frac{\det(\underline{N})\ \det(\underline{X})\ \det(\underline{D})}{\det(\underline{D})\ \det(\underline{D}\,\underline{Y} + \underline{N}\,\underline{X})}$$

$$= \frac{\det(\underline{N})\ \det(\underline{X})}{\det(\underline{D}\,\underline{Y} + \underline{N}\,\underline{X})} \qquad (3.38)$$

Equations (3.37) and (3.38) indicate that the right hand plane poles of the closed loop system are determined by the zeros of $\det(\underline{D}\ \underline{Y} + \underline{N}\ \underline{X})$ (see Nwokah, 1986, for a polynomial matrix based approach, similar to the above, for achieving robust stability without using singular–value type robustness measures).

Eq(3.37) shows that right hand plane poles in $\underline{P}$ (i.e. in $\det(\underline{D})$) result in non–minimum phase behaviour at the plant input.

Eq(3.38) indicates that right hand plane transmission zeros (Kailath, 1980) in $\underline{P}$ (or equivalently in $\underline{N}$) must appear in the closed loop input–output response since $\det(\underline{D}\,\underline{Y} + \underline{N}\,\underline{X})$ cannot sensibly be designed to cancel such zeros.

An interesting hypothesis which will not be investigated here is as follows. Since the design method is conservative, if stability alone is required over the plant uncertainty set, only the easiest of stability of the input or of the output need be satisfied.

The achievement of stability using this design method can be delayed until the final stage of the design although one would usually try to design (in the 2×2 case) $(1+g_1q_{11})$ and $(1+g_1p_{11})$ to be minimum phase so that no additional difficulty is obtained in the design of the second loop. Doyle's paper criticizing QFT (1986) failed to note this point.

An arcwise connected set $\mathcal{P}=\{P\}$ is defined such that $\forall\ P_1,P_2 \in\mathcal{P}$, P_1 can be perturbed to P_2 such that at any point on the path $\exists$ arbitrarily small neighbourhood in which all points on the path have arbitrarily close pole–zero structure (Tzafestas, 1984). For each arcwise connected sub–set of the plant set, stability only needs to be assured for one element (of that subset) to guarantee stability for the whole subset if there are maximum magnitude bounds and they are satisfied. In other words,

> "It is worth noting that even if the above proof [based on Schauder's fixed–point theorem] were not available, it would not be disastrous for this synthesis theory. It would only be necessary to guarantee that one $m\in\mathcal{M}$ [set of parameterisations of the plant], the system is stable and minimum phase. For then, this would be so $\forall m\in\mathcal{M}$, because by the continuity of the poles (and zeros) with respect to the parameters, the right hand side [of the equation of the transfer function] would have to be infinite (zero) at some ω in order that for some $m\in\mathcal{M}$ the system should be unstable (have a right half plane zero). However this synthesis procedure by definition precludes this."
>
> Horowitz, 1979

3.4.4 Some comments on the designs executed in this thesis

In the design of controllers for this research the following general points emerged:

a) At sufficiently low frequencies very great uncertainty reduction, disturbance rejection and input–output decoupling can be achieved by having sufficient gain in the loops which are to be closed. The constraint on the loop gain is as a result of position saturation limits at these frequencies – this limits the maximum controller roll–off and hence the low frequency gain (Eitelberg and Boje, 1989).

b) At intermediate frequencies (around the gain and phase cross–over frequencies in single loop designs) there is real design difficulty to achieve stability and large bandwidth while simultaneously limiting input levels.

c) At high frequency, actuator rate saturation and roll–off due to unmodelled dynamics cause some theoretical difficulties: Does it matter that the loop gain goes to zero very rapidly after the phase–cross over frequency? What is the consequence of the actuators not being able to respond in a linear fashion over the bandwidth of the loop transfer function? To avoid such problems one might modify Horowitz's "Universal high frequency bound" (UHFB), (Horowitz and Sidi, 1972) by making a bound which results from the plant high frequency gain uncertainty and infinite phase uncertainty. Obviously this latter bound precludes the achievement of "arbitrarily small sensitivity over arbitrarily large bandwidth" but such a desire is practically unreasonable (though theoretically interesting). Figure 3.4 compares the Horowitz UHFB to this bound for a single loop design with $|L/(1+L)| \leq 3$dB and $\Delta P(j\omega) = 10$dB.

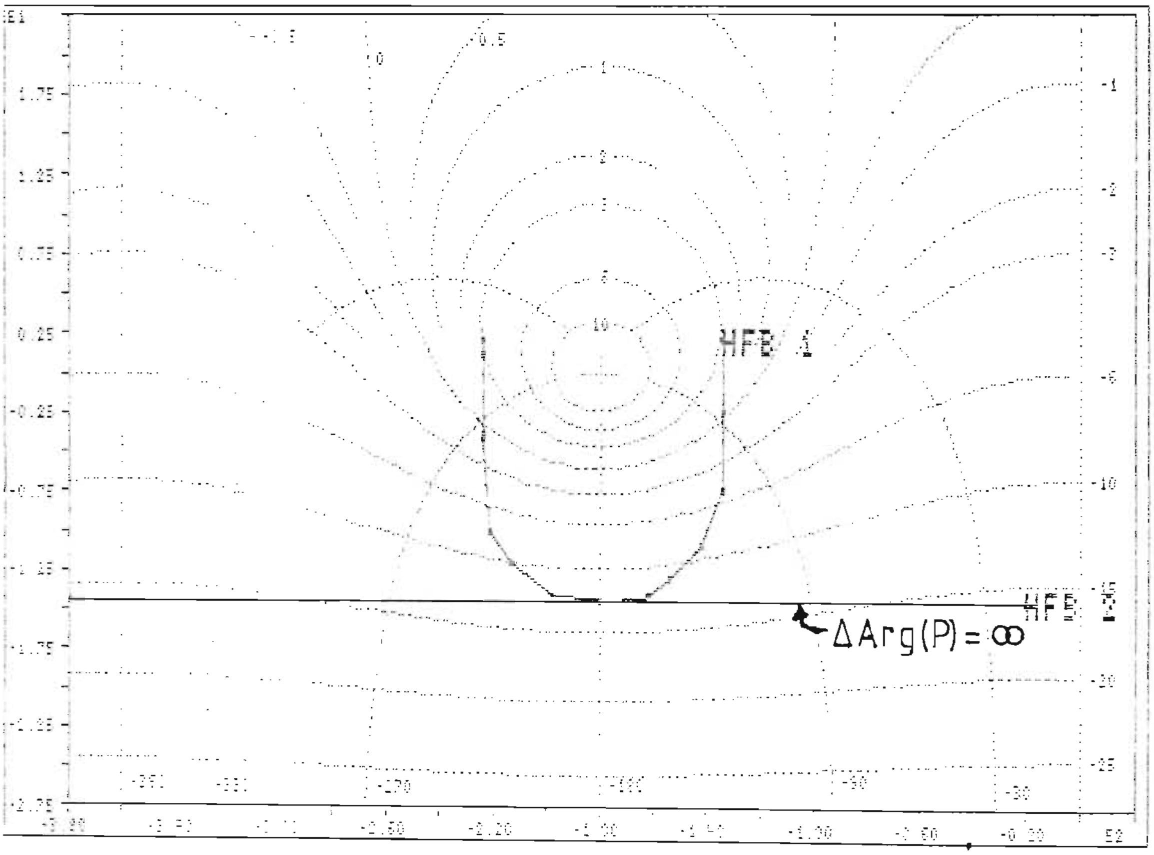


Figure 3.4 – Universal high frequency bounds with and without phase uncertainty

CHAPTER 4 – CONTROLLER DESIGN BASED ON DECOUPLING OF THE DISTURBANCE–TO–OUTPUT BEHAVIOUR

4.1 Design

4.1.1 Introduction and design equations

The first attempt at controlling the flying machine was based on the heuristically attractive notion of diagonalizing the plant model structure in some way (not necessarily by canceling the plant matrix by its inverse at the nominal point), so as to allow independent, single loop controllers to be designed for the height h(t) and pitch angle, $\theta(t)$.

If,

i) the wing actuators for the front and back wings are fast ($\hat{\alpha} = \alpha$, $\hat{\beta} = \beta$),
ii) the reaction torque on the body induced by the angular acceleration of the wings in turning can be ignored,
iii) the coefficients $k_\ell = k_{\ell f} + k_{\ell b}$ and $k_t = k_{tb} - k_{tf}$, and
iv) the terms, A and B in the model are small and can be ignored,
then the model, eq(2.21), can be written for the wing angles with respect to ground (implicitly with respect to $\theta(s)$),

$$\begin{bmatrix} Js^2+\mu_t s & 0 \\ 0 & ms^2+\mu_\ell s \end{bmatrix} \begin{bmatrix} \theta(s) \\ H(s) \end{bmatrix} = \begin{bmatrix} k_{tf} & -k_{tb} \\ k_{\ell f} & k_{\ell b} \end{bmatrix} \begin{bmatrix} \alpha(s)+\theta(s) \\ \beta(s)+\theta(s) \end{bmatrix} + \begin{bmatrix} E_t(s) \\ E_\ell(s) \end{bmatrix} \tag{4.1}$$

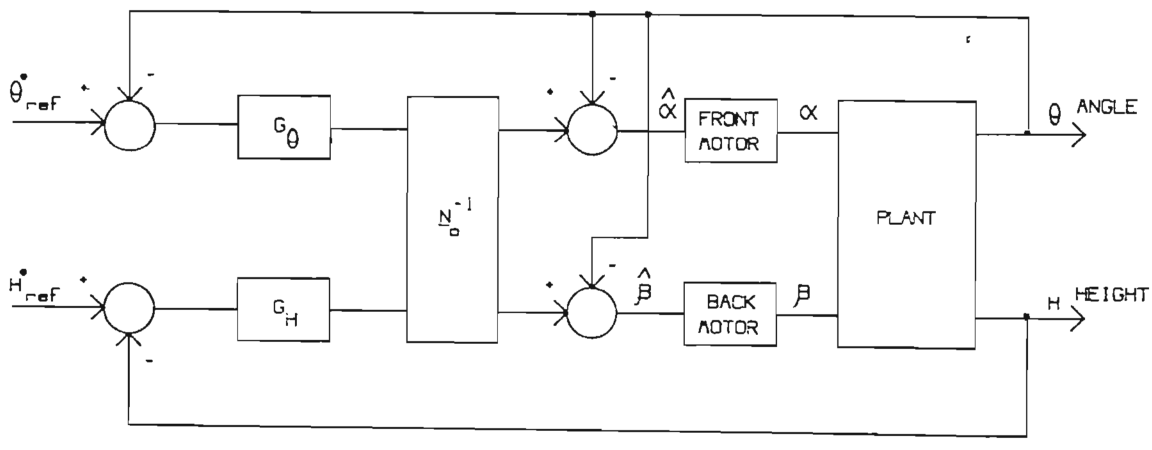


Figure 4.1 – Disturbance–to–output decoupling controller

Two single-loop controllers are designed for the problem (shown in Figure 4.1),

$$\begin{bmatrix} Js^2+\mu_t s & 0 \\ 0 & ms^2+\mu_\ell s \end{bmatrix} \begin{bmatrix} \theta(s) \\ H(s) \end{bmatrix} = \begin{bmatrix} k_\theta U_\theta(s) \\ k_H U_H(s) \end{bmatrix} + \begin{bmatrix} E_t(s) \\ E_\ell(s) \end{bmatrix} \tag{4.2}$$

or,

$$\theta(s) = P_\theta(s)\, U_\theta(s) + P_\theta(s)\, E_t(s) \tag{4.3}$$

with,

$$P_\theta(s) = \frac{k_\theta}{Js^2+\mu_t s}$$

and,

$$H(s) = P_H(s)\, U_H(s) + P_H(s)\, E_\ell(s) \tag{4.4}$$

with,

$$P_H(s) = \frac{k_H}{ms^2+\mu_\ell s}$$

to achieve disturbance reduction and command following of some sort. k_θ and k_H attempt to model the gain uncertainty in the numerator, $\underline{\mathbf{N}}$, for quantitative design by using equivalent plants,

$$P_{\theta(2i-1)}(s) = \frac{k_{tb}/k_{tb0}}{Js^2+\mu_t s} \qquad P_{\theta(2i)}(s) = \frac{k_{tf}/k_{tf0}}{Js^2+\mu_t s}$$

$$P_{H(2i-1)}(s) = \frac{k_{\ell b}/k_{\ell b0}}{ms^2+\mu_\ell s} \qquad P_{H(2i)}(s) = \frac{k_{\ell f}/k_{\ell f0}}{ms^2+\mu_\ell s}$$

(i=1,..,NumberOfPlants). This procedure doubles the number of plants in the plant set and selects the extreme gain deviations away from the nominal plant while preserving the relationship between derived variables. The plant set of eq(4.1) is not necessarily a subset of that of eq(4.2) and some approximation has been made. A large proportion of the gain uncertainty comes from squaring the uncertain wind speed in the lift and drag equations, eq(2.25) and eq(2.26) and the procedure above will correctly take this uncertainty into account. The singular value based "modern" control theory (Doyle and Stein, 1981) uses a more arbitrary uncertainty parameterisation than the one above. This makes it difficult to justify using such methods for solving (by design) practical problems.

The multivariable plant is controlled by setting,

$$\begin{bmatrix} \hat{\alpha}(s) \\ \hat{\beta}(s) \end{bmatrix} = \begin{bmatrix} k_{tf0} & -k_{tb0} \\ k_{\ell f0} & k_{\ell b0} \end{bmatrix}^{-1} \begin{bmatrix} U_\theta(s) \\ U_H(s) \end{bmatrix} - \begin{bmatrix} \theta(s) \\ \theta(s) \end{bmatrix} \tag{4.5}$$

where, $\hat{\alpha}(s)$ and $\hat{\beta}(s)$ are the front and back wing command signals respectively.

Remembering that the feedback path has a sign inversion, the controller with respect to the input–output behaviour is given by,

$$\underline{G}(s) = \begin{bmatrix} k_{tf0} & -k_{tb0} \\ k_{\ell f0} & k_{\ell b0} \end{bmatrix}^{-1} \begin{bmatrix} g_\theta(s) & 0 \\ 0 & g_H(s) \end{bmatrix} + \begin{bmatrix} 1 & 0 \\ 1 & 0 \end{bmatrix}$$

$$= \frac{1}{k_0} \begin{bmatrix} k_{\ell b0}\, g_\theta(s)+k_0 & k_{tb0}\, g_H(s) \\ -k_{\ell f0}\, g_\theta(s)+k_0 & k_{tf0}\, g_H(s) \end{bmatrix} \tag{4.6}$$

with,

$$k_0 = k_{tf0}k_{\ell b0} + k_{tb0}k_{\ell f0}$$

4.1.2 Right hand plane zeros in G(s)

The equivalent controller $\underline{G}(s)$ in eq(4.6) has right hand plane transmission zeros if $\det(\underline{G}(s)) = g_H(s)[g_\theta(s)-(k_{tb0}-k_{tf0})]$ is non-minimum phase. (Clearly both $g_\theta(s)$ and $g_H(s)$ will be designed stable and therefore $\underline{G}(s)$ will be stable.) $g_H(s)$ will be designed minimum phase and but $[g_\theta(s)-(k_{tb0}-k_{tf0})]$ is non-minimum phase if $-g_\theta(j\omega)$ encircles $(-(k_{tb0}-k_{tf0}),j0)$ on the Nyquist diagram. For a low order controller, this condition is equivalent to requiring $g_\theta(s) < (k_{tb0}-k_{tf0}) = 17.9$ (see Figure 4.2). Limiting the gain of the pitch angle controller avoids a change in sign of $[g_\theta(s)-(k_{tb0}-k_{tf0})]$ between $s=0$ and $s=\infty$, showing via the initial and final value theorems that the controller is minimum phase (or has an odd number of right hand plane transmission zeros). This result on non-minimum phase behaviour is recalled from a publication the title and author of which have not been traced.

The (2,1) entry of eq(4.6) is non-minimum phase if $\left[k_{\ell f0}/k_0\, g_\theta(j\omega) - 1\right]$ has a right hand plane zero, or equivalently if $-\, g_\theta(j\omega)$ has an encirclement of $(-k_0/k_{\ell f0}, j0)$. For a low order controller this requirement is $g_\theta(j0) < k_0/k_{\ell f0} = 122$. This is a weaker condition than for no right hand plane transmission zeros in $\underline{G}(s)$ above.

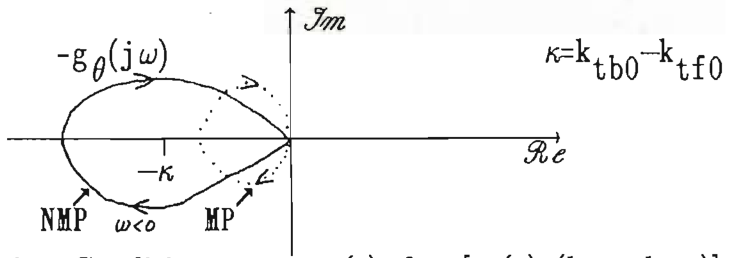


Figure 4.2 Conditions on $-g_\theta(s)$ for $[g_\theta(s) - (k_{tb0} - k_{tf0})]$ to be minimum / non-minimum phase.

The (1,1) entry of $\underline{G}(s)$ in eq(4.6) will only be non minimum phase if $\left[k_{\ell b0}/k_0\ g_\theta(j\omega)\right]$ encircles (-1,j0) on the Nyquist diagram. For the chosen nominal and a low order controller, this would require $g_\theta(j\omega_{\varphi c}) < 40$, where $\omega_{\varphi c}$ is the phase cross over frequency. This possibility is unlikely since at the phase cross-over frequency, a roll-off of about -40dB/decade must have been sustained for about 1 decade (Horowitz, 1963, Chapter 7), suggesting a steady state gain (of g_θ) in excess of 70dB would be required, making it unlikely that this function will be non minimum phase (see Figure 4.3.).

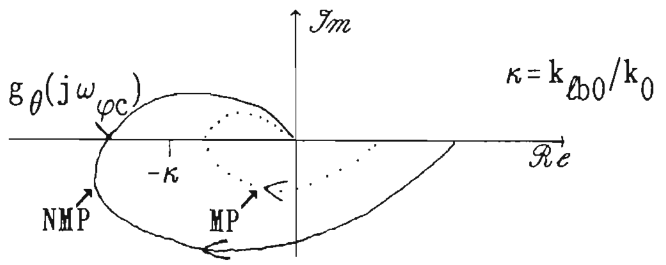


Figure 4.3 Conditions on $g_\theta(s)$ for $\left[k_{\ell b0}/k_0 g_\theta(j\omega) - 1\right]$ to be minimum / non-minimum phase.

It is not clear what the implications of having a controller with right hand plane transmission zeros but with minimum phase paths is. Eq(3.38) indicates that the transmission zeros will appear in the input–output response.

Specifications

In order to have some specifications note that for good disturbance rejection,

$$|\theta(s)/E_t(s)| = \left|\frac{P_\theta/k_\theta}{1+P_\theta g_\theta}\right| = \frac{|P_\theta|}{|k_\theta|}\left|\frac{1}{1+L_\theta}\right| \quad (P_\theta = \frac{k_\theta}{Js^2+\mu_t s}) \tag{4.7}$$

$$|H(s)/E_\ell(s)| = \left|\frac{P_H/k_H}{1+P_H g_H}\right| = \frac{|P_H|}{|k_H|}\left|\frac{1}{1+L_H}\right| \quad (P_H = \frac{k_H}{ms^2+\mu_\ell s}) \tag{4.8}$$

must be as small as possible. This is achieved for a low-passing $P_\theta(s)$ by ensuring that $\left|\frac{1}{1+L_\theta}\right|$ is small until $|P_\theta|$ becomes small, or equivalently by making the open loop bandwidth as large as possible. Three facts prevent the realization of high (infinite) loop bandwidth (with attendant infinitely good feedback properties) in this practical problem:

i) The plant has unmodelled dynamics at high frequency and an uncertain excess of poles over zeros. The plant templates resulting from this uncertainty become too wide at high frequency to allow the open loop transmission to roll off without violating the stability constrains if that roll off is initiated at too high a frequency.

ii) Any non-minimum phase behaviour (from right hand plane zeros or transport lag) result in so much phase lag at high frequency that stability becomes unachievable if the bandwidth specified is not compatible with the plant behaviour.

iii) The high controller gain required to achieve high bandwidth will invariably cause saturation at the plant input.

Examine the inputs, U_θ and U_H

$$\begin{bmatrix} u_\theta(t) \\ u_H(t) \end{bmatrix} = \begin{bmatrix} k_{tf}/k_\theta & -k_{tb}/k_\theta \\ k_{\ell f}/k_H & k_{\ell b}/k_H \end{bmatrix} \begin{bmatrix} \alpha(t) + \theta(t) \\ \beta(t) + \theta(t) \end{bmatrix} \tag{4.9}$$

The specifications, $\sigma_\alpha = \sigma_\beta \leq 0.5$ rad (eq(2.32a)), $\sigma_{\dot\alpha} = \sigma_{\dot\beta} \leq 2$ rad/s (eq(2.32b)) and $\sigma_\theta \leq 0.15$ rad, for the nominal plant (which has high gain) translate very (!) roughly to,

$$\sigma_{u_\theta} \approx (0.7\times0.5)\times\sqrt{k_{tf0}^2+k_{tb0}^2} = 18.0 \tag{4.10}$$

$$\sigma_{\dot u_\theta} \approx (0.7\times2.0)\times\sqrt{k_{tf0}^2+k_{tb0}^2} = 71.2 \tag{4.11}$$

$$\sigma_{u_H} \approx (0.7\times0.5)\times\sqrt{k_{\ell f0}^2+k_{\ell b0}^2} = 67.2 \tag{4.12}$$

$$\sigma_{\dot u_H} \approx (0.7\times2.0)\times\sqrt{k_{\ell f0}^2+k_{\ell b0}^2} = 269.0 \tag{4.13}$$

This is based on the variance of the sum of two independent, zero mean random variables (which α and β are not!), x and y,

$$\mathscr{E}\{(ax+by)^2\} = a^2\mathscr{E}\{x^2\} + b^2\mathscr{E}\{y^2\} + 2ab\mathscr{E}\{xy\}$$
$$= a^2\mathscr{E}\{x^2\} + b^2\mathscr{E}\{y^2\} \qquad (4.14)$$

and allows half of the available input power for each output (the factor $0.7 \approx 1/\sqrt{2}$).

One could attempt to formally calculate the variance of the modified inputs, for example,

$$\mathscr{E}\{u_\theta^2(t)\} = \mathscr{E}\{(k_{tf}/k_\theta(\alpha+\theta) - k_{tb}/k_\theta(\beta+\theta))^2\}$$
$$= \frac{1}{k_\theta^2}\Big[k_{tf}^2\sigma_\alpha^2 + k_{tb}^2\sigma_\beta^2 + (k_{tf}-k_{tb})^2\sigma_\theta^2 +$$
$$2k_{tf}(k_{tf}-k_{tb})\sigma_{\theta\alpha}^2 - 2k_{tb}(k_{tb}-k_{tf})\sigma_{\theta\beta}^2 - 2k_{tb}k_{tf}\sigma_{\beta\alpha}^2\Big] \qquad (4.15)$$

(if all signals have zero mean), etc. As there is no information on the covariances $\mathscr{E}\{\alpha\beta\}$, $\mathscr{E}\{\alpha\theta\}$ etc, this procedure looks pointless and the guess above will have to do.

Since the external disturbance to input transfer is given by,

$$|U_\theta/E_t| = \left|\frac{g_\theta P_\theta/k_\theta}{1+g_\theta P_\theta}\right| \qquad (4.16)$$

and,

$$|U_H/E_\ell| = \left|\frac{g_H P_H/k_H}{1+g_H P_H}\right| \qquad (4.17)$$

k=1 from eq(3.10). The bandwidths obtained via eq(3.17) and eq(3.20)for $\zeta = 0.4$ (corresponding to about 3dB overshoot) and k=1 are

Torque (pitch angle) loop

$$\omega_{n\theta} = \min\{(324/0.14/1.96),\ (\sqrt[3]{5069/0.14/1.85})\}$$
$$= 27 \text{ rad/s} \qquad (4.18)$$

Lift (height loop)

$$\omega_{nH} = \min\{(4516/1.9/1.96),\ (\sqrt[3]{7.236\times10^4/1.9/1.85})\}$$
$$= 27 \text{ rad/s} \qquad (4.19)$$

In both loops the bandwidth is limited by the input rate specification rather than the input amplitude specification. The design specifications are:

Pitch angle (4.20)

$|L_\theta(j\omega)| \leq 0$ dB at ω=30rad/s - the cut-off frequency of $L_\theta/(1+L_\theta)$

$|L_\theta/(1+L_\theta)| \leq 3$ dB $\forall\ \omega$ - "damping factor" of $L_\theta/(1+L_\theta)$, $\zeta_\theta \approx 0.4$

$|1/(1+L_\theta)| \leq 3$ dB $\forall\ \omega$ - stability margin for all plants

Height (4.21)

$|L_H(j\omega)| \leq 0$ dB at ω=30rad/s - the cut-off frequency of $L_H/(1+L_H)$
$|L_H/(1+L_H)| \leq 3$ dB $\forall\ \omega$ - "damping factor" of $L_H/(1+L_H)$, $\zeta_H \approx 0.4$.
$|1/(1+L_H)| \leq 3$ dB $\forall\ \omega$ - stability margin for all plants

Using the above considerations, controllers were designed and implemented.

Design of pitch angle loop

Figure 4.4 shows plant (P_θ) templates and bounds on $L_{\theta 0}(s) = g_\theta(s)\, m(s)\, P_{\theta 0}(s)$, ($P_{\theta 0}(s) = \dfrac{1}{Js^2+\mu_{t0}s}$ and m(s) accounts for the actuator (motor) dynamics) at ω=1 and ω=30 rad/s for the constraints, eq(4.20). Templates at other frequencies are very similar, having very little phase uncertainty ($\simeq 5°$) and about 10dB gain uncertainty. The controller designed for this loop assumed that the actuator behaviour, m(s), is dominantly second order with ω_n=170 rad/s and ζ=0.4. The resulting $g_\theta(s)$ is,

$$g_\theta(s) = \frac{30\ (s/12 + 1)}{(s/100)^2+2\times 0.7\times(s/100)+1} \tag{4.22}$$

Figure 4.4 and Figure 4.5 show $L_{\theta 0}(j\omega)$ on the Nichols chart for $\omega_{n\ motor}$ = 170 and 100 rad/s respectively.

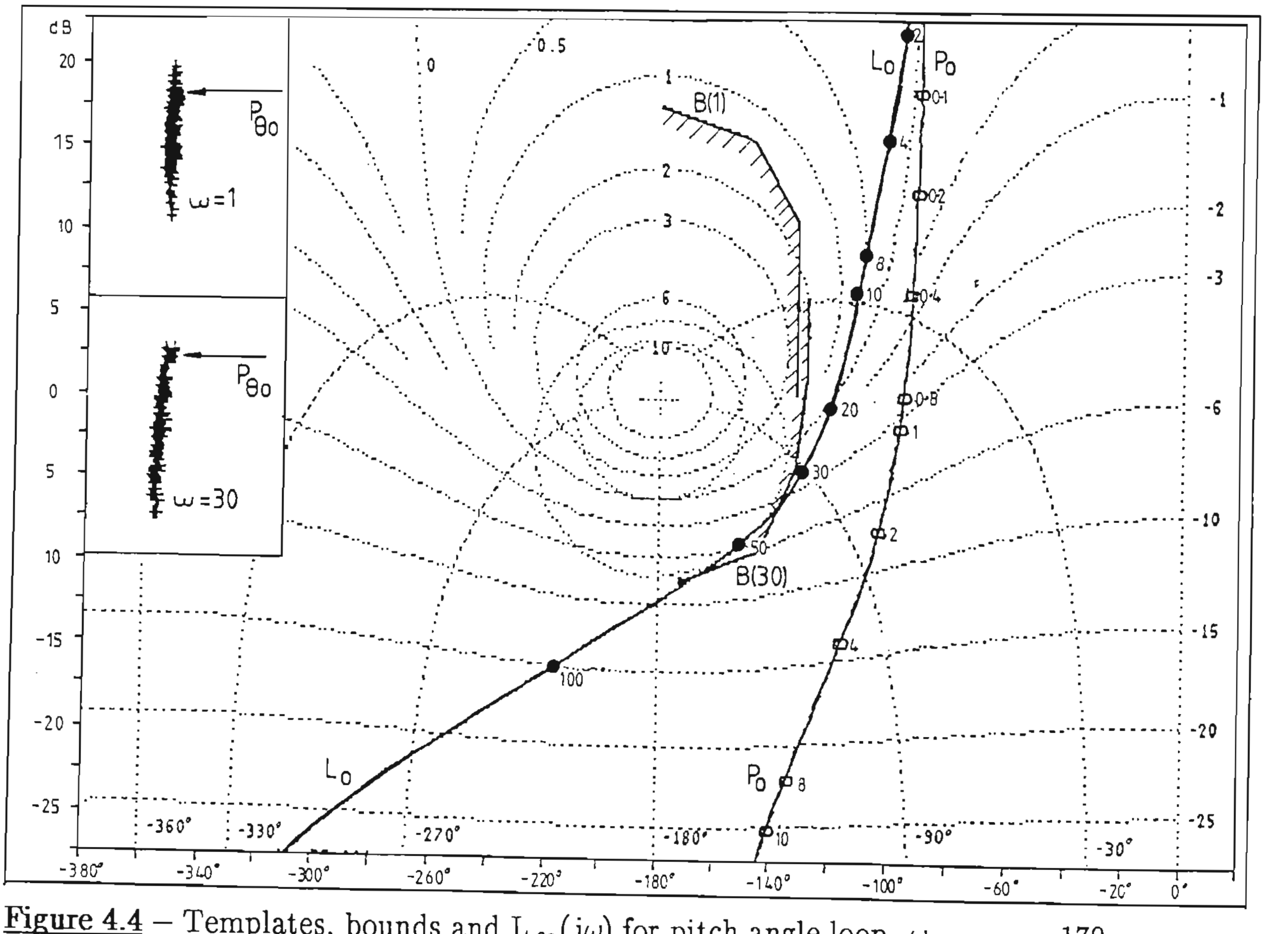


Figure 4.4 – Templates, bounds and $L_{\theta 0}(j\omega)$ for pitch angle loop. $\omega_{n\ motor}$=170 rad/s

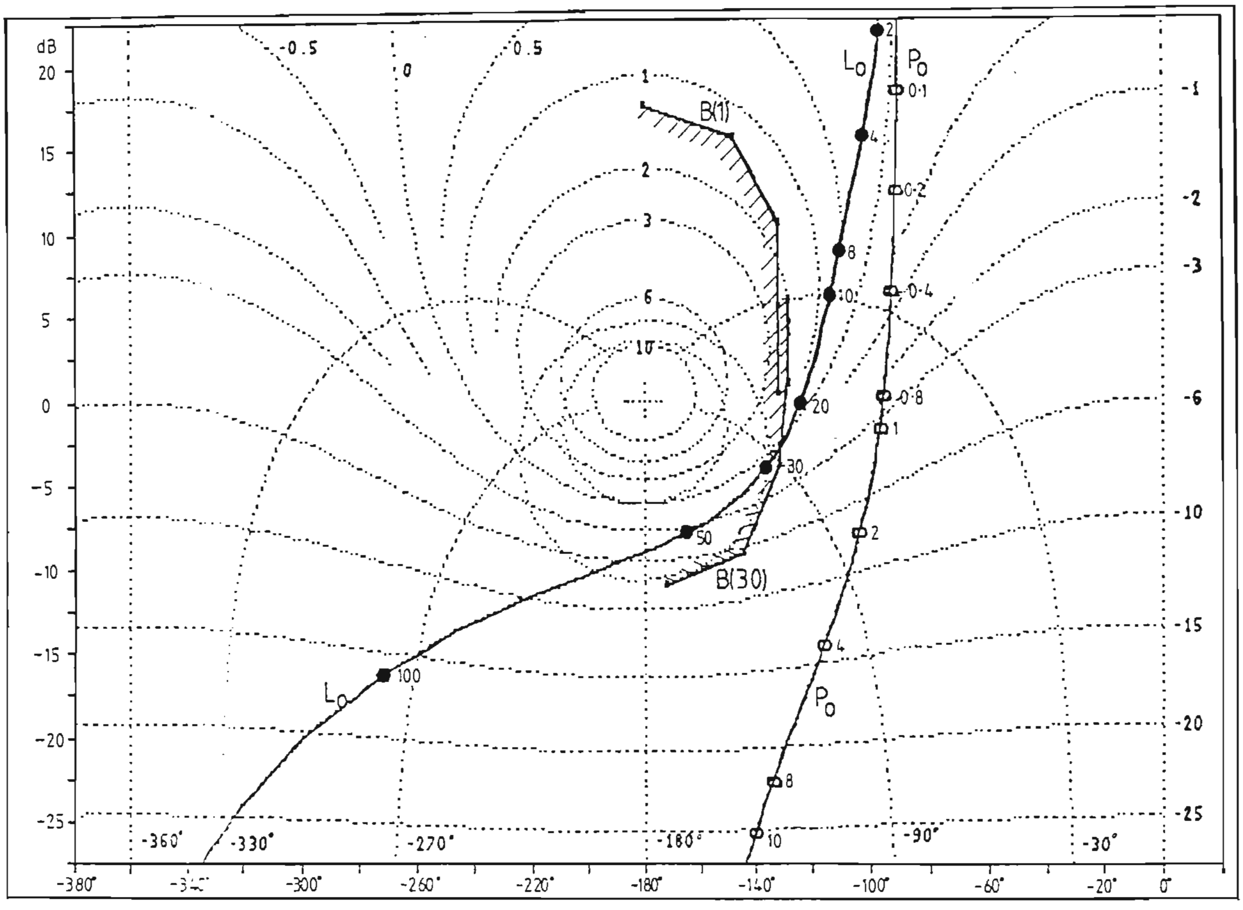


Figure 4.5 – Bounds and $L_{\theta 0}(j\omega)$ for pitch angle loop. $\omega_{n\ motor}$=100 rad/s

Height loop

Figure 4.6 shows plant (P_H) templates and bounds on $L_{H0}(s) = g_H(s)\ m(s)\ P_{H0}(s)$, $\left(P_{H0}(s) = \frac{1}{ms^2+\mu_{\ell 0}s}\right)$ and m(s) accounts for the actuator (motor) dynamics) at ω=1 and ω=30 rad/s for the constraints, eq(4.21). Templates at other frequencies are very similar, having very little phase uncertainty ($\simeq 5°$) and about 10dB gain uncertainty. The controller designed for this loop assumed that the actuator behaviour, m(s), is dominantly second order with ω_n=170 rad/s and ζ=0.4. The resulting $g_H(s)$ is,

$$g_H(s) = \frac{150\ (s/12 + 1)}{(s/50)^2+2\times0.7\times(s/50)+1} \tag{4.23}$$

Figure 4.6 and Figure 4.7 show $L_{H0}(j\omega)$ on the Nichols chart for ω_n(motor) = 170 and 100 rad/s respectively.

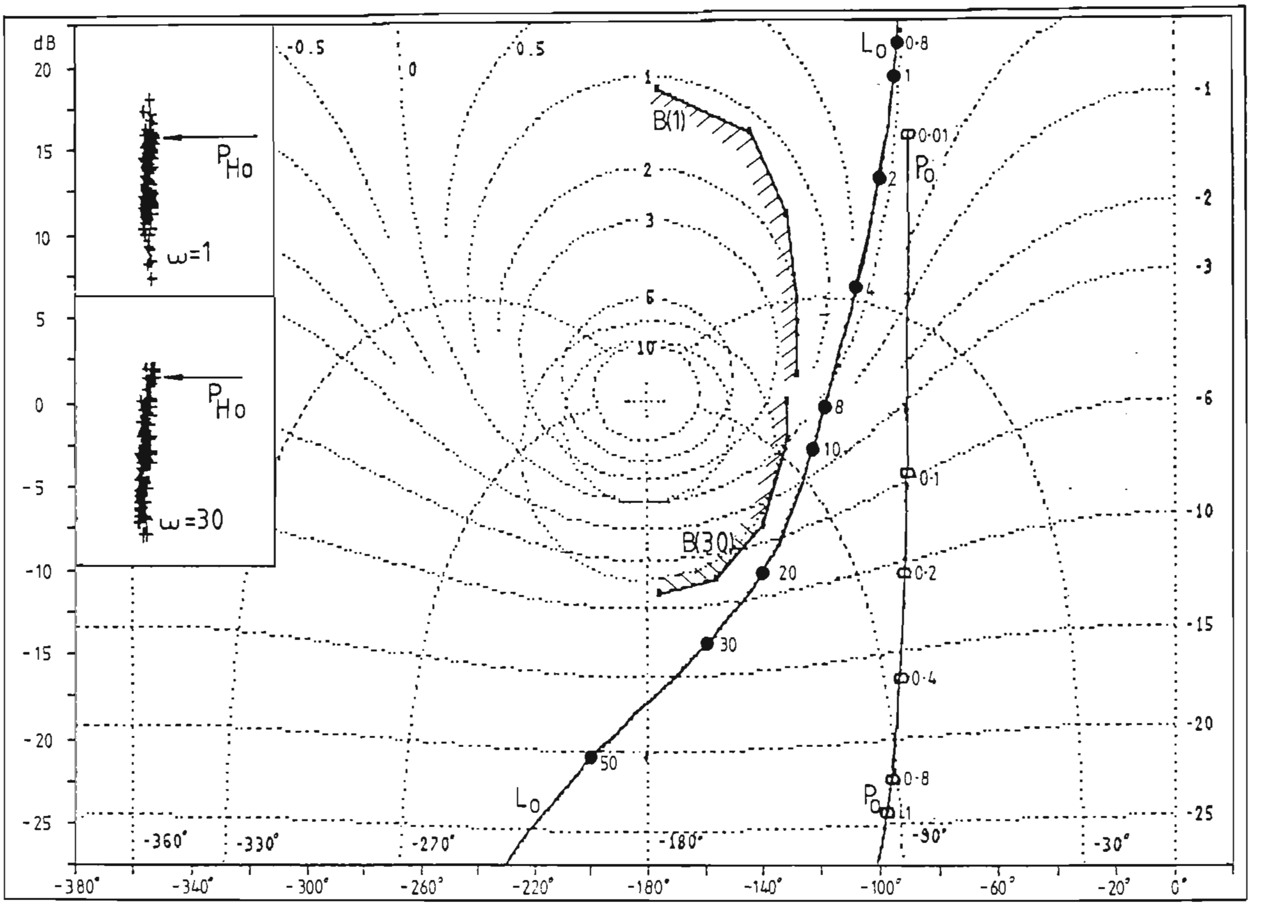


Figure 4.6 – Templates, bounds and $L_{H0}(j\omega)$ for height loop. $\omega_{n\ motor}$=170 rad/s

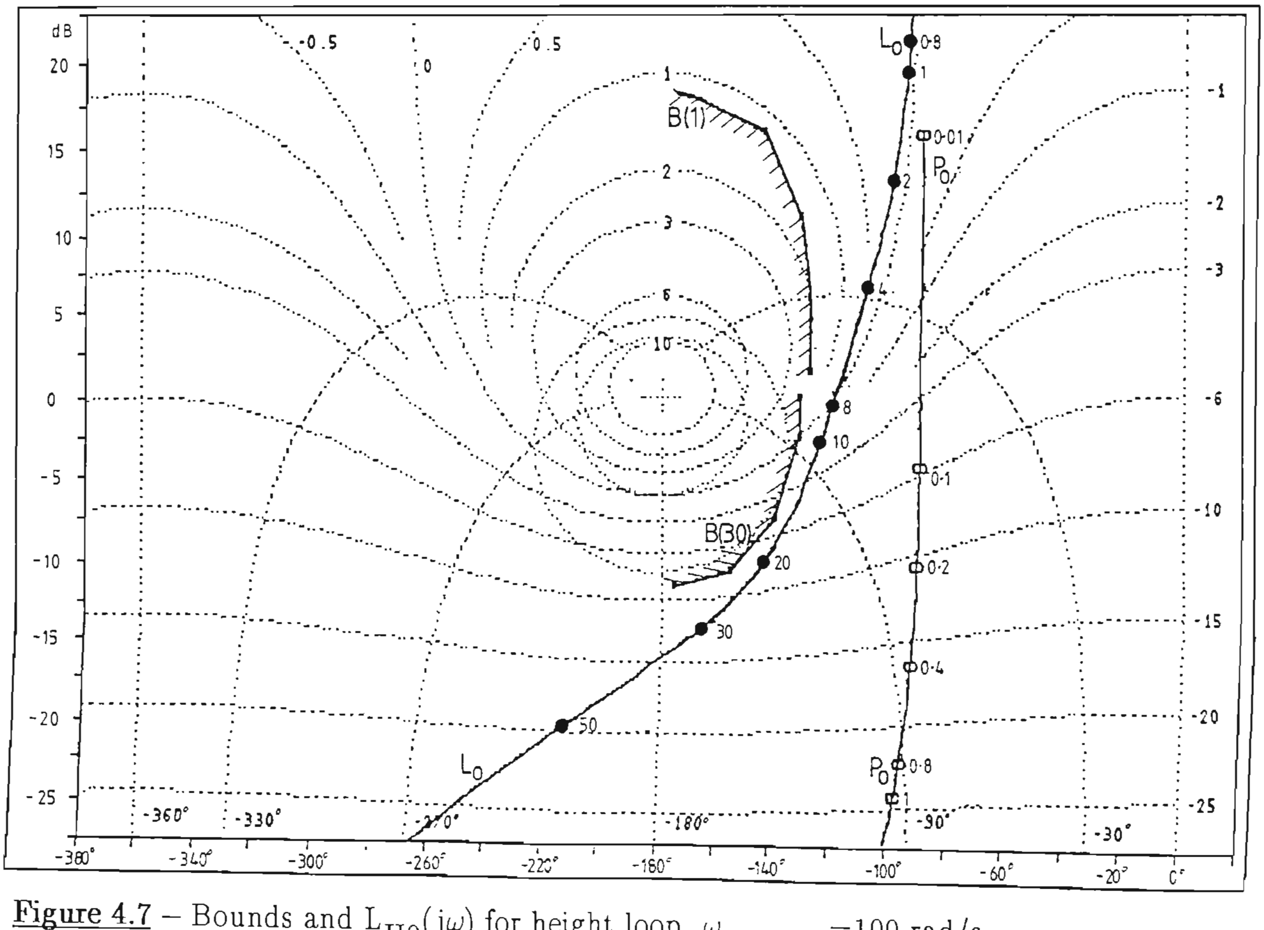


Figure 4.7 – Bounds and $L_{H0}(j\omega)$ for height loop. $\omega_{n\ motor}$=100 rad/s

4.2 Results

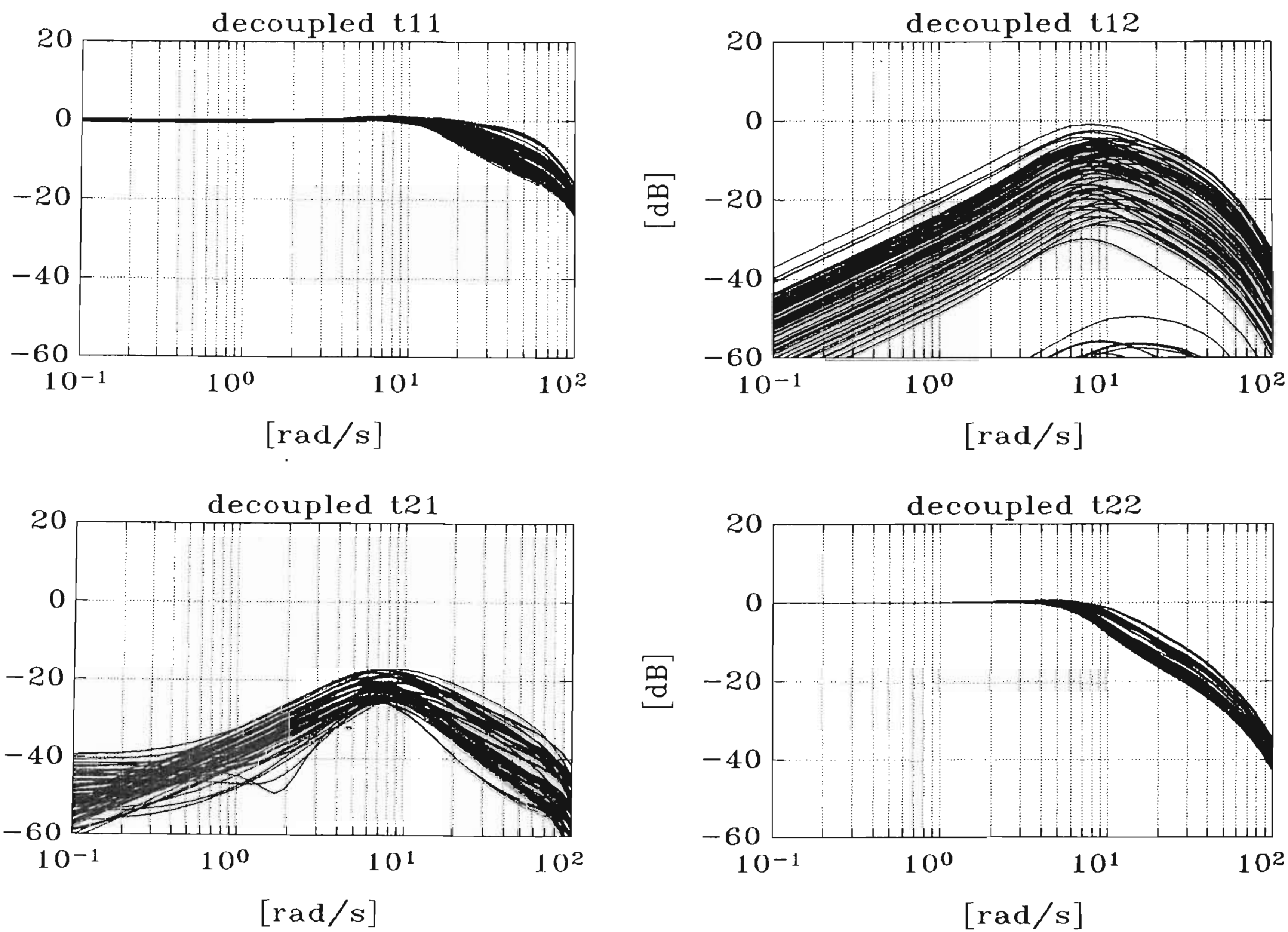
The Bode magnitude diagrams for the various transfer functions calculated at all the entries of the plant set used in the design are shown in Figures 4.8 (reference–output), 4.9 (disturbance–output) and 4.10 (disturbance– actual plant input). These Bode plots include the motor characteristics (with corner frequency ω_n=100 rad/s). These are presented as results as they were calculated after the design.

