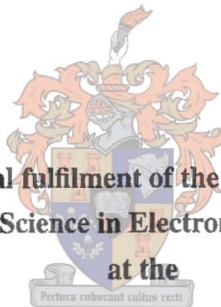


The Instrumentation and Initial Analysis of the Short-term Control and Stability Derivatives of an ASK-13 Glider

**Thesis presented in partial fulfilment of the requirements for the degree of
Master of Science in Electronic Engineering**



at the

University of Stellenbosch

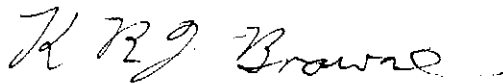
KEITH R.J. BROWNE

Supervisor: Prof. G.W. Milne

April, 2004

Declaration

I the undersigned, hereby declare that the work contained in this thesis is my own original work, unless otherwise stated, and has not previously, in its entirety or in part, been submitted at any university for a degree.

A handwritten signature in black ink that reads "K R J Browne". The letters are cursive and somewhat stylized.

Keith R.J. Browne April, 2004

Abstract

This thesis describes the process followed to determine the short-term control and stability derivatives of an ASK-13 glider (ZS-GHB). The short-term control and stability derivatives are obtained by parameter estimation done using data recorded in flight. The algorithm used is the MMLE3 implementation of a maximum likelihood estimator.

To collect the flight data sensors were installed in the ZS-GHB. Sensors measuring the control surface deflections, translation acceleration, angular rates and the dynamic and static pressure are needed to provide enough data for the estimation. To estimate accurate derivatives specific manoeuvres were flown by the pilot, to ensure that all the modes of the glider were stimulated.

The results reveal that the control and stability derivatives estimated from the flight data are not very accurate but are still suitable to be used in simulating the glider's motion.

Opsomming

Hierdie tesis beskryf die proses wat gebruik is om die kort periode beheer en stabiliteit afgeleides van 'n ASK-13 sweeftuig vas te stel. Die kort periode beheer en stabiliteit afgeleides is verkry deur parameter afskatting op data wat gedurende vlugte van die sweeftuig opgeneem is. Die algoritme wat gebruik is om die parameters af te skat is die MMLE3 voorstelling van 'n maksimale moontlikheid afskatter.

Om vlug data te versamel sensore moes in die sweeftuig geïnstalleer word. Die sensore meet beheer oppervlak hoeke, versnellings, hoeksnellhede en die dinamies en statiese lugdruk om te verseker dat daar genoeg data is vir die afskatting. Om die afgeskatte parameters akkuraad te kry moet die loods spesifieke manoeuvres vlieg om seker te maak dat al die moduse van die sweeftuig is gestimuleer.

Die resultate wat gelewer is 'n stel kort periode beheer en stabiliteit afgeleides wat nie akkuraad is nie, maar wat wel goed genoeg is om die bewegings van die sweeftuig te simuleer.

Acknowledgements

I would like to acknowledge the following people, who have had a large influence on this project.

Prof. G.W. Milne For providing the opportunity and piloting ZS-GHB.

Cape Gliding Club For the use of ZS-GHB.

G. Treurnicht For help manufacturing the mechanics of the sensors.

E. Strumpfer For measurement ideas and assistance.

The Hendriks family For their support.

My Family For their continued love and support.

Contents

Declaration	i
Abstract	ii
Opsoming	iii
Acknowledgements	iv
List of Symbols	xiii
1 Introduction	1
2 System Identification	3
2.1 Perspective on System identification	3
2.2 Maximum Likelihood Estimation	3
2.3 ML Estimation for Gaussian State-Space Models	4
2.4 Minimizing the <i>LLF</i>	6
2.4.1 Gauss-Newton Approximation of the Hessian	7
2.4.2 Levenberg-Marquardt Stabilization	7
3 Parameter Accuracy Indicators and Analysis Tools	9
3.1 Scatter Plot	10
3.1.1 Cramer-Rao Bound Theory	11
3.1.2 The Filtered Cramer-Rao Bound	11
3.2 The Confidence Ellipsoid	12
3.3 Using the Accuracy Indicators	14
3.3.1 Matching Responses	14
3.3.2 Scatter Plots	15
3.3.3 Final Check	17
4 Equations of Motion	18
4.1 Axes Systems	18
4.1.1 Leading Edge Datum	18

4.1.2	Body Axes	18
4.1.3	Wind Axes	19
4.1.4	Control Surfaces	20
4.2	Equations of Motion	21
4.3	Identifying the Parameters to be Estimated	23
5	Glider Instruments	24
5.1	Introduction	24
5.2	Desirable Characteristics of Sensors Producing Data for Estimation	24
5.2.1	Data Logger Characteristics	24
5.2.2	Desirable Transducer Characteristics	25
5.2.3	Conditions for Sensors with regards to flight safety	26
5.3	Data Logger	27
5.3.1	Signal Conditioning	27
5.3.2	Sampling	27
5.3.3	Logging	28
5.4	Control Surface Transducers	29
5.4.1	Ailerons	30
5.4.2	Elevator	33
5.4.3	Rudder	34
5.4.4	Trim	36
5.4.5	Air-brakes	38
5.5	Air data	39
5.5.1	Dynamic Pressure	39
5.5.2	Static pressure	40
5.6	Inertial Sensors	41
5.6.1	Angular Rate Sensors	41
5.6.2	Accelerometers	43
5.6.3	Mounting the Inertial Sensors	44
5.6.4	Calibrating the Mounted sensors	45
5.6.5	Scale Factor Adjustment	46
6	Preflight Data	48
6.1	Readily Available Data	48
6.2	Center of gravity and accelerometer position	49
6.2.1	Vertical Center of Gravity	49
6.2.2	Accelerometer positions	51
6.3	Mass Properties	52
6.3.1	Pitch Moment of Inertia	52

6.3.2	Roll Moment of Inertia	54
6.3.3	Yaw Moment of Inertia	57
6.3.4	Products of Inertia	60
6.3.5	Taking the Pilot into account	61
7	Lateral Derivative Estimation	63
7.1	Flight Test Procedure	63
7.2	Lateral Equations of Motion in a State-Space Model	65
7.2.1	Lateral Equations of Motion	65
7.2.2	State-space form and additional inputs	66
7.2.3	Output Equations	67
7.3	Estimation of Parameters	68
7.3.1	Pre-Processing of the data	68
7.3.2	Initial Parameters Values	68
7.3.3	Parameter Estimation Structure	69
7.4	Results	70
7.4.1	Initial Parameters and Convergence Issues	71
7.4.2	Scatter Plots	72
7.4.3	Combining the Parameters from Different Manoeuvres	78
7.4.4	Matching the Measured Data	79
8	Longitudinal Derivative Estimation	81
8.1	Longitudinal Equations of Motion in State-Space Model	81
8.1.1	Longitudinal equations of motion	81
8.1.2	State-space form and additional inputs	82
8.1.3	Output Equations	83
8.2	Estimation of Parameters	83
8.2.1	Parameter estimation structure	84
8.3	Results	84
8.3.1	Scatter Plots	85
8.3.2	Matching the Measured Data	90
9	Conclusions and Recommendations	93
9.1	Overview	93
9.2	Credibility of Estimated Parameters	93
9.3	Recommendations	94
9.4	Conclusion	95

A Lateral Manoeuvres	98
A.1 Flight 1	99
A.2 Flight 2	109
A.3 Flight 3	119
B Longitudinal Manoeuvres	129
B.1 Flight 1	130
B.2 Flight 2	135
B.3 Flight 3	141
C Circuit Schematics	155
D Program Code	160
D.1 Delphi Code	160
D.1.1 Code listing of: Seriallog.pas	160
D.1.2 Code listing of: Postflight.pas	162
D.2 Matlab Scripts	164
D.2.1 Preprocessing	164
D.2.2 Lateral Estimation	166
D.2.3 Longitudinal Estimation	178

List of Figures

1.1	The glider ZS-GHB	1
2.1	System identification diagram	5
3.1	Example of a Scatter Plot	10
3.2	Geometric Relationships between Confidence Ellipsoid and Accuracy Indicators	13
3.3	Good, Reasonable and Bad Response matches	15
3.4	Possible Scatter Plot Patterns	16
3.5	Percentage of Scatter from the Mean Estimated Value	16
4.1	Body Axes	19
4.2	Wind Axes	20
5.1	Flow of Data	27
5.2	Signal Conditioning Block Diagram	27
5.3	IMU mounted inside ZS-GHB	29
5.4	Aileron deflection sensor	31
5.5	Left aileron calibration measurements	31
5.6	Combined aileron deflection	32
5.7	Elevator deflection sensor	33
5.8	Elevator calibration measurements	33
5.9	Rudder deflection sensor	34
5.10	Diagram of Rudder deflection measurement	35
5.11	Rudder calibration measurements	36
5.12	Trim-tab Deflection sensor	37
5.13	Trim-tab calibration data	37
5.14	Air-brake extension sensor	38
5.15	Air-brake calibration data	39
5.16	Mounting for the inertial sensors and logging circuitry	44
6.1	Torque and moment arm diagram with glider leaning on its wing	50
6.2	Triangle for calculating tilt angle	50

6.3	Pitch moment of inertia experiment setup	52
6.4	Force diagram for pitch inertia experiments	53
6.5	Pitch moment of inertia scatter plot	54
6.6	Force diagram for roll inertia experiments	55
6.7	Roll moment of inertia scatter plot	56
6.8	Curve fitting of a roll inertia experiment	56
6.9	Turnstile for Yaw moment of inertia experiments	57
6.10	The rollers used to free the tail wheel	58
6.11	Scatter plot of Yaw moment of inertia estimations	59
6.12	Percent deviation from weighted mean of estimated yaw moments of inertia.	59
7.1	Typical manoeuvre shapes.	64
7.2	MLE fit of Flight data with pulse input.	70
7.3	MLE fit of Flight data with doublet input.	71
7.4	MLE fit of Flight data with four frequency input.	71
7.5	Scatter plot of C_{l_β} (Roll due to sideslip).	72
7.6	Scatter plot of $C_{l_{\delta_a}}$ (Roll due to aileron deflection).	73
7.7	Scatter plot of $C_{l_{\delta_r}}$ (Roll due to rudder deflection).	73
7.8	Scatter plot of C_{l_p} (Roll damping)	74
7.9	Scatter plot of C_{l_r} (Roll due to yaw rate).	74
7.10	Scatter plot of C_{n_β} (Yaw due to sideslip angle).	75
7.11	Scatter plot of $C_{n_{\delta_a}}$ (Yaw due to aileron deflection).	75
7.12	Scatter plot of $C_{n_{\delta_r}}$ (Yaw due to rudder deflection).	76
7.13	Scatter plot of C_{n_p} (Yaw due to roll rate).	76
7.14	Scatter plot of C_{n_r} (Yaw damping).	77
7.15	Scatter plot of C_{Y_β} (Side force due to sideslip angle).	77
7.16	Scatter plot of $C_{Y_{\delta_r}}$ (Side force due to rudder deflection.	78
7.17	Weighted mean model responses compared to measured data.	80
8.1	MLE fit of Flight data with pulse input.	84
8.2	MLE fit of Flight data with doublet input.	85
8.3	MLE fit of Flight data with four frequency input.	85
8.4	Scatter plot of C_{N_α} (Normal force due to angle of attack).	86
8.5	Scatter plot of $C_{N_{\delta_e}}$ (Normal force due to elevator deflection).	86
8.6	Scatter plot of C_{A_α} (Axial force due to angle of attack).	87
8.7	Scatter plot of $C_{A_{\delta_e}}$ (Normal force due to elevator deflection).	87
8.8	Scatter plot of C_{m_α} (Pitch rate due to angle of attack.	88
8.9	Scatter plot of $C_{m_{\delta_e}}$ (Pitch rate due to elevator deflection).	88
8.10	Scatter plot of C_{m_q} (Pitch rate damping).	89

8.11 Simulation with weighted mean parameters compared to flight data	90
8.12 Simulation with new weighted mean parameters compared to flight data	91
A.1 Manoeuvre no.1	99
A.2 Manoeuvre no.2	100
A.3 Manoeuvre no.3	101
A.4 Manoeuvre no.4	102
A.5 Manoeuvre no.5	103
A.6 Manoeuvre no.6	104
A.7 Manoeuvre no.7	105
A.8 Manoeuvre no.8	106
A.9 Manoeuvre no.9	107
A.10 Manoeuvre no.10	108
A.11 Manoeuvre no.11	109
A.12 Manoeuvre no.12	110
A.13 Manoeuvre no.13	111
A.14 Manoeuvre no.14	112
A.15 Manoeuvre no.15	113
A.16 Manoeuvre no.16	114
A.17 Manoeuvre no.17	115
A.18 Manoeuvre no.18	116
A.19 Manoeuvre no.19	117
A.20 Manoeuvre no.20	118
A.21 Manoeuvre no.21	119
A.22 Manoeuvre no.22	120
A.23 Manoeuvre no.23	121
A.24 Manoeuvre no.24	122
A.25 Manoeuvre no.25	123
A.26 Manoeuvre no.26	124
A.27 Manoeuvre no.27	125
A.28 Manoeuvre no.28	126
A.29 Manoeuvre no.29	127
A.30 Manoeuvre no.29	128
B.1 Manoeuvre no.1	130
B.2 Manoeuvre no.2	131
B.3 Manoeuvre no.3	132
B.4 Manoeuvre no.4	133
B.5 Manoeuvre no.5	134

B.6	Manoeuvre no.5	135
B.7	Manoeuvre no.6	136
B.8	Manoeuvre no.7	137
B.9	Manoeuvre no.8	138
B.10	Manoeuvre no.9	139
B.11	Manoeuvre no.10	140
B.12	Manoeuvre no.11	141
B.13	Manoeuvre no.12	142
B.14	Manoeuvre no.13	143
B.15	Manoeuvre no.14	144
B.16	Manoeuvre no.15	145
B.17	Manoeuvre no.16	146
B.18	Manoeuvre no.17	147
B.19	Manoeuvre no.18	148
B.20	Manoeuvre no.19	149
B.21	Manoeuvre no.20	150
B.22	Manoeuvre no.21	151
B.23	Manoeuvre no.22	152
B.24	Manoeuvre no.23	153
B.25	Manoeuvre no.24	154
C.1	Schematic for Signal Conditioning	156
C.2	Schematic for Pressure Sensors	157
C.3	Schematic for Inertial Measurement	158
C.4	Schematic for Control Surface Deflection Sensors	159

List of Symbols

a_n	normal acceleration, g
a_x, a_y, a_z	longitudinal, lateral, and vertical acceleration, g
b	reference span, m
C	state-space state output matrix
C_l, C_m, C_n	coefficients of roll, pitch, and yaw moment
C_L, C_D	coefficients of lift and drag
C_N, C_A	coefficients of normal and axial force
C_X, C_Y, C_Z	coefficients of longitudinal, lateral, and vertical force
c	reference chord, m
F	external applied force, N
F_X, F_Y, F_Z	components of external applied force, N
g	acceleration due to gravity, $m.s^{-2}$
hes	the approximation of the Hessian
I	identity matrix
I_{xx}, I_{yy}, I_{zz}	moments of inertia, $kg.m^2$
I_{xy}, I_{xz}, I_{yz}	cross products of inertia, $kg.m^2$
J	cost function
m	mass, kg
p	roll rate, $rad.s_{-1}$
q	pitch rate, $rad.s_{-1}$
\bar{q}	dynamic pressure, $N.m^2$
RR^T	innovation covariance
\widehat{RR}^T	sample innovation covariance
r	yaw rate, $rad.s_{-1}$
S	reference area, m^2
u	system input; or body X-axis wind-relative velocity, $m.s^{-1}$
V	total wind-relative velocity, $m.s^{-1}$
v	body Y-axis wind-relative velocity, $m.s^{-1}$
w	body Z-axis wind-relative velocity, $m.s^{-1}$

x	system state; or spatial x position, m
$x_{a_x}, x_{a_y}, x_{a_n}$	sensor X-axis positions, m
y	spatial y position, m
$y_{a_x}, y_{a_y}, y_{a_n}$	sensor Y-axis positions, m
z	system response
$z_{a_x}, z_{a_y}, z_{a_n}$	sensor Z-axis positions, m
α	angle of attack, rad
β	angle of sideslip, rad
δ_a	aileron deflection, rad
δ_e	elevator deflection, rad
δ_r	rudder deflection, rad
θ	parameter vector; or pitch attitude, rad
Φ	discrete state-space state matrix
ϕ	roll attitude, rad
ρ	air density, $kg.m^{-3}$
ψ	heading angle, rad

Subscripts:

b	bias
p, r, q	derivative with respect to the subscript
α, β, δ	derivative with respect to the subscript
z	measured quantity

Superscripts:

\sim	error signal
$\hat{}$	value produced by estimation

Chapter 1

Introduction

At the start of this project the ASK-13 glider of the Cape Gliding was undergoing a major overhaul. This provided an opportunity to install sensors into the fuselage of the glider, with the intent of quantifying the handling characteristics of ZS-GHB. Relying on previous work, a mathematical model for the dynamics of aircraft is available. What is required in this project is to develop the infrastructure that will make it possible to establish scientifically the parameter values in the model that will describe the handling characteristics of ZS-GHB.

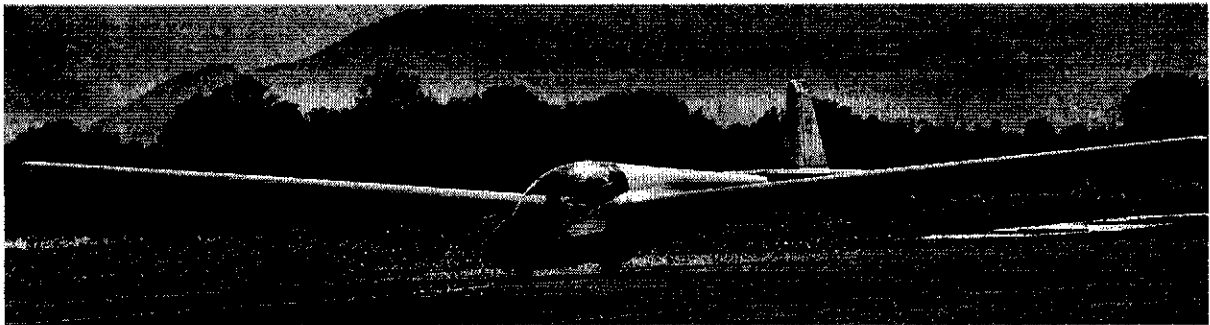


Figure 1.1: *The glider ZS-GHB*

The ASK-13 glider callsign ZS-GHB (Figure 1.1) was used as the flight test aircraft. It is owned by the Cape Gliding Club based at Worcester airfield. Development of the ASK-13 started in 1965. The designer, Rudolf Kaiser, improved his design of the two-seat Ka-2 and Ka-7 producing the ASK-13. The glider was built by Schleicher with mixed materials including metal, wood and fiberglass. The ASK-13 first flew in prototype form in July 1966 and by January 1978 a total of approximately 700 ASK-13's had been built. Over the years it has become a popular training glider.

The wings have a forward sweep of 6° at the quarter chord line and a dihedral of 5° and are mid-mounted allowing the use of a large blown canopy which allows all round view for both pilots. The construction of the wing uses a D-type leading edge torsion box of plywood with a fabric outer covering. The Schempp-Hirth air brakes extend from both the top and bottom of the wings. The ailerons are made of wood and are fabric-covered. The fuselage is a welded

steel tube structure with spruce stringers and covered with fabric, except for the nose which is covered with fiberglass. The turtle decking aft of the cockpit is a plywood shell. The wooden tail is covered in plywood except for the rear portion of the elevators and rudder, which is fabric-covered. The starboard elevator has a Flettner trim tab built in. Landing gear consists of a non-retractable sprung mono wheel with a retrofitted disc brake, a smaller nose wheel and a recently added tail wheel.

Span	16.0 m
Area	17.50 sq. m
Aspect ratio	14.6
Airfoil	Go 535/ 549
Empty weight	325 kg.
Payload	175 kg.
Gross weight	500 kg.
Wing loading	27.43 kg. / sq. m.
L/D max.	27 at 90 kph
Min. sink	0.81 m/s at 72 kph

Table 1.1: *Specifications of the ASK-13*

To extract the control and stability derivatives of ZS-GHB the response to specific manoeuvres needs to be recorded. To record the response of the glider sensors need to be installed into the glider. The sensors will have to measure the changes of the glider's state in the inertial reference frame and record the control inputs that caused the change in state. The MMLE3 estimation algorithm is then used to extract the control and stability derivatives from the recorded flight data. The rest of this document describes the maximum likelihood estimator used to estimate the parameters, the sensors and supporting circuitry used to collect the flight data, the information gathered to facilitate the estimation, and ends with a discussion of the estimation results.

Chapter 2

System Identification

2.1 Perspective on System identification

Identification is the process that is used to extract the characteristics of a dynamic system from recorded responses to known inputs. In the case of this project the control and stability derivatives of ZS-GHB will be estimated. System identification can be broken up into the search for two main types of models, parametric models and non-parametric models. Non-parametric models are models that primarily reproduce the responses of the system and are typically the result of frequency domain identification techniques. The non-parametric model will not be able to give information about the physical manifestation of the dynamic system. Parametric models produce a model based on the physics involved in generating the particular dynamics of the system. The parametric model is of more use, it has the ability to be adjusted with known changes in physical properties (pilot masses, air speed, altitude) making the model more universal and thus usable in a simulation environment.

This chapter describes the basic theory of the maximum likelihood estimator (MLE). The maximum likelihood estimator was used because of its availability and its proven record in estimating aircraft models [11], [9]. The formulation of the MLE used is the MMLE3. It can handle multiple-input-multiple-output (MIMO) linear time-independent models with measurement and process noise. The MMLE3 originates from the Ames Research Center where it was developed for estimating aircraft parameters and has proven to be very successful [21], [18], [20], [5], [6]. The estimation software routines were from [14]. The theory that follows is very brief. For more detail consult the references [4], [11], [9] and [14].

2.2 Maximum Likelihood Estimation

Maximum Likelihood Estimation is often used to extract information out of signals where it is known that the observations are not reliable. The output it provides is the most likely parameters considering the information given and the potential corruption of the data by noise. The most di-

rect method of deriving the MLE is using the Bayesian approach from [14]. In this approach the parameter vector θ is regarded as a random variable. It is assumed to contain unknown random constants with known probability densities (a priori knowledge). The derivation is started with a more general MAP (Maximum-a-Posteriori) estimator. The MAP estimator which, unlike the MLE, uses prior information and experiment data.

$$\hat{\theta}_{MAP} = \underset{\hat{\theta}}{\operatorname{arg\,max}} P(\hat{\theta}|z) \quad (2.1)$$

This means we will maximize the probability of $\hat{\theta}$ given the experiment data z . To manipulate $P(\hat{\theta}|z)$ into a measurable form, Bayes rule is used.

$$P(\hat{\theta}|z) = \frac{P(z|\hat{\theta}) \times P(\hat{\theta})}{P(z)} \quad (2.2)$$

$P(z|\hat{\theta})$ is the conditional probability of obtaining all the measured data given the specific parameters ($\hat{\theta}$). If each z is independent of the previous z then $P(z|\hat{\theta})$ is the product of the individual probabilities of each measurement, conditioned on all previous data and estimated parameters. Manipulating the MAP estimator into a form that can be minimized we use a negative logarithm and the estimator becomes

$$\hat{\theta}_{MAP} = \underset{\hat{\theta}}{\operatorname{arg\,min}} [-\log(P(\hat{\theta}|z))] \quad (2.3)$$

$$\log(P(\hat{\theta}|z)) = \log P(z|\hat{\theta}) + \log P(\hat{\theta}) - \log P(z) \quad (2.4)$$

$P(z)$ is unaffected by $\hat{\theta}$ hence it is ignored. To obtain the ML estimator, the prior knowledge of the parameter values is disregarded and $P(\hat{\theta})$ is assumed identical for all values of $\hat{\theta}$, and can therefore be ignored . This gives us the Log-likelihood function (*LLF*).

$$LLF \equiv -\log P(z|\hat{\theta}) \quad (2.5)$$

and the ML estimator:

$$\hat{\theta}_{ML} = \underset{\hat{\theta}}{\operatorname{arg\,min}} (LLF(\hat{\theta})) \quad (2.6)$$

If the parameters are assumed to have a Gaussian probability density there are further simplifications, which are described in the next section.

2.3 ML Estimation for Gaussian State-Space Models

This section describes the interconnection between the Kalman filter, the Gaussian state-space model and the estimator used to estimate the parameters. The model is called Gaussian because Gaussian noise inputs are included in the structure.

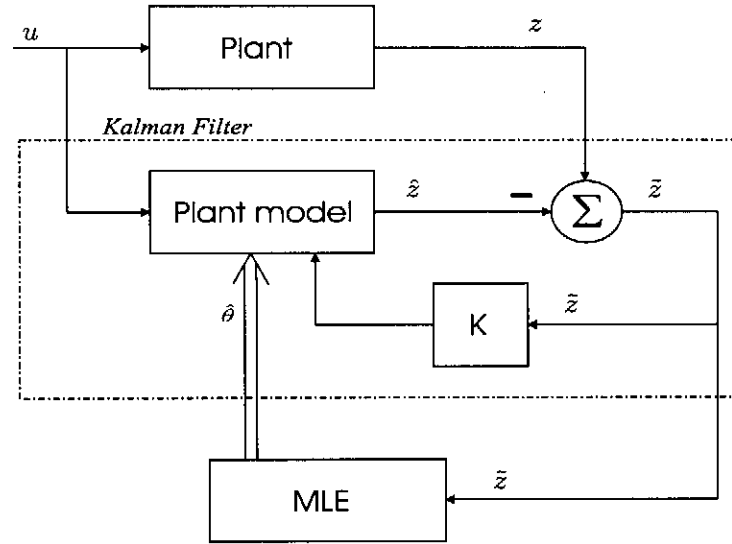


Figure 2.1: System identification diagram

The Kalman filter along with the output of the plant is used to produce the innovations \tilde{z}_i , which drives the estimator.

$$\tilde{z}_i = z_i - \hat{z}_i \quad (2.7)$$

The Kalman filter that produces \hat{z}_i uses the most recent estimate of the parameters. It is defined by

$$\hat{x}_{i+1} = \hat{\Phi} [(\mathbf{I} - \hat{\mathbf{K}}\hat{\mathbf{C}})\hat{x}_i + \hat{\mathbf{K}}(z_i - \hat{\mathbf{D}}u_i)] + \hat{\Gamma}u_i \quad (2.8)$$

The assumed model output is then

$$\hat{z}_i = \hat{\mathbf{C}}\hat{x}_i + \hat{\mathbf{D}}u_i \quad (2.9)$$

where the $\hat{\cdot}$ denotes matrices created using estimated parameters. Assuming that the model is linear and time-invariant with independent white noise, the probability of the measurements can be described as in [9] by

$$P(z_i | z_{1:i-1}, \hat{\theta}) = \frac{1}{[(2\pi)^m \det(\mathbf{R}\mathbf{R}^T)]^{1/2}} \exp\left(-\frac{1}{2} \tilde{z}_i^T (\mathbf{R}\mathbf{R}^T)^{-1} \tilde{z}_i\right) \quad (2.10)$$

Where $\mathbf{R}\mathbf{R}^T$ is the innovation covariance $E[\tilde{z}_i \tilde{z}_i^T]$. The total conditional probability would be given by

$$P(z | \hat{\theta}) = \prod_{i=1}^N P(z_i | z_{1:i-1}, \hat{\theta}) \quad (2.11)$$

$$P(z | \hat{\theta}) = \prod_{i=1}^N \frac{1}{[(2\pi)^m \det(\mathbf{R}\mathbf{R}^T)]^{1/2}} \exp\left(-\frac{1}{2} \tilde{z}_i^T (\mathbf{R}\mathbf{R}^T)^{-1} \tilde{z}_i\right) \quad (2.12)$$

Now substituting the conditional probability into equation 2.5 will produce:

$$LLF(\hat{\theta}) = \frac{1}{2} \sum_{i=1}^N \tilde{z}_i^T (\mathbf{R}\mathbf{R}^T)^{-1} \tilde{z}_i + \frac{N}{2} \log \det(\mathbf{R}\mathbf{R}^T) + \frac{Nm}{2} \log(2\pi) \quad (2.13)$$

The ML estimation problem is now one of minimizing a scalar function by adjusting $\hat{\theta}$. In equation 2.13 the last term is constant and can be ignored. What is left then is a function of \tilde{z} which is a function of $\hat{\theta}$, and $\mathbf{R}\mathbf{R}^T$. A result used by Maine and Illif in [14] is that when LLF is at a minimum, $\mathbf{R}\mathbf{R}^T$ has to be equal to $\widehat{\mathbf{R}\mathbf{R}^T}$. If $\mathbf{R}\mathbf{R}^T$ is unknown and needs to be estimated then we can use the sample innovations covariances ($\widehat{\mathbf{R}\mathbf{R}^T}$), which is calculated by

$$\widehat{\mathbf{R}\mathbf{R}^T} = \frac{1}{N} \sum_{i=1}^N \tilde{z}_i \tilde{z}_i^T \quad (2.14)$$

The LLF is in the form of a weighted least squares cost function.

$$J_{ML} = \frac{1}{2} \sum_{i=1}^N \tilde{z}_i^T W \tilde{z}_i \quad (2.15)$$

The cost function represents the sum of the energy in the weighted innovations. The choice of the weighting matrix W can make a large difference in the results of the minimization. Using the inverse of the innovations covariance (as required by the ML estimator) scales each of the inputs emphasizing the channels with low noise and the more reliable data and de-emphasizing noisy data.

2.4 Minimizing the LLF

The LLF is of the “sum of squares” form for which a number of efficient minimization algorithms exist.

$$J(\hat{\theta}) = \frac{1}{2} \sum_{i=1}^N \tilde{z}_i^T (\mathbf{R}\mathbf{R}^T)^{-1} \tilde{z}_i \quad (2.16)$$

If $\mathbf{R}\mathbf{R}^T$ is known, parameters only influence the cost function by the innovations (\tilde{z}). If $\mathbf{R}\mathbf{R}^T$ is unknown, it has to be updated after each iteration using equation 2.14. The MMLE3 algorithm makes use of the Gauss-Newton method [9] for minimizing the LLF . This method is a modified Newton-Raphson which is computationally efficient. It uses an approximation for the second gradient to increase the robustness of the algorithm. A side effect of this approximation is that the second gradient of the sensitivities does not have to be computed, thus saving time. The next two sections describe the approximation used for the second gradient in the Gauss-Newton method and the Levenberg-Marquardt stabilization which is used to aid convergence. The Levenberg-Marquardt stabilization aids the convergence of the Gauss-Newton method with starting conditions far from the minimum and/or identifiability problems.

2.4.1 Gauss-Newton Approximation of the Hessian

In the Newton-Raphson method [14] the parameters are updated by:

$$\hat{\theta}_{i+1} = \hat{\theta}_i - [\nabla_{\theta}^2 J(\hat{\theta}_i)]^{-1} [\nabla_{\theta} J(\hat{\theta}_i)] \quad (2.17)$$

Using the chain rule to calculate the gradient and second gradient of the cost function:

$$\nabla_{\theta} J = \sum_{i=1}^N (\nabla_{\theta} \tilde{z}_i)^T (\mathbf{R}\mathbf{R}^T)^{-1} \tilde{z}_i \quad (2.18)$$

$$\nabla_{\theta}^2 J = \sum_{i=1}^N (\nabla_{\theta} \tilde{z}_i)^T (\mathbf{R}\mathbf{R}^T)^{-1} \nabla_{\theta} \tilde{z}_i + \sum_{i=1}^N \nabla_{\theta}^2 \tilde{z}_i (\mathbf{R}\mathbf{R}^T)^{-1} \tilde{z}_i \quad (2.19)$$

The second gradient of the cost function is called the Hessian ($\nabla_{\theta}^2 J$). To aid the convergence an approximation of the Hessian is used in place of equation 2.19. The second term in the Hessian is the second gradient of the innovations. It is expensive to calculate, approaches zero at the cost function minimum (it is zero all the time for quadratic problems) and cannot be guaranteed positive-semi-definite. When the second term of the Hessian is not positive-semi-definite, problems with the convergence of the algorithm can occur. To circumvent the possibility of convergence problems the Gauss-Newton approximation of the Hessian is used in equation 2.21 to calculate the next iteration of parameter values. The Gauss-Newton Hessian approximation is:

$$\mathbf{hes} \equiv \sum_{i=1}^N (\nabla_{\theta} \tilde{z}_i)^T (\mathbf{R}\mathbf{R}^T)^{-1} \nabla_{\theta} \tilde{z}_i \quad (2.20)$$

The parameter update is then:

$$\hat{\theta}_{i+1} = \hat{\theta}_i - \mathbf{hes}^{-1} \nabla_{\theta} J \quad (2.21)$$

Using the **hes** approximation, the path to the minimum is altered to a more robust course down the gradient. The location of the minimum is unaffected by this approximation as the gradient calculation is unaffected

2.4.2 Levenberg-Marquardt Stabilization

The local estimate of the second gradient is not guaranteed to rotate and scale gradient for the optimal improvement of the parameters. When such problems with convergence occur it is frequently better to make smaller steps in the negative gradient direction until the quadratic area near the minimum is reached. These cases are typically non-quadratic and can be caused by initial parameter values being far from correct and/or local minima in the cost function. To select the smaller steps, the diagonal elements of the **hes** are scaled with a number slightly greater than unity e.g. 1.03. The Levenberg-Marquardt refinement of the Gauss-Newton method is

used to find this number. The diagonal elements of \mathbf{hes} are multiplied by a scalar $(1 + \mathit{marq})$. Having calculated the gradient and Hessian, up to five values of the marq are tried and the LLF is evaluated for each of the trial updates. If the LLF improves with the current value of marq the value of marq is divided by a convenient number $\sqrt{10}$ [14], thus increasing the parameter step size and rotating the step more toward the Newton direction. The marq that provides the best improvement in LLF is used in the calculation of the next jump. By the time the quadratic area near the minimum is reached, marq is so small that its effect has disappeared. The expensive gradient and Hessian matrices are calculated and then used for a number of trials making the extra computational load of the Levenberg-Marquardt stabilization small. The Levenberg-Marquardt Stabilization provides good starting properties without degrading the final convergence of the Newton algorithm [14].

Chapter 3

Parameter Accuracy Indicators and Analysis Tools

Parameter estimates from real systems are by their nature imperfect. To make effective use of parameter estimates the ability to gauge their accuracy is necessary. The methods that can be used are statistical, intuitive or may come from another problem specific source. If the accuracy of an estimated parameter cannot be quantified the parameter is worthless. This makes the indication of the accuracy just as important as the parameter estimate itself. The single most important factor in estimating accurate parameters from practical problems, is the insight that comes from the analyst's understanding of the system and the instrumentation used to measure it.

There are a few uses for the accuracy indicators. The first would be in the planning phase, where the instruments and the experiments can be simulated. This use is limited because of the lack of real-data. What it can do though is detect experiments with no hope of success at an early stage. The second use of the accuracy indicators is in the estimation process itself. The accuracy indicators can be used to find identification problems, faults in the programs or other sources of errors. The accuracy indicators can also be used for comparisons between parameter estimates from different experiments and data-sets. In situations where there are conflicting values, the accuracy measures can show which is likely to be more accurate. The third use is the presentation of the final estimates. If the estimates are going to be used in control applications the accuracy parameters are useful in the considerations of the sensitivity and robustness of the controller.

In the context of parameter estimation, what does the term accuracy mean? A system is never described exactly by the simplified model used for analysis. Regardless of the model, unexplained sources of error will remain. It is difficult to define accuracy if no correct model exists. To make this problem manageable, it is split into two parts: Modeling and Estimation. Modeling is the problem of finding suitable models of the system for the task required. Estimation is finding suitable parameter values for the model. To make the estimation work we assume

that the model describes the system exactly. Now precise and qualitative measures of estimation accuracy for the specific model can be generated. The parameter estimates that result from a certain model need to be judged as adequate and not as exact. This involves considerable engineering judgment combined with the available qualitative measures to determine the usefulness of parameters.

The accuracy indicators used for this project are presented in the rest of this section. It is by no means an exhaustive discussion of all the possible tools or techniques to determine parameter accuracy. More details and insights are available in the full reports: [10], [14] and [15]

3.1 Scatter Plot

The scatter plot, which is described in this section, is the most effective and convincing way of examining the accuracy of an estimated parameter. On the plot the estimated value of a particular parameter from multiple experiments is plotted with the value of the parameter on the Y-axis and the experiments along the X-axis. Using multiple experiments the scatter plot

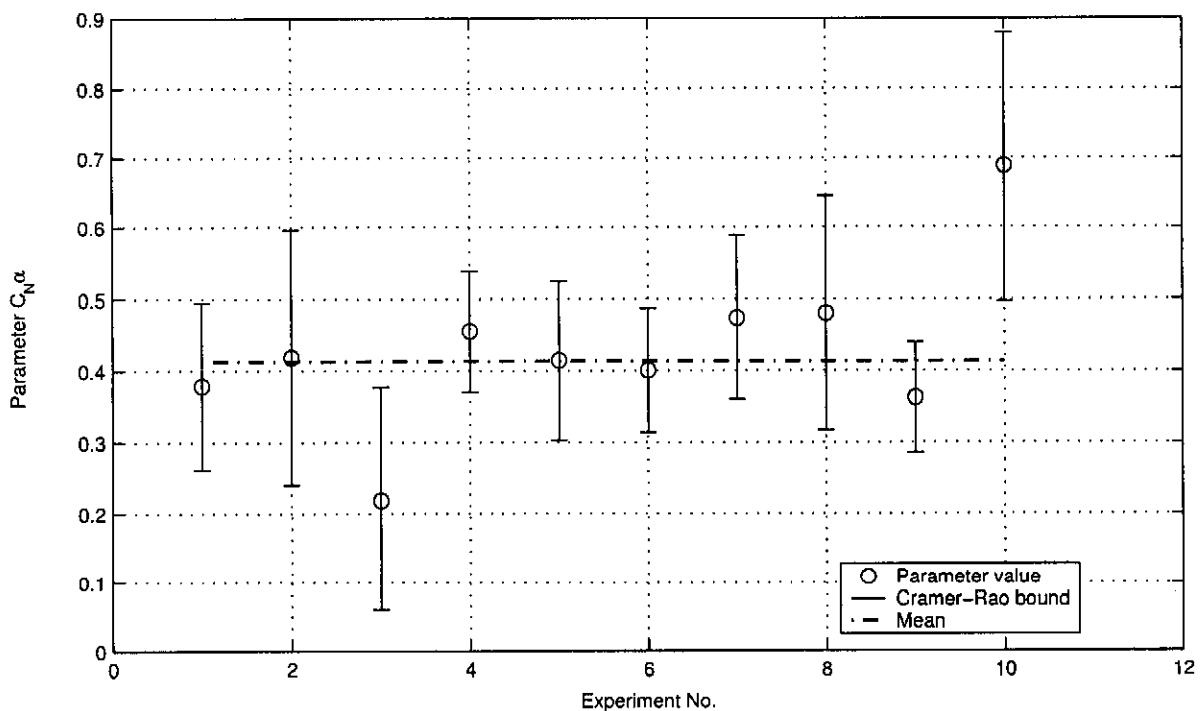


Figure 3.1: *Example of a Scatter Plot*

will reveal trends and make the relative differences between the same parameters from different experiments apparent. Typically a Gaussian cluster about the expected value of the parameter is visible. By plotting the theoretical error bounds of each experiment on the scatter plot, an even better understanding will be obtained. The error bounds will often resolve problems with conflicting or faulty data and reveal indiscernible trends and if few experiments are available is

the only means to measure the accuracy of a parameter. The error bound that is most commonly used is the Cramer-Rao lower bound.

3.1.1 Cramer-Rao Bound Theory

The Cramer-Rao bound provides a tool to evaluate the believability of an estimated parameter. The complete derivation of the Cramer-Rao bound is in reference [10]. To be able to use the Cramer-Rao bound the basic theory and computation is presented in the following text. The covariance matrix of parameters estimated from an unbiased estimator will always equal or exceed the inverse of the Fisher information matrix when their elements are compared. The Fisher information matrix is equivalent to \mathbf{hes} for ML estimations, provided equation 2.14 is satisfied. This is dictated by the Cramer-Rao inequality [14]. The ML estimator is asymptotically unbiased, consistent and efficient [17], [10]. These characteristics allow the assumption that for sufficient data length the inverse of the Fisher information matrix will approach the lower limit of the Cramer-Rao bound, which is the covariance matrix of the parameters [10]. The covariance matrix is approximated by:

$$E [(\hat{\theta} - \theta_{true})(\hat{\theta} - \theta_{true})^T] \approx \mathbf{hes}^{-1} \quad (3.1)$$

The theoretical standard deviation of a parameter is approximated by its Cramer-Rao bound.

$$\sigma_i \approx \text{Cramer-Rao bound} \approx [\mathbf{hes}^{-1}]_{ii}^{\frac{1}{2}} \quad (3.2)$$

An estimated parameter with a high Cramer-Rao bound is considered to be less believable than a parameter estimate with a low Cramer-Rao bound. It was found through analysis of experimental data that the Cramer-Rao bounds are 10 times less than the actual scatter [10]. The reason for this error, and the correction applied, are explain in the following section.

3.1.2 The Filtered Cramer-Rao Bound

The root of the error in the Cramer-Rao bounds comes from the spectral content of the noise and how it affects the estimation process. The result from investigations of Maine and Illif in [10] and theoretical explanation from Milne in [15] provide a reason for the error in the Cramer-Rao bound. To summarize the findings of these publications: The noise that affects the estimation process is the noise that occurs in the same spectral range as the response of the system being identified. Generally the algorithms used for estimators are developed to have good high frequency noise rejection properties but have difficulty in discerning between the noise and actual response on the same frequency. The error in the Cramer-Rao bound occurs because the theory assumes that all the innovation energy has a uniform spectral density, where in practice it is often in a limited band and affects the estimation more severely.

The Cramer-Rao bound cannot be accurately calculated if the spectral nature of the noise and and system bandwidth is not taken into account. The suggested method from Maine and

Illif, [10] for correcting the Cramer-Rao bounds is to scale the innovations covariance used in calculating the Cramer-Rao calculation. At the minimum of the LLF , if the estimation is working correctly, the innovations are a good representation of the noise. The scaling is done to get the power spectral density, which in theory is assumed constant up to the Nyquist frequency, to be equal to the peak innovations spectral density.

In aircraft estimation it has been found that the innovations tend to have low-pass characteristics. To obtain an estimate of the low-frequency spectral density of the innovations, the innovations are passed through a low-pass filter with cut-off at f_{-3dB} and then dividing the output power by the filter noise bandwidth. In the MMLE3 implementation the factor used to correct the Cramer-Rao bound is calculated by first calculating the cost function with the filtered innovations to get J_{fil} . The correction factor is then calculated by

$$\left(\frac{J_{fil}}{f_{-3dB}} \times \frac{f_{Nyquist}}{Nm} \right)^{0.5} \quad (3.3)$$

Where Nm , the length of the observation vector, is a close approximation of the cost function value for white residuals, [8]. Scaling the Cramer-Rao bound with this correction factor gives us the filtered Cramer-Rao bounds, which are a close approximation of the actual standard deviation of the parameters.

3.2 The Confidence Ellipsoid

The Confidence Ellipsoid is a construct which allows us to relate various parameter accuracy statistics to each other, to create a better understanding of the results produced by the ML estimator. In most estimation problems the number of parameters quickly escalates to more than three, making visualization impossible. The statistics used for accuracy determination try to reduce the problem to a one dimensional form, with various indicators for each parameter. Understanding the geometric relationships between these statistics leads to greater insight into the estimates and is helpful in finding problems with the estimation and gives clues to the possible solutions. The Confidence ellipsoid is defined by:

$$\delta\theta^T \mathbf{hes} \delta\theta = 1 \quad (3.4)$$

It is chosen to be defined in this way so that if one of the parameters is perturbed from the origin of the ellipsoid to a point on the ellipsoid the LLF changes by 0.5 units. This is done in order to be consistent with the Cramer-Rao bounds. The ellipsoid represents one standard deviation. Considering that if the LLF is very small, i.e. 0.01, we can assume that the actual parameter vector is likely to be very close to the center of the confidence ellipsoid. This can be used as a termination criterion for the minimization algorithm. Another point to note is that the Confidence Ellipsoid is not centered on the true minimum of the cost-function but on the estimated minimum

The confidence ellipsoid can be highly skewed and eccentric. To describe the shape of the confidence ellipsoid, the length and direction of its principle axes are used. These are given by V_i and $\lambda_i^{-1/2}$, where V_i and λ_i are the eigenvectors and corresponding eigenvalues of \mathbf{hes} . The symmetric nature of the \mathbf{hes} ensures that the principle axes are orthogonal. In [14] and [15] the geometric relationships between the accuracy parameters with a two dimensional confidence ellipsoid are clearly highlighted in the following way: the geometric features of a confidence

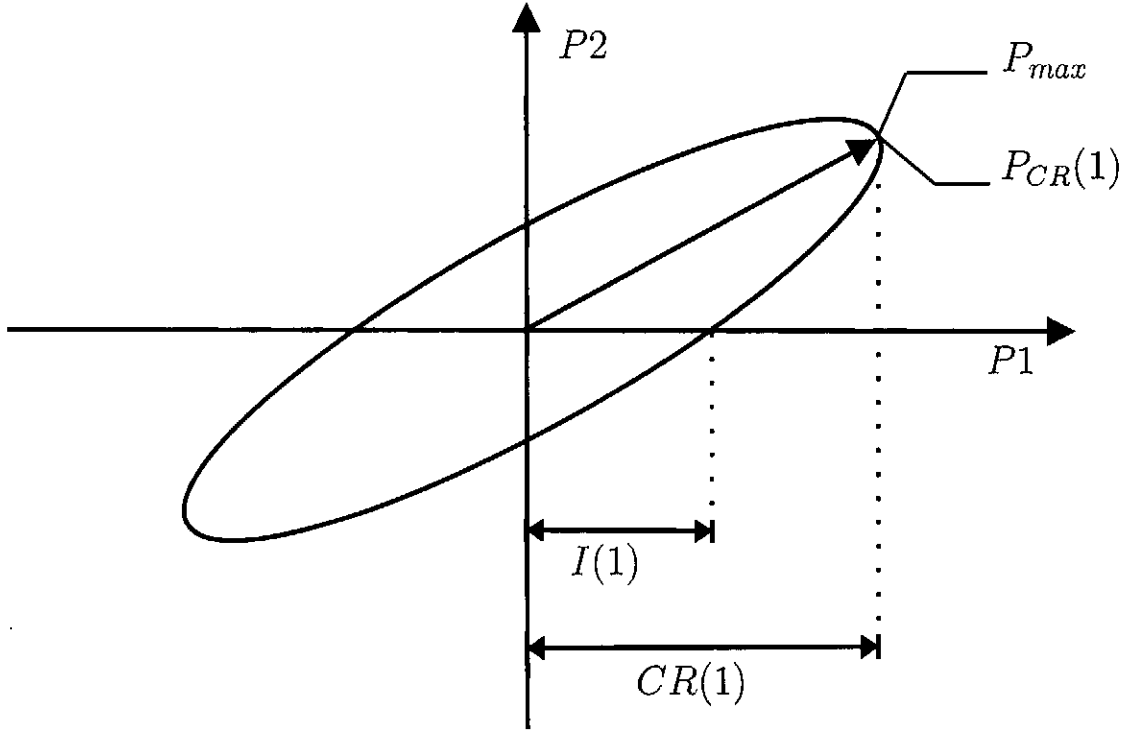


Figure 3.2: *Geometric Relationships between Confidence Ellipsoid and Accuracy Indicators*

ellipsoid are illustrated for the two dimensional case in Figure 3.2. These features will hold equally to higher dimensional cases.

- $CR(i)$ Cramer-Rao bound of the i th parameter. It is the standard deviation of $\hat{\theta}_i$, when estimated with other parameters. It is the projection onto θ_i of the furthest point of the confidence ellipsoid

$$CR(i) = [\mathbf{hes}^{-1}]_{ii}^{1/2} \quad (3.5)$$

- $P_{cr}(i)$ is the point on the confidence ellipsoid that gives the Cramer-Rao bound

$$P_{cr}(i) = \theta_{CR(i)} = \frac{\mathbf{hes}^{-1}(:, i)}{CR(i)} \quad (3.6)$$

where $\mathbf{hes}^{-1}(:, i)$ is the i th column of \mathbf{hes}^{-1} .

- $I(i)$ is called the insensitivity. It gives the change required by θ_i to move from the minimum to the confidence ellipsoid. It is equal to the Cramer-Rao bound if one parameter is estimated. In other words, changing the i th component of \mathbf{P} from its optimum by I_i increases the LLF by 0.5.

$$I_i = [\text{hes}_{ii}]^{-1/2} \quad (3.7)$$

- $GDOP(i)$ Geometric Dilution Of Precision, the factor by which the standard deviation of θ_i increases when estimated simultaneously with others parameters.

$$GDOP(i) = \frac{CR(i)}{I(i)} \geq 1 \quad (3.8)$$

- P_{max} Major principal axis of the confidence ellipsoid

$$P_{max} = \lambda_{min}^{-1/2} \hat{V}_{min} \quad (3.9)$$

where \hat{V}_{min} is the unit eigenvector of hes with minimum eigenvalue, λ_{min} .

3.3 Using the Accuracy Indicators

Judging if parameters are accurate and correct can be difficult. The whole purpose of the estimation is to find the unknown parameters or to validate parameters calculated from theory. If there are no exact parameters to compare the estimates there can be no exact measure of accuracy of the estimate. As result the accuracy indicators tend to be more qualitative than quantitative.

3.3.1 Matching Responses

Checking the accuracy of estimated parameters starts with comparing the responses of the actual system and the simulated response from a system using estimated parameters. The responses should be a close match before proceeding to look at the parameters, scatter and confidence ellipsoid indicators. Figure 3.3 shows three different responses. The responses give an indication of what good and bad response matches look like.

To help find possible cause of a misfit, the $GDOP$'s (eqn 3.8) and insensitivity (eqn 3.7) of parameters can be consulted. Tell-tale signs of unwanted correlation between sensitivities will be high $GDOP$'s and very low insensitivities and high Cramer-Rao bounds. For the important parameters in that estimation you would want the $GDOP$'s to be as low as possible (close to one). As extra checks a low Cramer-Rao bound and filtered Cramer-Rao bound are desired and the closer the bounds are to the insensitivities the better. Before consulting the scatter plot the analyst needs to be satisfied that the fit is good enough and that the $GDOP$'s and Cramer-Rao bounds of the parameters are low enough. The next step will be the scatter plot.

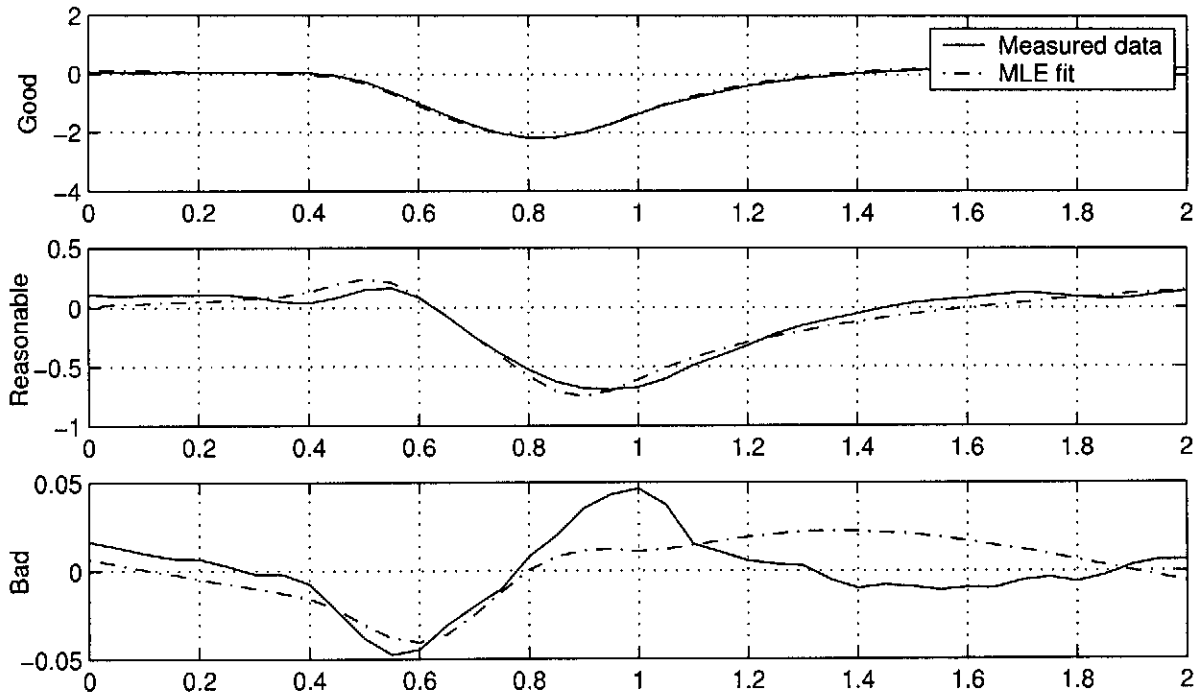


Figure 3.3: *Good, Reasonable and Bad Response matches*

3.3.2 Scatter Plots

On the scatter plot the same parameter from multiple experiments is plotted. The error bars are plotted over the parameters, using three times the filtered Cramer-Rao bound for half the bar length. Three times the filtered Cramer-Rao bound is the three sigma value for the standard deviation. This implies that there is a 99.73% probability that the other experiments will fall in this bound. We expect to see certain phenomenon in the scatter plot. The first is that the parameters are scattered around some expected value. The smaller the scatter between experiments the more confidence can be placed in the expected value as being the value best suited to representing the real system. Secondly, most of the parameters from the other experiments fall inside the 3 times Cramer-Rao bounds of each of the other experiments. There can be exceptions. In Figure 3.4 two exceptions are illustrated. The first experiment the parameter falls outside of the Cramer-Rao bounds of most of the other parameters and has a much larger Cramer-Rao bound than the other experiments in which all the other parameters are included. This is symptomatic of an experiment with poor input stimulus and/or insufficient length. The estimation is working properly and the Cramer-Rao bound shows that the value obtained in experiment 1 is unreliable. The second grouping is what is considered good. All the parameter values close to each other are inside the Cramer-Rao bounds from all the other experiments. The third group is where the parameter is outside the expected value and has a very small Cramer-Rao bound of its own. It may just be a bad experiment with unrecorded inputs or process noise and can be discarded. If there are many with low Cramer-Rao bounds and the scatter is high then identifiability problems

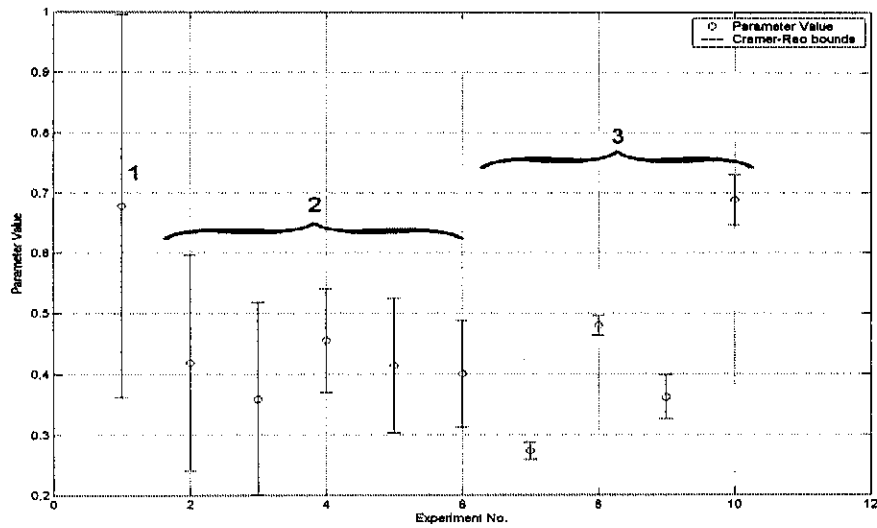


Figure 3.4: Possible Scatter Plot Patterns

exist and the experiments should be redesigned to be more suitable for estimating that parameter. The other possibility is that the filter frequency chosen by the user is too large. Examining the spectral content of the responses can help in finding the correct filter frequency. Typically it is $0.5Hz$ for aircraft. To gain greater insight into the severity of the estimated parameters scatter around the expected value, the parameters are normalized by the expected value. It is then possible to examine the scatter as a percentage deviation from the expected value. Figure 3.5 is the percent deviation plot of Figure 3.4.

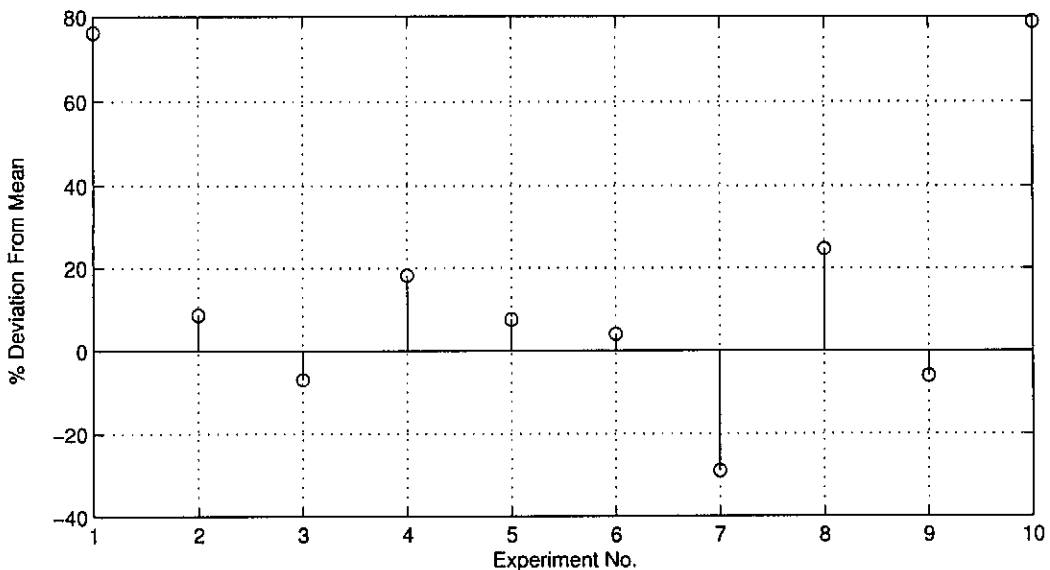


Figure 3.5: Percentage of Scatter from the Mean Estimated Value

3.3.3 Final Check

The final check is combining the estimated parameters from different experiments in a weighted mean. The weighted mean is calculated by weighing each parameter with its inverse covariance and taking the sum of all the parameters, divided by the sum of the weightings, [16].

$$\bar{\theta} = \frac{\sum \frac{1}{\sigma_i^2} \theta_i}{\sum \frac{1}{(\sigma_i)^2}} \quad (3.10)$$

With the expected covariance of the weighted mean to be:

$$\frac{1}{\sigma^2} = \sum_{i=1}^N \frac{1}{\sigma_i^2} \quad (3.11)$$

The result is a system that represents the least squares combination of parameters from all the experiments. This system is then used to produce responses, which are compared with the real system's responses. Success would mean that the system using the combined parameters will closely follow the real response.

Chapter 4

Equations of Motion

Before proceeding to describe the instrumentation of ZS-GHB, a certain amount of familiarity with the equations of motion governing the dynamics of the glider's flight is needed. This familiarity is needed to ensure that the correct instruments are installed to take the relevant measurements. This section describes the reference axes used, the interactions between the forces, moments and states of the aircraft and reveals the control and stability derivatives as the values which describe the handling qualities of the aircraft. The generalized equations are also modified for use on gliders, by removing engine and thrust terms in the equations of motion.

4.1 Axes Systems

Before the equations for the forces and moments can be derived the references on which they are based need to be defined.

4.1.1 Leading Edge Datum

On the glider, measurements need to be taken to relate the relative positions of certain parts and properties of interest. For example, the position of the center of gravity in relation to the accelerometers needs to be known. A standard documented origin for measurements is needed to ensure that measurements are repeatable and comparable in later work. The reference used for this work is the Leading Edge Datum. It is the leading edge of the wings at the wing root. The axes are in the same direction as the body axes, while the origin is at the Leading Edge Datum.

4.1.2 Body Axes

The axes system chosen for the measurement of the aerodynamic forces and moments is the standard NASA aircraft body-axis system. The origin is chosen at the center of gravity. The X-axis is positive forward out of the nose of the aircraft. The Y-Axis is positive out of the right

wing. The Z-axis is positive down out of the bottom of the aircraft to complete the right-hand axis system. The axis-system moves with the aircraft. The body axes forces are F_X , F_Y and F_Z . The normal force, $F_N = -F_Z$ and the axial force, $F_A = -F_X$, are often used instead because F_N and F_X are positive in normal flight. F_N and F_A relate more intuitively to lift and drag. The moments are the rolling l , pitching m and yawing n around the X,Y and Z axes respectively.

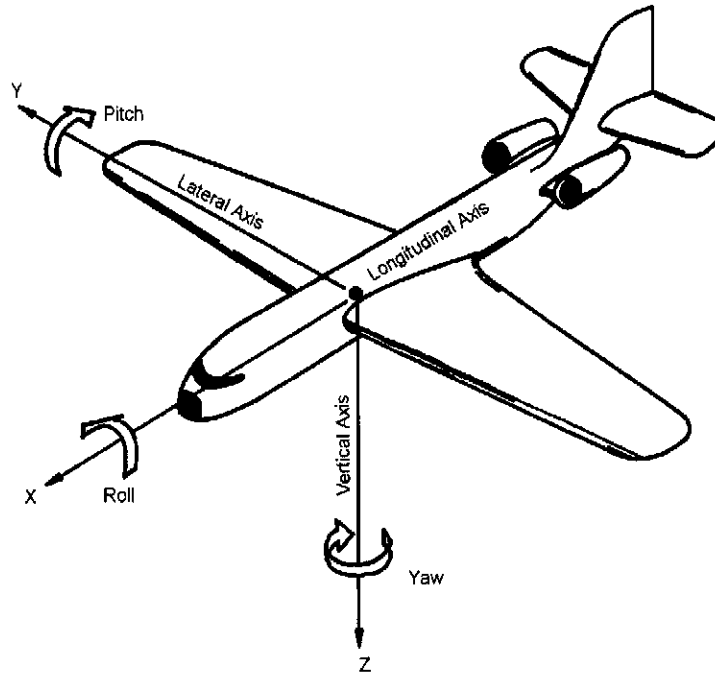


Figure 4.1: *Body Axes*

4.1.3 Wind Axes

Wind axes are used to define the forces that result from the airflow over the aircraft. The basis for this axes system is the relative velocity between the air and the aircraft, denoted by the wind velocity vector V , which is positive pointing in the direction from which the relative wind comes. The forces defined in this system are: Lift L , which is normal to the wind velocity vector and the body axis Y vector, and Drag D , which is in the opposite direction to V . The third direction is so seldom used it is left undefined but can be obtained from the cross product of the Drag and Lift vectors, and is analogous to the Y-axis of the body axes. This axis system is also fixed to the center of gravity of the aircraft. The wind axes rotate around the body axes as the wind velocity vector changes direction.

Two values that are important to stability and control analysis are the angle of attack (α) and the angle of side-slip (β), also called flow angles. These two angles relate the body axes to V .

The Cartesian components of V on the body axes are u, v and w , giving

$$V = \sqrt{u^2 + v^2 + w^2}$$

Angle of attack and side-slip are then defined by

$$\alpha = \tan^{-1} \frac{w}{u} \quad (4.1)$$

$$\beta = \sin^{-1} \frac{v}{V} \quad (4.2)$$

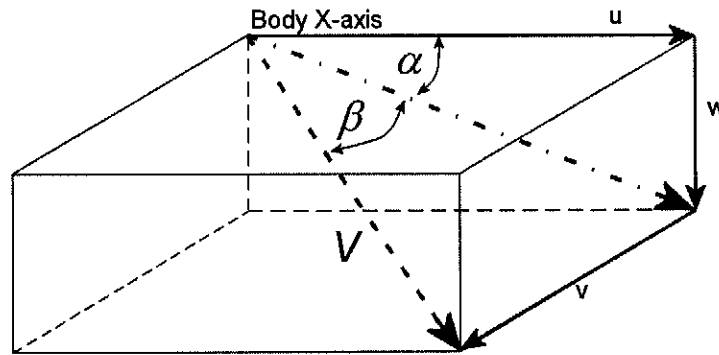


Figure 4.2: *Wind Axes*

The Euler angles describe the attitude of the aircraft with respect to the earth. The attitude is defined by three angles, heading (ψ), pitch (θ) and roll (ϕ). These three angles define the rotations of the aircraft body axes relative to the earth fixed axes, at any instant. The transformation of vector V_e from the earth axis to the body axes vector V_b is done with three rotations. The rotations are done in a specific order (ψ , θ then ϕ), which leads to a generalized formula for rotations called the Direction Cosine Matrix (DCM).

$$V_b = \begin{bmatrix} \cos\theta\cos\psi & \cos\theta\sin\psi & -\sin\theta \\ \sin\phi\sin\theta\cos\psi - \cos\phi\sin\psi & \sin\phi\sin\theta\sin\psi + \cos\phi\cos\psi & \sin\phi\cos\theta \\ \cos\phi\sin\theta\cos\psi + \sin\phi\sin\psi & \cos\phi\sin\theta\sin\psi - \sin\phi\cos\psi & \cos\phi\cos\theta \end{bmatrix} V_e$$

The inverse rotation is also possible by using the transpose of the DCM. Although the DCM is seldom used directly as defined above, it does illustrate the origin of many of the terms that appear in the equations of motion.

4.1.4 Control Surfaces

As with the Leading Edge Datum, a convention needs to be set for the control surfaces. The convention used in this project are taken from [11] and modified slightly. The convention is as follows: The rudder, trim and elevator are angular deflections measured normal to the hinge line. Positive deflection is trailing edge down or to the port side, depending on the orientation of

the control surface. The ailerons are antisymmetric control surfaces, designed so that if the port aileron is down the starboard aileron will have a lesser deflection up and vice versa. Therefore the aileron deflection is defined as the starboard aileron deflection minus port aileron deflection. The air brake is measured in millimeters extended out of the wing and can only be positive.

4.2 Equations of Motion

This section briefly describes the set of six-degree-of-freedom equations of motion for aircraft without engines.

A complete derivation of the equations is out of the scope of this project. The equations listed here are those formulated by Maine and Illif [11] and derived in [2]. The nine-nonlinear equations describe the motion of the glider.

$$\dot{V} = \bar{q}SC_{Dwind} \quad (4.3)$$

$$+ g(\cos\phi \cos\theta \sin\alpha \cos\beta + \sin\phi \cos\theta \sin\beta - \sin\theta \cos\alpha \cos\beta)$$

$$\dot{\alpha} = \frac{\bar{q}S}{mV \cos\beta} C_L + q - \tan\beta (p \cos\alpha + r \sin\alpha)$$

$$+ \frac{g}{V \cos\beta} (\cos\phi \cos\theta \cos\alpha + \sin\theta \sin\alpha) \quad (4.4)$$

$$\dot{\beta} = \frac{\bar{q}S}{mV} C_{Ywind} + p \sin\alpha - r \cos\alpha + \frac{g}{V} \cos\beta \sin\phi \cos\theta$$

$$+ \frac{\sin\beta}{V} (g \cos\alpha \sin\theta - g \sin\alpha \cos\phi \cos\theta) \quad (4.5)$$

$$\dot{p}I_{xx} - \dot{q}I_{xy} - \dot{r}I_{xz} = \bar{q}SbC_l + qr(I_{yy} - I_{zz}) + (q^2 - r^2)I_{yz} + pqI_{xz} - rpI_{xy} \quad (4.6)$$

$$-\dot{p}I_{xy} + \dot{q}I_{yy} - \dot{r}I_{yz} = \bar{q}ScC_m + rp(I_{zz} - I_{xx}) + (r^2 - p^2)I_{xz} + qrI_{xy} - pqI_{yz} \quad (4.7)$$

$$-\dot{p}I_{xz} - \dot{q}I_{yz} + \dot{r}I_{zz} = \bar{q}SbC_n + pq(I_{xx} - I_{yy}) + (p^2 - q^2)I_{xy} + rpI_{yz} - qrI_{xz} \quad (4.8)$$

$$\dot{\phi} = p + q \tan\theta \sin\phi + r \tan\theta \cos\phi \quad (4.9)$$

$$\dot{\theta} = q \cos\phi - r \sin\phi \quad (4.10)$$

$$\dot{\psi} = r \cos\phi \sec\theta + q \sin\phi \sec\theta \quad (4.11)$$

As is standard with aerodynamics the forces and moments affecting the equations of motion are represented with non-dimensionalized coefficients. The forces and moments that act on aircraft are a combination of functions of the aircraft states and the control deflections. The functions are seldom linear, but for the small perturbations used in these flight tests linearizations of the functions can be used. To obtain the most simple functions, the longitudinal and lateral dynamics are assumed to be uncoupled. The lateral coefficients are expansions of sideslip, roll rate, yaw rate, rudder and aileron controls.

The linearized expansion for the lateral coefficients are:

$$C_Y = C_{Y_\beta}\beta + C_{Y_p}\frac{pb}{2V} + C_{Y_r}\frac{rb}{2V} + C_{Y_\delta}\delta + C_{Y_b} \quad (4.12)$$

$$C_l = C_{l_\beta}\beta + C_{l_p}\frac{pb}{2V} + C_{l_r}\frac{rb}{2V} + C_{l_\delta}\delta + C_{l_b} \quad (4.13)$$

$$C_n = C_{n_\beta}\beta + C_{n_p}\frac{pb}{2V} + C_{n_r}\frac{rb}{2V} + C_{n_\delta}\delta + C_{n_b} \quad (4.14)$$

The longitudinal coefficients are formed from functions of angle of attack, pitch rate and elevator deflection and give the expansions:

$$C_N = C_{N_\alpha}\alpha + C_{N_q}\frac{qc}{2V} + C_{N_\delta}\delta + C_{N_b} \quad (4.15)$$

$$C_A = C_{A_\alpha}\alpha + C_{A_q}\frac{qc}{2V} + C_{A_\delta}\delta + C_{A_b} \quad (4.16)$$

$$C_m = C_{m_\alpha}\alpha + C_{m_q}\frac{qc}{2V} + C_{m_\delta}\delta + C_{m_b} \quad (4.17)$$

The expansions are made with a linear combination of the states, multiplied by their respective control or stability derivative. In all the expansions a bias coefficient is added with the subscript b . This bias term is a linear extrapolation of the coefficient from the average α and δ of the manoeuvre to zero α and δ . The term adds a degree of flexibility to account for modeling errors due to the non-linearity of aerodynamic effects.

Observation equations are needed to link the equations of motion to the quantities measured by the sensors. Observation equations need to take into account the sensor dynamics, sensor mounting position and need also to account for sensor bias. The measurable variables in the equations of motion are \bar{q} , α , β , V , p , q , r , θ , ϕ , ψ , a_n , a_x , a_y , \dot{p} , \dot{q} and \dot{r} . Practically, many of these are difficult to measure and require special sensors. The minimum variables to measure for estimation are \bar{q} , p , q , r , a_n , a_x and a_y , with the following observation equations.

$$a_n = \frac{\bar{q}S}{mg}C_N + \frac{x_{a_n}}{g}\dot{q} + \frac{z_{a_n}}{g}(q^2 + p^2) - \frac{y_{a_n}}{g}\dot{p} + a_{n_b} \quad (4.18)$$

$$a_x = -\frac{\bar{q}S}{mg}C_A + \frac{z_{a_x}}{g}\dot{q} - \frac{x_{a_x}}{g}(q^2 + r^2) - \frac{y_{a_x}}{g}\dot{r} + a_{x_b} \quad (4.19)$$

$$a_y = \frac{\bar{q}S}{mg}C_Y - \frac{z_{a_y}}{g}\dot{p} - \frac{y_{a_y}}{g}(p^2 + r^2) + \frac{x_{a_y}}{g}\dot{r} + a_{y_b} \quad (4.20)$$

$$p_z = p + p_b \quad (4.21)$$

$$q_z = q + q_b \quad (4.22)$$

$$r_z = r + r_b \quad (4.23)$$

$$\bar{q}_z = \bar{q} + \bar{q}_b \quad (4.24)$$

The dynamic pressure \bar{q} is used in two ways. The first to calculate the velocity and the second is with a combination of other geometric measurements used to non-dimensionalize the force and moment coefficients.

4.3 Identifying the Parameters to be Estimated

The equations of motion and the observation equations have many unknown parameters that need to be identified to describe the dynamics of ZS-GHB. The stability and control derivatives are the most important of these. They describe the dynamic behavior of the aircraft. Unfortunately many extra nuisance parameters need to be estimated. The nuisance parameters are the biases and initial conditions which do not contain any important information, but are needed to produce accurate simulations. Many of these parameters are simply unknown offsets on the sensors or initial conditions for the states of the aircraft.

Altogether there are 48 parameters that need to be identified. There are 25 control and stability derivatives ($C_{Y\beta}$ C_{Yp} C_{Yr} $C_{Y\delta}$ $C_{l\beta}$ C_{lp} C_{lr} $C_{l\delta}$ $C_{n\beta}$ C_{np} C_{nr} $C_{n\delta}$ $C_{N\alpha}$ C_{Nq} $C_{N\delta}$ $C_{A\alpha}$ C_{Aq} $C_{A\delta}$ $C_{m\alpha}$ C_{mq} $C_{m\delta}$ C_N C_A C_Y) 13 bias terms (C_{Y_b} C_{l_b} C_{n_b} C_{N_b} C_{A_b} C_{m_b} a_{n_b} a_{x_b} a_{y_b} p_b q_b r_b) and 7 initial conditions (p q r α β ϕ θ) That are identified from data gathered in flight tests. The 16 physical properties (m I_{xx} I_{yy} I_{zz} I_{xz} I_{yz} I_{xy} x_{cg} y_{cg} z_{cg} x_{a_n} z_{a_n} y_{a_n} S c_b) can be found before any flight tests are done in data sheets and by measurement and estimation experiments on the ground.

Chapter 5

Glider Instruments

5.1 Introduction

One of the tasks of this project was to instrument the ASK-13 glider ZS-GHB so that data could be collected in flight. The sum total of electronic instruments in this particular glider before this project was a radio and an electronic variometer. There were no instruments measuring the data that is needed for parameter estimation. This provided the opportunity to install suitable sensors for parameter estimation and not have to rely on sensors that are already installed that may produce unsuitable data. Firstly it is necessary to define what characteristics are desirable to have in the data gathering system, then install suitable transducers and logging equipment into the glider. Lastly the correct functioning of the sensors is verified and the sensors are calibrated.

5.2 Desirable Characteristics of Sensors Producing Data for Estimation

A measurement system that is designed with the desired characteristics in mind will not, or at least be a very limited, source of error or complication when the data is analyzed. The desired characteristics can be divided into two groups, the transducer characteristics such as noise repeatability, linearity, etc and the data logger characteristics such as sampling frequency, resolution and filtering.

5.2.1 Data Logger Characteristics

An important choice to be made is what the sampling frequency should be. The insights from Maine and Illif in [11] are useful in making this decision. The aircraft dynamics tend to be in the 0-13 Hz range with structural resonances between 20 and 30 Hz. Luckily the glider has no engine or generator so there are fewer vibrations than on a power plane. To stop aliasing of

the nuisance data, such as any vibrations not caused by the response of the glider, pre-filtering of the response data is needed. The question is where to place the corner frequency of the filter. To circumvent this problem of not knowing where to place the cut-off frequency, a much higher sampling rate is used. Where 30Hz sampling frequency is high enough to contain all the response data, reference [11] recommends 100-250Hz, [11] to capture all the other resonances, without folding them over onto frequencies where the aircraft response lies. These unwanted frequencies can later be removed by filtering the data before thinning the data to a lower sample rate.

Data has to be gathered simultaneously from all channels. At one instant all the sensors need to be sampled together. This is important to get as close to simultaneous as possible. If the delay between the sampling of the input signals and the output signals becomes too great the estimator cannot fit the model to the data.

Enough resolution in the amplitude of the signal is needed. Maine and Illif [11] have found that it is possible to do estimations from data where the resolution is 1/10 of the manoeuvre size. What is desirable is a resolution of at least 1/100 of the manoeuvre size. A signal conditioning circuit is necessary to adjust the bias and increase the amplitude of the signals, in order to make the most of the available range of the analogue to digital converter. The signal conditioning block must contribute the minimum possible noise or phase lags to the signals coming from the sensors.

All the data for the same instant should be kept together as a unit with a time-stamp. This will prevent the synchronization between the signals being lost.

5.2.2 Desirable Transducer Characteristics

The process of digitizing a signal will never be able to improve on the original analogue signal. The quality of the measurement comes from the transducer. If the transducer is of poor quality there is a good chance that the measurement will not be accurate. Measurement noise is undesirable. To avoid data corruption by noise the transducer must have a high signal-to-noise ratio. While the ML estimation can still produce good results with measurement noise, it remains preferable to keep noise levels as low as possible.

The accuracy quoted by many vendors is nothing more than an amplitude scale factor. The overall accuracy of a sensor is a combination of more than one characteristic of a sensor. The scale factor is more related to the resolution of the sensor than accuracy. For estimation, the resolution is more important than absolute accuracy [11]. What is important to know is what the sources of error are in the measurement.

Linearity in the measurement is desirable. Most transducers that are quoted with certain non-linearity limits are linear over most of the range but tend to become non-linear at the limits of the transducer's range. Care must be taken on installation to ensure that the normal operation will be in the linear range. Non-linear sensors can also be used but at the cost of added

complexity in processing and calibration, which could add unwanted errors in the data. The installation of the sensor can introduce non-linearities such as dead-bands, stiction and hysteresis. In manufacturing and installation of the sensors effort must be made limit the size of the non-linearities and to avoid saturation.

The sensor needs to take repeatable measurements. A shift in the bias point of the sensor can cause an error in the repeatability. If the data is for estimation purposes it can be corrected by estimating the bias of the signal.

Crosstalk is the measure of how much off-axis excitation will affect the output of the signal. Crosstalk is more applicable to inertial sensors. A good example is that you would not want yaw rate coupling into the roll rate gyro because of the mounting of the gyros not being orthogonal to each other.

The response time of the transducer needs to be suitable for the application. The gain and lag over the expected frequency range needs to be suitable to measure the response data. Installation can worsen the problem. A good example is the mounting of pressure sensors. Adding a long pipe between the pressure port and the sensor element will increase the lag of the sensor.

Mounting of the sensor can influence the accuracy of the measurements that it takes. Control surface deflection instruments need to be as close as possible to the surface avoiding linkages between the transducer and the control surface which can add the sorts of errors mentioned above. Accelerometers should be mounted as close as possible to the center of gravity, to limit the affect angular rates have on the sensors. The influence of the angular rate on the accelerometers can be corrected if the position of accelerometer relative to the center of gravity is known.

5.2.3 Conditions for Sensors with regards to flight safety

Not mentioned in the estimation theory is safety. It is taken for granted that the sensors in aircraft will not compromise the safety of the pilot. For this project particular care was taken with this aspect of sensor installation. The sensors that were installed were not standard aircraft sensors and the aircraft was not designed to have these sensors fitted. All the sensors are retro-fitted in positions deemed suitable for them. The control surfaces are of particular concern. If the sensor on a particular control surface fails in any way, it must not inhibit the function of the control surface. The pilot must still have full control of the aircraft even if the sensor jams. Weak links are mandatory to separate the control surfaces from the transducer. The weak links must not add undesirable characteristics to the sensors that will affect the accuracy negatively.

During a crash the instrumentation must not add to the hazards. The instruments need to be secured in such a way that they will not break loose during impact. This immediately places conditions on the position, encasement and type of mounting that the equipment has to ensure a reasonable crash worthiness.

5.3 Data Logger

The data logging side of the measurement equipment can be broken up into three sections.

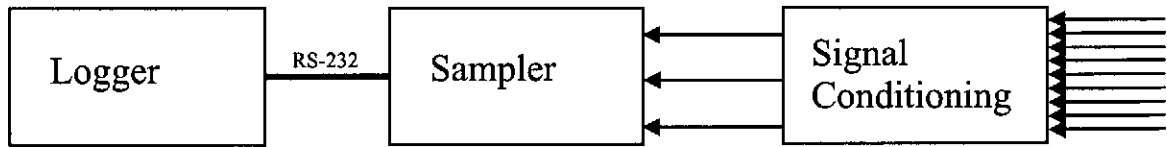


Figure 5.1: *Flow of Data*

5.3.1 Signal Conditioning

The purpose of the signal conditioning is to manipulate the incoming signals from the sensors to utilise the full range of the analogue to digital converter (ADC). This maximum range is obtained by using two consecutive amplifiers. The first is used to get the midrange voltage of the sensor up to the midrange voltage of the ADC and the second gain to increase the peak to peak amplitude to 95% of full scale of the ADC. Another design consideration is flexibility. The dynamic range of the properties that would be measured was unknown. Only after the first flights could the gain of the amplifiers be set. The sensors in the aircraft may need adjustment if the glider has a hard landing. Sensor zero references can shift from the shock.

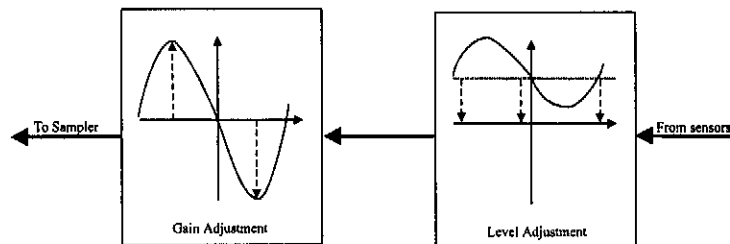


Figure 5.2: *Signal Conditioning Block Diagram*

The schematic of the circuit is available in appendix C.

5.3.2 Sampling

The sampling is done by a data logger that was developed as a final year B.Sc(Eng) project described in [7]. The data logger uses a micro-controller with onboard ADC. The specifications of the data logger are evaluated with the requirements in section 5.2.1.

Resolution: The micro-controller has a 10 bit analogue to digital converter. If the signal conditioning block is set to provide output that covers 95% of the ADC's range, then the 10 bits is sufficient. Of the available 1024 codes 972 will be used.

The ADC is set to take 100 samples per second. This is well above the expected natural frequencies of 13 Hz as well as expected structural resonances at 30 Hz for aircraft in general [11]. The natural frequencies of the glider will be significantly lower than 13 Hz . This sampling frequency will provide sufficient space, in the frequency domain for the off-line digital filtering and resampling that takes place.

Considering the simultaneous sampling of all signals. The micro-controller has only one ADC and to sample all twenty channels that are needed, multiplexers are used. All the channels cannot be sampled simultaneously. The sample time between consecutive channels is 122 μs and between the first and last channels is 2.3ms. Considering that the time between the first channel and the last channel is 23% of the time between samples, the different channels cannot be considered as simultaneously sampled. The data logger is not suitable for the simultaneous sampling required. But the data will be thinned down from 100 samples per second to 20 samples per second for the estimation. At this thinned rate the time between the first and the last channel is 5% of the time between samples. This is acceptable for our purposes.

The logger samples all 20 channels and then transmits the data in one packet. 19 channels are available for sensors and one channel is used to sample a saw-tooth signal generated by the micro-controller for testing. The data is transmitted via an asynchronous RS-232 link at 115 kilobaud. The data is sent as a string of ASCII characters. The total packet length is 42 bytes. Two of the bytes at the beginning are used to identify the beginning of a packet followed by the 20 channels of data 40 bytes long. The sampling is controlled with an enable line which when connected to ground starts the sampling and the transmission of the data.

5.3.3 Logging

The data from the data logger is stored on a hard-disk. In-flight the hard-disk is in a laptop, on the ground it can be in any PC with a RS-232 port and the correct software. The laptop is used for its versatility. The laptop can be programmed easily to change its function, no new electronics need to be built and there is room for expansion. The data is stored the moment it is received as a line of ASCII characters in a text file. No processing of the data is done in flight.

The laptop is configured to suit its task as logger. Once switched on it automatically starts the logging program. When the first data is received the laptop writes the data to a file that uses the date and the current time as filename. The logging process's priority is changed in the Windows 95 scheduler to time critical ensuring that it will dominate the time allocation of the CPU. Every 1000 samples the file is closed and opened again. This ensures that the FAT of the hard-drive is kept up to date in case the laptop is switched off accidentally or loses power in flight, so that the data already recorded is not lost. The parallel port is used as the man machine interface. Plugged into the port is a led that glows when the logging program is running, as well as a switch which closes the program and shuts down the laptop.

5.4 Control Surface Transducers

At the start of this project the ZS-GHB used was undergoing a major overhaul, providing the opportunity to install sensors while the airframe was not covered. There were no sensors of this nature in the glider before the project. All the sensors had to be built and could not be completely tested before the glider was finished and ready for flight. Simplicity was an important consideration to reduce the chances of failure. Replacing one of the sensors inside the glider would be a major exercise. There is limited access to the sensors.

Two types of sensors are needed for parameter estimation, namely sensors that measure the actuators and sensors that measure the response to the actuation. The actuators or inputs to the glider are the control surfaces. The air data sensors measure the current state of the glider's environment. The responses or outputs are the angular rates and translation accelerations measured by the inertial sensors. The input sensors are mounted in the aircraft as close as possible to the control surfaces, to increase accuracy of the measurement. The inertial sensors are mounted in a removable sensor pack (IMU) which also contains the data logging equipment. The IMU (figure 5.3) when installed into ZS-GHB fits inside the fuselage just above the wing roots. This section describes the details of the sensors and their mounting in the glider.

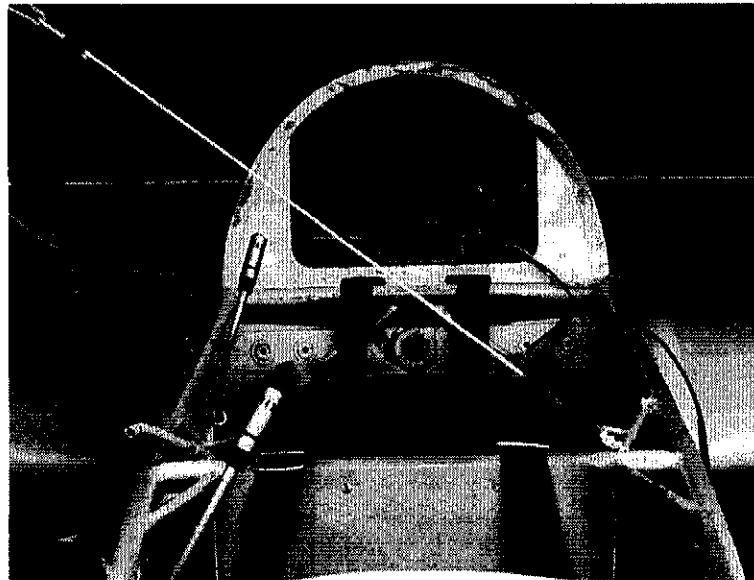


Figure 5.3: *IMU mounted inside ZS-GHB*

The control surfaces that are critical for the parameter estimation are the ailerons, elevator and rudder. Sensors on the air-brakes and trim were also installed to provide data for later experiments. Simplicity being important, it was decided to use a device that responds linearly to deflection angles. The device chosen to measure the deflection is a servo potentiometer. The servo potentiometer is suited for these measurements because it is linear, accurate, stable, is simple to implement and does not require a complex calibration procedure, which if done incorrectly would introduce errors into the estimation. A high quality precision servo potentiometer

was used. The advantages of using this high quality precision potentiometer over lower quality types is the lower linearity error of 0.5%, compared to 2% of other precision potentiometers and 5% of common all-purpose potentiometers. The variation of resistance between potentiometers is not a concern, as each control surface is calibrated once the instrument is installed. The rotational life of the precision servo-potentiometer is higher than normal potentiometers by a factor of 40, giving it up to 10 million cycles. For moving parts that will be in use all the time during flight this extends the functional life of the sensors considerably. The shaft of the particular potentiometer chosen is mounted on bearings. The bearing-mounted shaft adds a greater robustness to the potentiometer and reduces the torque required to move the wiper. A lowest possible starting torque is desired to put the least amount of stress on the weak links to reduce possible dead-band effects on the measurement. The specifications of the Vishay-Sfernice ECS servo-potentiometer used are:

Resistance	10	$k\Omega$
Resistance tolerance	± 20	%
Linearity	± 0.5	%
Temperature Coeff	± 400	$ppm/^{\circ}C$
Power Rating at 40°	1	W
Rotational Life	$> 10^7$	cycles
Mechanical rotation	360	°
Electrical rotation	340 ± 4	°
Starting torque(max)	28×10^{-4}	Nm
Running torque (max)	21×10^{-4}	Nm

Table 5.1: *Specifications for Precision Potentiometer*

For each of the control surfaces there is a unique way of attaching the potentiometer to measure the deflection.

To calibrate the control surfaces, the complete logging instrumentation was used as it is flown during flight tests. The voltages used for the calibration data are those measured by the ADC after the signal conditioning.

5.4.1 Ailerons

The aileron measurement is sensed from inside the fuselage. This is not the most ideal position. The ideal position is at the hinge of the aileron in the wing. It was decided to keep the sensor inside the fuselage to avoid complications and potential sensor damage when the wings are removed and replaced for inspections and out landings. Just behind the rear pilot seat the controls for elevator and aileron split off to the wings and tail. This is where the aileron deflec-

tion measurement is taken. To take the measurement, the universal joint on the control splitter

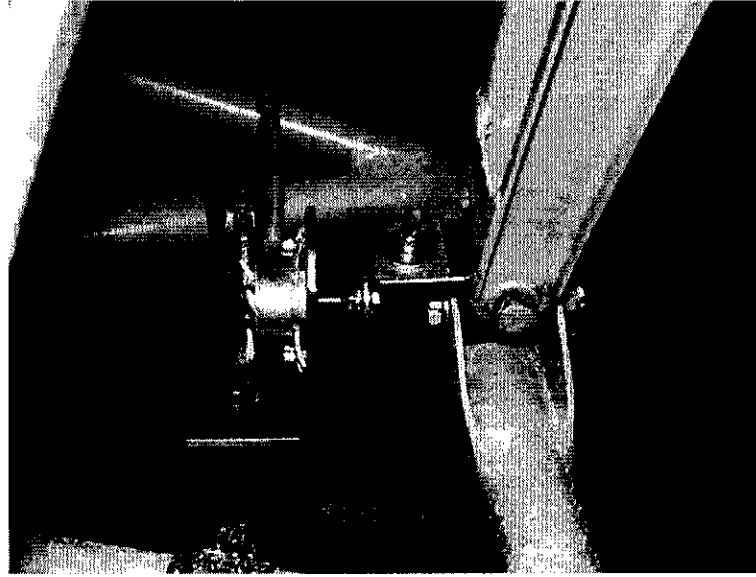


Figure 5.4: *Aileron deflection sensor*

was replaced with a modified universal joint with an added tongue. The tongue is added in a position so that it will only move if the aileron is actuated. The potentiometer is mounted on the airframe and a beryllium copper L-bracket connects the potentiometer to the tongue (figure 5.4). The bracket is held in place with friction from washers on the shaft of the potentiometer. If the potentiometer were to jam, the L-bracket will slip and bend allowing the ailerons to function normally.

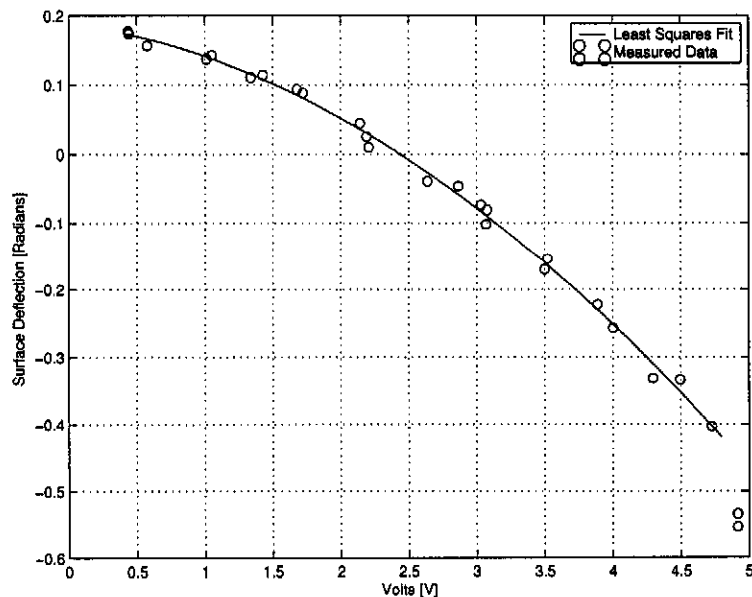


Figure 5.5: *Left aileron calibration measurements*

To calibrate the aileron sensor the glider was held level on the ground and an inclinometer

was attached to the left aileron. Then the aileron was moved through its entire range of movement with measurements being taken at intervals. To check for hysteresis and repeatability the aileron was moved through its range 4 times in both directions. The data reveals that the movement of the aileron deflection is parabolic with the control, becoming more sensitive as the deflection increases in the negative direction. The aileron sensor saturates as the left aileron reaches maximum negative deflection. The saturation is undesirable but the saturated portion of the measurement is seldom used under normal flight conditions and was never entered during any of the flight tests. The saturated region was ignored when fitting a curve to the measured points. The aileron is an anti-symmetric control surface, meaning the two ailerons work oppo-

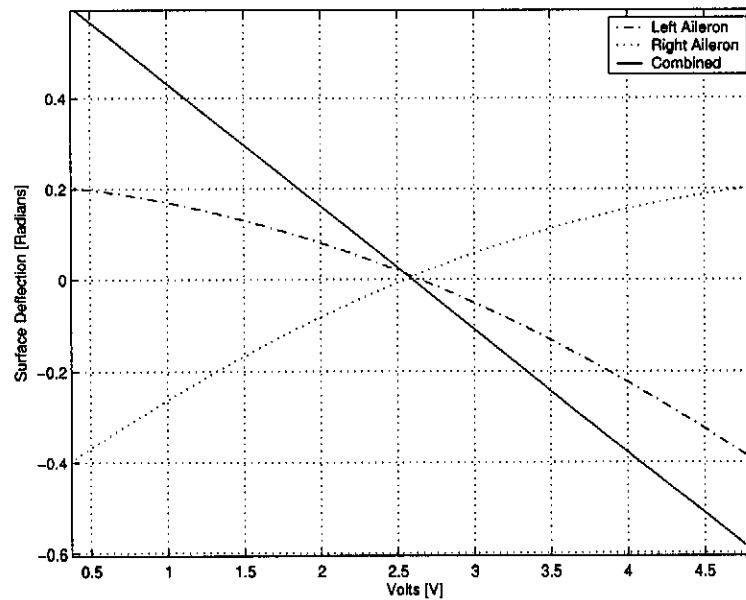


Figure 5.6: *Combined aileron deflection*

site to each other with different magnitudes of deflection. To get the total effect of the ailerons they need to be combined. The standard method for doing this is to subtract the starboard aileron from the port aileron [11]. Applying this standard to the aileron measurement produces a linear calibration curve for the total aileron deflection. Where $\delta_a = 0.6262 - 0.2566V$.

Potentiometer excitation	5 V
Scale factor	$-0.2566 \text{ radians/V}$
Offset	0.6262 radians
RMS noise	92.5 mV

Table 5.2: *Calibration Data for Aileron*

5.4.2 Elevator

The elevator measurement is sensed in the back of the fuselage close to the tail but not right at the elevator. There is one link between the elevator and measurement. The deflection of the elevator is taken off a pin which holds the elevator control rod in a support bracket. The rotation is transmitted to the potentiometer by a lever arm. The lever arm is made up of a short brass rod which is joined to a plastic covered spring that acts as the weak link, bending if the potentiometer jams.

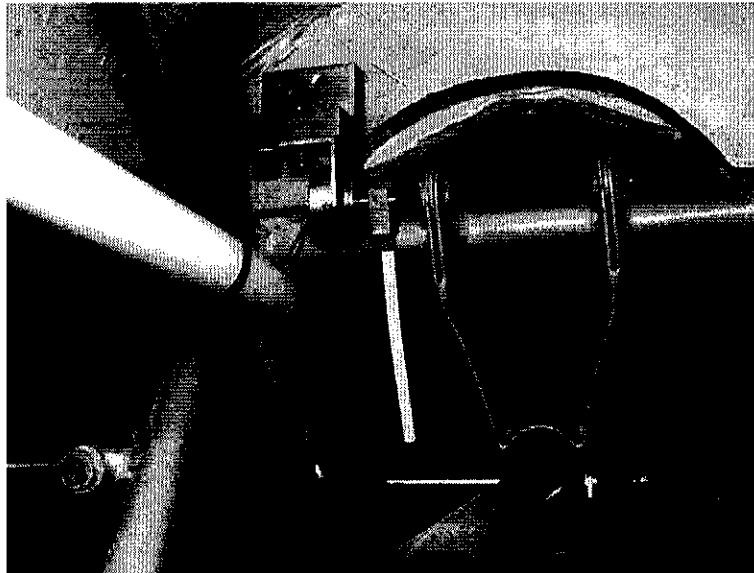


Figure 5.7: *Elevator deflection sensor*

To calibrate the elevator the same process as the one used with ailerons was used. The

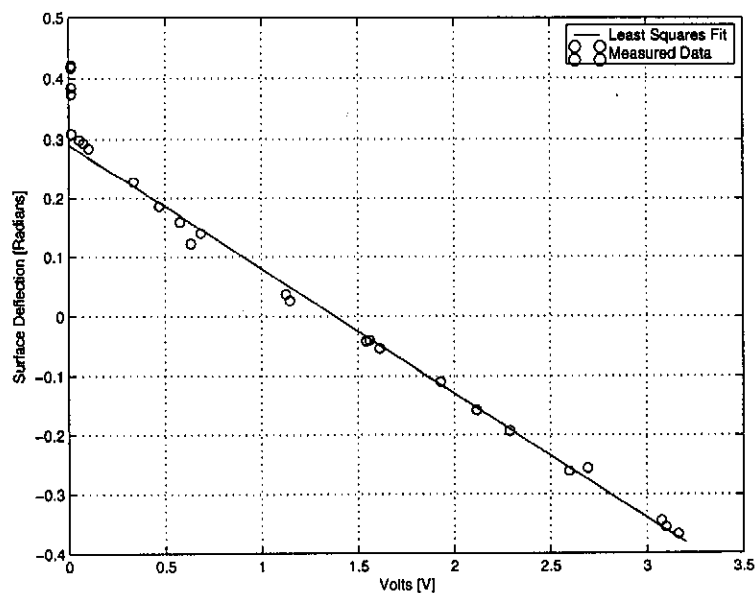


Figure 5.8: *Elevator calibration measurements*

elevator sensor saturates with the control stick far forward and out of the normal flight range. The saturated region was ignored when calculating the scale factors and offset. The least squares curve fitted to the data is a straight line. Where $\delta_e = 0.2908 - 0.2103V$.

Potentiometer excitation	5 V
Scale factor	$-0.2103 \text{ radians/V}$
Offset	0.2908 radians
RMS noise	12 mV

Table 5.3: Calibration data for the elevator

5.4.3 Rudder

The rudder proved to be the most difficult sensor to install. It is very difficult to measure the deflection of the rudder at the hinge. The vertical stabilizer on which the rudder is mounted is a closed plywood construction with no access to its inside. The next option is to take the measurement lower down the actuation chain. The rudder is controlled by two cables that run down the left and right sides of the fuselage, from the pedals back to the rudder. It soon became

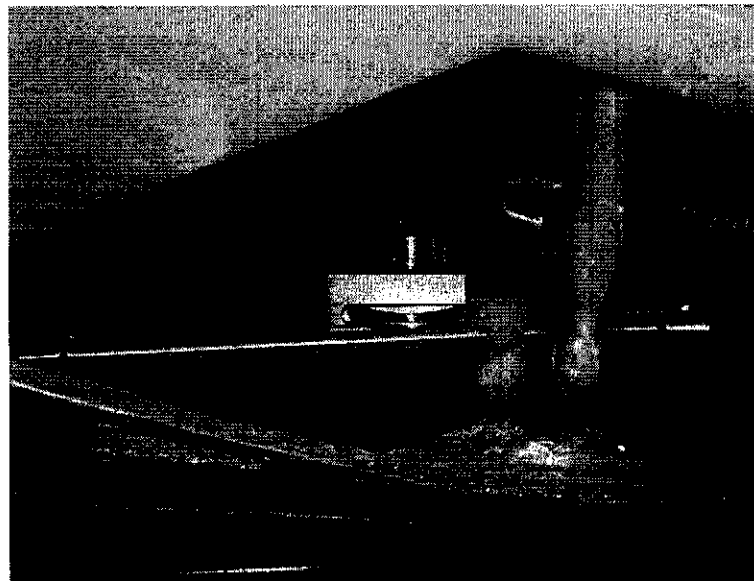


Figure 5.9: Rudder deflection sensor

apparent that the measurement would have to be taken from the cable. The pedals would not be a good place, as they are too far removed from the rudder and the sensor can easily be damaged by the pilot's feet moving around the pedals. The cable is not an ideal place to take a measurement since it can potentially stretch and flap. The cable is 5mm thick with more than 1 ton breaking strain. It is over-specified for its function, so stretch should be minimum if not

zero. The flapping is limited by eyes through which the cable passes along its track. The design for the sensor shown in figure 5.9 was inspired by the mechanics used for zero backlash in PC floppy disk drives.

The sensor uses a potentiometer that is rotated by a pulley. The pulley is driven by a thin cable that is stretched by an aluminium tube that is attached to the rudder cable. To limit flapping, an eye is placed on the sensor to guide the rudder cable and the aluminium tube. The sensor is mounted onto the fuselage just behind the cockpit. The mounting position was chosen because it is the rearmost position where limited access to the sensor is possible if the wings are removed.

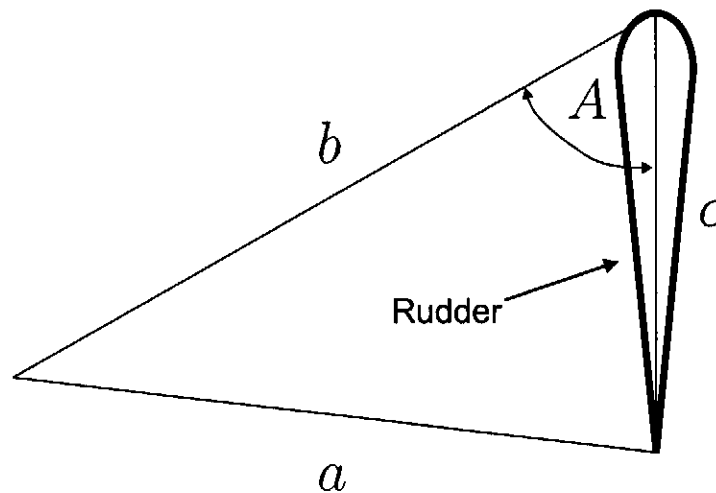


Figure 5.10: *Diagram of Rudder deflection measurement*

The vertical placement of the rudder prevented the use of the inclinometer to take deflection measurements for calibration. The rudder deflection was measured using measuring tape. In figure 5.10 the span of the rudder makes up side-c. A second side is chosen by extending a line from the hinge point of the rudder to an arbitrary point (side-b). This length is fixed. Side-a extends from the end of side-b to the tip of the rudder (side-c). It is side-a that varies with the deflection of the rudder and is recorded. To convert the linear measurement from the measuring tape to the deflection in terms of an angle the cosine rule was used: $a^2 = b^2 + c^2 - 2bc \cos(A)$. The rudder was moved across its full range in both directions 4 times.

Potentiometer excitation	5 V
Scale factor	0.2924 radians/V
Offset	-0.7436 radians
RMS noise	15.6 mV

Table 5.4: *Calibration data for the rudder*

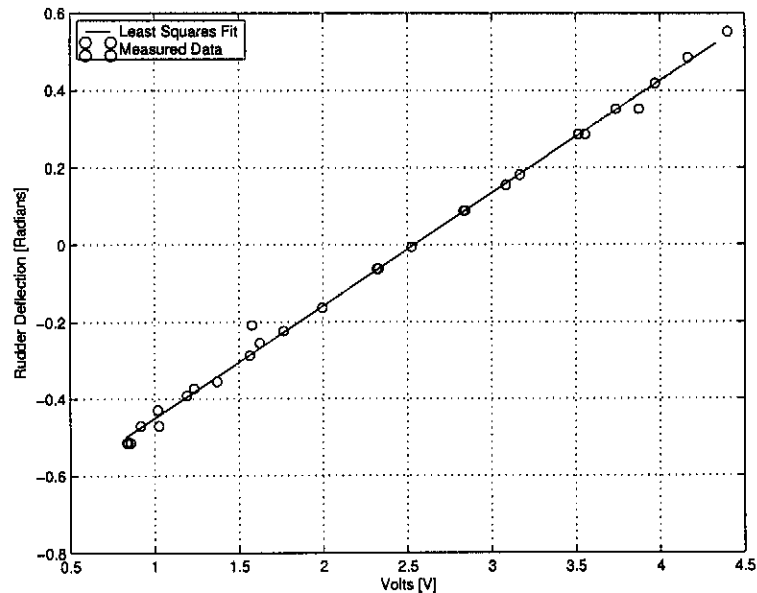


Figure 5.11: *Rudder calibration measurements*

The rudder has no saturation and a least-squares fit (figure 5.11) of a straight line fits the data well and $\delta_r = 0.2924V - 0.7436$.

5.4.4 Trim

The trim control goes from a lever in the cockpit via two steel wires to the Flettner trim tab in the elevator. Its not an electric control. The two wires are used in the same way as the rudder control uses the cable. Compared to the rudder cable the trim wires are a lot stiffer. The same technique as the rudder is used to take the measurement (Figure 5.12). The trim sensor is also mounted aft of the cockpit and is accessible with the wings taken off. The trim is not a critical control surface. It is used to reduce pilot workload during flight. With this in mind this measurement does not have an obvious weak link built in. The trim was calibrated by using the inclinometer. The problem with calibrating the trim tab is that the deflection is a function of the elevator deflection. The trim tab position is not used as an input in the estimation at all, making a full characterization of the trim tab deflection unnecessary. The trim tab was calibrated for the elevator at zero deflection.

Potentiometer excitation	5 V
Scale factor	0.1175 radians/V
Offset	-0.1794 radians
RMS noise	10.7 mV

Table 5.5: *Trim tab calibration*

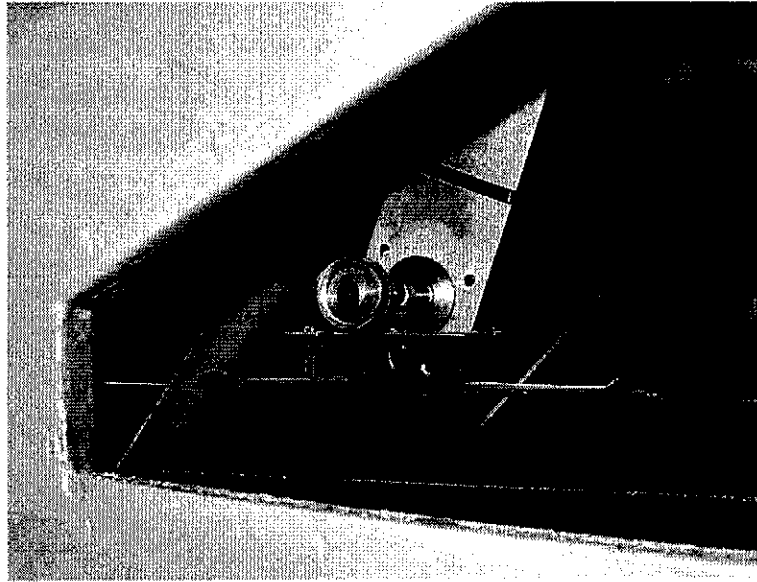


Figure 5.12: *Trim-tab Deflection sensor*

The data shows (figure 5.13) the trim sensor saturates at -0.2 rad this is due to the sensor not covering the complete range of the trim tabs movement. At 0.25 rad the graph shows another saturation occurs, this is where the sensor has moved but the trim tab has not. The trim tab does move in this area when the elevator is deflected in the negative direction. The saturated areas were ignored when the calibration curves were fitted. Over the unsaturated region the trim tab sensor output is a linear function of the trim tab deflection $\delta_t = 0.1175V - 0.1794$.

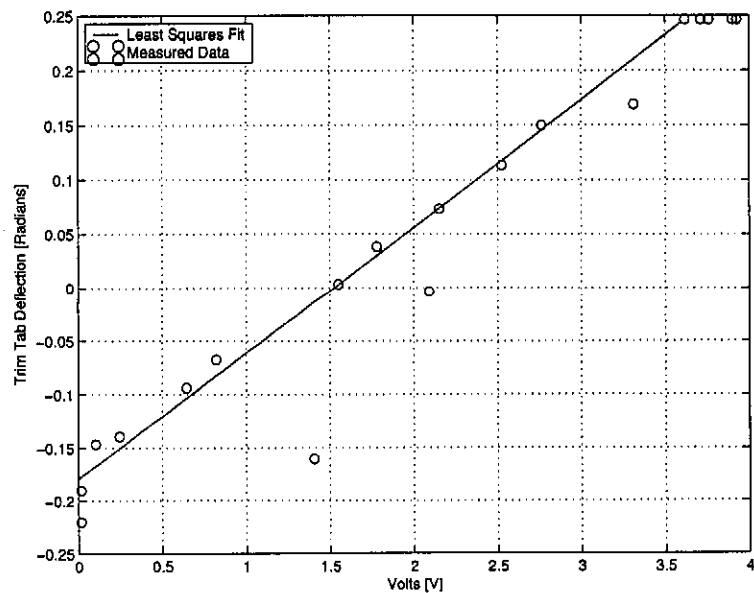


Figure 5.13: *Trim-tab calibration data*

5.4.5 Air-brakes

The Air-brake extension is sensed from the air-brake control arm where the air-brake actuation is split to go into the wings. The control arm is connected directly to the shaft of the potentiometer. As a safety feature a 30mm length of plastic covered spring is used, followed by a 100mm length of 2mm cable. If the potentiometer jams the torque from the control arm will unwind the spring and then the cable, while not hindering the extension of the air-brakes.