Although the controller design is decoupled for output behaviour at the nominal of a simplified plant model, it is coupled with respect to plants in the plant set, away from the nominal. Figure 4.8, showing the input–output behaviour with no pre–filter ($\underline{F}=\underline{I}$) highlights the coupling, especially in the t_{12} ($H_{ref} \rightarrow \theta$) path.

In the test data presented below all commands were simply filtered using a first order low–passing filter with a cut–off frequency of 0.7 rad/s. This aspect of the design was largely ignored in order to concentrate on the feedback controller design. Notice that the low corner frequency of the pre–filter avoids exciting the transmission paths at those frequencies where coupling appears to be a problem.

Simulated data for the flying machine with the above controller structure and design is presented in Figures 4.11 to 4.13:
4.11) Disturbance regulation when measured disturbances are added to the wing torques and drags (using disturbance data from test R22).
4.12) Response to a step change in height reference of 0.63m.
4.13) Response to a step change in pitch angle reference of 62°.

Actual flight data for the flying machine with the above controller structure and design is presented in Figures 4.14 to 4.17:
4.14) Disturbance regulation with constant height and pitch angle set–points, Test # Dec3 (c.f. Figure 4.11). Figure 4.14b shows wing rate demands for this test, obtained by differentiating the measured data numerically.
4.15) Response to a step changes in height error of 0.63m, Test # Dec4 (c.f. Figure 4.12).
4.16) Response to a step changes in pitch angle error of 62°, Test # Dec5 (c.f. Figure 4.13).
4.17) Response after the pitch angle measurement potentiometer failed and gave momentarily a false reading of –130°, Test # Dec2.



<u>Figure 4.8</u> – Bode magnitude diagrams for $\underline{T}_{Y/R}$ (No pre–filter)

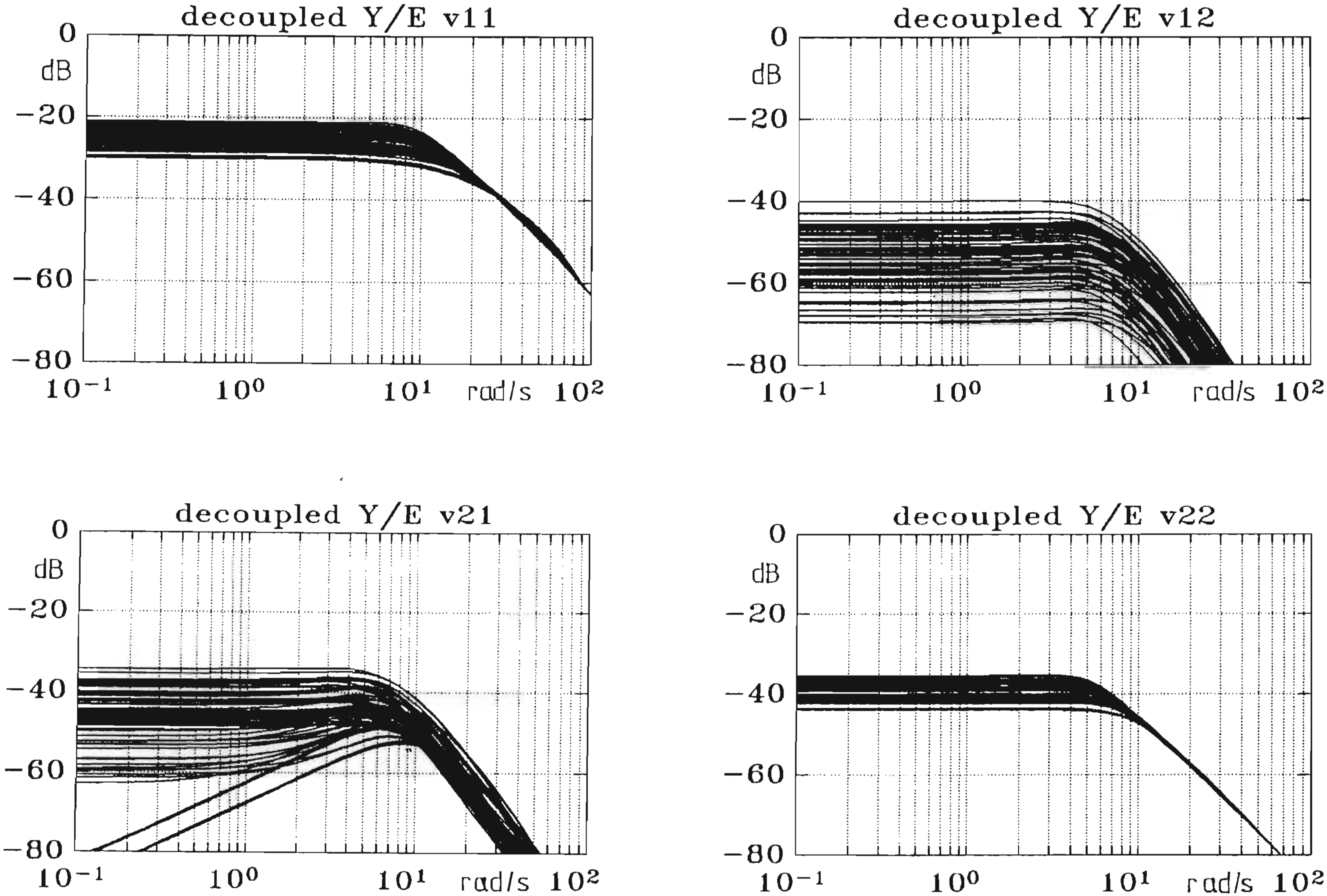


Figure 4.9 – Bode magnitude diagrams for $\underline{T}_{Y/E}$

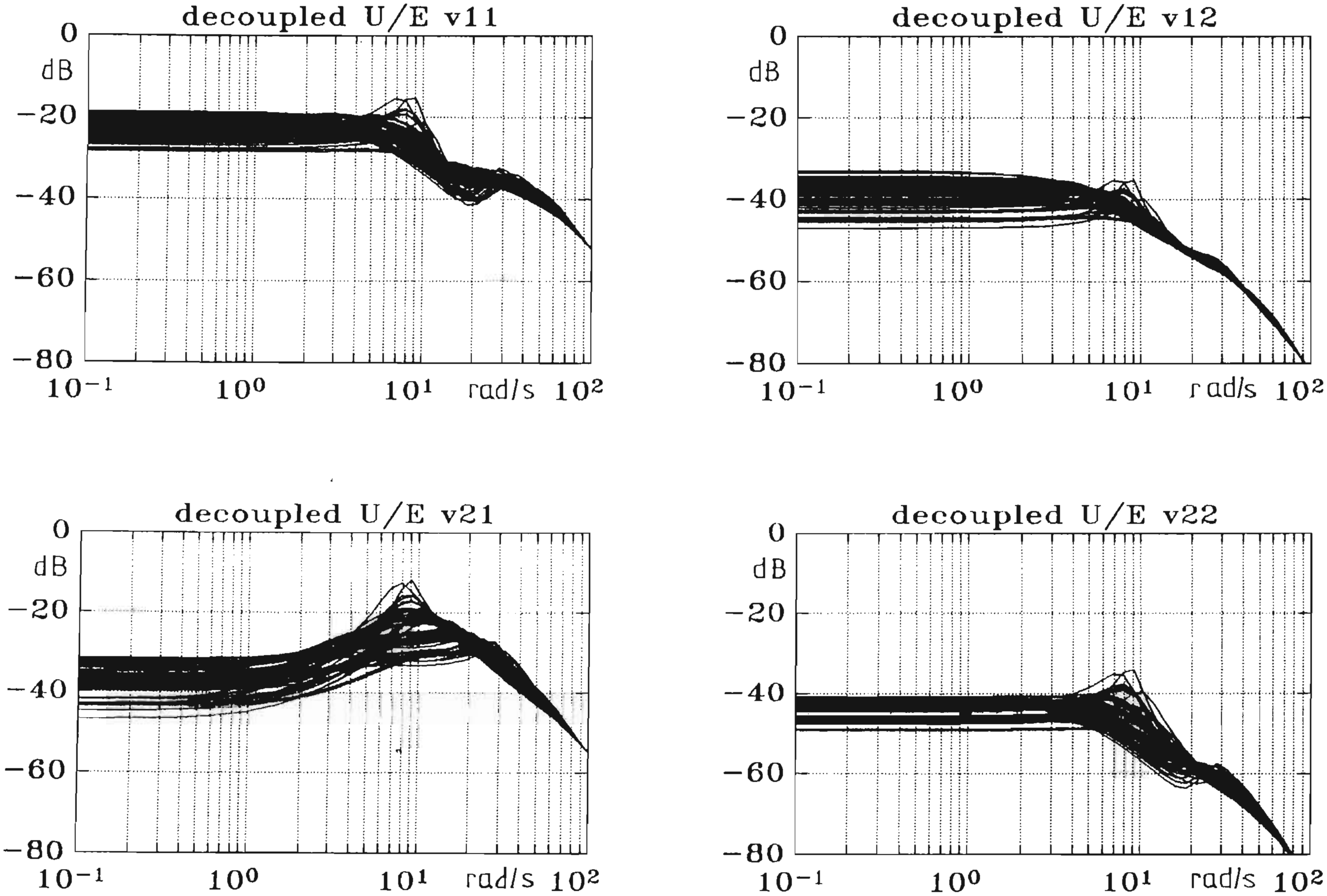


Figure 4.10 – Bode magnitude diagrams for $\underline{T}_{U/E}$

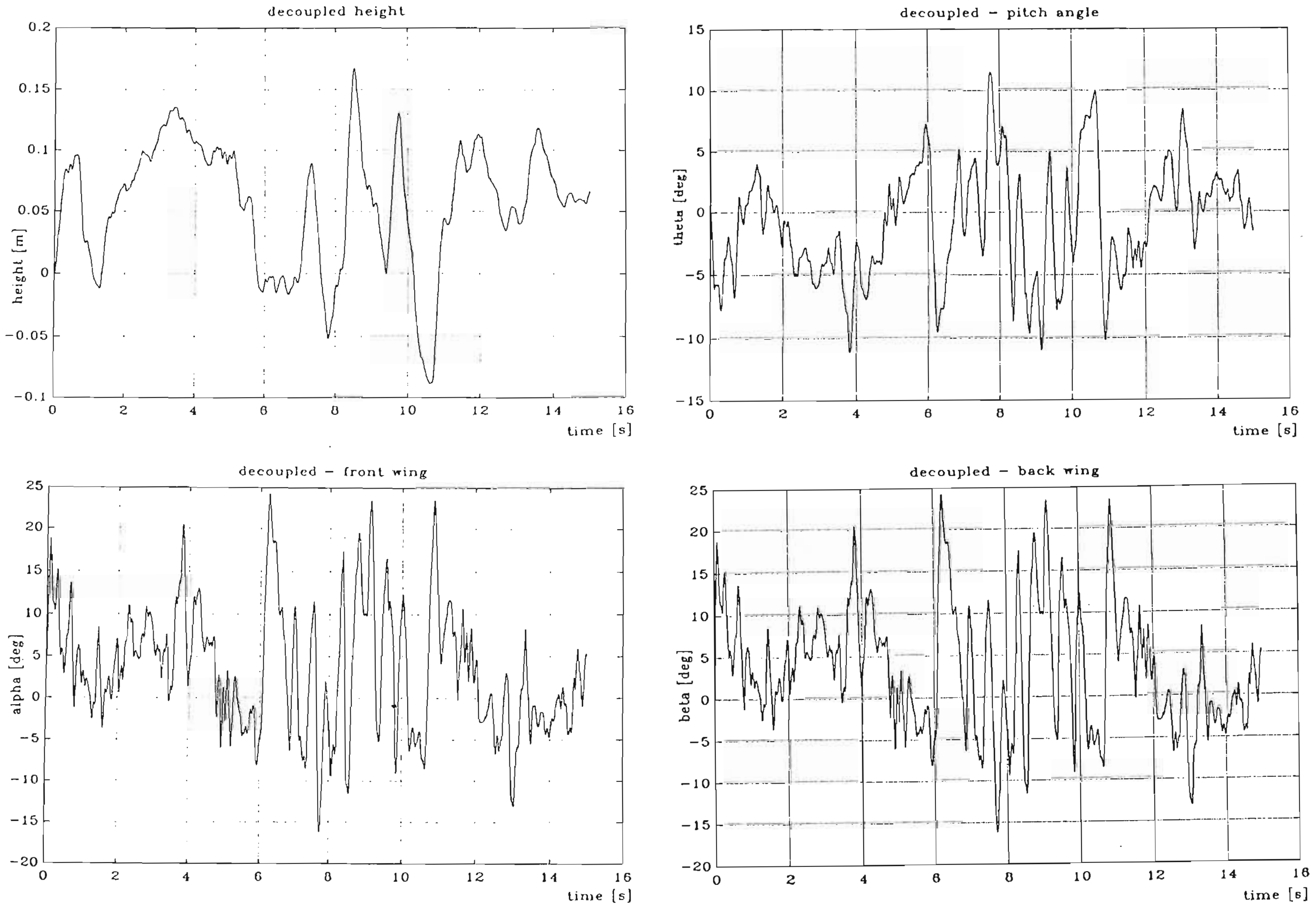


Figure 4.11 – Simulated disturbance regulation with measured disturbances from test R22

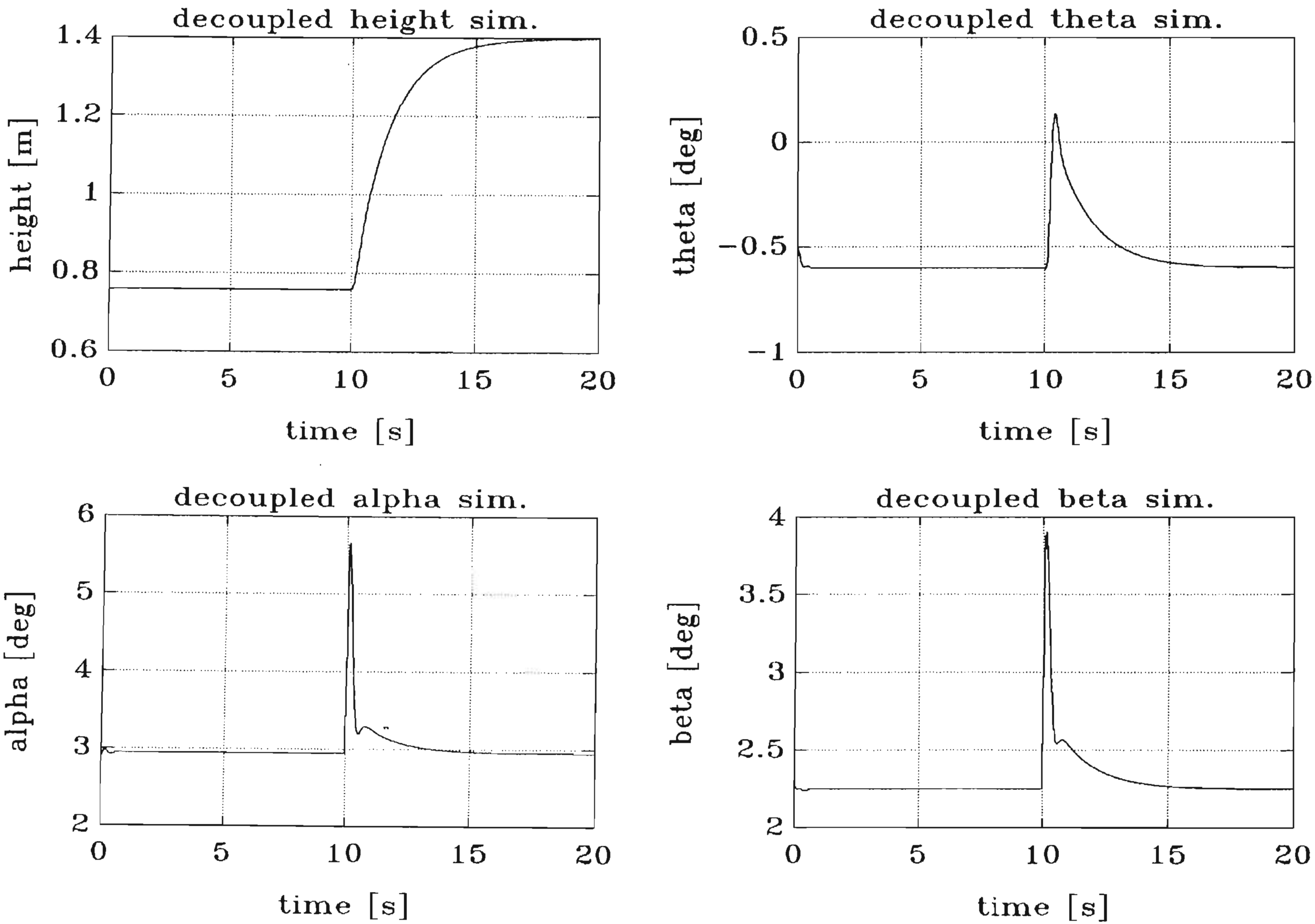


Figure 4.12a – Simulated response to a step change in height reference of 0.63m (no disturbances, with pre–filter)

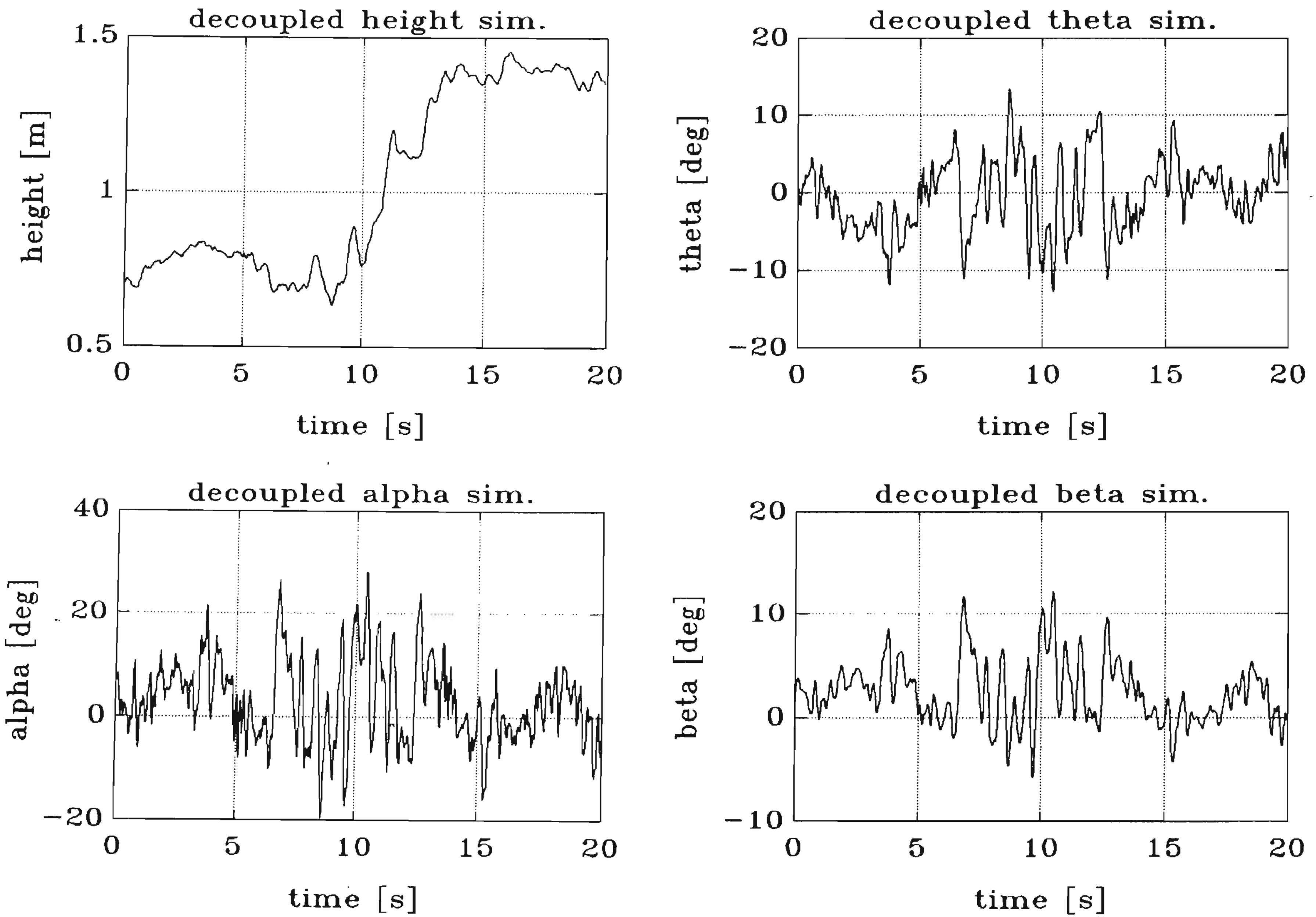


Figure 4.12b – Simulated response to a step change in height reference of 0.63m (disturbances from test R22, with pre–filter)

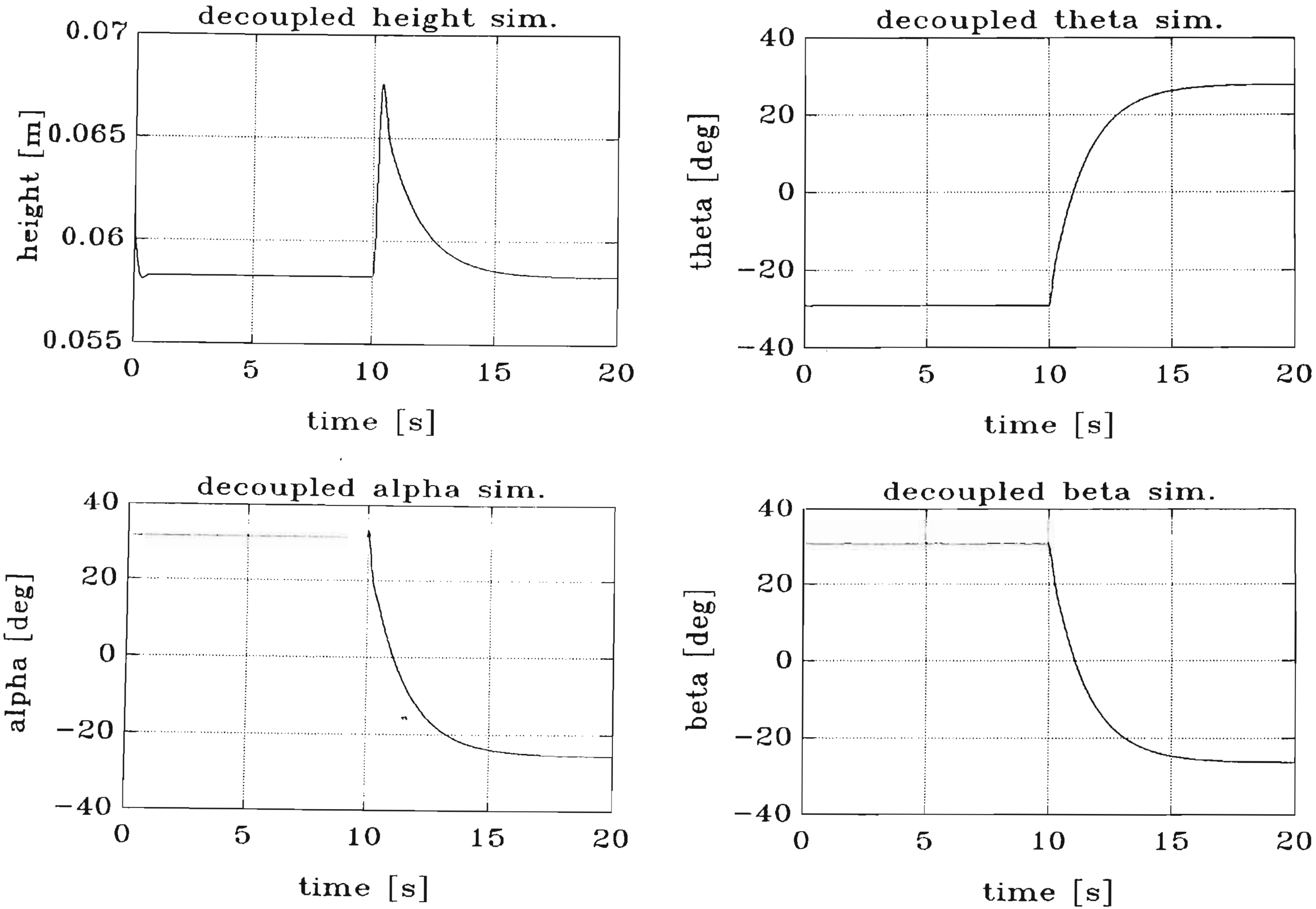


Figure 4.13a – Simulated response to a step change in pitch angle reference of 62°
(no disturbances, with pre–filter)

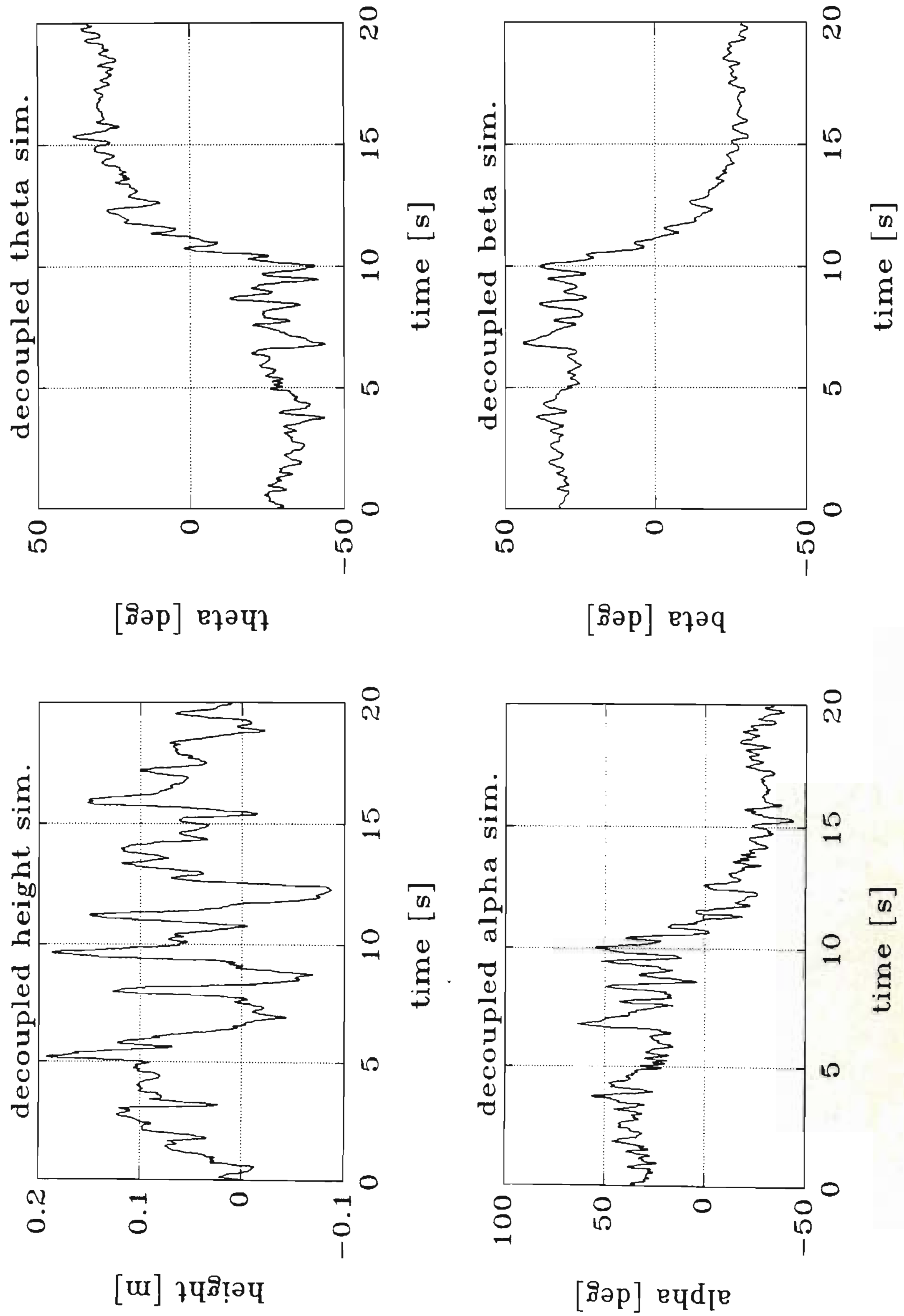


Figure 4.13b – Simulated response to a step change in pitch angle reference of 62° (disturbances from test R22, with pre–filter)

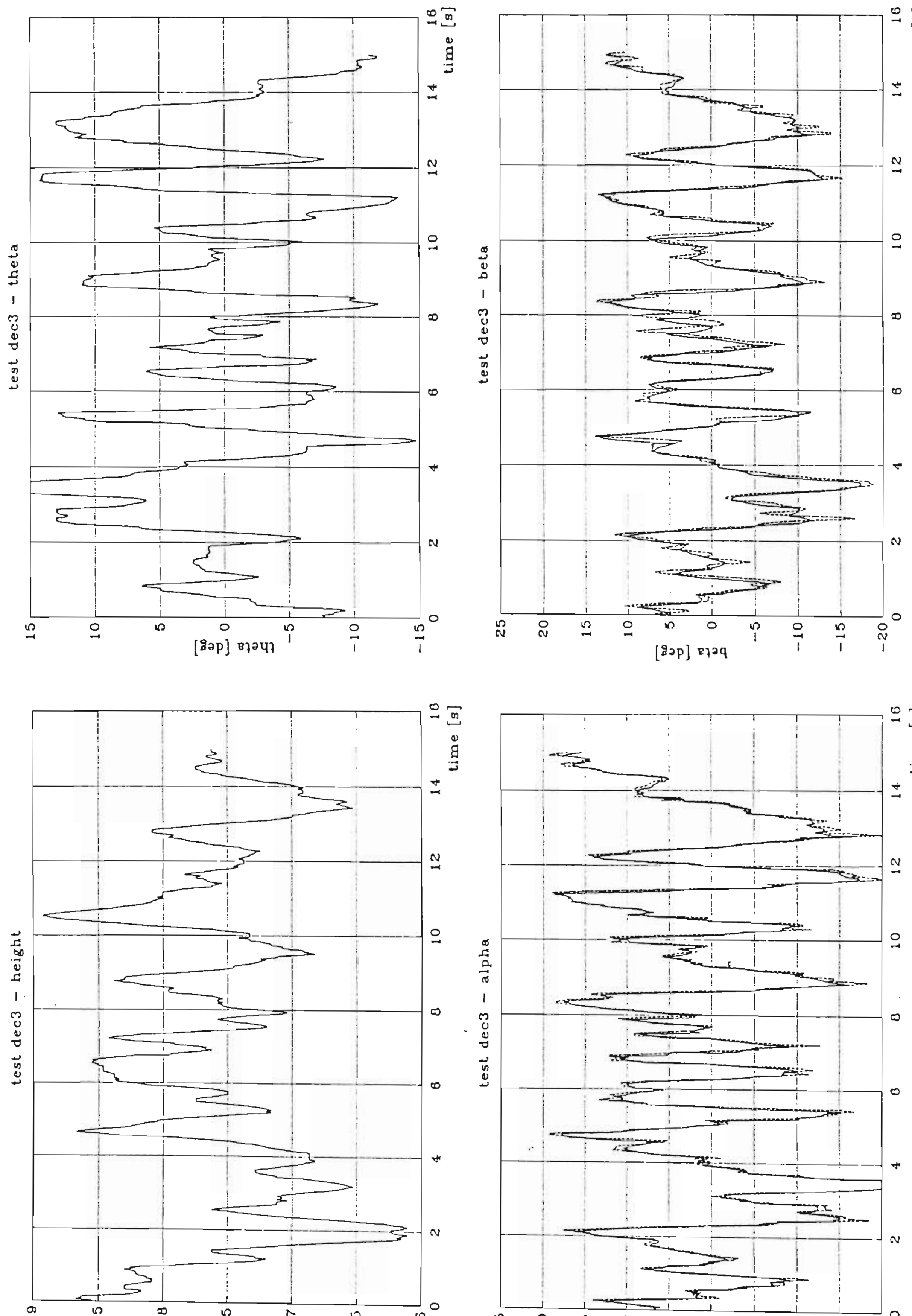


Figure 4.14 – Flight data (Test # Dec3) – disturbance regulation with constant height and pitch angle set–points (with pre–filter)

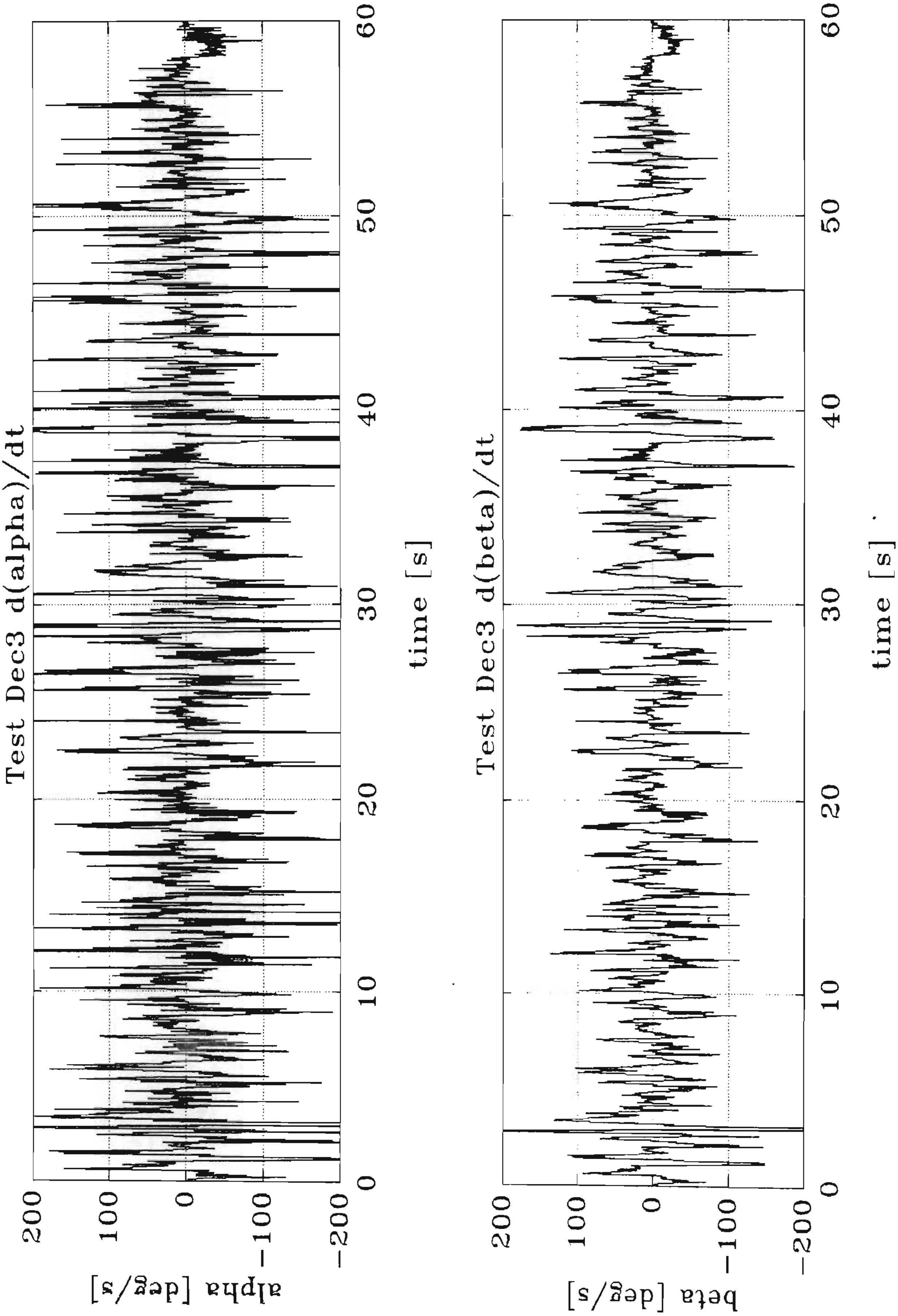


Figure 4.14b – Flight data (Test # Dec3) – wing rate demands for front and back wings

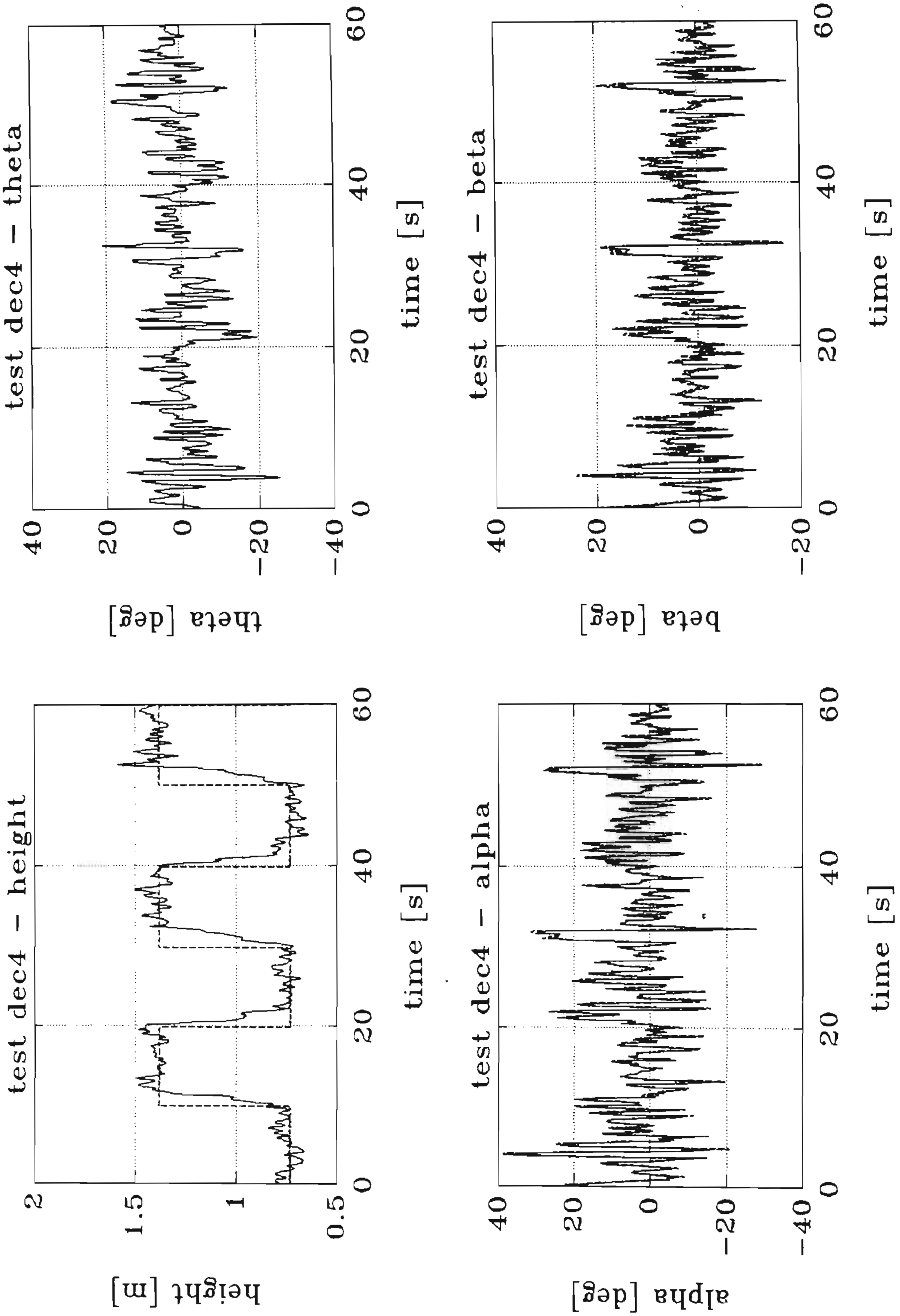


Figure 4.15a – Flight data (Test # Dec4) – response to a step changes in height error of 0.63m (with pre–filter)

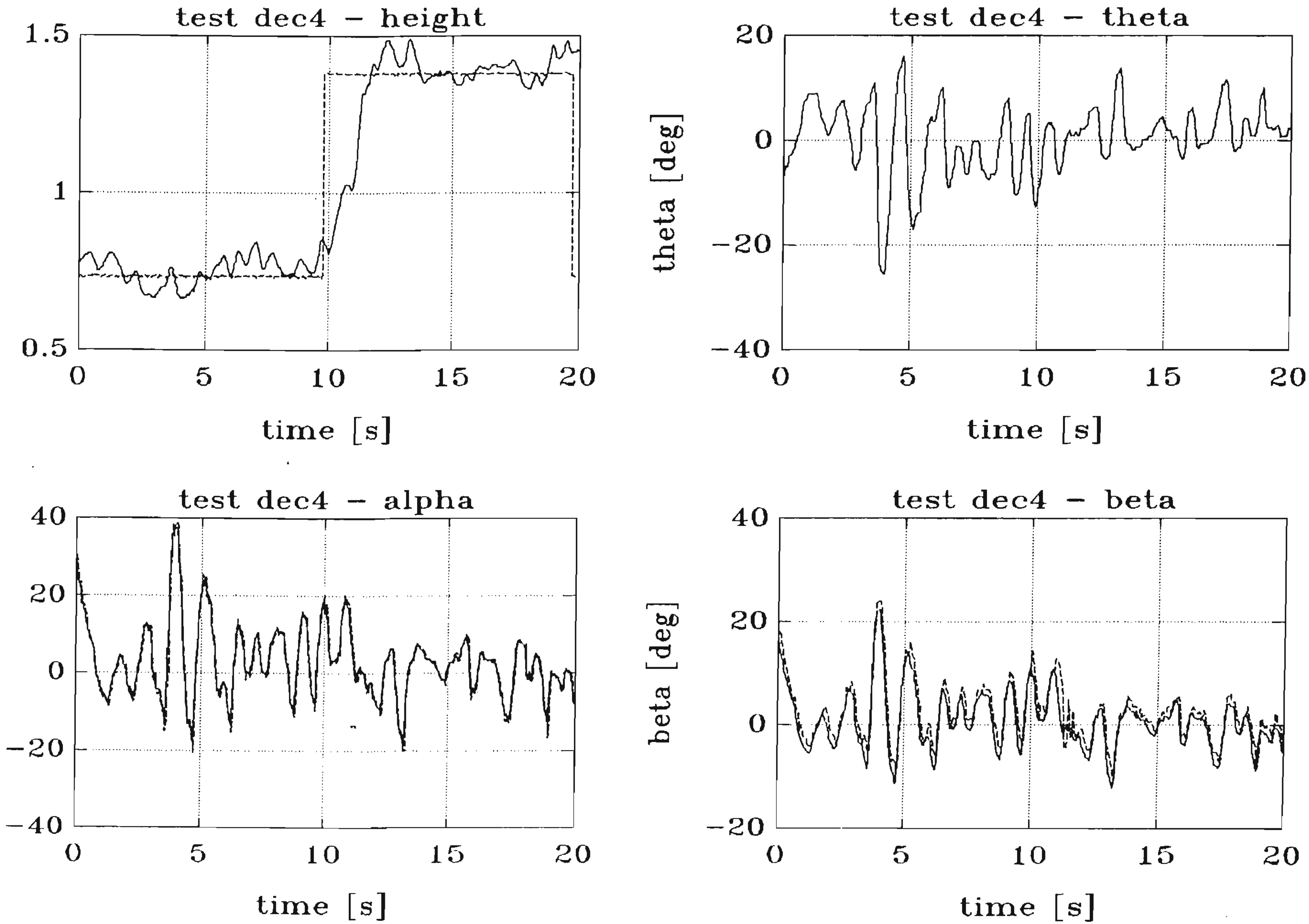


Figure 4.15b – Detail from Figure 4.15a – response to a step changes in height error of 0.63m (with pre–filter)