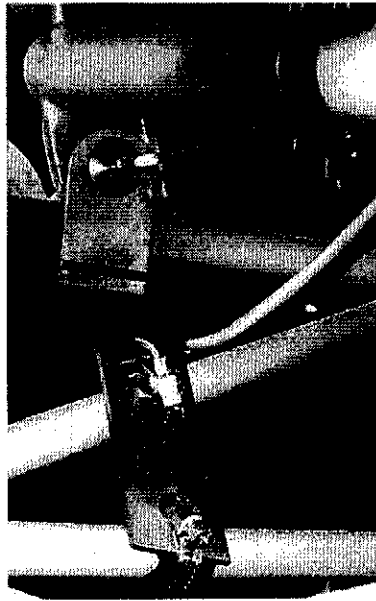


Figure 5.14: *Air-brake extension sensor*

The air-brakes extend vertically out of the top and the bottom of the wings. To calibrate the sensor the air-brakes were extended and retracted with measurements being taken at intervals. The measurements were taken using a tape measure to measure the distance above the wing surface that the air-brake extends. A least-squares fit of a straight line to the calibration data (figure 5.15) produces the following function: $\delta_{air-brake} = 156.3 - 40.29V$.

Potentiometer excitation	5 V
Scale factor	-40.29 mm/V
Offset	156.3 mm
RMS noise	7.35 mV

Table 5.6: *Calibration for the air-brakes*

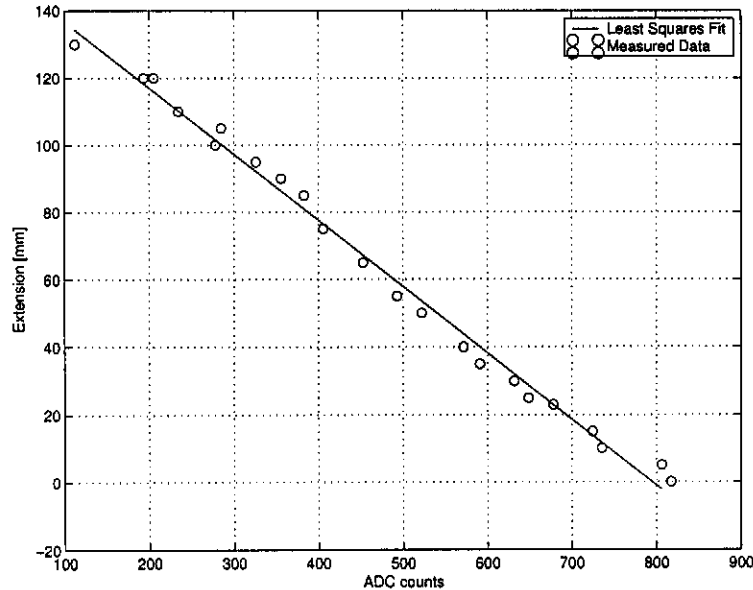


Figure 5.15: *Air-brake calibration data*

5.5 Air data

The air data measurements were not used as inputs or outputs. The dynamic pressure and the pressure altitude are used to define the environment in which the glider is flying. The air data forms part of the initialization for the parameter estimation algorithm.

Instead of adding a complete air data boom, which is normally done in flight tests, it was decided to couple into the existing pressure ports. The pitot pressure port is in the front of the nose and the static pressure ports are on either side of the nose. The pressure transducers were placed underneath the rear pilot's instrument panel. The location of the sensors was chosen for its accessibility to the all the pipes from the pressure ports. There was some concern that the distance between the ports and the sensor elements might introduce a lag in the pressure measurements. Considering that none of the manoeuvres would be aggressive and cause large quick pressure changes and that the pressure measurements are initialization conditions, the lag introduced by the position of the sensors would not affect the parameter estimation.

5.5.1 Dynamic Pressure

The dynamic pressure is measured using a two port peizo-resistive device. The high pressure port is attached to the pitot pressure and the low pressure port to the static pressure. The sensor element used is a 170PC Manufactured by Honeywell. Expected maximum dynamic pressure with the ASK-13 Vne (Velocity not to be exceeded) being 130 km/h is 801 Pa. The circuit schematic is in appendix C. The calibration data for the dynamic pressure sensor is listed in table 5.8 where $\bar{q} = 1097 - 219.5V$.

Range	1.7	kPa
Scale factor	16	mV/kPa
Null output shift	0.03	%FS
Sensitivity shift	± 4	%FS
Repeatability and hysteresis	± 0.25	%FS

Table 5.7: Specifications for the 170PC Differential pressure sensor

Scale factor	-219.5 Pa/V
Offset	1.097 kPa
RMS Noise	0.7 mV

Table 5.8: Calibration data of the dynamic pressure sensor

A water manometer was used to provide the pressure excitation for the calibration of dynamic pressure sensor. The output of the sensor use for calibration is the voltage measured by the ADC.

5.5.2 Static pressure

The pressure altitude sensor is connected to the static port. The sensor used here is a Motorola MPX4115A temperature compensated absolute pressure sensor.

Range	15-115	kpa
Scale factor	45.9	mV/kpa
Accuracy	± 1.5	%FS
Offset stability	± 0.5	%FS

Table 5.9: Specifications for the MPX4115A absolute pressure sensor

The data sheet quotes the accuracy as a combination of the linearity, temperature hysteresis and pressure hysteresis. The glider will seldom fly over 8000ft (75.2 kPa) and for the flight-tests was towed to 4000ft (87.5 kPa) from 650ft MSL (98.96 kPa). The pressure range of the sensor that needs to be monitored is 98.96-84.3 kPa (650-5000ft). After signal conditioning the static pressure is given by $P_s = 104.7 - 4.04V$ has a new scale factor, offset and noise.

The offset will drift as the atmospheric pressure changes. The pressure measured on the ground needs to be noted if the static pressure is going to be used for altitude measurement.

Scale factor	-4.04 kPa/V
Offset	104.7 kPa at STP
RMS Noise	1.1 mV

Table 5.10: *Characteristics of the static pressure sensor*

The formula for obtain altitude above sea-level in feet from the pressure is:

$$Altitude = \left(1 - \left(\frac{P_s}{P_0}\right)^{0.190263}\right) 145.44 \times 10^3 \quad (5.1)$$

where P_s is the measured pressure and P_0 is the standard pressure of 101.325 kPa

5.6 Inertial Sensors

To measure the movement of the glider, six sensors are used. Three angular rate sensors measure the angular rates and three accelerometers measure the translation acceleration.

5.6.1 Angular Rate Sensors

To choose suitable angular rate sensors for the glider, three factors were considered, maximum rate measurable, bias drift and cost. The maximum angular roll rate, which can be the highest angular rate, of the glider is by approximation from the pilots $30^\circ.s^{-1}$. There was no measured data available to base this decision on. The low-cost angular rate sensors available are well above $30^\circ.s^{-1}$, most are in the range from $50^\circ.s^{-1}$ and upwards. The choice of angular rate sensors was determined by cost and bias drift. Drift and cost are directly linked: the more expensive the gyro the lower the drift. The exponential relationship between drift and cost makes the low drift navigation grade sensors too costly for this project.

Technology	Price(US\$)	Drift
Ring laser gyro	> 100000	< 0.0015 °/h
Fiber optic Gyro	1700 - 50 000	0.01 °/h
Mems	100 - 3000	1 - 200 °/h
Peizo gyro	30 - 100	150 - 300 °/h

Table 5.11: *Comparison of angular rate sensor technology with price and drift*

Having zero bias drift is not that critical, as the bias of the angular rate sensors can be estimated as one of the unknown parameters in the parameter estimation. The manoeuvres are short enough to assume the bias drift is zero during the manoeuvre. The cheapest option would be to

use peizo-gyro technology. The disadvantage of the peizo-gyro technology is the hysteresis that is present in the measurements of the cheapest angular rate sensors [12]. The angular rate sensor selected was the Systron Donner AQRS MEMS sensor, its main characteristics are listed in table 5.12. For its price it provides a drift that is low enough to be ignored during manoeuvres. The longest expected manoeuvre of 40 s should have a bias drift of $< 0.02^\circ \cdot s^{-1}$.

Range	± 75	$^\circ \cdot s^{-1}$
Scale factor	33	$^\circ \cdot s^{-1} / V$
Bias drift due to temperature	< 4.5	$^\circ \cdot s^{-1}$
Short term bias stability (100sec)	< 0.005	$^\circ \cdot s^{-1}$
Threshold resolution	≤ 0.004	$^\circ \cdot s^{-1}$
Output noise	0.025	$^\circ \cdot s^{-1} / \sqrt{Hz}$
Alignment error	≤ 3	$^\circ$

Table 5.12: Specifications for an AQRS angular rate sensor

The scale factor of each of the angular rate sensors needs to be calibrated to be certain of the value. Without a specifically designed sensor calibration rate table, a different method had to be used to find the scale factor. In the method used the sensor is moved through a known angle (e.g. 90°) while logging the rate output. Integrating the output of the sensor will then produce the angle through which the sensor has traveled. Knowing the sampling time and logging the rate output in volts will give enough information to calculate the scale factor. To be sure the scale factor is the same for both positive and negative directions, the experiment was completed in both directions and repeated ten times. The final scale factor that is used for the rate sensors is the average from the ten experiments.

$$scale\ factor = \frac{90^\circ}{\Delta t \times \sum (Measurements - Offset)} \quad (5.2)$$

The scale factors for the axes are: X-axis $32.87^\circ \cdot s^{-1} / V$, Y-axis $32.92^\circ \cdot s^{-1} / V$ and Z-axis $32.85^\circ \cdot s^{-1} / V$. These scale factors are quite close to the manufacturer's specifications. The manufacturer also specifies a maximum variation 5% in scale factor that includes the effects of temperature drift. So it can be expected to find the scale factor between $31.35^\circ \cdot s^{-1} / V$ and $34.65^\circ \cdot s^{-1} / V$. Misalignment of the gyro inside its packaging and misalignment of the packaging to the mounting will also have a slight effect on the scale factor with this particular experiment. In section 5.6.4 the misalignment of the gyro's is shown to be small, allowing us to accept the scale factors obtained from the scale factor experiment. The mounting of the angular rate sensors and accelerometers will later be described in section 5.6.3.

5.6.2 Accelerometers

Accelerometers are used to measure the specific forces on the glider. To select suitable accelerometers, similar factors to the angular rates sensors (maximum acceleration, cost and drift) were considered, as well as the signal to noise ratio. The glider will not exceed $2g$ acceleration positive or negative in the flight tests, due to design limitations. A device with the range of -2 to $+2g$ will be sufficient. Hard landings may exceed these limits but are of no use to the parameter estimation for the short-term control and stability derivatives. The recent development of MEMS accelerometers has provided cost effective means of measuring acceleration with reasonable accuracy. At the time of purchase accelerometers from Analogue Devices and Motorola were the most readily available.

Part	Range	Sensitivity	Noise	No of axis
ADXL105	± 5	200 mV/g	$500\mu g/\sqrt{Hz}$	1
ADXL202	± 2	312 mV/g	$500\mu g/\sqrt{Hz}$	2
MMA1270D	± 2.5	750 mV/g	$700\mu g/\sqrt{Hz}$	1

Table 5.13: Comparison of accelerometers

The ADXL202 of Analogue Devices was selected over of the Motorola MMA1270. The ADXL202's full range, as well as drift and low noise are suitable for our application. Its greatest advantage over other devices is that it comes with two accelerometers mounted perpendicular to each other in a single package. This device costs less than the MMA1270D, which is a single axis device. The 2-axes package helps with alignment of the accelerometers as now only one other accelerometer needs to be aligned with the other two. The device also has a built in low-pass filter with the corner frequency set by a discrete capacitor added by the user. This filter reduces the bandwidth of the accelerometers and reduces the noise level on the output.

Range	± 2	g
Nonlinearity	0.2	% of FS
Alignment error	± 1	$^{\circ}$
Alignment error between X and Y	± 0.001	$^{\circ}$
Scale factor	312	mV/g
Scale factor drift due to temperature	0.13	%/ $^{\circ}C$
Bias drift due to temperature	2	mg/ $^{\circ}C$

Table 5.14: Specifications for an ADXL202 accelerometer

If the accelerations are too small and the signal to noise ratio is not large enough, the data will deliver poor estimations. The bandwidth of the the accelerometers is set to 50Hz. Tabulated

in the data sheet of the ADXL202, the smallest acceleration measurable with a bandwidth of 50 Hz is 5mg. The glider should at least be capable of doing manoeuvres in the 100mg range for the flight tests.

5.6.3 Mounting the Inertial Sensors

To gather useful data from the inertial sensors they need to be mounted with a specific orientation to each other. The mounting relative to each other is of particular importance in order to avoid cross talk in the measurements. The three accelerometers are mounted with their sensitive axes orthogonal to each other. The angular rate sensors are mounted orthogonally to each other and aligned with the specific force measurements. Mounting the sensors this way produce six unique signals, without cross-coupling between sensors, that can measure motion with six degrees of freedom. In figure 5.16 the IMU mounting is fixed alongside the data logging circuits.

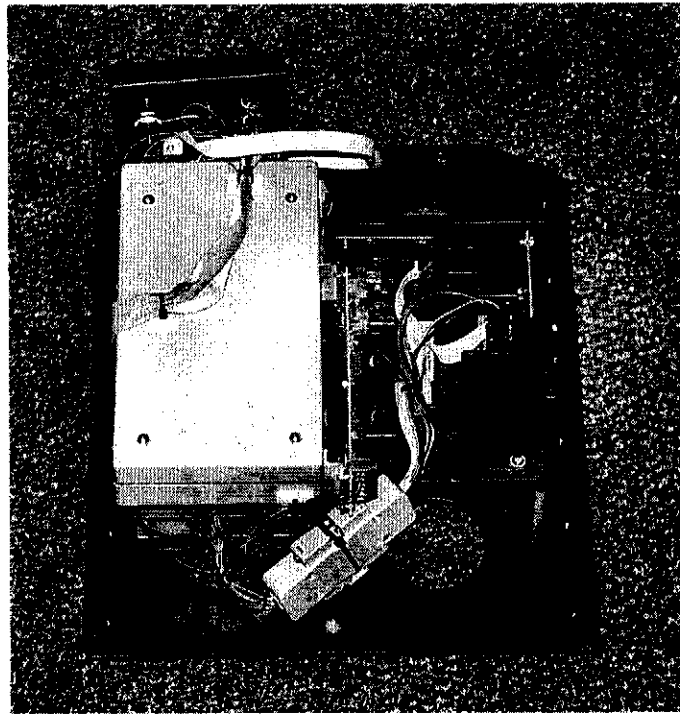


Figure 5.16: *Mounting for the inertial sensors and logging circuitry*

5.6.4 Calibrating the Mounted sensors

Once the sensors were mounted it was necessary to check the alignment of the sensors. Although care was taken during the manufacture of the mounting, there is no guarantee that there is no misalignment of the three orthogonal surfaces used for the sensor mounting. To measure the cross talk the mounting was placed on a level table that can be spun. One of the angular rate sensors was lined up to the tables' axis of rotation. All three angular rate sensors are connected to a data logger. The table was then spun at a known rate approximately $55^\circ.s^{-1}$ and the data from all three angular rate sensors is logged. This process is repeated with each axis of the mounting being aligned with the tables' rotating axis. To calculate the miss-alignment with the other angular rate sensors, the measurement from the excited angular rate sensors is taken and the arccosine with one of the off axis angular rate sensors is calculated to extract the angle. For example, if the x axis rate sensor is stimulated with $55.8^\circ.s^{-1}$ and we obtain a reading of $-0.115^\circ.s^{-1}$ for the y-axis gyro the angle between the two rate sensor's is

$$\begin{aligned}\theta &= \arccos\left(\frac{-0.115}{55.8}\right) \\ \theta &= 90.118^\circ\end{aligned}$$

(5.3)

The perfect angle between sensors would be 90° . This is not realistically obtainable. There will be some error in alignment that will cause the non-zero reading from the remaining angular rate sensors. The calculations are repeated for both off-axis angular rate sensors for each axis being stimulated. The results are listed in table 5.15. For estimating control and stability derivatives the misalignment of the gyros relative to each other is too small to warrant adding correction terms in the measurement equations.

Misalignment with	X	Y	Z
With X-axis excited	0°	0.12°	1°
With Y-axis excited	-0.1°	0°	-0.85°
With Z-axis excited	0.85°	0.73°	0°

Table 5.15: *Misalignment of angular rate sensors*

The alignment of the accelerometers was also checked. The standard force applied for testing is $1g$. A method of tilt sensing is used. Readings at $+1g$ and $-1g$ are taken for each axis. With these readings the scale factor as well as the offset can be determined for each accelerometer. To ascertain how much the alignment is out the accelerometer mounting is rotated by 90° in the direction of one of the two remaining force measurements. If the X-axis alignment is being analyzed the accelerometer mounting is rotated in the direction of either the Z-axis or Y-axis. A reading is then taken, which should correspond to the value for zero force,

which is the offset. The misalignment is found by taking the difference in the measured zero and offset and finding the arcsine of this value with gravity (equation 5.4).

$$\text{misalignment} = \arcsin \left(\frac{\text{Measurement} - \text{Offset}}{1g} \right) \quad (5.4)$$

This would for example if the X-axis is being analyzed and the accelerometer mounting block is rotated in the Z-axis direction given the misalignment angle between the X and Z-axes.

The above method was followed and repeated five times to determine the misalignment between the accelerometers. From these experiments it was found that the accelerometers were mounted with a reasonable accuracy. This is to be expected as one of the devices contains two axis with a guaranteed orthogonal specification of 0.01° and misalignment with the package of less than 1°

Misalignment with	X-axis	Y-axis	Z-axis
X-axis	0	< 0.246°	< 0.246°
Y-axis	-0.355°	0	1.42°
Z-axis	1.8°	2.34°	0

Table 5.16: *Misalignment of the accelerometers*

The table shows how much the sensitive axis of the particular accelerometer is tilted in the direction of another axis on the mounting block. The X-axis accelerometer is quoted with a < because the measurement is below the noise floor of the device. The 2.34° misalignment of the Z-axis in the direction of the Y-axis will introduce a percentage error of 0.083% in the measurement of Z-axis acceleration. Considering that Maine and Illif [11] strongly recommend that no small corrections be used on sensor data, the accelerometers alignment is accepted as is without any correction factors.

5.6.5 Scale Factor Adjustment

The first few times ZS-GHB flew with the instruments installed it was discovered that the gain of the accelerometers and angular rate sensors in the signal conditioning circuit required adjustment. The accelerometer readings were smaller than expected requiring that the gain be increased. The angular rate sensors saturated, their gain had to be reduced. Table 5.17 lists the scale factors and RMS noise of each of the sensors of the IMU after the adjustment.

Measurement	Scale factor	RMS noise	RMS noise in measurement
X gyro	$10.7^{\circ}.s^{-1}.V^{-1}$	$17.9mV$	$0.19^{\circ}.s^{-1}$
Y gyro	$10.7^{\circ}.s^{-1}.V^{-1}$	$18.3mV$	$0.19^{\circ}.s^{-1}$
Z gyro	$10.5^{\circ}.s^{-1}.V^{-1}$	$14.2mV$	$0.15^{\circ}.s^{-1}$
X accelerometer	$2.39m.s^{-2}.V^{-1}$	$29.6mV$	$0.07m.s^{-2}$
Y accelerometer	$2.41m.s^{-2}.V^{-1}$	$34.5mV$	$0.06m.s^{-2}$
Z accelerometer	$5.45m.s^{-2}.V^{-1}$	$9.2mV$	$0.05m.s^{-2}$

Table 5.17: *Inertial sensors scale factors and noise level*

Chapter 6

Preflight Data

There are many parameters in the equations of motion that affect the dynamics of the glider. Some of them cannot be estimated from flight data. These parameters are the physical attributes of the glider that are used to non-dimensionalize the aerodynamic force and moment coefficients. These parameters need to be known before any control and stability derivative estimations can be made. Not all of the physical attributes that were needed are recorded in data books. The moment of inertia values, which are critical, are not published. To collect the unknown data various measurements were taken and ground based experiments executed.

The following data was collected: mass, moments of inertia, position of the center of gravity, the wing geometry and the position of the accelerometers. How this data was collected is the subject for the rest of this chapter.

6.1 Readily Available Data

The important data for the wing is recorded in [13] and is displayed in table 6.1 After the major

Wing span	b	16 m
Mean chord	c	1.09 m
Surface area	S	17.50 m^2

Table 6.1: *Wing data*

overhaul ZS-GHB was inspected and a weight and balance report was completed. The weight and balance report is required by the Soaring Society of South Africa to ensure that the glider is safe to fly and that the center of gravity is inside the required limits for stability. The quantities reported are listed in table 6.2. The information recorded in the weight and balance report is limited. What is still required are the Y and Z positions of the center of gravity and the moments of inertia. Once all this information is available adjustments need to be made to take the pilot's influence on the mass properties into account.

Weight at front wheel	226.6 <i>kg</i>
Weight at rear wheel	99.3 <i>kg</i>
Empty weight	325.9 <i>kg</i>
Distance to front wheel from LE	1520 <i>mm</i>
Distance to Rear wheel from LE	-4974 <i>mm</i>
Center of Gravity	-459 <i>mm</i>

Table 6.2: *Weight and Balance data*

6.2 Center of gravity and accelerometer position

The position of the center of gravity is measured from the Leading Edge Datum. The weight and balance report supplies the X position. The Y position is taken from the symmetry of the aircraft and is zero. The Z position of the center of gravity and the position of the accelerometers had to be measured.

6.2.1 Vertical Center of Gravity

The vertical center of gravity is calculated from experimental measurements taken on the ground. On the ground in the ZY plane the glider rolls around the main and tail wheels. This is very much like an inverted pendulum. By tilting the glider, the center of gravity moves laterally off the center-line and out of a balanced position. The displaced center of mass creates a torque around the wheels, which will cause the glider to fall over to one side. The glider is tilted over till a wingtip touches the ground. The wing then holds the glider up and the resultant torque is zero (Figure 6.1). Measuring the weight at the wingtip and measuring the distance from the wing tip to wheel the torque caused by the center of gravity is calculated. What is of more interest is the moment arm of the center of gravity.

$$\begin{aligned}
 \sum T &= 0 \\
 0 &= l_1 g m_{wing\ tip} - l_2 g m_{glider} \\
 l_2 &= \frac{l_1 m_{wing\ tip}}{m_{glider}} \\
 l_2 &= \frac{8(5.4)}{325.9} \\
 l_2 &= 0.1326\ m
 \end{aligned}$$

(6.1)

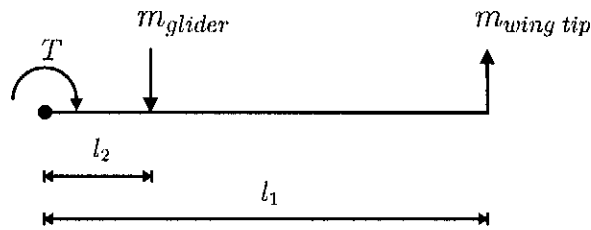


Figure 6.1: Torque and moment arm diagram with glider leaning on its wing

by using trigonometry the moment arm can be related to the vertical position of the center of gravity measured from the main wheel.

$$z_{mw} = \frac{l_2}{\cos \phi_{mw}} \quad (6.2)$$

The angle ϕ_{mw} is the angle from the vertical which the glider is rotated to get the wing tip to touch the ground. Knowing the length of all three sides of the triangle (the distance from the main wheel to the wing-root, the length of the wing and the measured distance between the main wheel and the wingtip) is enough data to calculate the angle from the horizontal to the axis of symmetry of the glider. The angle is calculated by using the cosine rule.

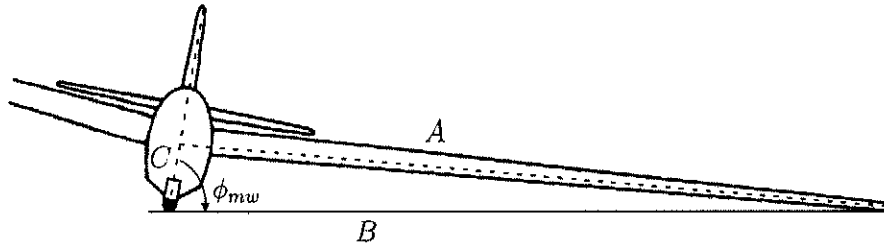


Figure 6.2: Triangle for calculating tilt angle

$$\begin{aligned} A^2 &= B^2 + C^2 - 2BC \cos \theta \\ \phi_{mw} &= \arccos \left(\frac{B^2 + C^2 - A^2}{2BC} \right) \\ \phi_{mw} &= \arccos \left(\frac{8.129^2 + 0.768^2 - 8^2}{2(8.212)(0.768)} \right) \\ \phi_{mw} &= 12.38^\circ \end{aligned} \quad (6.3)$$

Using a measured value for the distance between the wingtip and main wheel, as opposed to a calculated value, automatically takes into account the dihedral of 6° and the possible flexing of

the wing. Having ϕ_{mw} and the moment arm l_2 the vertical position of the center of gravity can be calculated using equation 6.2

$$\begin{aligned} z_{mw} &= \frac{l_2}{\cos \phi_{mw}} \\ z_{mw} &= \frac{0.1326}{\cos (12.38)} \\ z_{mw} &= 0.6183 \text{ m} \end{aligned} \tag{6.4}$$

This position is from the main wheel up. Referencing the position to the Leading Edge Datum gives the vertical center of gravity a position of $z_{cg} = -0.1497 \text{ m}$. What is important to note is that this is the position for the center of gravity for the empty glider and it will change once the pilot is sitting in the glider.

6.2.2 Accelerometer positions

The accelerometer measurements are sensitive to the position of the accelerometers, relative to the center of gravity. The further away from the center of gravity that the accelerometers are mounted the greater the effect angular rates and angular accelerations have on the acceleration measurements. If the positions of the accelerometers are known it is possible to account for the effect of the angular accelerations and rates on the accelerometers. The position of the accelerometers are measured relative to the Leading Edge datum. With this data it will be possible to calculate the position relative to the center of gravity. As mentioned before, the center of gravity will change once a pilot is sitting in the glider and will vary with different pilot masses. The measured position of the IMU is listed in table 6.3.

Direction	Measurment
X	-1.04 m
Y	0 m
Z	-0.202 m

Table 6.3: *IMU position referenced to Leading Edge Datum*

6.3 Mass Properties

The weight and balance report from the Cape Gliding Club provides the total mass of the glider (325.9 kg) but none of the moments of inertia. The equations of motion require the inertia tensor to calculate the angular rates. The symmetry of the glider reduces some of the terms to zero leaving equation 6.5 the moments of inertia around each body axis and the product of inertia I_{xz} .

$$I = \begin{bmatrix} I_{xx} & 0 & -I_{xz} \\ 0 & I_{yy} & 0 \\ -I_{xz} & 0 & I_{zz} \end{bmatrix} \quad (6.5)$$

The rest of this section describes the experiments done to obtain the moments of inertia around each axis and the calculation of the I_{xz} product of inertia for the empty glider.

6.3.1 Pitch Moment of Inertia

The pitch inertia is estimated from data collected during ground experiments. To create the relevant data the glider is pitched around its main wheel. To do this a strain gauge is attached to the handles on the rear of the fuselage. Lifting and lowering the tail with the strain gauge applies a torque around the main wheel and causes a pitching rotation. Measuring the force with the strain gauge, the torque being applied around the main wheel can be calculated if the distance from the strain gauge to the main wheel is known. Logging the pitch rate and force in the strain gauge while lifting and lowering the tail provides enough data to estimate the pitch inertia. There are complications. The center of gravity is very close to the main wheel and

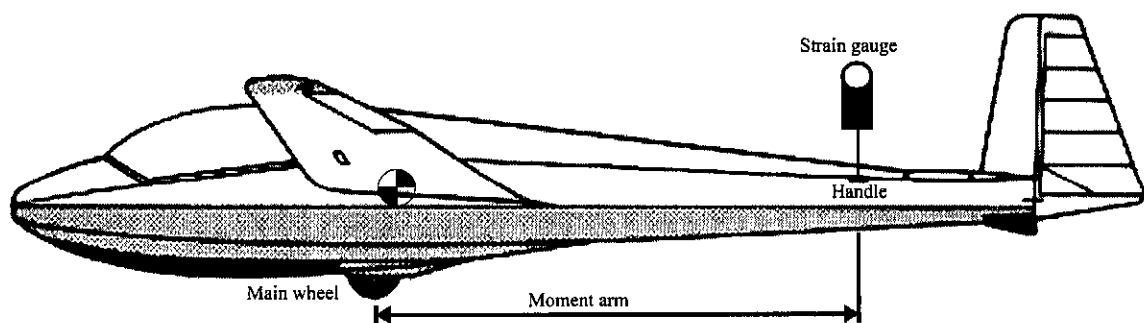


Figure 6.3: Pitch moment of inertia experiment setup

lifting the tail too far will flip the glider onto its front wheel. This is a problem because the strain gauge used can only measure tension. Thus the mass of the glider is needed to provide a restoring force. The restoring force varies with pitch angle. If the movement is kept small the variation in the restoring force is linear and is easy to take into account as feedback. The last complication is that the inertia is measured around the main wheel and not the center of gravity.

This is to be corrected later by using the parallel axis theorem. The equations that describe the

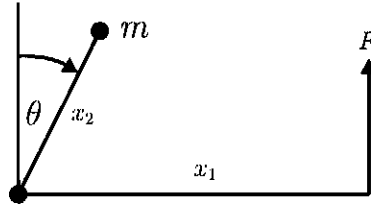


Figure 6.4: Force diagram for pitch inertia experiments

glider pitching on the ground are:

$$\begin{aligned} I\ddot{\theta} &= mgx_2 \sin \theta - x_1 F \\ \ddot{\theta} &= \frac{mgx_2 \theta}{I_{yy}} - \frac{x_1 F}{I_{yy}} \end{aligned} \quad (6.6)$$

Where I_{yy} is the pitch inertia, $\ddot{\theta}$ is the pitch angular acceleration, m is the mass of the glider, x_2 is the distance from the main wheel to the center of gravity, θ is the pitch angle around the main wheel, x_1 is the moment arm for the strain gauge and F is the force measured by the strain gauge.

The state-space model used in the estimation has a couple of additional parameters to it. Biases are added to the force and pitch rate measurements to take into account the unknown sensor biases. The state-space model of the pitching motion on the ground is:

The state equations:

$$\begin{bmatrix} \dot{\theta} \\ \ddot{\theta} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ \frac{mgx_2}{I_{yy}} & 0 \end{bmatrix} \begin{bmatrix} \theta \\ \dot{\theta} \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ -\frac{x_1}{I_{yy}} & T_b \end{bmatrix} \begin{bmatrix} F \\ 1 \end{bmatrix} \quad (6.7)$$

The output equations:

$$\dot{\theta} = [0 \ 1] \begin{bmatrix} \theta \\ \dot{\theta} \end{bmatrix} + [0 \ \theta_b] \begin{bmatrix} F \\ 1 \end{bmatrix} \quad (6.8)$$

The parameters estimated are the initial conditions, the biases, the center of gravity feedback term (mgx_2 in equation 6.7) and the pitch inertia. Although it is not necessary to estimate the center of gravity feedback term it was thought prudent to do so. There is a possibility that errors in the position measurements of the center of gravity exist. By estimating this term, small errors can be adjusted, thus producing a more accurate pitch moment of inertia.

The weighted mean of all the estimated pitch moments of inertia is 956.5 kg.m^2 with $\sigma = 56.59$. The value for σ is calculated from equation 3.11 the expected σ of the weighted least squares. Figure 6.5 is the scatter plot showing the estimated pitch inertia and one sigma values for the error bounds. Although the error bounds vary from very small to very large there is a tight cluster around the expected value. Using the parallel axis theorem the moment of inertia

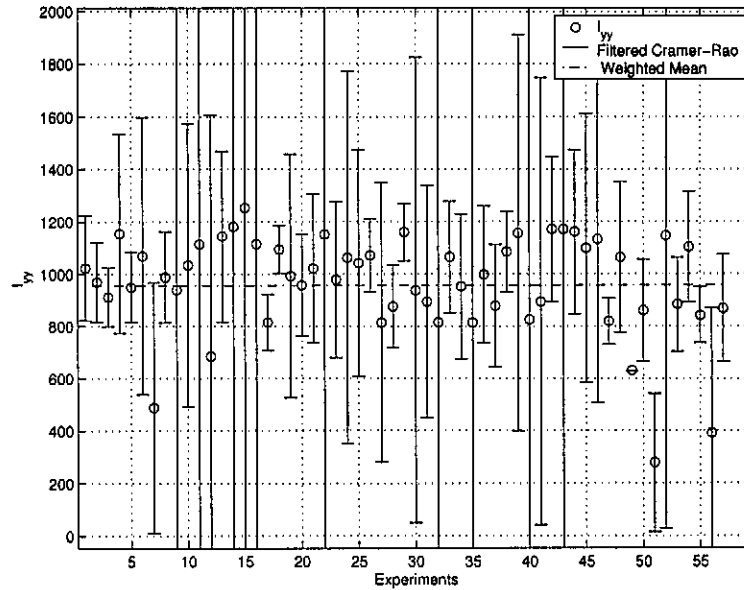


Figure 6.5: Pitch moment of inertia scatter plot

around the center of gravity is calculated. The moment of inertia around the center of gravity is the smallest compared to moments of inertia taken at any other location.

$$\begin{aligned}
 I_{main\ wheel} &= I_{cg} + md^2 \\
 I_{cg} &= I_{main\ wheel} - md^2 \\
 I_{cg} &= 956.5 - 325.9(0.643)^2 \\
 I_{cg} &= 821.8 \text{ kg.m}^2
 \end{aligned}$$

Where d is the distance from the main wheel to the center of gravity, taking vertical and longitudinal position into account.

6.3.2 Roll Moment of Inertia

The roll moment of inertia is found in a similar way to the pitch inertia. A force is applied using the strain gauge to measure the induced torque around the main wheel. The resulting roll rate is then logged. The same form of state-space model for the pitch moment of inertia is used for the roll moment of inertia. In the experiment the strain gauge was attached to the air-brake pivot point. There are two reasons for this choice of attachment. The air-brake pivot point

provides a strain resistant hard point around which the strain gauge rope was easily looped. This attachment avoided including the flexible outer parts of the wing in the moment arm. This attachment is better than the wingtip for the data collection because the wing bending modes are stimulated less, reducing a possible source of process noise in the data. To provide more of a restoring force to keep the strain gauge rope in tension, a weight is added to the end of the wing. The weight on the end of the wing also moves the center of gravity off the center-line, providing a large range of rotation around the main wheel to do the experiments with. The weight on the end of the wing changes the torque equations slightly when compared to equation 6.6.

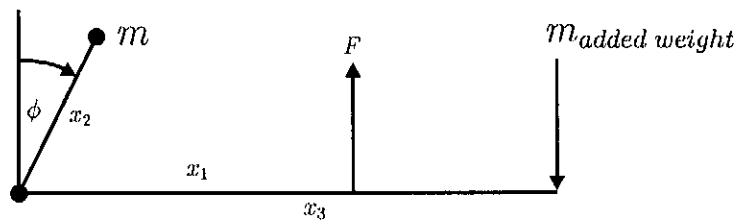


Figure 6.6: Force diagram for roll inertia experiments

$$\begin{aligned}
 I_{xx} \ddot{\phi} &= mgx_2 \sin \phi - x_1 F + x_3 m_{\text{added weight}} g \cos(\phi + 6^\circ) \\
 \ddot{\phi} &= \frac{mgx_2 \phi}{I_{xx}} - \frac{x_1 F}{I_{xx}} + \frac{x_3 m_{\text{added weight}} g}{I_{xx}}
 \end{aligned} \tag{6.9}$$

The term for the added weight in equation 6.9 is also a function of ϕ but has the dihedral angle of the wings added to it. This term varies 0.5% of its maximum over the entire range of the roll moment of inertia experiment. To simplify the equation this term is kept constant.

If the glider is pulled too quickly it can gain too much momentum. The problem occurs when there is no longer an exciting force and too little restoring torque and the strain gauge loses tension. At that point the measurement of the input is lost, making the identification difficult or even impossible. The data where this has occurred is ignored. The weighted mean of all the estimated roll moments of inertia of the glider is $2800 \text{ kg}\cdot\text{m}^2$ with $\sigma = 9.3715e - 011$. The scatter plot (figure 6.7) shows very little scatter and extremely small Cramer-Rao bounds. This estimation is suspiciously good, but no reason could be found not to have confidence in the estimated moment of inertia. The matching of the simulated and gathered data is good, the Cramer-Rao bounds are small and the initial conditions estimated with the inertia are all within reasonable bounds where they can be expected. The weighted mean value is accepted as a reasonable estimate of the roll moment of inertia of the glider. This inertia is around the main wheel and includes the inertia of the added weight. The inertia is referenced to the center

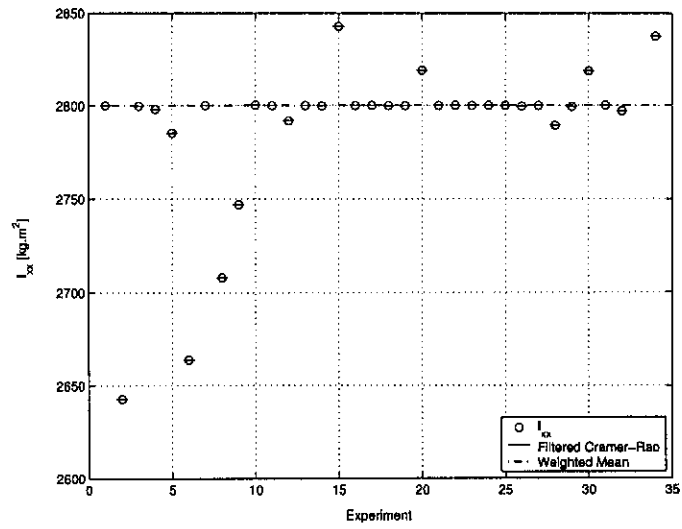


Figure 6.7: Roll moment of inertia scatter plot

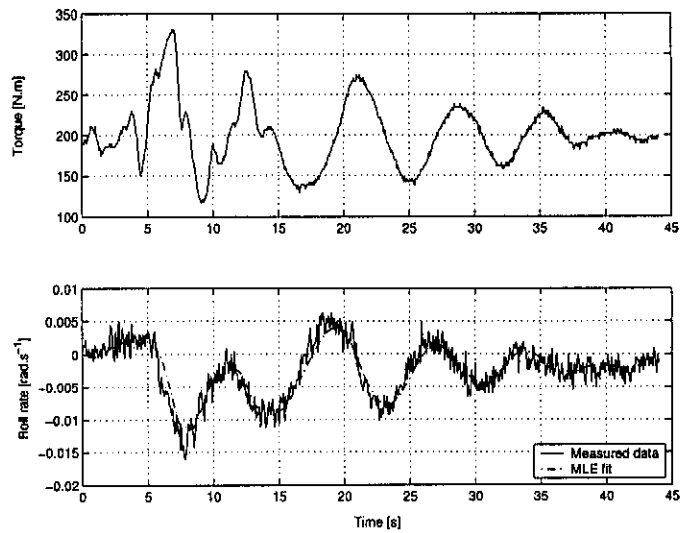


Figure 6.8: Curve fitting of a roll inertia experiment

of gravity, using the parallel axis theorem (equation 6.9) and the inertia of the added weight is removed. The roll moment of inertia around the center of gravity is $2544 \text{ kg}\cdot\text{m}^2$.

$$\begin{aligned}
 I_{main\ wheel} + I_{added\ weight} &= I_{cg} + md^2 \\
 I_{cg} &= I_{main\ wheel} - md^2 - I_{added\ weight} \\
 I_{cg} &= 2800 - 325.9(0.618)^2 - 2.0542(16/2)^2 \\
 I_{cg} &= 2544 \text{ kg}\cdot\text{m}^2
 \end{aligned}
 \tag{6.10}$$

6.3.3 Yaw Moment of Inertia

The yaw moment of inertia is found in a similar manner to the roll and pitch moments of inertia. The difference is that the force in opposition to the strain gauge is supplied by an elastic cord and to get close to friction free yawing, the glider is pushed up onto a turnstile. The tail wheel is

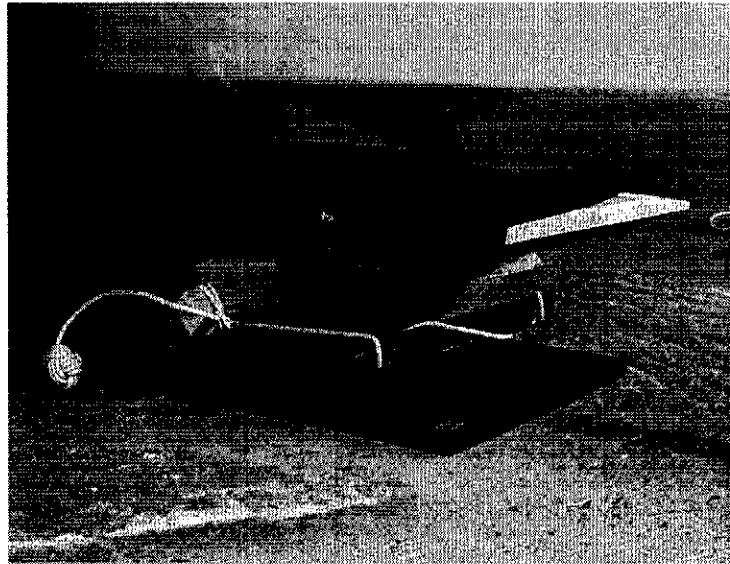


Figure 6.9: Turnstile for Yaw moment of inertia experiments

put onto rollers to allow the main wheel to swivel freely on the turnstile. The ideal attachment point for the strain gauge and the elastic cord is at the handles on the rear of the fuselage. This provides two forces with the same moment arm applying a torque around the main wheel. To describe the interaction between the forces, and angular rates, equation 6.11 was used

$$I_{zz}\ddot{\psi} = x_1 F_{elastic} - x_1 F \tag{6.11}$$

A complication is that the spring constant of the elastic cord is unknown. The spring constant was experimentally determined by moving the glider over the range of deflection used for the

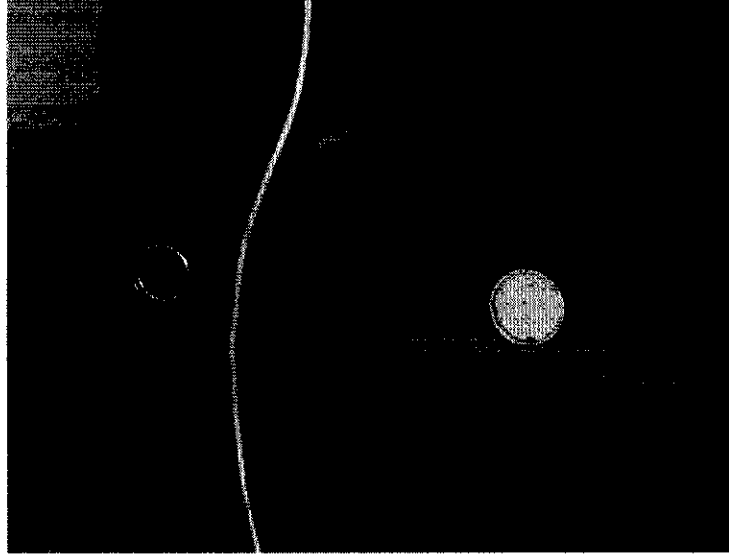


Figure 6.10: *The rollers used to free the tail wheel*

inertia experiments, and at 5cm intervals measuring the force in the strain gauge. The range that the glider can yaw was limited to 33cm. This is due to the limited range of the roller set-up for the tail wheel. It needs to be noted that the displacement was measured at the tail wheel, while the force was applied at the handle which is approximately a meter closer to the main wheel. From the data collected the elastic cord has a linear spring constant of $K = 0.14076 N.mm^{-1}$ over the range used. It would seem that if a long piece of elastic cord is used (approximately 2m) with a very short displacement the spring constant is linear. The spring constant was calculated in $N.mm^{-1}$ but would be more useful as the spring constant in terms of the yaw angle of the glider. The conversion is done by equation 6.12

$$\begin{aligned} K_{\psi} &= Kx_4 \sin \psi \\ K_{\psi} &= Kx_4 \psi \end{aligned} \tag{6.12}$$

where x_4 is the distance from the main wheel to the tail wheel. The angle through which ψ moves is small, allowing the linearization of $\sin \psi$ to ψ . Using equation 6.12 in equation 6.11 the yaw motion can be described by:

$$\begin{aligned} I_{zz}\ddot{\psi} &= x_1 F_{elastic} - x_1 F \\ I_{zz}\ddot{\psi} &= x_1 Kx_4 \psi - x_1 F \\ \ddot{\psi} &= \frac{x_1 Kx_4 \psi}{I_{zz}} - \frac{x_1 F}{I_{zz}} \end{aligned} \tag{6.13}$$

The weighted mean of the yaw moments of inertia estimated is $3265 kg.m^2$ with $\sigma = 0.0478$. The scatter plot of the estimation figure 6.11 shows very small Cramer-Rao bounds which is

good, although the scatter of the parameters appears large. If figure 6.12 is consulted you will notice that 8 out of the 12 experiments are within 3% of the weighted mean, which is a tight cluster.

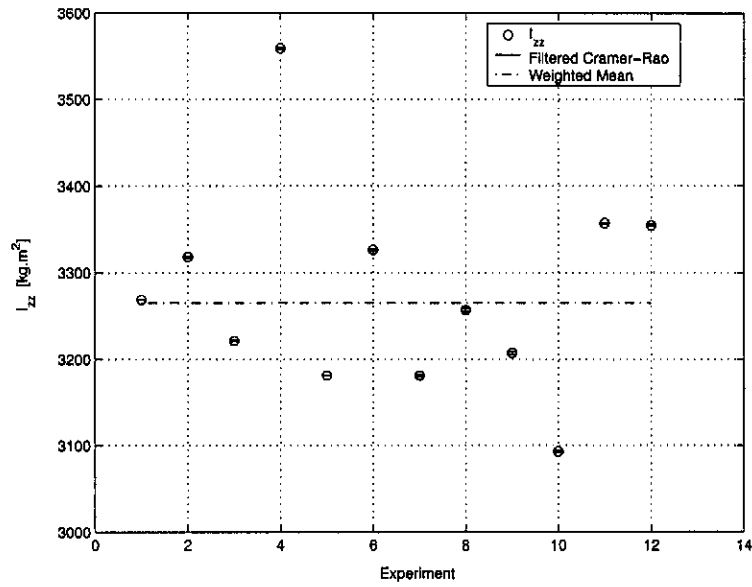


Figure 6.11: Scatter plot of Yaw moment of inertia estimations

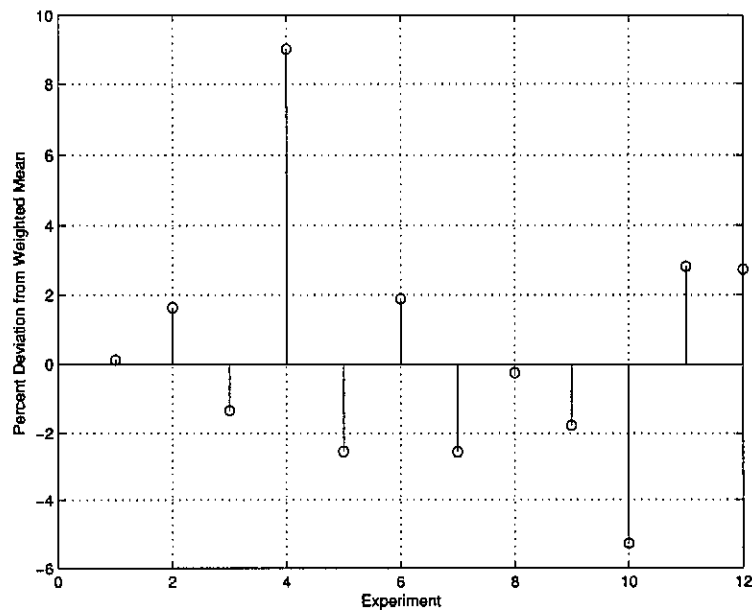


Figure 6.12: Percent deviation from weighted mean of estimated yaw moments of inertia.

As with the other two moments of inertia, this moment of inertia has also been found around the main wheel and not the center of gravity. Using the parallel axis theorem (equation 6.9) the yaw moment of inertia around the center of gravity is 3254 kg.m^2

6.3.4 Products of Inertia

The product of inertia is defined with respect to two orthogonal planes, as the product of the mass of the element and the perpendicular distance from the planes to the element [3]. The product of inertia for the planes y-z and x-z for a body made up of differential elements is $I_{xy} = \int_m xy \, dm$. The product of inertia can be positive, negative or zero, depending on the signs of the coordinates of the mass elements. If one or both of the orthogonal planes are planes of symmetry, then the product of inertia is zero.

The products of inertia for the glider are needed to complete the inertia tensor. Experimental techniques for determining the products of inertia do exist, but they require expensive machinery which is not available. Thus experimental determination of the products of inertia is not a viable option for this project. The products of inertia were calculated using the mass properties of the fuselage and the wings and their relative positions from the center of gravity. The glider's symmetry in the x-z plane reduces the number of calculations that are needed. The symmetry ensures that the I_{xy} and I_{yz} products of inertia are zero, leaving only I_{xz} to be calculated.

To calculate the product of inertia, the masses of the various components and the positions of the center of gravity of each component relative to the center of gravity of the entire glider are needed. The masses and positions are listed in table 6.4. Having no information for the center of gravity of the wings, the position was assumed by considering the internal structure of the wings. The product of inertia I_{xz} is then given by equation 6.14.

Measurement	Wings	Fuselage
Mass	161 kg	164.9 kg
X	0.1790 m	-0.1757 m
Z	-0.1497 m	0.1462 m

Table 6.4: Mass and position of wings and fuselage relative to center of gravity

$$I_{xz} = x_{fuselage} z_{fuselage} m_{fuselage} + x_{wing} z_{wing} m_{wing}$$

$$I_{xz} = -8.637 \, \text{kg.m}^2$$

The completed inertia tensor is:

$$I = \begin{bmatrix} 821.8 & 0 & 8.637 \\ 0 & 2544 & 0 \\ 8.637 & 0 & 3254 \end{bmatrix} \quad (6.14)$$

Note that I_{xz} is approximately 1% of the roll inertia. It is very small and could be ignored. The usefulness of the inertia tensor calculated here is questionable. It does not include the effect

of the pilot. Considering that the glider never flies without a pilot it will never be used in the above form. The inertia tensor needs to be adjusted as described below to take the influence of the pilot's mass into account.

6.3.5 Taking the Pilot into account

The ASK-13 is a training glider with space for two pilots. For the flight tests there was only one pilot. The pilot's mass and position of that mass have a large effect on the position of the center of gravity, the moments of inertia and the products of inertia. To illustrate the importance of taking the pilot's mass into account, a pilot weighing 75 kg is 19% of the take-off weight of the glider.

The pilot's mass distribution needs to be modeled. The simplest model would be a sphere of uniform density. The radius used is 0.6 m a third of an average person's height. The moments of inertia for the pilot would then be given by equation 6.15.

$$I_{xx} = I_{yy} = I_{zz} = \frac{2}{5}m_{pilot}r^2 \quad (6.15)$$

The center of gravity is located approximately 0.2 m above the middle of the seat where the pilot is sitting.

The new center of gravity for the glider is calculated first, to provide the new reference point for the moments and product of inertia. The Y position remains at zero. To calculate the X position

$$x_{cg} = \frac{x_{old\ cg}m_{empty} + x_{pilot}m_{pilot}}{m_{empty} + m_{pilot}} \quad (6.16)$$

$$x_{cg} = \frac{-0.459(325.9) + 0.920(75)}{325.9 + 75} \quad (6.17)$$

$$x_{cg} = -0.201\ m$$

To calculate the Z position

$$z_{cg} = \frac{z_{old\ cg}m_{empty} + z_{pilot}m_{pilot}}{m_{empty} + m_{pilot}}$$

$$z_{cg} = \frac{0.146(325.9) + 0.294(75)}{325.9 + 75} \quad (6.18)$$

$$z_{cg} = 0.174\ m$$

The new center of gravity positions are referenced to the Leading Edge datum, with a positive X-axis forward and Z-axis down. The new center of gravity is now used to calculate the moments of inertia.

The new moments of inertia are calculated by using the known moments of inertia and the center of gravity around which they were calculated. The parallel axis theorem (equation 6.9) is used to calculate the moment of inertia around the new center of gravity.

$$I_{xx\text{new}} = I_{xx\text{old}} + m(y^2 + z^2) \quad (6.19)$$

The two moments of inertia, empty glider and pilot, are then added together to form the new moment of inertia. The new values are listed in table 6.5.

	ZS-GHB	Pilot	Total [$kg.m^2$]
I_{xx}	822	51.623	962.36
I_{yy}	2565.9	461.16	3027.1
I_{zz}	3275.7	456.47	3732.2

Table 6.5: Moments of inertia of ZS-GHB 75kg pilot in the front seat around combined center of gravity

The products of inertia also need to be adjusted. The simplest method of doing this would be to add a term for the pilot into the original calculations. This produces equation 6.20.

$$I_{xz} = x_{fuselage}z_{fuselage}m_{fuselage} + x_{wing}z_{wing}m_{wing} + x_{pilot}z_{pilot}m_{pilot} \quad (6.20)$$

$$I_{xz} = -0.7955 \text{ kg.m}^2 \quad (6.21)$$

Now that the all the mass properties are known, the data that is collectible before flight tests begin is complete.

Chapter 7

Lateral Derivative Estimation

It is well known [1] that the motion of conventional aircraft can be decoupled into the lateral and longitudinal modes of motion.

This section describes the lateral derivatives estimation. It starts with the design of the manoeuvres to provide suitable data. The state-space model used to describe the lateral dynamics of the glider is then defined, and the pre-processing that is done to the data before the estimation algorithm is run is described. Finally the parameters estimated from the data are presented and discussed.

7.1 Flight Test Procedure

To perform identification, manoeuvres have to be flown to suitably stimulate the dynamic response of the aircraft. The tests are preferably done in non-turbulent air, which generally means there is little available lift to regain the altitude lost during the manoeuvre. The typical flight sequence is as follows:

1. Data logging is started with the glider on the ground
2. The glider is towed up to 4000 *ft* and released
3. The pilot then starts doing manoeuvres, returning to stable trimmed flight at a predetermined speed between manoeuvres.
4. If the pilot finds lift he attempts to gain altitude to prolong the flight so that more manoeuvres can be done.
5. More manoeuvres are done.
6. When the glider reaches 1500 *ft* the pilot must commit to landing and manoeuvres are stopped.
7. The glider lands and logging is stopped.

During a flight both lateral and longitudinal manoeuvres can be done. The same wave-forms can be used on all the control surfaces to produce data.

There are a few factors of glider flight that affect the data from experiments negatively. Although helpful in prolonging the flight-tests, the updrafts can have a negative side-effect. The turbulence that the updrafts cause in manoeuvres will enter the state-space model as unmodeled process noise. This increases the error in the estimate parameters. The ideal would be to do a flight test with all the manoeuvres at a particular altitude with no turbulence. With a glider this is very difficult, as it has no thrust to maintain altitude and when there are updrafts the air is turbulent.

Particular attention is paid to the choice of manoeuvres that the pilot flies. To get a good estimate of the parameters the manoeuvre must excite all the modes of the glider. In research done by Shafer [19] it was found that square-wave inputs provided better data for estimation than sine wave inputs. The impulse and doublet (positive then negative pulse) are good examples of typical manoeuvres. The four-frequency manoeuvre, a combination of a square wave

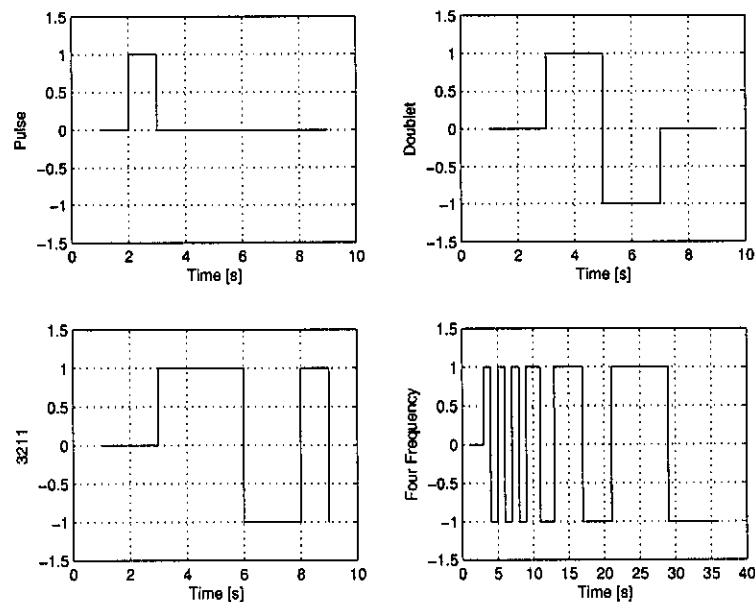


Figure 7.1: Typical manoeuvre shapes.

with four different frequencies, has a good broad range of frequencies and extended stimulation time. It provides good data for estimation purposes and it is easier for the pilot to fly than the traditional 3211 manoeuvre used in flight tests. The Four-frequency manoeuvre is made by first moving the control backwards and forwards four times as quickly as possible then twice at half the fastest speed, then once at a quarter of the fastest speed and once at an eighth of the speed. The pilot can maintain the timing by counting to four with each step in frequency. In reality, the pilot may stop the manoeuvre early if the glider orientation becomes dangerous. For this project the four-frequency is the manoeuvre most often performed during the flight-tests. The decision to use this manoeuvre is based on previous success in [16] and that a large portion of

the manoeuvre can be done before large, potentially dangerous, angular rates have developed in the response. Intermittently impulses, doublets and 3211's were also used to provide a variety of manoeuvres. The amplitude of the manoeuvres is important. The estimation algorithm works well if there is a large difference between the signal amplitude and the noise amplitude. The pilot is therefore instructed to make the input signals as large as possible, but not too large as this would accentuate the non-linearities that exist in the equations of motion.

The lateral motion involves two control surfaces, the rudder and the ailerons. In normal flight the ailerons and rudder are always used together to make co-ordinated turns. The rudder is used to counteract the adverse yaw that the ailerons cause. For the parameter estimation the effect of each control surface is needed separately. To do this in each manoeuvre first the aileron, then the rudder completes the test waveform. The result is data that has the glider's response from independent control actions from both rudder and aileron.

7.2 Lateral Equations of Motion in a State-Space Model

This section gathers together all the equations of motion involved with the lateral motion. A state-space model is then formed from the equations with the accompanying linearizations to be used in the estimation algorithm.

7.2.1 Lateral Equations of Motion

The state and observation equations as described here originate from Maine and Illif,[11] and are suitable for most estimation applications. These state equations are based upon the equations of motion in section 4.2. Various simplifications have been made and are best seen by comparing the equations below with section 4.2. The gravity term about the measured Euler angle ϕ is linearized. Other terms are linearized with measured data and bias unknowns are added. The state equations are:

$$\dot{\beta} = \frac{\bar{q}S}{mV}C_Y + p(\alpha_b + \sin \alpha_c) - r \cos \alpha_c + \frac{g}{V} \sin \phi \cos \theta \quad (7.1)$$

$$\dot{p}I_{xx} - \dot{r}I_{zz} = \bar{q}SbC_l + qr(I_{yy} - I_{zz}) + pqI_{xz} \quad (7.2)$$

$$\dot{r}I_{zz} - \dot{p}I_{xx} = \bar{q}SbC_n + qp(I_{xx} - I_{yy}) - qrI_{xz} \quad (7.3)$$

$$\dot{\phi} = p + r \tan \theta \sin \phi + q \tan \theta \cos \phi \quad (7.4)$$

The observation equations are:

$$p_z = p + p_b \quad (7.5)$$

$$r_z = r + r_b \quad (7.6)$$

$$a_y = \frac{\bar{q}S}{mg}C_Y - \frac{z_a}{g}\dot{p} + \frac{x_a}{g}\dot{r} - \frac{y_a}{g}(p^2 - r^2) \quad (7.7)$$

The forces and moment coefficients are functions of the state equations and expand to form

$$C_Y = C_{Y_\beta}\beta + C_{Y_p}\frac{pb}{2V} + C_{Y_r}\frac{rb}{2V} + C_{Y_\delta}\delta + C_{Y_b} \quad (7.8)$$

$$C_l = C_{l_\beta}\beta + C_{l_p}\frac{pb}{2V} + C_{l_r}\frac{rb}{2V} + C_{l_\delta}\delta + C_{l_b} \quad (7.9)$$

$$C_n = C_{n_\beta}\beta + C_{n_p}\frac{pb}{2V} + C_{n_r}\frac{rb}{2V} + C_{n_\delta}\delta + C_{n_b} \quad (7.10)$$

7.2.2 State-space form and additional inputs

A linear state-space form of the equations is needed by the MMLE3 identification toolbox for Matlab. To get the State-space formulation many extra inputs apart from the control surface signals were added. These extra signals are from the addition of a unity bias signal, making it possible to estimate the bias of signals and inputs for signals that exist due to the linearization of the equations.

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & I_{xx} & -I_{xz} \\ 0 & 0 & -I_{xz} & I_{zz} \end{bmatrix} \begin{bmatrix} \dot{\beta} \\ \dot{\phi} \\ \dot{p} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} \frac{\bar{q}S}{mV}C_{Y_\beta} & 0 & (\alpha_b + \sin\alpha) & -\cos\alpha \\ 0 & 0 & 1 & 0 \\ \bar{q}SbC_{l_\beta} & 0 & \bar{q}SbC_{l_p}\frac{b}{2V} & \bar{q}SbC_{l_r}\frac{b}{2V} \\ \bar{q}SbC_{n_\beta} & 0 & \bar{q}SbC_{n_p}\frac{b}{2V} & \bar{q}SbC_{n_r}\frac{b}{2V} \end{bmatrix} \begin{bmatrix} \beta \\ \phi \\ p \\ r \end{bmatrix} + \begin{bmatrix} 0 & \frac{\bar{q}S}{mV}C_{Y_{\delta r}} & \frac{\bar{q}S}{mV}C_{Y_b} & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ \bar{q}SbC_{l_{\delta a}} & \bar{q}SbC_{l_{\delta r}} & \bar{q}SbC_{l_b} & 0 & 0 & 1 & 0 & 0 \\ \bar{q}SbC_{n_{\delta a}} & \bar{q}SbC_{n_{\delta r}} & \bar{q}SbC_{n_b} & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \delta_a \\ \delta_r \\ 1 \\ u4 \\ u6 \\ u7 \\ u8 \end{bmatrix} \quad (7.11)$$

Where:

$$u4 = \frac{g}{V} \sin \phi \cos \theta \quad (7.12)$$

$$u5 = (r \sin \phi + q \cos \phi) \tan \theta \quad (7.13)$$

$$u6 = qr(I_{yy} - I_{zz}) + qpI_{xz} \quad (7.14)$$

$$u7 = pq(I_{xx} - I_{yy}) - qrI_{xz} \quad (7.15)$$

$$u8 = p^2 - r^2 \quad (7.16)$$

Two extra parameters are added so that the effects of the angle of attack can be include in the estimation. They are: $C_a = -\cos\alpha$ and $C_{a_s a} = \alpha_b + \sin\alpha$.

The above equations are of the form $P\dot{x} = Qx + Ru$ The state-space form:

$$\dot{x} = Ax + Bu \quad (7.17)$$

$$y = Cx + Du \quad (7.18)$$

$$(7.19)$$

required by the MMLE3 toolbox are obtain from:

$$A = P^{-1}Q \quad (7.20)$$

$$B = P^{-1}R \quad (7.21)$$

$$(7.22)$$

7.2.3 Output Equations

In the output equations (C and D matrices of the state-space model) only the responses that were measured are incorporated: the roll rate(p), yaw rate (r) and acceleration in the Y direction (a_y). It will be these signals that the estimator will use to make the measured and predicted responses.

$$\begin{bmatrix} p \\ r \\ a_y \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ C_{31} & C_{32} & C_{33} & C_{34} \end{bmatrix} \begin{bmatrix} \beta \\ \phi \\ p \\ r \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ D_{31} & D_{32} & D_{33} & D_{34} & D_{35} & D_{36} & D_{37} & D_{38} \end{bmatrix} \begin{bmatrix} \delta_a \\ \delta_r \\ 1 \\ u4 \\ u6 \\ u7 \\ u8 \end{bmatrix} \quad (7.23)$$

Where using Matlab notation (To improve readability):

$$\begin{bmatrix} C_{31} \\ C_{32} \\ C_{33} \\ C_{34} \end{bmatrix}^T = \begin{bmatrix} \frac{\bar{q}S}{mg} C_{Y\beta} \\ 0 \\ 0 \\ 0 \end{bmatrix}^T - \frac{z_a}{g} A(3,:) + \frac{x_a}{g} A(4,:) \quad (7.24)$$

$$\begin{bmatrix} D_{31} \\ D_{32} \\ D_{33} \\ D_{34} \\ D_{35} \\ D_{36} \\ D_{37} \\ D_{38} \end{bmatrix}^T = \begin{bmatrix} 0 \\ \frac{\bar{q}S}{mg} C_{Y\delta_r} \\ \frac{\bar{q}S}{mg} C_{Y\beta} \\ 0 \\ 0 \\ 0 \\ 0 \\ -\frac{y_a}{g} \end{bmatrix}^T - \frac{z_a}{g} B(3, :) + \frac{x_a}{g} B(4, :) \quad (7.25)$$

7.3 Estimation of Parameters

7.3.1 Pre-Processing of the data

The data collected in the flight-tests is not immediately ready for the estimation algorithm. The data is stored in a text file as ASCII characters. The recorded data is then converted to 20 channels of numeric data saved as separate columns in a text file, a format Matlab can read. The first step in Matlab is to downsample the signals to a $20Hz$ sampling rate using the `resample` command. Before the `resample` function resamples, it filters the data. The filter used to filter the glider data is a 500-order Hamming window with a $-6dB$ gain at $10Hz$. The filter window is chosen for its unity gain in the passband and sharp roll-off at the corner frequency. The high number of orders was used to get a sharper roll-off. The phase shift of the filter is not a concern. The `resample` function uses the `filtfilt` function to filter the data. The `filtfilt` function achieves zero phase shift by passing the data through the filter then flipping the data back to front and passing it through the filter again, correcting the phase shift of the first pass.

After the filtering, the data is placed in a Matlab structure. In the structure each signal is named. Other elements are added to the structure to record miscellaneous information, the weather conditions during the flight, pilot information and history of how the data has been manipulated. Another vector is added to the structure, containing the beginning and end indexes of each manoeuvre. The next manipulation of the data occurs just before the estimation algorithm is run. Here the data is scaled and shifted using the calibration data to convert the signals into engineering measuring units. The signals that originate from the linearization (u_4, u_6, u_7, u_8) are also calculated at this point.

7.3.2 Initial Parameters Values

It is important to choose initial parameter values close to the actual values of the parameters to

be estimated. This can prevent the algorithm from settling in a false minima and speed up the estimation process considerably. Normally when doing parameter estimation on aircraft there would be a set of control and stability derivatives available. The first parameters that were used as initial parameters were the parameters from a Cessna Citation II. After the first successful estimation with the data, the newly estimated parameters were used as the initial parameters for the rest of the manoeuvres.

7.3.3 Parameter Estimation Structure

Complex relationships often exist between initial conditions, biases and the derivatives, in certain cases not all parameters can be estimated together, but all still need to be estimated. The different phases of the MMLE3 algorithm can be used to help circumvent this problem. The first phase of the estimation is a quadratic phase (*pidq*). It is suitable for estimating parameters that form a quadratic cost function. These are typically the parameters that affect the output linearly, such as the initial conditions and bias terms and the derivatives in the B and D matrices. The next phase is the Marquardt (*pidm*) phase where the Levenberg-Marquardt stabilization is used, This phase is suitable to estimate parameters with non-quadratic cost functions and difficult starting conditions. In the Marquardt phase all the parameters can be estimated together or those that were found in the quadratic phase can be left out. Nuisance parameters estimated in the quadratic phase that are correlated to other parameters are left out of the Marquardt phase. Once the Marquardt phase has brought the estimation close to the minimum, the Constrained Newton (*pidf*) phase starts. This phase uses a constraint to ensure a positive-definite measurement noise covariance matrix. In the (*pidf*) phase the sample innovation covariance matrix is used to weight the cost function, to ensure a maximum likelihood estimate. The biases or initial conditions should be estimated in the final phase again to provide biases and initial conditions for the newly estimated parameter.

The following parameters are estimated for the lateral motion:

- *pidq*: The 5 initial values for the states, $C_{Y\beta_b}, C_{Y\dot{\beta}}, C_{l_b}, C_{n_b}, C_a$.
- *pidm*: $C_{Y\beta}, C_{Y\dot{\beta}_r}, C_{l\beta}, C_{l_p}, C_{l_r}, C_{l_{\delta a}}, C_{l_{\delta r}}, C_{n\beta}, C_{n_p}, C_{n_r}, C_{n_{\delta a}}, C_{n_{\delta r}}, C_{a_{S a}}, C_{Y\beta_b}, C_{Y\dot{\beta}}, C_{l_b}, C_{n_b}$.
- *pidf*: $C_{Y\beta}, C_{Y\dot{\beta}_r}, C_{l\beta}, C_{l_p}, C_{l_r}, C_{l_{\delta a}}, C_{l_{\delta r}}, C_{n\beta}, C_{n_p}, C_{n_r}, C_{n_{\delta a}}, C_{n_{\delta r}}$.

To find this structure for the estimation is time consuming. Working from basic principles and knowledge of the system, a starting configuration can be found that does converge. With careful analysis of the error between the simulated responses and the actual data and accuracy indicators, particularly the insensitivities and the GDOP's, the parameters causing the indentifiability problems can be recognized and shifted to a different phase of the estimation, to produce better results. It is important to note that there are no initial conditions and biases estimated in

the final phase. The reason for this being that the algorithm does not converge if these parameters are estimated in the final phase. This is a symptom of over parameterization for the data that is available. Also, the MMLE3 process noise option was never used. The process noise option converts the open-loop estimation to a closed-loop estimation, where the Kalman filter tracks the response by using the Kalman gain to make corrections. This option provides a better fit possibly at the expense of the accuracy of the estimated parameters.

7.4 Results

The data used in the estimation is from three different flight-tests using the same types of manoeuvres. The estimation structure in the previous section resulted in all the manoeuvres producing estimated parameters, which provide a reasonably good response fits with the measured data. Figures 7.2, 7.3 and 7.4 show the typical fit that was achieved with most of the manoeuvres. In appendix A are the plots of each manoeuvre and plots of the error between the measured and simulated responses. In the following subsections the results will be discussed.

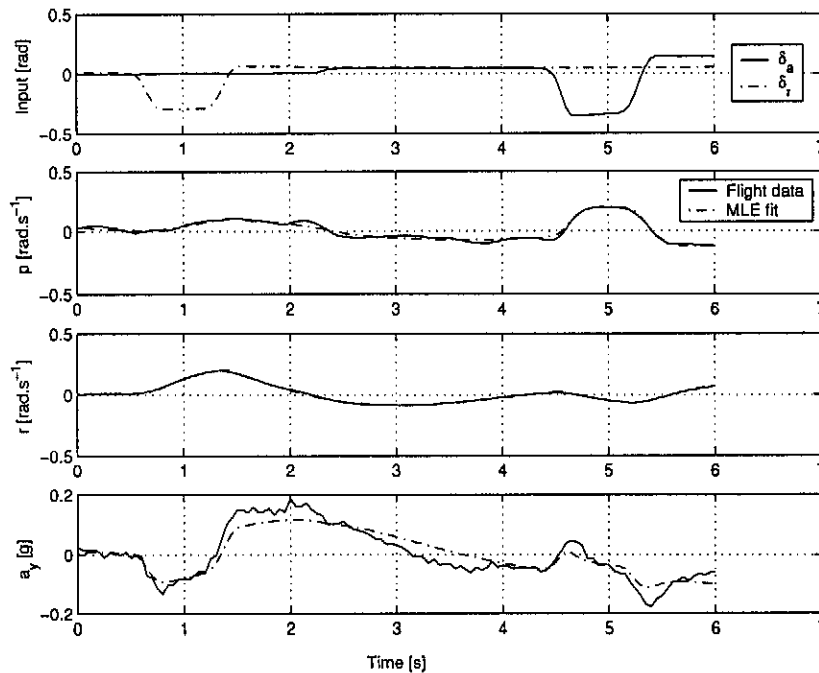


Figure 7.2: MLE fit of Flight data with pulse input.

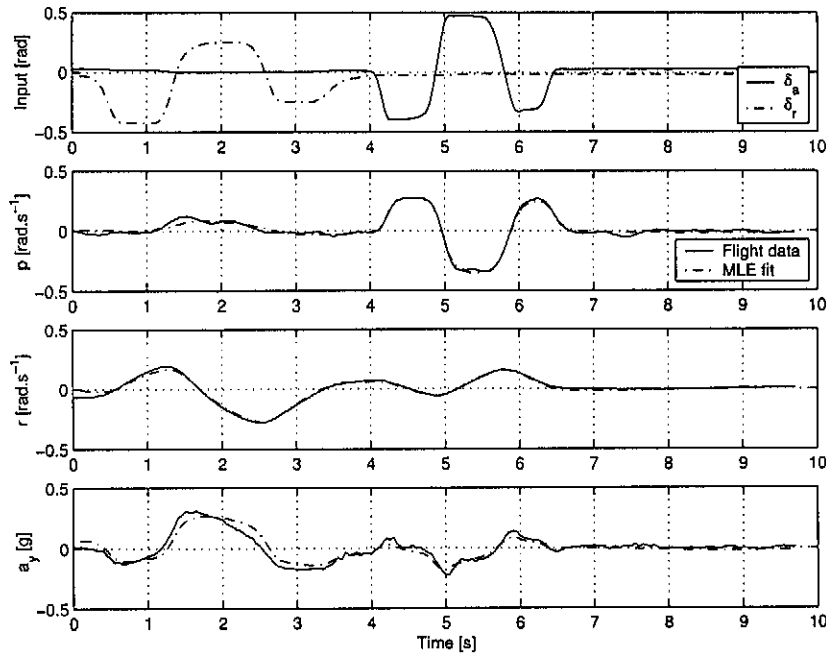


Figure 7.3: MLE fit of Flight data with doublet input.

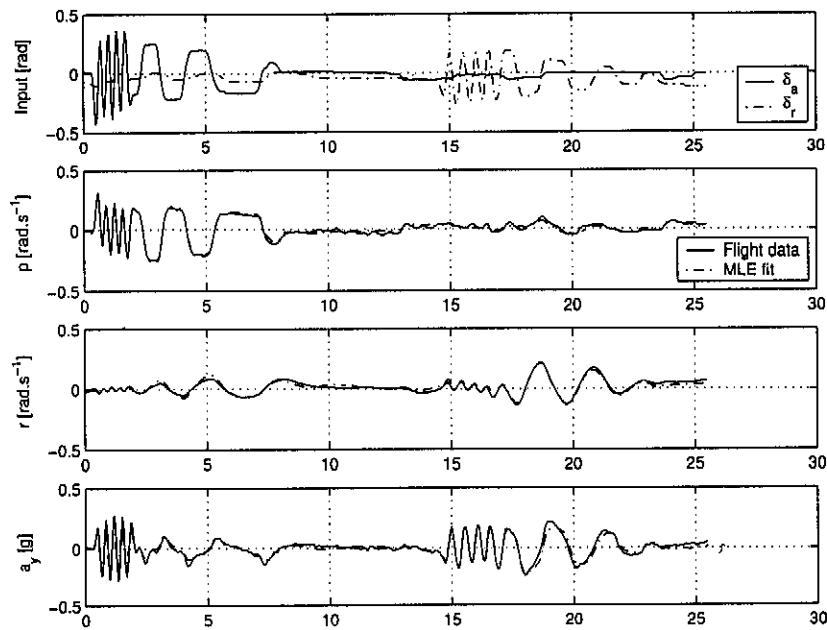


Figure 7.4: MLE fit of Flight data with four frequency input.

7.4.1 Initial Parameters and Convergence Issues

It was found that the estimated derivatives are sensitive to the starting parameters used in the estimation algorithm. Before the estimation algorithm is run it is given an initial parameter vector to start from. A set of initial parameter vectors would be a grouping of parameter vectors that occupy a particular region of the parameter space. The problem that occurs is that with a certain set of initial parameter vectors the algorithm will converge to a particular minimum that

is not the same as the minimum reached by a different set of initial parameter vectors. Thus it is impossible to say that the final parameters listed in these results represent the global minimum of the cost-function. The fact that the global minimum cannot be guaranteed calls into question the credibility of the estimated parameters.

The parameters listed later in this section are obtained using an initial parameter vector that results in final parameters that fit the measured data well and have the correct polarity for the control and stability derivatives.

A possible cause of the local minimums encountered could be a combination of the model complexity and over parameterization. Further study is required to aid the selection of an initial parameter vector set that will converge on the global minimum. The next step is to consult the scatter plots of the parameters.

7.4.2 Scatter Plots

The scatter plots show good cluster around a particular value (Figures 7.5 to 7.10). The scatter between the parameters is large for most of the parameters. The Cramer-Rao bounds tend to be high for all of the parameters. For good confidence in the parameter, the Cramer-Rao bound should be at least an order of magnitude smaller than the parameter and the scatter should be smaller. This is not the case for these estimations. The scatter plots for all the stability and control derivatives follow:

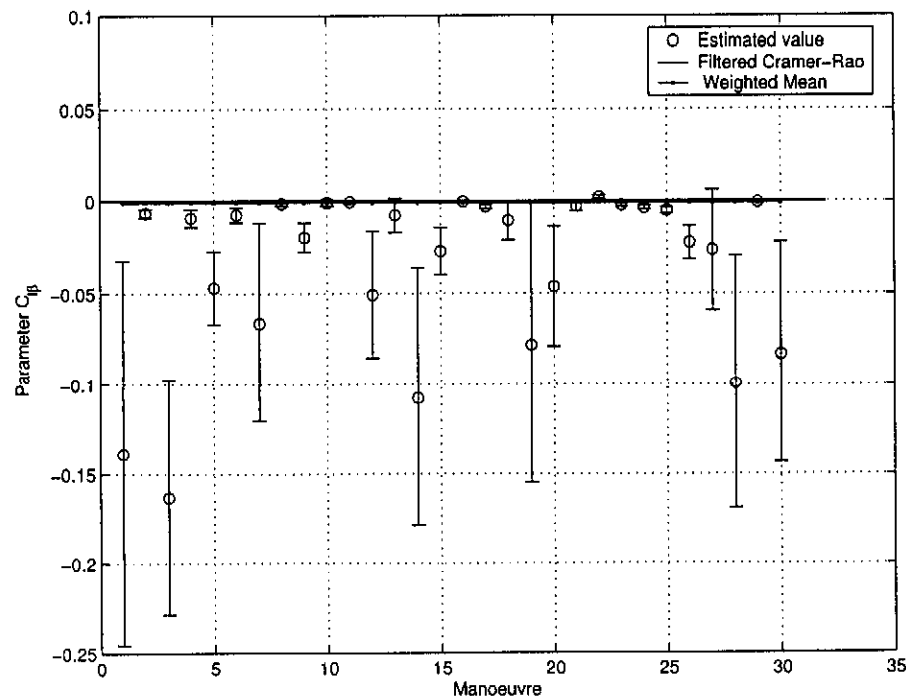


Figure 7.5: Scatter plot of $C_{l\beta}$ (Roll due to sideslip).

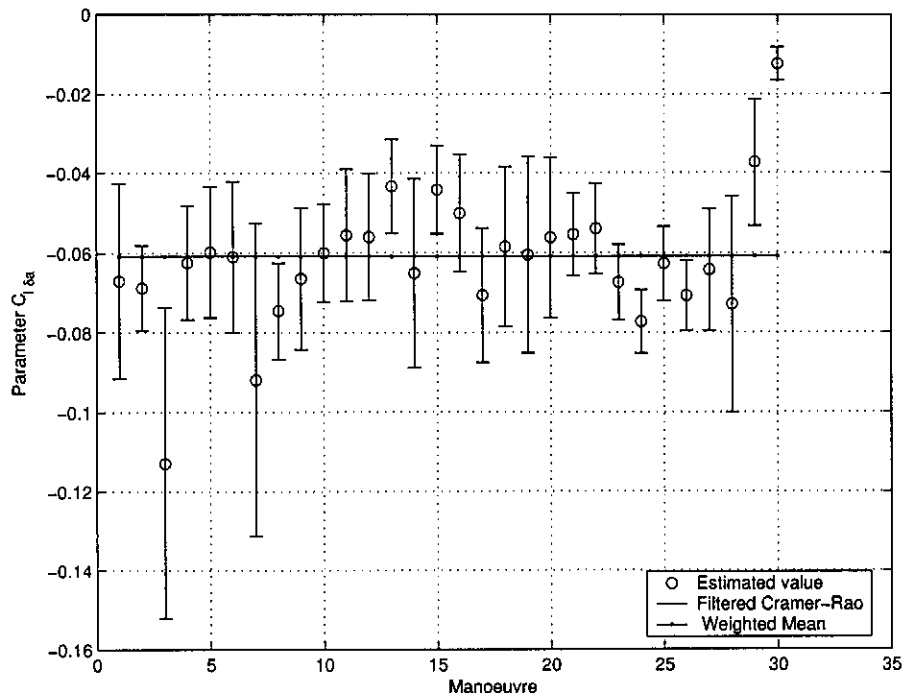


Figure 7.6: Scatter plot of $C_{l\delta a}$ (Roll due to aileron deflection).

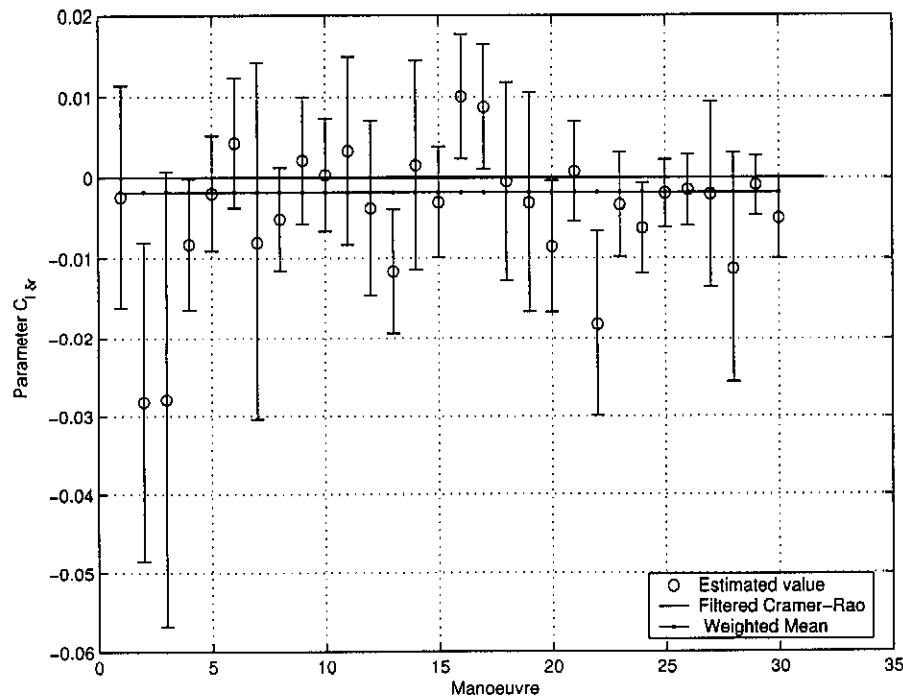


Figure 7.7: Scatter plot of $C_{l\delta r}$ (Roll due to rudder deflection).

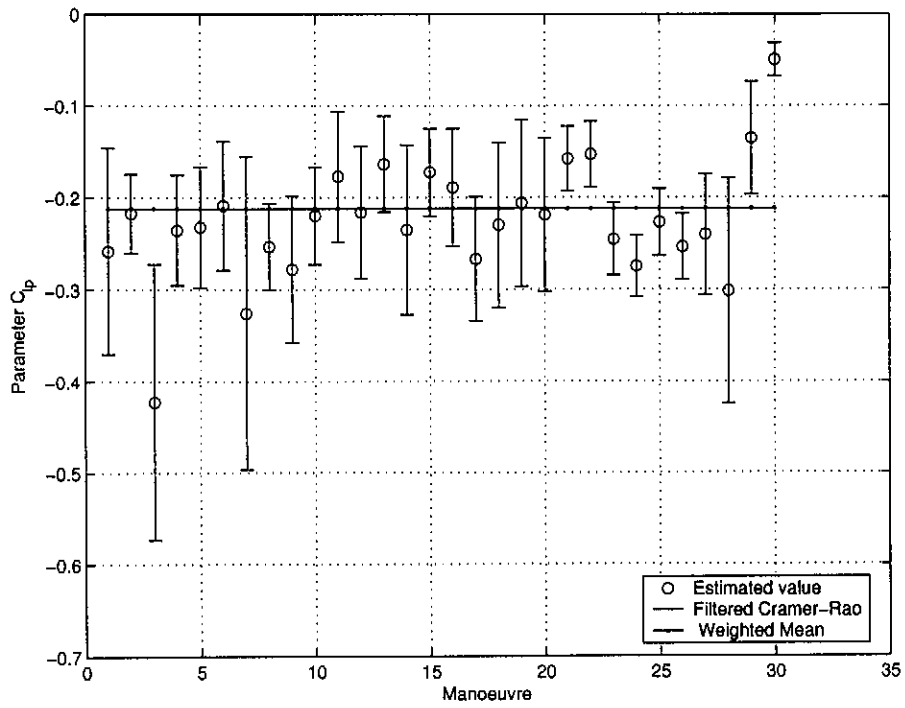


Figure 7.8: Scatter plot of C_{l_p} (Roll damping).

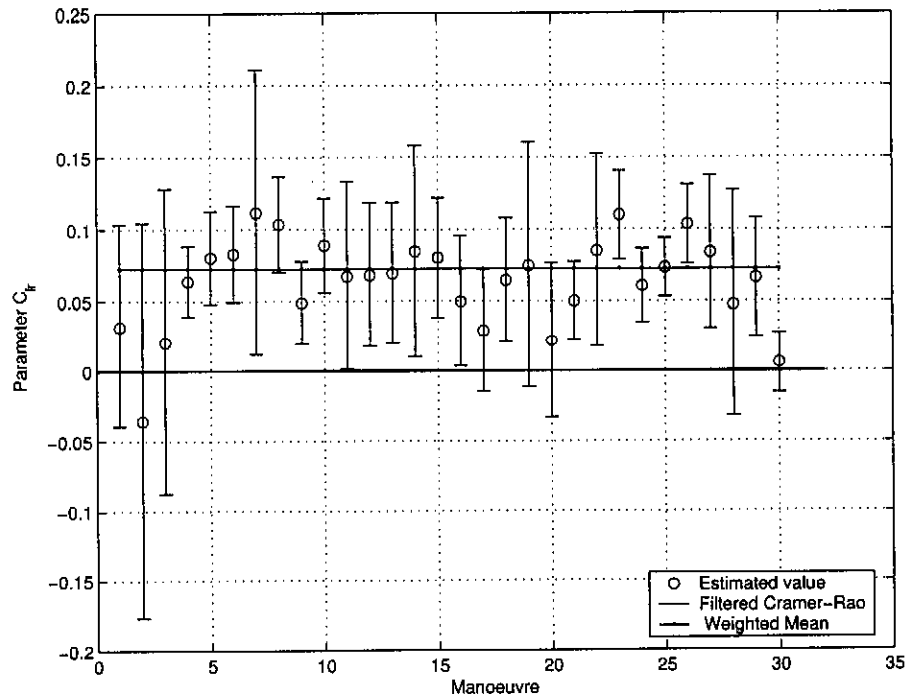


Figure 7.9: Scatter plot of C_{l_r} (Roll due to yaw rate).

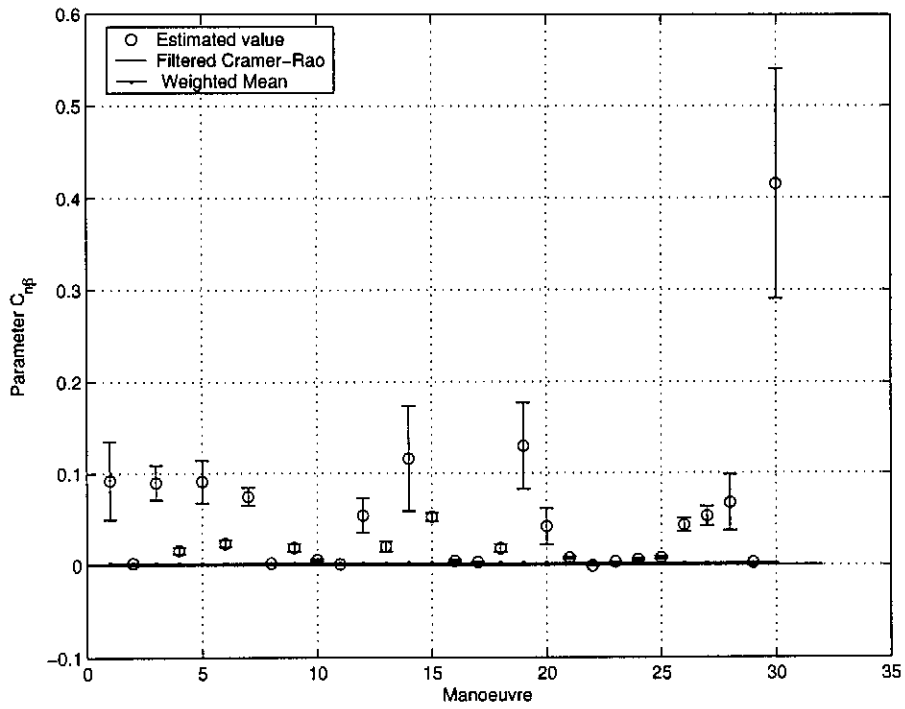


Figure 7.10: Scatter plot of $C_{n\beta}$ (Yaw due to sideslip angle).

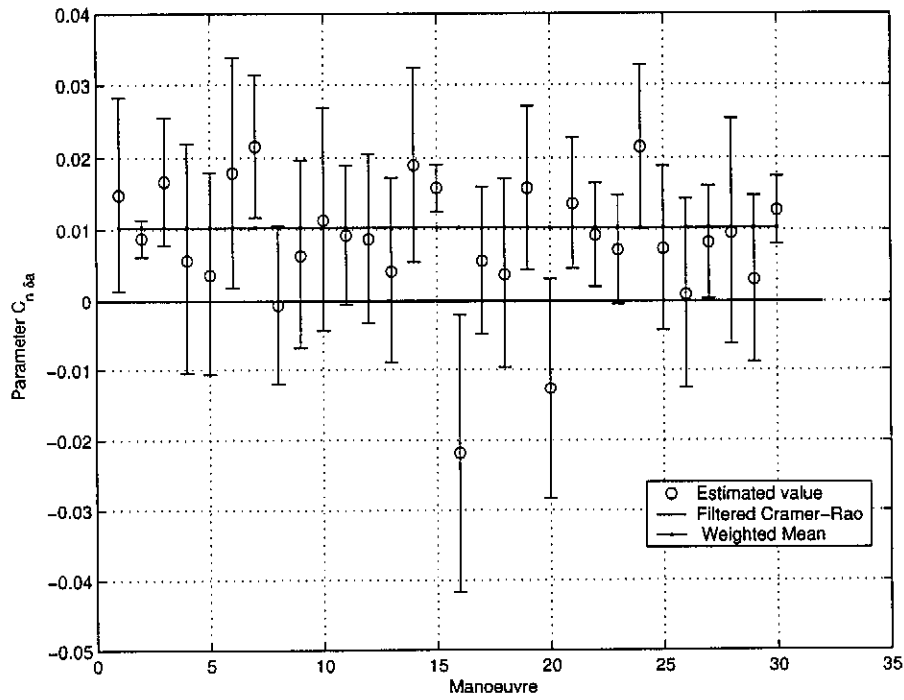


Figure 7.11: Scatter plot of $C_{n\delta a}$ (Yaw due to aileron deflection).

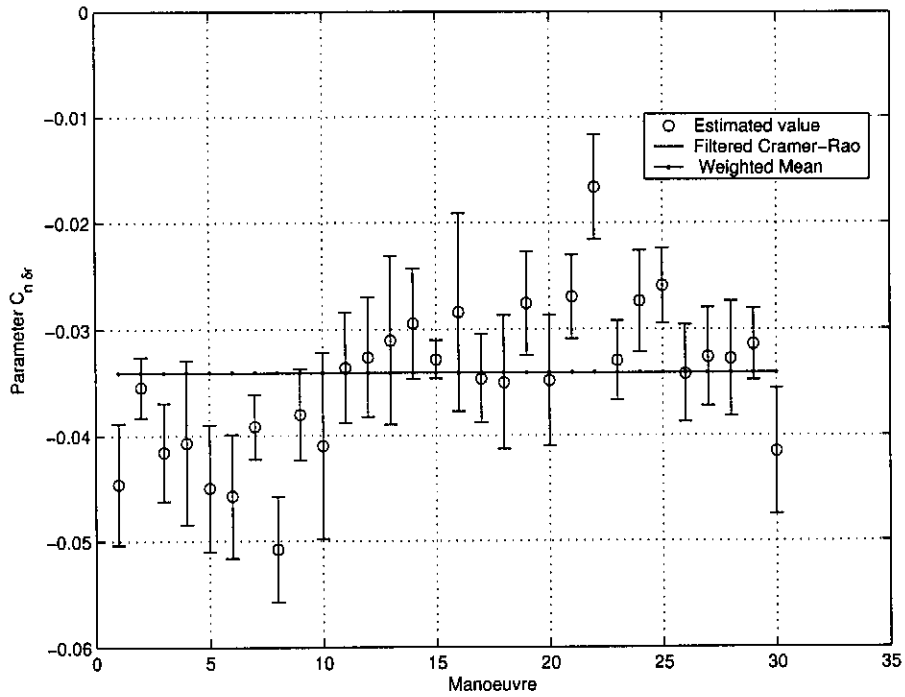


Figure 7.12: Scatter plot of $C_{n\delta r}$ (Yaw due to rudder deflection).

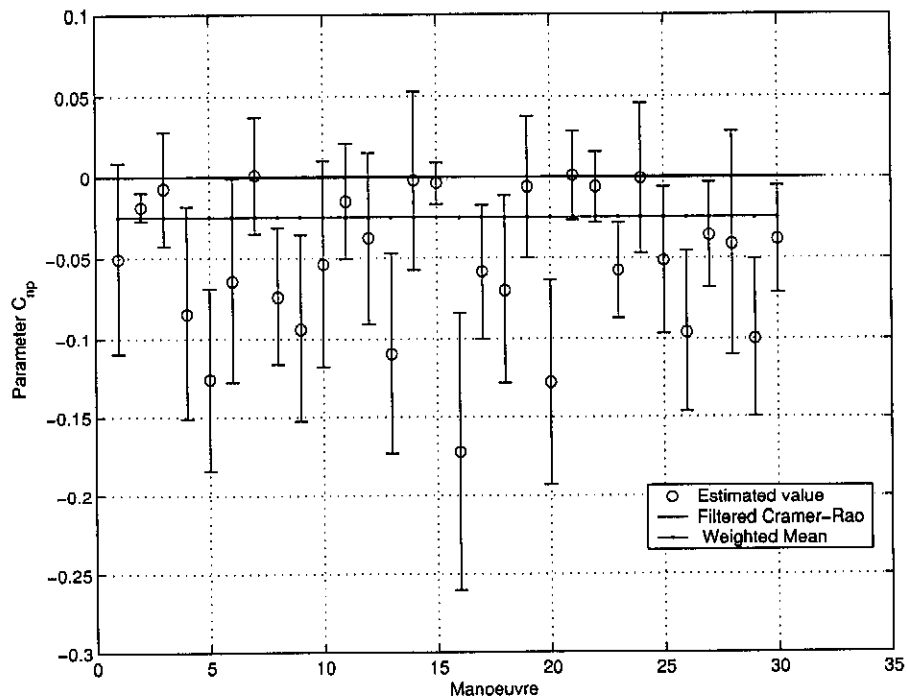


Figure 7.13: Scatter plot of C_{np} (Yaw due to roll rate).

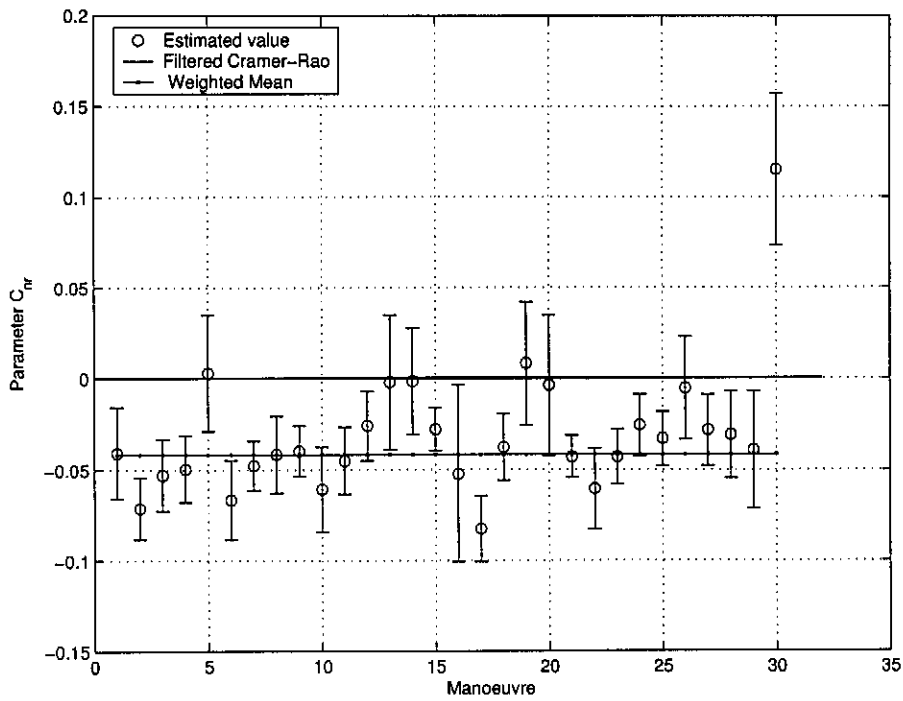


Figure 7.14: Scatter plot of C_{nr} (Yaw damping).

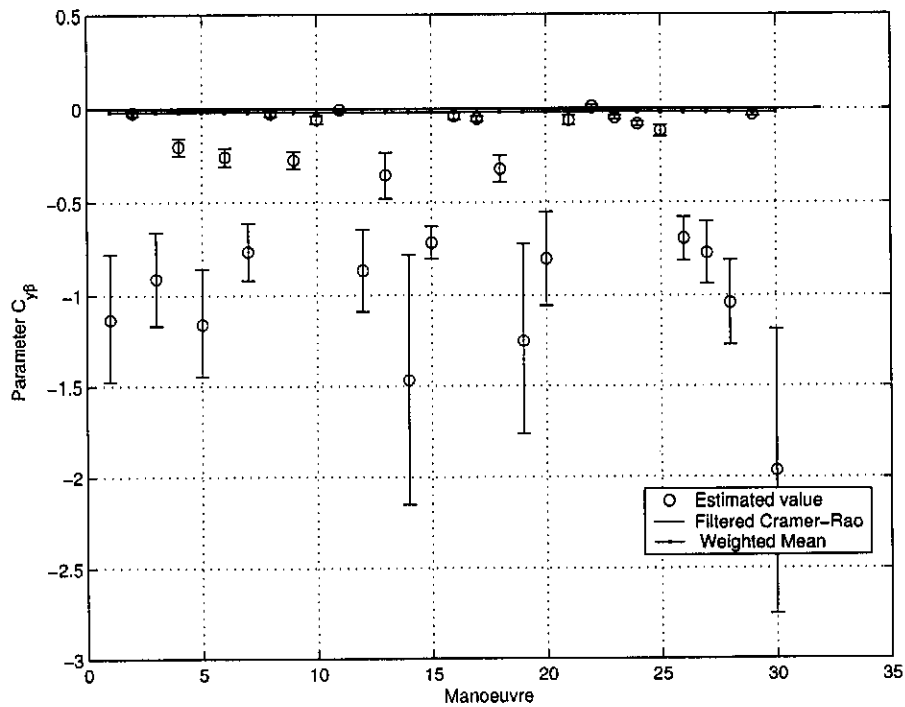


Figure 7.15: Scatter plot of $C_{Y\beta}$ (Side force due to sideslip angle).

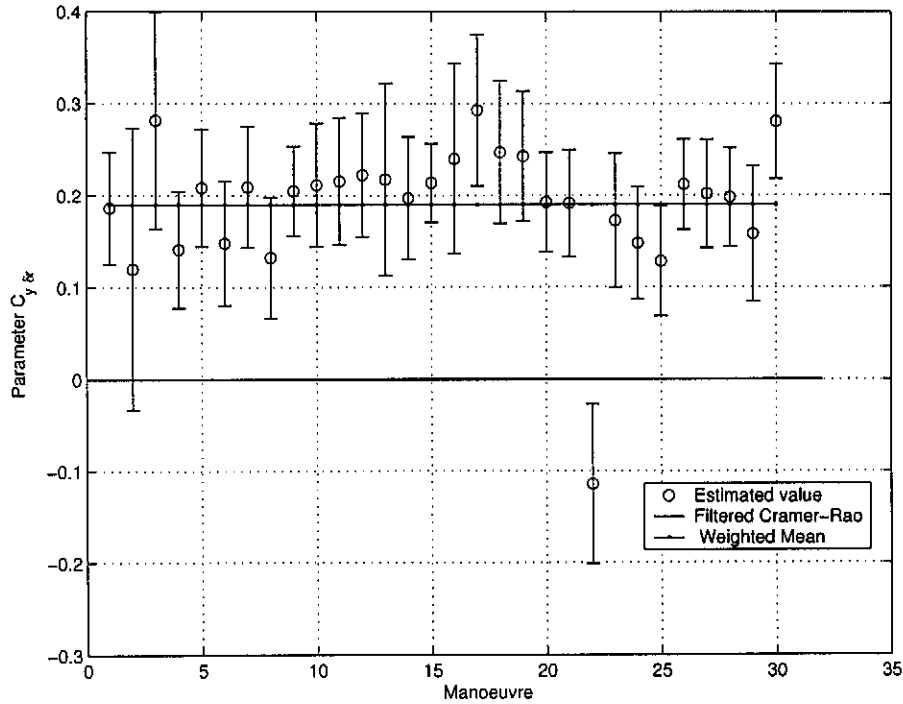


Figure 7.16: Scatter plot of $C_{Y_{\beta r}}$ (Side force due to rudder deflection).

Attention must be drawn to the scatter plots of the β derivatives, figures 7.5, 7.10, 7.16. The β derivatives have a large scatter, the Cramer-Rao bounds increase from zero outward resulting in a weighted mean close to zero, this is in contrast to the other parameters that exhibit large Cramer-Rao bounds and a weighted mean in the center of the scatter. The scatter and Cramer-Rao bounds of the β derivatives requires further investigation to understand the mechanisms causing the scatter and odd Cramer-Rao bounds of these estimated derivatives.

7.4.3 Combining the Parameters from Different Manoeuvres

To provide a single value for each of the parameters from all of the manoeuvres, all the different estimations of the same parameter are combined in a weighted mean. The inverse of the Cramer-Rao bound is used to weight the parameters with equation 7.26.

$$\bar{\theta} = \frac{\sum \frac{1}{\sigma_i^2} \theta_i}{\sum \frac{1}{(\sigma_i)^2}} \tag{7.26}$$

Also expected standard deviation for $\bar{\theta}$ is calculated from equation 7.27.

$$\frac{1}{\hat{\sigma}_{wls}^2} = \sum_{i=1}^N \frac{1}{\sigma_i^2} \tag{7.27}$$

The effect is that the larger the Cramer-Rao bound, and thus the potential error, the less its corresponding estimated parameter will affect the mean. Table 7.1 lists the $\bar{\theta}$ of the parameters and the $\hat{\sigma}_{wls}$.

Parameter	$\bar{\theta}$	$\hat{\sigma}_{wls}$
$C_{Y\beta}$	-0.0195	0.0025
$C_{Y\delta_r}$	0.1899	0.0125
$C_{l\beta}$	-0.0001	0.0002
C_{l_p}	-0.2120	0.0107
C_{l_r}	0.0728	0.0075
$C_{l\delta_a}$	-0.0608	0.0026
$C_{l\delta_r}$	-0.0018	0.0014
$C_{n\beta}$	0.00016	0.00013
C_{n_p}	-0.0247	0.0055
C_{n_r}	-0.0416	0.0036
$C_{n\delta_a}$	0.0103	0.0015
$C_{n\delta_r}$	-0.034	0.0008

Table 7.1: Weighted mean parameters and expected standard deviation

As a final check the weighted mean parameters are used in a simulation and compared with measured data.

7.4.4 Matching the Measured Data

The $\bar{\theta}$ parameters are taken and packed into the dynamic model used to estimate the parameters and simulated with control signals from the recorded flight data. The response that $\bar{\theta}$ produces is then compared with the response measured on the glider. Before simulating with the $\bar{\theta}$ parameters, another estimation run is done. The control and stability derivatives are fixed to the $\bar{\theta}$ parameters and new bias and initial condition values are estimated for the test manoeuvre.

A Four-frequency manoeuvre is used to provide the control and response data. This manoeuvre was the most complicated flown and is most likely to reveal the short-comings of the $\bar{\theta}$ parameters. Figure 7.17 shows that the $\bar{\theta}$ parameters are a reasonable representation of ZS-GHB's lateral dynamics. The $\bar{\theta}$ parameters produce small errors with the measured response. Examining figure 7.17 closer: The manoeuvre can be divided into two sections, the aileron stimulation and the rudder stimulation. During the aileron stimulation:

- Roll rate (p): The $\bar{\theta}$ parameters provide a good fit to the measured roll rate. There are still misfits that could be improved; the errors are slightly larger in the low frequency portion of the manoeuvres. These errors could be from non-linearities caused by the high bank angle.
- Yaw rate (r): The timing of the $\bar{\theta}$ parameter's response matches the measured data. The

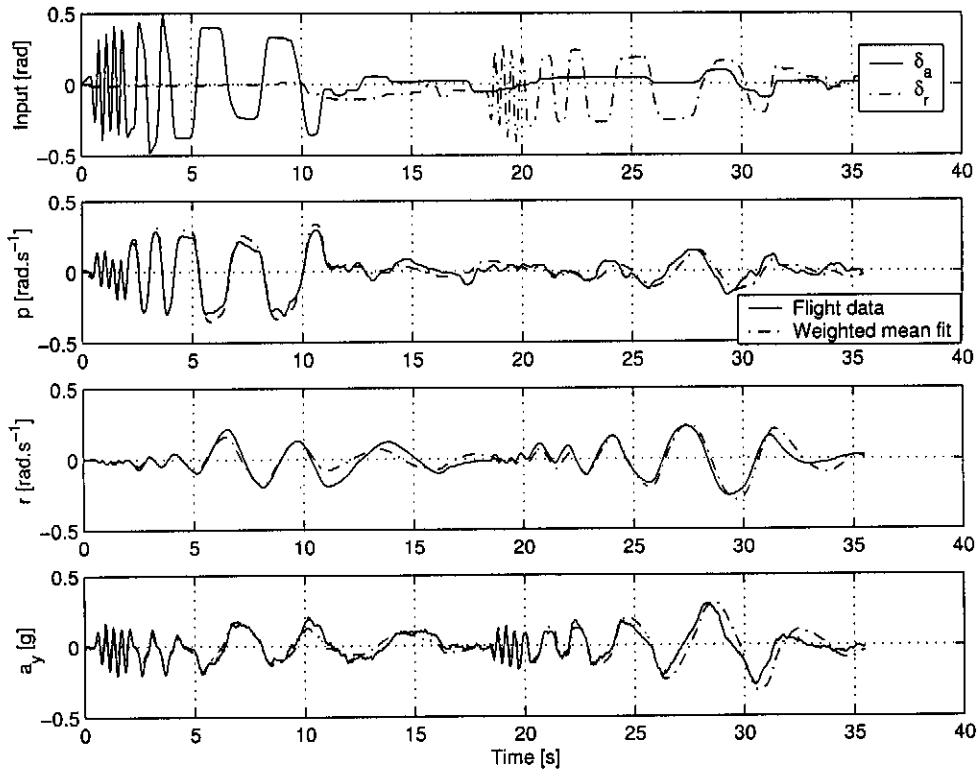


Figure 7.17: *Weighted mean model responses compared to measured data.*

amplitude does not always fit well. When the aileron is actuated the fit is initially good but the simulated responses' amplitude does not increase in the low frequency portion of the manoeuvre as the measured response does, revealing that parameters in the A matrix are not quite correct.

- Lateral acceleration (a_y): There is a reasonably good match through out the aileron manoeuvre, but there are still a few misfits. The force derivatives are known to be difficult to estimate.

During the rudder stimulation:

- Roll rate (p): The rudder induces very little roll. The algorithm has managed to fit this data reasonably well, but a few large errors do exist.
- Yaw rate (r): The fit from the estimated system fits the data well except for a slight timing error in the lowest frequency of the manoeuvre.
- Lateral acceleration (a_y): The fit during the rudder stimulation is reasonably good, but a large timing error develops toward the end of the manoeuvre, preventing a good fit.

Considering that all of the parameters have very large Cramer-Rao bounds and scatter, the fit of the simulation is surprisingly good.

Chapter 8

Longitudinal Derivative Estimation

The longitudinal motion involves the pitch rate (q) and the normal acceleration (a_n) and axial acceleration (a_x). To isolate this motion from the general flight the pilot only uses the elevator and holds the aileron and rudder still. The process used to obtain the longitudinal derivatives is the same as for the lateral derivatives. The manoeuvres used are described in section 7.1

8.1 Longitudinal Equations of Motion in State-Space Model

8.1.1 Longitudinal equations of motion

The glider will always fly at low angles of attack. This fact allows certain simplifications to the general equations of longitudinal motion to be made. The simplifications made by Maine and Illif in [11]

- The approximation $C_L = C_N$ can be used.
- The θ state equation can be omitted.