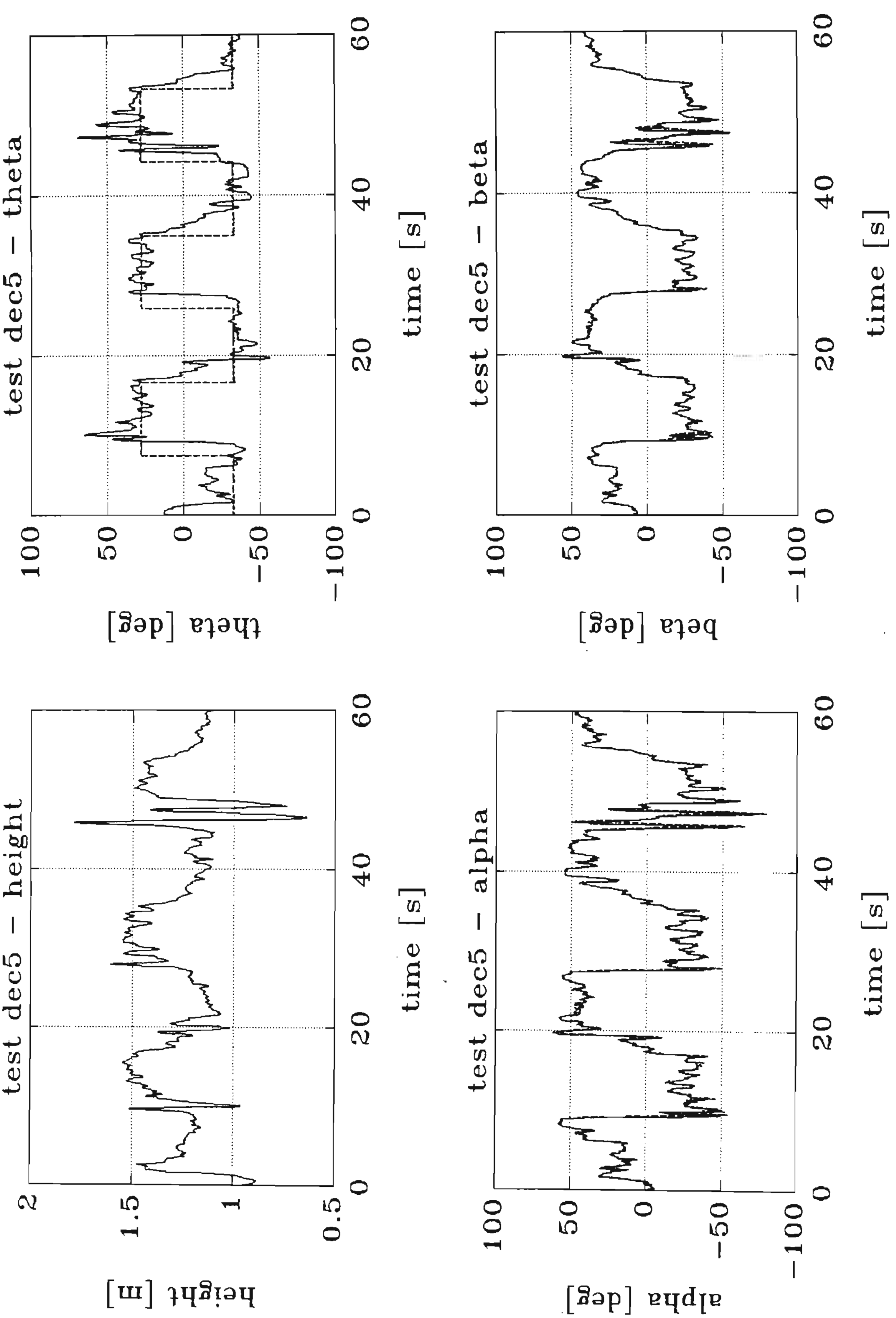


Figure 4.16a – Flight data (Test # Dec5) – response to a step changes in pitch angle error of 62° (with pre–filter)

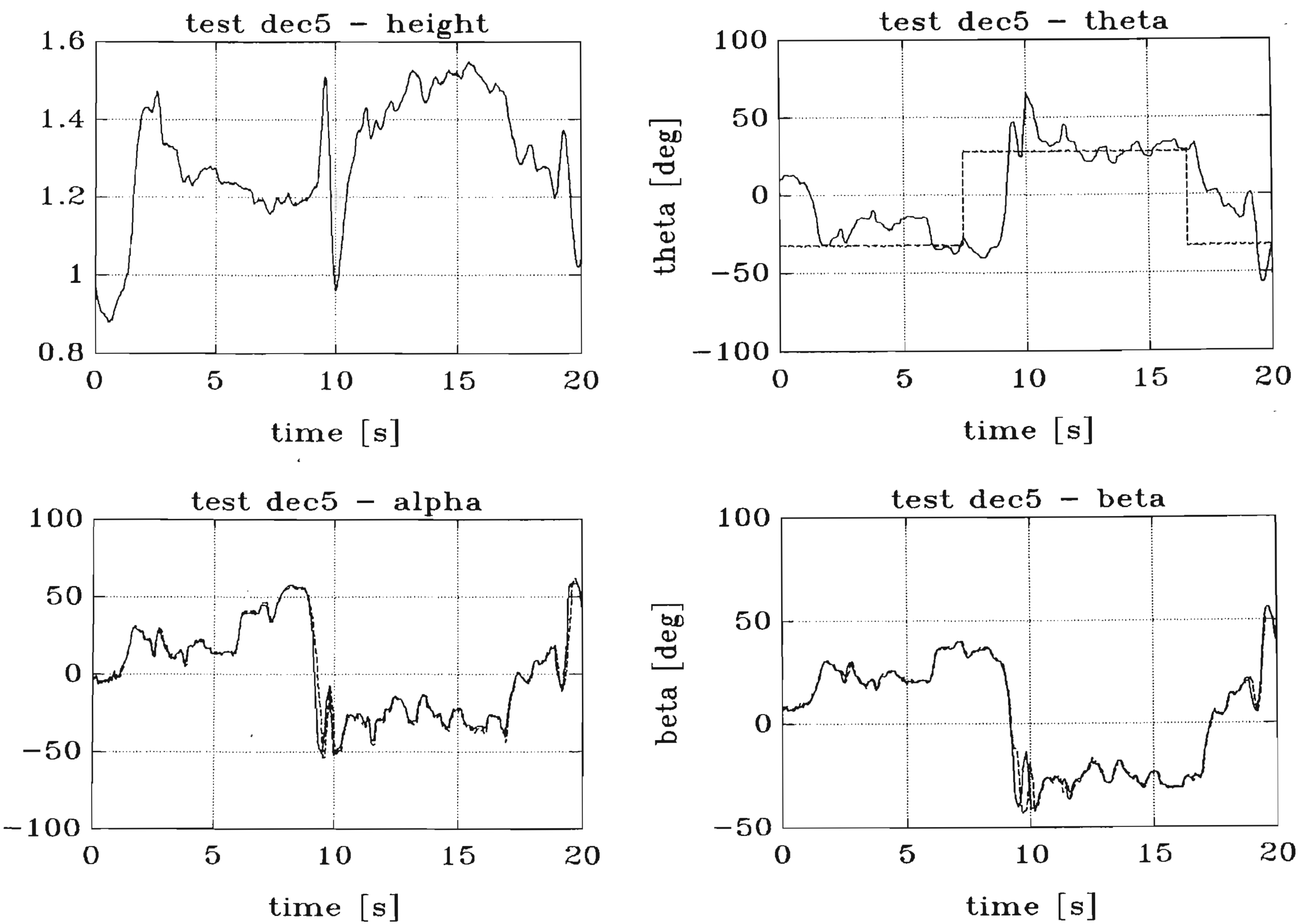


Figure 4.16b – Detail from Figure 4.15a – response to a step changes in pitch angle error of 62° (with pre-filter)

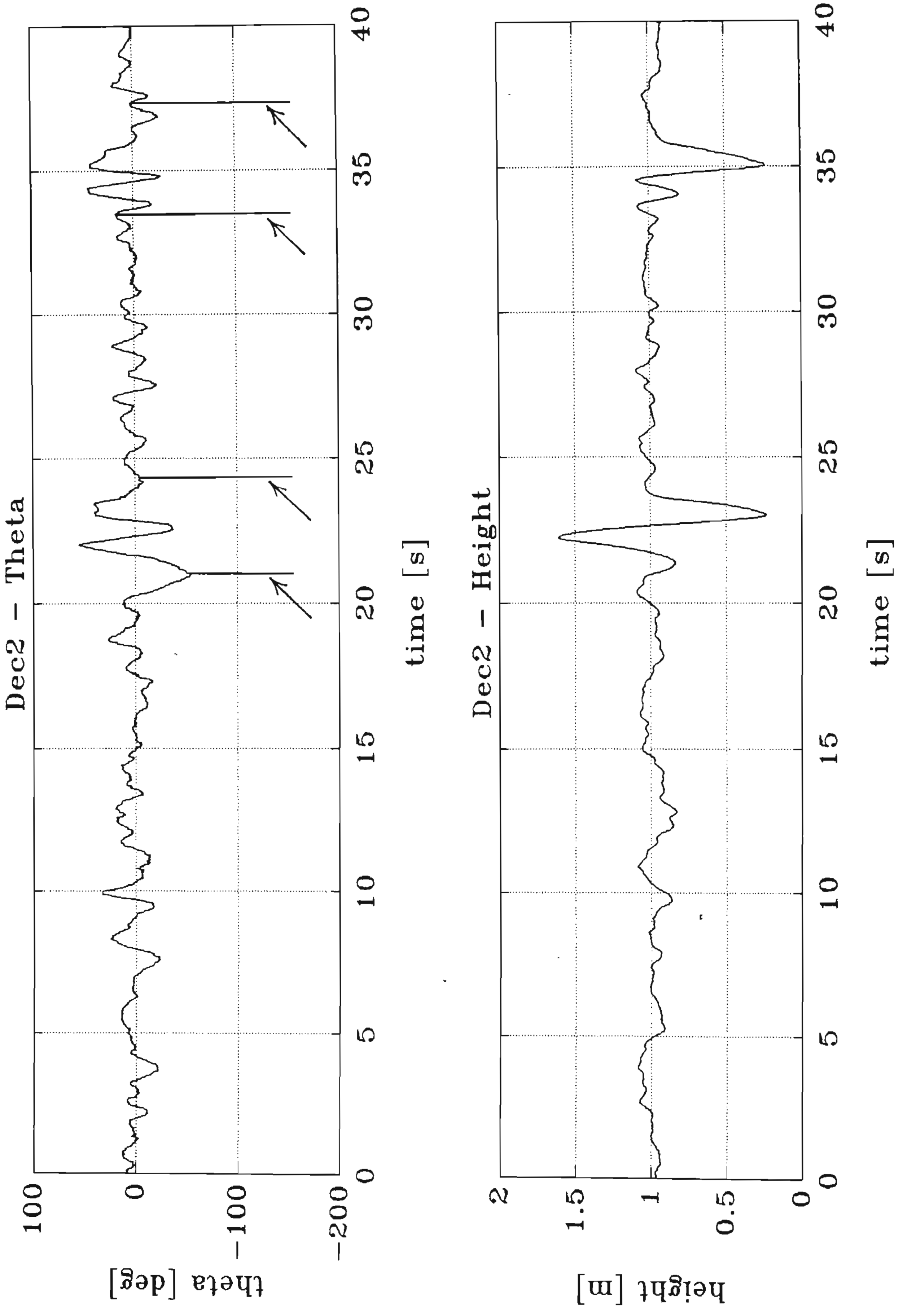


Figure 4.17 – Flight data (Test # Dec2) – response after the pitch angle measurement potentiometer failed and gave momentarily a false reading of –130°

4.3 Comments and conclusions

1) The simulation data and real test data are optically very similar, increasing confidence that the physical model, its parameters and the disturbance data reasonably reflect the true system.

2) Comparing parts (a) (no disturbance) and (b) (disturbance) of Figures 4.12 and 4.13, it is evident that a very significant portion of the control action is required for disturbance regulation.

3) The recovery of the flying machine during test # Dec2 (Figure 4.17) is remarkable. (That the failure was detected is also remarkable as the anti–alias filters on the computer based data acquisition system gently removed all traces of the glitch – this data comes from an oscilloscope.) The measurement failure is equivalent to a very large torque disturbance pulse and the recovery after the pitch angle measurement is restored can attributed to the fact that the controller structure results in the wings being turned relative to the wind direction. (No amount of feedback control, be it quantitative, robust, fuzzy, woolly, etc can cope with such a sensor failures – only feedforward control or sensor redundancy makes sense in such an eventuality.)

4) It was observed (see the flight data above) that the front wing activity was larger than that of the back wing, often being in rate and amplitude saturation (see Figure 4.14b). The third design (Chapter 6) was partially motivated by the desire to rectify this (share the actuator loads more equally).

5) The non–linearity of the plant response is seen clearly in the pitch angle step test, Dec5. Amongst other possibilities, this could be due to position dependent wind speeds or the unmodelled influence of the front wing on the back wing airflow

6) The controller described in this chapter provided the best results obtained. This is attributed to the sensible use of the controlling surfaces, derived from insight into the plant behaviour. The design achieved the specified loop bandwidths and the wings appear to have been well utilized.

CHAPTER 5 – (ANTI) DIAGONAL CONTROLLER

5.1 Introduction and design

This chapter describes the design, implementation and testing of a diagonal controller for the flying machine. The direct application of plant input design with a diagonal controller and no modification of the input or output specifications is thus illustrated. Better results were obtained using the other (non–diagonal) controller structures.

In terms of the plant model, the controller structure chosen is anti–diagonal, the pitch angle being controlled via the back wing and the height via the front wing. The motivation for this coupling of outputs to inputs is that,
a) The back wing – to – height transfer function is non minimum phase – to climb first requires turning the airframe nose up by turning the back wing down. Notice that although the plant (2,2) element in eq(2.21) (or eq(5.9) below) is non–minimum phase the simple plant model does not have any right hand plane transmission zeros. The problem of right hand plane transmission zeros has been described by Horowitz, Oldak and Yaniv (1986) with the conclusion that only one row of loop transmissions (input–output behaviour) need suffer from non–minimum phase restrictions. From their paper it is not obvious however what the cost of realizing their result is in terms of plant input levels.
b) The pitch angle loop is expected to require higher bandwidth than the height loop for better attenuation of pitch angle errors. If the pitch angle magnitude is too large, the design is not valid and there is little chance of recovery. Relatively larger height errors can be tolerated as the height dynamics does not depend on height. For the purpose of high bandwidth, the back wing – torque gain is larger ($k_{tb}>k_{tf}$), requiring less gain in the controller. The pitch angle loop is designed first.
c) Without the back wing, the front–wing to pitch–angle transfer function is unstable. Without saturation, this would not be a problem but if the front wing saturates in position, the torque it provides will have the wrong sign to correct pitch angle errors. A similar effect will occur with front wing rate saturation.

The wing actuators are modelled by $\underline{M}(s)=\mathrm{diag}\{m_{\alpha}(s),m_{\beta}(s)\}$. For small signals the actuators behave as second order systems with a corner frequency, $\omega_n\approx100$rad/s and a damping factor, $\zeta\approx0.4$. The actuators exhibited some backlash due to mechanical play. The actuator dynamics are included with the respective controllers to give,

$$\underline{G}(s) = \begin{bmatrix} m_\alpha & 0 \\ 0 & m_\beta \end{bmatrix} \begin{bmatrix} 0 & g_{12} \\ -g_{21} & 0 \end{bmatrix} = \begin{bmatrix} 0 & g_{\alpha H} m_\alpha \\ -g_{\beta\theta} m_\beta & 0 \end{bmatrix} \tag{5.1}$$

The negative sign is so that the controller, $g_{\beta\theta}$, is implemented with positive low frequency gain. This is required for stability as can be seen by by a simple argument in this problem:

a) The nominal plant is open–loop stable.

b) Small positive gains for $g_{\beta\theta}(s)$ and $g_{\alpha H}(s)$ will not change the open loop stability of the nominal (+ angle error → increase back wing angle; + height error → decrease front wing angle).

c) The stable elements of the plant set (and the plant nominal) form a connected set, and the closed loop system stability cannot change as the gain and phase of the controller is adjusted unless the magnitude goes through infinity (Nyquist point) at some frequency for at least one plant element.

d) If the unstable elements of the plant set are also considered then closed–loop stability is achieved by sufficiently large, positive low frequency gain.

e) The controller is designed with sufficiently high gain to satisfy (d), without violating (c)

Output design equations

Consider the design equations for the output,

$$(\underline{P}^{-1} + \underline{G})\, \underline{T}_{Y/E} = \underline{N}^{-1} \tag{5.2}$$

with,

$$\underline{P}^{-1} = \underline{N}^{-1}\underline{D} = \frac{1}{k} \begin{bmatrix} k_{\ell b} J s^2 + k_{\ell b}\mu_t s - k & k_{tb} m s^2 + k_{tb}\mu_\ell s \\ -(k_{\ell f} J s^2 + k_{\ell f}\mu_t s + k) & k_{tf} m s^2 + k_{tf}\mu_\ell s \end{bmatrix} = \begin{bmatrix} \frac{1}{q_{11}} & \frac{1}{q_{12}} \\ -\frac{1}{q_{21}} & \frac{1}{q_{22}} \end{bmatrix}$$

$$\underline{T}_{Y/E} = \begin{bmatrix} t_{\theta t} & t_{\theta\ell} \\ t_{Ht} & t_{H\ell} \end{bmatrix}$$

$$\underline{N}^{-1} = \frac{1}{k} \begin{bmatrix} k_{\ell b} & k_{tb} \\ -k_{\ell f} & k_{tf} \end{bmatrix}$$

and,

$$k = k_{\ell b} k_{tf} + k_{tb} k_{\ell f}$$

Writing eq(5.2) implicitly element by element,

$$t_{\theta t} = \left[\frac{1}{1+L_{\theta Y}}\right] \left[q_{21}\right] \left[k_{\ell f}/k + \left[\frac{1}{q_{22}}\right] t_{Ht}\right] \tag{5.3}$$

$$t_{\theta \ell} = \left[\frac{1}{1+L_{\theta Y}}\right] \left[q_{21}\right] \left[k_{tf}/k + \left[\frac{1}{q_{22}}\right] t_{H\ell}\right] \tag{5.4}$$

$$t_{Ht} = \left[\frac{1}{1+L_{HY}}\right] \left[q_{12}\right] \left[k_{\ell b}/k + \left[\frac{1}{q_{11}}\right] t_{\theta t}\right] \tag{5.5}$$

$$t_{H\ell} = \left[\frac{1}{1+L_{HY}}\right] \left[q_{12}\right] \left[k_{tb}/k + \left[\frac{1}{q_{11}}\right] t_{\theta \ell}\right] \tag{5.6}$$

with,

$L_{\theta Y} = q_{21}\, g_{\beta\theta}\, m_\beta$ – **output** pitch angle loop transfer function
$L_{HY} = q_{12}\, g_{\alpha H}\, m_\alpha$ – **output** height loop transfer function

After applying the Schwartz inequality and substituting the bound, $\underline{A}$ (from eq(3.23)), for elements of $\underline{T}_{YE}$ appearing on the right hand side of the equations (eq(3.25)), for $g_{\beta\theta}(s)$, the design equations are,

$$|t_{\theta t}| \le \left|\frac{1}{1+L_{\theta Y}}\right| \left|q_{21}\right| \left[k_{\ell f}/k + \left|\frac{1}{q_{22}}\right| a_{Ht}\right] \le a_{\theta t} \tag{5.7}$$

$$|t_{\theta \ell}| \le \left|\frac{1}{1+L_{\theta Y}}\right| \left|q_{21}\right| \left[k_{tf}/k + \left|\frac{1}{q_{22}}\right| a_{H\ell}\right] \le a_{\theta \ell} \tag{5.8}$$

<u>Input design equations</u>
For the input, the specifications are,

$$(\underline{G}^{-1} + \underline{P})\, \underline{V}_{UE} = -\, \underline{P}^* \tag{5.9}$$

with,

$$P = \underline{D}^{-1}\underline{N} = \begin{bmatrix} \dfrac{k_{tf}}{Js^2+\mu_t s+k_t} & -\dfrac{k_{tb}}{Js^2+\mu_t s+k_t} \\ \dfrac{k_{\ell f}Js^2+k_{\ell f}\mu_t s+k}{(Js^2+\mu_t s+k_t)(ms^2+\mu_\ell s)} & \dfrac{k_{\ell b}Js^2+k_{\ell b}\mu_t s-k}{(Js^2+\mu_t s+k_t)(ms^2+\mu_\ell s)} \end{bmatrix}$$

and

$$\underline{P}^* = \underline{D}^{-1} = \begin{bmatrix} \dfrac{-1}{Js^2+\mu_t s+k_t} & 0 \\ \dfrac{-\,k_\ell}{(Js^2+\mu_t s+k_t)(ms^2+\mu_\ell s)} & \dfrac{-\,1}{ms^2+\mu_\ell s} \end{bmatrix}$$

Writing eq(5.9) implicitly,

$$v_{\beta t} = \frac{L_{\theta U}}{1+L_{\theta U}} \left[\frac{k_{tf}}{k_{tb}} v_{\alpha t} + \frac{1}{k_{tb}}\right] \tag{5.10}$$

$$v_{\beta\ell} = \frac{L_{\theta U}}{1+L_{\theta U}}\left[\frac{k_{tf}}{k_{tb}}\, v_{\alpha\ell}\right] \tag{5.11}$$

$$v_{\alpha t} = -\frac{L_{HU}}{1+L_{HU}}\left[\frac{k_{\ell b}Js^2+k_{\ell b}\mu_t s-k}{k_{\ell f}Js^2+k_{\ell f}\mu_t s+k}\, v_{\beta t} + \frac{k_\ell}{k_{\ell f}Js^2+k_{\ell f}\mu_t s+k}\right]$$

$$= -\frac{L_{HU}}{1+L_{HU}}\left[K_1(s)\, v_{\beta t} + K_2(s)\right] \tag{5.12}$$

$$v_{\alpha\ell} = -\frac{L_{HU}}{1+L_{HU}}\left[\frac{k_{\ell b}Js^2+k_{\ell b}\mu_t s-k}{k_{\ell f}Js^2+k_{\ell f}\mu_t s+k}\, v_{\beta\ell} + \frac{Js^2+\mu_t s+k_t}{k_{\ell f}Js^2+k_{\ell f}\mu_t s+k}\right]$$

$$= -\frac{L_{HU}}{1+L_{HU}}\left[K_1(s)\, v_{\beta\ell} + K_3(s)\right] \tag{5.13}$$

with,

$L_{\theta U} = p_{12}\, g_{\beta\theta}\, m_\beta$ – **input** ∠attack loop transfer function

$L_{HU} = p_{21}\, g_{\alpha H}\, m_\alpha$ – **input** height loop transfer function

After applying the Schwartz inequality and substituting the bound, $\underline{B}$, (from eq(3.24)) for elements of $\underline{V}_{UE}$ appearing on the right hand side of the equations (equations (3.26), (3.27) and(3.28)), for $g_{\beta\theta}(s)$, the design equations are,

$$|v_{\beta t}| \leq \left|\frac{L_{\theta U}}{1+L_{\theta U}}\right| \left[\frac{1}{k_{tb}} + \frac{k_{tf}}{k_{tb}}\, b_{\alpha t}\right] \leq b_{\beta t} \tag{5.14}$$

$$|v_{\beta\ell}| \leq \left|\frac{L_{\theta U}}{1+L_{\theta U}}\right| \left[\frac{k_{tf}}{k_{tb}}\, b_{\alpha\ell}\right] \leq b_{\beta\ell} \tag{5.15}$$

To get an idea of what the specifications should be, solve eq(5.2) for the output specification at steady state,

$$\underline{T}_{Y/E}(j0) = \begin{bmatrix} \dfrac{k_{\ell f}}{k\,(1+g_\beta)} & \dfrac{-k_{tf}}{k\,(1+g_\beta)} \\[2ex] \dfrac{k_\ell + k_{\ell b}\, g_\beta}{k\, g_\alpha\, (1+g_\beta)} & \dfrac{k_t + k_{tb}\, g_\beta}{k\, g_\alpha\, (1+g_\beta)} \end{bmatrix} \tag{5.16}$$

with,

$g_\alpha = g_\alpha(j0)$, $g_\beta = g_\beta(j0)$

The steady state gains of the controllers can be expected to be not more than 1.0 (0dB) : A 10° pitch angle error should result in a back wing position of no more than 20° with respect to the ground (at which angle the wing starts to stall) and a height error of 0.1m should result in a front wing deflection of no more than 5°.

Larger steady state gains could result in position saturation of the wings. In this case at nominal plant values (Table 2.1),

$$\underline{T}_{EY0}(j0) = \begin{bmatrix} 4.1 & -1.8 \\ 29. & 4.2 \end{bmatrix} \times 10^{-3} \tag{5.17}$$

with $g_\alpha(j0) = g_\beta(j0) = 1$

The nominal was chosen at relatively high $k_{..}$ values. For plants with lower steady state gains (by a factor of about 3), $T_{EY}(j0)$ will have 10 dB larger magnitude. This means that the plant disturbance rejection is better with high wind speed (plant gain), which makes physical sense. The sting in the tail is that the disturbances can be expected to increase in magnitude as the wind speed increases. This was measured qualitatively in Section 2.3.2

The controller will be designed to be strictly proper (having an excess of poles over zeros in each channel) and will therefore be open circuit at high enough frequencies. This means that

$$\lim_{s \to j\infty} \underline{T}_{EY}(s) = \lim_{s \to j\infty} \underline{P}(s) = \lim_{s \to j\infty} \frac{1}{s^2} \begin{bmatrix} \frac{k_{tf}}{J} & \frac{-k_{tb}}{J} \\ \frac{k_{\ell f}}{m} & \frac{k_{\ell b}}{m} \end{bmatrix} \tag{5.18}$$

At nominal values this is,

$$\lim_{s \to j\infty} \underline{T}_{EY0}(s) = \lim_{s \to j\infty} \frac{1}{s^2} \begin{bmatrix} 185 & -313 \\ 27 & 80 \end{bmatrix} \tag{5.19}$$

Since at low frequencies, the q_{ij} are essentially low–passing, $q_{ij}/(1+L_{ij}) \approx 1/g_{ij}$ in equations (5.3) to (5.6) and t_{ij} are therefore expected to be flat at low frequencies. For specifications on $|\underline{T}_{EY}(j\omega)|$ the low–pass characteristic with asymptotes, $|\underline{T}_{EY0}(j0)|$ +10dB (to allow for low gain plants and some overshoot) and $|\underline{T}_{EY0}(j\infty)|$ will be used. If the resulting disturbance rejection is unacceptable, the gain of the controllers will be increased to reduce $|\underline{T}_{EY}|$ over some (low) frequency range. The high frequency asymptote is fixed by the plant structure.

From eq(5.9), examine the steady–state transfer of disturbance to the plant input for the nominal plant

$$\underline{V}_{UE0}(j0) = \frac{-1}{k(1+m_\beta g_{\beta\theta})} \begin{bmatrix} k_{\ell b} m_\beta g_{\beta\theta} + k_\ell & k_{tb} m_\beta g_{\beta\theta} + k_t \\ -k_{\ell f} m_\beta g_{\beta\theta} & k_{tf} m_\beta g_{\beta\theta} \end{bmatrix} \tag{5.20}$$

(The infinite steady–state plant gain in the height loop accounts for the absence of $g_{\alpha H}(j0)$ above.)

If $g_{\beta\theta}(j0) = 1$ as postulated before,

$$\underline{Y}_{UE0}(j0) = \begin{bmatrix} -28.9 & -4.2 \\ 4.1 & -1.8 \end{bmatrix} \times 10^{-3} \tag{5.21}$$

and lower (global) plant gain would reduce the above.

Consider eq(3.17) and eq(3.20) for the position and rate at the input as a result of disturbances. Allow each disturbance input (torque and lift) to use 1/2 the available input (power) and let $\zeta = 0.4$ (allowing about 3dB overshoot in the disturbance to input transfer functions),

$$\sigma_u^2/2 = 0.25/2 \approx \omega_n \frac{\eta}{k^2} \frac{\pi}{4\zeta} \tag{5.22}$$

$$\sigma_{\dot{u}}^2/2 = 4/2 \approx \omega_n^3 \frac{\eta}{k^2} \frac{\pi}{4\zeta} \tag{5.23}$$

(Table 2.1, eq(2.32)) and $\eta_t = 0.14$, $\eta_\ell = 1.9$ (Table 2.1, eq(2.30) and eq(2.29) respectively)

As was the case in the first design (Chapter 4), the bandwidth is limited by the input rate specification. The cut–off frequencies determined by position are,

$$\begin{bmatrix} \omega_{n\ \alpha t} & \omega_{n\ \alpha\ell} \\ \omega_{n\ \beta t} & \omega_{n\ \beta\ell} \end{bmatrix} = \begin{bmatrix} 5.4\times10^2 & 1.9\times10^3 \\ 2.7\times10^4 & 1.0\times10^4 \end{bmatrix} \text{rad/s} \tag{5.24}$$

and by rate are,

$$\begin{bmatrix} \omega_{n\ \alpha t} & \omega_{n\ \alpha\ell} \\ \omega_{n\ \beta t} & \omega_{n\ \beta\ell} \end{bmatrix} = \begin{bmatrix} 21 & 31 \\ 76 & 55 \end{bmatrix} \text{rad/s} \tag{5.25}$$

From considerations of the wing actuator (plant input) rate limits, the input loop gain cross–over frequencies are therefore,

$$|L_{HU}(j21)| \leq 0\text{dB} \qquad \text{(front wing)} \tag{5.26}$$

and

$$|L_{\theta U}(j55)| \leq 0\text{dB} \qquad \text{(back wing)} \tag{5.27}$$

Another consideration in the input specification is the following. Assume that $v_{\beta t}$ and $v_{\beta\ell}$ can be designed to be low passing transfer functions without excessive overshoot as desired. It is not certain whether or not $v_{\alpha t}$ and $v_{\alpha\ell}$ can also be designed to be low–passing as the transfer functions, $K_1(s)$, $K_2(s)$, and $K_3(s)$ from eq(5.12) and eq(5.13) have large overshoot around 30 rad/s. (The (normalized) transfer functions, $|K_1(s)|$, $|\frac{k}{k_\ell} K_2(s)|$, and $|\frac{k}{k_t} K_3(s)|$ are shown in Figure 5.1 for the nominal plant.) Unless $\frac{L_{HU}}{1+L_{HU}}$ starts rolling off before about 10 rad/s, the influence of the terms K_1, K_2 and K_3 will cause design difficulty. (Careful attention to possible advantages which might arise from the phase relationship between the terms could highlight possible cancellations although at high frequency where unstructured uncertainty and unmodelled dynamics are a problem, subtraction is not a reliable method for reducing gain).

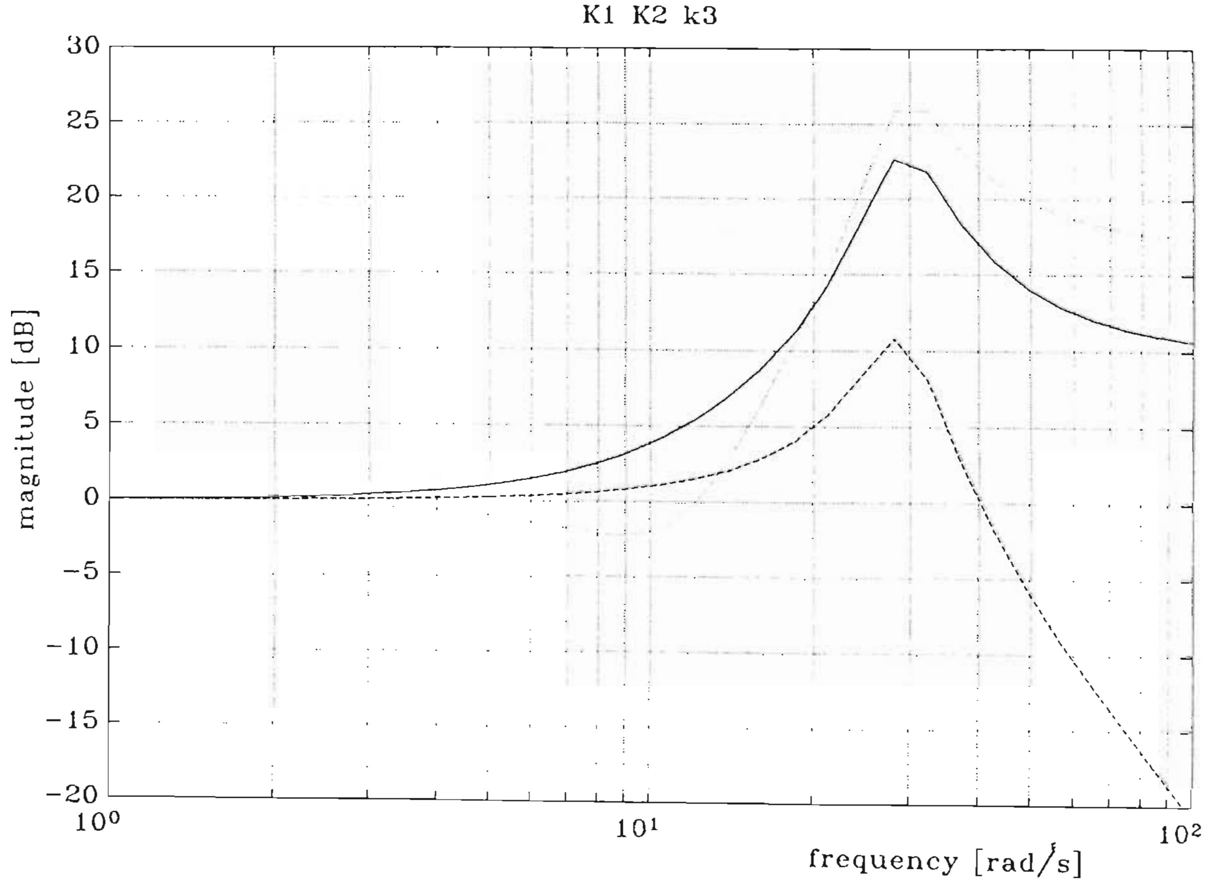


Figure 5.1 – Normalized K_1 (solid), K_2 (dashed) and K_3 (dotted) from eq(5.12) and eq(5.13)

Summary of Specifications

1) It must fly.

2) The actuators must not wear out too quickly and actuator limits must be avoided.

3) The output loops must have the highest bandwidth possible so that $\left|\frac{1}{1+L_{.Y}}\right|$ << 0dB until q_{ij} << 0dB

4) To avoid high amplification of disturbances at the plant inputs and outputs, the simple specifications,

$\left|\frac{1}{1+L_{\cdot}}\right| \leq 6\text{dB}$ and $\left|\frac{L_{\cdot}}{1+L_{\cdot}}\right| \leq 3\text{dB}$ were chosen for all loops.

5) From the wing actuator (plant input) rate limits, the input loop gain cross–over frequencies are, $|L_{HU}(j20)| \leq 0\text{dB}$ and $|L_{\theta U}(j55)| \leq 0\text{dB}$.

6) $\frac{L_{HU}}{1+L_{HU}}$ starts rolling off before $10'$rad/s.

Clearly the input and output specifications necessitate a compromise.

Design of pitch angle loop

The first loop designed was the pitch angle. Bounds for the $L_{\theta\cdot}$ and some templates are shown in Figure 5.2 (input) and Figure 5.3 (output). The nominal plant was chosen as $p_{0\ 12}$ for input and output loops. As a result the the nominal for the output loop lies outside the template and may itself violate the output loop boundaries. This notion of designing the loop transfer function for a (fictitious) nominal plant, lying outside the (real) plant set was introduced in Section 3.1 but needs justification for the benefit of those less familiar with quantitative feedback design. The Nichols chart provides a graphical mapping from L to L/(1+L), irrespective of what L is – it can be a set, a point or both. If P_0 is displaced from an element of the plant set, P_i by D and the controller, G satisfies some specification on L/(1+L) for $L=P_iG$, the same controller causes the original specifications to be satisfied.

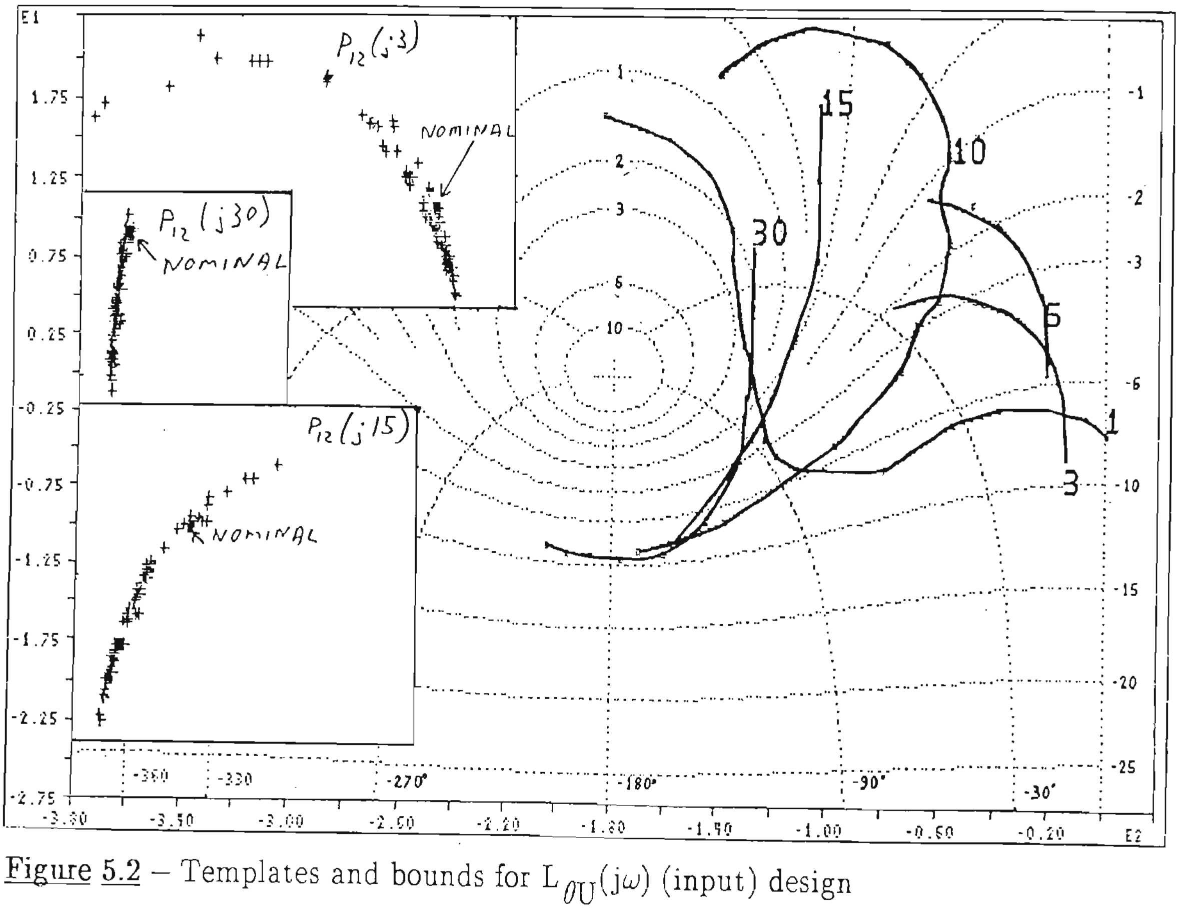
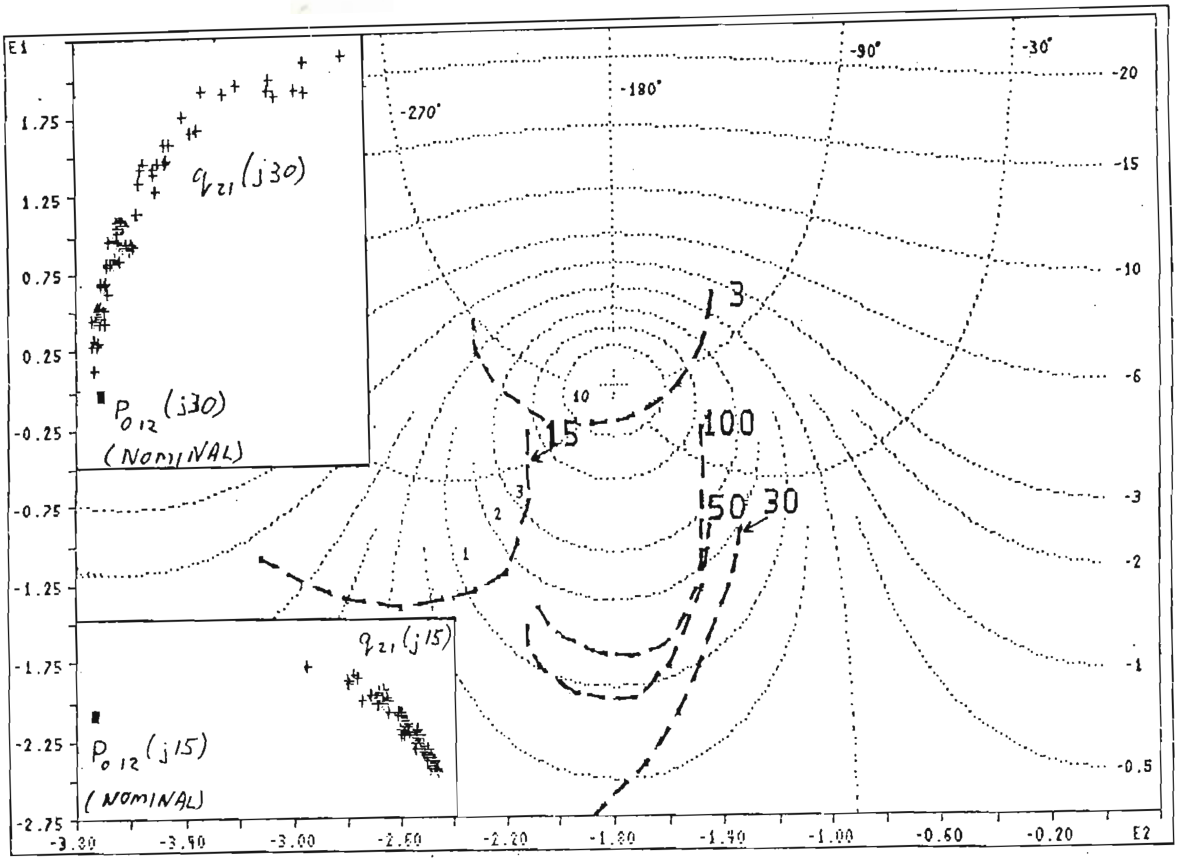


Figure 5.2 – Templates and bounds for $L_{\theta U}(j\omega)$ (input) design



<u>Figure 5.3</u> – Templates and bounds for $L_{\theta Y}(j\omega)$ (output) design

The controller designed to satisfy the pitch angle loop specifications is,

$$g_{\beta\theta}(s) = \frac{(s/1.5+1)(s/30+1)}{(s/0.65+1)((s/150)^2+(s/150)+1)} \tag{5.28}$$

The nominal input pitch angle loop transfer function resulting from this controller is shown in Figure 5.4

<u>Design of height loop</u>

The pitch angle loop was closed and templates constructed for the design of the height loop (following eq(3.35) and eq(3.36)). The templates for this loop are those for an exact SISO design. The nominal plant selected for the design execution was $p_{0\ 21} \neq p'_{0\ 2}$ of eq(3.36), which accounts for the nominal again being outside the template. The templates show that the actual plant has significantly more phase lag than the nominal and this phase lag makes achievement of any sort of bandwidth in the height loop impossible with the (anti) diagonal controller and pitch angle controller fixed by eq(5.28) above. Figure 5.5 shows the bounds on the nominal input height loop transfer function and an $L_{H0}(j\omega)$ (which violates the bounds) corresponding to the controller,

$$g_{\alpha H} = \frac{0.5\ (s/1+1)(s/3+1)}{(s/0.21+1)((s/100)^2+2(s/100)+1)} \tag{5.29}$$

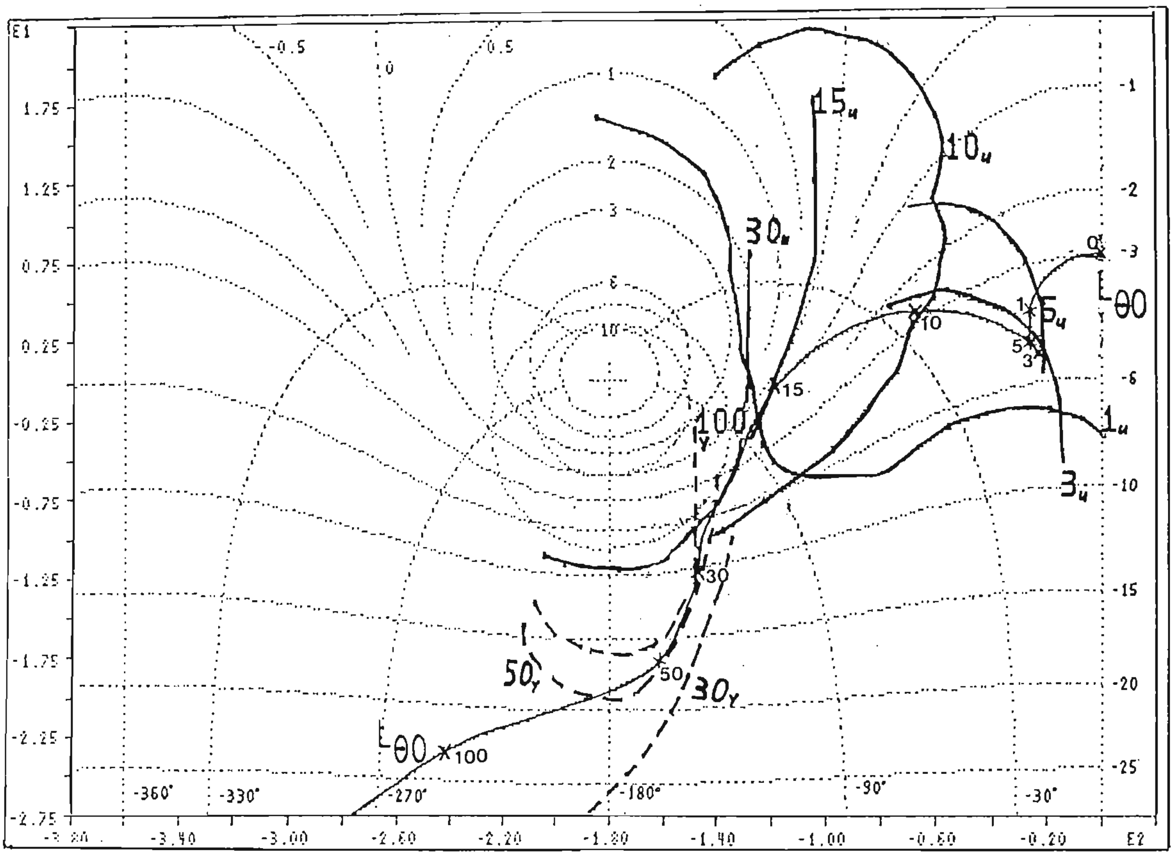


Figure 5.4 – Nominal input pitch angle loop transfer function, $L_{\theta 0}(j\omega)$ design

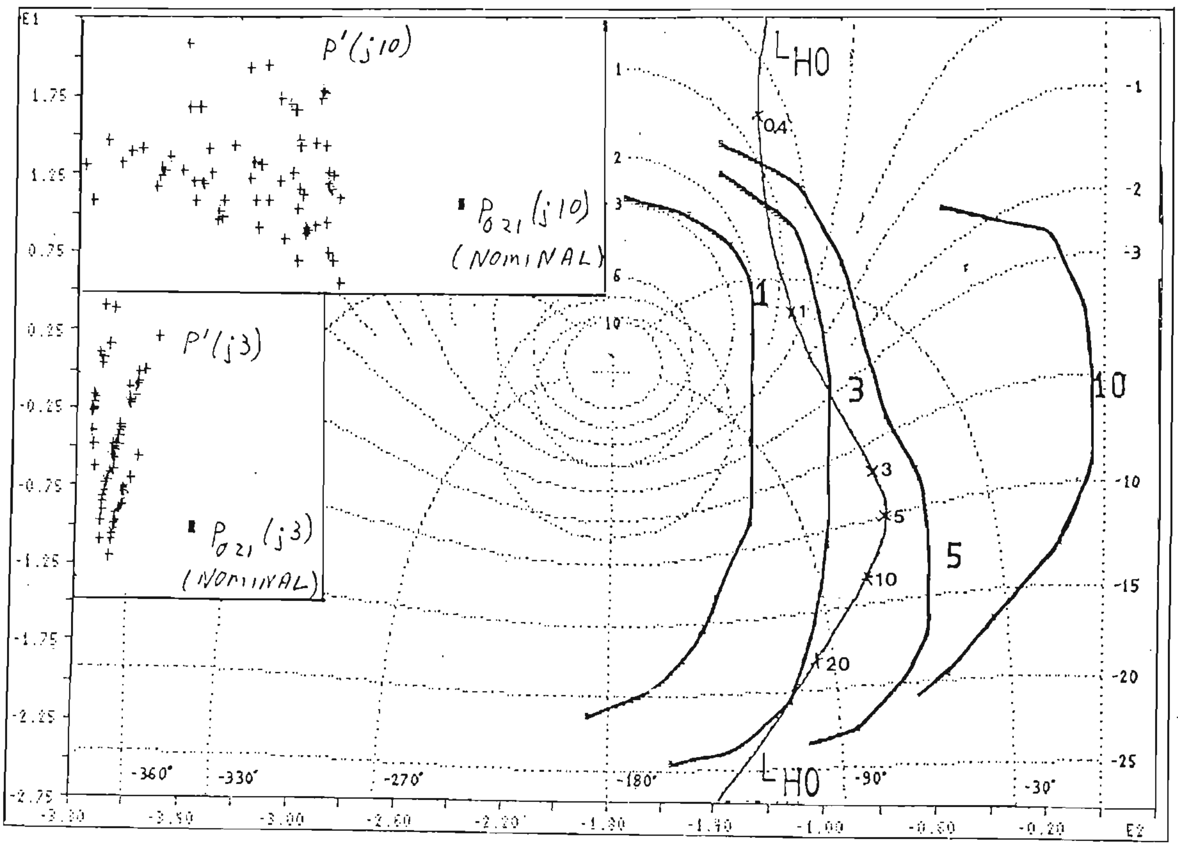


Figure 5.5 – Templates and bounds for $L_H(j\omega)$ design and L_{H0} violating bounds

5.2 Results

The Bode magnitude diagrams for the various transfer functions calculated at all the entries of the plant set used in the design are shown in Figures 5.6 (reference–output), 5.7 (disturbance–output) and 5.8 (disturbance–input). These Bode plots include the motor characteristics (with corner frequency ω_n=100 rad/s).