The state equations:

$$\dot{\alpha} = -\frac{\bar{q}S}{mV}(C_N + \dot{\alpha}_b) + q + \frac{g}{V}(\cos \phi \cos \theta \cos \alpha + \sin \theta \sin \alpha) \quad (8.1)$$

$$\dot{q}I_{yy} = \bar{q}ScC_m \quad (8.2)$$

The observation equations for the sensors that are installed in the glider:

$$q_z = q + q_b \quad (8.3)$$

$$a_n = \frac{\bar{q}S}{mg}C_N + \frac{x_a}{g}\dot{q} + \frac{z_a}{g}(q^2 + p^2) - \frac{y_a}{g}\dot{p} \quad (8.4)$$

$$a_x = -\frac{\bar{q}S}{mg}C_A + \frac{z_a}{g}\dot{q} - \frac{x_a}{g}(q^2 + r^2) - \frac{y_a}{g}\dot{r} \quad (8.5)$$

The expansion of the aerodynamic coefficients is:

$$C_N = C_{N_\alpha}\alpha + C_{N_\delta}\delta + C_{N_b} \quad (8.6)$$

$$C_A = C_{A_\alpha}\alpha + C_{A_\delta}\delta + C_{A_b} \quad (8.7)$$

$$C_m = C_{m_\alpha}\alpha + C_{m_\delta}\delta + C_{m_q}\frac{qc}{2V} + C_{A_b} \quad (8.8)$$

8.1.2 State-space form and additional inputs

The state-space form is completed as with the lateral derivative estimation. Extra input terms are added to take care of the biases and a gravity linearization inputs.

$$\begin{bmatrix} \dot{\alpha} \\ \dot{q} \end{bmatrix} = \begin{bmatrix} -\frac{\bar{q}S}{mV}C_{N_\alpha} & 1 \\ -\frac{\bar{q}Sc}{I_{yy}}C_{m_\alpha} & -\frac{\bar{q}Sc^2}{I_{yy}2V}C_{m_q} \end{bmatrix} \begin{bmatrix} \alpha \\ q \end{bmatrix} + \begin{bmatrix} -\frac{\bar{q}S}{mV}C_{N_\delta} & 1 & -\frac{\bar{q}S}{mV}(C_{N_b} + \dot{\alpha}_b) & 0 & 0 & 0 & 0 & 0 \\ -\frac{\bar{q}Sc}{I_{yy}}C_{m_\delta} & 0 & -\frac{\bar{q}Sc^2}{I_{yy}2V}C_{m_b} & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \delta \\ u2 \\ u3 \\ u4 \\ u5 \\ u6 \\ u7 \\ u8 \end{bmatrix} \quad (8.9)$$

Where:

$$u2 = \frac{g}{V}(\cos \phi \cos \theta \cos \alpha + \sin \theta \sin \alpha) \quad (8.10)$$

$$u3 = 1 \quad (8.11)$$

$$u4 = q^2 \quad (8.12)$$

$$u5 = p^2 \quad (8.13)$$

$$u6 = r^2 \quad (8.14)$$

$$u7 = \dot{p} \quad (8.15)$$

$$u8 = \dot{r} \quad (8.16)$$

The gravity input $u2$ is calculated using ϕ , θ and α calculated before the estimation algorithm is run, from the angular rate measurements. Drift on the angular rate sensors and incorrect initial conditions introduce errors into the term. To attempt to improve the quality of this input the algorithm was run and the initial values estimated were used to calculate the gravity term

and the algorithm was run again with the new $u2$ ad initial value, this did not improve the results. The initial value of α increased with each successive estimation and would not converge to a specific value. To complete the estimation process the value of α for calculating the gravity term was fixed at $0.1rad$.

8.1.3 Output Equations

The output equations have extra terms added to correct for the accelerometers not being mounted at the center of gravity.

$$\begin{bmatrix} q \\ a_n \\ a_x \end{bmatrix} = C \begin{bmatrix} \alpha \\ q \end{bmatrix} + D \begin{bmatrix} \delta \\ u2 \\ u3 \\ u4 \\ u5 \\ u6 \\ u7 \\ u8 \end{bmatrix} \quad (8.17)$$

$$C = \begin{bmatrix} 0 & 1 \\ \frac{\bar{q}S}{mg}C_{N\alpha} - \frac{x_a}{g} \frac{\bar{q}Sc}{I_{yy}}C_{m\alpha} & -\frac{x_a}{g} \frac{\bar{q}Sc^2}{I_{yy}2V}C_{m\alpha} \\ -\frac{\bar{q}S}{mg}C_{A\alpha} - \frac{z_a}{g} \frac{\bar{q}Sc}{I_{yy}}C_{m\alpha} & -\frac{z_a}{g} \frac{\bar{q}Sc^2}{I_{yy}2V}C_{m\alpha} \end{bmatrix} \quad (8.18)$$

$$D = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -\frac{\bar{q}S}{mg}C_{N\delta} - \frac{x_a}{g} \frac{\bar{q}Sc}{I_{yy}}C_{m\delta} & 0 & -\frac{\bar{q}S}{mg}C_{N_b} - \frac{x_a}{g} \frac{\bar{q}Sc}{I_{yy}}C_{m_b} & \frac{z_a}{g} & \frac{z_a}{g} & 0 & \frac{y_a}{g} & 0 \\ -\frac{\bar{q}Sc}{mg}C_{A\delta} - \frac{z_a}{g} \frac{\bar{q}Sc}{I_{yy}}C_{m\delta} & 0 & -\frac{\bar{q}S}{mg}C_{A_b} - \frac{z_a}{g} \frac{\bar{q}Sc}{I_{yy}}C_{m_b} & -\frac{x_a}{g} & 0 & \frac{x_a}{g} & 0 & -\frac{y_a}{g} \end{bmatrix} \quad (8.19)$$

8.2 Estimation of Parameters

The estimation of the longitudinal derivative follows the same process as the lateral derivative estimation. During the pre-processing the longitudinal and lateral data is kept in the same data file. The first stage of the pre-processing is done to all the data and stored. Before the algorithm is started in the Matlab script, the filtered data is scaled to engineering units. To get the initial values for the parameters the exact same method as with the lateral derivative estimation was applied. The Cessna Citaion II longitudinal data was taken as the first initial parameters, the algorithm produced a good convergence with the first attempt.

8.2.1 Parameter estimation structure

The following parameters are estimated for the longitudinal motion in the different phases of the algorithm:

- *pidq*: The 2 initial conditions, $C_{N_{\delta e}}, C_{A_{\delta e}}, C_{m_{\delta e}}, C_{m_q}, C_{N_b}, C_{A_b}, C_{m_b}, C_{q_b}$.
- *pidm*: $C_{N_\alpha}, C_{A_\alpha}, C_{m_\alpha}, C_{N_{\delta e}}, C_{A_{\delta e}}, C_{m_{\delta e}}, C_{m_q}, C_{N_b}, C_{A_b}, C_{m_b}, C_{q_b}, \dot{\alpha}_b$.
- *pidf*: $C_{N_\alpha}, C_{A_\alpha}, C_{m_\alpha}, C_{N_{\delta e}}, C_{A_{\delta e}}, C_{m_{\delta e}}, C_{m_q}$.

The longitudinal estimation would not converge if all of the parameters were estimated in the final stage, as they should be. After removing the biases and initial conditions from the final phase, convergence was possible, but the GDOP's were unacceptably high (in excess of 10). By estimating C_{m_q} in the quadratic phase as well as the other phases the GDOP's and Cramer-Rao bounds dropped significantly. For the estimation structure mentioned above, the GDOPs for most of the parameters in most of the manoeuvres are below 2.

8.3 Results

The estimated parameters produce good curve fits when compared to the flight data. Figure 8.1 to 8.3 below show the curve fitting for each of the types of manoeuvres used. In appendix B are the plots of each manoeuvre and plots of the error between the measured and simulated responses.

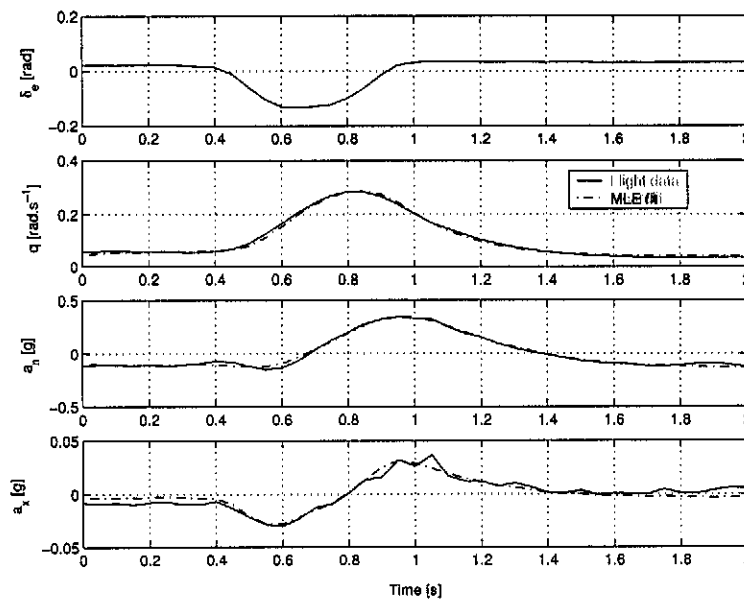


Figure 8.1: MLE fit of Flight data with pulse input.

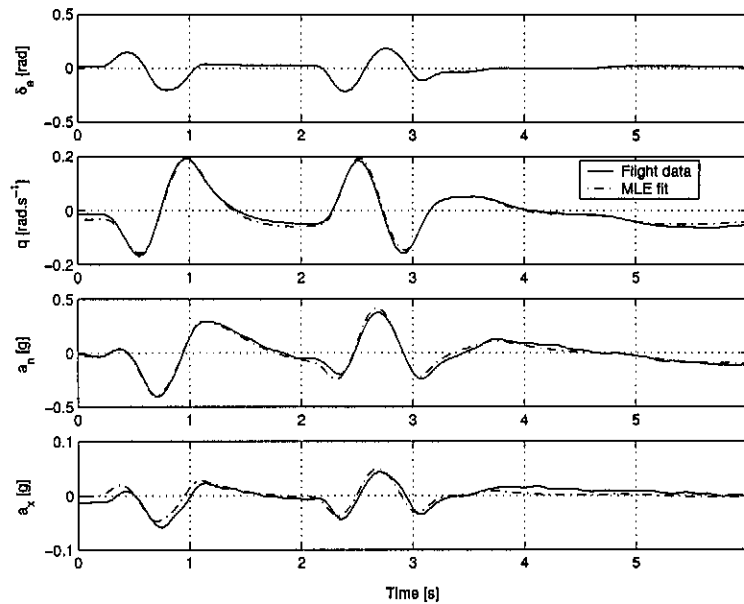


Figure 8.2: MLE fit of Flight data with doublet input.

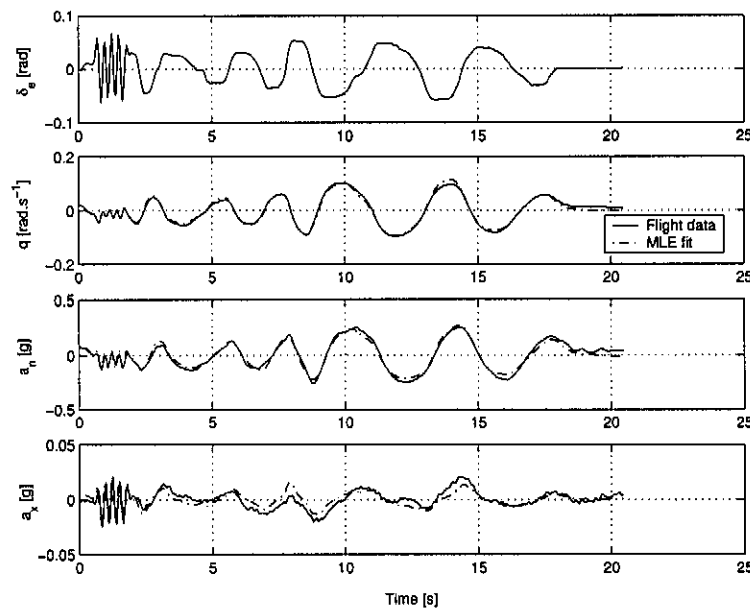


Figure 8.3: MLE fit of Flight data with four frequency input.

8.3.1 Scatter Plots

The scatter plots do not look as promising as the response fits. The scatter on most of the plots is high with Cramer-Rao bounds having inconsistencies with the scatter. Some parameters with large scatter have small Cramer-Rao bounds and parameters with a low scatter have large Cramer-Rao bounds. What should occur is parameters with large scatter should have a large Cramer-Rao bound and parameters with a low scatter should have low Cramer-Rao bounds. The inconsistency between the scatter and the Cramer-Rao bounds warrants further investigation.

The parameters estimated from the data of the second flight (manoeuvre 6 to 10) show marked differences to the other parameters estimated. During this particular flight the wind was quite strong with a large amount of turbulence adding unmodeled process noise to the glider's responses. The effect is that the estimator does not produce the same parameters as with flight in stiller conditions of the other manoeuvres. When calculating the weighted mean the parameters estimated from the second flight are not included. The following figures are the scatter plots from the estimation.

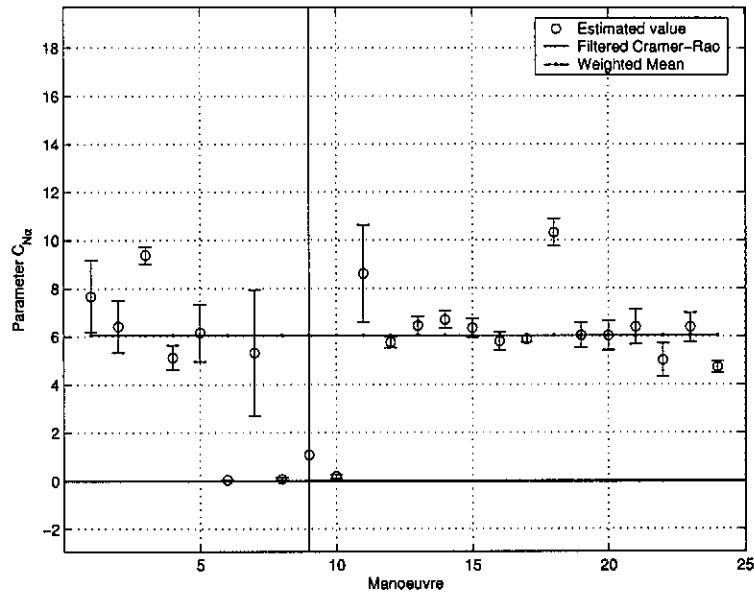


Figure 8.4: Scatter plot of C_{N_α} (Normal force due to angle of attack).

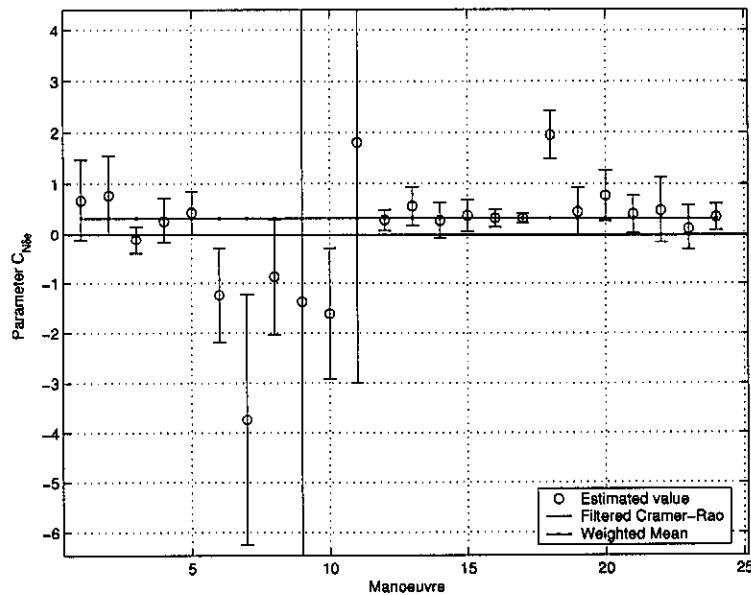


Figure 8.5: Scatter plot of $C_{N_{\delta_e}}$ (Normal force due to elevator deflection).

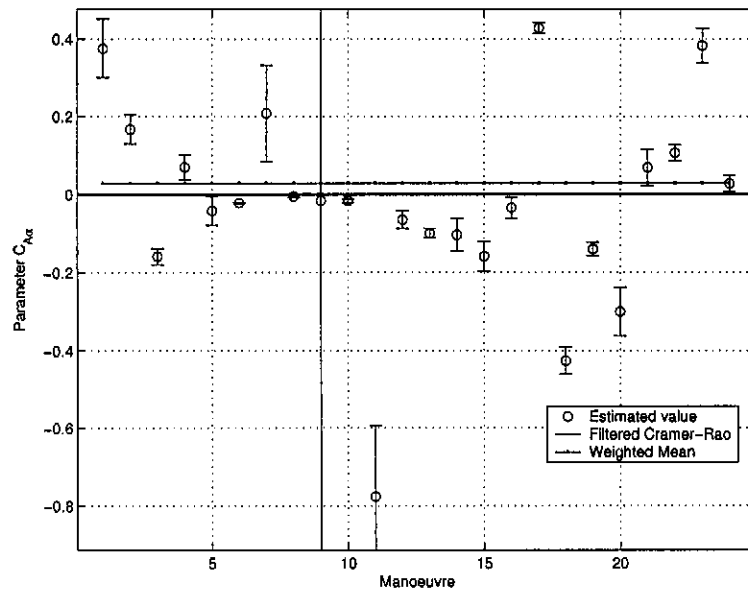


Figure 8.6: Scatter plot of $C_{A\alpha}$ (Axial force due to angle of attack).

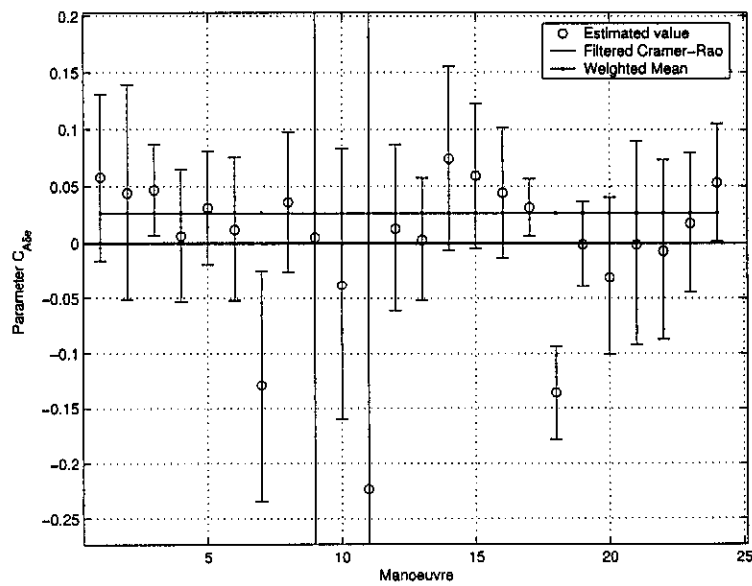


Figure 8.7: Scatter plot of $C_{A\delta e}$ (Normal force due to elevator deflection).

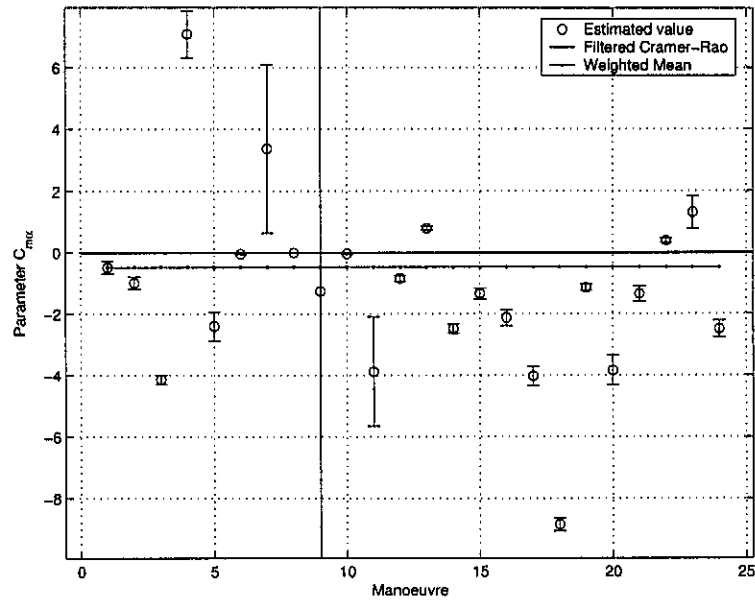


Figure 8.8: Scatter plot of C_{m_α} (Pitch rate due to angle of attack).

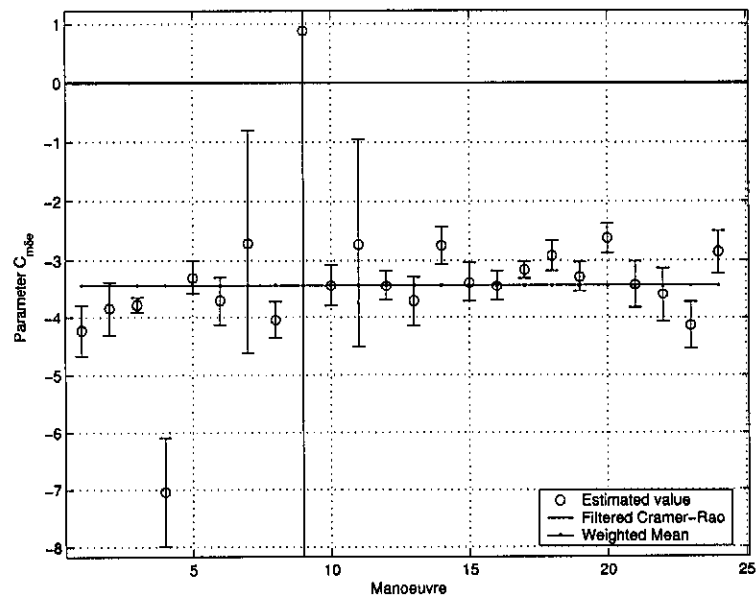


Figure 8.9: Scatter plot of $C_{m_{\delta_e}}$ (Pitch rate due to elevator deflection).

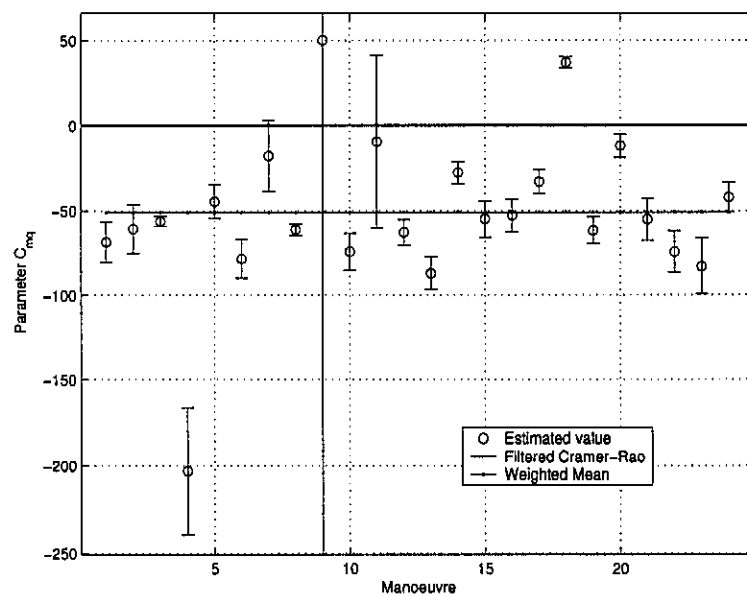


Figure 8.10: Scatter plot of C_{mq} (Pitch rate damping).

8.3.2 Matching the Measured Data

To be able to use the estimated parameters the $\bar{\theta}$ parameters must be a reasonable representation of the longitudinal dynamics and fit the response to any manoeuvre. New biases and initial conditions are first estimated for the $\bar{\theta}$ parameters before the simulation is run. Comparing the responses from the simulation with the flight data, it is found that the high frequency portion of the manoeuvre fits well. As the frequency of the control input drops, the fit becomes worse. The timing of the signals remains correct but the magnitude of the simulation increases to prevent a good fit.

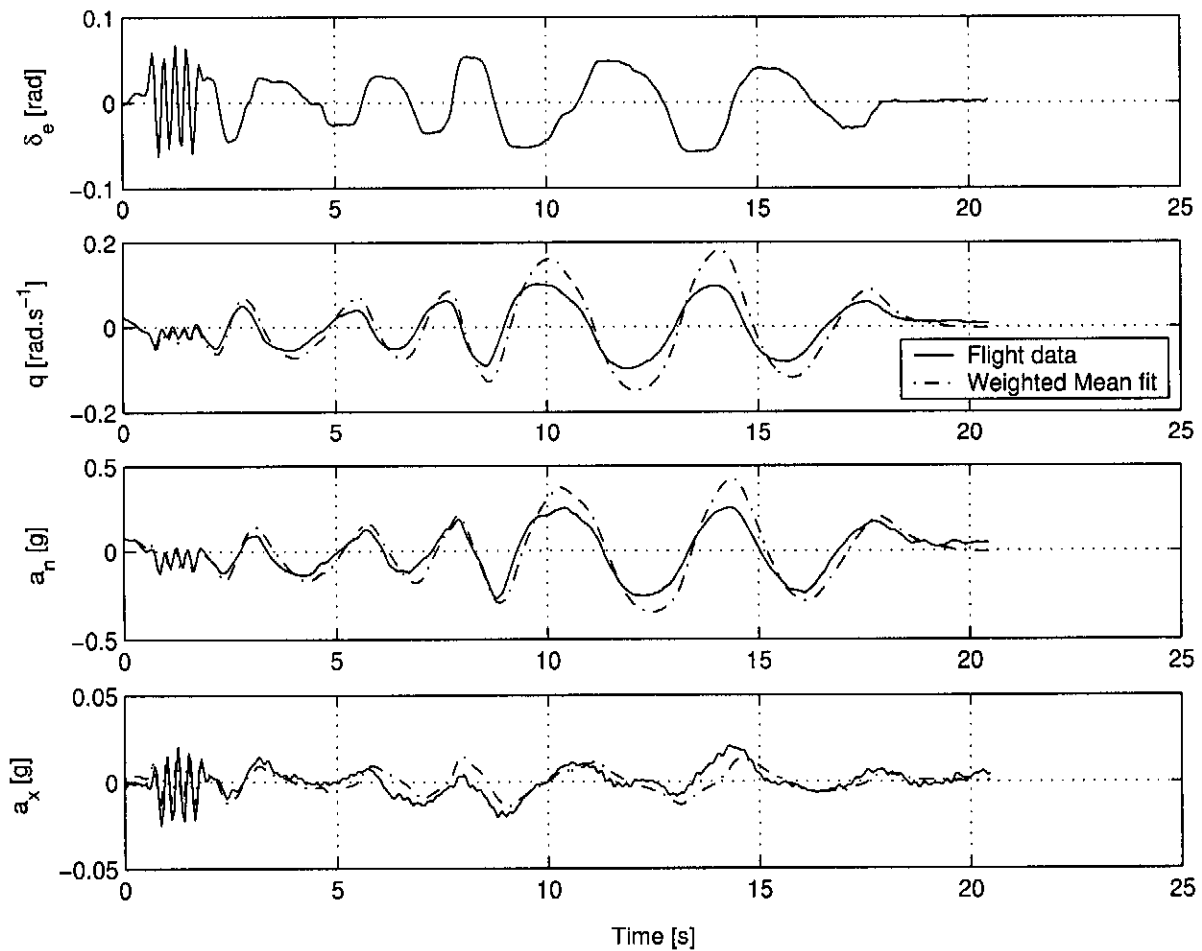


Figure 8.11: Simulation with weighted mean parameters compared to flight data

The steadily increasing error between the simulation and measured data raises the suspicion that parameters in the A matrix are not the correct values. To check, the simulation is run again but only using the weighted mean parameters for the B,C and D matrices and the manoeuvre specific estimated A matrix parameters. The result was an improved fit of the pitch rate and normal acceleration but not the Axial acceleration. Further investigation revealed that the parameters C_{N_α} , C_{m_α} and C_{A_α} could be the cause of the misfit of the data.

To obtain parameters that would fit most of the manoeuvres well, the α parameters (C_{N_α} ,

$C_{m\alpha}$, $C_{A\alpha}$) were fixed at their $\bar{\theta}$ values. The estimation algorithm was run again for all the manoeuvres. The new $\bar{\theta}$ parameters along with re-estimated biases and initial conditions for the particular manoeuvre are used in a simulation and the response of the simulation is compared with the measured data. The new $\bar{\theta}$ parameters are capable of producing a reasonable fit with the measured response.

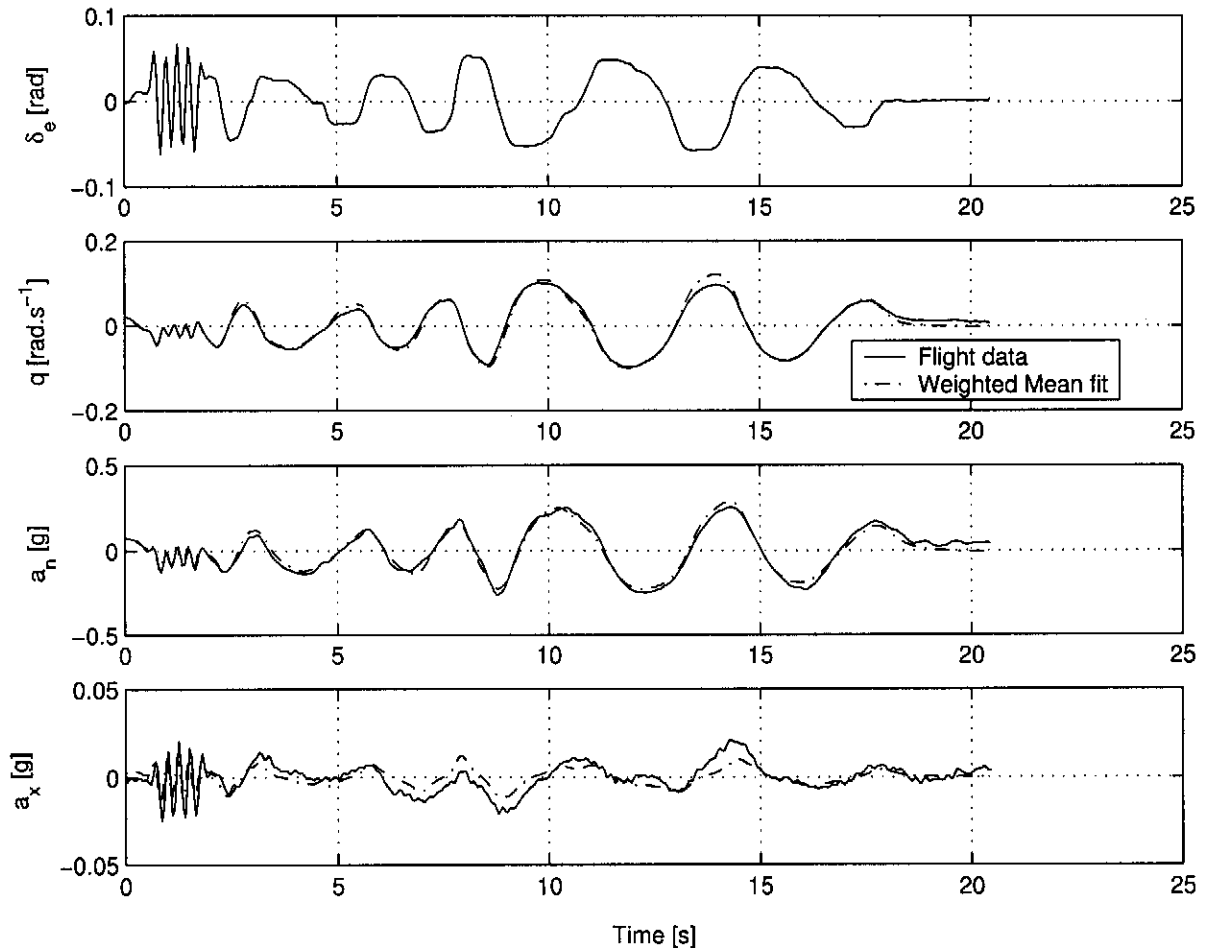


Figure 8.12: Simulation with new weighted mean parameters compared to flight data

The problem of choosing the parameters and fixing the values is that the other parameters become biased. Even though the simulation fits the measured aircraft responses with the new parameters it cannot be said that the parameters are an actual representation of the physical aircraft. However, it is now a model of the dynamics of the aircraft. To improve on this result would mean being able to accurately estimate the α parameters. To do this α needs to be measured.

The parameters that result from the longitudinal estimation are listed in table 8.1. The original weighted means along with their expected $\hat{\sigma}_{wls}$ values are listed alongside the fixed α parameters and the new $\bar{\theta}$ and resulting expected $\hat{\sigma}_{wls}$.

Parameter	Original		Fixed	
	$\bar{\theta}$	$\hat{\sigma}_{wls}$	$\bar{\theta}$	$\hat{\sigma}_{wls}$
C_{N_α}	6.0482	0.0857	6.0482	N/A
C_{A_α}	0.0288	0.0057	0.0288	N/A
C_{m_α}	-0.4655	0.0335	-0.4655	N/A
$C_{N_{\delta_e}}$	0.3317	0.0621	0.2815	0.0690
$C_{A_{\delta_e}}$	0.0268	0.0127	0.0425	0.0156
$C_{m_{\delta_e}}$	-3.4222	0.0632	-3.5712	0.0673
C_{m_q}	-50.9960	1.8663	-59.353	0.9683

Table 8.1: *Estimated Longitudinal Control and Stability Derivatives*

Chapter 9

Conclusions and Recommendations

9.1 Overview

During the course of this project the following was achieved:

- Sensors were installed and calibrated in the airframe of ZS-GHB, to measure the control surface deflections.
- Pressure transducers were installed in ZS-GHB to measure dynamic and static pressure.
- An inertial measurement unit was developed and installed in ZS-GHB to measure the translation accelerations and angular rates.

With equipment installed into ZS-GHB, it was possible to do experiments to investigate the handling characteristics of ZS-GHB. The moments of inertia and vertical center of gravity were determined from ground experiments. Data, from a set of flight tests was used to extract a first set of control and stability derivatives. The parameters identified can produce responses that match the measured responses reasonably well.

9.2 Credibility of Estimated Parameters

Reviewing the results certain unexpected phenomenon are noticed:

- The structure to estimate parameters is not ideal. All the parameters should be estimated in the final phase of the algorithm.
- The choice of initial parameters can affect the minimum in which the estimation algorithm converges.
- The estimated parameter values exhibit a high scatter between manoeuvres.

- Cramer-Rao bounds are inconsistent. Certain parameters (eg. C_{m_α}) with large scatter have very small bounds. Other parameters with less scatter have large bounds (eg. $C_{l_{\delta a}}$). A third group of parameters has very small bounds on some manoeuvres and large bounds for other manoeuvres (eg. C_{Y_β}).

Considering the above points, the estimated parameters cannot be considered credible, and casts doubt as to how useful the parameters can be in representing the dynamics of the glider. The $\bar{\theta}$ of the parameters from Sections 7.4 and 8.3 are able to produce a reasonably good proof of fit. The proof of fit allows the use of the estimated parameters for the simulation of responses, but $\bar{\theta}$ does not represent the physical aerodynamic properties of the glider.

9.3 Recommendations

The control and stability derivatives estimated in this project are not accurate. To improve on the results more effort must be put into the estimation process. It is important to focus this effort in the correct areas. The MMLE3 algorithm is considered state-of-the-art and has produced many accurate parameter estimations with the state-space model implemented here.

The currently installed sensors are accurate enough to produce suitable data. To improve the data more sensors will have to be installed on ZS-GHB. In a similar project [16], where good estimates were obtained, the Euler angles during the manoeuvre were available. Also Maine and Illif [11] recommend that θ and ϕ are measured. The Euler angles have an important contribution in the calculation of the gravity term used for linearization and the α and β states. If θ is measured a θ state can be added to the longitudinal motion state-space model, which will benefit the estimation of the longitudinal derivatives. The extra data can be produced by the roll, pitch and yaw rates but requires a Kalman filter to ensure accurate measurements of attitude. The flow angles should also be measured. The observation of the flow angles will improve the α and β derivative estimates, which will indirectly help with the estimation of the other parameters [11]. It is common accepted practice to install an air-data boom onto the aircraft during flight-tests. The air-data boom, once calibrated accurately measures the flow angles, the dynamic and static pressure and air temperature. The addition of the sensors mentioned here would significantly increase the accuracy of the estimates.

On a theoretical level there are certain phenomenon that need to be investigated:

- The inability of the algorithm to converge when estimating all the parameters in the final phase.
- The high scatter of the same parameter between different manoeuvres.
- The anomalies that occur in the Cramer-Rao bounds between different manoeuvres.

9.4 Conclusion

At the conclusion of the work of this masters degree thesis it is appropriate to stand back and consider what has been achieved. At the start of this project ZS-GHB was a bare airframe and was undergoing a major overhaul. Sensors were added to measure the response of the glider and the control surface deflections. Software to manage the data produced by the sensors was developed. Physical properties of the glider, the vertical center of gravity, the inertia properties and the short-term control and stability derivatives were estimated.

The derivatives combined from many manoeuvres have yielded a model that can reproduce responses that match the data measured during flights of ZS-GHB. The derivatives can consequently be used in a flight simulator and produce realistic responses to the resolution discernible by normal pilots.

The precision of the estimates is doubtful because of the failure of the Cramer-Rao bounds as an accuracy indicator and large scatter in the estimates. These issues will have to be addressed in further work, which has been made possible by the achievements in this project.

Bibliography

- [1] BRYSON, A., *Control of Spacecraft and Aircraft*. Princeton University Press, 1993.
- [2] E.DUKE, ANTONIEWICZ, F., and KRAMBEER, K., "Derivation an definition of a linear aircraft model." tech. rep., August 1988. NASA reference publication 1207.
- [3] HIBBELER, R., *Engineering Mechanics Dynamics*. Prentice Hall, 1995.
- [4] ILLIF, K. and MAINE, R., "More Than You May Want to Know About Maximum Likelihood Estimation." 1984. AIAA Paper 84-2070.
- [5] ILLIF, K. and WANG, C., "X-29A Lateral-Directional Stability and Control Derivatives Extracted From High-Angle-of-Attack Flight Data." tech. rep., December 1996. NASA Technical Paper 3664.
- [6] ILLIF, K. and WANG, C., "Flight-Determined Subsonic Longitudinal Stability and Control Derivatives of the F-18 High angle of Attack Research Vehcile (HARV) With thrust vectoring." tech. rep., December 1997. NASA Technical Paper 97-206539.
- [7] LUCKE, R., "The Glider Data Gatherer." November 2001. University of Stellenbosch Electronic Engineering Scription Report.
- [8] MAINE, R. and ILLIF, K., "User's Manual for MMLE3, a General Fortran Program for Maximum Likelihood Parmeter Estimation." November 1980. NASA tech paper 1563.
- [9] MAINE, R. and ILLIF, K., "Identification of Dynamic Systems Theory and Formulation." 1985. NASA Reference Publication 1138.
- [10] Maine ,R.E. and Illif ,K.W., "The Theory and Practice of the Estimating the Accuracy of Dynamic Flight-Determined Coefficients." July 1981. NASA Reference Publication 1077.
- [11] Maine ,R.E. and Illif ,K.W., "Application of Parameter Estimation to Aircraft Stability and Control The Output-Error Approach." June 1986. NASA Refence Publication.
- [12] MCMORDIE, D., "Towards Pronking with a Hexapod." July 2002. Thesis,McGill University,Montreal,Canada.

- [13] MILNE, G., "Private discussion."
- [14] MILNE, G., "MMLE3 Identification Toolbox for State-Space System Identification using Matlab." 2000. User's Manual for Control Models Identification Routines.
- [15] MILNE, G., "Interpereting Error Sources in Output Error Parameter Estimation with Coloured Noise." August 2001. AIAA Paper 2001-4199.
- [16] MILNE, G., JULIANA, S., and HERMANSYAH, "Initial Analysis of the Short-Term Control and Stability and Control Derivatives of the Cessna Citation II using the Maximum Likelihood Method." December 2002.
- [17] NORTON, J., *An Inroduction to Identification*. Academic Press, 1986.
- [18] R.MAINE and MURRAY, J., "Application of parameter estimation to highly unstable aircraft." tech. rep., August 1986. NASA Technical Memorandum 88266.
- [19] SHAFER, M., "Flight Investigation of Various Control Inputs intended for Parameter Estimation." tech. rep., August 1984. NASA Technical Memorandum 85901.
- [20] SIM, A., "Flight Characteristics of a modified Schweizer SGS1-36 sailplane at low and very high angles of attack." tech. rep., August 1990. NASA Technical Paper 3022.
- [21] TANNER, R. and MONTGOMERY, T., "Stability and control derivative estimates obtained from flight data for the Beech 99." tech. rep., April 1979. NASA Technical Memorandum 72863.

Appendix A

Lateral Manoeuvres

The manoeuvres are grouped in flights. There were three different flights that collected data while manoeuvres were flown.

A.1 Flight 1

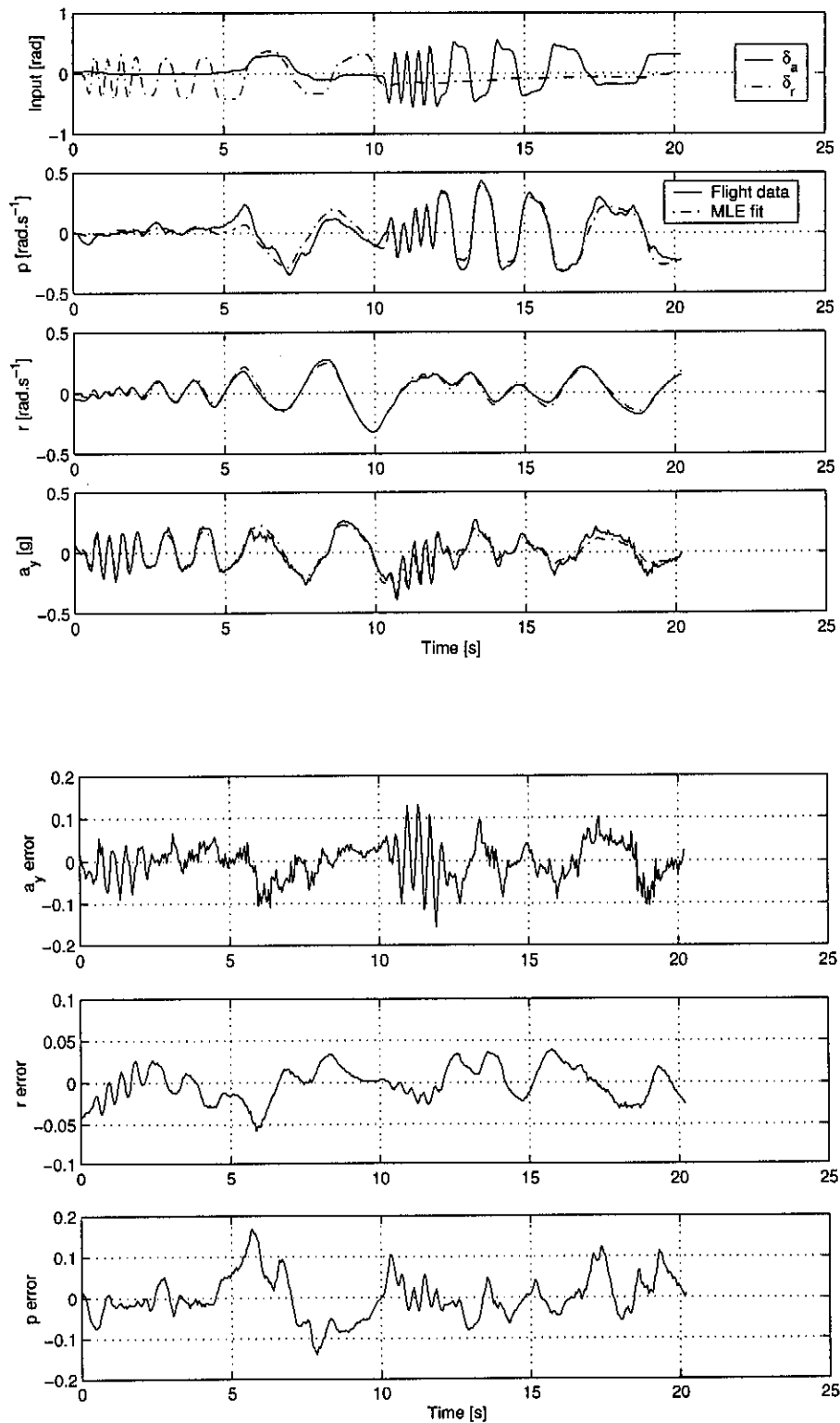
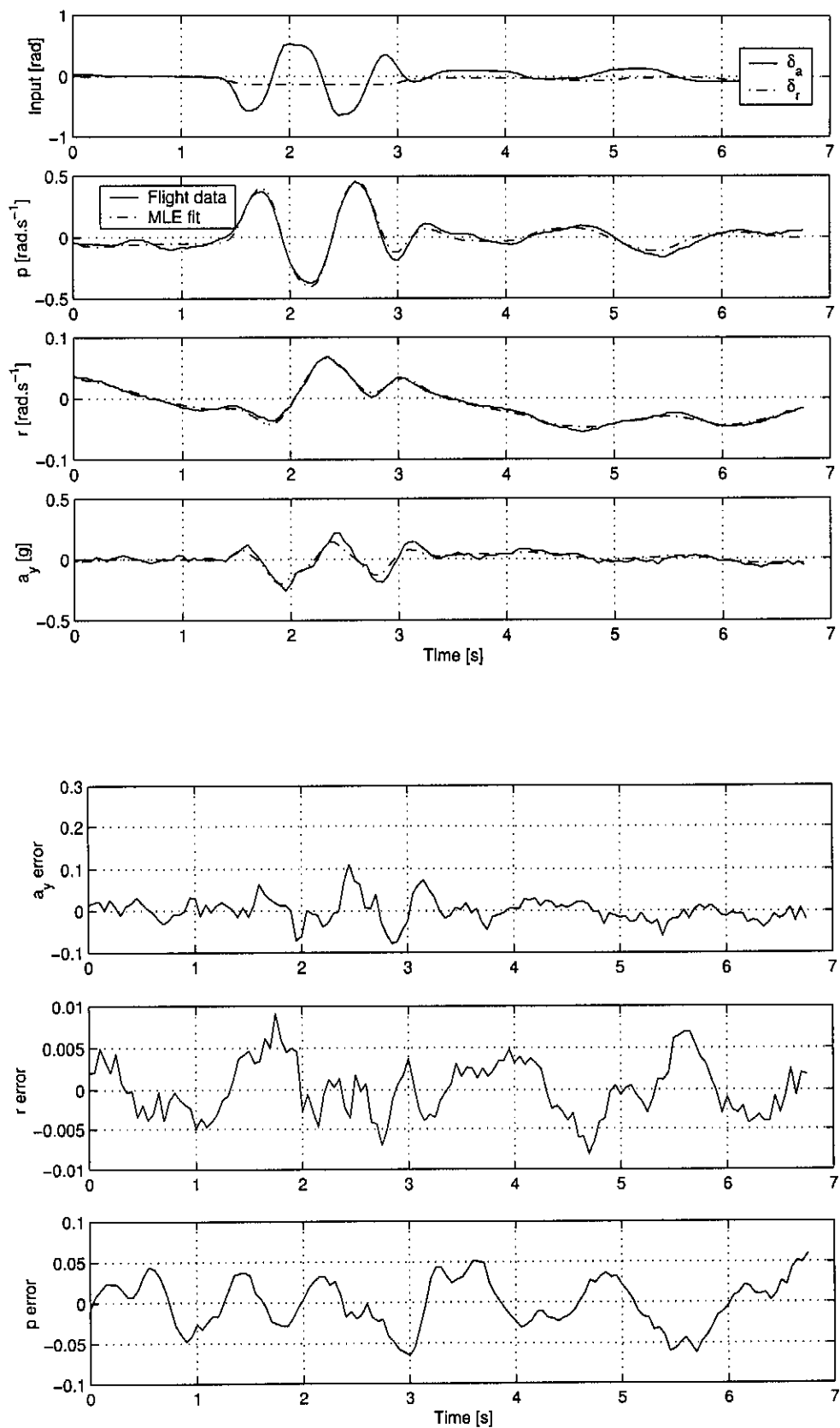