Simulated data for the flying machine with the above controller structure and design is presented in Figures 5.9 and 5.10:

5.9) Disturbance regulation when measured disturbances are added to the wing torques and drags (using disturbance data from test R22).

5.10) Response to a step change in height reference of 0.75m.

Actual flight data for the flying machine with the above controller structure and design is presented in Figures 5.11 and 5.12:

5.11) Disturbance regulation with constant height and pitch angle set–points, Test # Diag1. Figure 5.11a was annotated using a video–tape recording of the flight and shows where the flying machine was held at the beginning of the flight, and where the pitch angle sensor (potentiometer) failed at the end of the flight. Figure 5.11b shows a portion (presented in Boje, 1989) of the data record from Figure 5.11a and can be compared to the simulation result, Figure 5.9. Figure 5.11c shows wing rate demands obtained by differentiating the position demand signals numerically.

5.12) Attempted height step response test with step changes in height error of 0.75m. It is possible (from viewing video material) that the pitch angle sensor did not function reliably during this test. The simulation data, Figure 5.10, suggests better (when compared to Figure 5.12b), although still poor, height regulation

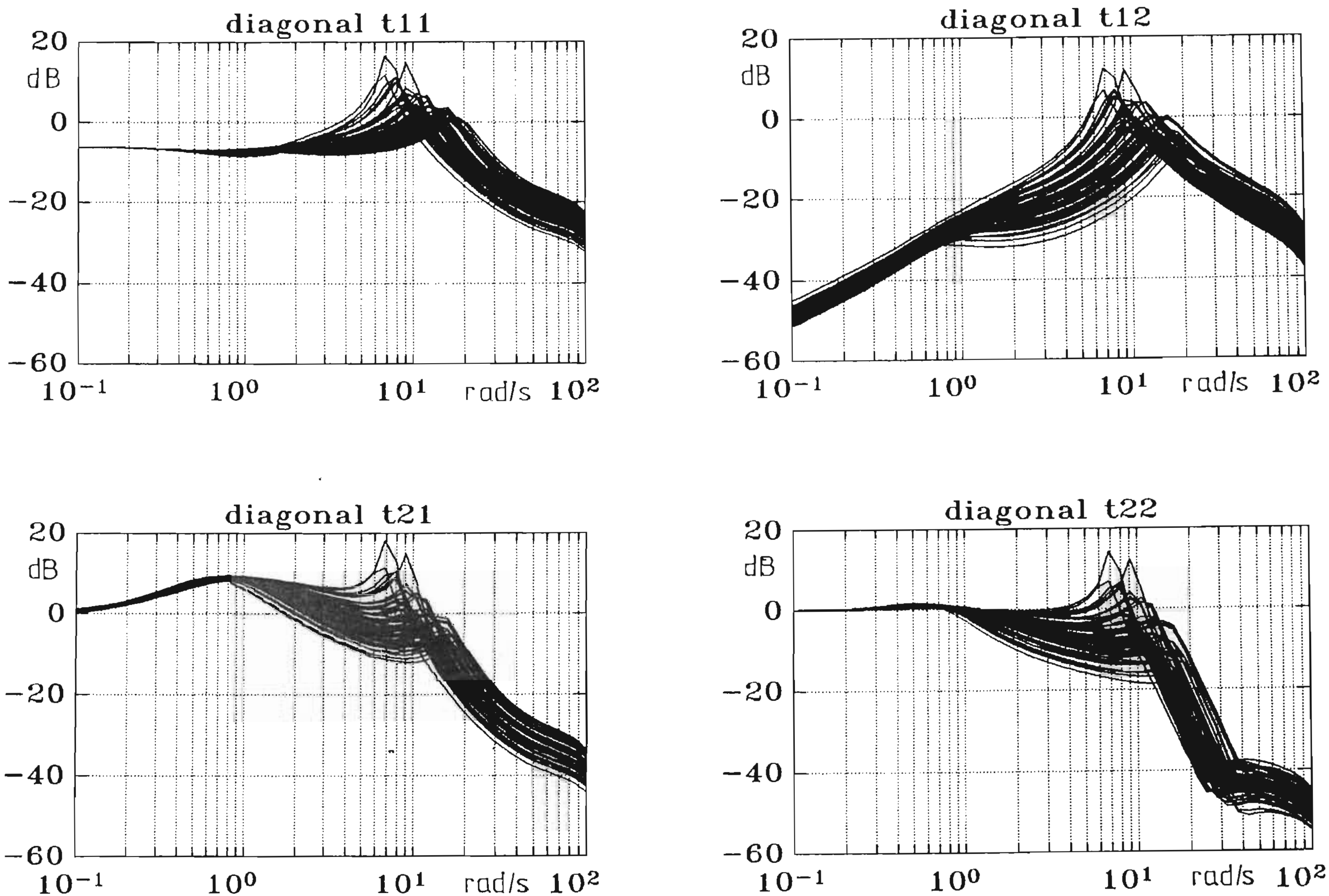


Figure 5.6 – Bode magnitude diagrams for $\underline{T}_{Y/R}$

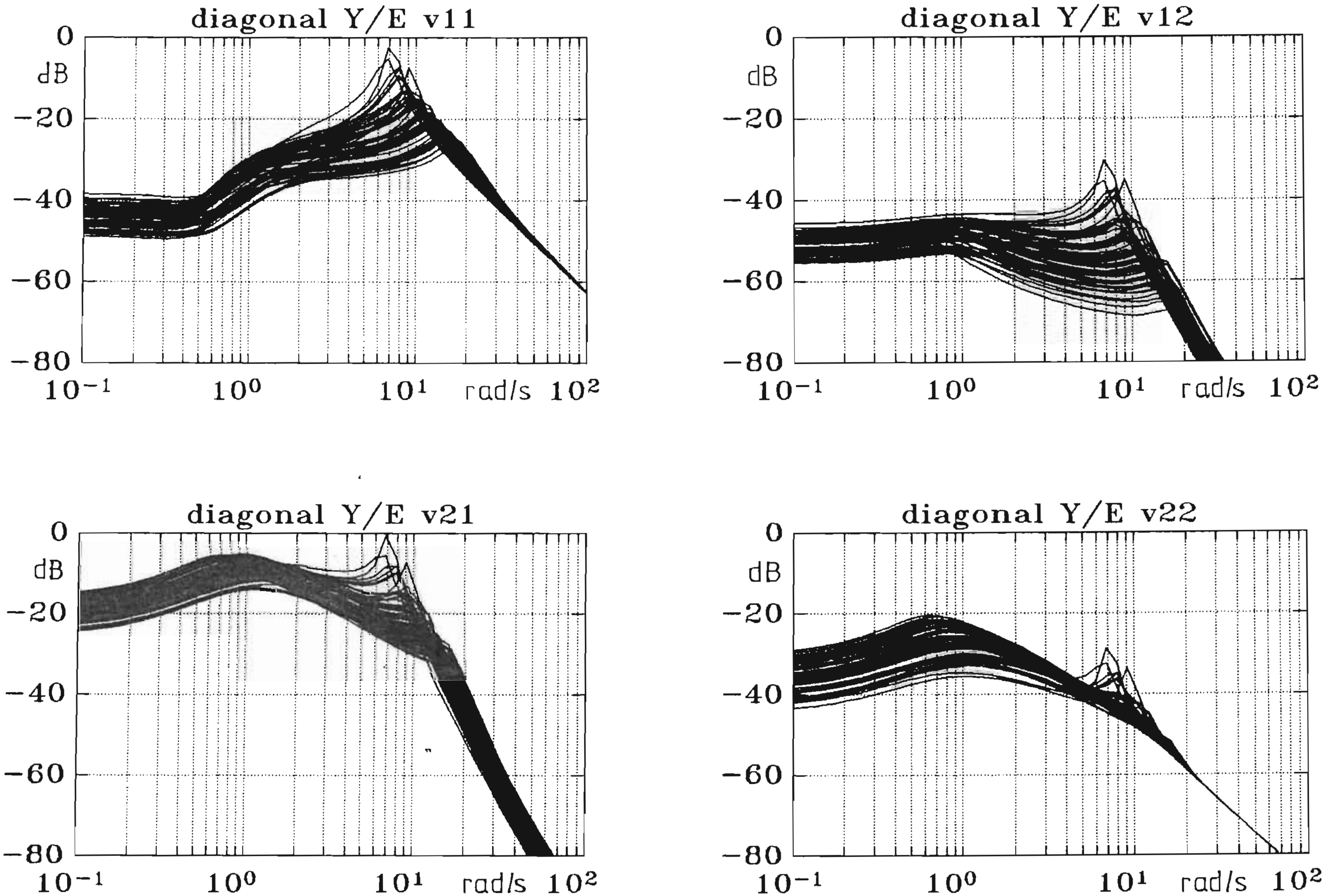


Figure 5.7 – Bode magnitude diagrams for $\underline{T}_{Y/E}$

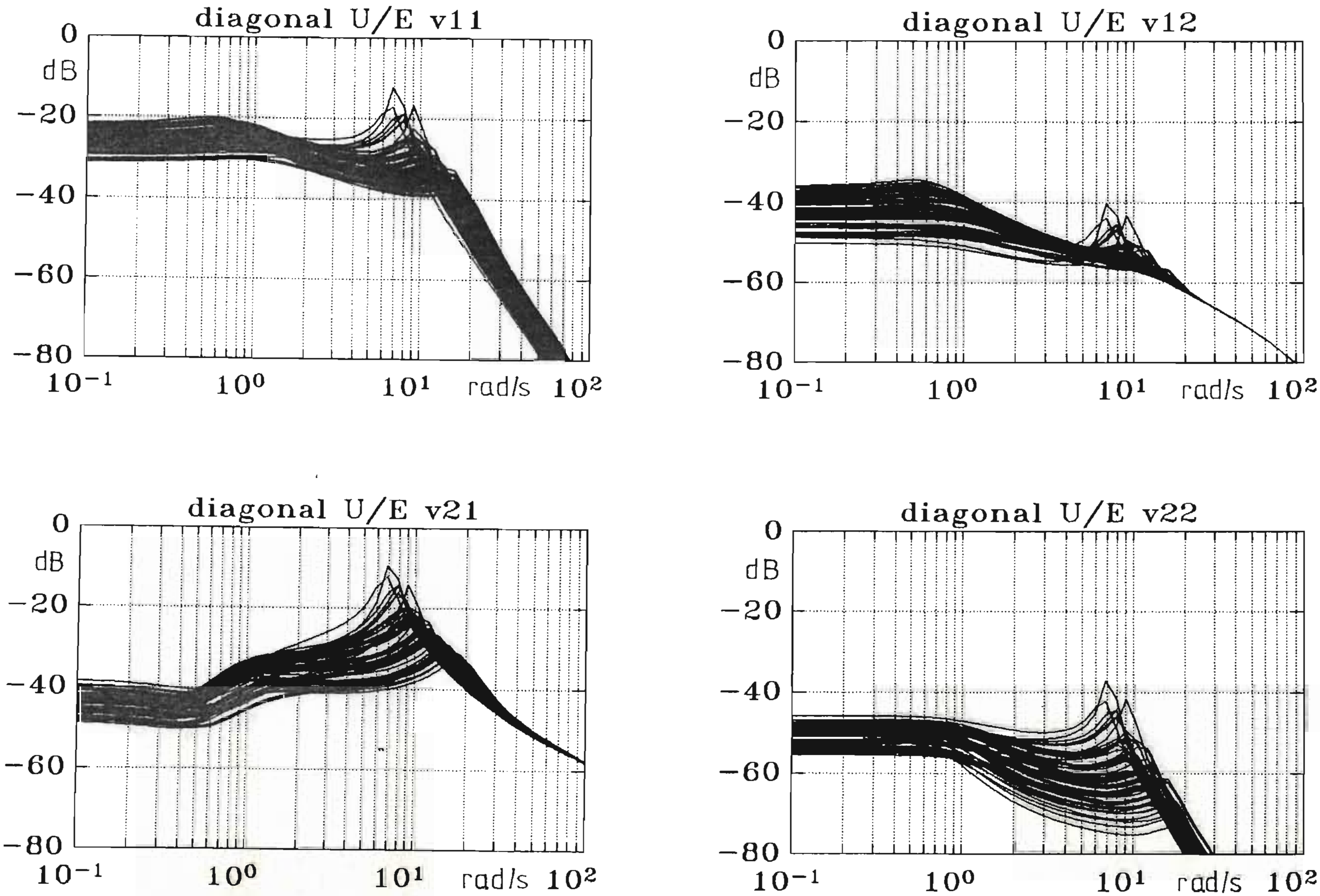


Figure 5.8 – Bode magnitude diagrams for $\underline{T}_{U/E}$

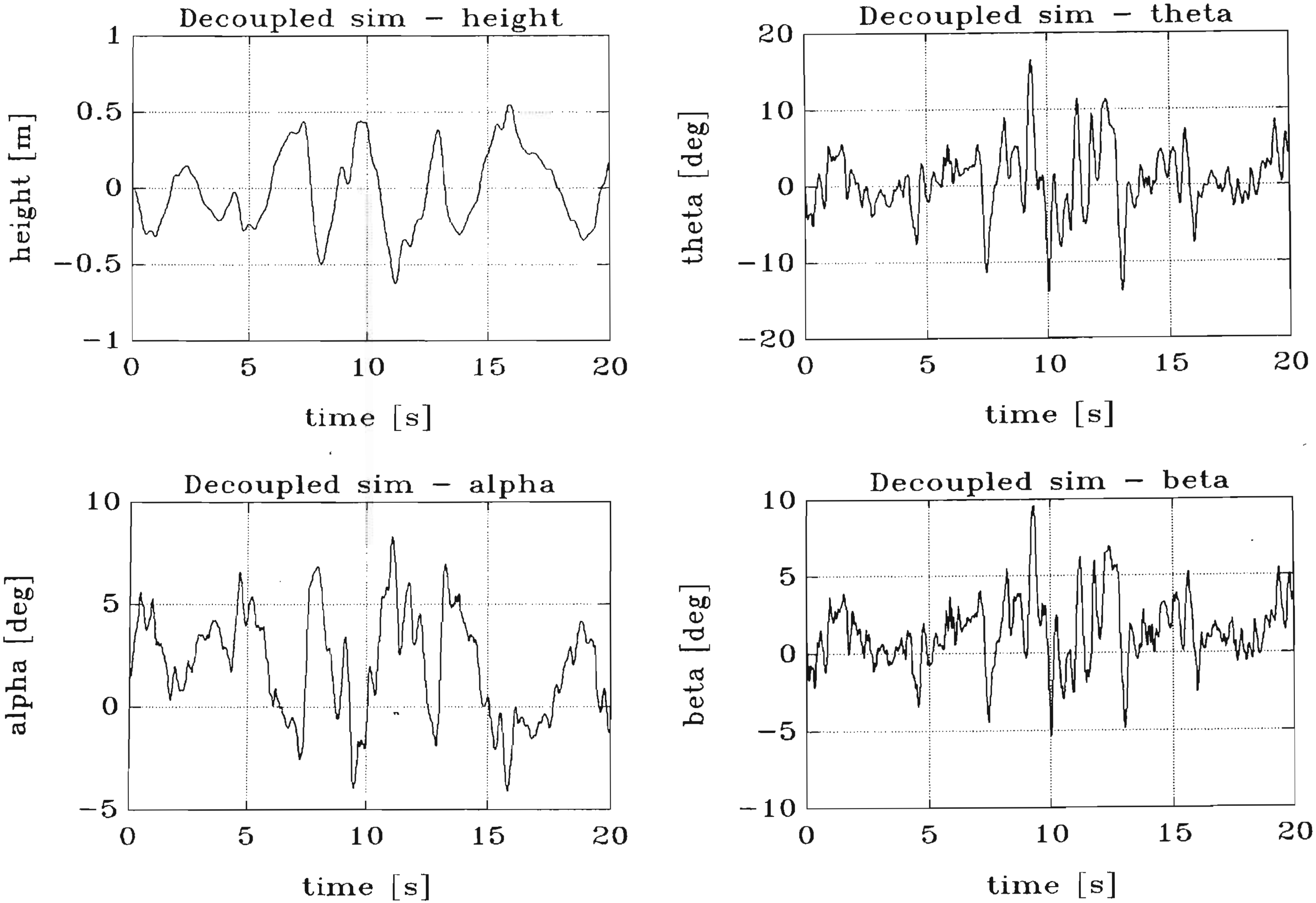


Figure 5.9 – Simulated disturbance regulation with measured disturbances

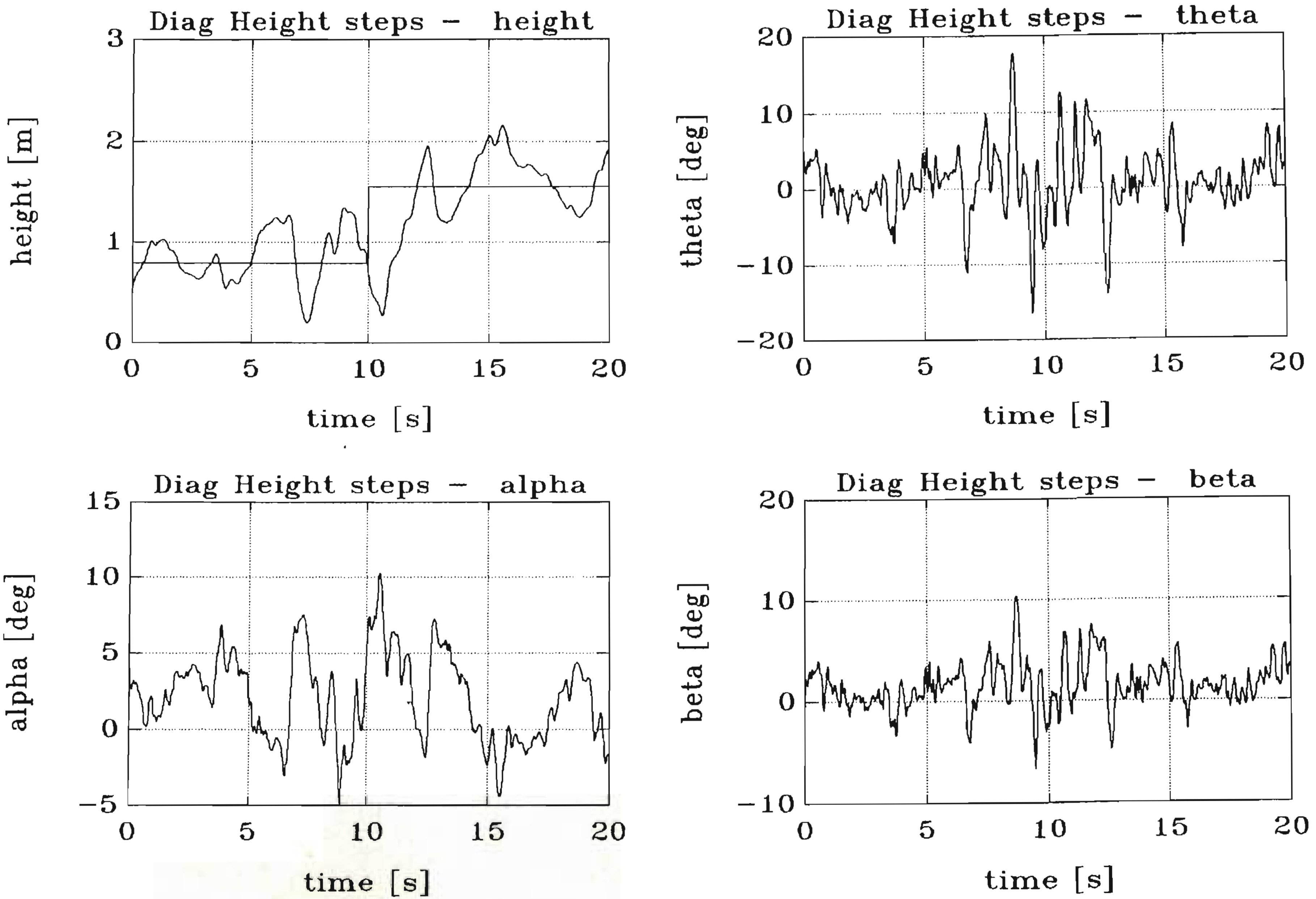


Figure 5.10 – Simulated response to a step change in height reference of 0.75m (disturbances from test R22)

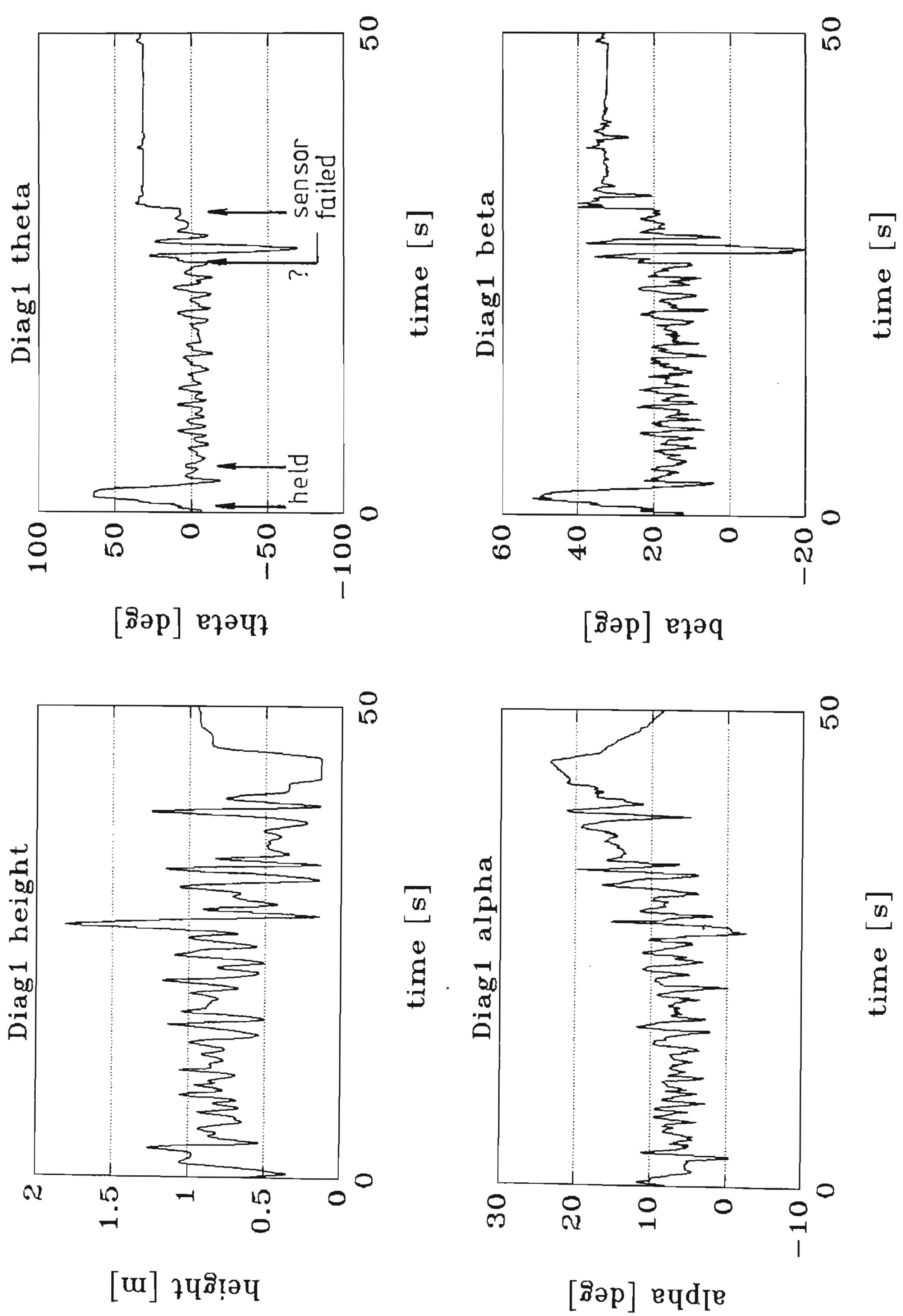


Figure 5.11a – Flight data (Test # Diag1) – disturbance regulation with constant height and pitch angle set–points

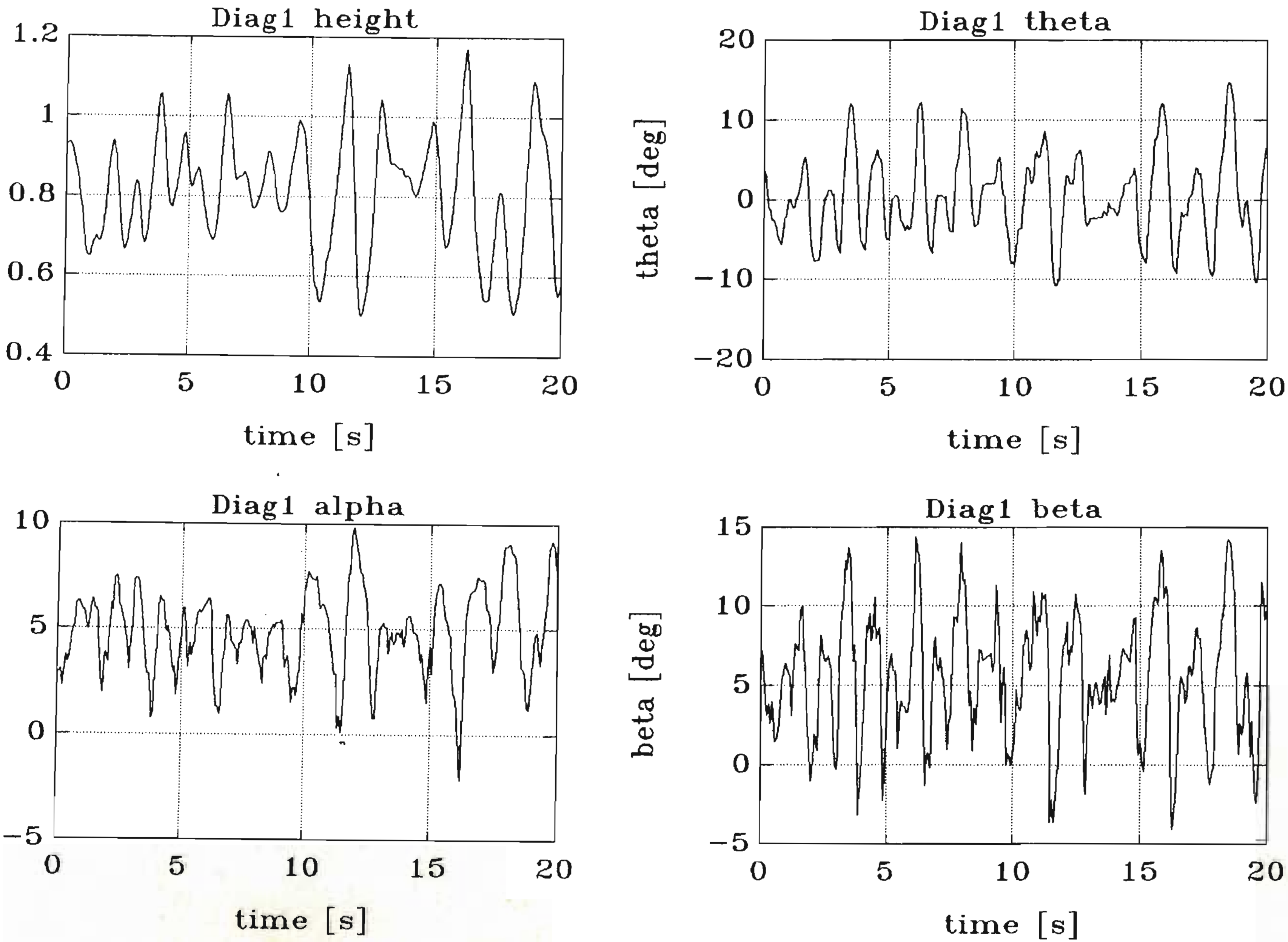


Figure 5.11b – Detail from Figure 5.11a – disturbance regulation with constant height and pitch angle set–points

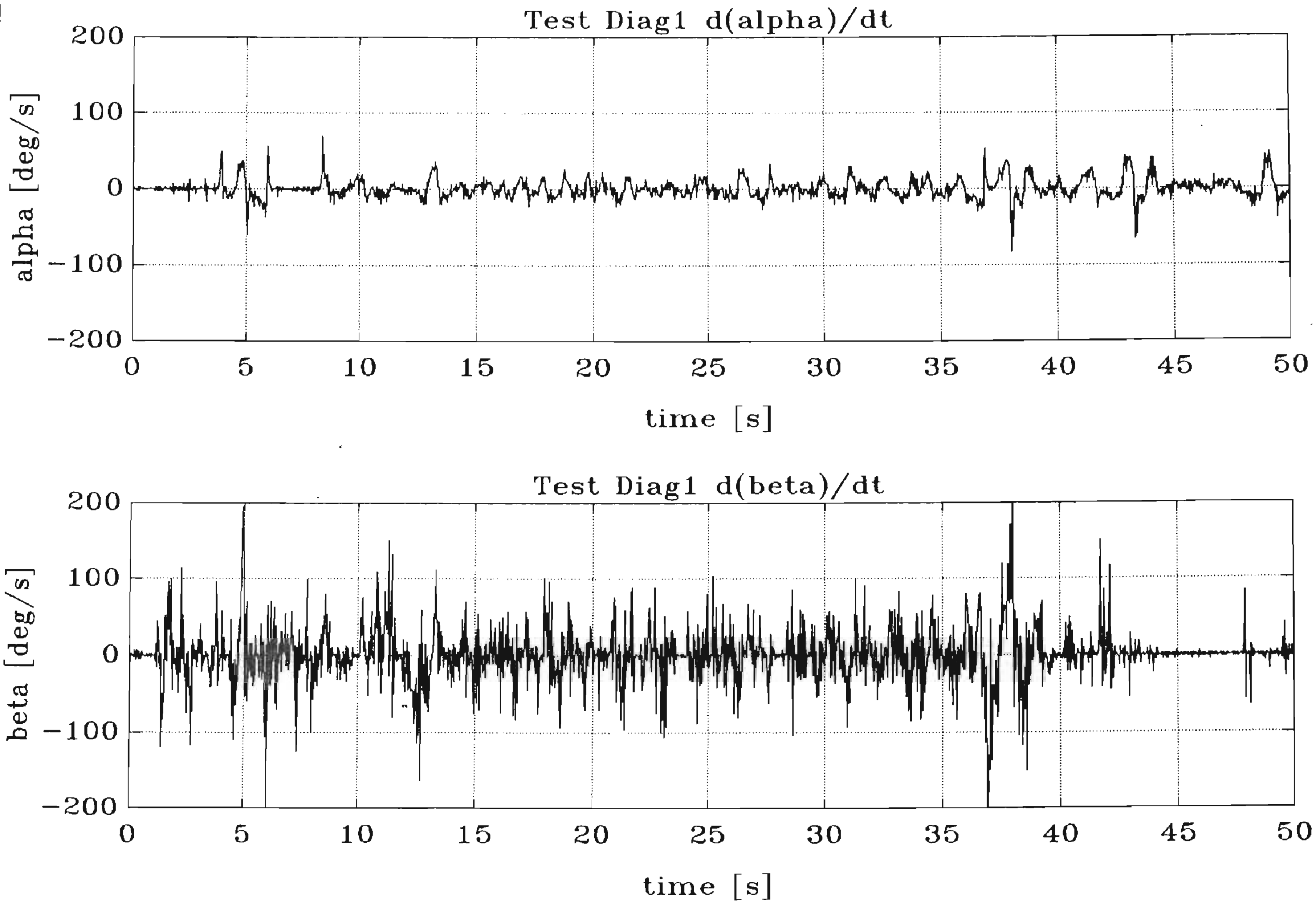


Figure 5.11c – Flight data (Test # Diag1) – Front and back wing rate demands

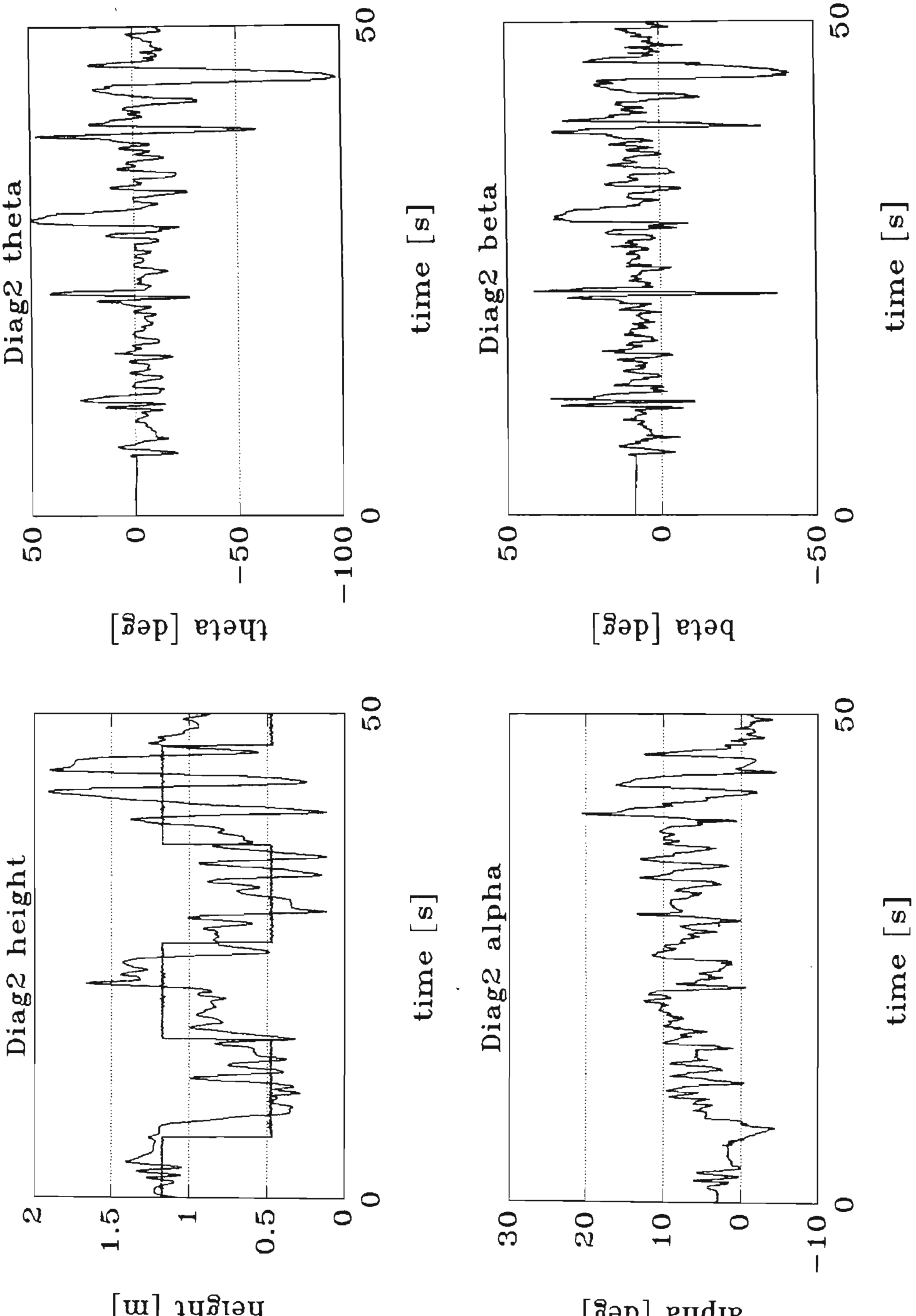


Figure 5.12a – Flight data (Test # Diag2) – response to a step changes in height error of 0.75m

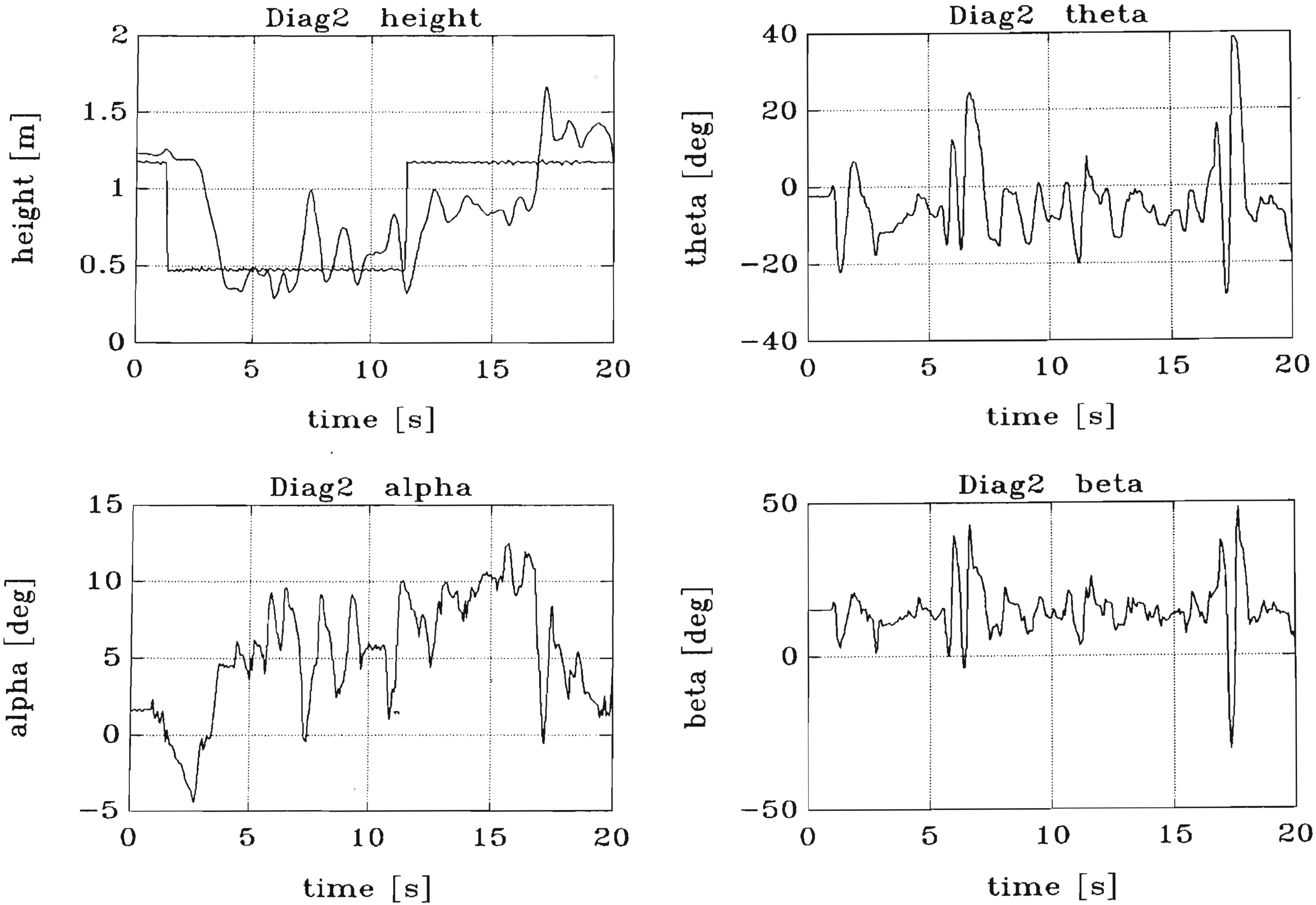


Figure 5.12b – Detail from Figure 5.12a – response to a step changes in height error of 0.75m

5.3 Conclusions

1) Figure 5.11c highlights a fundamental problem with this design. Because it was not possible to achieve the desired bandwidth in the height loop, the front wing is clearly under–utilized. Because the specifications were not rigorous, the consequence to the pitch angle loop design of not achieving specifications in the height loop were not investigated further than the drawing of Bode plots of the resulting closed loop behaviour.

2) When there are no other problems, qualitatively the pitch angle regulation is good and the height regulation is rather poor. As the design is approximate due to the use of linearisations about some operating points and is also conservative due to the use of the Schwartz inequality, eq(3.27), it was hoped that the flying machine might still provide reasonable performance despite the violation of design bounds in the height loop (Figure 5.5). The oscillations in height and angle of attack (at about 5 rad/s) are not surprising, considering the violation of the design boundaries but reducing overall gain in the height loop to avoid violation of the bounds resulted in such poor height regulation that the machine crashed (physical boundaries were violated!).

CHAPTER 6 – QUASI DECOUPLING CONTROLLER

6.1 Introduction and design

This chapter describes the design, implementation and testing of a controller designed by first modifying the plant input so that the (single input single output dynamic) controllers' outputs result in the wings being turned in the correct direction to regulate height and pitch angle errors. As the controller described here was designed in a very short time, the design is less complete than the previous two designs. Simulations are not presented, only actual results and design specifications are based on experience and heuristics rather than via rough analysis as in the previous designs.

The diagonal controller described in Chapter 5 has closed loop disturbance to input transfer functions with factors of the form, 1/(wing angle to lift gain) and 1/(wing angle to torque gain), most clearly seen in eq(5.14) and eq(5.15) where the factor is $(1/k_{tb})$. In an attempt to make better use of the plant inputs, a controller structure which turns the wings at a rate determined by their maximum rates in the direction to correct errors was investigated. As the rate limits of the front and back wings are the same (200°/s), for a diagonal controller, this corresponds to modifying the open loop plant input via the filter,

$$\underline{\mathbf{H}} = \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix} \tag{6.1}$$

so that the modified plant numerator matrix is,

$$\underline{\mathbf{N}}\,\underline{\mathbf{H}} = \begin{bmatrix} k_{tf}+k_{tb} & k_{tf}-k_{tb} \\ k_{\ell f}-k_{\ell b} & k_{\ell f}+k_{\ell b} \end{bmatrix} = \begin{bmatrix} \Sigma_t & -\Delta_t \\ -\Delta_\ell & \Sigma_\ell \end{bmatrix} \tag{6.2}$$

As $\Delta_t \approx k_t$ and $\Sigma_\ell \approx k_\ell$ (Appendix 3.1 gives the formulas for evaluating the relevant the parameters from eq(2.21)),

$$\underline{\mathbf{P}}' = \underline{\mathbf{D}}^{-1}\underline{\mathbf{N}}\,\underline{\mathbf{H}} = \begin{bmatrix} \dfrac{\Sigma_t}{J(s)} & \dfrac{-\Delta_t}{J(s)} \\ \dfrac{-(\Delta_\ell J s^2+\Delta_\ell \mu_t s-2k)}{J(s)\,M(s)} & \dfrac{\Sigma_\ell J s^2+\Sigma_\ell \mu_t s}{J(s)\,M(s)} \end{bmatrix} \tag{6.3}$$

and,

$$\underline{\mathbf{P}}'^{-1} = \underline{\mathbf{H}}^{-1}\,\underline{\mathbf{N}}^{-1}\,\underline{\mathbf{D}} = \frac{1}{2k}\begin{bmatrix} \Sigma_\ell J s^2+\Sigma_\ell \mu_t s & \Delta_t(ms^2+\mu_\ell s) \\ \Delta_\ell J s^2+\Delta_\ell \mu_t s-2k & \Sigma_t(ms^2+\mu_\ell s) \end{bmatrix} \tag{6.4}$$

The elements are labeled, $[P'^{-1}]_{ij} = \dfrac{1}{q_{ij}}$.

The controller is,

$$\underline{\mathbf{G}}(s) = \begin{bmatrix} g_\theta(s) & 0 \\ 0 & g_H(s) \end{bmatrix} \tag{6.5}$$

and the design proceeds along the lines of the design in Chapter 5. Eq(6.3) and eq(6.4) potentially ease the QFT design (compared to the corresponding eq(5.2) and eq(5.9) of Chapter 5) because the elements which require over–design (off–diagonal in this case and diagonal in Chapter 5) are smaller, relative to the elements used for the design. Heuristically, this control structure also makes better use of the available actuator power

<u>Output design</u>

$$(\underline{P}'^{-1} + \underline{G})\, \underline{T}_{EY} = (\underline{N}\,\underline{H})^{-1} \tag{6.6}$$

$$\begin{bmatrix} \Sigma_\ell J s^2 + \Sigma_\ell \mu_t s + 2k g_\theta(s) & \Delta_t (m s^2 + \mu_\ell s) \\ \Delta_\ell J s^2 + \Delta_\ell \mu_t s - 2k & \Sigma_t (m s^2 + \mu_\ell s) + 2k g_H(s) \end{bmatrix} \begin{bmatrix} t_{\theta t} & t_{\theta\ell} \\ t_{Ht} & t_{H\ell} \end{bmatrix} = \begin{bmatrix} \Sigma_\ell & \Delta_t \\ \Delta_\ell & \Sigma_t \end{bmatrix}$$

Writing eq(6.6) implicitly element by element,

$$t_{\theta t} = \left[\frac{1}{1+L_{\theta Y}}\right] [q_{11}] \left[\Sigma_\ell/(2k) - \left[\frac{1}{q_{12}}\right] t_{Ht}\right] \tag{6.7}$$

$$t_{\theta\ell} = \left[\frac{1}{1+L_{\theta Y}}\right] [q_{11}] \left[\Delta_t/(2k) - \left[\frac{1}{q_{12}}\right] t_{H\ell}\right] \tag{6.8}$$

$$t_{Ht} = \left[\frac{1}{1+L_{HY}}\right] [q_{22}] \left[\Delta_\ell/(2k) - \left[\frac{1}{q_{21}}\right] t_{\theta t}\right] \tag{6.9}$$

$$t_{H\ell} = \left[\frac{1}{1+L_{HY}}\right] [q_{22}] \left[\Sigma_t/(2k) - \left[\frac{1}{q_{21}}\right] t_{\theta\ell}\right] \tag{6.10}$$

with,

$L_{\theta Y} = q_{11}\, g_\theta\, m$ – output angle of attack loop transfer function

$$q_{11} = \frac{2k}{\Sigma_\ell (J s^2 + \mu_t s)}$$

$L_{HY} = q_{22}\, g_H\, m$ – output height loop transfer function

$$q_{22} = \frac{2k}{\Sigma_t (m s^2 + \mu_\ell s)}$$

After applying the Schwartz inequality and substituting the bound, A (eq(3.23)), for elements of $\underline{T}_{YE}$ appearing on the right hand side of the equations (eq(3.25)), for $g_\theta(s)$, the design equations are,

$$t_{\theta t} \le \left|\frac{1}{1+L_{\theta Y}}\right| |q_{11}| \left[\Sigma_\ell/(2k) + \left|\frac{1}{q_{12}}\right| a_{Ht}\right] \tag{6.11}$$

$$t_{\theta\ell} \le \left|\frac{1}{1+L_{\theta Y}}\right| |q_{11}| \left[\Delta_t/(2k) + \left|\frac{1}{q_{12}}\right| a_{H\ell}\right] \tag{6.12}$$

Input design

$$(\underline{G}^{-1} + \underline{P}')\, \underline{H}^{-1}\, \underline{V}_{UE} = -\, D^{-1}$$

$$(\underline{G}^{-1} + \underline{P}')\, \underline{V}'_{UE} = -\, D^{-1} = \underline{P}^{*} \qquad (6.13)$$

$$\begin{bmatrix} \frac{1}{g_{\theta}(s)} + \frac{\Sigma_t}{J(s)} & \frac{-\Delta_t}{J(s)} \\ \frac{-(\Delta_{\ell} Js^2 + \Delta_{\ell}\mu_t s - 2k)}{J(s)\, M(s)} & \frac{1}{g_H(s)} + \frac{\Sigma_{\ell} Js^2 + \Sigma_{\ell}\mu_t s}{J(s)\, M(s)} \end{bmatrix} \begin{bmatrix} v'_{1t} & v'_{1\ell} \\ v'_{2t} & v'_{2\ell} \end{bmatrix} = \begin{bmatrix} \frac{-1}{J(s)} & 0 \\ \frac{-k_{\ell}}{J(s)M(s)} & \frac{-1}{M(s)} \end{bmatrix}$$

Writing eq(6.13) implicitly,

$$v'_{1t} = \frac{L_{\theta U}}{1+L_{\theta U}} \left[\frac{\Delta_t}{\Sigma_t} v'_{2t} - \frac{1}{\Sigma_t} \right] \qquad (6.14)$$

$$v'_{1\ell} = \frac{L_{\theta U}}{1+L_{\theta U}} \left[\frac{\Delta_t}{\Sigma_t} v'_{2\ell} \right] \qquad (6.15)$$

$$v'_{2t} = \frac{L_{HU}}{1+L_{HU}} \left[\frac{\Delta_{\ell}(Js^2+\mu_t s)-2k}{\Sigma_{\ell}(Js^2+\mu_t s)} v_{1t} - \frac{k_{\ell}}{\Sigma_{\ell}(Js^2+\mu_t s)} \right] \qquad (6.16)$$

$$v'_{2\ell} = \frac{L_{HU}}{1+L_{HU}} \left[\frac{\Delta_{\ell}(Js^2+\mu_t s)-2k}{\Sigma_{\ell}(Js^2+\mu_t s)} v_{1\ell} - \frac{Js^2+\mu_t s}{\Sigma_{\ell}(Js^2+\mu_t s)} \right] \qquad (6.17)$$

with,

$L_{\theta U} = p'_{11}\, g_{\theta}\, m$ – **input** $\angle$attack loop transfer function

$L_{HU} = p'_{22}\, g_{\alpha}\, m$ – **input** height loop transfer function

After applying the Schwartz inequality and substituting a bound, $\underline{B}'$ for elements of $\underline{V}'_{UE}$ appearing on the right hand side of the equations, for $g_{\theta}(s)$, the design equations are,

$$v'_{1t} \leq \left| \frac{L_{\theta U'}}{1+L_{\theta U'}} \right| \left[\frac{\Delta_t}{\Sigma_t} b'_{2t} + \frac{1}{\Sigma_t} \right] \qquad (6.18)$$

$$v'_{1\ell} \leq \left| \frac{L_{\theta U'}}{1+L_{\theta U'}} \right| \left[\frac{\Delta_t}{\Sigma_t} b'_{2\ell} \right] \qquad (6.19)$$

To get an idea of what the specifications should be, solve eq(6.6) for the output specification at steady state,

$$\underline{T}_{EY}(j0) = \begin{bmatrix} \frac{1}{k_t + \Sigma_t g_{\theta}} & \frac{\Delta_{\ell} g_{\theta}}{\Sigma_{\ell} g_H (k_t + \Sigma_t g_{\theta})} \\ \frac{\Delta_t}{\Sigma_{\ell}(k_t + \Sigma_t g_{\theta})} & \frac{1}{\Sigma_{\ell} g_H} \end{bmatrix} \qquad (6.20)$$

with,

$$g_H = g_H(j0),\ g_\theta = g_\theta(j0)$$

As in the anti diagonal controller of Chapter 5, the steady state gains of the controllers cannot be expected to be more than 1.0 (=0dB). This gain has the following implications for this structure: A 10° angle of attack error will result in a back wing position of 20° with respect to the ground (at which angle the wing starts to stall) and a front wing position of 0° with respect to the ground. A height error of 0.1m will result in both wings turning 5° up. Larger steady state gains could result in position saturation of the wings. In this case at nominal values,

$$\underline{T}_{EY0}(j0) = \begin{bmatrix} 11.4 & 10.4 \\ 0.84 & 4.12 \end{bmatrix} \times 10^{-3} \tag{6.21}$$

(with $g_\alpha(j0) = g_\beta(0) = 1$)

As the nominal is chosen at relatively high $k_{..}$ values, $\max(T_{EY}(j0))$ could have 10 dB larger magnitude.

As the controller will be designed to be strictly proper it will be open circuit at high enough frequencies and,

$$\lim_{s \to j\infty} \underline{T}_{EY}(s) = \lim_{s \to j\infty} \underline{P}(s) = \lim_{s \to j\infty} \frac{1}{s^2} \begin{bmatrix} \frac{k_{tf}}{J} & \frac{-k_{tb}}{J} \\ \frac{k_{\ell f}}{m} & \frac{k_{\ell b}}{m} \end{bmatrix} \tag{6.22}$$

At nominal values this is,

$$\lim_{s \to j\infty} \underline{T}_{EY0}(s) = \lim_{s \to j\infty} \frac{1}{s^2} \begin{bmatrix} 185 & -313 \\ 27 & 80 \end{bmatrix} \tag{6.23}$$

Since at low frequencies, the q_{ij} are essentially integrators, $q_{ij}/(1+L_{ij}) \approx 1/g_{ij}$ in equations (5.3) to (5.6) and t_{ij} are therefore expected to be flat at low frequencies. For specifications on $|\underline{T}_{EY}(j\omega)|$ the low–pass characteristic with asymptotes, $|\underline{T}_{EY0}(j0)|$ +10dB (to allow for low gain plants and some overshoot) and $|\underline{T}_{EY0}(j\infty)|$ might be used and this specification modified as a result of tests or simulation. The high frequency transfer of disturbance to the plant output is fixed by the plant structure.

One could follow a similar procedure from eq(6.13) to examine the steady–state transfer of disturbance to the plant input for the nominal plant. Notice that from eq(6.13) onwards, the direct control during the design of the plant input has been lost as a result of the filter, $\underline{\mathbf{H}}$. (Compare this to the design in Chapter 4). Specifications for the modified inputs, $\underline{\mathbf{U}}' = \underline{\mathbf{H}}^{-1}\ \underline{\mathbf{U}}$ could be developed and design using the modified plant $\underline{\mathbf{P}}'$ from eq(6.3) executed.

A more direct method than the above is to define the following ad–hoc specifications and modify the final design using the insight obtained via the design:

Summary of Specifications

1) It must fly.

2) The output loops must have the highest bandwidth possible so that $\left|\frac{1}{1+L_{\cdot Y}}\right|$ << 0dB until q_{ij} << 0dB

3) To avoid high amplification of disturbances at the plant inputs and outputs, $\left|\frac{1}{1+L_{\cdot}}\right| \leq$ 3dB and $\left|\frac{L_{\cdot}}{1+L_{\cdot}}\right| \leq$ 3dB are required for all loops.

4) To ensure that the controllers can be realized simply, the controllers may not have more than +20dB/decade roll–up and must roll off around the corner frequency of the wing actuators.

These specifications are very rough and really only ensure some sort of robust stability via (3) and (practical) realizability via (4).

Pitch angle loop design

Bounds for the $L_{\theta\cdot}$ and some templates are shown in Figure 6.1 (input) and Figure 6.2 (output). The nominal plant was chosen as $p_{0\ 12}$ for input and output loops. As a result the the nominal for the output loop lies outside the template and may itself violate the output loop boundaries. Note that the input and output bounds are much closer to each other in this design than in the diagonal controller design of Chapter 5. This justifies the statement made in Section 6.1 that the controller structure eases the design problem.

The controller designed to satisfy the angle of attack loop specifications is,

$$g_\theta(s) = \frac{(s+1)(s/10+1)}{(s/0.21+1)\left((s/150)^2+2(s/150)+1\right)} \tag{6.24}$$

The worst case bounds and nominal input angle of attack loop transfer function resulting from this controller, $L_{\theta 0}(j\omega)$, with actuator corner frequency of 100 rad/s is shown in Figure 6.3.

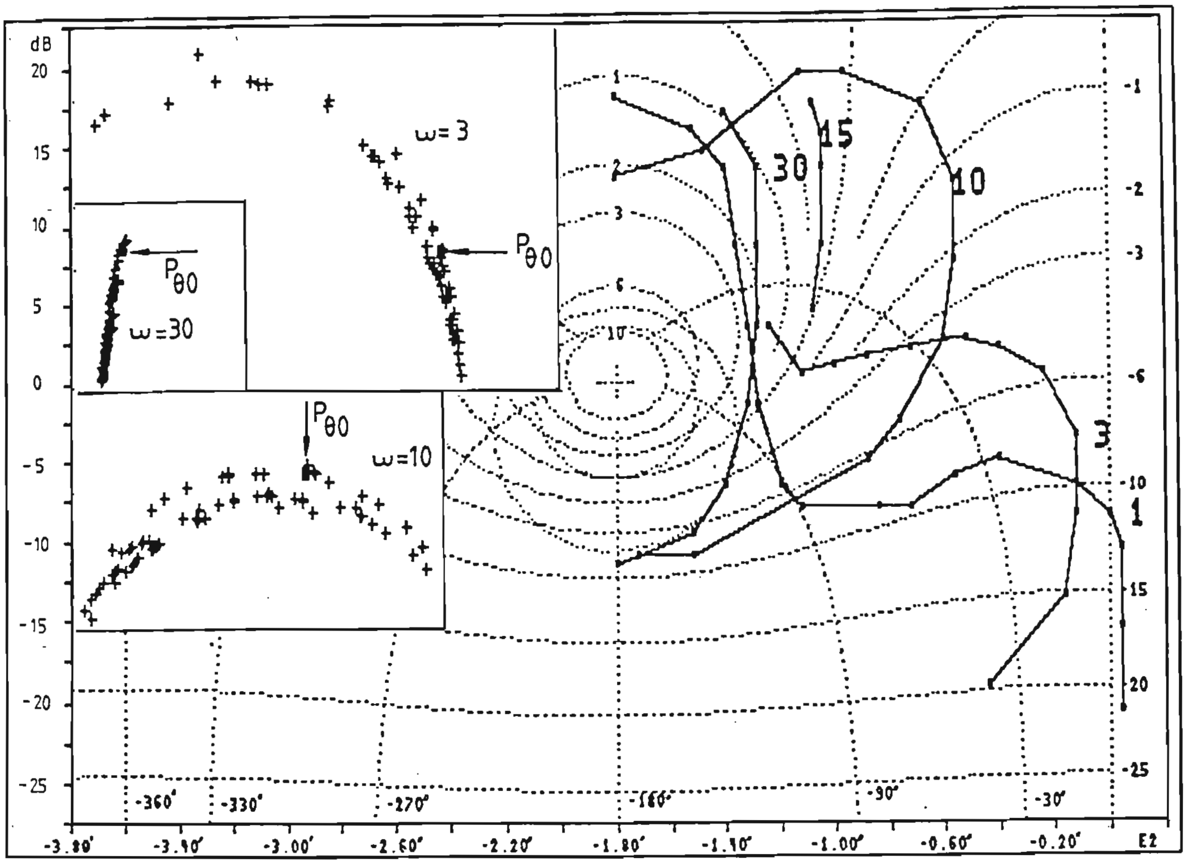


Figure 6.1 – Templates and bounds for $L_{\theta U'}(j\omega)$ (input) design

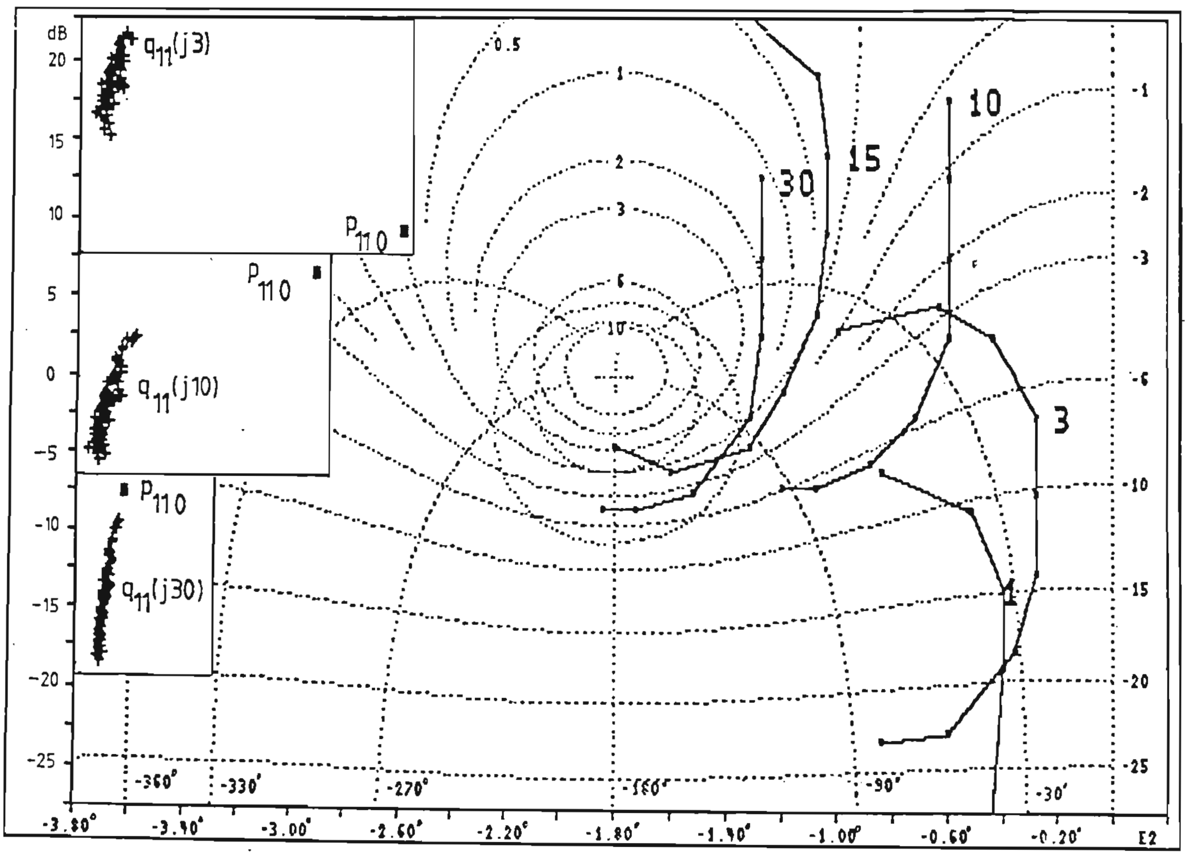


Figure 6.2 – Templates and bounds for $L_{\theta Y}(j\omega)$ (output) design

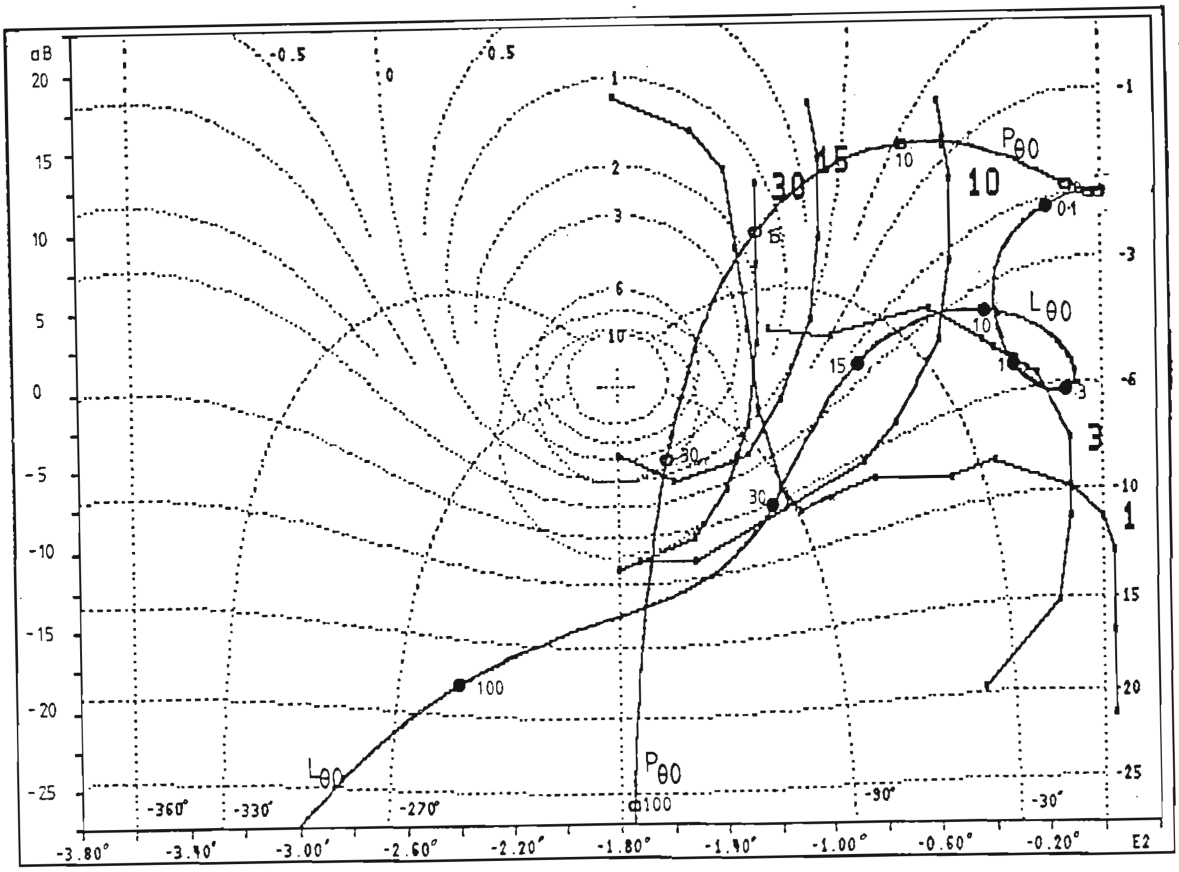


Figure 6.3 – Worst case bounds and nominal input pitch angle loop transfer function, $L_{\theta 0}(j\omega)$ design

Height loop design

The pitch angle loop was closed and templates constructed for the design of the height loop. The templates for this loop are those for an exact SISO design but the numerator terms of the closed loop $\underline{T}$ have not been evaluated so that the height design is principally a stability design. The nominal plant selected for the design execution of this second step was $p_{0\,21} \neq p'_{0\,2}$ of eq(3.36), which accounts for the nominal again being outside the template. This choice of nominal is justified by the ease of keeping track of the second order $p_{0\,21}$ rather than the high order $p'_{0\,2}$. Compared to the design of Chapter 5, the nominal is quite close to the plant templates. Figure 6.4 shows the bounds on the nominal input height loop transfer function, some relevant templates and a loop transfer function corresponding to the controller,

$$g_H(s) = \frac{(s+1)(s/3+1)}{(s/0.21+1)((s/50)^2+1.4(s/50)+1)} \tag{6.25}$$

Notice that the bounds are more than satisfied. The original design was modified to give more phase lead around 5 rad/s when some poorly damped oscillations were observed at that frequency. The final "as implemented" result only is shown.

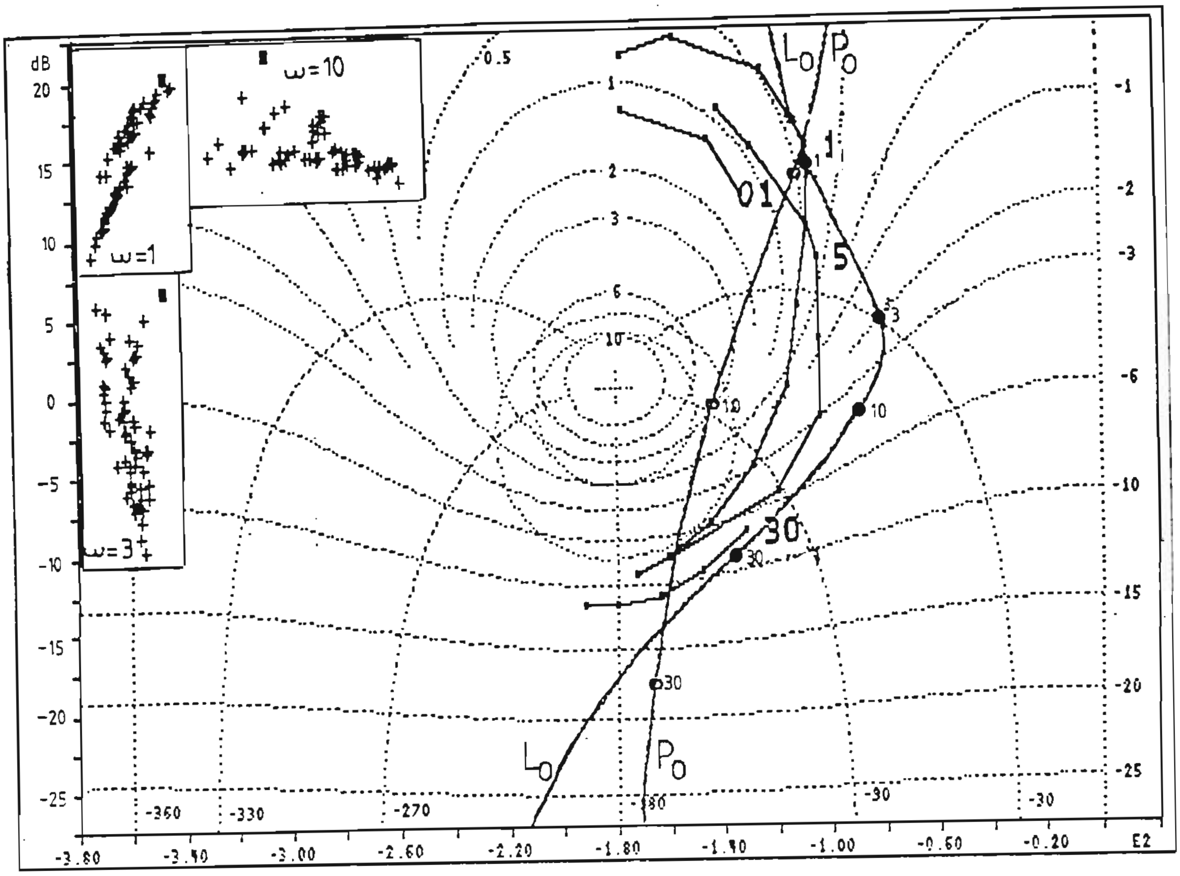


Figure 6.4 – Templates and bounds for $L_H(j\omega)$ design and $L_{H0}(j\omega)$ more than satisfying bounds. $\omega_{n\ motor} = 100$ rad/s

6.2 Results

The Bode magnitude diagrams for the various transfer functions calculated at all the entries of the plant set used in the design are shown in Figures 6.5 (reference–output), 6.6 (disturbance–output) and 6.7 (disturbance–input). These Bode plots include the motor characteristics (with corner frequency ω_n=100 rad/s).

Actual flight data for the flying machine with the above controller structure and design is presented in Figures 6.8 and 6.9:

6.8) Disturbance regulation with constant height and pitch angle set–points, test One1. Figure 6.8a was annotated using a video–tape recording of the flight and shows where the rear actuator failed at the end of the flight test (the final drive shaft sheared off at the spline). Figure 6.8b shows a portion of the data record from Figure 6.8a where the flight was at its best. Figure 6.8c shows wing rate demands obtained by differentiating the position demand signals numerically.
6.9) Disturbance regulation with constant height and pitch angle set–points, Test # One2.

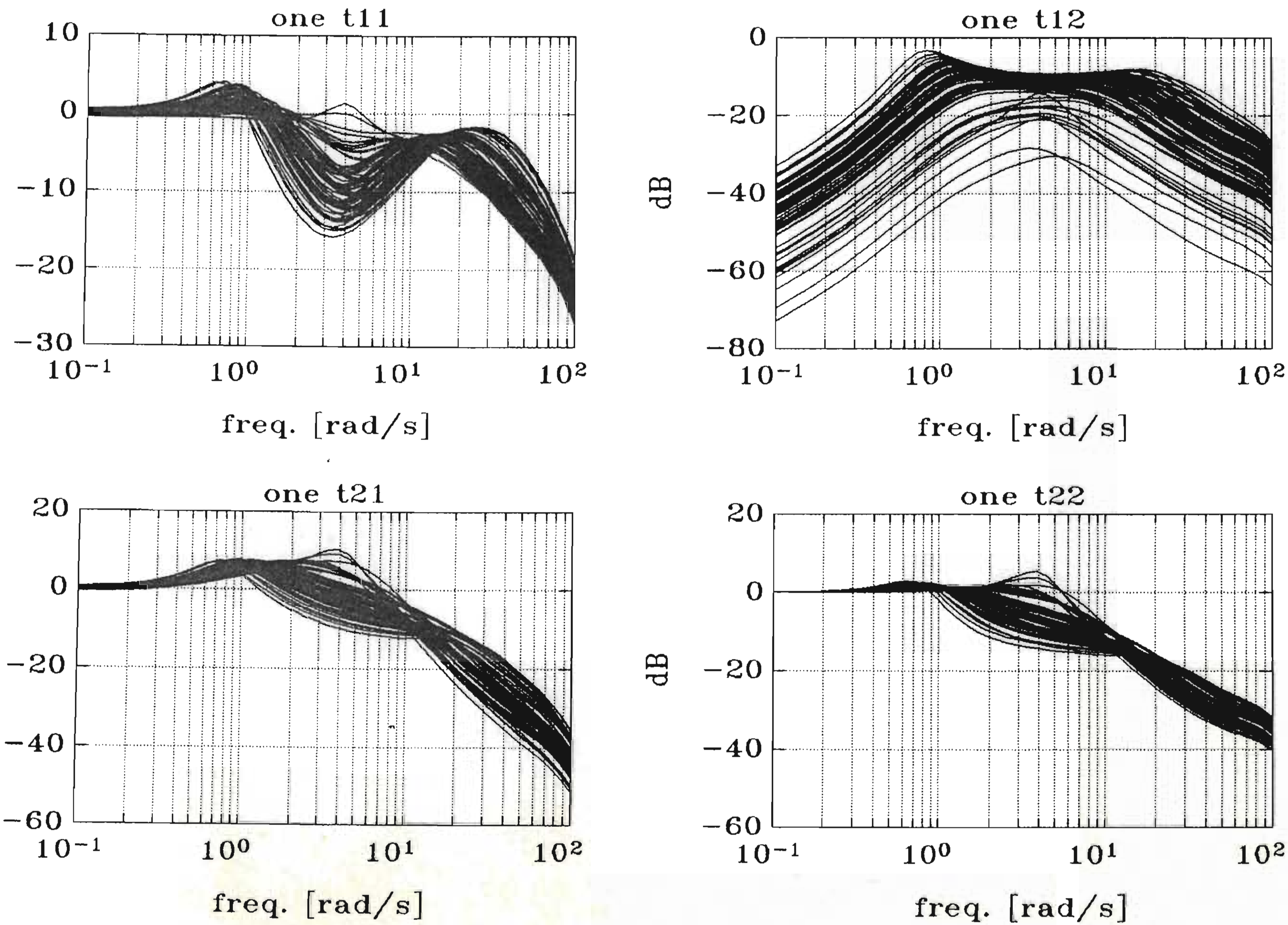


Figure 6.5 – Bode magnitude diagrams for $\underline{T}_{Y/R}$

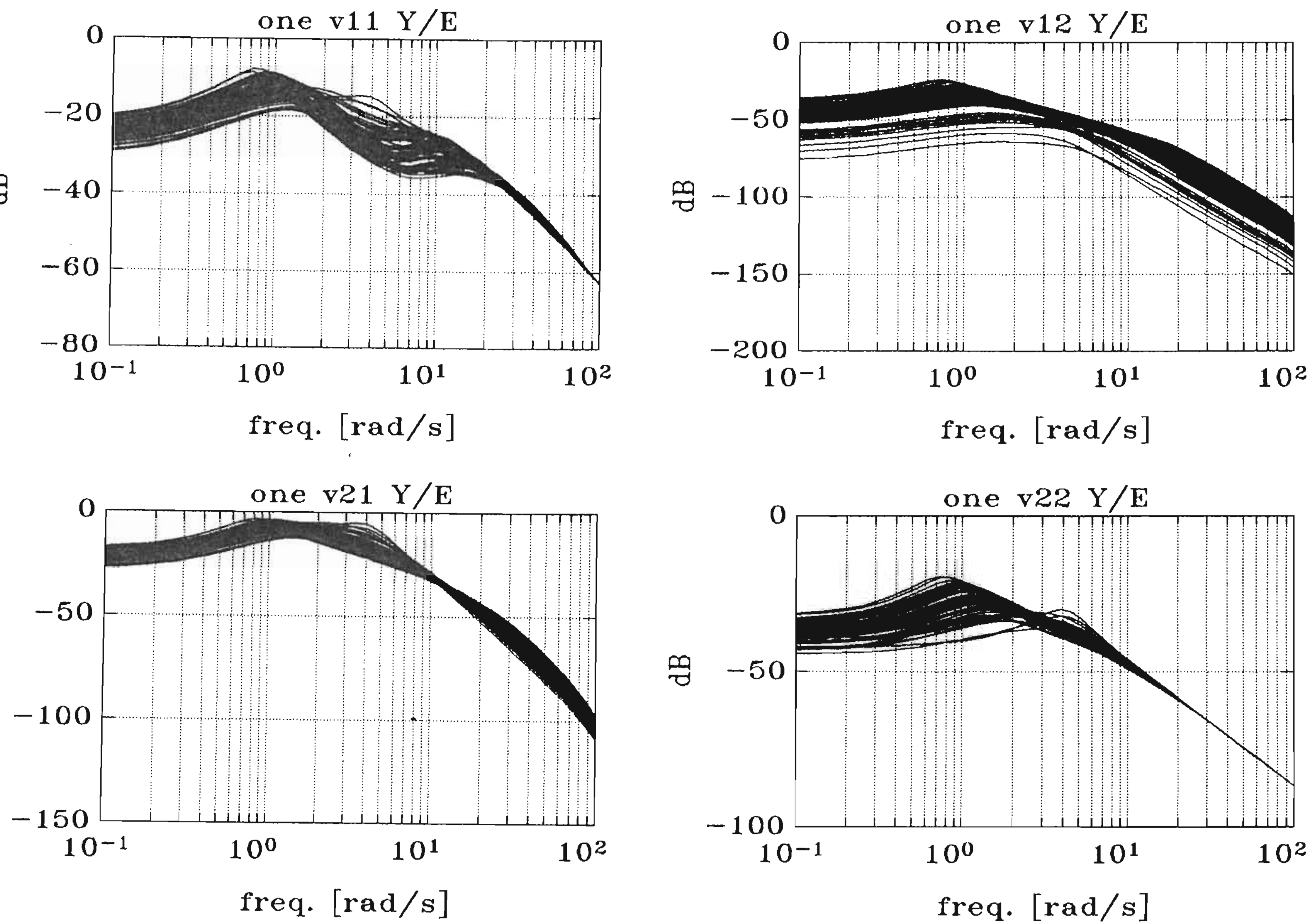


Figure 6.6 – Bode magnitude diagrams for $\underline{T}_{Y/E}$

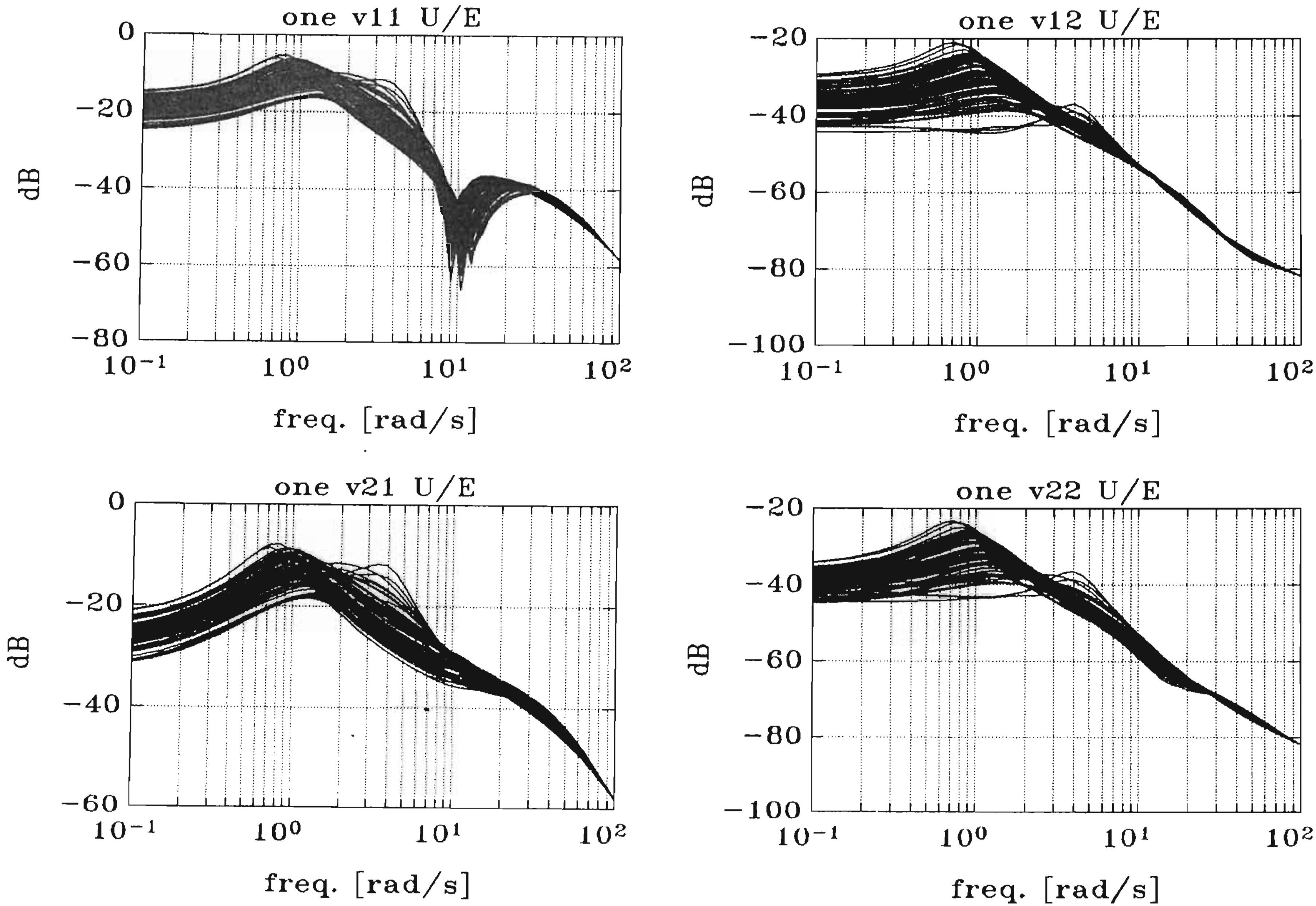


Figure 6.7 – Bode magnitude diagrams for $\underline{T}_{U/E}$

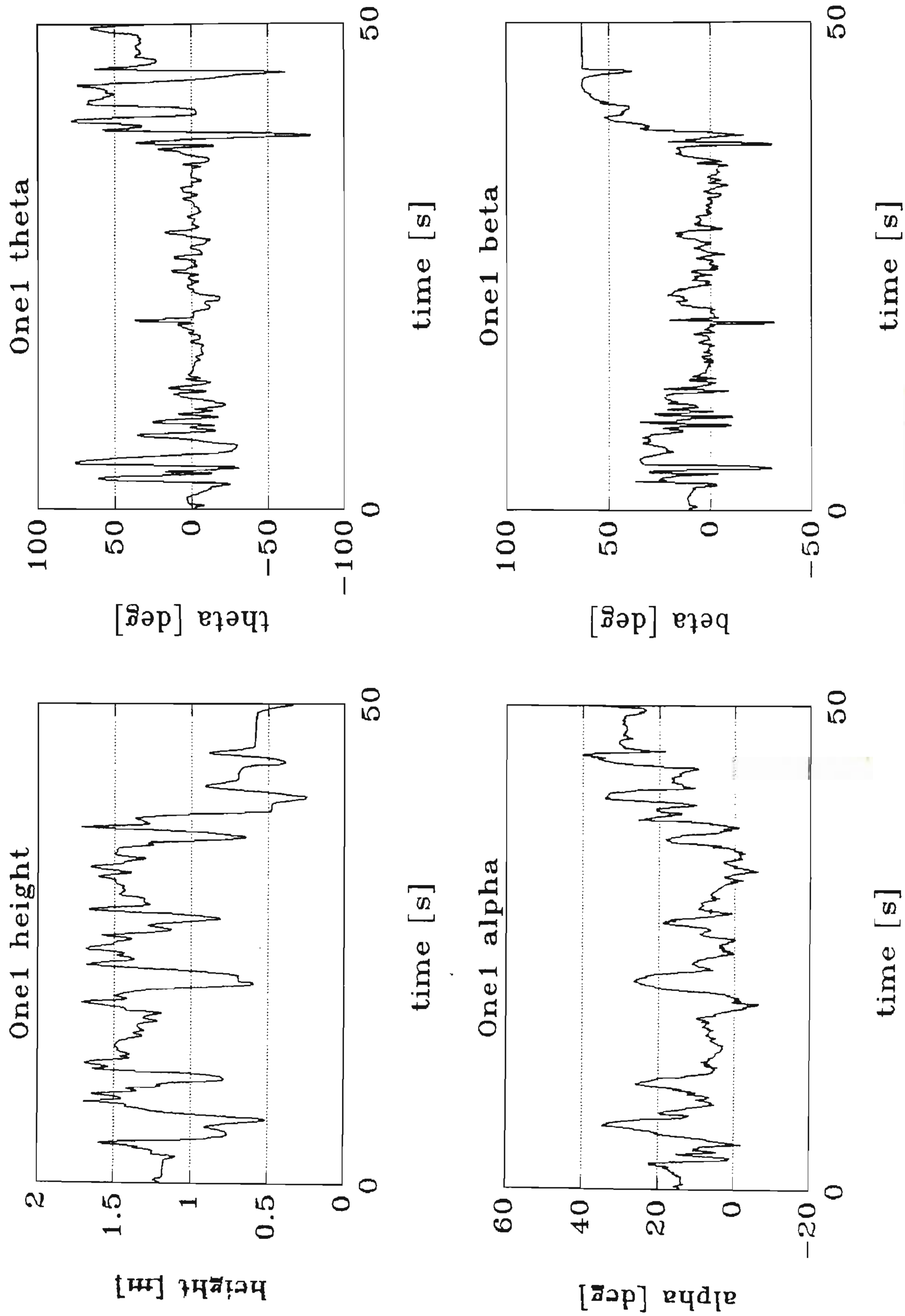


Figure 6.8a – Flight data (Test # One1) – disturbance regulation with constant height and pitch angle set–points