Figure A.1: Manoeuvre no.1

**Figure A.2: Manoeuvre no.2**

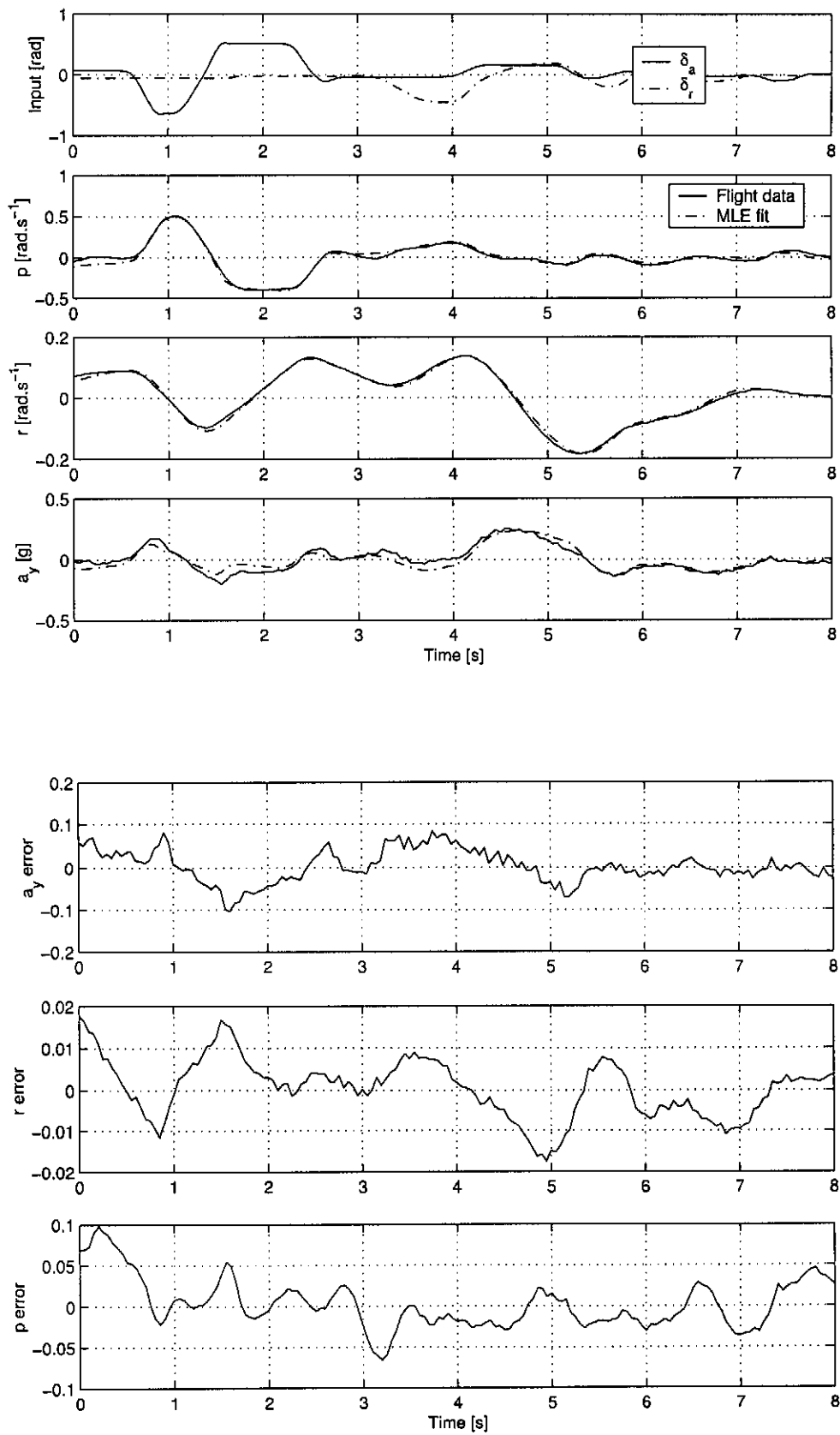


Figure A.3: *Manoeuvre no.3*

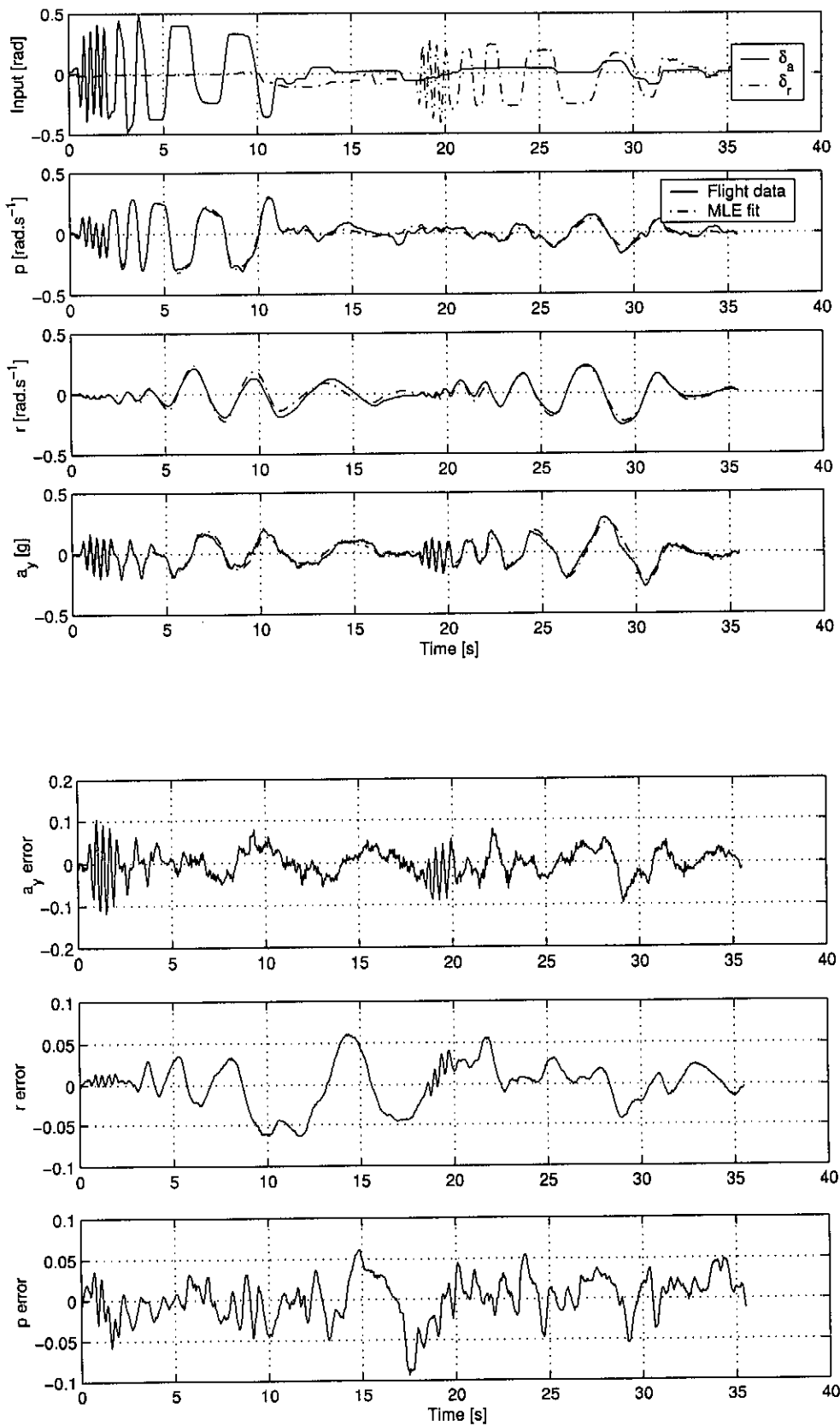


Figure A.4: Manoeuvre no.4

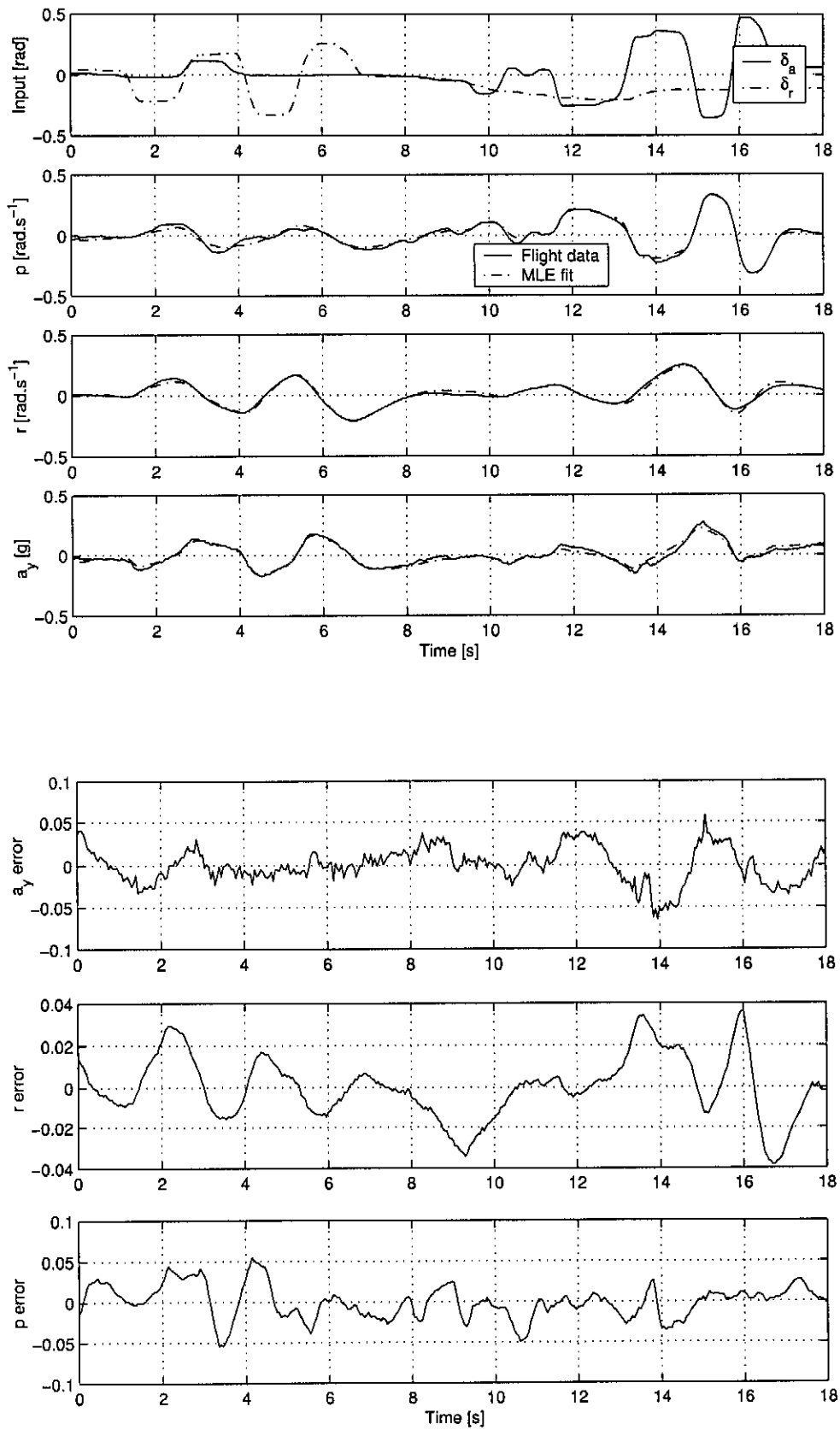


Figure A.5: Manoeuvre no.5

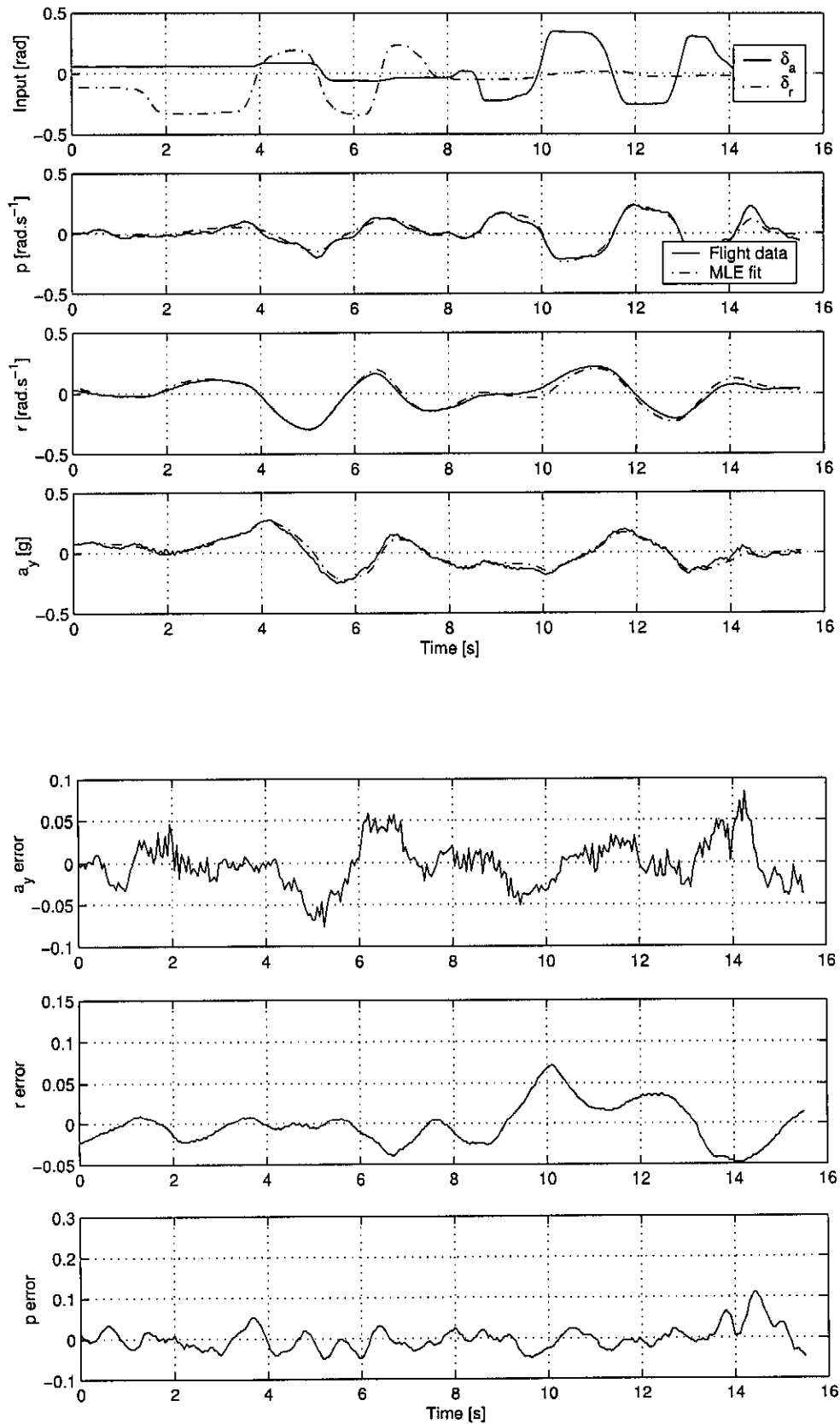


Figure A.6: Manoeuvre no.6

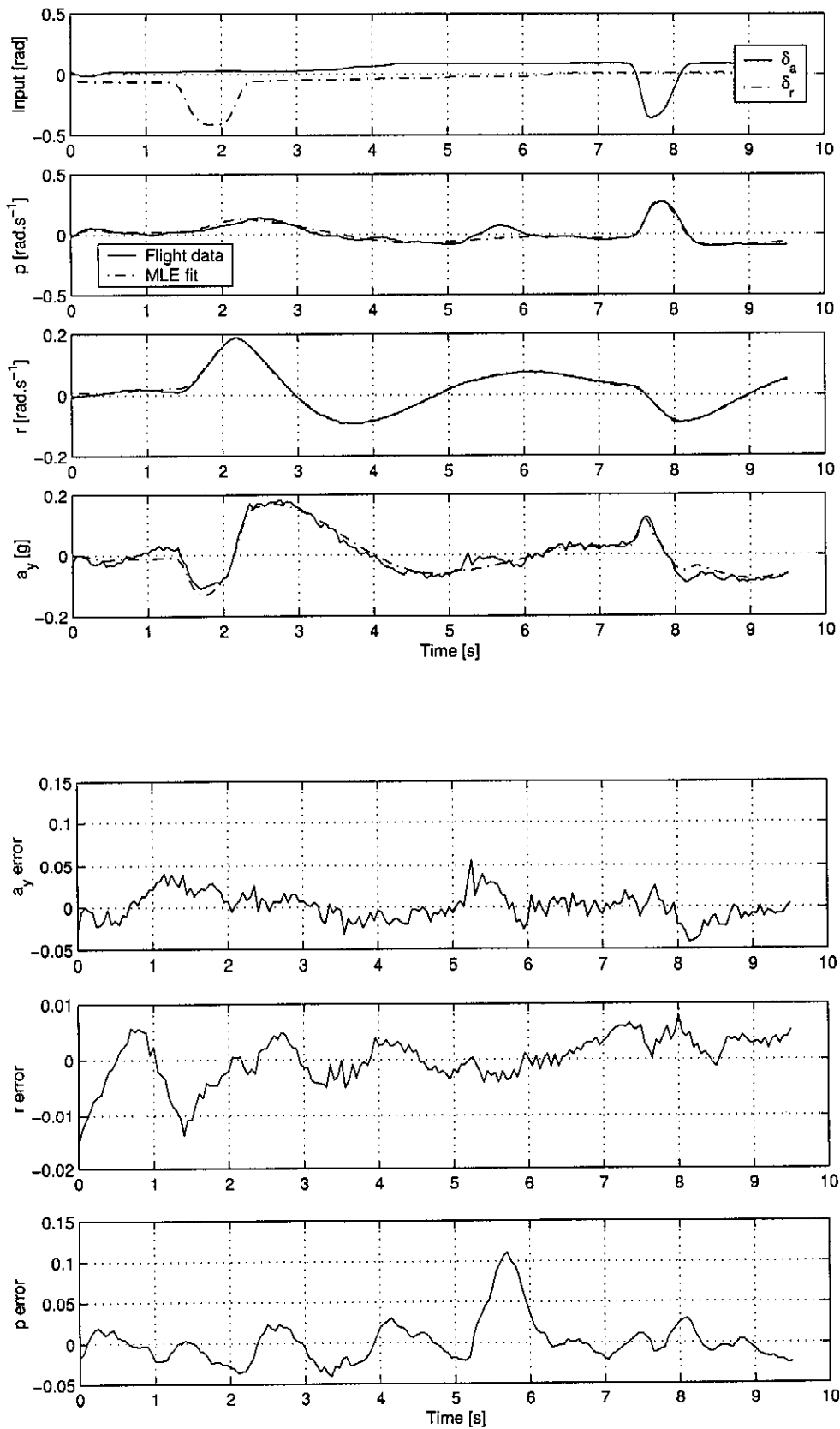


Figure A.7: Manoeuvre no.7

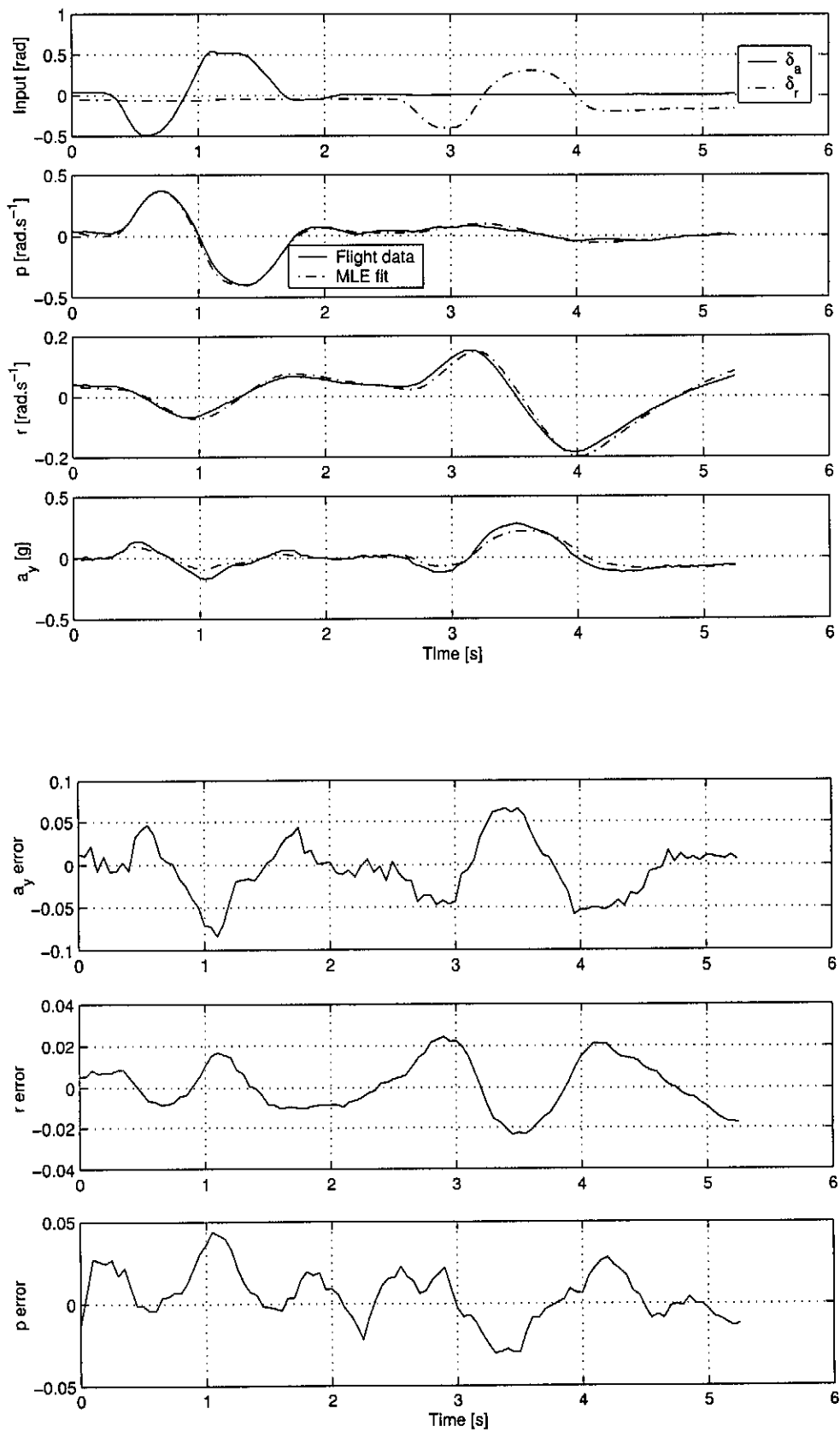
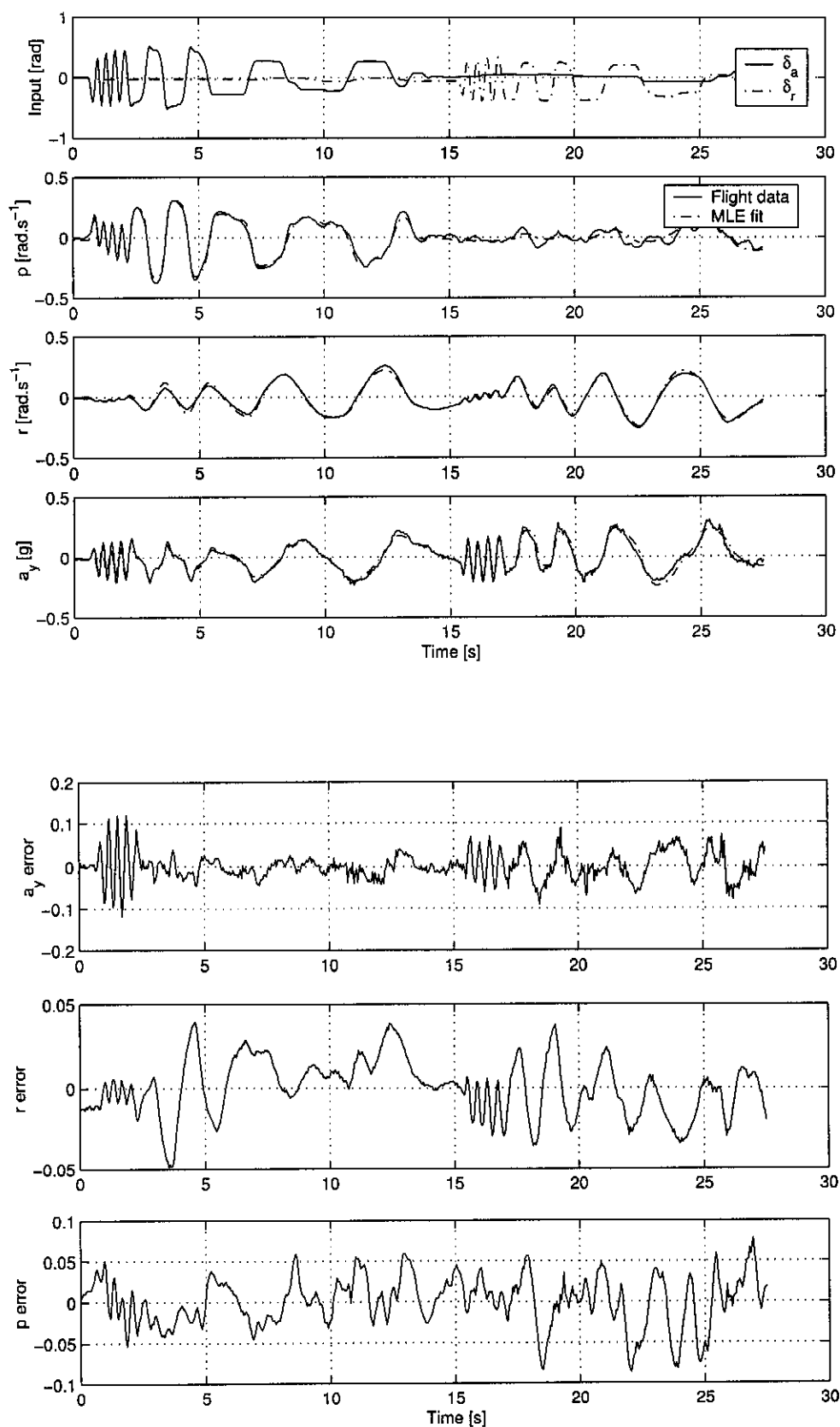