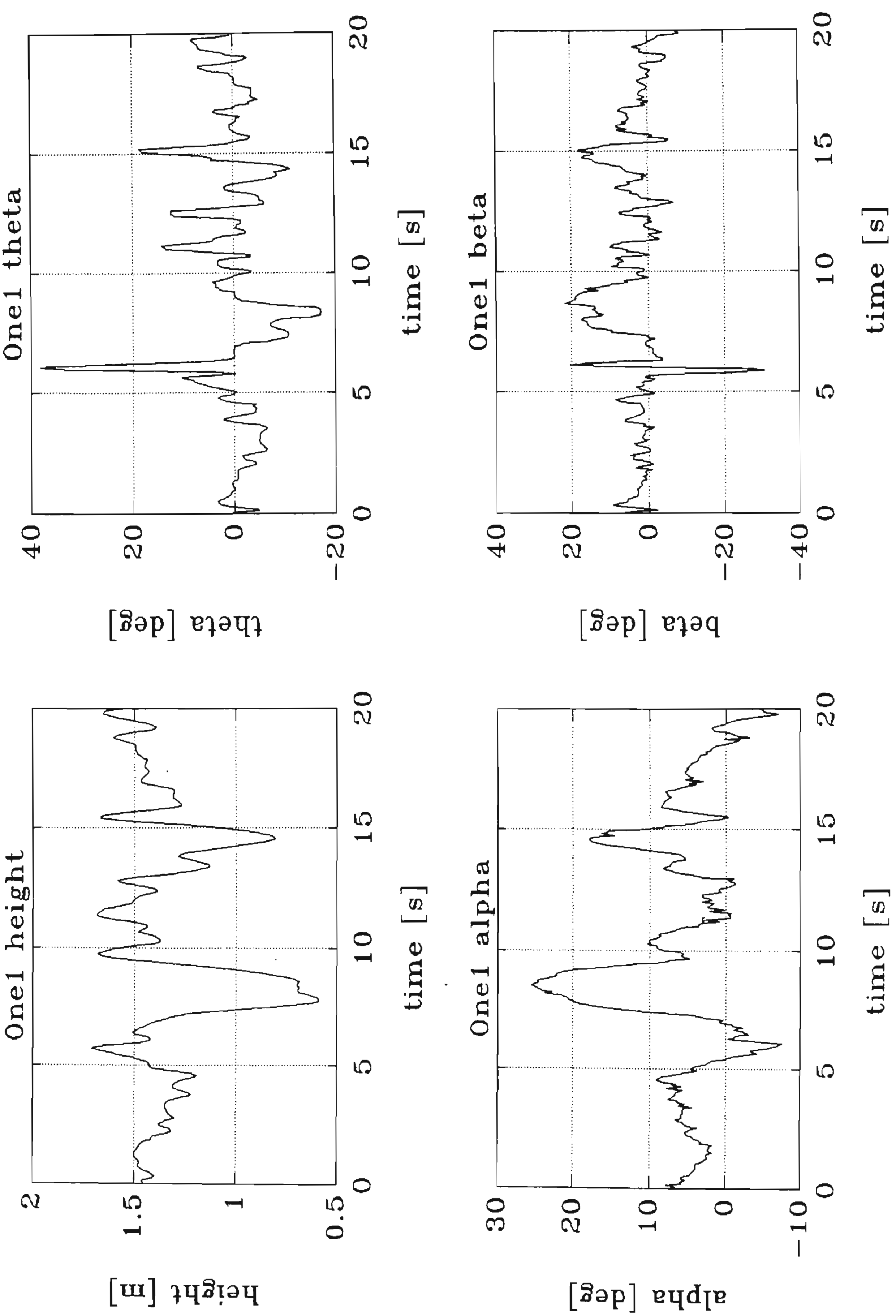


Figure 6.8b – Detail from Figure 6.8a – disturbance regulation with constant height and pitch angle set–points

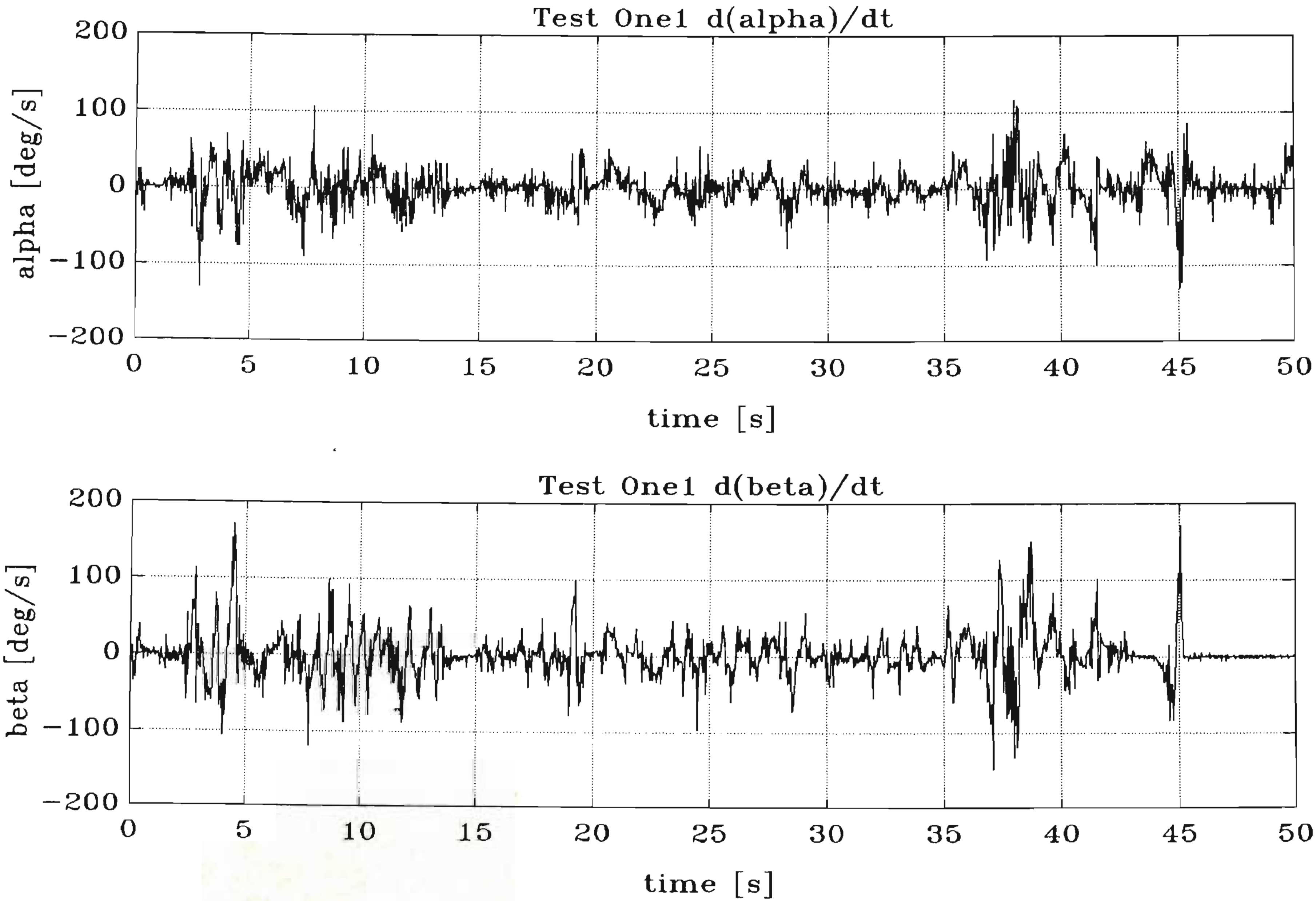


Figure 6.8c – Flight data (Test # One1) – Front and back wing rate demands

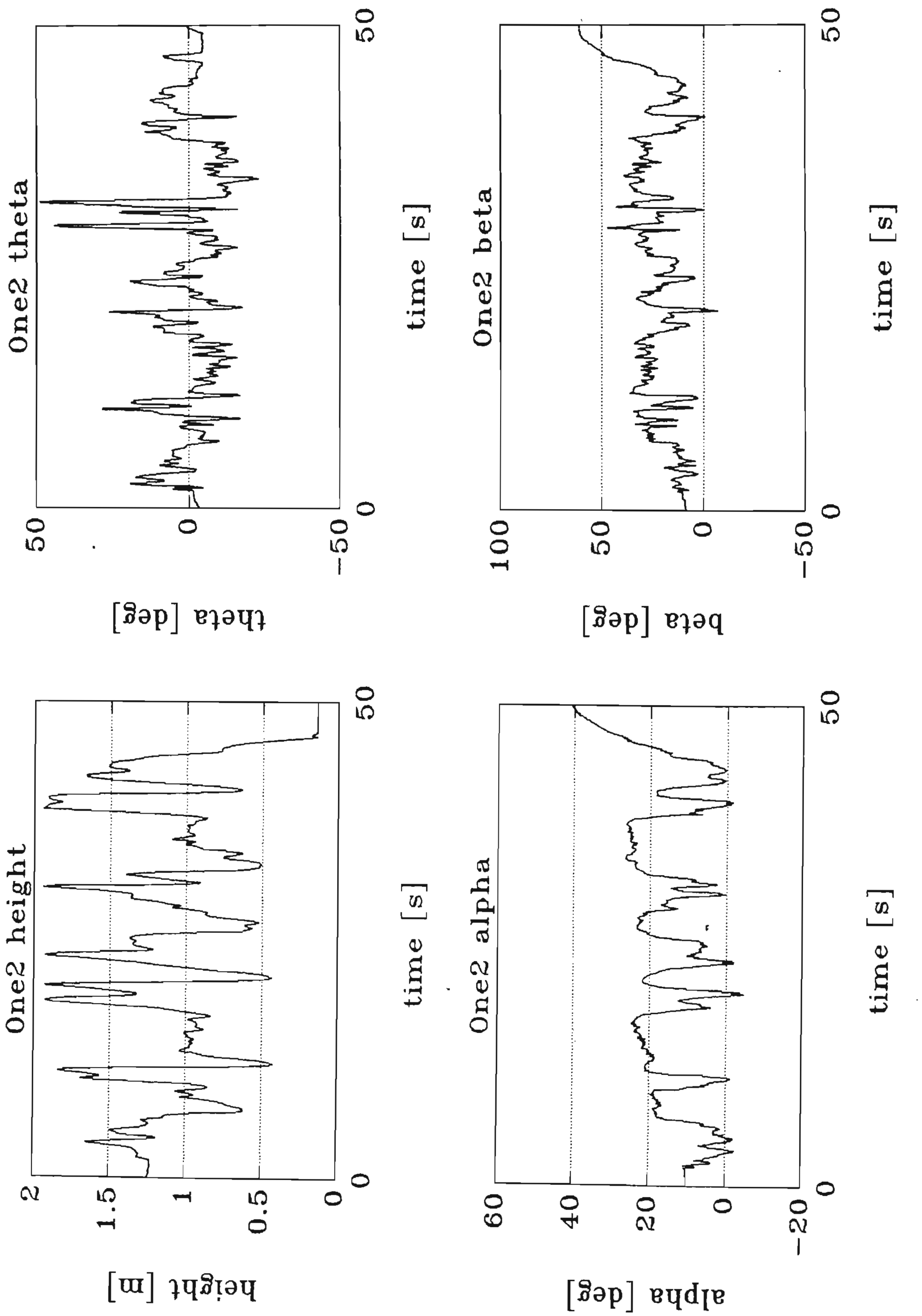


Figure 6.9 – Flight data (Test # One2) – disturbance regulation with constant height and pitch angle set–points

6.3 Conclusions

The dive in height at time t $\approx$ 7s in Figure 6.8b seemed quite characteristic of this controller.

The design apparently makes better use of the front wing than the diagonal controller of Chapter 5 did (Figure 6.8c) but does not utilize the wing turning capacity (bandwidth) as fully as the design of Chapter 4.

This rapid design highlights the advantages of classical design. Although quantitative design specifications were virtually non–existent (as they seem to be in many other practical problems), it was possible to design the single–loop controllers rapidly once the control structure had been fixed. After design of the pitch angle loop, the height loop was tuned as indicated above. Other experiments demonstrated instability when the height loop gain was increased by only 7dB (the captured data does not show this instability clearly as the flying machine was held on its leash for most of the test)

CHAPTER 7 – CONCLUSIONS

This thesis has described the development of a practical problem, a "flying machine", for testing control theory. The construction, modelling, parameter estimation and simulation of the flying machine has been described. It has been shown how plant input specifications can be included in the general QFT framework. Three controllers with different structures have been designed using QFT (with input constraints) to take parameter uncertainty into account and trade off disturbance attenuation against rate and amplitude saturation at the wing angle actuators. The controllers have been implemented and tested and the practical results compared to simulations. Defining suitable specifications during the course of a design has been seen to be very much more demanding than a direct design–to–specification problem.

This thesis has been strongly oriented towards the practical aspects of controller design. The application of "modern" control theory (LQG, singular value, H_∞ and μ synthesis) to the problem stated here would not be a trivial matter and the outcome of such an exercise is uncertain. As one of the fundamental reasons for feedback control is uncertainty reduction, modern control is in this instance best left to academics. (With the elegant linear quadratic regulator and observers etc being taught as a firm favorite in most universities' undergraduate courses, the author, as a graduate engineer attracted to control engineering, did not clearly understand the fundamental reasons for requiring feedback!) An example of how academics often are often unaware of practical problems can be found in the paper of Laughlin, Rivera and Morari (1987) in which a 19^{th} order controller which has (unrealizable) +20 dB/decade roll–up at high frequency is proposed to robustly control a plant which does not have the same uncertainties as the specified first order plant with time delay has. This robustness is optimal in an arbitrarily defined way! This is not an indictment against the modern robust control research as there are many open questions in the control of practical systems.

In the designs presented here, strictly quantitative feedback design has been violated in the following ways:

1) The plant sets used in the design are based on the linearisation of a non–linear plant model about a number of realistic operating points. Horowitz's QFT requires a "linear equivalent" plant set derived by taking the ratio of Laplace transforms of input and output signals under a number of assumptions. The linearisations result in state dependent uncertainties and the "QFT" is not rigorous with respect to such models.

2) One of the fundamental reasons for requiring feedback in the flying machine problem is uncertain wind speed and wind direction. This uncertainty is plant gain uncertainty but wind vector variations disturb the plant and should therefore be regarded as a source of disturbance – fortunately feedback helps to reduce parameter uncertainty and disturbances in the same frequency ranges via $(\underline{I}+\underline{L})^{-1}$. This duality between disturbance rejection and uncertainty reduction explains the success of linear quadratic Gaussian regulator (LQG) theory for solving problems with plant uncertainty if one is able to guess (or find by exhaustive simulation) suitable weighting matrices (or equivalent disturbance models). See for example IEEE Automatic Control Transactions, 1971, Special issue on LQG problems. (The LQG design for disturbance reduction is hoped to be able to cope with parameter uncertainty.) Horowitz has investigated this problem (1963, Section 8.15, p.441). Incidentally, separating the parameter variation and disturbances during system parameter estimation is an open research problem (Boje, 1986, p.139).

This work has possible neglected theory and rigour because of the large number of practical problems which have been encountered. It has however been shown that controller design can formally include input specifications. This and the existing tools of Horowitz's Quantitative Feedback Theory have been successfully used to design and evaluate three controller structures for a practical plant. This plant requires feedback for each of the three fundamental reasons for having that feedback, stability, uncertainty reduction and disturbance rejection.

REFERENCES

ATHERTON D P *Nonlinear Control Engineering*, Van Nostrand, Reinhold Company, New York, 1982, ISBN 0–442–30486–2.

BERTIN J J and SMITH M L *Aerodynamics for engineers*, Prentice–Hall, Englewood Cliffs, 1979, ISBN 0–113–018234–6.

BOJE E *Linear model identification for multivariable continuous time systems*, MSc Eng Thesis, University of Natal, Durban, South Africa, July 1986.

BOJE E "Linear model identification for multivariable continuous time systems" *8th IFAC/IFORS Symposium on Identification and System Parameter Estimation*, Beijing PR China, August 1988, pp. 633–638

BOJE E "Multivariable quantitative feedback design with plant input specifications", Proceedings of 1989 IEEE International Conference on control and Applications, Jerusalem, Israel, 3–6 April 1989

BOJE E and EITELBERG E "On non–minimum phase behaviour in d.c. servo systems", Submitted to: *Transactions of South African IEE* for Special issue on Power Electronics and Electric Drives, 1989(?)

COOPER G R and McGILLEM C D *Probabilistic methods of signal and system analysis*, Holt, Reinhart and Winston, New York, 1971, ISBN 0–03–0084291–3.

D'SOUZA A F and GARG V K Advanced dynamics – Modeling and Analysis, Prentice–Hall, Englewood Cliffs, New Jersey, 1984, ISBN 0–13–011312–3

DOYLE J C "Quantitative Feedback Theory (QFT) and Robust Control", *Proceedings of American Control Conference*, pp.1691–1698, 1986.

DOYLE J C and STEIN G "Multivariable feedback design: Concepts for a classical/modern synthesis" *IEEE Transactions*, AC–26, pp.4–16, 1981.

EITELBERG E "Modular simulation of large stiff systems", *Progress in Modelling and Simulation* (ed. CELLIER F E), Academic Press, London, 1982.

EITELBERG E "A simple A_2–stable numerical method for state space models with stiff oscilllations", Mathematics and Computers in Simulation, XXV, 1983, pp.346–355, North Holland.

EITELBERG E "Continuous–time system representation with exact macro–difference expressions", *International Journal of Control*, Vol.47, No.5, pp.1207–1212, 1988.

EITELBERG E and BOJE E "Some practical low frequency boundds in quantitative feedback degign" IEEE International conference on control and applications, Jerusalem, 3–6 April 1989, Paper WP–2–1

FOWLES G R *Analytical mechanics* (Third Edition), Holt, Rinehart and Winston, New York, 1977, ISBN 0–03–089725–4.

GOLUBEV B *Nonlinear Feedback Systems with Large Uncertainty*, PhD Thesis, Weizmann Institute of Science, Rehovot, Israel, 1983.

HOROWITZ I *Synthesis of Feedback Systems*, Academic Press, New York, 1963, Library of Congress Catalog No 63–12033.

HOROWITZ I "Quantitative synthesis of uncertain multiple input–output feedback systems", *International Journal of Control*, Vol.30, No.1, pp.81–106, 1979.

HOROWITZ I "Improved design technique for uncertain multiple–input–multiple–output feedback systems", *International Journal of Control*, Vol.36, No.6, pp.977–988, 1982a.

HOROWITZ I "Quantitative feedback theory", *IEE Proceedings*, Vol. 126, Part D, No. 2, November 1982(b).

HOROWITZ I "Quantitative Feedback Theory – Reply to Doyle's Criticisms" *Proceedings of American Control Conference*, pp.622–627, Minneapolis, MN June 10–12, 1987.

HOROWITZ I M and SIDI M "Synthesis of feedback systems with large plant ignorance for prescribed time–domain tolerances", *International Journal of Control*, Vol.16, No.2, pp.287–309, 1972.

HOROWITZ I M, OLDAK S, and YANIV O "An important property of non–minimum phase multiple–input–multiple–output feedback systems", *International Journal of Control*, Vol.44, No.3, pp.677–688, 1986.

HOROWITZ I M and LIAO Y K "Quantitative non–linear compensation for saturating unstable uncertain plants", *International Journal of Control*, Vol.44, No.4, pp.1137–1146, 1986.

IEEE Transactions on Automatic Control, *Special issue on LQG Problems*, December 1971.

KAILATH T *Linear Systems*, Prentice Hall, Englewood, Cliffs, 1980, ISBN 0–13–536961–49.

KING K N *TopSpeed Modula–2 Language Tutorial*, Jensen and Partners International, 1988

LAUGHLIN D L, RIVERA D E and MORARI M "Smith predictor for robust control" *International Journal of Control*, Vol.46, No.2, pp.477–504, 1987.

NWOKAH O D I "On nonsingular value based design of controllers for robust stability" *IEE Proceedings*, Vol. 133, Part D, No. 2, March 1986.

TZAFESTAS S G (Editor) *Multivariable Control – New Concepts and Tools* D Reidel, Dordrecht, Holt, 1984, ISBN 90–277–1829–6

WIRTH N Programming in Modula–2, Second Edition, Springer–Verlag, Berlin, 1983

Appendix 1 – Circuit diagrams for actuators and controllers

A1.1 Actuators

The actuators were powered using the push–pull amplifier arrangement shown in Figure A1.1. The use of an operational amplifier to supply the transistor base drive is possibly crude but is effective. The diode circuit on the reference input is to clip demand signals which are in excess of the position saturation limits of the actuators (this protects the actuator from large currents by preventing position saturation). The gain of the inner loop was adjusted to get the fastest response possible without excessive overshoot.

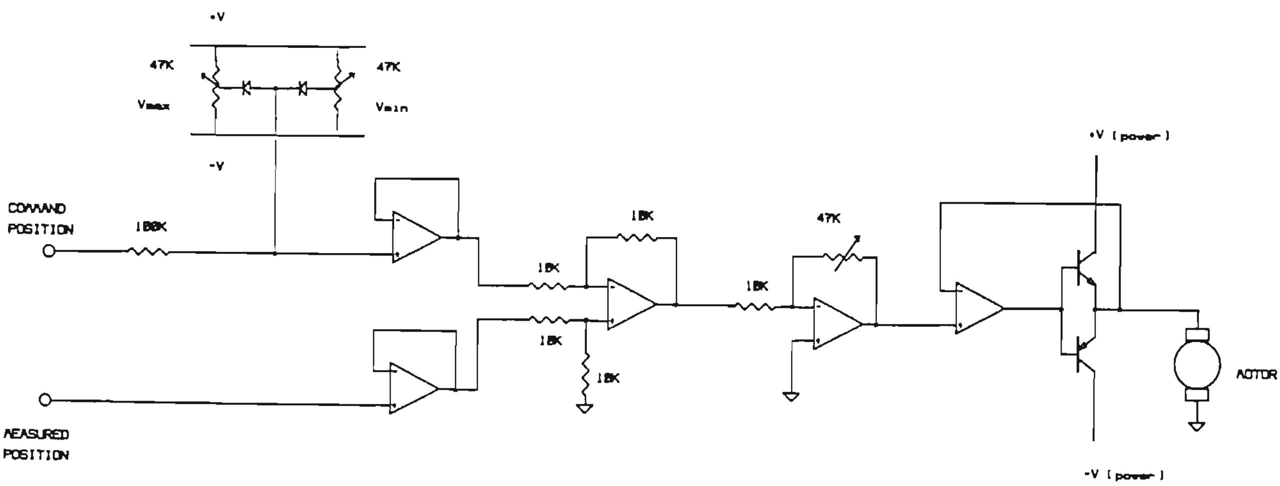
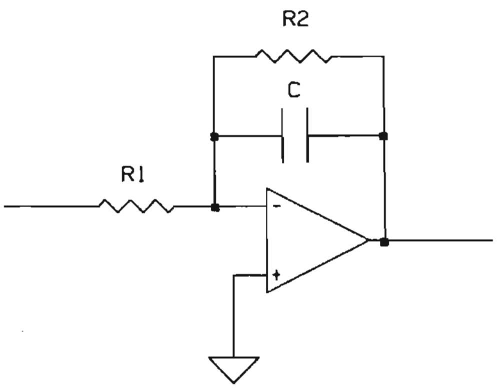


Figure A1.1 – Wing actuator drive electronics

A1.2 Controller implementation

The controllers were all constructed as active analog filters, as a series of first and second order elements. LM–741 operational amplifiers (or their equivalent) were used throughout. Figure A1.2 show first order low–pass, first order lead/lag and second order (with first order numerator) elements. Component values for the second order sections of controllers in Chapters 4, 5 and 6 are,



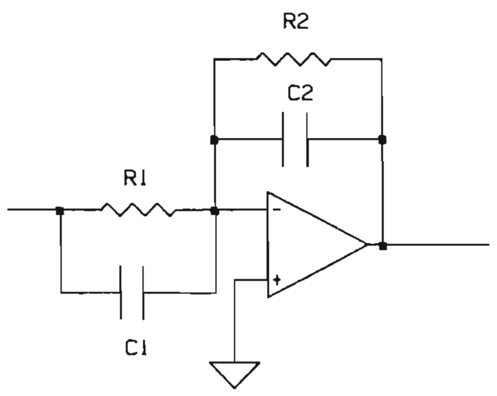


Figure A1.2 First order low pass element **Figure A1.3** First order lead/lag element

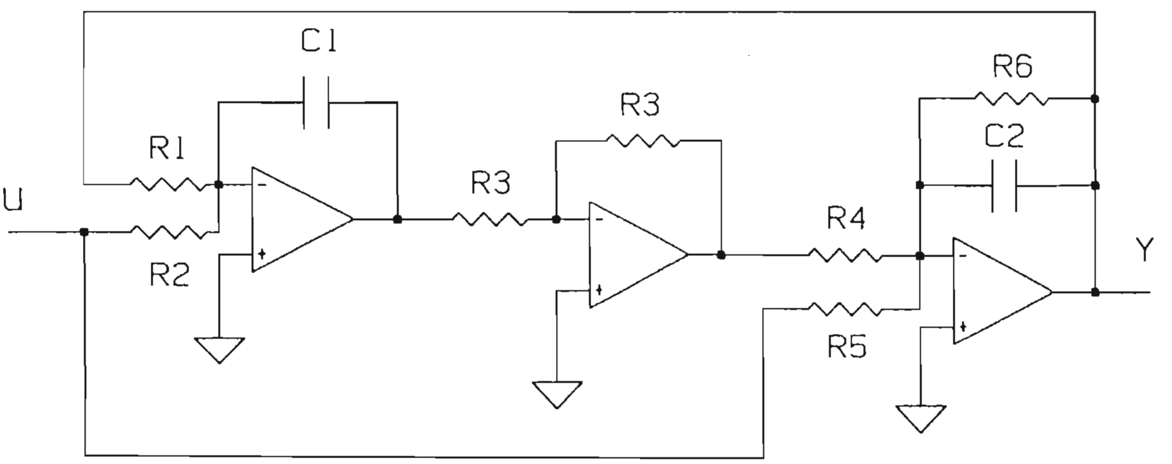


Figure A1.4 Second order strictly proper element

Decoupled controller

g_θ: r1=100kΩ; r2=100kΩ; r3=10kΩ; r4=10kΩ; r5=10kΩ; r6=3.9kΩ; r7=22kΩ; c1=0.33μF; c2=0.33μF

g_H: r1=120kΩ; r2=120kΩ; r3=10kΩ; r4=10kΩ; r5=33kΩ; r6=15kΩ; r7=47kΩ; c1=0.33μF; c2=0.33μF

Quasi–decoupling controller

g_θ: r1=85kΩ; r2=22kΩ; r3=10kΩ; r4=10kΩ; r5=2.2kΩ; r6=470Ω; r7=10kΩ; c1=1μF; c2=1μF

g_H: r1=47kΩ; r2=47kΩ; r3=10kΩ; r4=10kΩ; r5=82kΩ; r6=12kΩ; r7=15kΩ; c1=1μF; c2=1μF

Diagonal controller

g_θ: r1=27kΩ; r2=27kΩ; r3=10kΩ; r4=10kΩ; r5=1.8kΩ; r6=1.5kΩ; r7=6.8kΩ; c1=1μF; c2=1μF

g_θ: r1=330kΩ; r2=330kΩ; r3=10kΩ; r4=10kΩ; r5=330Ω; r6=330kΩ; r7=4.7kΩ; c1=1μF; c2=1μF

Appendix 2 – On non-minimum phase behaviour in d.c. servo systems

This appendix contains the un-refereed paper bearing the above title, which has been submitted to the South African Institute of Electrical Engineers *Transactions*.

(Note: The above paper has since been published in *SAIEE Transactions, Special issue on power electronics and electric drives*, Vol 80, No 1, September 1989)

ON NON–MINIMUM PHASE BEHAVIOUR IN D.C.SERVO SYSTEMS

Edward Boje
Eduard Eitelberg

stract

eraction between the power circuit and the analog measurement and control circuits ich can result in unsatisfactory, non–minimum phase behaviour is examined in low power ect current (d.c.) servo systems. Understanding the cause of this non–minimum phase aviour allows the limitations resulting from it either to be avoided or to be included orously in the system design.

ntroduction

amiliar characteristic of non–minimum phase systems with one right hand plane zero is t the step response is initially away from the final value (this holds for all stable systems h an odd number of right hand plane zeros as can be seen by applying the initial and final ue theorems). It has long been known (at least to "classical" and applied control ineers) that there are fundamental limitations on the performance achievable by (linear) lback if the plant to be controlled has one or more right hand plane zeros (or dead time) rowitz, 1963, [1]). A rule of thumb which is sometimes used in systems with one right d zero (at $s = z$), is that the loop gain crossover frequency ω_c should be less than half the t hand plane zero's cut–off frequency ($\omega_c \leq z/2$). (Horowitz and Liao (1984, [2]) give e results on performance which can be achieved outside this usual frequency range for –minimum phase systems and give further references.) In order to obtain high ormance (with respect to disturbance reduction, command following and model rtainty reduction) from a servo system, high loop bandwidth is required. This lwidth cannot always be achieved in non–minimum phase systems because the right half e zeros contribute phase lag (as opposed to a left half plane zero's lead) and gain to the transfer function. As economy limits the design of most systems, there is a limit on how l" components can be. The engineering objective in servo system design is to achieve fications on performance with the least over–design.

paper considers the application of low power (say less than 10 Watts) d.c. motors for ion or velocity control, where careless design can result in unsatisfactory and possibly pected behaviour as a result of right hand plane zeros in the open loop system. The action between the power circuit driving the motor (or any other actuator) and the g feedback circuit (the sensors and controller) is often the culprit. This note examines o simple examples of such interaction to give an understanding of the mechanism by iich non–minimum phase behaviour can appear.

odels are developed for the two cases examined, namely ;
the use of common conductors for sensors and motor power circuits (linear example – ction 2) and,
the use of a power supply with internal resistance (non–linear example – Section 3).

e examples illustrate the need for separating the measurment and feedback circuits from e power circuit. Once this is understood, obvious design measures will avoid most oblems.

Common conductors for measurement and power signals – linear example

e circuit shown in Figure 1 is of a small servo system where the controller and actuator cuits are separated by some distance and a single "ground" conductor has been used to e weight or cost. The design problem in Figure 1 is that the motor armature current, i_A, ects the measured position, θ_{meas}, via resistance of the "ground" conductor, R_g. The use a single wire for "ground" of both signal and power circuits is not recommended pecially in higher power applications where safety and equipment protection against ervoltage must be considered) but the control system designer may not be in a position to er a plant (and wiring) which has been designed to meet other requirements.

del

symbols in Figure 1 have the following meaning,

ameters:

J – the total inertia of the motor and load (referred to the motor)

μ – the total rotational friction of the motor and load

R_g – the resistance of the common "ground" wire

R – the resistance of the motor armature and power circuit, including the "ground" resistance

L – the armature inductance

als (in the Laplace domain) :

D(s) (= $\mathscr{L}\{d(t)\}$) – disturbance torque

V(s) (= $\mathscr{L}\{v(t)\}$) – applied voltage

E(s) – back e.m.f. of the motor

$I_A(s)$ – armature current

$\theta(s)$ – the actual shaft position (not measured explicitly)

$\theta_{meas}(s)$ – the measured (feedback) position signal

(Note that without disturbance, D(s) and/or parameter uncertainty there is no requirement for using a feedback controller at all.)

Assumptions for a simple linear model are,

$$
\begin{aligned}
E(s) &= k_{emf}\, s\, \theta(s) \qquad \text{(back e.m.f. proportional to speed of rotation)} \\
&= V(s) - (L\,s + R)\, I_A(s)
\end{aligned} \tag{1}
$$

$$
\begin{aligned}
T(s) &= k_T\, I_A(s) \qquad \text{(torque proportional to armature current)} \\
&= (J\,s^2 + \mu\,s)\,\theta(s) + D(s)
\end{aligned} \tag{2}
$$

From eqs (1) and (2), the voltage across the motor as a function of position and disturbance is,

$$
V(s) = \left[\frac{(Ls + R)(Js + \mu)}{k_T} + k_{emf}\right] s\,\theta(s) + \frac{(Ls + R)}{k_T} D(s) \tag{3}
$$

When L, R, J, μ are small with respect to k_T, eq(3) simplifies (as expected) to

$$
V(s) = k_{emf}\, s\, \theta(s) \qquad \text{(motor speed proportional to applied voltage)} \tag{4}
$$

for all frequencies ($s = j\omega$) of interest, and a controller with gain alone (no dynamic elements) may provide the specified (desired) closed loop behaviour.

If the position is scaled from a measured voltage by means of the circuit shown in Figure 1, then a component of the measured position is determined by $R_g\, I_A(s)$, say,

$$
\begin{aligned}
\theta_{meas}(s) &= \theta(s) + k_{meas} I_A(s) \\
&= \left[\frac{k_{meas}}{k_T}(J\,s^2 + \mu\,s) + 1\right]\theta(s) + \frac{k_{meas}}{k_T} D(s)
\end{aligned} \tag{5}
$$

The behaviour of the circuit depends on the sign of k_{meas}/k_T. The sign of k_{meas} is determined by the sign of the position potentiometer supply and the sign of k_T depends on the electrical connection of the motor. (In Figure 1, k_{meas}/k_T has the same sign as the position measuring potentiometer supply if the (positive) current i_A results in increasing $\theta(t)$.) The potentiometer resistance is reasonably assumed >> R_g.

torising the position ($\theta(s)$) component of eq(5) alone,

$$\theta_{meas}(s) = (s/a + 1)\,(s/b + 1)\,\theta(s) \tag{6}$$

h,

$$a, b = \frac{\mu \pm \sqrt{\mu^2 - 4\,J\,k_T\,/\,k_{meas}}}{2\,J}$$

single "earth" therefore affects the controller design as follows:

If k_{meas}/k_T is negative, the measurement of θ results in the open loop transfer racteristic having a right hand plane zero (situated on the real axis) with the esponding limitations on achievable closed loop behaviour.

If k_{meas}/k_T is positive, then one might be tempted to use the numerator terms oduced into the loop during controller design so as to obtain phase lead without requiring amic elements in the controller itself. The behaviour of such an arrangement will usually nacceptable for the following reasons:
oise in the measurement and current signals will be amplified in an undisciplined manner igh frequency (especially at the motor input)
he location of the zeros is not certain as a result of parameter (J, k_T, k_{meas} and μ) ertainty
ne zeros will be poorly damped if $4\,J\,k_T\,/\,k_{meas} >> 0$.

ılts

hs 1a and 1b show "before" and "after" results for closed–loop step response in a tical problem when the controller was simply a gain element. Note that these graphs measured position – the open loop plant does not exhibit non–minimum phase viour, only the measurement. Before separation of common conductors, the motor ng circuit saturated with all but small command signals.

wer supply with internal resistance – non–linear example

e 2 shows a common problem in which the servo system's power supply has internal lance. For simplicity only power supply resistance, R_s, is considered here. To stand the problem, it is assumed that only the measurement (of speed, position, etc) is ed by the power supply voltage fluctuations although in practice the controller circuit vulnerable to power supply variations.

quations are developed as before, except in the time domain:

$$T(t) = k_T\, i_A(t) = J\,\frac{d^2\theta(t)}{dt^2} + \mu\,\frac{d\theta(t)}{dt} + d(t) \tag{7}$$

(torque proportional to armature current)

The terminal voltage of the supply is given by,

$$v_o(t) = 2\,V_s - R_s(i_n(t) + i_p(t)) \tag{8}$$

The voltage with respect to 0V, measured by the position sensing circuit is,

$$v_\theta(t) = \frac{R_\theta}{R_m}\,v_o(t) - (V_s - i_n(t)\,R_s)$$

$$= \frac{1}{R_m}\Big[\,2\,V_s\,(R_\theta(t) - R_m/2) + R_s\,[(R_m - R_\theta(t))\,i_n(t) - R_\theta(t)\,i_p(t)]\Big] \tag{9}$$

Without supply resistance, the position would be scaled as,

$$\theta(t) = k_\theta\,2\,V_s\,\frac{R_\theta(t) - R_m/2}{R_m} \tag{10}$$

(if $R_\theta = R_m/2$ corresponds to $\theta = 0$). The measured position is,

$$\theta_{meas}(t) = k_\theta\,v_\theta(t) \tag{11}$$

If the current drawn by the control and measurement circuits are small when compared to i_A, then current only flows in one leg of the supply,

Case 1 : $i_A > 0$, $i_p \approx i_A$ and $i_n \approx 0$

$$\theta_{meas}(t) = \theta(t) - k_\theta\,\frac{R_s\;R_\theta(t)}{R_m}\,i_A(t) \tag{12}$$

Case 2 : $i_A < 0$, $i_p \approx 0$ and $i_n \approx -i_A$

$$\theta_{meas}(t) = \theta(t) - k_\theta\,\frac{R_s\;(R_m - R_\theta(t))}{R_m}\,i_A(t) \tag{13}$$

Unlike the previous example, this case is non–linear as both $R_\theta(t)$ and $i_A(t)$ are functions of $\theta(t)$. The conditions for a non–minimum phase term which can be seen by finding the small signal linearisation (denoted $\Delta\theta(t)$) of eqs (12) and (13) with respect to a steady state operating condition (denoted subscript *). Eq (12) becomes,

$$\Delta\theta_{meas}(t) = \Delta\theta - \frac{k_\theta\ R_s}{k_T\ R_m}\left[R_{\theta*}\left[J\,\frac{d^2\Delta\theta(t)}{dt^2} + \mu\,\frac{d\Delta\theta(t)}{dt}\right] + \Delta R_\theta(t)\left[J\ \frac{d^2\theta(t)}{dt^2} + \mu\ \frac{d\theta(t)}{dt}\right]\Bigg|_*\ \right]$$

(14)

he system moves around rest, $\left.\frac{d^2\theta}{dt^2}\right|_* = \left.\frac{d\theta}{dt}\right|_* = 0$ and the small signal transfer function n angle to measured angle is,

$$\Delta\theta_{meas}(s) = \left[1 - \frac{k_\theta\ R_s\ \ R_{\theta*}}{k_T\ R_m}\left[Js^2 + \mu\,s\right]\right]\Delta\theta(s) \qquad (15)$$

>0)

negative $i_A(t)$ the procedure is the same and results in the small signal transfer function,

$$\Delta\theta_{meas}(s) = \left[1 - \frac{k_\theta\ R_s\ \ (R_m - R_{\theta*})}{k_T\ R_m}\left[Js^2 + \mu\,s\right]\right]\Delta\theta(s) \qquad (16)$$

zeros of eqs (15) and (16) lie on the real axis, one in the left and one in the right half e. In the Appendix it is shown that if the power supply to the position measuring ntiometer may be reversed, the resulting behaviour is minimum phase behaviour around est position.

lts

hs 2a, 2b and 2c show closed–loop results from a laboratory experiment set up to rate the above results – the behaviour had been observed in practical problems but no was available for presentation. The controller used was a simple gain element. To ve the results, $R_s = 1\Omega$ resistors were placed in series with a low impedance power y. The rating of the motor was 12V, 250mA and the armature resistance was 27Ω.

h 2a shows the non–minimum phase behaviour predicted by eqs (15) and (16). Note the ent magnitudes of positive and negative edged step responses, predicted by eqs (15) and

1 2b shows the effect of changing the sign of the position measuring circuit (eqs (A4) A5)). Although the measurement the start of the step response is minimum phase, this shows non–minimum phase measurement around some non–steady state condition can be interpreted using eq (14).

Graph 2c shows the effect of separating the power and measurement circuits using the technique suggested below, without removing the 1Ω power supply resistors.

Reducing or eliminating the non–minimum phase problem

The non–minimum phase behaviour caused by interaction between measurement and motor current can be substantially reduced by a simple isolation circuit such as that shown in Figure 3. Typically power supply isolation is also required for the feedback circuits – this latter problem can be analysed in a similar manner to the measurement one above.

As the expense of complete power supply separation may not be warranted in lower performance systems, note that it is not imperative to remove power supply dependence in the sensor/control circuits. It is only required to place the non–minimum phase zero at such high frequency that either it has no effect or the design can cope with the extra phase lag and gain introduced.

4) Conclusion

The large loop bandwidths required in high performance servo systems, can only be achieved if the open loop transfer function is minimum phase (at least in the frequency range where the loop transmission will be larger than unity). This note has shown that the power supply to the sensors should be isolated from that to the motor armature if problems are to be avoided. Similar analysis would show that in general the sensors and control circuits should be as insensitive as possible to supply fluctuations for practical systems to work as expected. The formulas given here can be used to quantitatively describe the feedback problem if for some reason the non–minimum phase behaviour cannot be avoided.

5) Acknowledgements

The experimental apparatus used to generate the results of Section 3 was constructed by Mr S.K. Moodley who also produced the schematic/circuit diagrams. The first author acknowledges the financial assistance of the Council for Scientific and Industrial Research.

6) References

[1] HOROWITZ, I. M. – *Synthesis of Feedback Systems*, 1963, Academic Press, New York

[2] HOROWITZ, I. M. and LIAO, Y. K. – "Limitations of non–minimum–phase feedback systems", *International Journal of Control*, 1984, Vol 40, No 5, pp 1003–1013

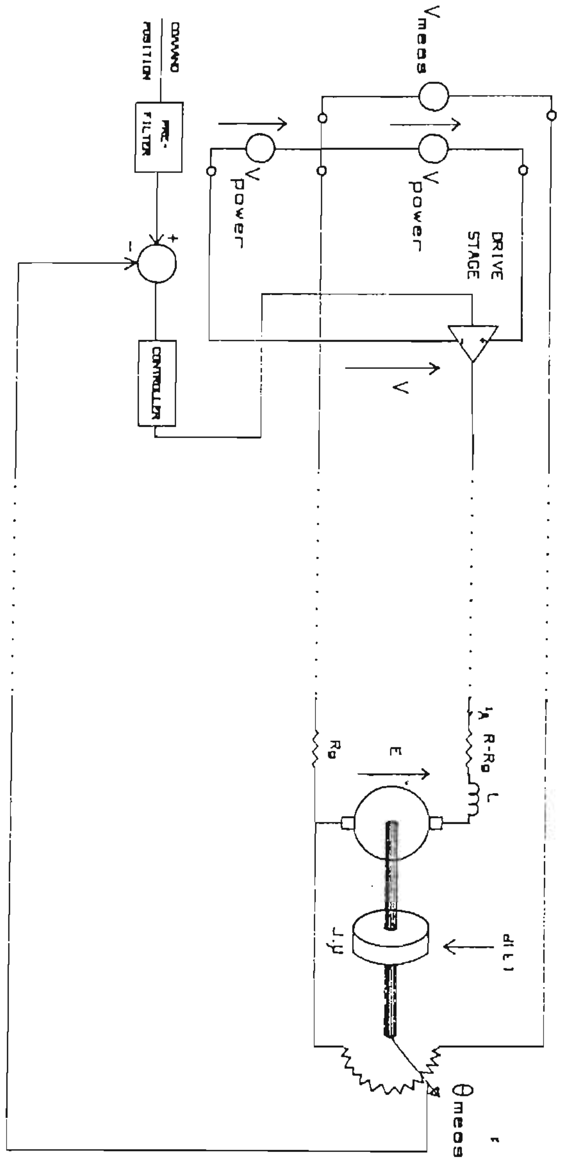


Figure 1-Linear problem with common ground controller

7) Authors

E Boje – Department of Electrical Engineering, University of Durban–Westville, Private Bag X54001, Durban 4000.
E Eitelberg – Department of Electrical Engineering, University of Durban–Westville.