Figure A.8: Manoeuvre no.8

**Figure A.9: Manoeuvre no.9**

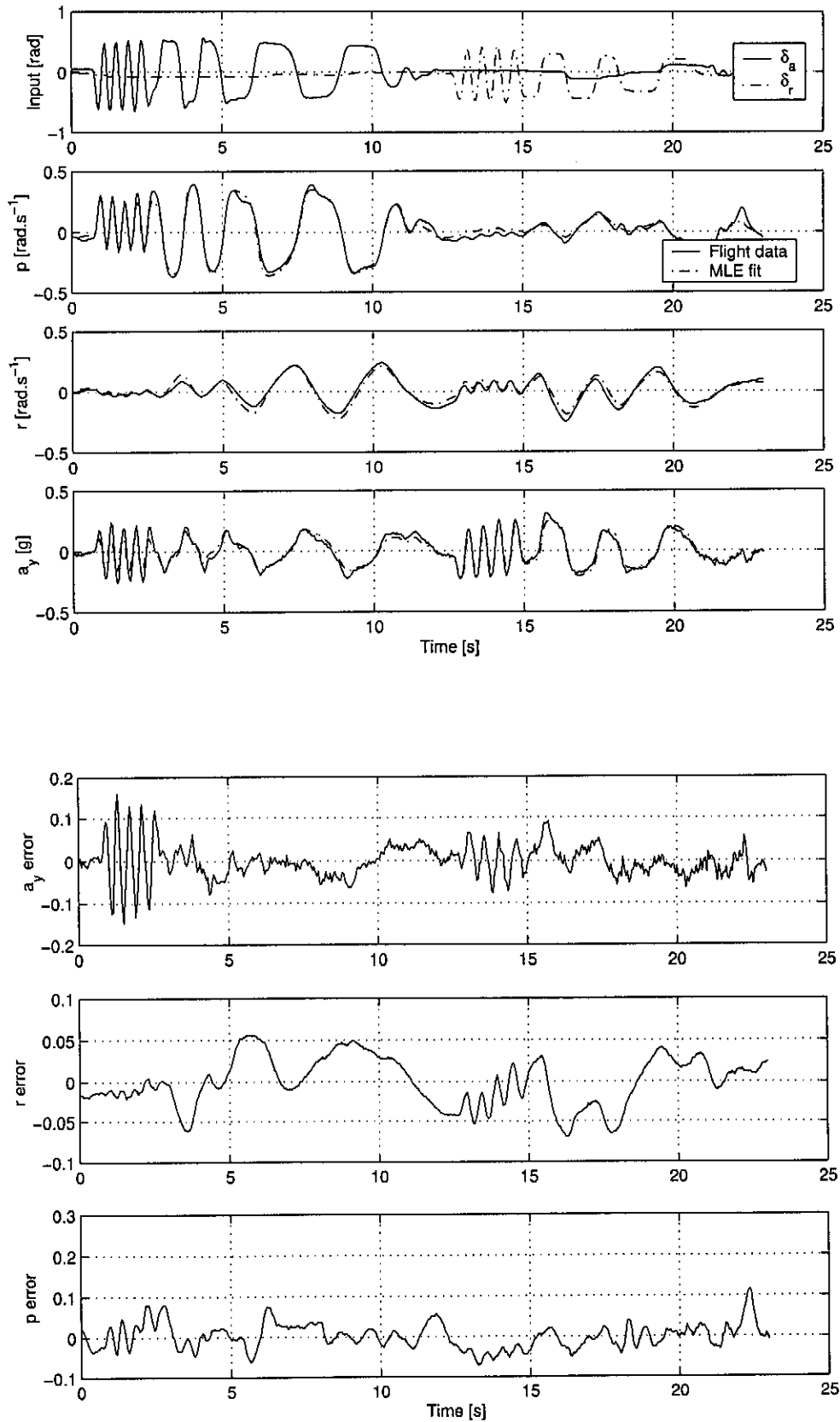


Figure A.10: Manoeuvre no.10

A.2 Flight 2

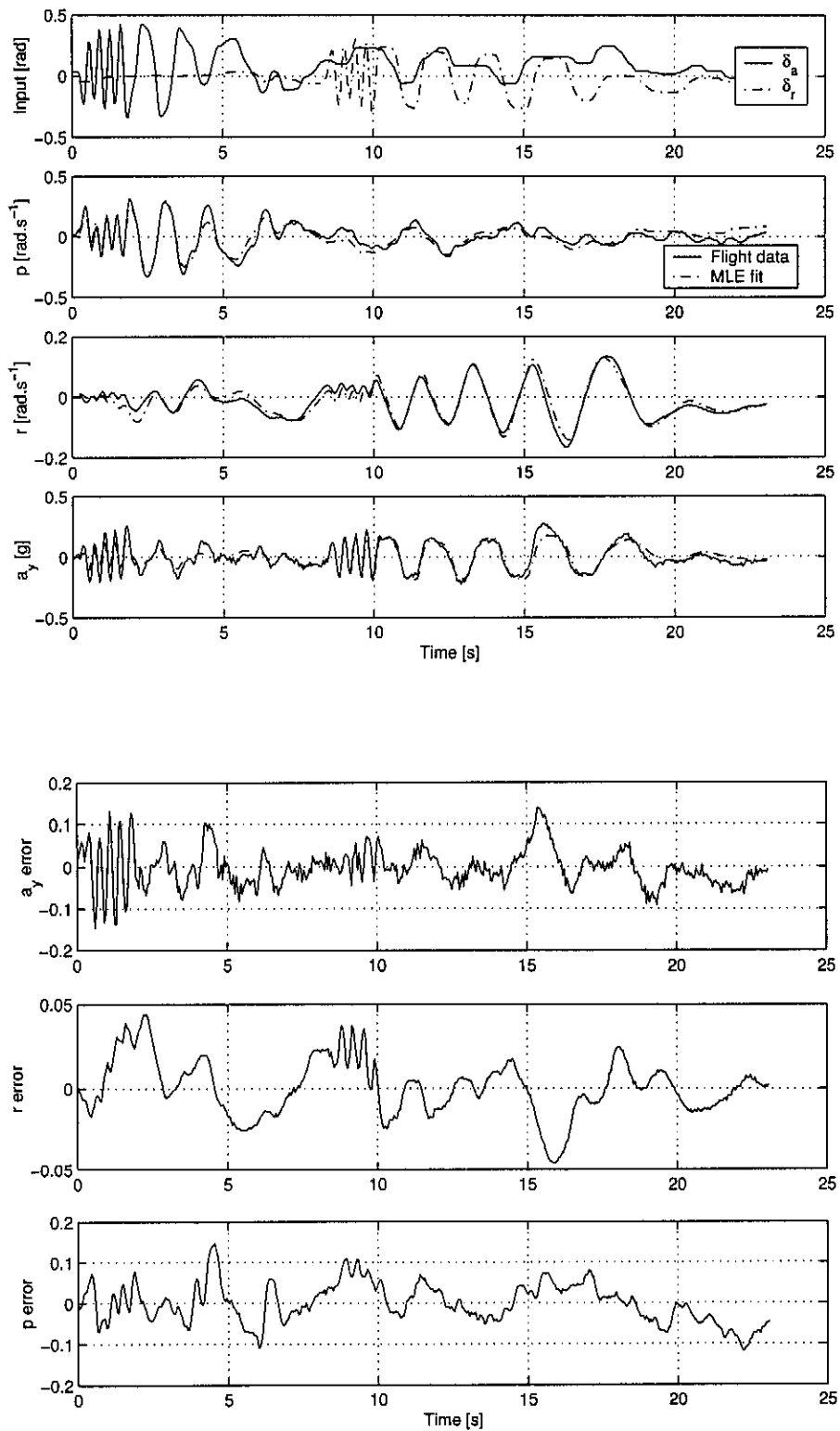


Figure A.11: Manoeuvre no.11

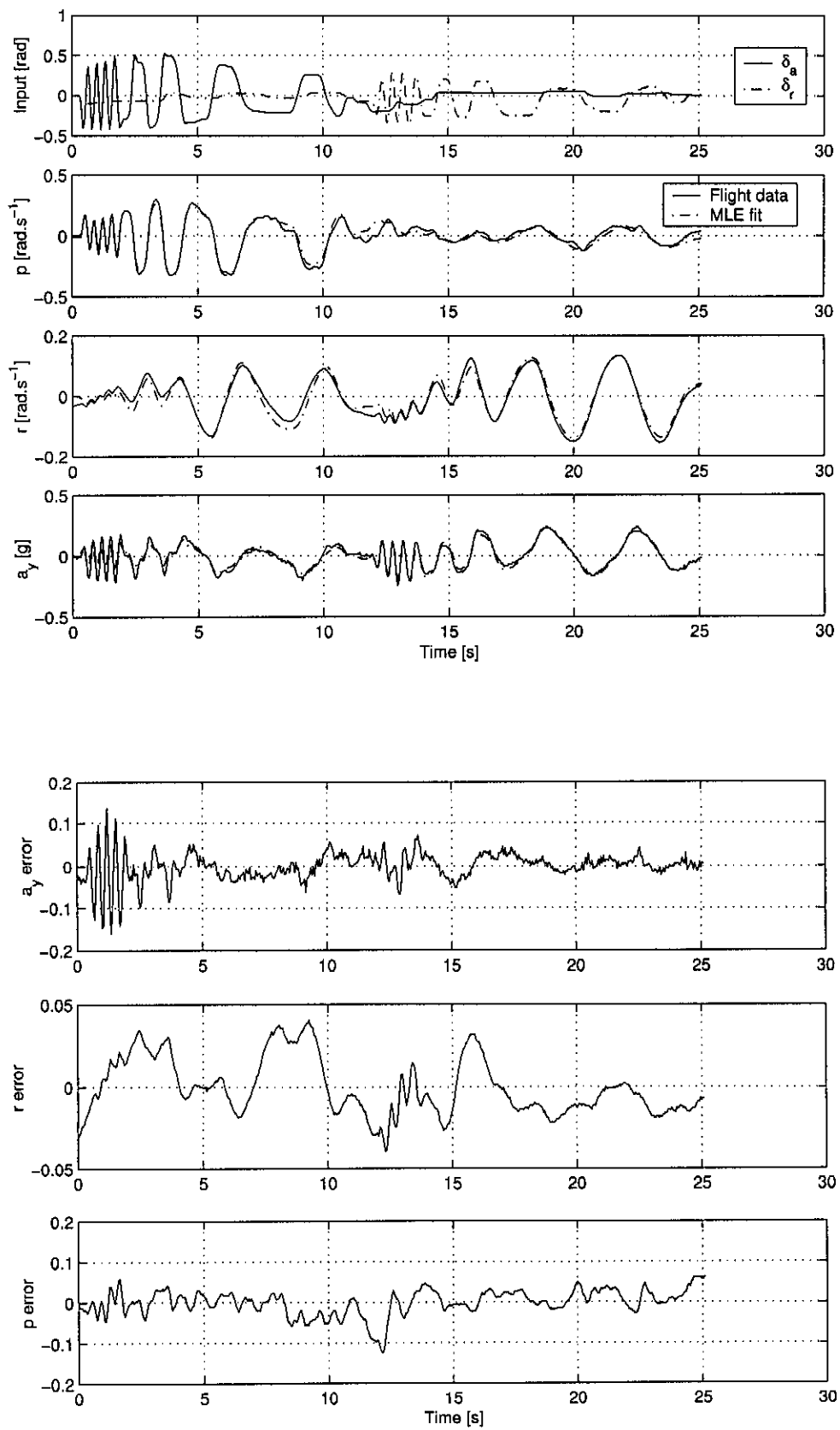


Figure A.12: Manoeuvre no.12

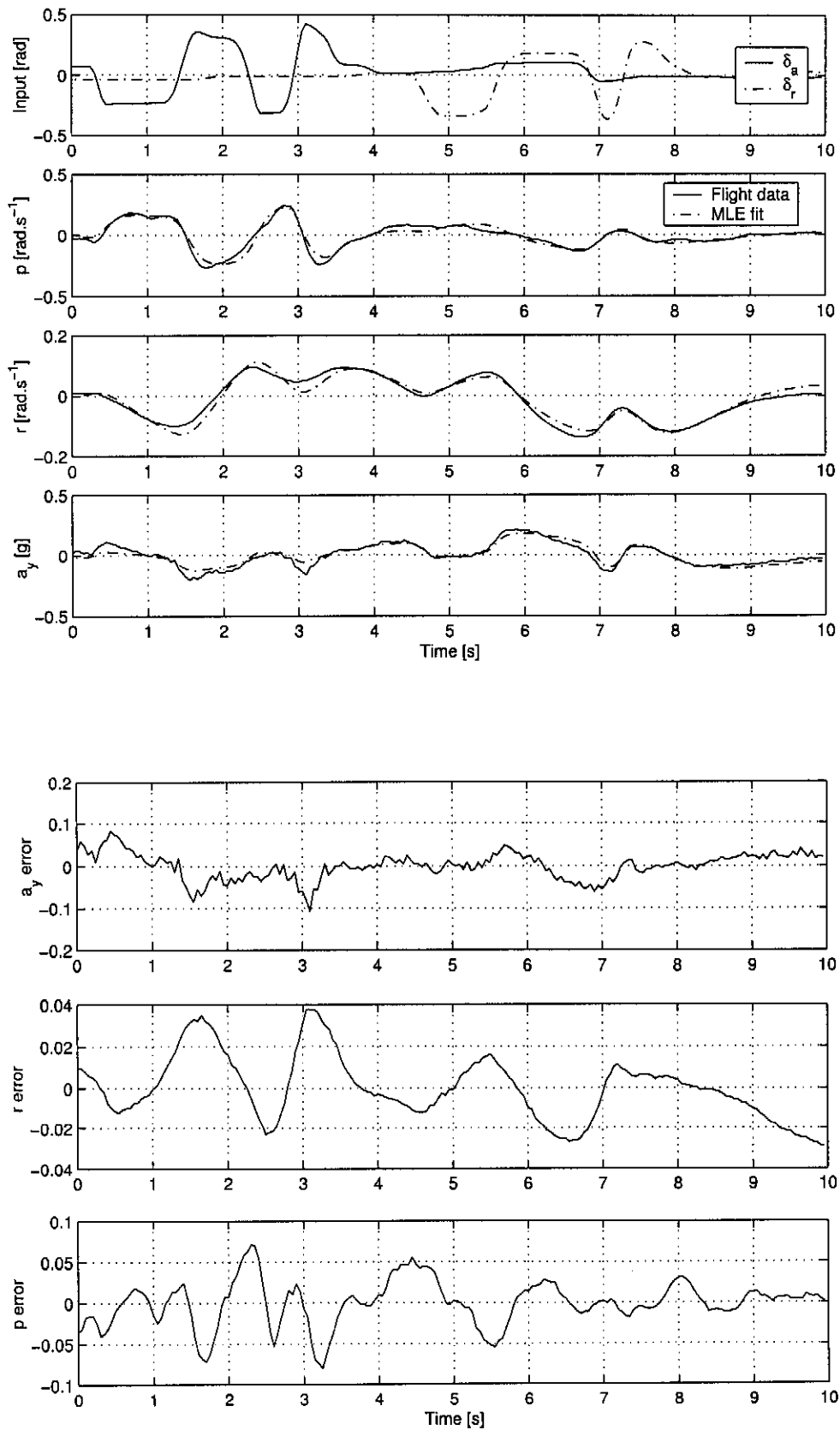


Figure A.13: Manoeuvre no.13

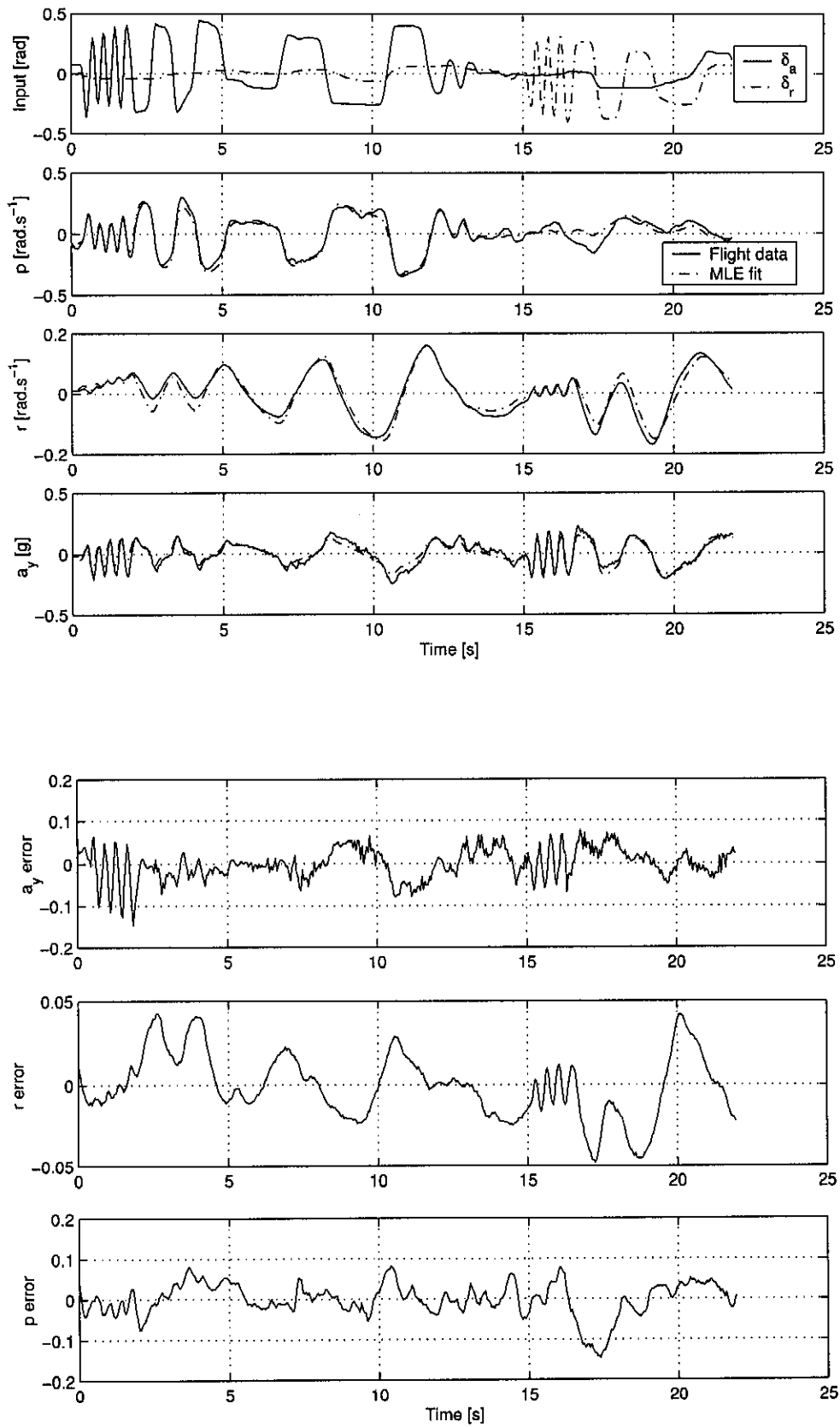


Figure A.14: *Manoeuvre no.14*

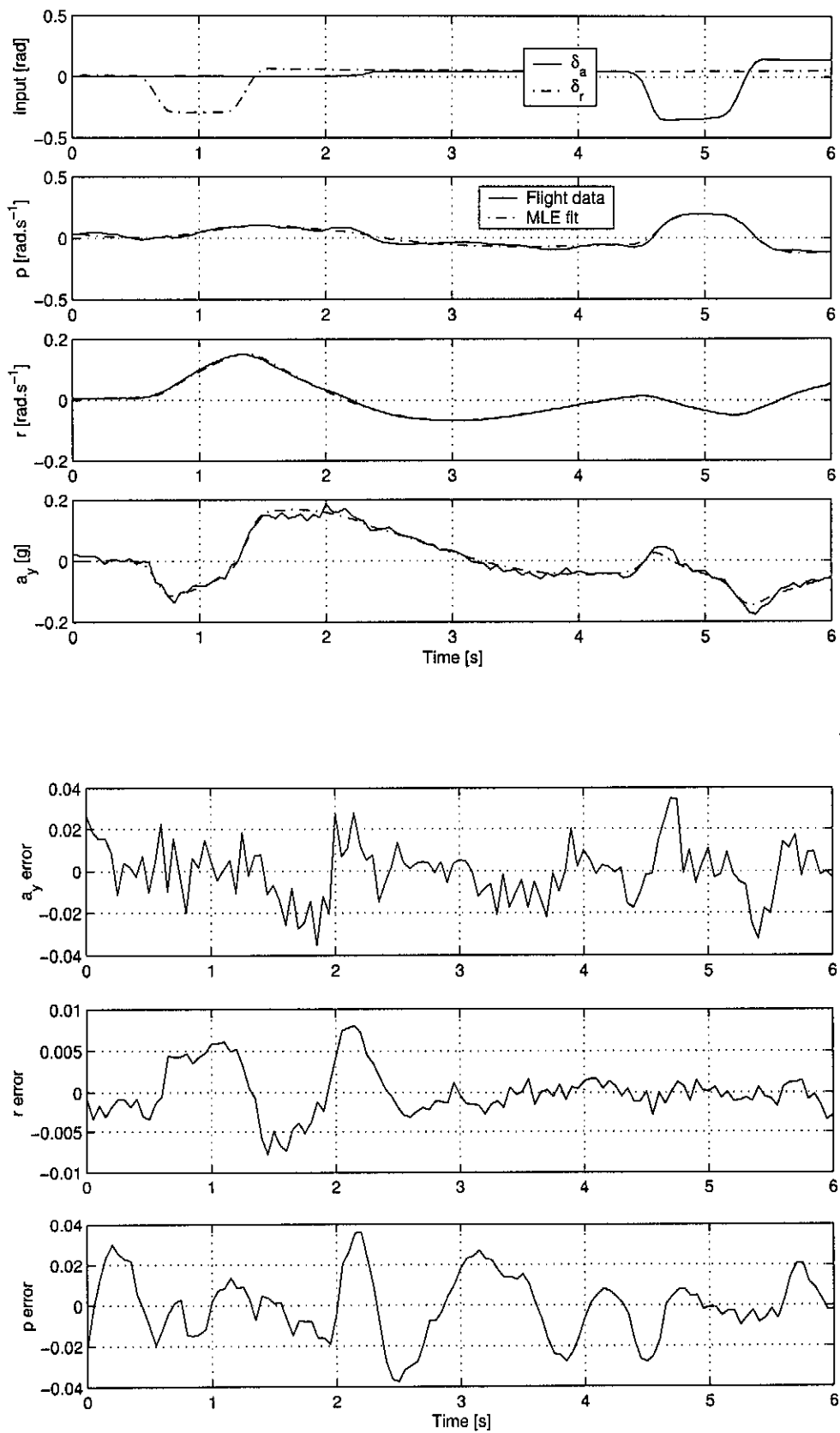


Figure A.15: Manoeuvre no.15

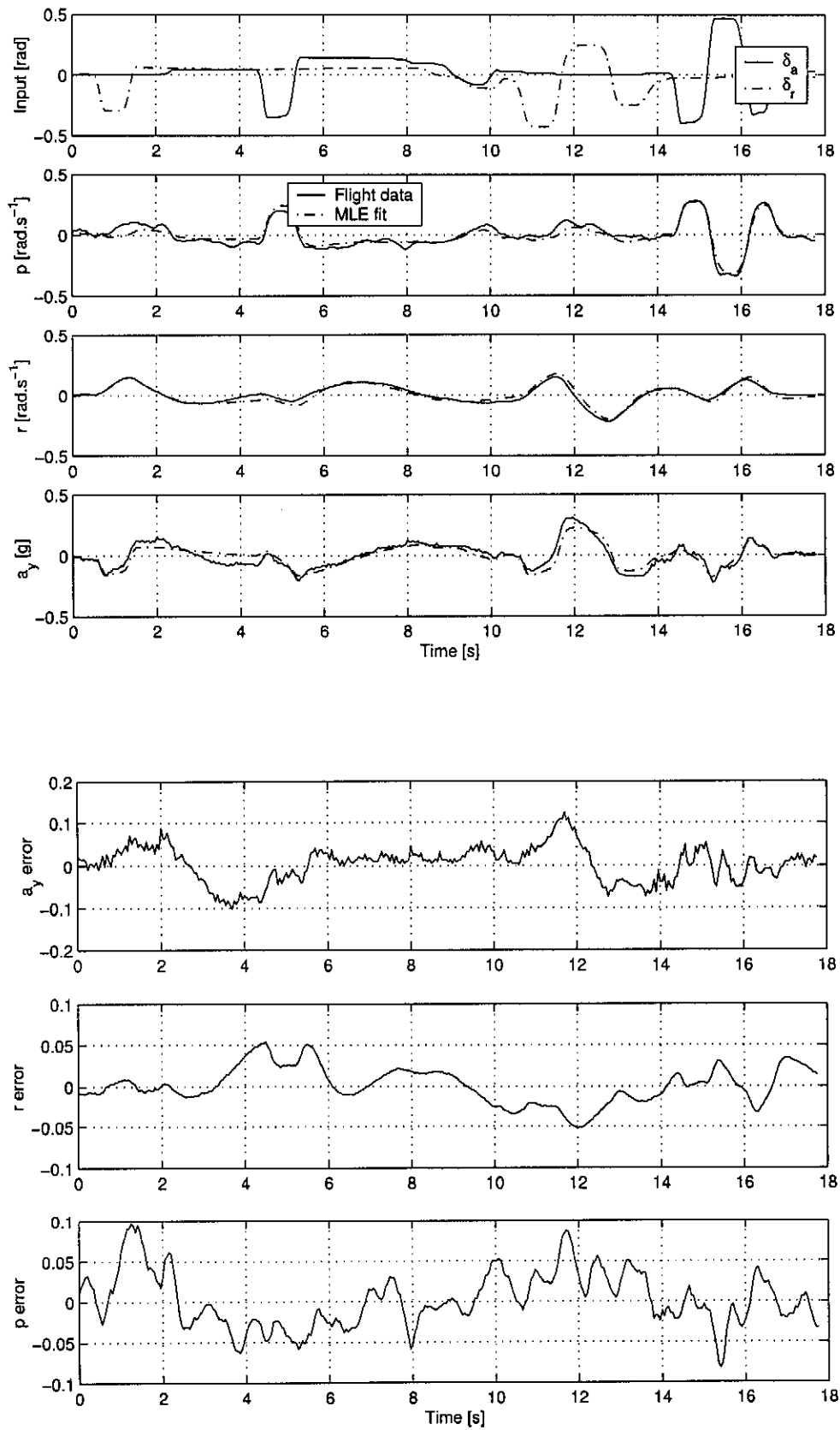


Figure A.16: Manoeuvre no.16

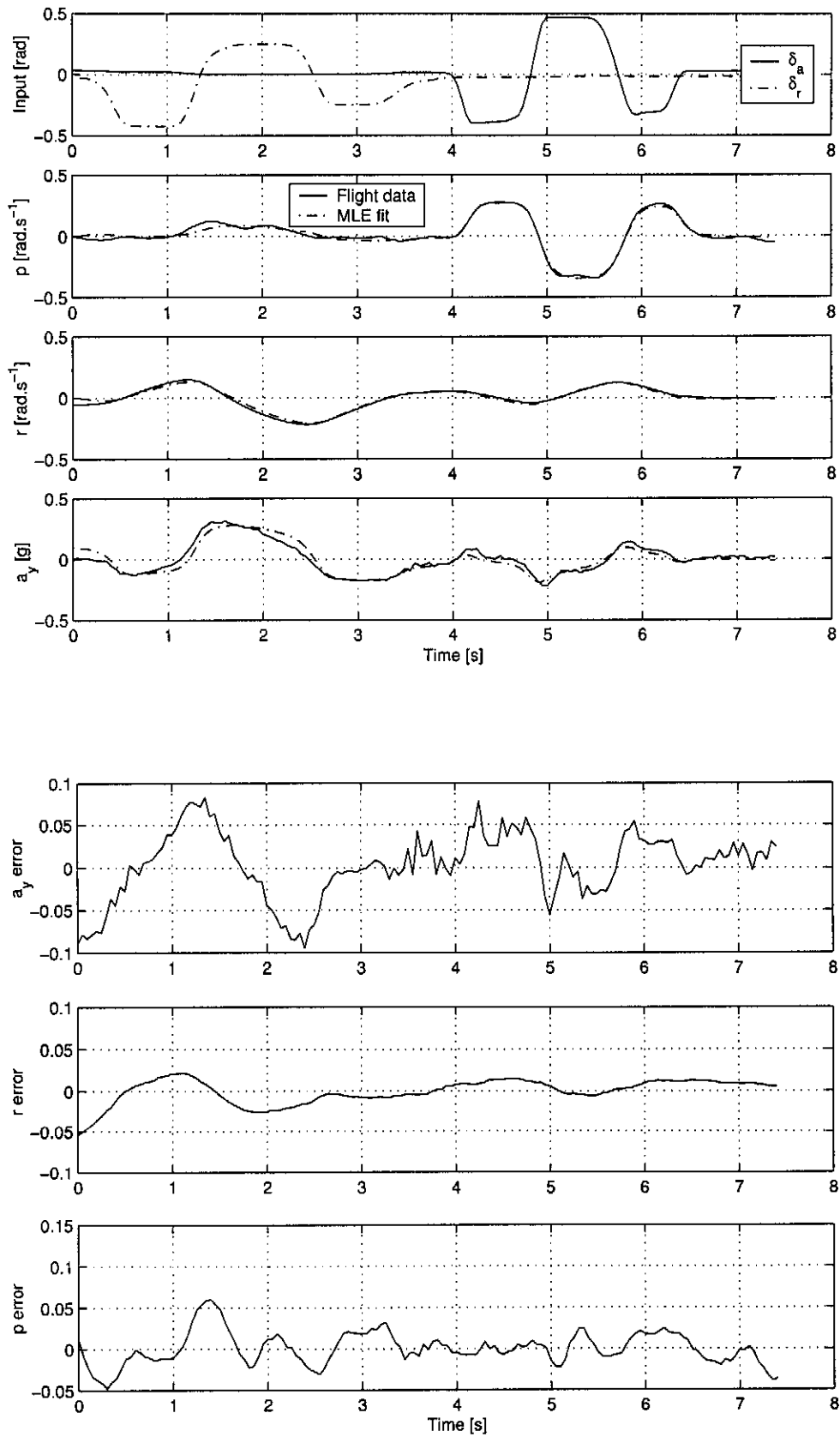


Figure A.17: Manoeuvre no.17

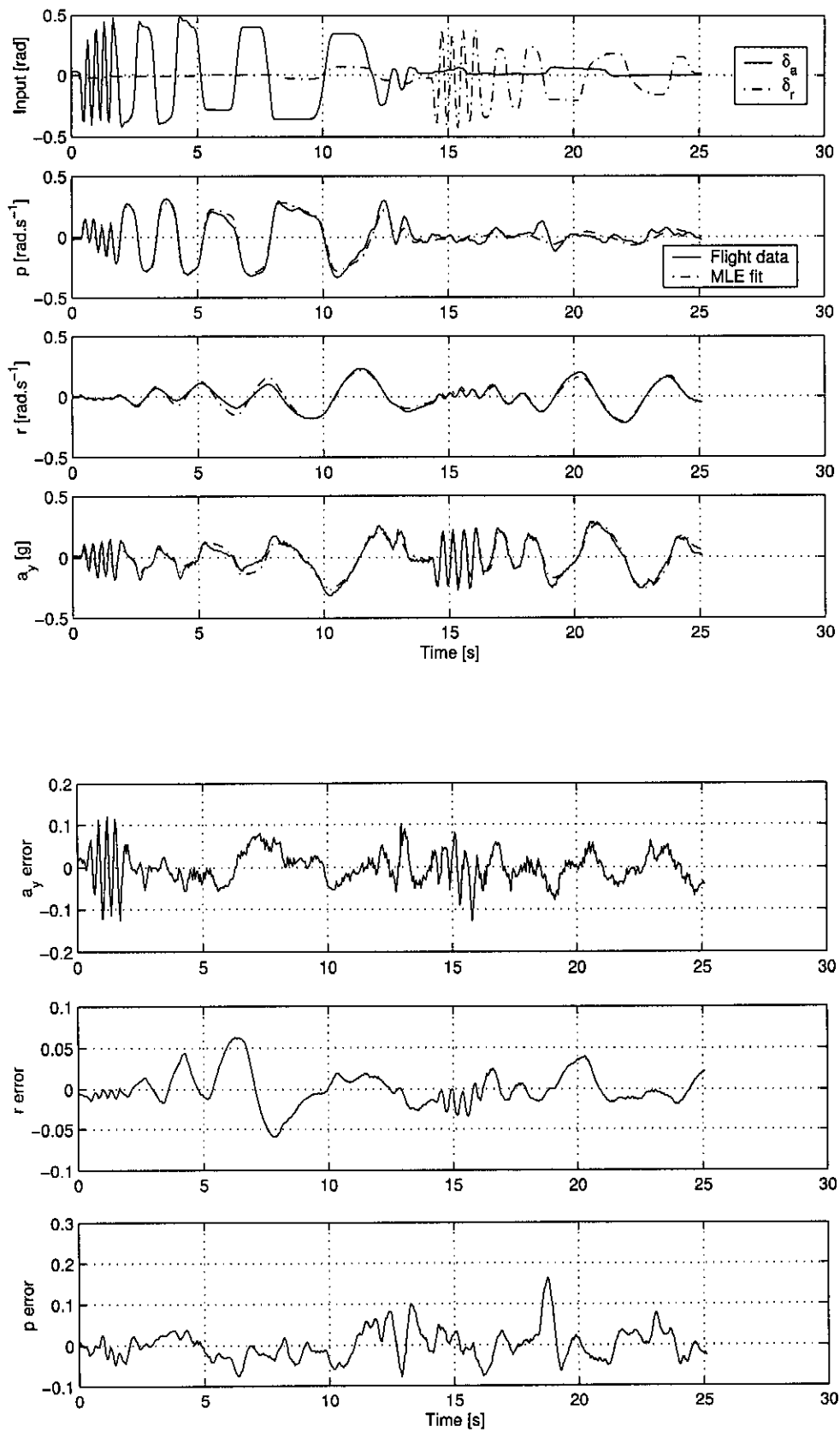


Figure A.18: *Manoeuvre no.18*

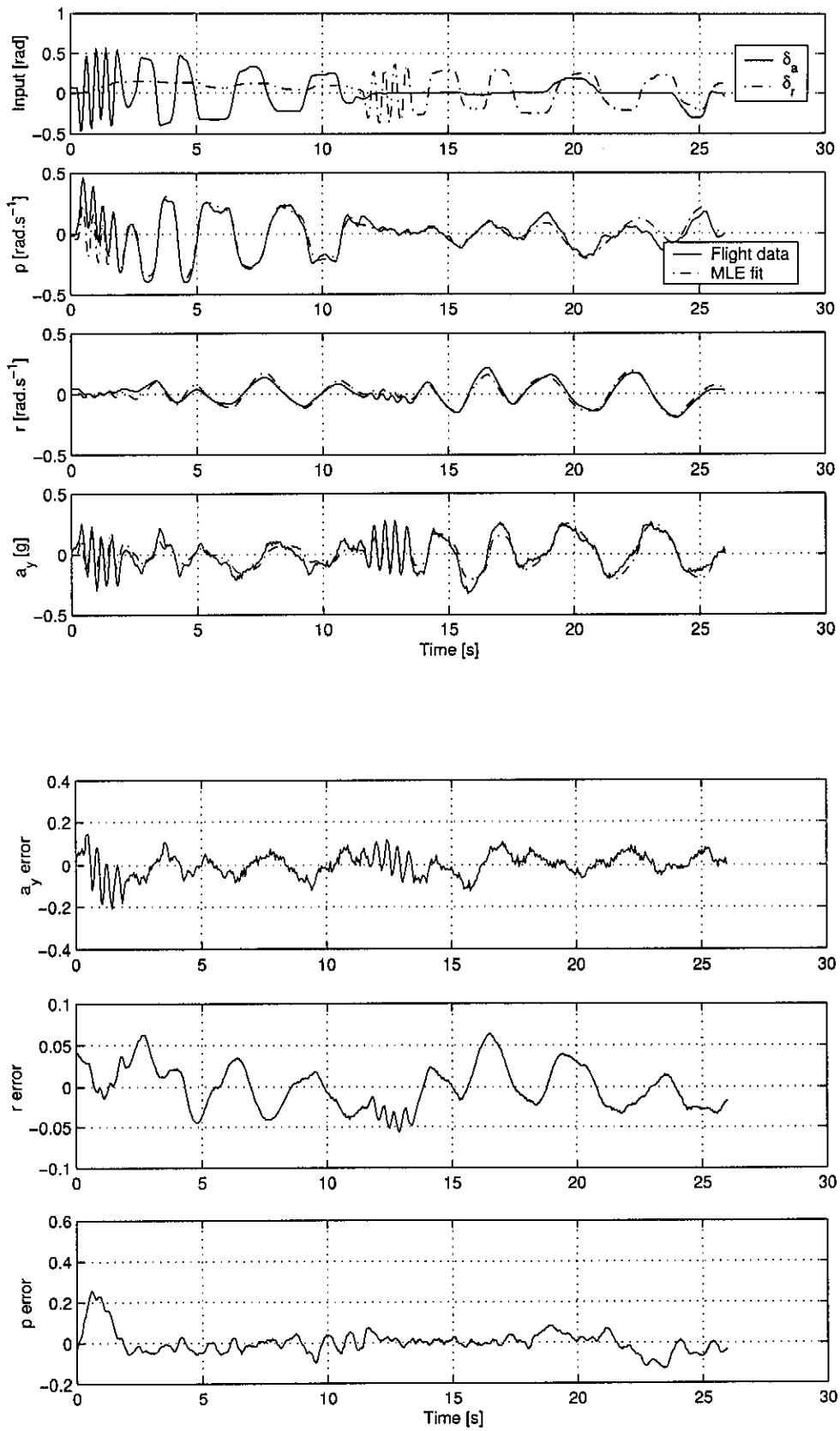


Figure A.19: *Manoeuvre no.19*

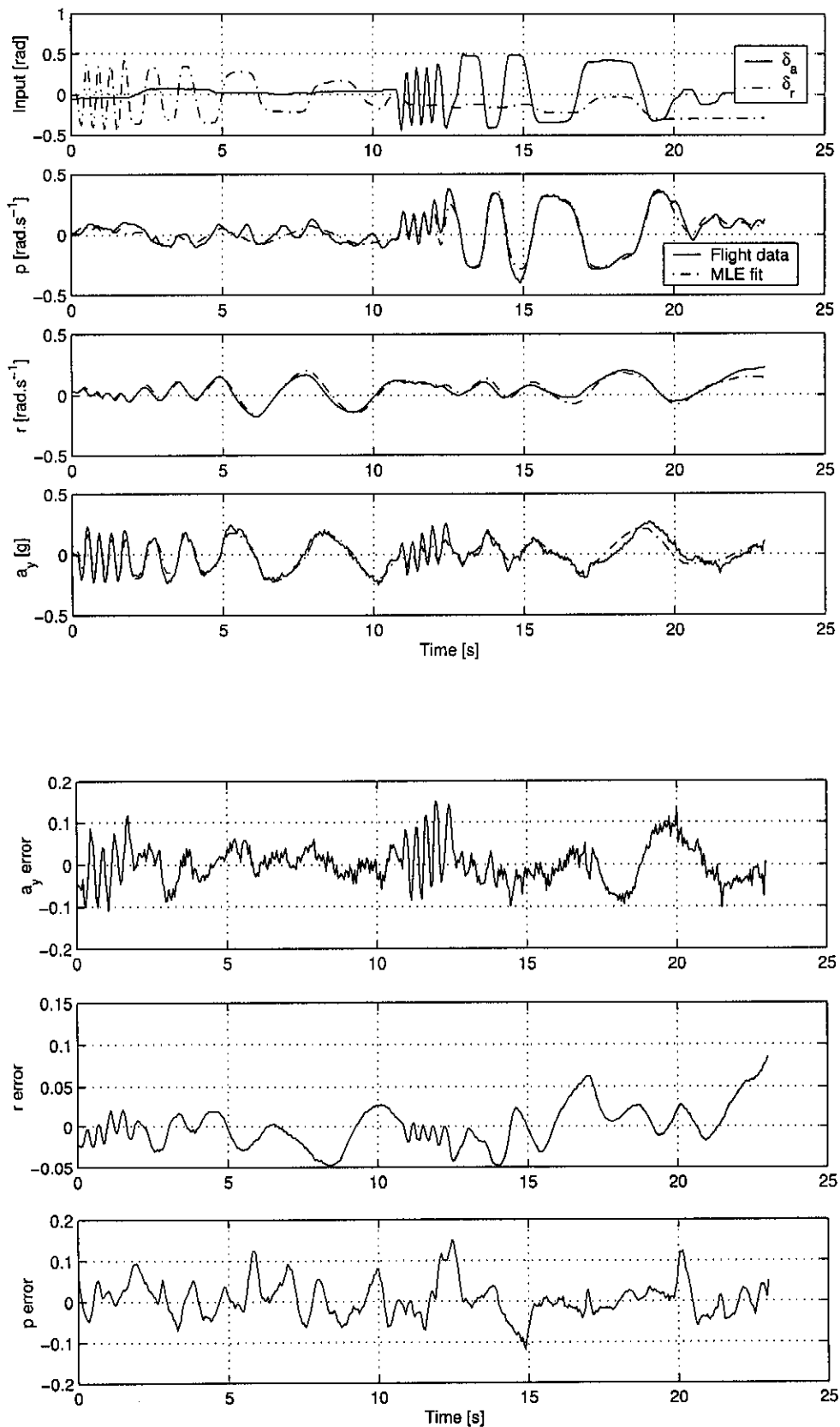


Figure A.20: *Manoeuvre no.20*

A.3 Flight 3

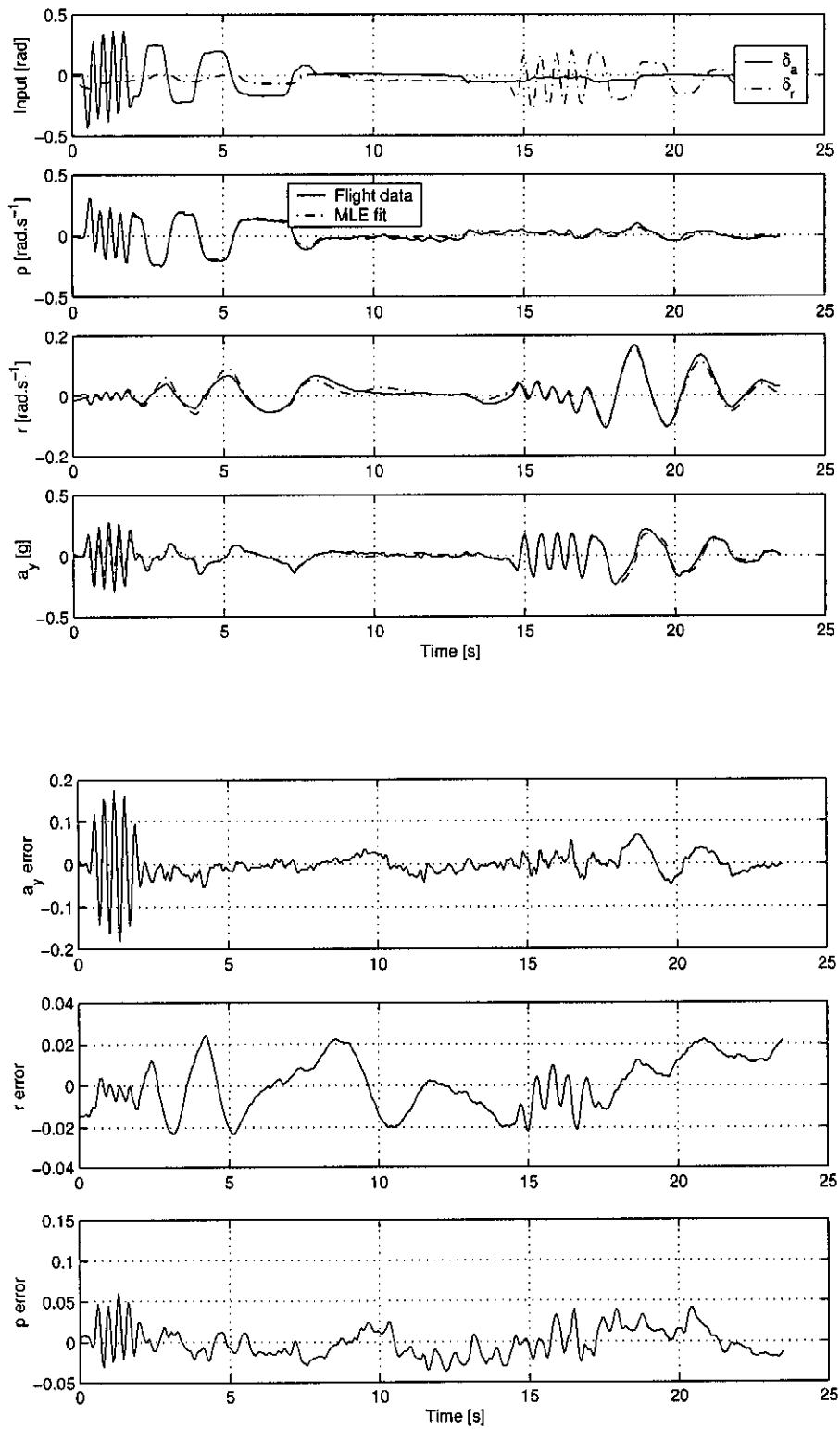


Figure A.21: Manoeuvre no.21

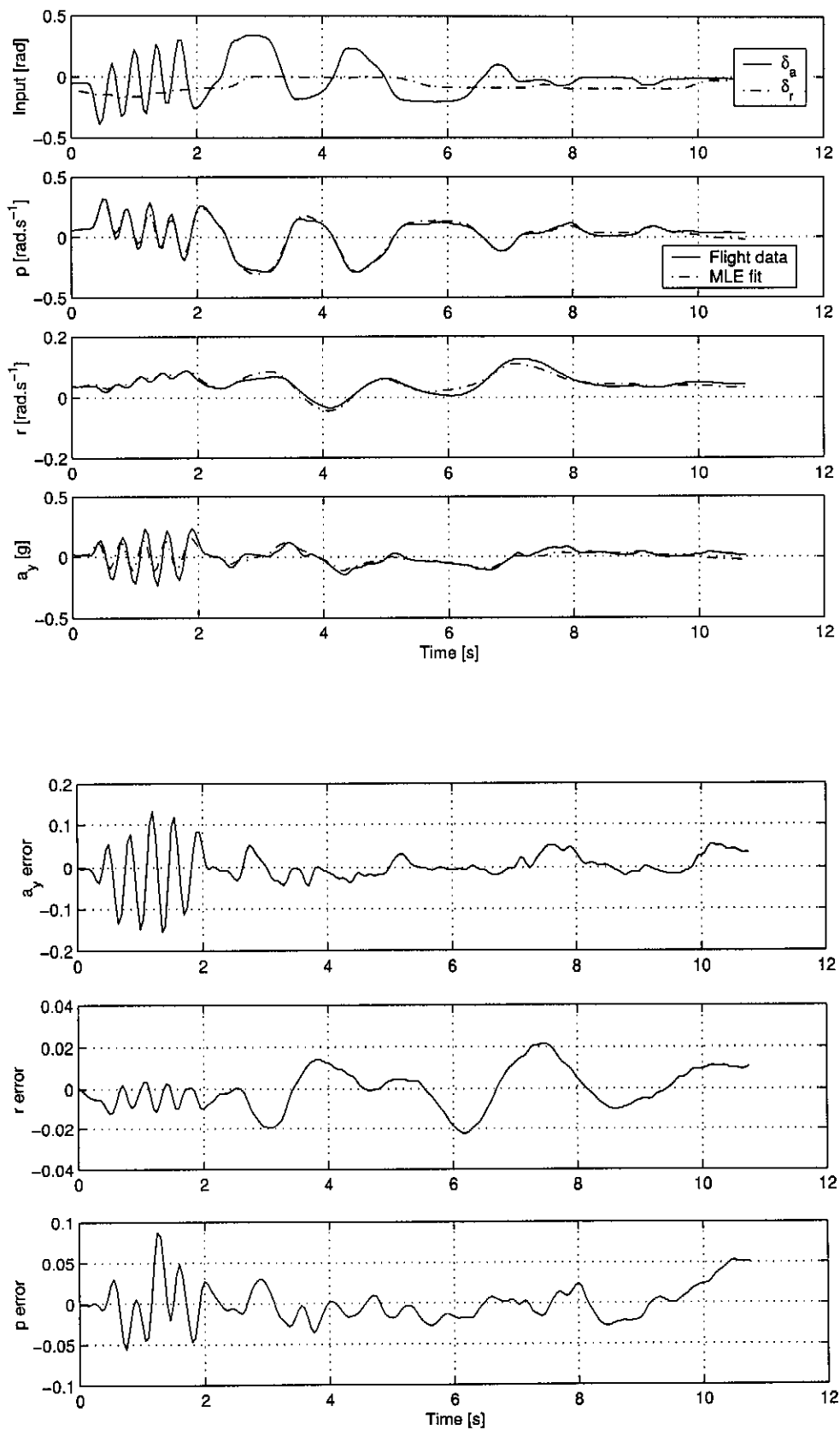


Figure A.22: *Manoeuvre no.22*

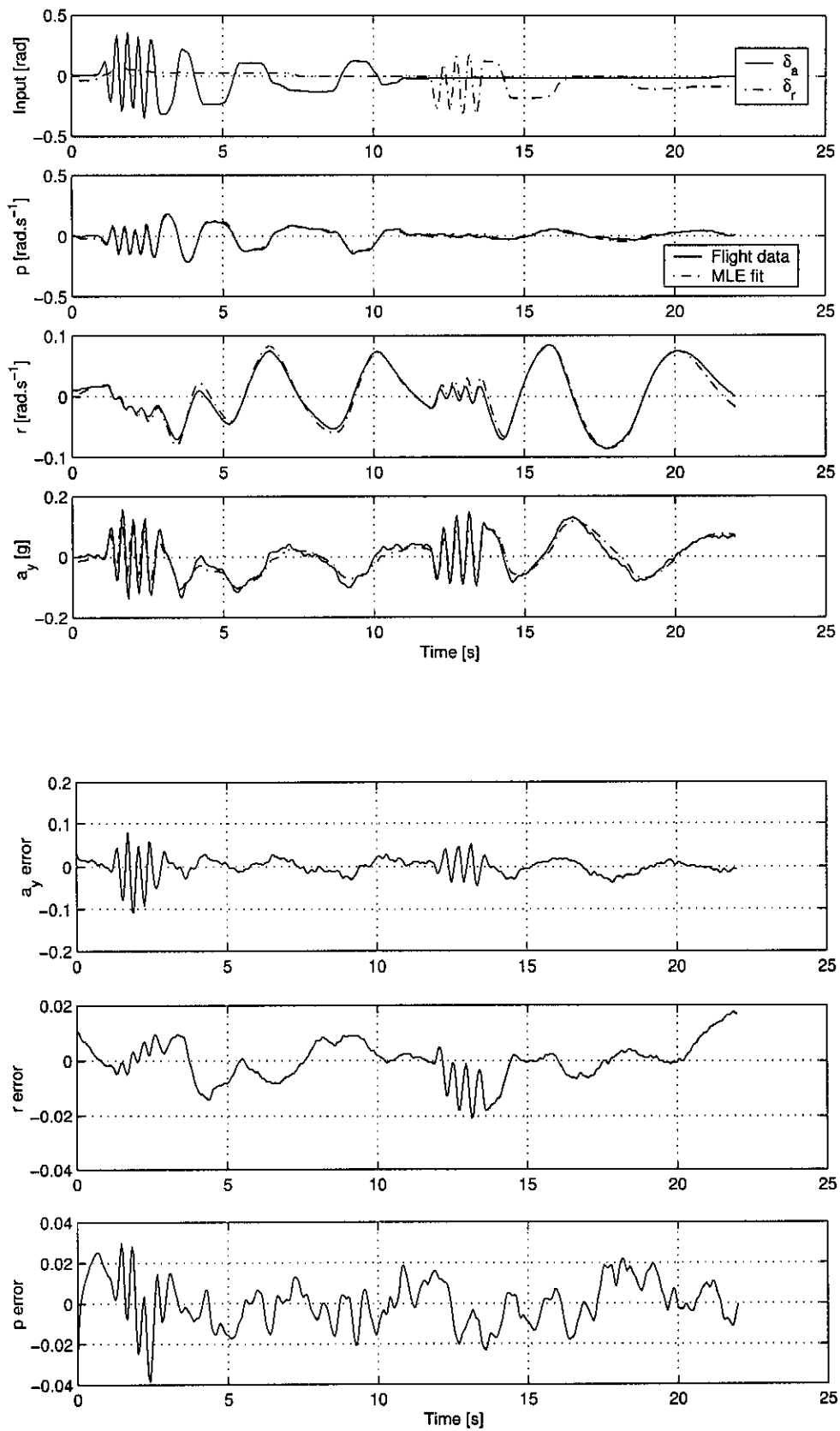


Figure A.23: Manoeuvre no.23

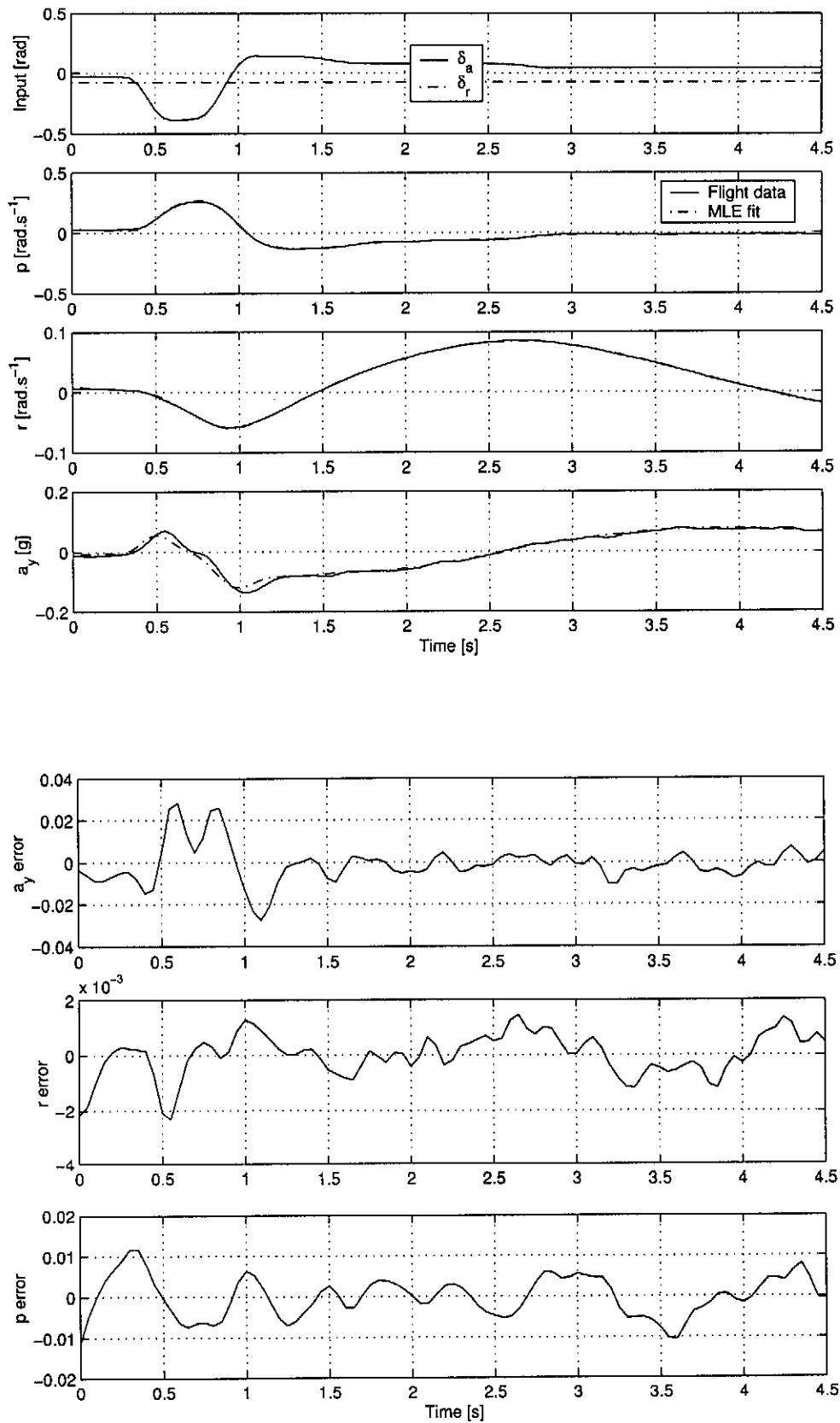


Figure A.24: Manoeuvre no.24

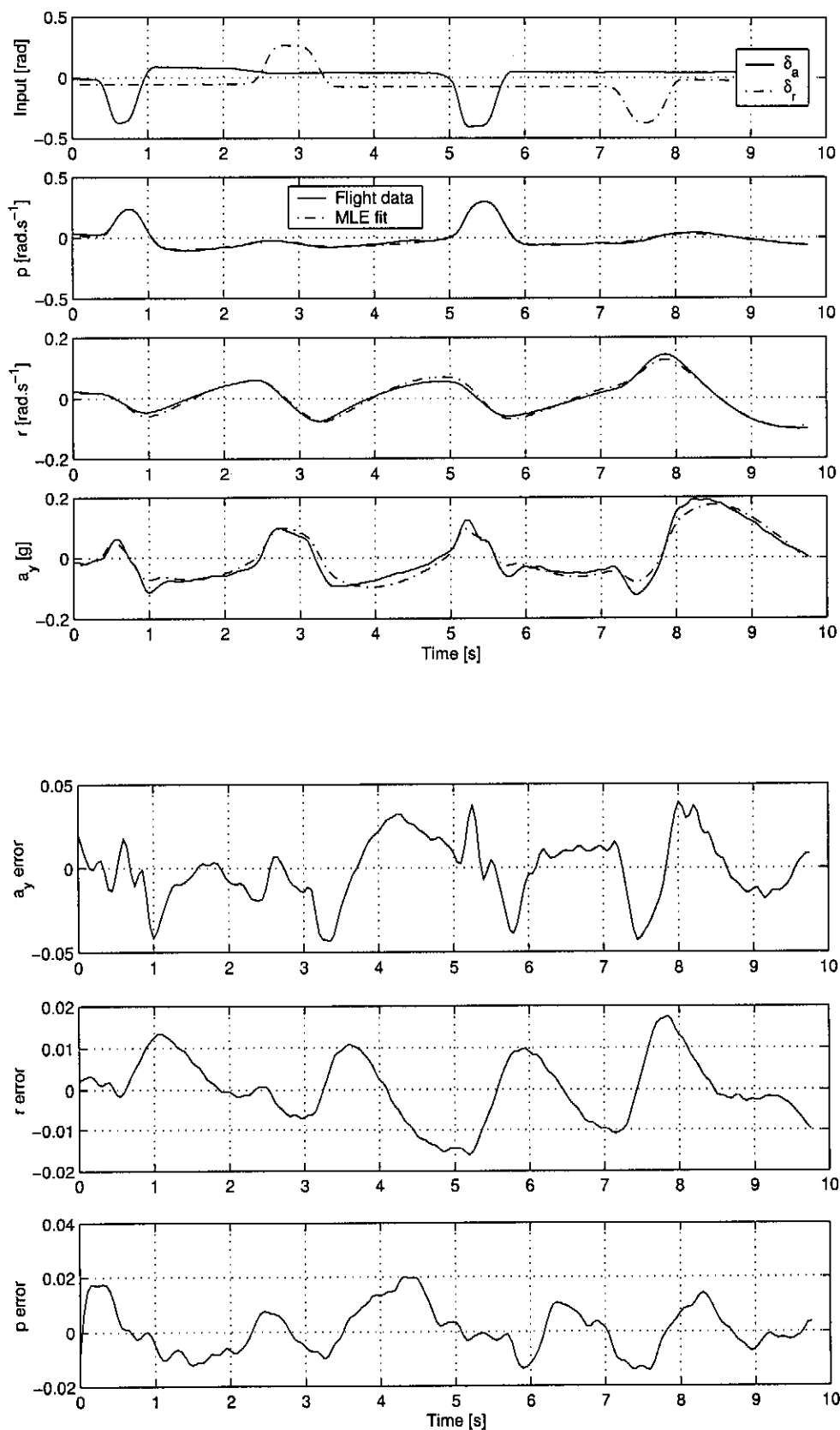


Figure A.25: *Manoeuvre no.25*

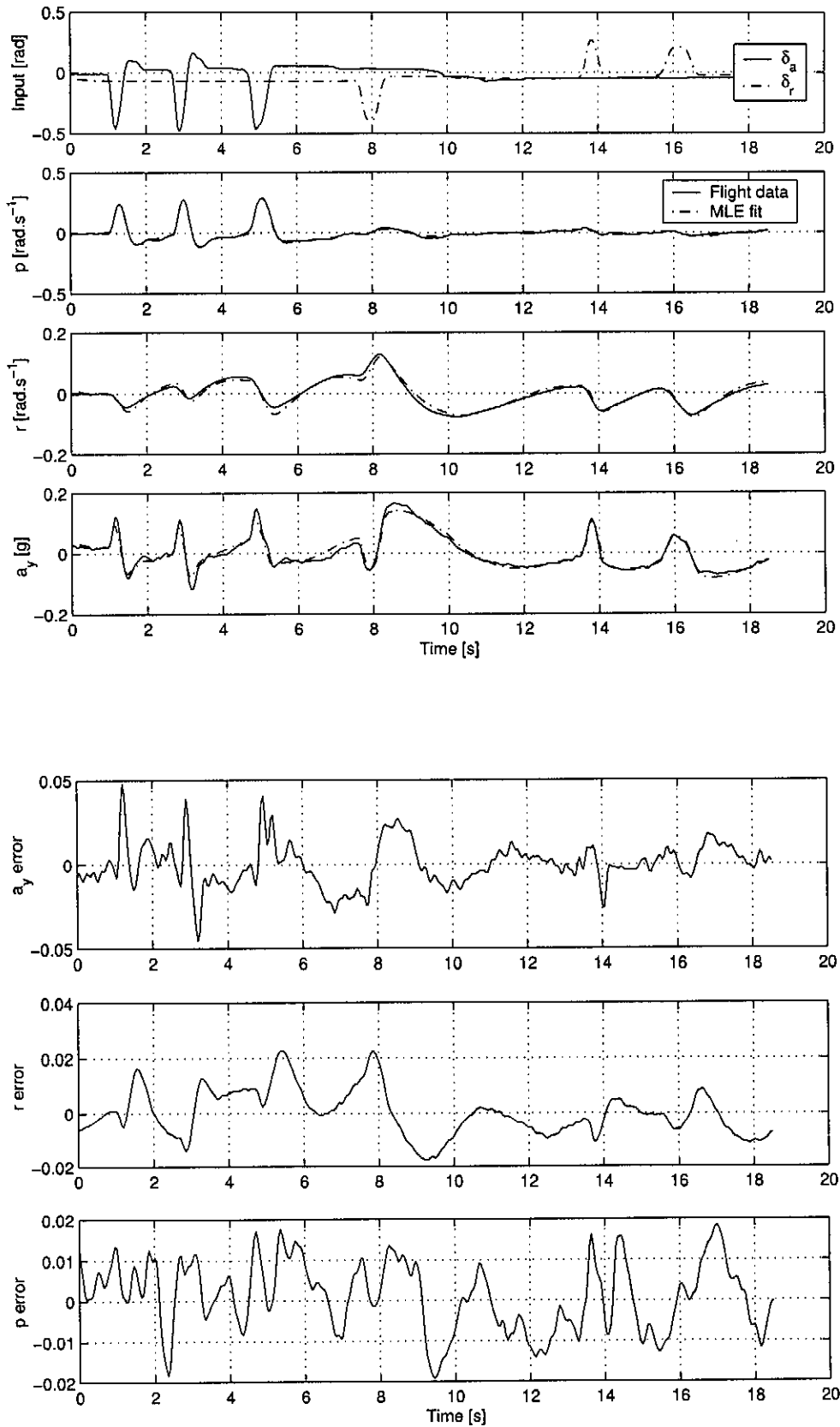


Figure A.26: Manoeuvre no.26

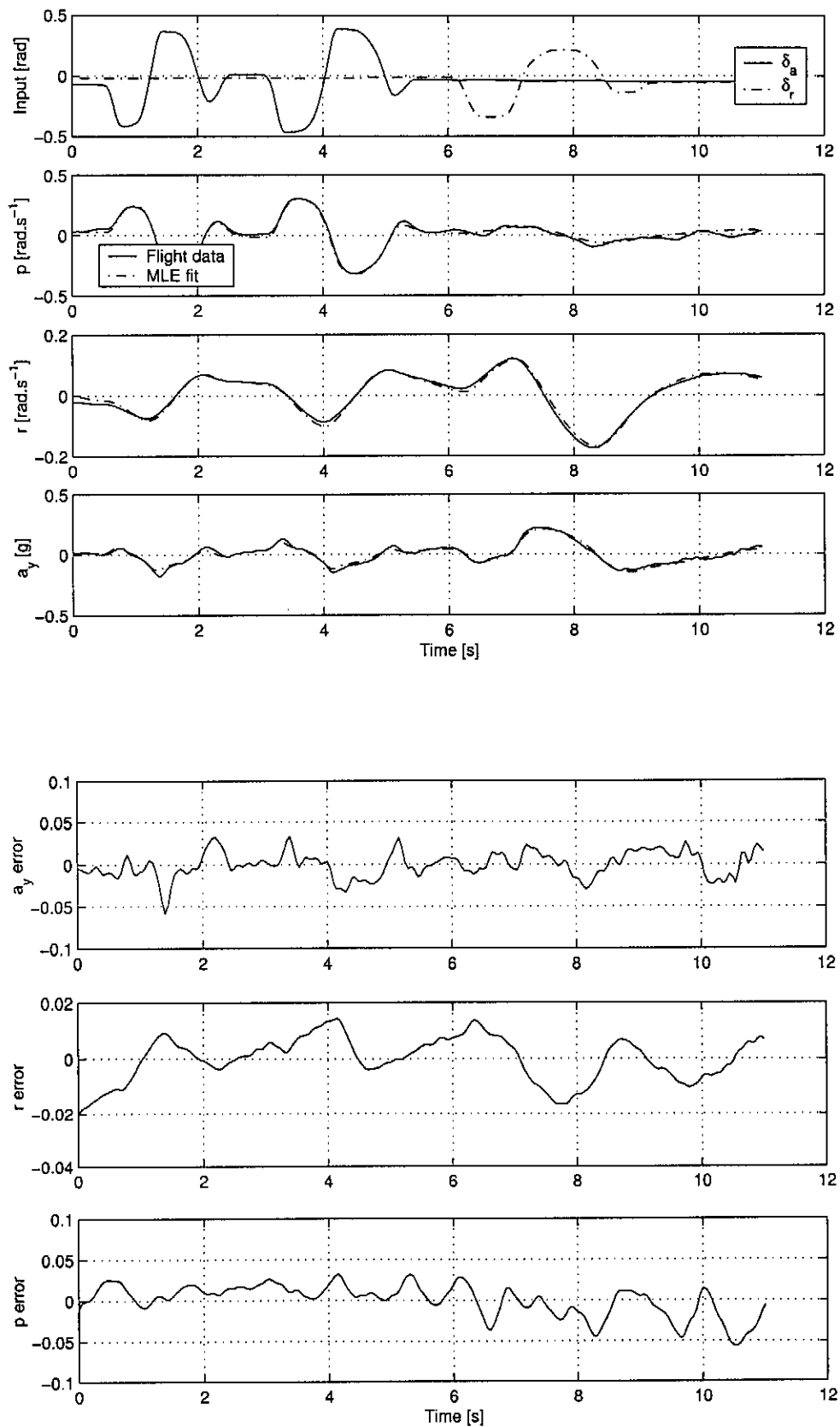


Figure A.27: Manoeuvre no.27

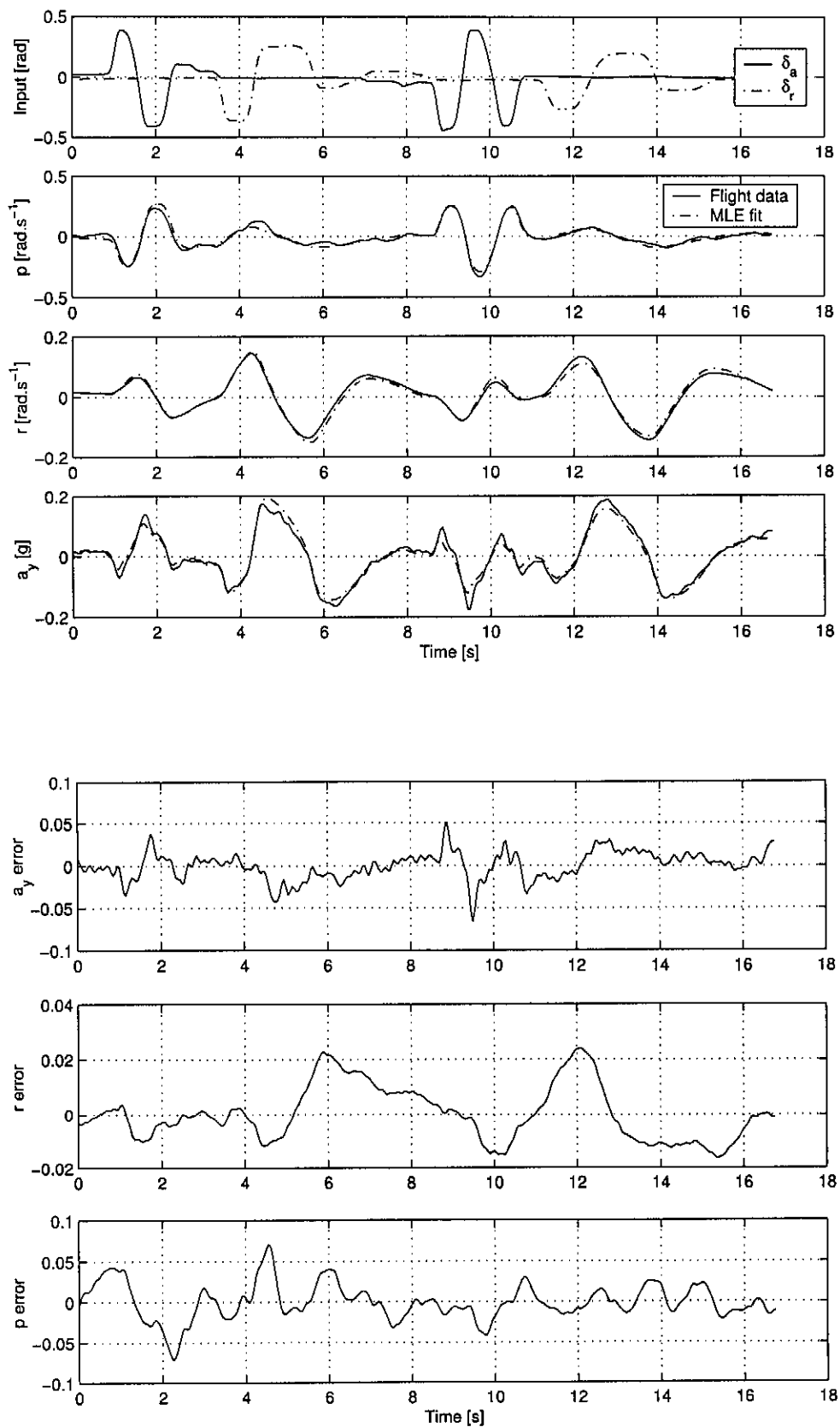


Figure A.28: *Manoeuvre no.28*

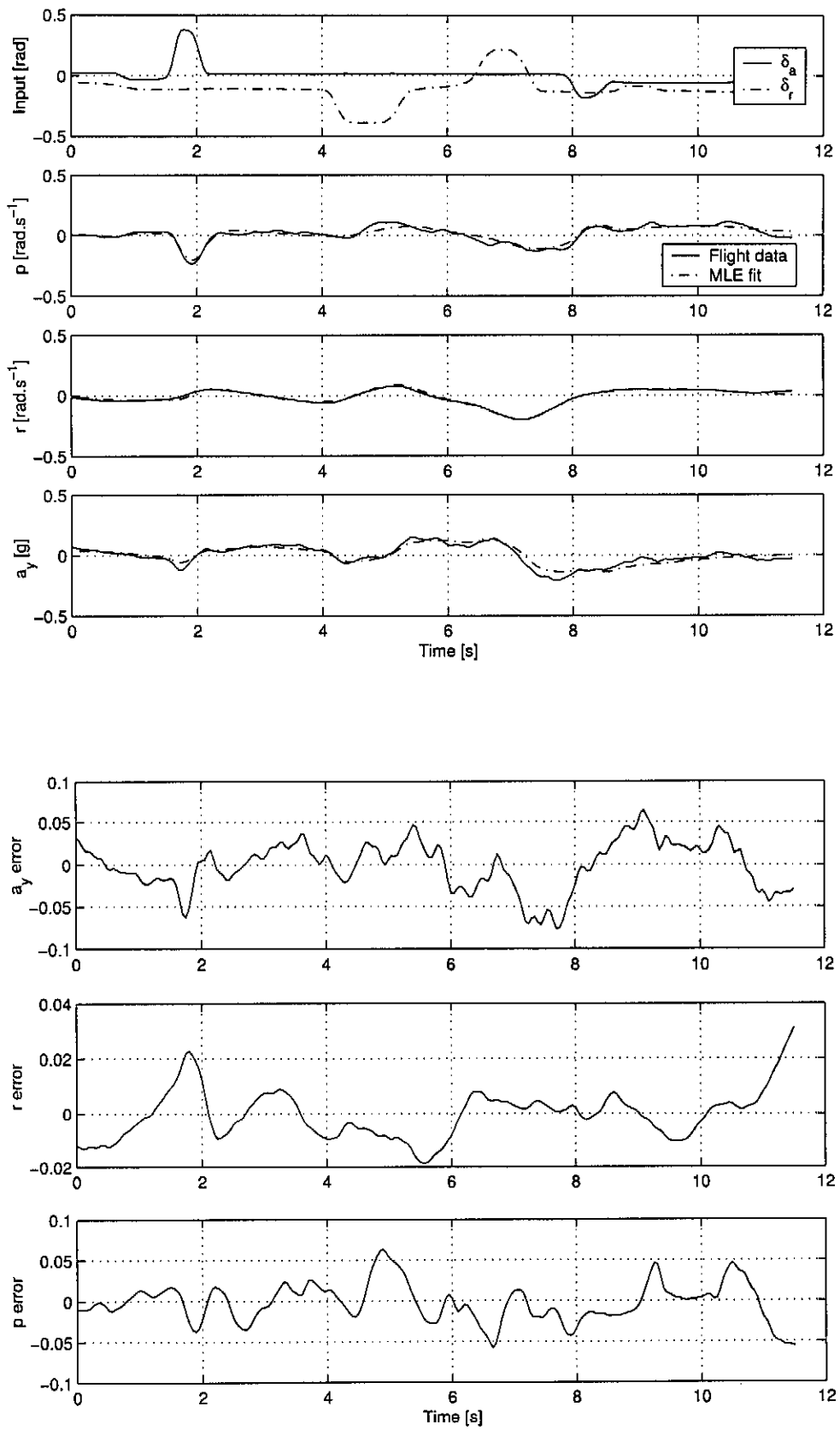


Figure A.29: *Manoeuvre no.29*

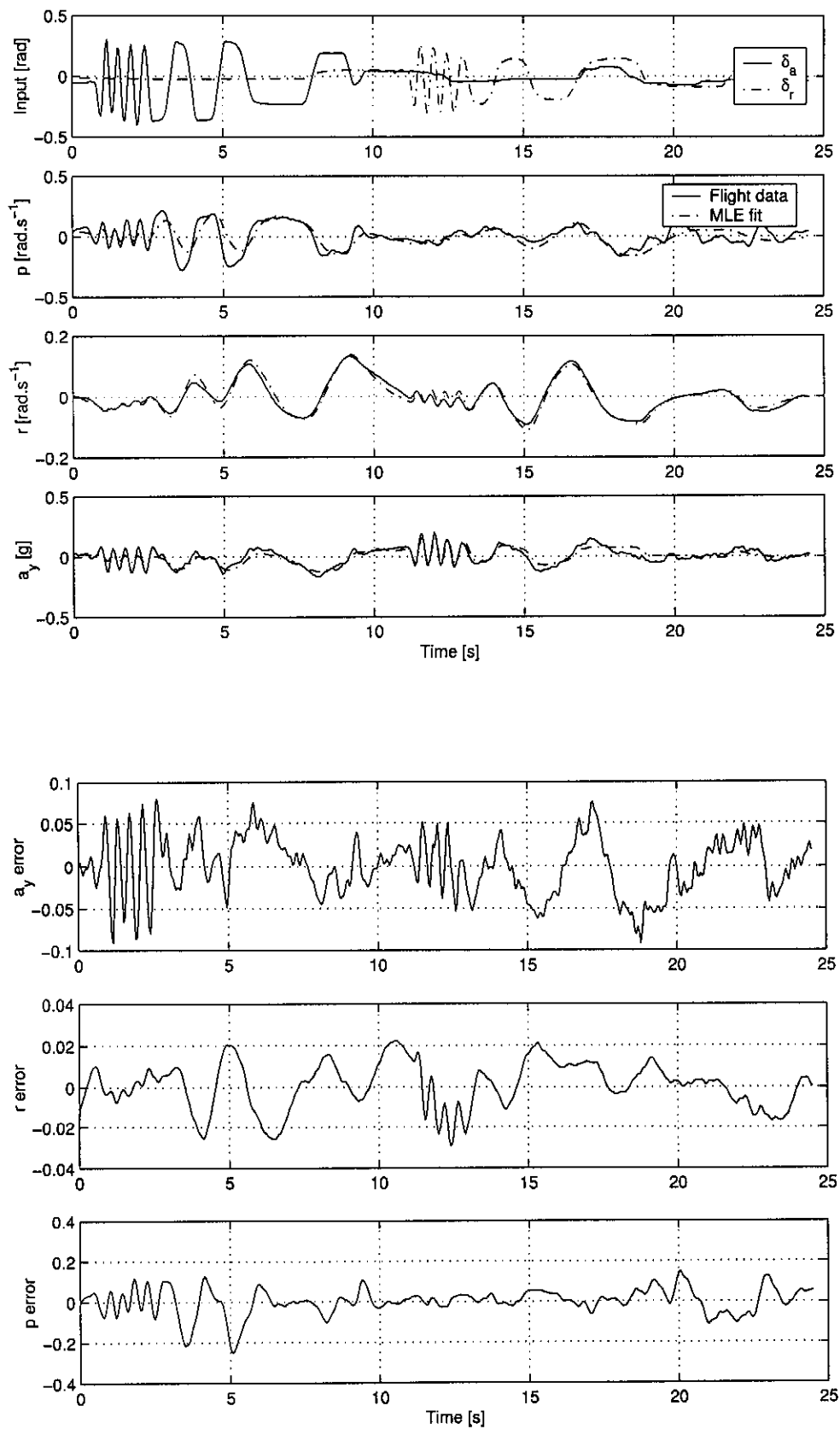


Figure A.30: *Manoeuvre no.29*

Appendix B

Longitudinal Manoeuvres

The manoeuvres are grouped in flights. There were three different flights that collected data while manoeuvres were flown.

B.1 Flight 1

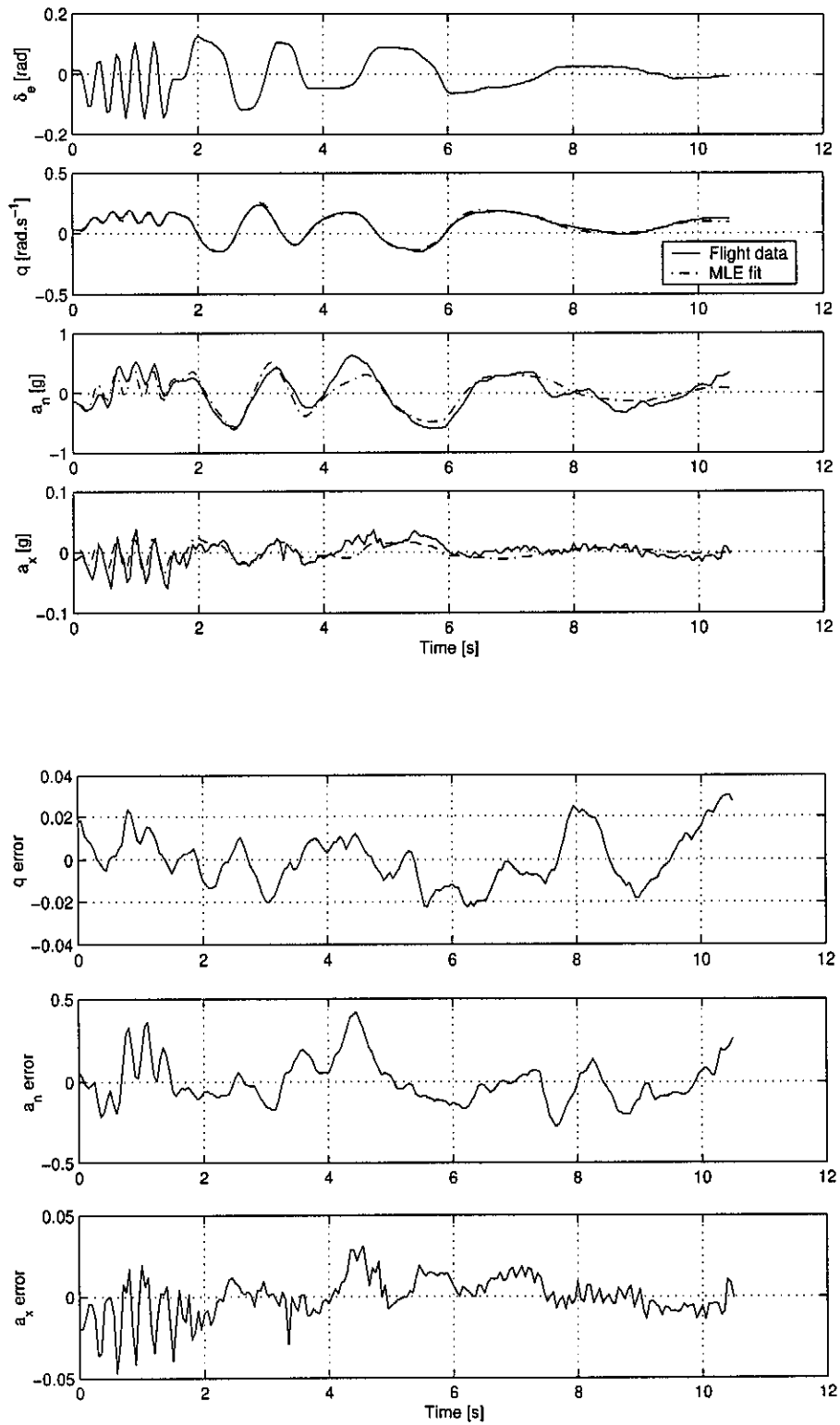


Figure B.1: Manoeuvre no. 1

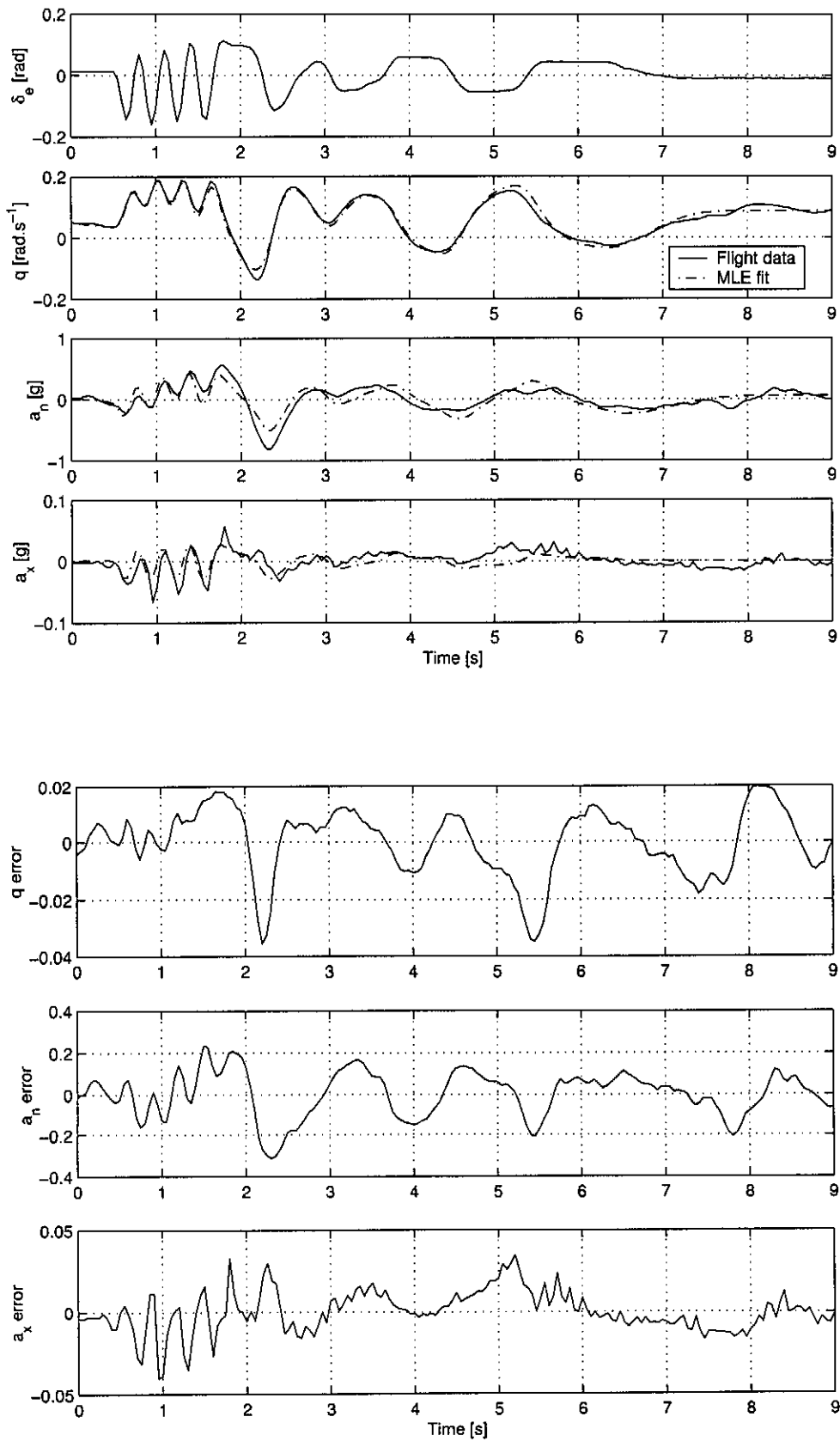
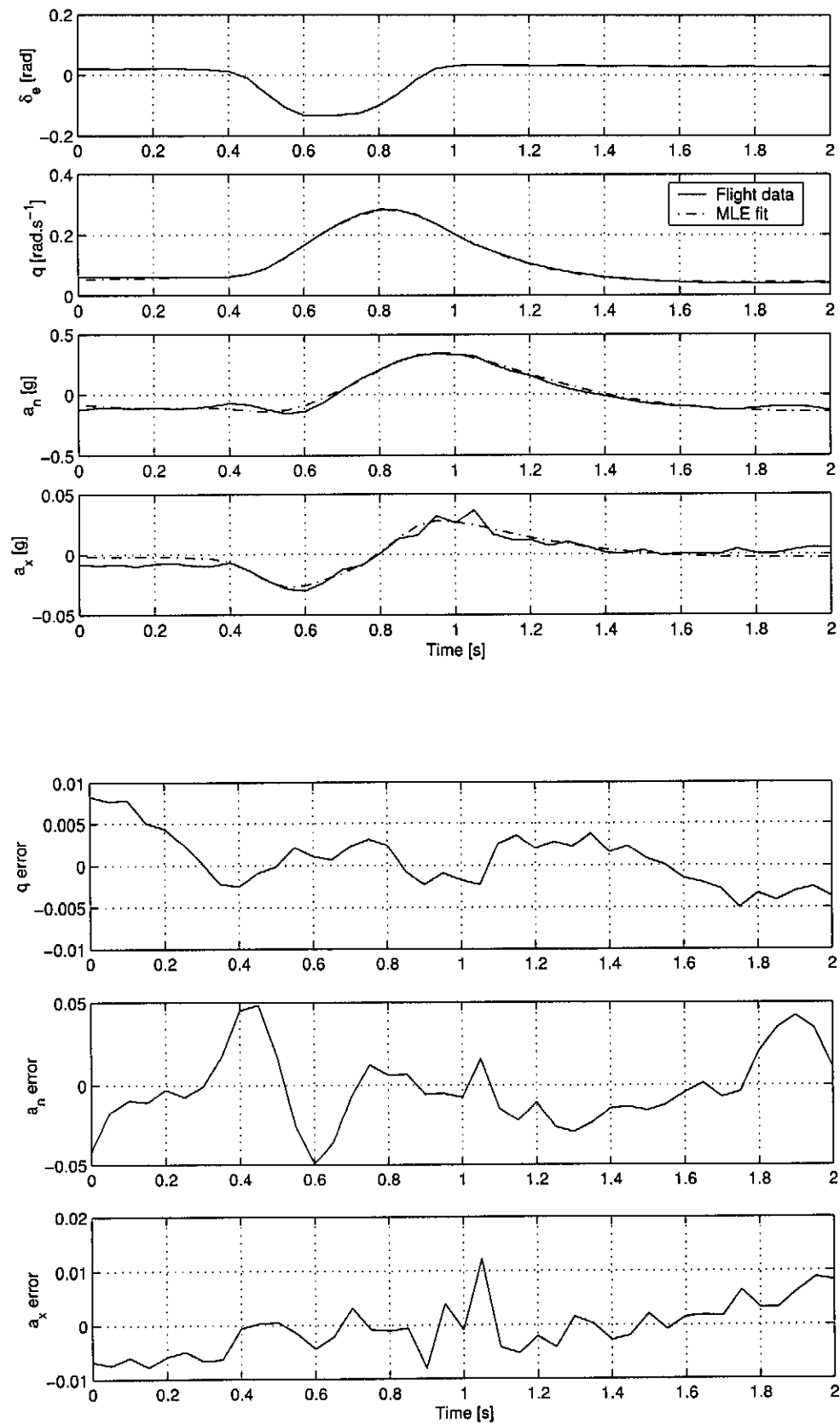
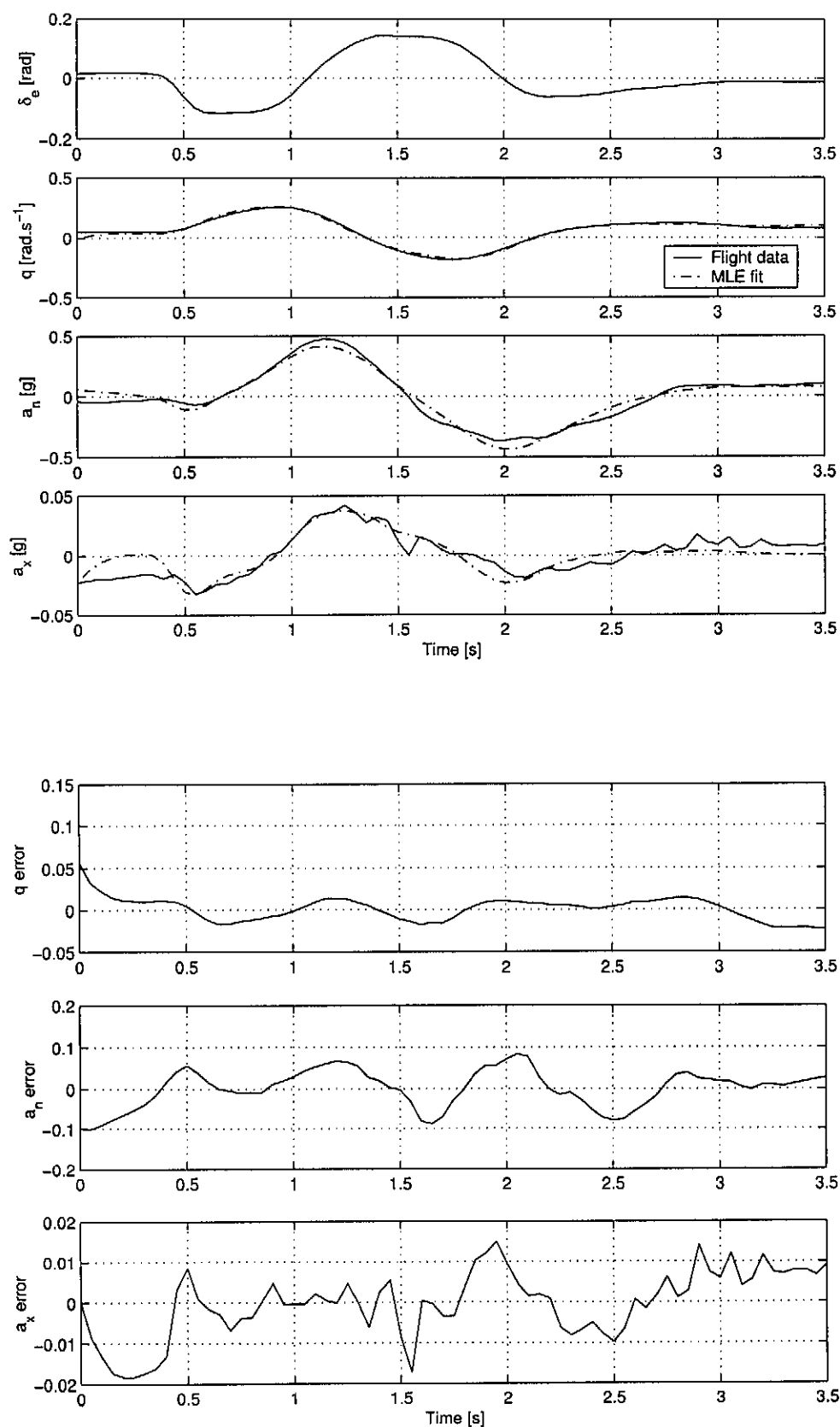