Appendix – Minimum phase case with supply resistance

The non–linear example (with power supply resistance) is considered for the case when the terminals of the position sensing circuit are swapped. The voltage with respect to 0V, measured by the position sensing circuit is then,

$$v_\theta(t) = -\frac{R_\theta}{R_m} v_o(t) + (V_s - i_p(t)\, R_s)$$

$$= -\frac{1}{R_m}\Big[\, 2\, V_s\, (R_\theta(t) - R_m/2) + R_s\, [(R_m - R_\theta(t))\, i_p(t) - R_\theta(t)\, i_n(t)]\Big] \qquad \text{(A1)}$$

As before, without supply resistance the position would be scaled as,

$$\theta(t) = k_\theta\, 2\, V_s \frac{R_\theta(t) - R_m/2}{R_m} \qquad \text{(A2)}$$

(if $R_\theta = R_m/2$ corresponds to $\theta = 0$). The measured position is,

$$\theta_{meas}(t) = -\, k_\theta\, v_\theta(t) \qquad \text{(A3)}$$

If the current drawn by the control and measurement circuits are small when compared to i_A, then current only flows in one leg of the supply,

Case 1 : $i_A > 0$, $i_p \approx i_A$ and $i_n \approx 0$

$$\theta_{meas}(t) = \theta(t) + k_\theta \frac{R_s\, (R_m - R_\theta(t))}{R_m}\, i_A(t) \qquad \text{(A4)}$$

Case 2 : $i_A < 0$, $i_p \approx 0$ and $i_n \approx -\, i_A$

$$\theta_{meas}(t) = \theta(t) + k_\theta \frac{R_s\, R_\theta(t)}{R_m}\, i_A(t) \qquad \text{(A5)}$$

Linearisation of eqs (A4) and (A5) explains the minimum phase behaviour around rest, observed in the results.

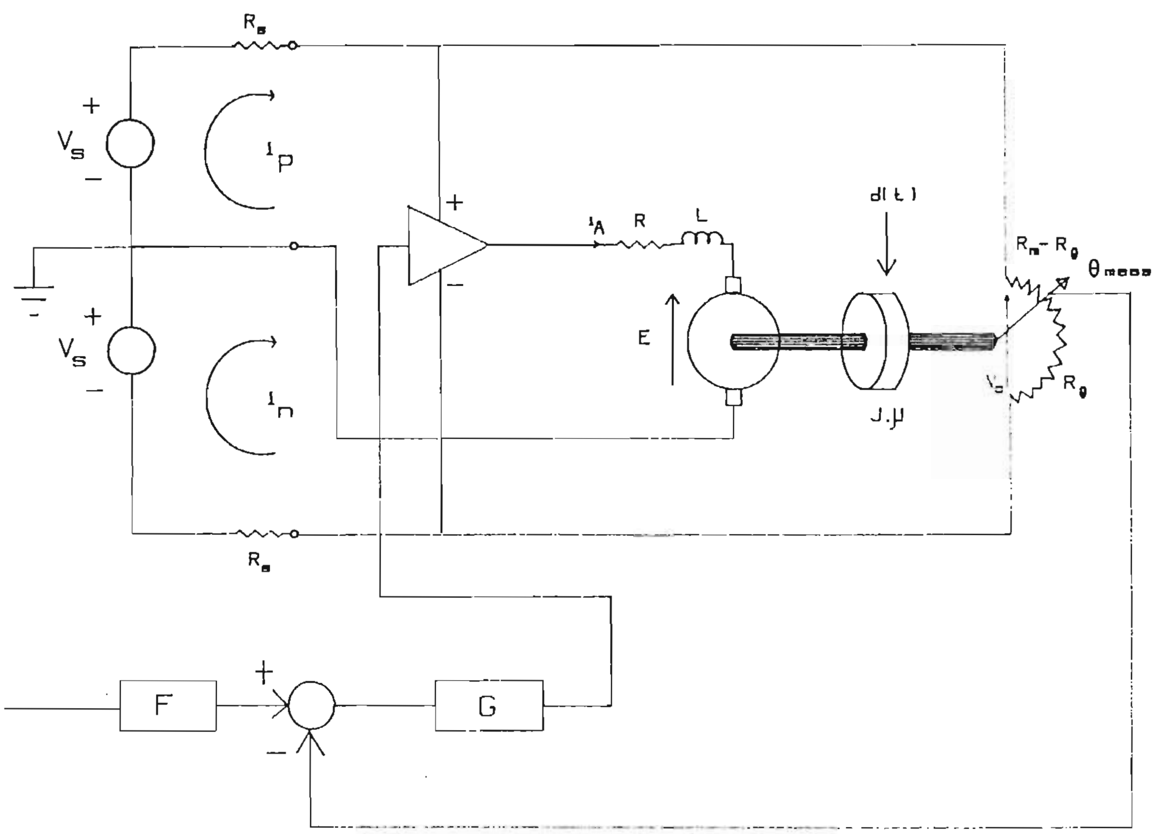


Figure 2-Non linear problem-
power supply dependence

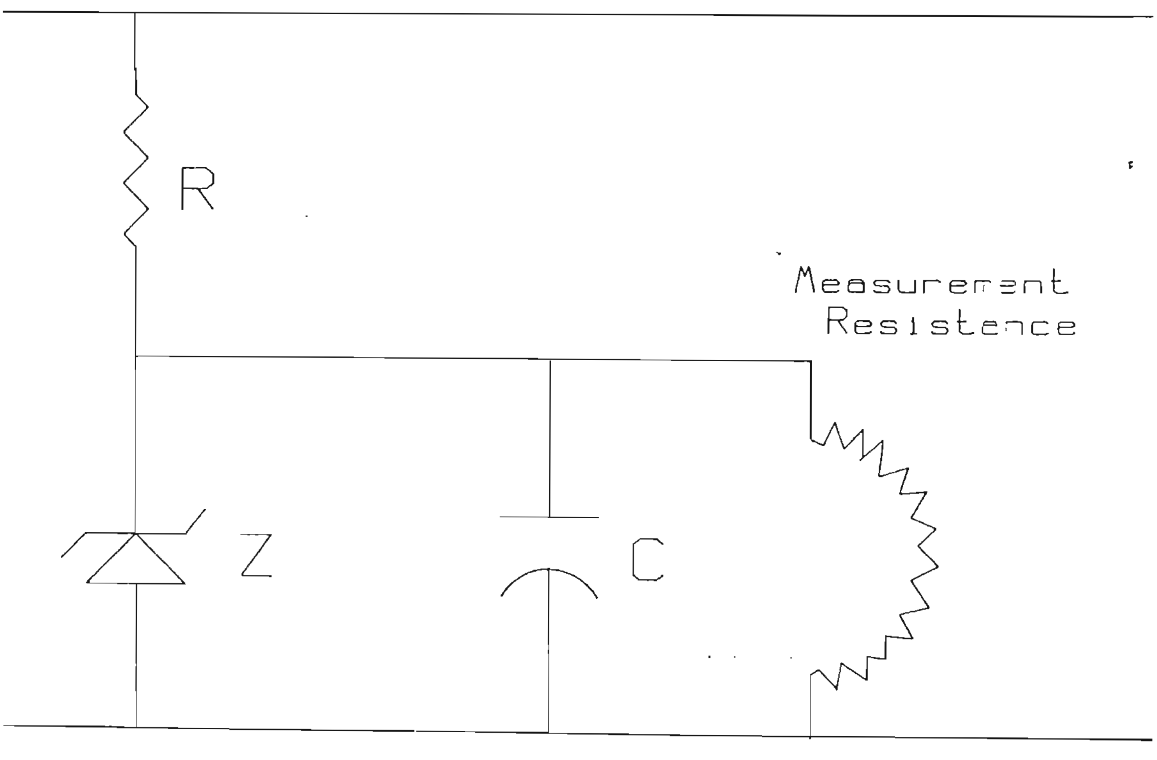
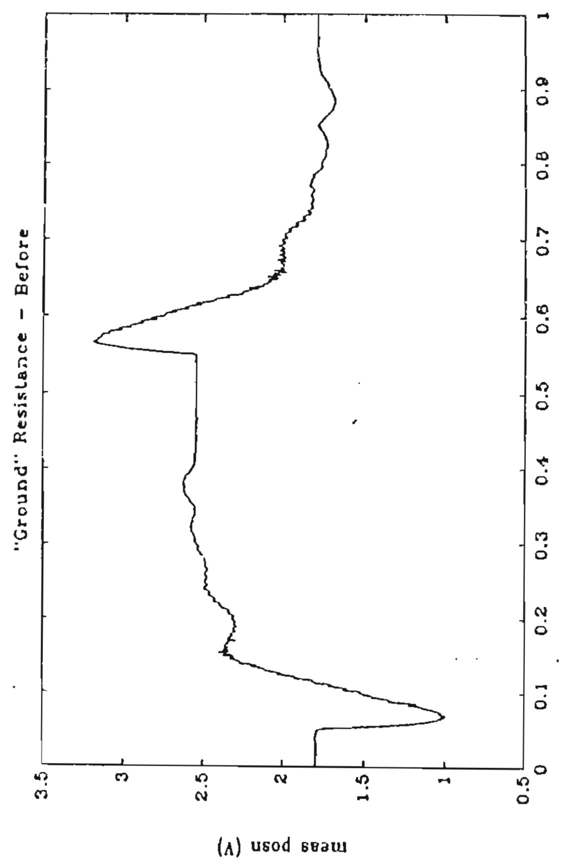
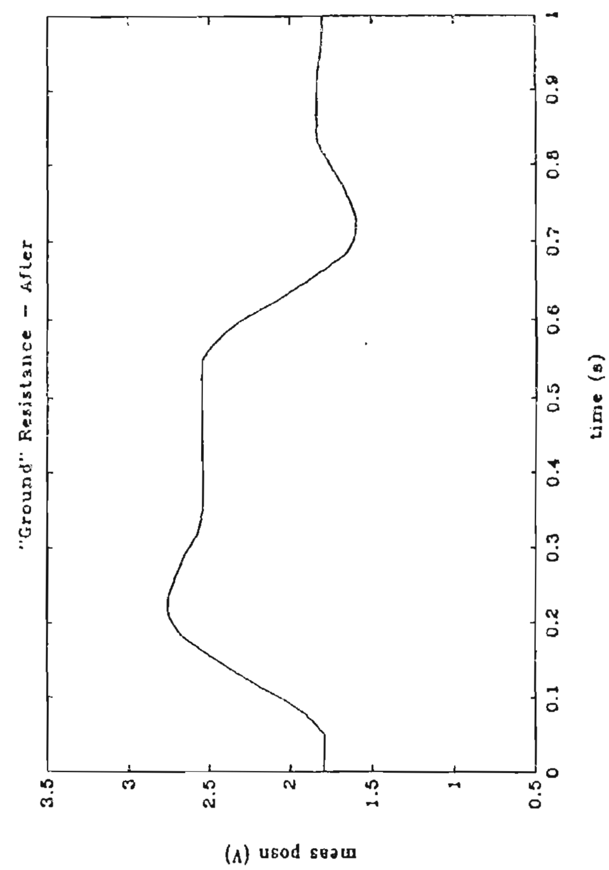
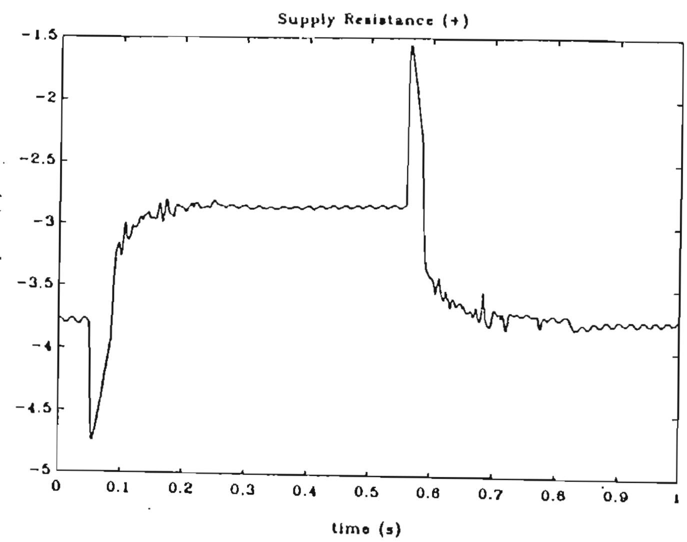
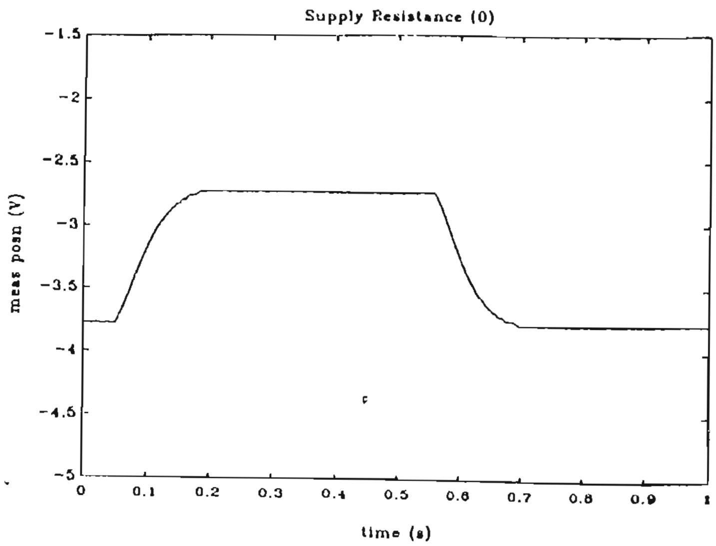


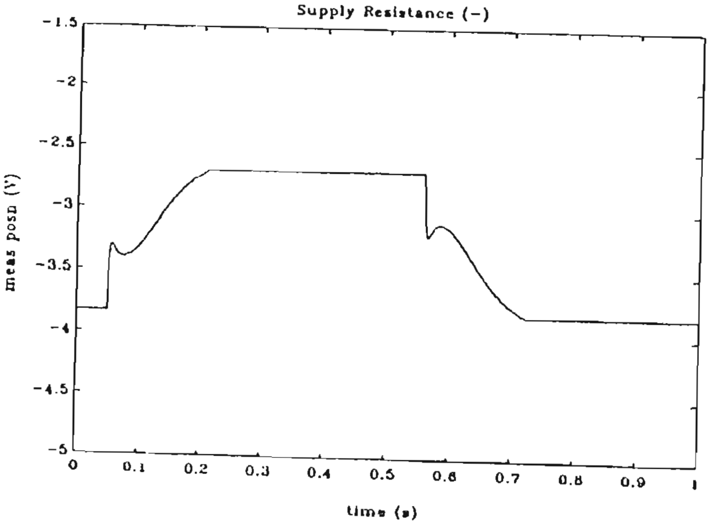
Figure 3- Power Supply











Graph 2 Non-Linear example - Supply Resistance

Appendix 3 – Computer programs

This appendix contains source code in Modula–2 for the following Programs:

3.1) TEMPLATE – a program for evaluating plant parameters from the base (independent) parameters in Table 2.1. This program was modified for each controller structure investigated. The output of this program is compatible with "DESIGN". "TEMPLATE" gives the formulas (from eq(2.21)) for evaluating,

$$\mu_t = \mu_{t0} - \left.\frac{\partial T}{\partial \vartheta}\right|_{o.p.} \qquad \mu_\ell = \mu_{\ell 0} - \left.\frac{\partial F}{\partial \dot{h}}\right|_{o.p.}$$

$$k_t = - \left.\frac{\partial T}{\partial \theta}\right|_{o.p.} \qquad k_{tf} = \left.\frac{\partial T}{\partial \alpha}\right|_{o.p.}$$

$$k_{tb} = \left.\frac{\partial T}{\partial \beta}\right|_{o.p.} \qquad A = - \left.\frac{\partial T}{\partial \dot{h}}\right|_{o.p.}$$

$$k_\ell = \left.\frac{\partial F}{\partial \theta}\right|_{o.p.} \qquad k_{\ell f} = \left.\frac{\partial F}{\partial \alpha}\right|_{o.p.}$$

$$k_{\ell b} = \left.\frac{\partial F}{\partial \beta}\right|_{o.p.} \qquad B = \left.\frac{\partial F}{\partial \vartheta}\right|_{o.p.}$$

(Note: Since submission and acceptance of this work, Mr S Govender has brought a sign error in the term ($+/- v_{vertical}\sin(\theta) + v_{horizontal}\cos(\theta)$) term (which occurs a number of times) to the author's attention. The error is corrected on the software listings but has not been corrected back into the designs. Fortunately the error does not affect the results significantly for the small angles (<15°) and low vertical velocity relative to the wind velocity used to construct templates. The second term can be expected to be at worst 50 times larger than the first.)

```
:mplate;

'O;

'LEX
 Complex, SetComplex, AddComplex, SubtComplex, MultComplex,
 DivComplex, ScaleComplex, dB, arg, ReadComplex, WriteComplex;

 WrLngReal, RdLngReal, WrStr, RdStr, WrCard, KeyPressed, RdKey, WrLn;
 LIB
 Sin, Cos, Pow, ATan2 (*X,Y : LONGREAL*), Exp, Log;

14159265358979314;

--------------------------------------------------------------------*)
 sqr (a:LONGREAL) : LONGREAL;

a;

--------------------------------------------------------------------*)
 LiftDrag(     x : LONGREAL;
          VAR cL,dL,cD,dD : LONGREAL);

OCEDURE ABS(y : LONGREAL) : LONGREAL;
GIN
 y > 0.0 THEN RETURN y
SE RETURN -y; END;
D ABS;

 (-5.3054*x*x + 5.0055*ABS(x))*0.5*1.2;
 (-10.6109*ABS(x) + 5.0055)*0.5*1.2;
 (3.5730*x*x + 0.0174)*0.5*1.2;
  7.1461*x*0.5*1.2;
rag;

--------------------------------------------------------------------*)

0529;
16;
43;
24;
2.75;
1.0;
2.75;
1.0;
= 0.0;
.0;

ol, i, ix, iy, ic : CARDINAL;

Minv, Minv, kOverJM, One, Zero, omega,
ew, mA, mB, gT, gH, g12, g21 : Complex;

heta, betaPtheta, k, klf, klb, ktf, ktb, kt, kl, windF, windB,
', vbX, vbY,    vSQf, vSQb,
```

```
  ooopx2f, ooopx2b, (* 1/(1+x^2) *)
  cLF, dLF, cLB, dLB, cDF, dDF, cDB, dDB,
  cosT, sinT, A, B,
  cosTgf, sinTgf, gammab, cosTgb, sinTgb, cosGf, cosGb, sinGf, sinGb, gammaf,
  tdot, hdot, theta, inertia, mass, uturn, ulift,
  minarg, maxarg, maxdB, mindB, w, w2    : LONGREAL;

  ok : BOOLEAN;
  InputBuffer, OutputBuffer : ARRAY[1..(1024+FIO.BufferOverhead)] OF BYTE;
  InputFile, OutputFile : FIO.File;
  FileName : ARRAY[1..40] OF CHAR;
  Reply : CHAR;
  index : CARDINAL;
  Output, ModArg, ModArgInverse : ARRAY [1..2] OF ARRAY [0..500] OF LONGREAL;

BEGIN
   index := 1;
   One := SetComplex(1.0, 0.0);
   Zero := SetComplex(0.0, 0.0);

  REPEAT
    WrStr('Enter input file ');
    RdStr(FileName);
  UNTIL FIO.Exists(FileName);
  InputFile := FIO.Open(FileName);
  FIO.AssignBuffer(InputFile, InputBuffer);
  WrStr('Enter w ');
  w := RdLngReal();
  omega := SetComplex(0.0,  w);
  w2 := w*w;

  FIO.EOF := FALSE;
  WHILE NOT(FIO.EOF) DO
    theta   := FIO.RdLngReal(InputFile);  (* in degrees *)
    tdot    := FIO.RdLngReal(InputFile);   (* in rad/s   *)
    hdot    := FIO.RdLngReal(InputFile);   (* in m/s     *)
    inertia:= FIO.RdLngReal(InputFile);
    mass    := FIO.RdLngReal(InputFile);
    uturn   := FIO.RdLngReal(InputFile);
    ulift   := FIO.RdLngReal(InputFile);

    alphaPtheta   := FIO.RdLngReal(InputFile);
    windF  := FIO.RdLngReal(InputFile);

    betaPtheta    := FIO.RdLngReal(InputFile);
    windB  := FIO.RdLngReal(InputFile);

    IF FIO.OK AND NOT(FIO.EOF) THEN
      theta := theta/180.0*pi;
      alphaPtheta := alphaPtheta/180.0*pi;
      betaPtheta := betaPtheta/180.0*pi;

      cosT := Cos(theta);
      sinT := Sin(theta);

      vfX := windF-tdot*rf*sinT; (* front horizontal *)
      vfY := -(hdot+tdot*rf*cosT);  (* front vertical *)

      vbX := windB+tdot*rb*sinT;  (* back horizontal *)
      vbY := -(hdot-tdot*rb*cosT); (* back vertical *)
```

```
af := ATan2(vfX, vfY);
ab := ATan2(vbX, vbY);

Drag((alphaPtheta+gammaf),cLF,dLF,cDF,dDF);
Drag((betaPtheta+gammab),cLB,dLB,cDB,dDB);

 := sqr(vfX) + sqr(vfY);
 := sqr(vbX) + sqr(vbY);

x2f := 1.0/(1.0+sqr(vfY/vfX));
x2b := 1.0/(1.0+sqr(vbY/vbX));

gf := Cos(theta+gammaf);
gf := Sin(theta+gammaf);

gb := Cos(theta+gammab);
gb := Sin(theta+gammab);

f := Cos(gammaf);
f := Sin(gammaf);

b := Cos(gammab);
b := Sin(gammab);

n := uturn - af*sqr(rf)*(
  2.0*(-vfX*sinT - vfY*cosT) * (cLF*cosTgf+cDF*sinTgf)
  + vSQf * ((dLF+cDF)*cosTgf + (dDF-cLF)*sinTgf) *
  ooopx2f * (sinT*vfY - cosT*vfX) / sqr(vfX) )
  +
  ab*sqr(rb)*(
  2.0*(vbX*sinT + vbY*cosT) * (cLB*cosTgb+cDB*sinTgb)
  + vSQb * ((dLB+cDB)*cosTgb + (dDB-cLB)*sinTgb)*
  ooopx2b * (-sinT*vbY + cosT*vbX) / sqr(vbX) );

= - af*rf*(-2.0*rf*tdot*
  (vfX*cosT - vfY*sinT) * (cLF*cosTgf+cDF*sinTgf)
  + vSQf * ((dLF+cDF)*cosTgf + (dDF-cLF)*sinTgf)*
  (1.0+ooopx2f*tdot*rf*(cosT*vfY + sinT*vfX) / sqr(vfX) ))

  + ab*rb*( 2.0*tdot*rb*
  (vbX*cosT - vbY*sinT) * (cLB*cosTgb+cDB*sinTgb)
  + vSQb * ((dLB+cDB)*cosTgb + (dDB-cLB)*sinTgb)*
  (1.0-ooopx2b*tdot*rb*(cosT*vbY + sinT*vbX) / sqr(vbX) ));

= af*rf*vSQf*(cosTgf*dLF + sinTgf*dDF);
= ab*rb*vSQb*(cosTgb*dLB + sinTgb*dDB);


- af*rf*(-2.0*vfY*(cLF*cosTgf+cDF*sinTgf)
+ vSQf * ((dLF+cDF)*cosTgf + (dDF-cLF)*sinTgf)*
(-ooopx2f)/vfX )
+
ab*rb*(-2.0*vbY*(cLB*cosTgb+cDB*sinTgb)
+ vSQb * ((dLB+cDB)*cosTgb + (dDB-cLB)*sinTgb)*
(-ooopx2b)/vbX );

t equ *)
= af*vSQf*(cosGf*dLF + sinGf*dDF);
= ab*vSQb*(cosGb*dLB + sinGb*dDB);
```

```
      kl := af*(-2.0*rf*tdot*(vfX*cosT - vfY*sinT)*
             (cLF*cosGf+cDF*sinGf)

            + vSQf*((dLF*cosGf+dDF*sinGf)*
               (1.0+ooopx2f*tdot*rf*(cosT*vfY + sinT*vfX) / sqr(vfX))

            + (cDF*cosGf-cLF*sinGf) * ooopx2f
              * tdot*rf*(cosT*vfY + sinT*vfX) / sqr(vfX))    )

             + ab*(2.0*rb*tdot*(vbX*cosT + vbY*sinT)*
            (cLB*cosGb+cDB*sinGb)

            + vSQb*((dLB*cosGb+dDB*sinGb)*
              (1.0-ooopx2b*tdot*rb*(cosT*vbY - sinT*vbX) / sqr(vbX))

            +(cDF*cosGf+cLF*sinGf) * ooopx2b
            *tdot*rb*(-cosT*vbY - sinT*vbX) / sqr(vbX)) );

      B :=  af*(-2.0*rf*(vfX*sinT + vfY*cosT) * (cLF*cosGf+cDF*sinGf)
            + vSQf * ((dLF+cDF)*cosGf+(dDF-cLF)*sinGf) * (-ooopx2f)
               *rf*(-sinT*vfY + cosT*vfX) / sqr(vfX))

             + ab*(2.0*rb*(vbX*sinT + vbY*cosT) * (cLB*cosGb+cDB*sinGb)
            + vSQb * ((dLB+cLF)*cosGb+(dDB-cLB)*sinGb) * ooopx2b
              *rb*(-sinT*vbY + cosT*vbX) / sqr(vbX)) ;

      ulift := ulift - af*(-2.0*vfY*(cLF*cosGf+cDF*sinGf)
           + vSQf * ((dLF+cDF)*cosGf + (dDF-cLF)*sinGf) * (-ooopx2f) / vfX )
           - ab*(-2.0*vbY*(cLB*cosGb+cDB*sinGb)
           + vSQb * ((dLB+cDB)*cosGb + (dDB-cLB)*sinGb) * (-ooopx2b)  / vbX );
      k := ktf*klb+ktb*klf;


      WrLngReal(A,5,11);
      WrLngReal(B,5,11);
      WrLngReal(kl,5,11);
      WrLngReal(kt,5,11);
      WrLngReal(klf,5,11);
      WrLngReal(klb,5,11);
      WrLngReal(ktf,5,11);
      WrLngReal(ktb,5,11);
      WrLngReal(ulift,5,11);
      WrLngReal(uturn,5,11);
      WrLn();

     END;
    END;
    FIO.Close(InputFile);

END template.
```

3.2) Source code for Program "IFLYNEW" to simulate the disturbance decoupling controller of Chapter 4.

For the simulation, the plant states are chosen as follows,

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \end{bmatrix} = \begin{bmatrix} \theta \\ \dot{\theta} \\ h \\ \dot{h} \\ \alpha \\ \dot{\alpha} \\ \beta \\ \dot{\beta} \end{bmatrix} \begin{array}{l} \text{pitch angle} \\ \text{pitch rate} \\ \text{height} \\ \text{vertical velocity} \\ \text{front wing position} \\ \text{front wing rate} \\ \text{back wing position} \\ \text{back wing rate} \end{array}$$

and the state differential equation is,

$$\dot{x}_1 = x_2$$
$$\dot{x}_2 = \frac{1}{J}(-\mu_{t0} + \text{Torque})$$
$$\dot{x}_3 = x_4$$
$$\dot{x}_4 = \frac{1}{M}(\mu_{\ell 0} + \text{TotalLift})$$
$$\dot{x}_5 = x_6$$
$$\dot{x}_6 = -\omega_{n\alpha}^2 x_5 - 2\zeta_\alpha \omega_{n\alpha} x_6$$
$$\dot{x}_7 = x_8$$
$$\dot{x}_8 = -\omega_{n\beta}^2 x_7 - 2\zeta_\beta \omega_{n\beta} x_8$$

3.3) Those MODULES from program "HORFLY" which have been modified from "IFLYNEW" above, in order to simulate the anti–diagonal controller of Chapter 5.

```
:flyNew;

:O IMPORT RdLngReal, WrLngReal, RdCard, WrLn, WrStr,
          WrCard, WrChar;
latrix IMPORT RealMatrix, allocateMatrix, deallocateMatrix, Entry, putEnt

               MatrixResult, writeMatrix,
               makeZeroMatrix, makeIdentityMatrix, Gauss;

IATHLIB IMPORT Exp;
'lyStart IMPORT pi, zero, vector;
riteToF IMPORT WriteToFile, results;

' CONTROLLERS;
' PLANTend;

-------------------------------------------------------------*)
:E Global(VAR PLANTdu, CONTROLdu : vector);

 solve : du[i] - du[i]/dx[j]*X[j]*du[j] = du[i]/dx[j]*dx[j] *)

 : RealMatrix;
, ent2, rank : CARDINAL;
mpat, temp : LONGREAL;


 = b is partitioned : [     I(p)      -du/dx*X(c) | du/dx*dx(c)]
                      [-du/dx*X(p)      I(c)      | du/dx*dx(p)]  *)

teMatrix (A, 4, 4);
teMatrix (b, 4, 1);
entityMatrix(A);
= CONTROLLERS.OrderTheta;
= ent1 + CONTROLLERS.OrderHeight;

 row of A *)
ry( 1.0+Entry(1,1,PLANTend.X), 1, 1, A);
ry( Entry(1,2,PLANTend.X)    , 1, 2, A);

= -CONTROLLERS.Kinv[1,1]*Entry(1,ent1,CONTROLLERS.X)
  -CONTROLLERS.Kinv[1,2]*Entry(1,ent2,CONTROLLERS.X);
ry( temp                    , 1, 3, A);

= -CONTROLLERS.Kinv[1,1]*Entry(2,ent1,CONTROLLERS.X)
  -CONTROLLERS.Kinv[1,2]*Entry(2,ent2,CONTROLLERS.X);
ry( temp                    , 1, 4, A);

 row of A *)
ry( Entry(1,1,PLANTend.X)    , 2, 1, A);
ry( 1.0+Entry(1,2,PLANTend.X), 2, 2, A);
= -CONTROLLERS.Kinv[2,1]*Entry(1,ent1,CONTROLLERS.X)
  -CONTROLLERS.Kinv[2,2]*Entry(1,ent2,CONTROLLERS.X);
ry( temp                    , 2, 3, A);
= -CONTROLLERS.Kinv[2,1]*Entry(2,ent1,CONTROLLERS.X)
  -CONTROLLERS.Kinv[2,2]*Entry(2,ent2,CONTROLLERS.X);
ry( temp                    , 2, 4, A);

 row of A *)
ry( Entry(1,1,PLANTend.X)    , 3, 1, A);
ry( Entry(1,2,PLANTend.X)    , 3, 2, A);
```

```
  (* 4th row of A *)
  putEntry( Entry(3,1,PLANTend.X)    , 4, 1, A);
  putEntry( Entry(3,2,PLANTend.X)    , 4, 2, A);

  (* b *)
  temp := -PLANTend.dx[1] + CONTROLLERS.Kinv[1,1]*CONTROLLERS.dx[ent1]
                          + CONTROLLERS.Kinv[1,2]*CONTROLLERS.dx[ent2];
  putEntry( temp                  , 1, 1, b);
  temp := -PLANTend.dx[1] + CONTROLLERS.Kinv[2,1]*CONTROLLERS.dx[ent1]
                          + CONTROLLERS.Kinv[2,2]*CONTROLLERS.dx[ent2];
  putEntry( temp                  , 2, 1, b);
  putEntry( -PLANTend.dx[1]          , 3, 1, b);
  putEntry( -PLANTend.dx[3]          , 4, 1, b);

  Gauss(A, b, rank, incompat, zero);
  IF MatrixResult.Error THEN
    WrStr('matrix error in Global');
    WrLn;
  END;
  PLANTdu[1] := Entry(1,1,b);
  PLANTdu[2] := Entry(2,1,b);
  CONTROLdu[1] := Entry(3,1,b);
  CONTROLdu[2] := Entry(4,1,b);

  deallocateMatrix(A);
  deallocateMatrix(b);

END Global;
(*-------------------------------------------------------------------*)

VAR
    PLANTdu, CONTROLdu : vector;
    nexttick, time, ticklength, deltaT,
    theta, height, AplusThat, BplusThat, tref, href : LONGREAL;
    ares, bres, hres, tres : results;
    NumberSteps, step : CARDINAL;
    tick : BOOLEAN;
BEGIN

  WrStr('Initialising FlyingMachine');
  WrLn;
  WrStr('Enter deltaT ');
  deltaT := RdLngReal();
  WrLn;
  WrStr('Enter NumberSteps ');
  NumberSteps := RdCard();
  WrLn;
  WrStr('Tick length ');
  ticklength := RdLngReal();
  WrLn;

  nexttick := 0.0;
  time := 0.0;

  AplusThat := 0.0;
  BplusThat := 0.0;

  theta := 0.0;
  height := 0.0;
```

```
ep := 1 TO NumberSteps DO
r('step ');
rd(step,4);
 := time + deltaT;
 := 0.0;
 := 0.0;

 :=-0.5;
ime > 11.0 THEN
ref := -0.5+1.0*(1.0-Exp(-0.7*(time-11.0)));


ime >= nexttick THEN
k:=TRUE;
tttick := time+ticklength;

k:=FALSE;


step] := height;
step] := theta*180.0/pi;
step] := (AplusThat-theta)*180.0/pi;
step] := (BplusThat-theta)*180.0/pi;

end.Plant(deltaT, AplusThat-theta, BplusThat-theta,
                                        step, tick, TRUE);
OLLERS.Controllers(deltaT, tref-0.017-theta, href+0.11-height);

l(PLANTdu, CONTROLdu);

end.PlantState(PLANTdu, height, theta);
OLLERS.ControllerState(CONTROLdu, AplusThat, BplusThat);



Height');
File(hres,NumberSteps);

THETA ');
File(tres,NumberSteps);

ALPHA ');
File(ares,NumberSteps);

 BETA ');
ile(bres,NumberSteps);

l.EndPlant;

ERS.EndControl;

w.
```

```
TION MODULE Matrix;

.t: 28.05.1988
  : Eduard Eitelberg & Edward Boje
  : LOGITECH MODULA-2/86


 of real matrix operations, Gaussian elimination, eigenvalues:
============================================================

ocateMatrix
llocateMatrix
sOf
umnsOf
ry
Entry
dMatrix
teMatrix
eZeroMatrix
eIdentityMatrix
yMatrix
yBlock
leMatrix
metrizeMatrix
Matrix
tractMatrix
tiplyMatrix

ss

ocateEigVals
llocateEigVals
ensionOf
EigValsEntry
ndEigVal
culateEigVals

rices (declared of type RealMatrix) and eigenvalue arrays (declared of ty

lues) must be initialised in the user program with allocateMatrix and
eEigVals respectively. This does not guarantee any particular numerical
for the matrix or eigenvalue entries, however. The user is responsible
 release of memory through deallocation with deallocateMatrix and
ateEigVals, else user program may end with heap and stack collision.

 matrix size (Rows*Columns=MaxSize) is currently 8189 reals (LOGITECH
ntation limit), leading to 90 as the (theoretically) maximum number of
ues (91*91=8281 would be greater than 8189=MaxSize).

ord, MatrixResult, contains an error indicator of matrix operations
 corresponding message , it is set as a side-effect in most procedures.
 constants, transp=TRUE and notransp=FALSE, may be used for better
lity in some procedure calls.

ever use direct matrix assignments, such as A:=B. This is formally
it leads to memory management problems. Use instead the procedure
rix, or suchlike.


 constant definitions are for info only. They are not exported
```

```
CONST
    MaxCard = 65535;
    MaxSize = 8189;
(* 8 Bytes per real and 2 Bytes per cardinal = 8189*8+4 = 65516 *)
    MaxEVDim = 90;      (* 91*91=8281 > 8189=MaxSize *)
    MaxSpaceAllocatable = 65520; *)

CONST
    MessageLength = 80;
    transp = TRUE;
    notransp = FALSE;

TYPE  RealMatrix;

TYPE  MessageType = ARRAY [0..(MessageLength-1)] OF CHAR;
TYPE  MatrixResultType = RECORD
                           Error: BOOLEAN;
                           Message: MessageType;
                         END (*record*);

TYPE  ComplexNumber = RECORD
                        Real : LONGREAL;
                        Imag : LONGREAL;
                      END (* record *);
TYPE EigenValues;

VAR   MatrixResult: MatrixResultType;
VAR   BlankMessage: MessageType;
      NoErrorMessage: MatrixResultType;


(***************************************************************************)

PROCEDURE allocateMatrix (VAR A : RealMatrix; Rows, Columns : CARDINAL);
(*
Rows*Columns of reals and 2 cardinals are allocated (on the heap) for matrix
A and its dimensions. A entries cannot be be assumed to have any given value.
If A is not too big and the memory is available, MatrixResult.Error=FALSE.
Else allocateMatrix will be left immediately, MatrixResult.Error=TRUE.
*)

(*-------------------------------------------------------------------------*)
PROCEDURE deallocateMatrix (VAR A : RealMatrix);
(* No error message here. Memory is released on the heap, if A # NIL. *)

(*-------------------------------------------------------------------------*)
PROCEDURE RowsOf(VAR A : RealMatrix) : CARDINAL;
(* No error message here - not needed after successful allocation. *)

(*-------------------------------------------------------------------------*)
PROCEDURE ColumnsOf(VAR A : RealMatrix) : CARDINAL;
(* No error message here - not needed after successful allocation. *)

(*-------------------------------------------------------------------------*)
PROCEDURE Entry(i,j : CARDINAL; VAR A : RealMatrix) : LONGREAL;
(* i and j must be in the row and column range respectively. Else error
   is set and 0.0 is returned. *)

(*-------------------------------------------------------------------------*)
PROCEDURE putEntry(x: LONGREAL; i,j : CARDINAL; VAR A : RealMatrix);
```

```
nd j must be in the row and column range respectively. Else error
 set and nothing done. *)

--------------------------------------------------------------------*)
URE readMatrix(VAR A: RealMatrix);
s procedure reads in matrix entries row by row, from a source opened
or to its call, if A has been allocated successfully. Else error is set
 A is not read. Most suited for reading from files. Uses ReadReal. *)

--------------------------------------------------------------------*)
URE writeMatrix(VAR A: RealMatrix; Out: CHAR);
s procedure writes out matrix entries row by row, to a device/file opened
or to its call, if A has been allocated successfully. Else error is set
 A is not written. The format is real, 12 positions, 4 digits accuracy,
olumns at a time.
tead of WriteLn, CR LF is explicitly used, hence output redirection to
nter via Termbase.AssignWrite and Termbase.UnAssignWrite is possible.
 printing, the appropriate (IBM) printer graphics character table has to
activated/selected - see your printer manual.
 = d or D : nice display/printer format is written. If a file has been
            opened then this format is output to the file.
 = f or any other character : number of rows and number of columns are
            followed by the matrix entries. Recommended output to file.
            This format is compatible to readMatrix.
mple of redirection to EPSON LX-800 printer:

ROCEDURE  print(c: CHAR);          (* WriteProcedure *)
EGIN                                      (* Must be *)
 DOSCALL(5H,c);                          (* provided *)
ND print;                               (* in the user *)
AR done : BOOLEAN;                        (* module. *)

ssignWrite(print,done);               (* Redirect output to printer. *)
rite(33C); Write(164C); Write(1C); (* Set character graphics table. *)
riteString('Matrix A:');
rite(15C); Write(12C);     (* Replaces universally MODULA-2 WriteLn. *)
     CR             LF, since WriteLn is not understood by the pinter. *)
riteMatrix(A,'d');                          (* Display-format output. *)
rite(33C); Write(100C);                            (* Reset printer. *)
nAssignWrite(done);                       (* Output to screen again. *)
of example.


-------------------------------------------------------------------*)
RE makeZeroMatrix(VAR A: RealMatrix);
s all A entries 0, if A has been allocated successfully. Else error is
and nothing is done. *)

-------------------------------------------------------------------*)
RE makeIdentityMatrix(VAR A: RealMatrix);

 square matrices, MatrixResult.Error=FALSE. Else makeIdentityMatrix
 left immediately with MatrixResult.Error=TRUE.


-------------------------------------------------------------------*)
RE copyMatrix(VAR A: RealMatrix; transposeA: BOOLEAN;VAR B: RealMatrix);

 may be transposed. Row and column numbers must be compatible and
 must not be same, MatrixResult.Error=FALSE.
```

```
Else copyMatrix will be left immediately, MatrixResult.Error=TRUE.
*)

(*---------------------------------------------------------------------*)
PROCEDURE copyBlock(VAR A: RealMatrix;
                        FromRowA, ToRowA, FromColumnA, ToColumnA: CARDINAL;
                    VAR B: RealMatrix; FromRowB, FromColumnB: CARDINAL);
(*
An indicated block of A is copied into B, so that the "upper left corner" of
the block is as indicated in B. A and B can be same. A and B must be big enough
to contain the block as indicated, MatrixResult.Error=FALSE.
Else copyBlock will be left immediately, MatrixResult.Error=TRUE.
*)

(*---------------------------------------------------------------------*)
PROCEDURE scaleMatrix(VAR A: RealMatrix; x: LONGREAL);
(*
A:= x*A, if A is available.
*)

(*---------------------------------------------------------------------*)
PROCEDURE symmetrizeMatrix(VAR A: RealMatrix);
(*
This forces A to symmetry. Only for square matrices, MatrixResult.Error=FALSE.
Else symmetrizeMatrix will be left immediately with MatrixResult.Error=TRUE.
*)

(*---------------------------------------------------------------------*)
PROCEDURE addMatrix(VAR A: RealMatrix; transposeA: BOOLEAN;
                    VAR B: RealMatrix; transposeB: BOOLEAN; VAR C: RealMatrix);
(*
C:= A+B, where C, A and B can be same.
A and/or B may be transposed, but then C must be different to the transposed
matrices/matrix, MatrixResult.Error=FALSE. Else addMatrix will be left
immediately, MatrixResult.Error=TRUE.
All row and column numbers must be compatible, MatrixResult.Error=FALSE.
Else addMatrix will be left immediately, MatrixResult.Error=TRUE.
*)

(*---------------------------------------------------------------------*)
PROCEDURE subtractMatrix(VAR A: RealMatrix; transposeA: BOOLEAN;
                         VAR B: RealMatrix; transposeB: BOOLEAN; VAR C: RealMatr
ix);
(*
C:= A-B, where C, A and B can be same.
A and/or B may be transposed, but then C must be different to the transposed
matrices/matrix, MatrixResult.Error=FALSE. Else addMatrix will be left
immediately, MatrixResult.Error=TRUE.
All row and column numbers must be compatible, MatrixResult.Error=FALSE.
Else addMatrix will be left immediately, MatrixResult.Error=TRUE.
*)

(*---------------------------------------------------------------------*)
PROCEDURE multiplyMatrix(VAR A: RealMatrix; transposeA: BOOLEAN;
                         VAR B: RealMatrix; transposeB: BOOLEAN; VAR C: RealMatr
ix);
(*
C:= A*B, where C must be different to A and B, A and B can be same.
A and/or B may be transposed.
All row and column numbers must be compatible and C must be different to
A and B, MatrixResult.Error=FALSE. Else multiplyMatrix will be left
immediately, MatrixResult.Error=TRUE.
```

```
---------------------------------------------------------------------*)
RE Gauss(VAR A, B: RealMatrix; VAR RankOfA: CARDINAL;
         VAR MaxIncomp: LONGREAL; Zero: LONGREAL);

ssumes that A and B are different but have the same number of rows.
not need to be square.

n reduction is done on matrix A, whilst doing the same row operations
adjacent matrix B. Full pivoting is used. Three permutation vectors of
ls are used (allocated temporarily in Gauss).

lthough both matrices, A and B will be changed, the equati
                              A*X=B
qually well before and after Gauss. An eventual equation
tibility is not scaled during Gauss.

ns with:
ity, in case of full rank square A.
mber of diagonal 1's equal to its calculated rank, in case A has not
rows than columns. The other diagonal entries are 0 (or near-0).
se of full pivoting, these 1's and 0's may be intermittent.
has less rows than columns, then the pivot 1's are not (necessarily) on
iagonal.

ns with the corresponding fully or partially 'reduced' adjacent matrix.

esult.Error=TRUE if an error occurs, set as side-effect together with
      the message in the MatrixResult.Message.
      The possible errors are:
      a) Incompatible matrix sizes.
      b) Not enough memory for temporary allocation.
      c) System overdetermined (a non-zero value found in any
         non-pivot row of B).

      is the number of pivot rows of the matrix A (system
      underdetermined if less than number of A columns).

p     returns the absolute largest value in the non-pivot rows of the
      matrix B (=0 => incompatibility, =0 => no incompatibility).
      Note, these rows are not scaled during reduction.
      It is set to 0 if RankOfA equals number of A rows.

      is the user-determined value below which zero is assumed to exist.


---------------------------------------------------------------------*)
E allocateEigVals(VAR EV : EigenValues; Dimension : CARDINAL);

ion of reals, Dimension of booleans and 1 cardinal are allocated
heap) for eigenvalues EV and its dimension. EV entries cannot be be
to have any given value. If EV is not too big and the memory is
e, MatrixResult.Error=FALSE. Else allocateEigVals will be left
ely, MatrixResult.Error=TRUE.

---------------------------------------------------------------------*)
E deallocateEigVals(VAR EV : EigenValues);
ror message here. Memory is released on the heap, if EV = NIL. *)
---------------------------------------------------------------------*)
```

```
PROCEDURE DimensionOf(VAR EV : EigenValues) : CARDINAL;
(* No error message here - not needed after successful allocation. *)
(*---------------------------------------------------------------------*)
PROCEDURE getEigValsEntry(i: CARDINAL; VAR EV: EigenValues;
                          VAR EVEntry: ComplexNumber);
(* Error, if i is greater than dimension of EV. Whether the result has
   actually been found through calculation, must be checked with foundEigVal.
*)
(*---------------------------------------------------------------------*)
PROCEDURE foundEigVal(i: CARDINAL; VAR EV : EigenValues) : BOOLEAN;
(* Error, if i is greater than dimension of EV. *)
(*---------------------------------------------------------------------*)

PROCEDURE calculateEigVals(VAR EV: EigenValues; VAR A: RealMatrix);
(* This co-ordinates the evaluation of the eigenvalues of the matrix A:
   A    - The matrix whose eigenvalues are required. Must be allocated by the
          user with equal number of rows and columns.
   EV   - The set of eigenvalues, whose entries EVEntry of type ComplexNumber
          may be extracted with getEigValsEntry(i, EV, EVEntry), provided
          foundEigVal(i,EV) is TRUE. Else the corresponding eigenvalue has not
          been found, since the algorithm is not guaranteed to converge.
          EV must be allocated by the user (with allocateEigVals) with the
          number of entries equal to the number of A rows.

   Another matrix, of the same size as A, and two single-column matrices,
   with as many rows as A, are allocated temporarily during this procedure.
*)
(*---------------------------------------------------------------------*)

PROCEDURE print(c: CHAR);          (* WriteProcedure *)

(*---------------------------------------------------------------------*)

END Matrix.
```

```
ION MODULE FlyStart;

3.1415926536;
= 1.0E-6;

or = ARRAY[1..10] OF LONGREAL;

Start.
```

```
NTATION MODULE FlyStart;
Start.
```

```
DEFINITION MODULE COMPLEX;

TYPE
  Complex = RECORD
             Rel, (* Real part *)
(* Imaginary part *) Imag: LONGREAL;
               END;
VAR
   ComplexDivideResult : BOOLEAN;

  PROCEDURE SetComplex(x, y : LONGREAL) : Complex;

  PROCEDURE AddComplex(C1, C2: Complex) : Complex;
  (* Procedure to add two complex numbers *)

  PROCEDURE SubtComplex(C1, C2: Complex) : Complex;
  (* Procedure to subtract  two complex numbers *)

  PROCEDURE MultComplex(C1, C2: Complex) : Complex;
  (* Procedure to multiply two complex numbers *)

  PROCEDURE DivComplex(C1, C2: Complex) : Complex;
  (* Procedure to divide two complex numbers and return TRUE *)
  (* if operation is successful, FALSE for division by zero *)

  PROCEDURE ScaleComplex(C : Complex; Scale : LONGREAL) : Complex;
  (* Scales Complex by Scale *)

  PROCEDURE dB(C : Complex) : LONGREAL;

  PROCEDURE arg(C : Complex) : LONGREAL;

  PROCEDURE ReadComplex() : Complex;
   (* Procedure to read complex number *)

  PROCEDURE WriteComplex(C: Complex (* input *));
  (* Procedure to output a complex number *)

END COMPLEX.
```

```
ITION MODULE PLANTend;
latrix IMPORT RealMatrix;

FlyStart IMPORT vector;


 RealMatrix;
 vector;

URE Plant(deltaT, AlphaHat, BetaHat : LONGREAL; step : CARDINAL;
                                   tick, disturb : BOOLEAN);
URE PlantState(VAR du : vector;
               VAR   height, theta : LONGREAL);
            (* update plant's state vector *)

URE EndPlant;  (* Must be called before main module ends to
                                          deallocate memory *)

ANTend.
```

```
DEFINITION MODULE CONTROLLERS;

FROM Matrix IMPORT RealMatrix;
FROM FlyStart IMPORT vector;

VAR
    X : RealMatrix;
    dx : vector;
    OrderTheta, OrderHeight : CARDINAL;
    Kinv : ARRAY[1..2] OF ARRAY[1..2] OF LONGREAL;

PROCEDURE Controllers(deltaT, theta, height : LONGREAL);
PROCEDURE ControllerState(VAR du : vector;
                          VAR AplusThat, BplusThat : LONGREAL);
                              (* update controllers' state vector *)
PROCEDURE EndControl;    (* must be called before end of main module to
                                                deallocate memory *)

END CONTROLLERS.
```

```
MENTATION MODULE PLANTend;
*)
T FloatExc;

T FIO;

MATHLIB IMPORT Sin, Cos, ATan2;

IO IMPORT
      RdLngReal, WrLngReal, RdCard, RdInt, WrInt,
      WrLn, RdStr, WrStr,WrCard, WrChar;
enOutput, CloseOutput, OpenInput, CloseInput, *)
FlyStart IMPORT vector, pi, zero  (*print*);

Matrix
ORT RealMatrix, allocateMatrix, deallocateMatrix, Entry, putEntry,
    MatrixResult, writeMatrix, makeZeroMatrix, makeIdentityMatrix, Gauss;

WriteToF
ORT results, WriteToFile;



9.81;            (* m/s² *)
sity = 1.2;
aMass = 0.8; (* = 1.5-0.7 = mass - antimass *)


Length = 1000;

urbance = ARRAY[1..DistLength] OF LONGREAL;

tPosn, FrontRate : results;

 vector;
LF, distLB, distDF, distDB : disturbance;

F,windB : LONGREAL; (* Now global to keep wind speed from one to the next

Wind  : CARDINAL;
tia,           (* Moment of inertia about pivot*)
Front,  (* Total area of front wings *)
Back,   (* Total area of back wings m²*)

n,       (* Rotational friction *)
t,       (* Sliding friction *)
n0,       (* Rotational friction *)
t0,       (* Sliding friction *)
        (* Kg *)
reToPivot, (* pivot in front of centre of mass *)
tToPivot,  (* m *)
ToPivot,   (* m *)
Front, DampingFront, OmegaSqFront, TwoOmegaDFront,
Back,  DampingBack,  OmegaSqBack,  TwoOmegaDBack,
pha, maxAlpha, minBeta, maxBeta, maxAlphaRate, maxBetaRate : LONGREAL;
----------------------------------------------------*)


RE CoefficientLift(angle : LONGREAL; VAR cL, dL : LONGREAL);
```

```
    PROCEDURE SGN(y : LONGREAL) : LONGREAL;
    BEGIN
    IF y > 0.0 THEN RETURN 1.0 ELSE RETURN -1.0; END;
    END SGN;

CONST
   rho = 1.2;
BEGIN
    cL := SGN(angle)*(-5.3054*angle*angle + 5.0055*ABS(angle))*0.5*rho;
    dL := (-10.6109*ABS(angle) + 5.0055)*0.5*rho;
END CoefficientLift;

(*----------------------------------------------------------*)
PROCEDURE CoefficientDrag(angle : LONGREAL; VAR cD, dD : LONGREAL);

CONST
   rho = 1.2;
BEGIN
    cD := (3.5730*angle*angle + 0.0174)*0.5*rho;
    dD :=  7.1461*angle*0.5*rho;

END CoefficientDrag;

(*----------------------------------------------------------*)

PROCEDURE Wind() : LONGREAL ;
CONST
    MeanSpeed = 20.0;
    NumberWinds = 6;
VAR
    temp : LONGREAL;
    WSpeed : ARRAY[1..NumberWinds] OF LONGREAL;
BEGIN
  RETURN 20.0;   (* for the time being *)
END Wind;

(*----------------------------------------------------------*)

PROCEDURE Plant(deltaT, AlphaHat, BetaHat : LONGREAL; step : CARDINAL;
                                          tick, disturb : BOOLEAN);

(* AlphaHat = desired value of alpha *)

VAR
  Phi, Fmat : RealMatrix;
  f : vector;
  cosT, sinT : LONGREAL;
  dLF, dLB, dDF, dDB, cLF, cLB, cDF, cDB : LONGREAL;
  gammaf, gammab : LONGREAL;
              (* lift etc given by cLF derivative by dLF *)

  alpha, beta, theta, tdot, height, hdot     : LONGREAL;

(*-----------------------------------------------------------------------*)
PROCEDURE Wings;

VAR temp : LONGREAL;

BEGIN
    (* Wing movements in controller form*)
```

```
 front wings *)
pha := x[5];
tEntry (-1.0,                          5, 6, Phi);
tEntry (OmegaSqFront,                  6, 5, Phi);
mp := Entry(6,6,Phi);
tEntry ((temp + TwoOmegaDFront), 6, 6, Phi);

tEntry(OmegaSqFront, 6,2,Fmat);

5] := x[6];
6] :=  - OmegaSqFront*x[5] - TwoOmegaDFront*x[6] + OmegaSqFront*AlphaHat;

 back wings *)
ta := x[7];
Entry (-1.0,                          7, 8, Phi);
Entry (OmegaSqBack,                   8, 7, Phi);
np := Entry(8,8,Phi);
Entry ((temp + TwoOmegaDBack), 8, 8, Phi);

Entry(OmegaSqBack,  8,3,Fmat);

] := x[8];
] :=  - OmegaSqBack*x[7] - TwoOmegaDBack*x[8] + OmegaSqBack*BetaHat;
gs;

-------------------------------------------------------------*)

RE Plant1;      (* calculate some lifts, drags, etc *)

EDURE sqr(a:LONGREAL) : LONGREAL;
N
ETURN a*a;
sqr;


 rf, af, ab, kt, ktf, ktb, klf, klb, kl, A, B, vfX, vfY, vbX, vbY,
f, vSQb,
px2f, ooopx2b, (* 1/(1+x^2) *)

T, sinT, cosTgf, sinTgf, cosTgb, sinTgb, cosGf, sinGf, cosGb, sinGb,
p : LONGREAL;

OverT, TotalLift, Torque : LONGREAL;
, col, i : CARDINAL;


 AreaFront;
 AreaBack;
 FrontToPivot;
 BackToPivot;

:k THEN
lF := Wind();
lB := Wind();

20.0; windB:=20.0;

( now for some dangerous stuff *)
```

```
  IF step>450 THEN windF:=15.0; windB:=15.0;
  ELSE windF:=20.0; windB:=20.0;
  END;
*)

  cosT := Cos(theta);
  sinT := Sin(theta);

  vfX := windF-tdot*rf*sinT; (* front horizontal *)
  vfY := -(hdot+tdot*rf*cosT);  (* front vertical *)

  vbX := windB+tdot*rb*sinT;  (* back horizontal *)
  vbY := -(hdot-tdot*rb*cosT); (* back vertical *)

  gammaf := ATan2(vfX, vfY);
  gammab := ATan2(vbX, vbY);

  CoefficientLift((alpha+theta+gammaf),cLF,dLF);
  CoefficientLift((beta+theta+gammab),cLB,dLB);
  CoefficientDrag((alpha+theta+gammaf),cDF,dDF);
  CoefficientDrag((beta+theta+gammab),cDB,dDB);

  vSQf := sqr(vfX) + sqr(vfY);
  vSQb := sqr(vbX) + sqr(vbY);

  ooopx2f := 1.0/(1.0+sqr(vfY/vfX));
  ooopx2b := 1.0/(1.0+sqr(vbY/vbX));

  cosTgf := Cos(theta+gammaf);
  sinTgf := Sin(theta+gammaf);

  cosTgb := Cos(theta+gammab);
  sinTgb := Sin(theta+gammab);

  cosGf := Cos(gammaf);
  sinGf := Sin(gammaf);

  cosGb := Cos(gammab);
  sinGb := Sin(gammab);

  uturn := uturn0 - af*sqr(rf)*(
        2.0*(-vfX*sinT - vfY*cosT) * (cLF*cosTgf+cDF*sinTgf)
        + vSQf * ((dLF+cDF)*cosTgf + (dDF-cLF)*sinTgf) *
        ooopx2f * (sinT*vfY - cosT*vfX) / sqr(vfX) )
        +
        ab*sqr(rb)*(
        2.0*(vbX*sinT + vbY*cosT) * (cLB*cosTgb+cDB*sinTgb)
        + vSQb * ((dLB+cDB)*cosTgb + (dDB-cLB)*sinTgb)*
        ooopx2b * (-sinT*vbY + cosT*vbX) / sqr(vbX) );

  kt := - af*rf*(-2.0*rf*tdot*
        (vfX*cosT - vfY*sinT) * (cLF*cosTgf+cDF*sinTgf)
        + vSQf * ((dLF+cDF)*cosTgf + (dDF-cLF)*sinTgf)*
        (1.0+ooopx2f*tdot*rf*(cosT*vfY + sinT*vfX) / sqr(vfX) ))

        + ab*rb*( 2.0*tdot*rb*
        (vbX*cosT - vbY*sinT) * (cLB*cosTgb+cDB*sinTgb)
        + vSQb * ((dLB+cDB)*cosTgb + (dDB-cLB)*sinTgb)*
        (1.0-ooopx2b*tdot*rb*(cosT*vbY + sinT*vbX) / sqr(vbX) ));
```

```
 := af*rf*vSQf*(cosTgf*dLF + sinTgf*dDF);
 := ab*rb*vSQb*(cosTgb*dLB + sinTgb*dDB);

 - af*rf*(-2.0*vfY*(cLF*cosTgf+cDF*sinTgf)
 + vSQf * ((dLF+cDF)*cosTgf + (dDF-cLF)*sinTgf)*
 (-ooopx2f)/vfX )
 +
 ab*rb*(-2.0*vbY*(cLB*cosTgb+cDB*sinTgb)
 + vSQb * ((dLB+cDB)*cosTgb + (dDB-cLB)*sinTgb)*
 (-ooopx2b)/vbX );

n lift equ *)
:= af*vSQf*(cosGf*dLF + sinGf*dDF);
:= ab*vSQb*(cosGb*dLB + sinGb*dDB);

= af*(-2.0*rf*tdot*(vfX*cosT - vfY*sinT)*
   (cLF*cosGf+cDF*sinGf)

  + vSQf*((dLF*cosGf+dDF*sinGf)*
    (1.0+ooopx2f*tdot*rf*(cosT*vfY + sinT*vfX) / sqr(vfX))

  + (cDF*cosGf-cLF*sinGf) * ooopx2f
   * tdot*rf*(cosT*vfY + sinT*vfX) / sqr(vfX))    )

  + ab*(2.0*rb*tdot*(vbX*cosT - vbY*sinT)*
 (cLB*cosGb+cDB*sinGb)

 + vSQb*((dLB*cosGb+dDB*sinGb)*
   (1.0-ooopx2b*tdot*rb*(cosT*vbY + sinT*vbX) / sqr(vbX))

 +(cDF*cosGf+cLF*sinGf) * ooopx2b
 *tdot*rb*(-cosT*vbY - sinT*vbX) / sqr(vbX)) );

 af*(-2.0*rf*(vfX*sinT + vfY*cosT) * (cLF*cosGf+cDF*sinGf)
 + vSQf * ((dLF+cDF)*cosGf+(dDF-cLF)*sinGf) * (-ooopx2f)
    *rf*(-sinT*vfY + cosT*vfX) / sqr(vfX))

  + ab*(2.0*rb*(vbX*sinT + vbY*cosT) * (cLB*cosGb+cDB*sinGb)
 + vSQb * ((dLB+cLF)*cosGb+(dDB-cLB)*sinGb) * ooopx2b
   *rb*(-sinT*vbY + cosT*vbX) / sqr(vbX)) ;

 := ulift0 - af*(-2.0*vfY*(cLF*cosGf+cDF*sinGf)
+ vSQf * ((dLF+cDF)*cosGf + (dDF-cLF)*sinGf) * (-ooopx2f) / vfX )
- ab*(-2.0*vbY*(cLB*cosGb+cDB*sinGb)
+ vSQb * ((dLB+cDB)*cosGb + (dDB-cLB)*sinGb) * (-ooopx2b)  / vbX );

ulate I/deltaT - J *)

try(-1.0,                           1, 2, Phi);
try( kt/inertia,                    2, 1, Phi);
= Entry(2,2,Phi);
try((temp + uturn/inertia),2, 2, Phi);
try(-ktf/inertia,                   2, 5, Phi);
try( ktb/inertia,                   2, 7, Phi);
try(-1.0,                           3, 4, Phi);
try(-kl/mass,                       4, 1, Phi);
= Entry(4,4,Phi);
try((temp + ulift/mass),   4, 4, Phi);
try(-klf/mass,                      4, 5, Phi);
```

```
  putEntry(-klb/mass,                      4, 7, Phi);

(* Calculate f *)

  Torque := vSQf*(cLF*cosTgf + cDF*sinTgf) * FrontToPivot * af
          - vSQb*(cLB*cosTgb + cDB*sinTgb) * BackToPivot * ab
          + mass*g*CentreToPivot*cosT;

  TotalLift :=  af*vSQf*(cLF*cosGf+cDF*sinGf)
             + ab*vSQb*(cLB*cosGb+cDB*sinGb) - DeltaMass*g;


(*
  WrLngReal(Torque,3,11);
  WrLngReal(TotalLift,3,11);
*)
  IF disturb THEN
    i := (step MOD DistLength) + 1;
    Torque := Torque + (distLF[i]*cosT + distDF[i]*sinT)*FrontToPivot
                     - (distLB[i]*cosT + distDB[i]*sinT)*BackToPivot;
    TotalLift := TotalLift + distLF[i] + distLB[i];
  END;

(*
  WrLngReal(Entry(4,4,Phi),3,11);
  WrLngReal(ulift,3,11);
*)
  (* Model *)
  f[1] := x[2];
  f[2] := 1.0/inertia*(-uturn0*x[2] + Torque);
  f[3] := x[4];
  f[4] := 1.0/mass*(-ulift0*x[4] + TotalLift);

END Plant1;

(*-----------------------------------------------------------------------*)

VAR
    OneOverT, incompat, temp : LONGREAL;
    rank, row, col : CARDINAL;
    done : BOOLEAN;

BEGIN
  allocateMatrix(Phi,8,8);
  makeZeroMatrix(Phi);
  allocateMatrix(Fmat,8,3);
  makeZeroMatrix(Fmat);

(* Calculate I/deltaT *)
  OneOverT := 1.0/deltaT;
  FOR row := 1 TO 8 DO
    col := row;
    putEntry(OneOverT,row,col,Phi);
  END;

  FrontPosn[step] := x[5];
  FrontRate[step] := x[6];
  theta := x[1];   tdot := x[2];
  height := x[3];  hdot := x[4];
```

```
s;
tl;

;
r('PlantJacobian');
;
row := 1 TO 8 DO
)R col := 1 TO 8 DO
  WrLngReal(Entry(row,col,Phi),2,9);
ND;
rLn;

)w make Phi*[dx¦X] = [f¦F] *)
row := 1 TO 8 DO putEntry(f[row],row,1,Fmat); END;

;(Phi, Fmat, rank, incompat, zero);
ttrixResult.Error THEN
;tr('matrix error in Plant');
.n;

re test MatrixResult *)

w construct [dx¦X] *)
ow := 1 TO 8 DO
[row] := Entry(row,1,Fmat);
mp := Entry(row,2,Fmat);
tEntry(temp,row,1,X);
mp := Entry(row,3,Fmat);
tEntry(temp,row,2,X);

ocateMatrix(Phi);
ocateMatrix(Fmat);

pha+theta > 25.0/180.0*pi THEN
tr('a+t> ');

pha+theta < -5.0/180.0*pi THEN
tr('a+t< ' );

ta+theta > 25.0/180.0*pi THEN
tr('b+t> ');

ta+theta < -5.0/180.0*pi THEN
tr('b+t< ');

it;

------------------------------------------------------------*)

E PlantState(VAR du : vector;
             VAR   height, theta : LONGREAL);
         (* update plant's state vector *)

 CARDINAL;
tRateSaturated, FrontAmplitudeSaturated,
```

```
    BackRateSaturated, BackAmplitudeSaturated    : BOOLEAN;
    temp : ARRAY[1..2] OF LONGREAL;
BEGIN

  WrLn;
  WrStr('PlantState');

  FOR i := 1 TO 8 DO
    temp[1] := Entry(i,1,X);
    temp[2] := Entry(i,2,X);
    x[i] := x[i] + dx[i];
    x[i] := x[i] + temp[1]*du[1];
    x[i] := x[i] + temp[2]*du[2];
(*    WrLngReal(x[i],2,10); *)
  END;

(*
  WrLn;
  FOR i := 1 TO 8 DO
    WrLngReal(dx[i],2,10);
    WrLngReal(Entry(i,1,X),2,10);
    WrLngReal(Entry(i,2,X),2,10);
    WrLngReal(du[1],2,10);
    WrLngReal(du[2],2,10);
    WrLn;
  END;
*)
  (* Now check for saturation *)

  (* front wings *)

  IF (x[5] > maxAlpha) THEN
    FrontAmplitudeSaturated := TRUE;
    x[5] := maxAlpha;
    IF (x[6] > 0.0) THEN x[6] := 0.0 END;
  ELSIF (x[5] < minAlpha) THEN
    FrontAmplitudeSaturated := TRUE;
    x[5] := minAlpha;
    IF (x[6] < 0.0) THEN x[6] := 0.0 END;
  ELSE
    FrontAmplitudeSaturated := FALSE;
  END;

  IF (x[6] > maxAlphaRate) THEN
    FrontRateSaturated := TRUE;
    x[6] := maxAlphaRate;
  ELSIF (x[6] < -maxAlphaRate) THEN
    FrontRateSaturated := TRUE;
    x[6] := -maxAlphaRate;
  ELSE
    FrontRateSaturated := FALSE;
  END;

  (* back wings *)
  IF (x[7] > maxBeta) THEN
    BackAmplitudeSaturated := TRUE;
    x[7] := maxBeta;
    IF (x[8] > 0.0) THEN x[8] := 0.0 END;
  ELSIF (x[7] < minBeta) THEN
    BackAmplitudeSaturated := TRUE;
```

```
[7] := minBeta;
F (x[8] < 0.0) THEN x[8] := 0.0 END;
E
ackAmplitudeSaturated := FALSE;
;

x[8] > maxBetaRate) THEN
ackRateSaturated := TRUE;
8] := maxBetaRate;
F (x[8] < -maxBetaRate) THEN
ckRateSaturated := TRUE;
8] := -maxBetaRate;
:
ckRateSaturated := FALSE;


rontAmplitudeSaturated THEN
rStr ('FAS ');

rontRateSaturated THEN
rStr ('FRS ');

ackAmplitudeSaturated THEN
rStr ('BAS ');

ackRateSaturated THEN
rStr ('BRS ');

;

nt := x[3];
a := x[1];
antState;

--------------------------------------------------------------------*)

RE EndPlant;  (* Must be called before main module ends to
                                              deallocate memory *)

ARDINAL;


('Rate Enter N ');
RdCard();

ToFile(FrontRate,N);
('Posn ');

ToFile(FrontPosn,N);

ocateMatrix(X);
Plant;
--------------------------------------------------------------------*)

 CARDINAL;
utBuffer : ARRAY[1..(1024+FIO.BufferOverhead)] OF BYTE;
utFile   : FIO.File;
eName    : ARRAY[1..40] OF CHAR;

 (* Initialization of MODULE PLANT *)
```

```
WrStr('Initialising PLANT');
WrLn;
WrStr('Plant data  ');
REPEAT
  WrStr('Enter input file ');
  RdStr(FileName);
UNTIL FIO.Exists(FileName);
InputFile := FIO.Open(FileName);
FIO.AssignBuffer(InputFile, InputBuffer);

inertia := FIO.RdLngReal (InputFile);
AreaFront := FIO.RdLngReal(InputFile);
AreaBack := FIO.RdLngReal(InputFile);
uturn0 := FIO.RdLngReal(InputFile);
ulift0 := FIO.RdLngReal(InputFile);
mass := FIO.RdLngReal(InputFile);
CentreToPivot := FIO.RdLngReal(InputFile);
FrontToPivot := FIO.RdLngReal(InputFile);
BackToPivot := FIO.RdLngReal(InputFile);
OmegaFront := FIO.RdLngReal(InputFile);
DampingFront := FIO.RdLngReal(InputFile);
OmegaBack := FIO.RdLngReal(InputFile);
DampingBack := FIO.RdLngReal(InputFile);
windF := FIO.RdLngReal(InputFile);
windB := windF;
FOR i := 1 TO 8 DO
  x[i] := FIO.RdLngReal(InputFile);
END;
(* Read in degrees and seconds *)
minAlpha := FIO.RdLngReal(InputFile);  maxAlpha := FIO.RdLngReal(InputFile);
maxAlphaRate := FIO.RdLngReal(InputFile);
minBeta := FIO.RdLngReal(InputFile);   maxBeta := FIO.RdLngReal(InputFile);
maxBetaRate := FIO.RdLngReal(InputFile);

FIO.Close(InputFile);

(* Do some conversions *)
minAlpha := minAlpha/180.0*pi;
maxAlpha := maxAlpha/180.0*pi;
minBeta := minBeta/180.0*pi;
maxBeta := maxBeta/180.0*pi;
maxAlphaRate := maxAlphaRate/180.0*pi;
maxBetaRate := maxBetaRate/180.0*pi;

OmegaSqFront := OmegaFront*OmegaFront;
TwoOmegaDFront := OmegaFront*DampingFront*2.0;

OmegaSqBack := OmegaBack*OmegaBack;
TwoOmegaDBack := OmegaBack*DampingBack*2.0;


WrStr ('Front lift dist ');
REPEAT
  WrStr('Enter input file ');
  RdStr(FileName);
UNTIL FIO.Exists(FileName);
InputFile := FIO.Open(FileName);
FIO.EOF := FALSE;
FIO.AssignBuffer(InputFile, InputBuffer);
```

```
 i := 1 TO DistLength DO
IF NOT(FIO.EOF) THEN
  distLF[i] := FIO.RdLngReal(InputFile);
ELSE
  distLF[i] := 0.0;
END;

.Close(InputFile);

r ('Front drag dist ');
AT
Str('Enter input file ');
Str(FileName);
L FIO.Exists(FileName);
tFile := FIO.Open(FileName);
EOF := FALSE;
AssignBuffer(InputFile, InputBuffer);
i := 1 TO DistLength DO
F NOT(FIO.EOF) THEN
 distDF[i] := FIO.RdLngReal(InputFile);
LSE
 distDF[i] := 0.0;
ND;

Close(InputFile);

r ('Back lift dist ');
AT
Str('Enter input file ');
Str(FileName);
L FIO.Exists(FileName);
tFile := FIO.Open(FileName);
EOF := FALSE;
AssignBuffer(InputFile, InputBuffer);
i := 1 TO DistLength DO
 NOT(FIO.EOF) THEN
 distLB[i] := FIO.RdLngReal(InputFile);
SE
 distLB[i] := 0.0;
D;

lose(InputFile);

 ('Back drag dist ');
T
 tr('Enter input file ');
 tr(FileName);
  FIO.Exists(FileName);
 File := FIO.Open(FileName);
 OF := FALSE;
 ssignBuffer(InputFile, InputBuffer);
  := 1 TO DistLength DO
  NOT(FIO.EOF) THEN
 distDB[i] := FIO.RdLngReal(InputFile);
 SE
 distDB[i] := 0.0;
 D;

 lose(InputFile);
```

```
  NextWind := 1;

  allocateMatrix(X,8,2);

END PLANTend.
(*
  inertia := 0.143;           (* Moment of inertia about pivot*)
  AreaFront := 0.0529; (* Total area of front wings *)
  AreaBack := 0.16;    (* Total area of back wings m²*)
  uturn := 0.11;       (* Rotational friction *)
  ulift := 6.6;        (* Sliding friction *)
  mass := 1.5;         (* Kg *)
  CentreToPivot := 0.01; (* pivot in front of centre of mass *)
  FrontToPivot := 0.43;  (* m *)
  BackToPivot := 0.24;   (* m *)

  AlphaOffset := 0.123;
  BetaOffset := 0.062;
*)
```

```
ENTATION MODULE CONTROLLERS;

 FIO;
) IMPORT RdLngReal, WrLngReal, RdStr, RdCard, WrLn,  WrStr;

atrix IMPORT RealMatrix, allocateMatrix, deallocateMatrix, Entry, putEntry

              MatrixResult, writeMatrix,
              makeZeroMatrix, makeIdentityMatrix,
              (* addMatrix, subtractMatrix,  multiplyMatrix,*)  Gauss;

lyStart IMPORT vector, pi, zero;

:Order = 3;

rameter = ARRAY[0..(MaxOrder-1)] OF LONGREAL;

 vector;
 bT, aH, bH : parameter;

RE Controllers(deltaT, Utheta, Uheight : LONGREAL);

, Fmat : RealMatrix;
 vector;
, col, index : CARDINAL;
OverT : LONGREAL;
ompat : LONGREAL;
k : CARDINAL;
e : BOOLEAN;

ateMatrix(Phi,OrderTheta+OrderHeight,OrderTheta+OrderHeight);
trixResult.Error THEN
Str('Failed to allocate PLANT.Phi');

ateMatrix(Fmat,OrderTheta+OrderHeight,3);
trixResult.Error THEN
Str('Failed to allocate PLANT.Fmat');

eroMatrix(Phi);

rT := 1.0/deltaT;
w := 1 TO OrderTheta+OrderHeight DO
:= row;
ntry(OneOverT,row,col,Phi);

troller theta *)
w := 2 TO OrderTheta DO
:= row-1;
ntry(-1.0,row,col,Phi);

w := 1 TO OrderTheta-1 DO
ntry(aT[row-1],row,OrderTheta,Phi);

ry ((Entry(OrderTheta,OrderTheta, Phi) + aT[OrderTheta-1]),
          OrderTheta,OrderTheta,Phi);
```

```
  f[1] := -aT[0]*x[OrderTheta] + bT[0]*Utheta;
  FOR row := 2 TO OrderTheta DO
    f[row] := x[row-1] - aT[row-1]*x[OrderTheta] + bT[row-1]*Utheta;
  END;

  (* Controller height *)
  FOR row := OrderTheta+2 TO OrderTheta+OrderHeight DO
    col := row-1;
    putEntry(-1.0,row,col,Phi);
  END;
  FOR row := OrderTheta+1 TO OrderTheta+OrderHeight-1 DO
    index := row-1-OrderTheta;
    putEntry(aH[index],row,OrderTheta+OrderHeight,Phi);
  END;
  index := OrderTheta+OrderHeight;
  putEntry ((Entry(index, index, Phi) + aH[OrderHeight-1]),
                                          index, index, Phi);

  f[OrderTheta+1] := -aH[0]*x[OrderTheta+OrderHeight] + bH[0]*Uheight;
  FOR row := OrderTheta+2 TO OrderTheta+OrderHeight DO
    index := row-1-OrderTheta;
    f[row] := x[row-1] - aH[index]*x[OrderTheta+OrderHeight]
                      + bH[index]*Uheight;
  END;

  (* Now make Phi*[dx|X] = [f|F] *)
  makeZeroMatrix(Fmat);
  FOR row := 1 TO OrderTheta+OrderHeight DO
    putEntry(f[row],row,1,Fmat);
  END;
  FOR row := 1 TO OrderTheta DO
    putEntry(bT[row-1],row,2,Fmat);
  END;

  FOR row := OrderTheta+1 TO OrderTheta+OrderHeight DO
    index := row-OrderTheta-1;
    putEntry(bH[index],row,3,Fmat);
  END;

(*  AssignWrite(print,done);

  WrStr('In controller');
  WrLn;
  writeMatrix(Phi,'d');
  writeMatrix(Fmat,'d');
  UnAssignWrite(done);
*)
  Gauss(Phi, Fmat, rank, incompat, zero);
  IF MatrixResult.Error THEN
    WrStr('matrix error in Controllers');
    WrLn;
  END;

(*  AssignWrite(print,done);

  writeMatrix(Fmat,'d');
  WrLn;
  UnAssignWrite(done);
*)
```

```
, construct [dx|X] *)
ow := 1 TO OrderTheta+OrderHeight DO
[row] := Entry(row,1,Fmat);

LngReal(dx[i],9); *)

Entry(Entry(row,2,Fmat),row,1,X);
Entry(Entry(row,3,Fmat),row,2,X);

; *)
cateMatrix(Phi);
cateMatrix(Fmat);

rollers;

-------------------------------------------------------------*)
E ControllerState(VAR du : vector;
                  VAR AplusThat, BplusThat : LONGREAL);
                     (* update controllers' state vector *)

CARDINAL;

r('ControllerState');
;

:= 1 TO OrderTheta+OrderHeight DO
 := x[i] + dx[i] + Entry(i,1,X)*du[1] + Entry(i,2,X)*du[2];
LngReal(x[i],9);
Str(' ');

; *)

hat := Kinv[1,1]*x[OrderTheta] + Kinv[1,2]*x[OrderTheta+OrderHeight];
hat := Kinv[2,1]*x[OrderTheta] + Kinv[2,2]*x[OrderTheta+OrderHeight];

rollerState;
-------------------------------------------------------------*)

 EndControl;     (* must be called before end of main module to
                                       deallocate memory *)

ateMatrix(X);
ntrol;
--------------------------------------------------------------*)


 : CARDINAL;
Buffer : ARRAY[1..(1024+FIO.BufferOverhead)] OF BYTE;
File   : FIO.File;
ame    : ARRAY[1..40] OF CHAR;

Initialising CONTROLLERS');

Controller data  ');

('Enter input file ');
(FileName);
```

```
  UNTIL FIO.Exists(FileName);
  InputFile := FIO.Open(FileName);
  FIO.AssignBuffer(InputFile, InputBuffer);
  OrderTheta := FIO.RdCard(InputFile);
  OrderHeight := FIO.RdCard(InputFile);
  FOR i:= 0 TO OrderTheta-1 DO
    bT[i] := FIO.RdLngReal(InputFile);
  END;
  FOR i:= 0 TO OrderTheta-1 DO
    aT[i] := FIO.RdLngReal(InputFile);
  END;

  FOR i:= 0 TO OrderHeight-1 DO
    bH[i] := FIO.RdLngReal(InputFile);
  END;
  FOR i:= 0 TO OrderHeight-1 DO
    aH[i] := FIO.RdLngReal(InputFile);
  END;

  FOR i:= 1 TO 2 DO
    FOR j:= 1 TO 2 DO
      Kinv[i,j] := FIO.RdLngReal(InputFile);
    END;
  END;

  FOR i := 1 TO OrderTheta+OrderHeight DO
    x[i] := FIO.RdLngReal(InputFile);
  END;
  FIO.Close(InputFile);

  allocateMatrix(X,OrderTheta+OrderHeight,2);

END CONTROLLERS.
```

```
orfly;

T FloatExc;
T FIO;
IO IMPORT RdLngReal, WrLngReal, RdStr, RdCard, WrLn,  WrStr, WrCard;

MATHLIB IMPORT Exp;

Matrix IMPORT RealMatrix, allocateMatrix, deallocateMatrix, Entry, putEnt

              MatrixResult, writeMatrix,
              makeZeroMatrix, makeIdentityMatrix,
              (* addMatrix, subtractMatrix,  multiplyMatrix,*)  Gauss;

FlyStart IMPORT pi, zero, vector;
WriteToF IMPORT WriteToFile, results;

T HorCtl;
T PLANTend;  (* The "end" refers to my tether *)

-----------------------------------------------------------*)
RE Global(VAR PLANTdu, CONTROLdu : vector);

 solve : du[i] - du[i]/dx[j]*X[j]*du[j] = du[i]/dx[j]*dx[j] *)

 : RealMatrix;
Theta, entHeight, rank : CARDINAL;
mpat, temp : LONGREAL;


 = b is partitioned : [    I(p)     -du/dx*X(c) | du/dx*dx(c)]
                      [-du/dx*X(p)     I(c)     | du/dx*dx(p)]   *)

teMatrix (A, 4, 4);
teMatrix (b, 4, 1);
entityMatrix(A);
ta := HorCtl.OrderTheta;
ght := entTheta + HorCtl.OrderHeight;

 row of A *)
ry( -Entry(1,entHeight,HorCtl.X)    , 1, 3, A);
ry( -Entry(2,entHeight,HorCtl.X)    , 1, 4, A);

 row of A *)
ry( -Entry(1,entTheta,HorCtl.X)    , 2, 3, A);
ry( -Entry(2,entTheta,HorCtl.X)    , 2, 4, A);

 row of A *)
ry( Entry(1,1,PLANTend.X)     , 3, 1, A);
ry( Entry(1,2,PLANTend.X)     , 3, 2, A);

 row of A *)
ry( Entry(3,1,PLANTend.X)     , 4, 1, A);
ry( Entry(3,2,PLANTend.X)     , 4, 2, A);

)
y( -HorCtl.dx[entHeight] , 3, 1, b);
y( -HorCtl.dx[entTheta]  , 2, 1, b);
y( -PLANTend.dx[1]            , 3, 1, b);
y( -PLANTend.dx[3]            , 4, 1, b);
```

```
  Gauss(A, b, rank, incompat, zero);
  IF MatrixResult.Error THEN
    WrStr('matrix error in Global');
    WrLn;
  END;
  PLANTdu[1] := Entry(1,1,b);
  PLANTdu[2] := Entry(2,1,b);
  CONTROLdu[1] := Entry(3,1,b);
  CONTROLdu[2] := Entry(4,1,b);

  deallocateMatrix(A);
  deallocateMatrix(b);

END Global;
(*-----------------------------------------------------------*)

VAR
    PLANTdu, CONTROLdu : vector;
    nexttick, time, ticklength, deltaT,
    ffilter, href, tref, theta, height, alphaIn, betaIn,
                          CtlThetaOut, CtlHeightOut : LONGREAL;
    ares, bres, hres, tres : results;
    NumberSteps, step : CARDINAL;
    tick : BOOLEAN;
BEGIN

  WrStr('Initialising FlyingMachine');
  WrLn;
  WrStr('Enter deltaT ');
  deltaT := RdLngReal();
  WrLn;
  WrStr('Enter NumberSteps ');
  NumberSteps := RdCard();
  WrLn;
  WrStr('Tick length ');
  ticklength := RdLngReal();
  WrLn;

  nexttick := 0.0;
  time := 0.0;

  alphaIn :=0.0;
  betaIn := 0.0;

  theta := 0.0;
  height := 0.0;
  ffilter := 0.0;

  FOR step := 1 TO NumberSteps DO
    WrStr('step ');
    WrCard(step,4);
    time := time + deltaT;
    IF time >= nexttick THEN
      tick:=TRUE;
      nexttick := time+ticklength;
    ELSE
      tick:=FALSE;
    END;

    hres[step] := height;
```

```
s[step] := theta*180.0/pi;
s[step] := (alphaIn+0.027)*180.0/pi;
s[step] := (betaIn+0.014)*180.0/pi;


tr('Alpha '); WrLngReal(alphaIn,5,10);
tr('Beta ');WrLngReal(betaIn,5,10);WrLn;

NTend.Plant(deltaT, (alphaIn+0.027), (betaIn+0.014), step, tick, TRUE);
f := 0.0;
f := 0.0;

f :=0.0;
time > 11.0 THEN
href := 0.75*(1.0-Exp(-0.7*(time-11.0)));
;

Ctl.Controllers(deltaT, tref-theta,  href-height);

bal(PLANTdu, CONTROLdu);

NTend.PlantState(PLANTdu, height, theta);

Ctl.ControllerState(CONTROLdu, CtlThetaOut, CtlHeightOut);
r('Theta '); WrLngReal(CtlThetaOut,3,10); WrLngReal(theta/pi*180.0,3,10)

r('Height ');WrLngReal(CtlHeightOut,3,10);WrLn;

aIn := CtlHeightOut;
In := CtlThetaOut;




'Height');
oFile(hres,NumberSteps);

'THETA ');
oFile(tres,NumberSteps);

'ALPHA ');
oFile(ares,NumberSteps);

'BETA ');
oFile(bres,NumberSteps);

nd.EndPlant;

.EndControl;

ly.
```

```
DEFINITION MODULE HorCtl;

FROM Matrix IMPORT RealMatrix;
FROM FlyStart IMPORT vector;

VAR
    X : RealMatrix;
    dx : vector;
    OrderTheta, OrderHeight : CARDINAL;

PROCEDURE Controllers(deltaT, Utheta, Uheight : LONGREAL);
PROCEDURE ControllerState(VAR du : vector;
                          VAR CtlThetaOut, CtlHeightOut : LONGREAL);
                              (* update controllers' state vector *)
PROCEDURE EndControl;    (* must be called before end of main module to
                                                  deallocate memory *)

END HorCtl.
```

```
ENTATION MODULE HorCtl;

 FIO;

) IMPORT RdLngReal, WrLngReal, RdStr, RdCard, WrLn,  WrStr;

atrix IMPORT RealMatrix, allocateMatrix, deallocateMatrix, Entry, putEntry

              MatrixResult, writeMatrix,
              makeZeroMatrix, makeIdentityMatrix,
              (* addMatrix, subtractMatrix,  multiplyMatrix,*)  Gauss;

yStart IMPORT vector, pi, zero;

Order = 3;

ameter = ARRAY[0..(MaxOrder-1)] OF LONGREAL;

 vector;
 bT, aH, bH : parameter;

RE Controllers(deltaT, Utheta, Uheight : LONGREAL);

, Fmat : RealMatrix;
 vector;
, col, index : CARDINAL;
OverT : LONGREAL;
ompat : LONGREAL;
k : CARDINAL;
e : BOOLEAN;

ateMatrix(Phi,OrderTheta+OrderHeight,OrderTheta+OrderHeight);
rixResult.Error THEN
Str('Failed to allocate PLANT.Phi');

ateMatrix(Fmat,OrderTheta+OrderHeight,3);
rixResult.Error THEN
Str('Failed to allocate PLANT.Fmat');

roMatrix(Phi);

rT := 1.0/deltaT;
w := 1 TO OrderTheta+OrderHeight DO
:= row;
ntry(OneOverT,row,col,Phi);

troller theta *)
w := 2 TO OrderTheta DO
:= row-1;
ntry(-1.0,row,col,Phi);

w := 1 TO OrderTheta-1 DO
ntry(aT[row-1],row,OrderTheta,Phi);

ry ((Entry(OrderTheta,OrderTheta, Phi) + aT[OrderTheta-1]),
```

```
                   OrderTheta,OrderTheta,Phi);

f[1] := -aT[0]*x[OrderTheta] + bT[0]*Utheta;
FOR row := 2 TO OrderTheta DO
  f[row] := x[row-1] - aT[row-1]*x[OrderTheta] + bT[row-1]*Utheta;
END;

(* Controller height *)
FOR row := OrderTheta+2 TO OrderTheta+OrderHeight DO
  col := row-1;
  putEntry(-1.0,row,col,Phi);
END;
FOR row := OrderTheta+1 TO OrderTheta+OrderHeight-1 DO
  index := row-1-OrderTheta;
  putEntry(aH[index],row,OrderTheta+OrderHeight,Phi);
END;
index := OrderTheta+OrderHeight;
putEntry ((Entry(index, index, Phi) + aH[OrderHeight-1]),
                                     index, index, Phi);

f[OrderTheta+1] := -aH[0]*x[OrderTheta+OrderHeight] + bH[0]*Uheight;
FOR row := OrderTheta+2 TO OrderTheta+OrderHeight DO
  index := row-1-OrderTheta;
  f[row] := x[row-1] - aH[index]*x[OrderTheta+OrderHeight]
                     + bH[index]*Uheight;
END;

(* Now make Phi*[dx|X] = [f|F] *)
makeZeroMatrix(Fmat);
FOR row := 1 TO OrderTheta+OrderHeight DO
  putEntry(f[row],row,1,Fmat);
END;
FOR row := 1 TO OrderTheta DO
  putEntry(bT[row-1],row,2,Fmat);
END;

FOR row := OrderTheta+1 TO OrderTheta+OrderHeight DO
  index := row-OrderTheta-1;
  putEntry(bH[index],row,3,Fmat);
END;

Gauss(Phi, Fmat, rank, incompat, zero);
IF MatrixResult.Error THEN
  WrStr('matrix error in Controllers');
  WrLn;
END;

(* now construct [dx|X] *)
FOR row := 1 TO OrderTheta+OrderHeight DO
   dx[row] := Entry(row,1,Fmat);


   putEntry(Entry(row,2,Fmat),row,1,X);
   putEntry(Entry(row,3,Fmat),row,2,X);
END;
deallocateMatrix(Phi);
deallocateMatrix(Fmat);

END Controllers;
```

```
-------------------------------------------------------------*)
RE ControllerState(VAR du : vector;
                        VAR CtlThetaOut, CtlHeightOut : LONGREAL);
                            (* update controllers' state vector *)

 CARDINAL;

tr('ControllerState');
n;

 := 1 TO OrderTheta+OrderHeight DO
] := x[i] + dx[i] + Entry(i,1,X)*du[1] + Entry(i,2,X)*du[2];
rLngReal(x[i],9);
rStr(' ');


n; *)

etaOut := x[OrderTheta];
ightOut := x[OrderTheta+OrderHeight];

trollerState;
-------------------------------------------------------------*)

RE EndControl;     (* must be called before end of main module to
                                          deallocate memory *)

ocateMatrix(X);
ontrol;
---------------------------------------------------------------*)


 : CARDINAL;
tBuffer : ARRAY[1..(1024+FIO.BufferOverhead)] OF BYTE;
tFile   : FIO.File;
Name    : ARRAY[1..40] OF CHAR;


'Initialising CONTROLLERS');

'Controller data  ');

r('Enter input file ');
r(FileName);
FIO.Exists(FileName);
ile := FIO.Open(FileName);
signBuffer(InputFile, InputBuffer);
heta := FIO.RdCard(InputFile);
eight := FIO.RdCard(InputFile);
= 0 TO OrderTheta-1 DO
] := FIO.RdLngReal(InputFile);

= 0 TO OrderTheta-1 DO
] := FIO.RdLngReal(InputFile);


: 0 TO OrderHeight-1 DO
] := FIO.RdLngReal(InputFile);
```

```
  FOR i:= 0 TO OrderHeight-1 DO
    aH[i] := FIO.RdLngReal(InputFile);
  END;

  FOR i := 1 TO OrderTheta+OrderHeight DO
    x[i] := FIO.RdLngReal(InputFile);
  END;
  FIO.Close(InputFile);

  allocateMatrix(X,OrderTheta+OrderHeight,2);

END HorCtl.
```