Figure B.2: *Manoeuvre no.2*

**Figure B.3:** *Manoeuvre no.3*

**Figure B.4:** *Manoeuvre no.4*

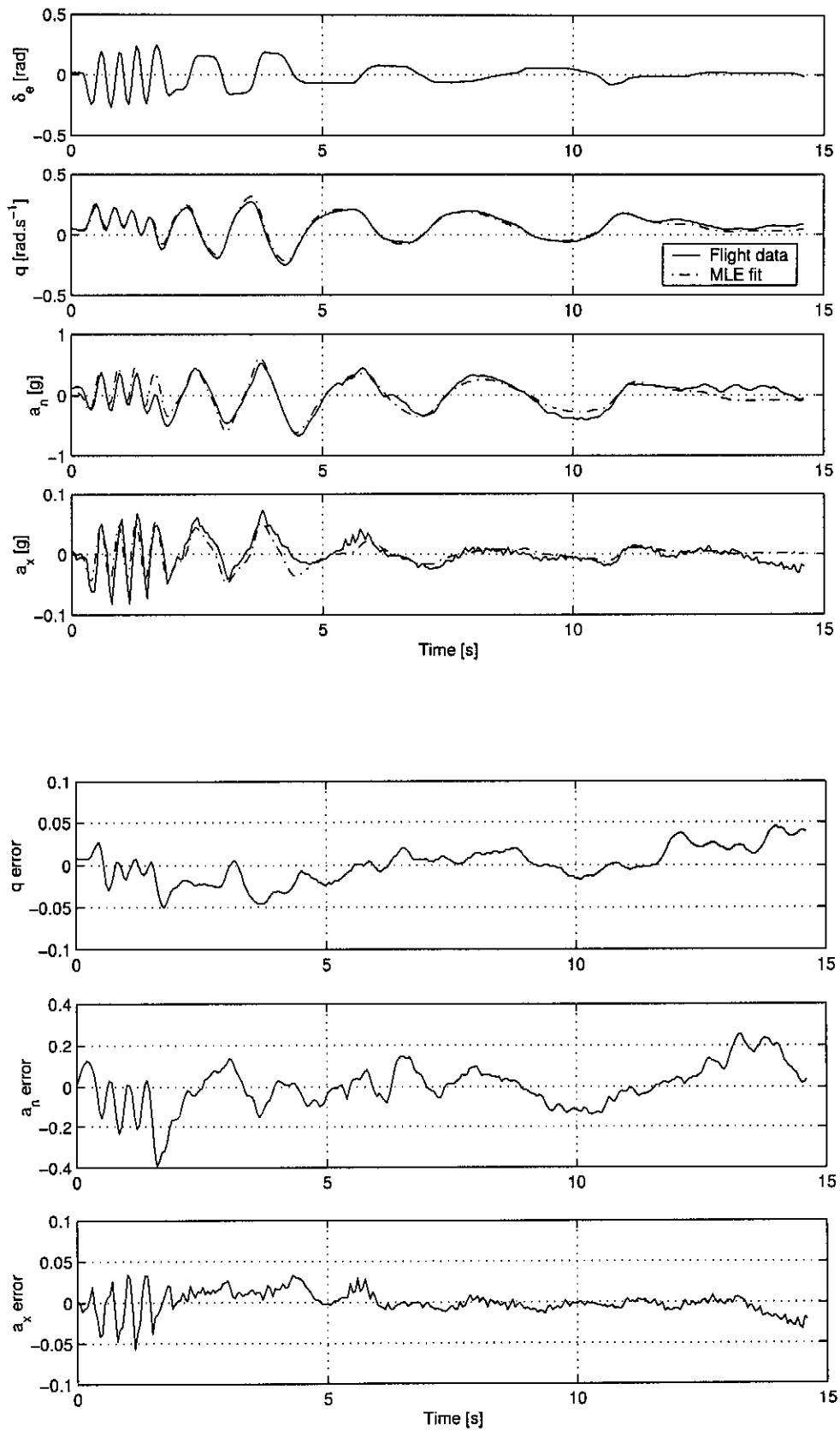


Figure B.5: *Manoeuvre no.5*

B.2 Flight 2

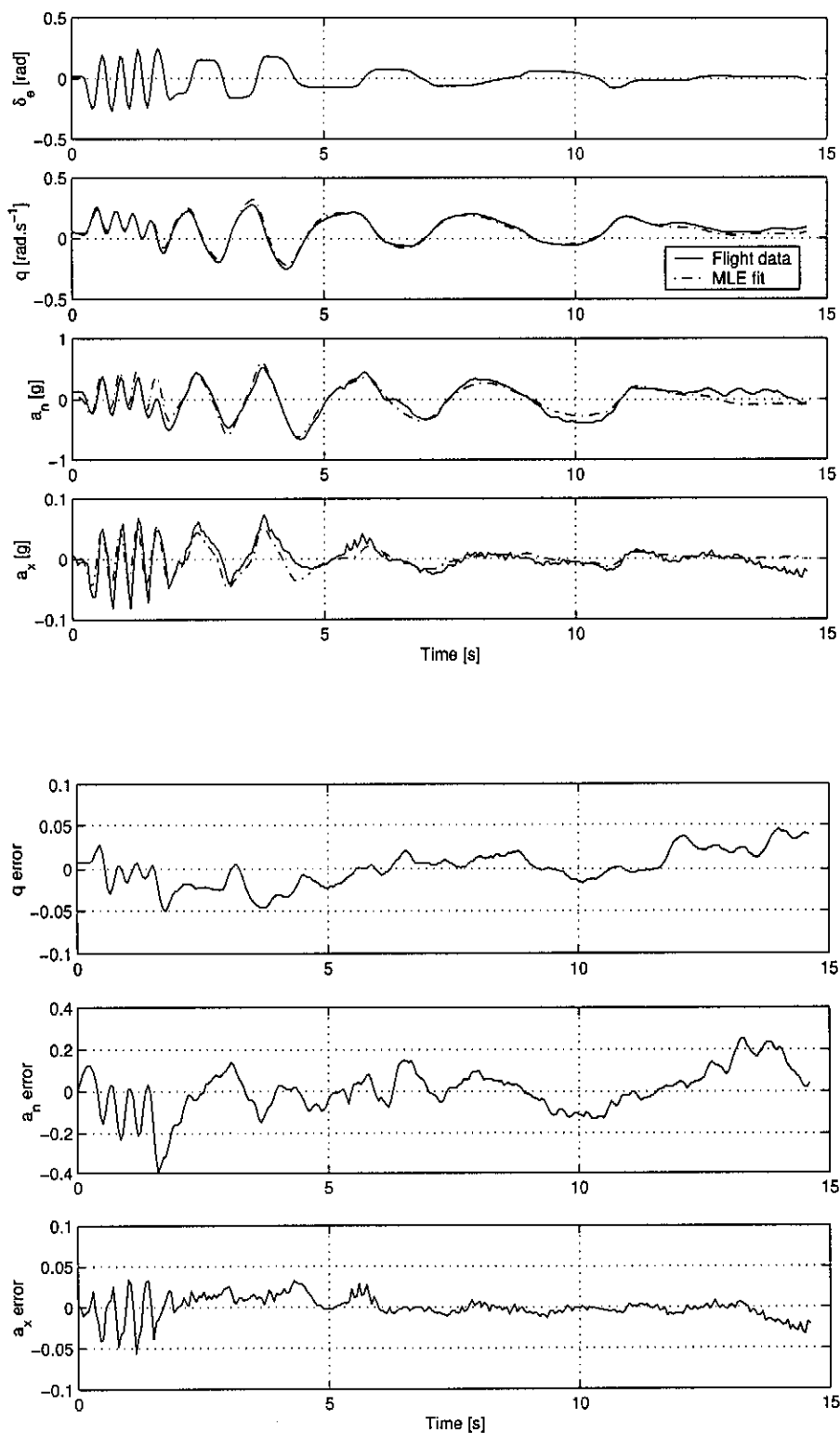


Figure B.6: Manoeuvre no.5

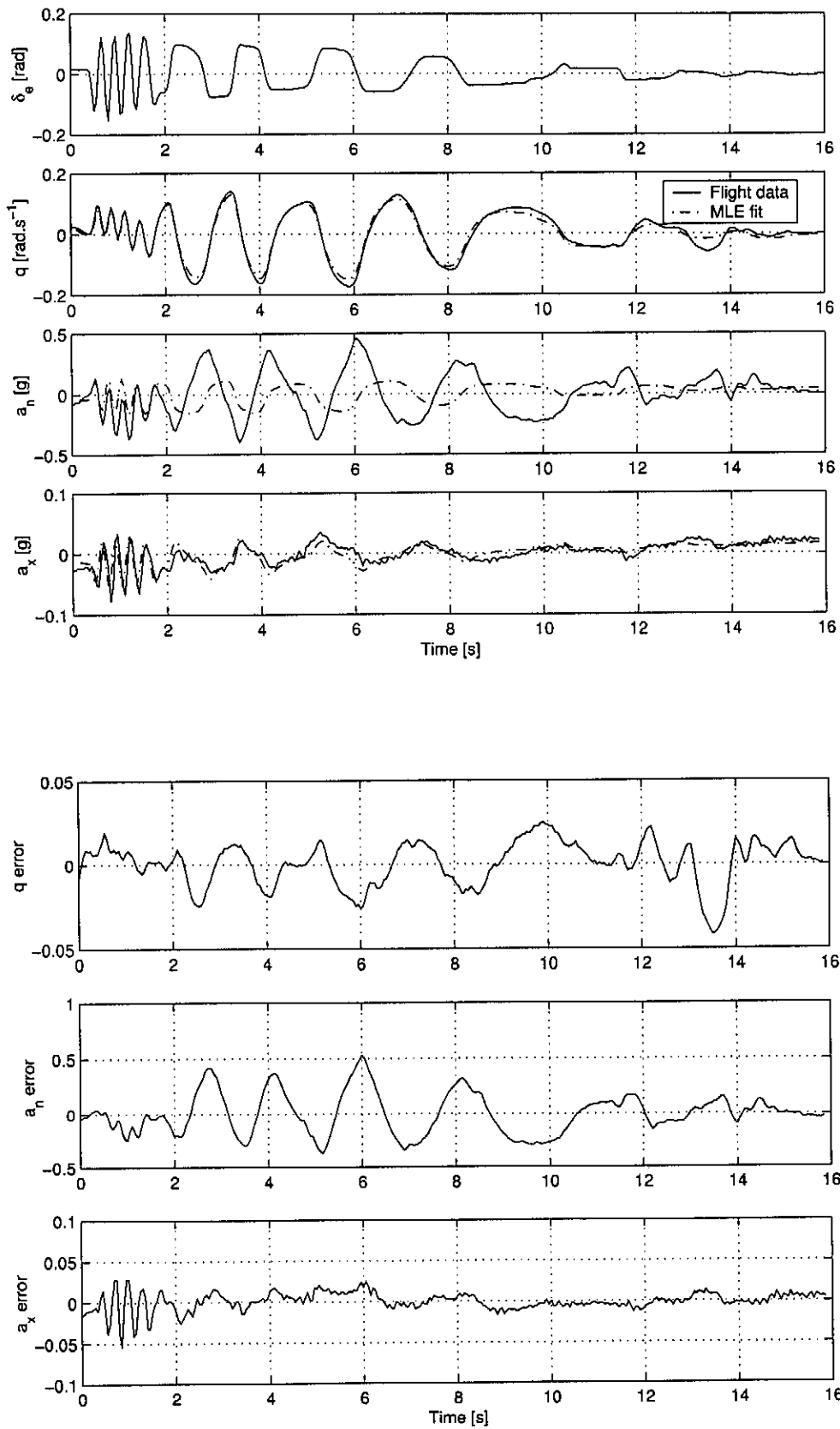


Figure B.7: Manoeuvre no.6

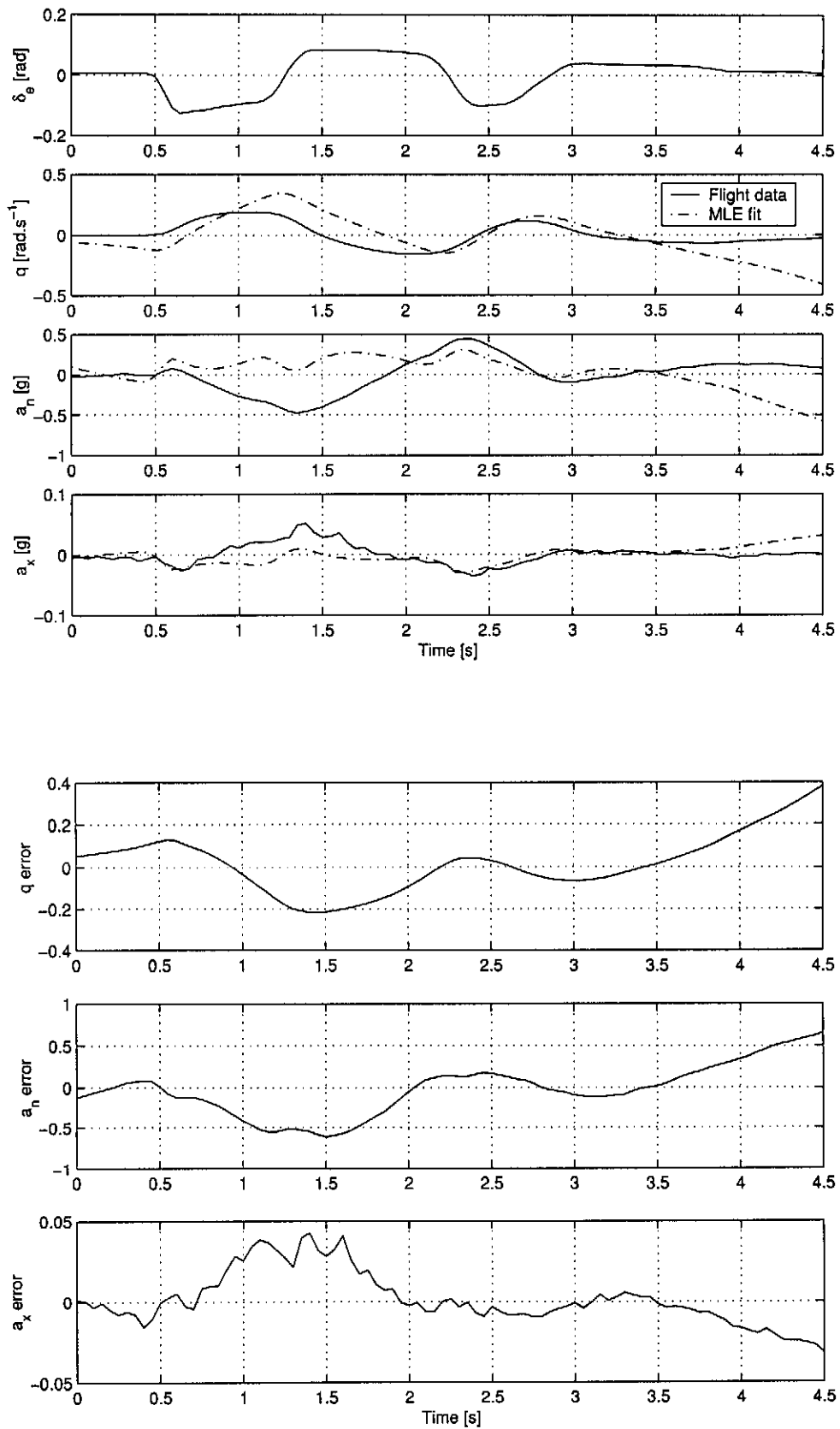


Figure B.8: Manoeuvre no.7

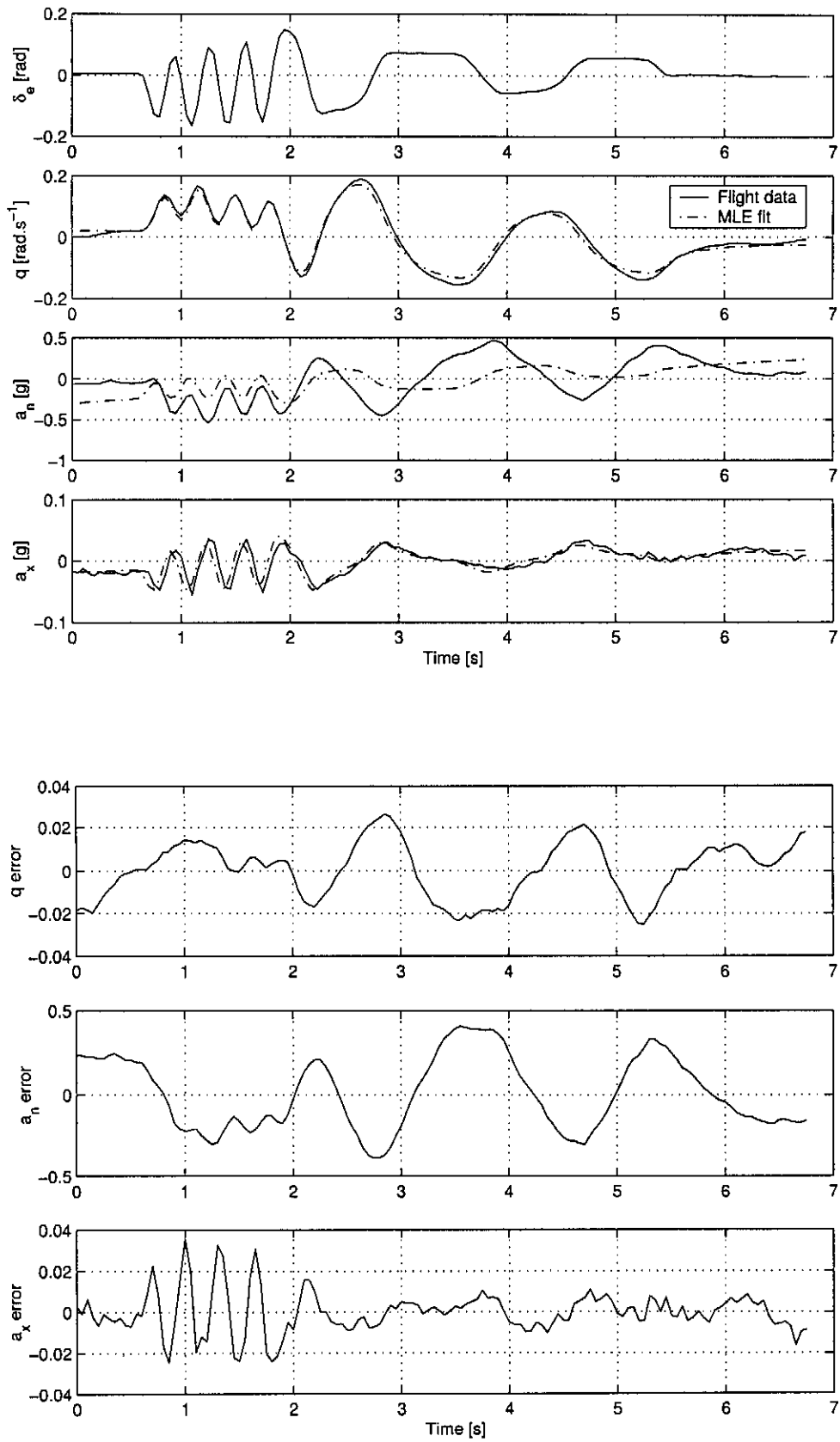


Figure B.9: *Manoeuvre no.8*

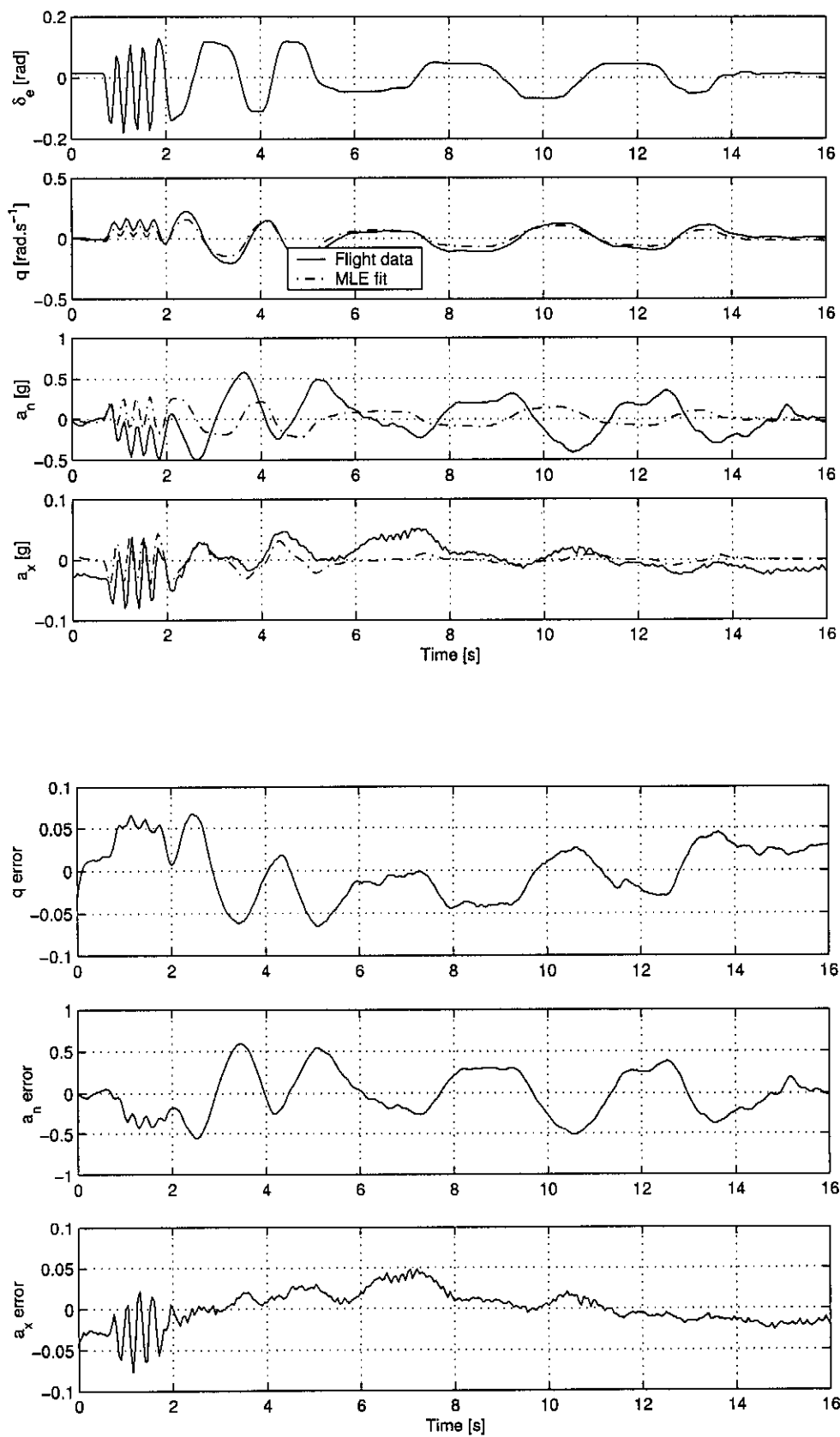
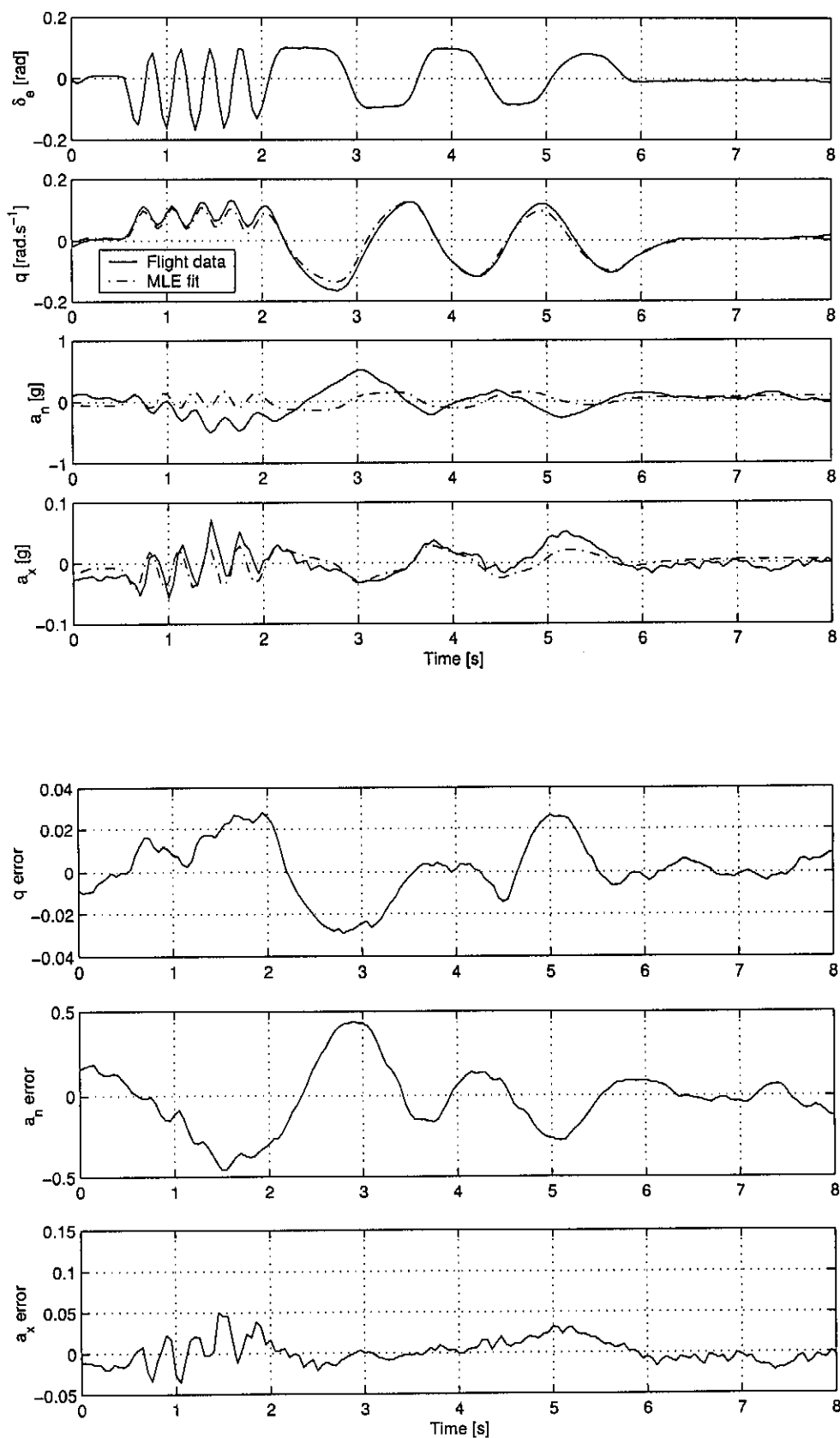


Figure B.10: Manoeuvre no.9

**Figure B.11:** Manoeuvre no.10

B.3 Flight 3

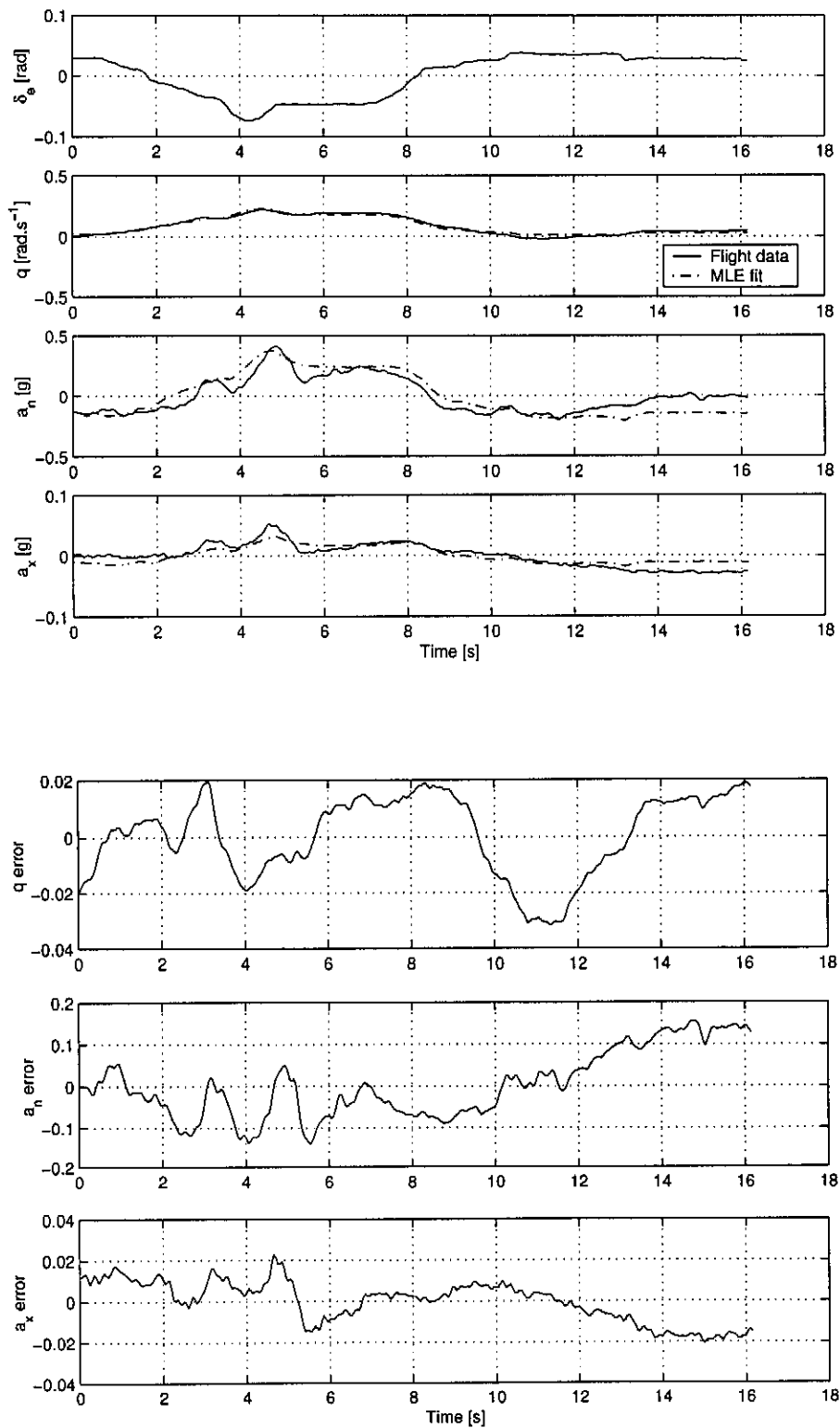


Figure B.12: Manoeuvre no.11

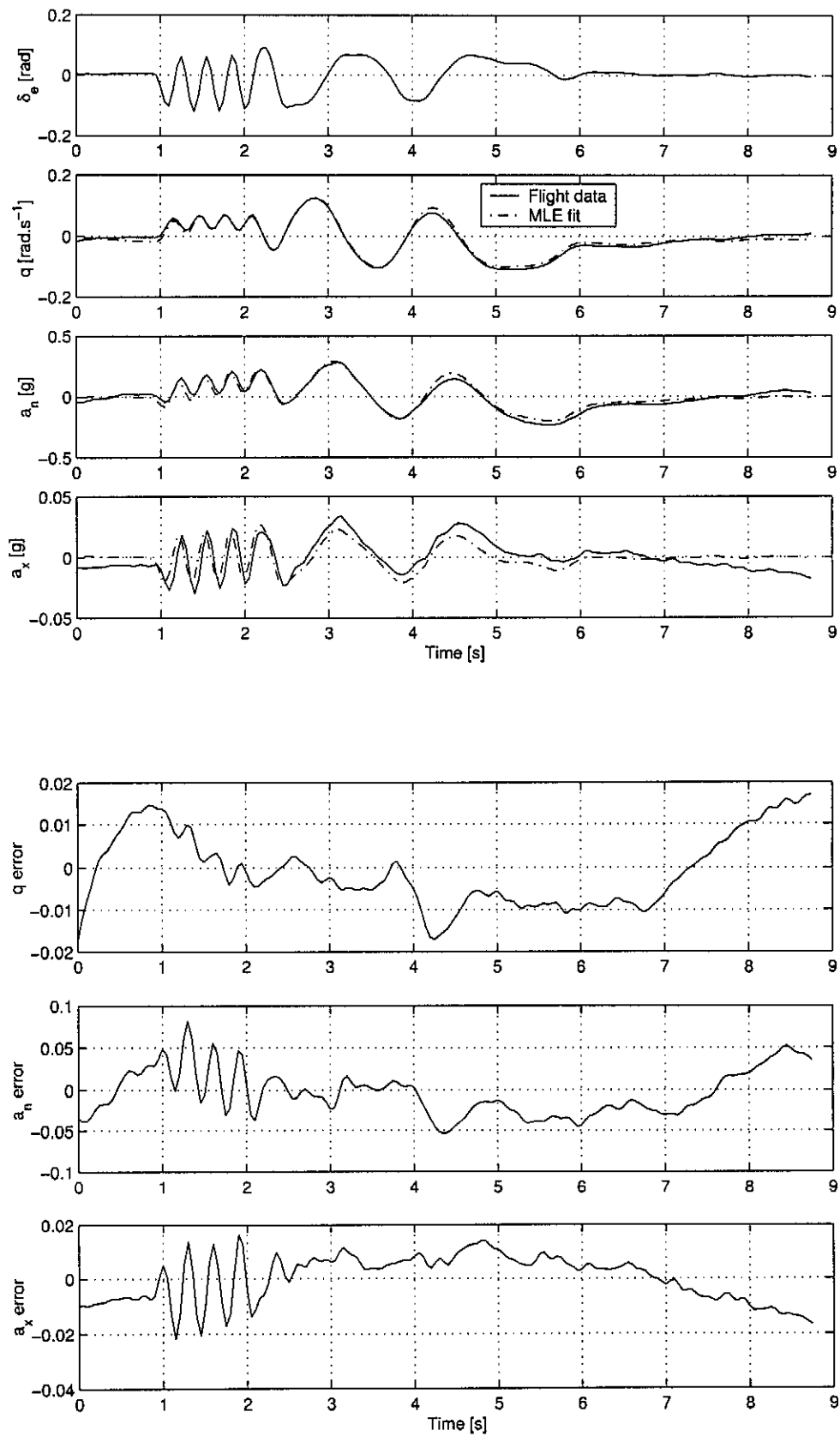


Figure B.13: *Manoeuvre no.12*

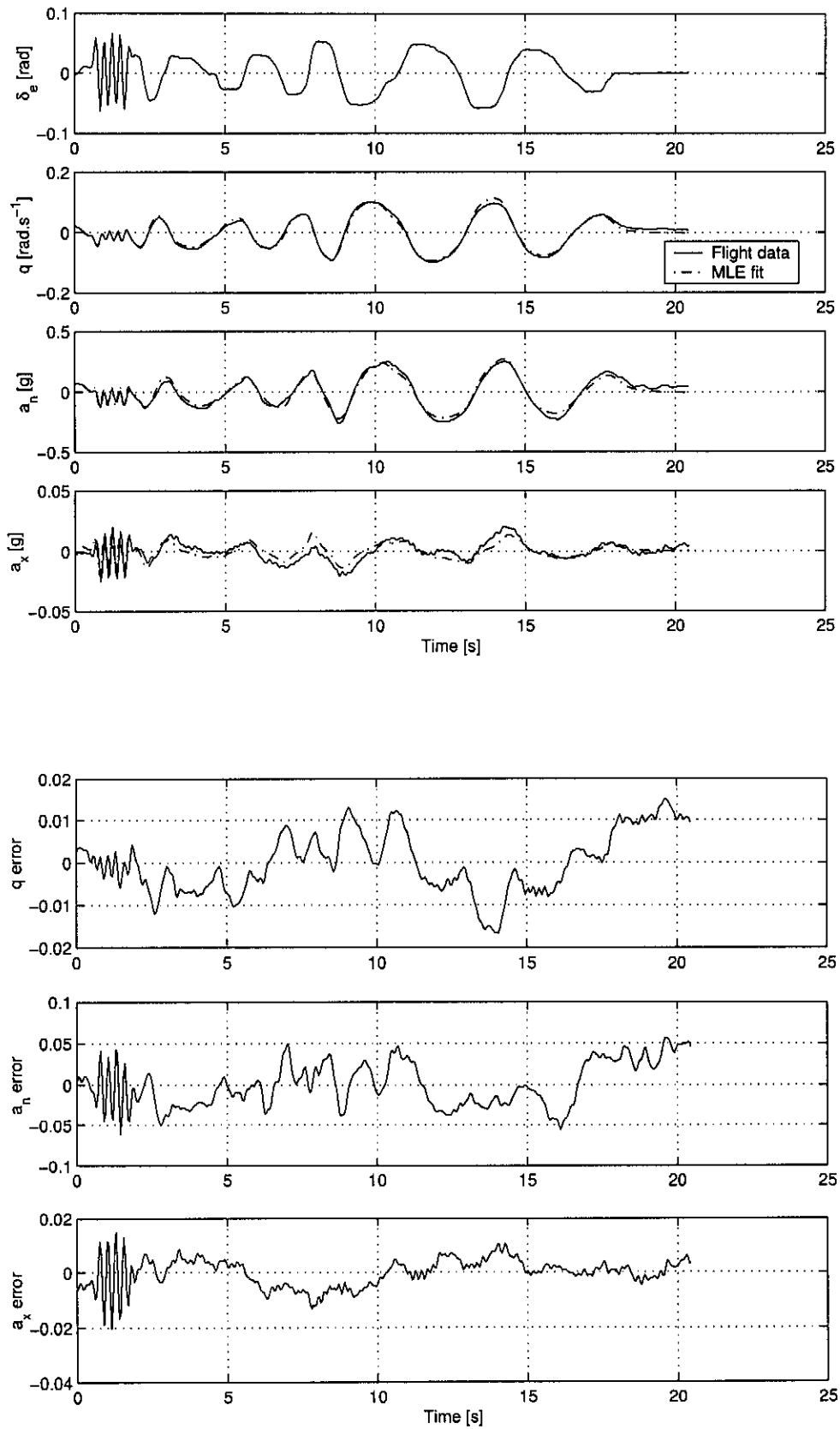


Figure B.14: *Manoeuvre no.13*

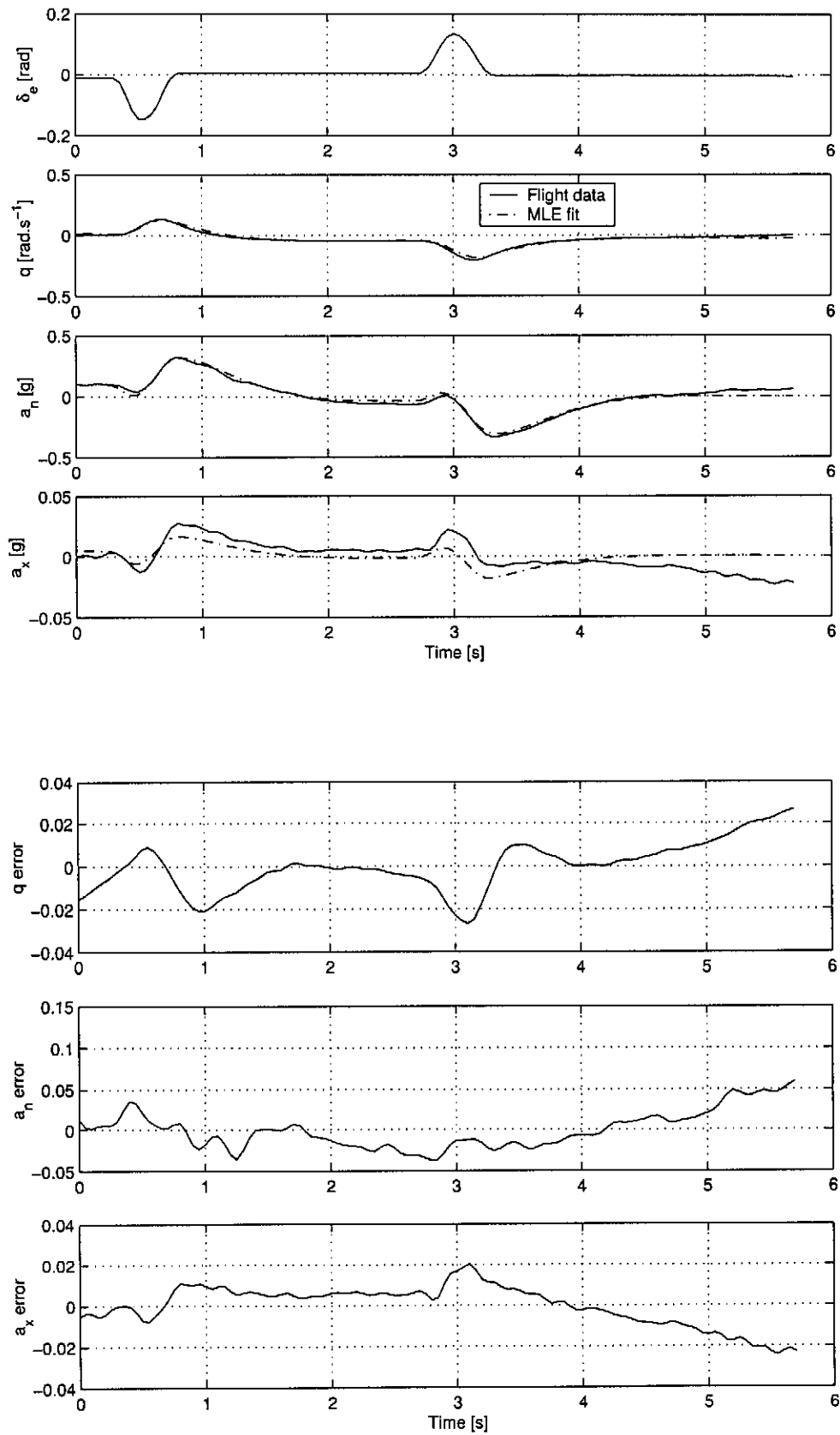


Figure B.15: Manoeuvre no.14

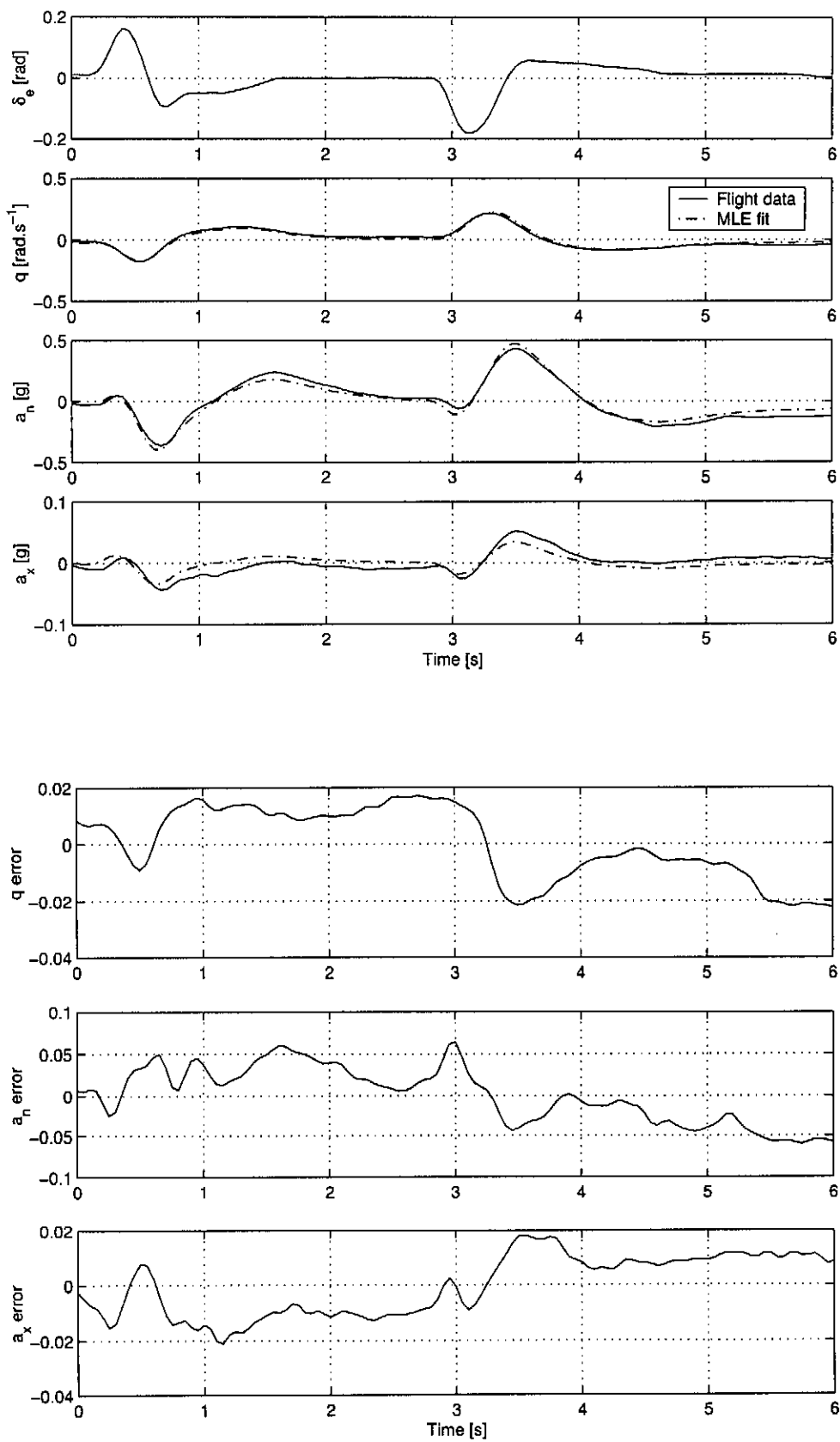


Figure B.16: Manoeuvre no.15

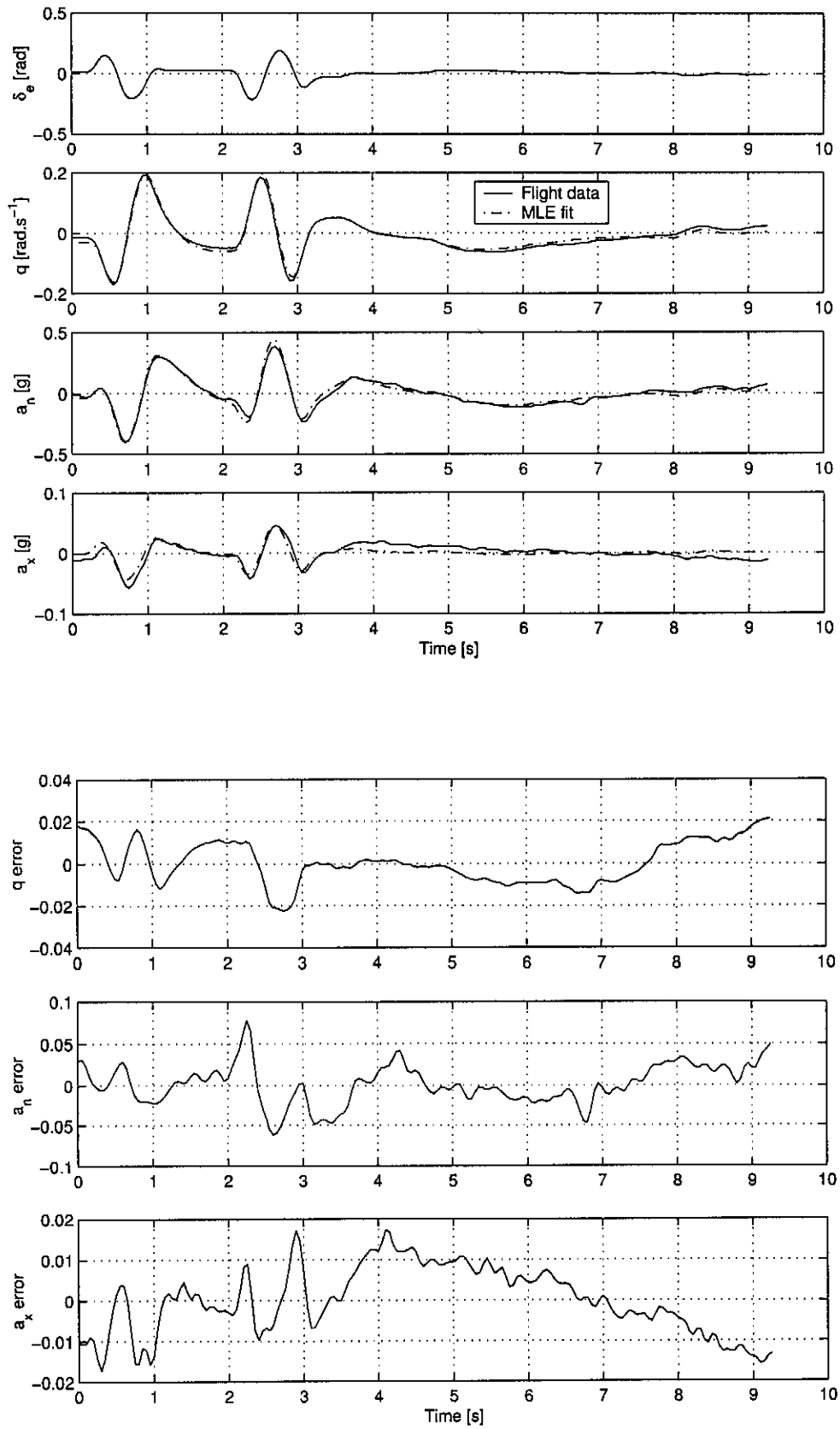


Figure B.17: Manoeuvre no.16

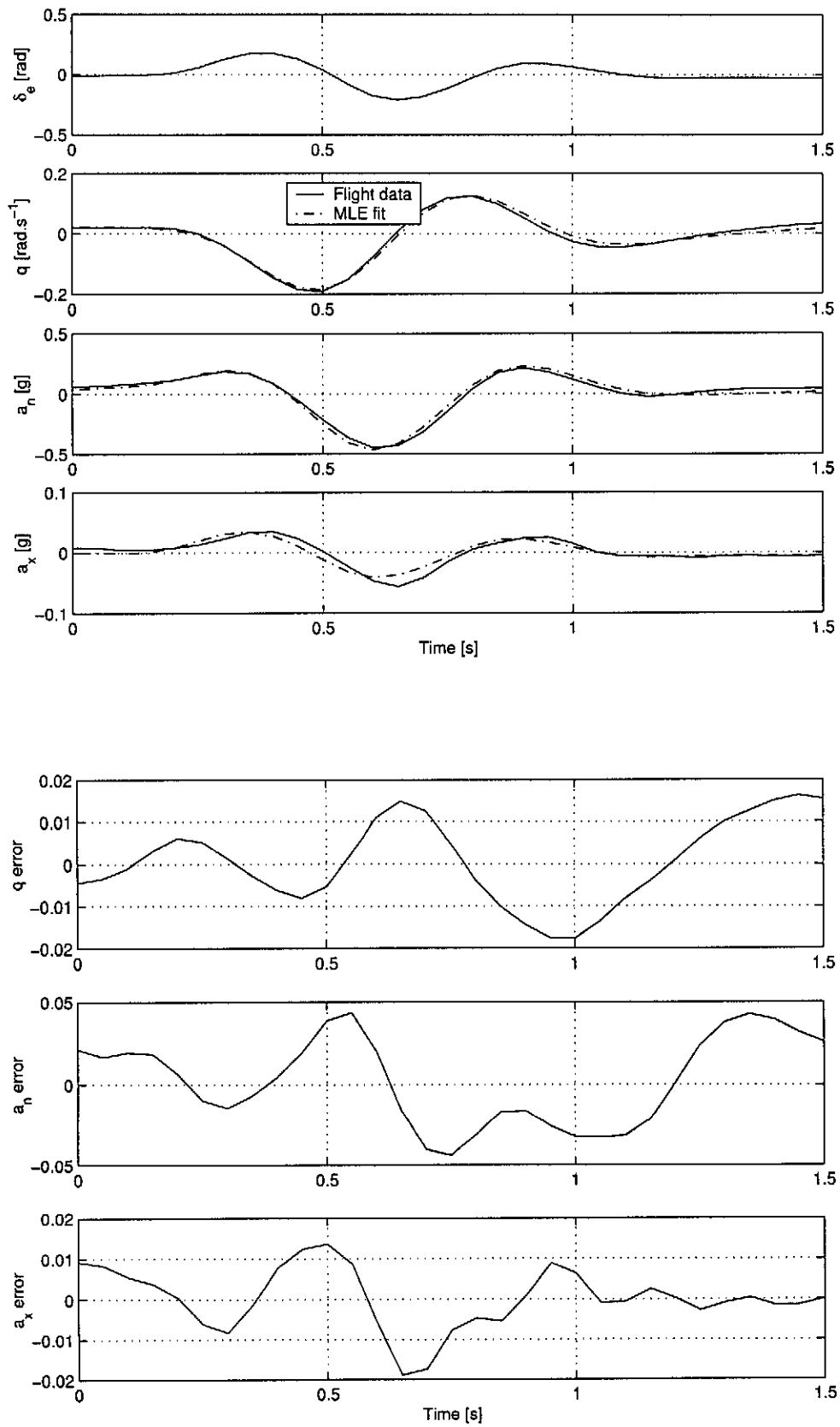
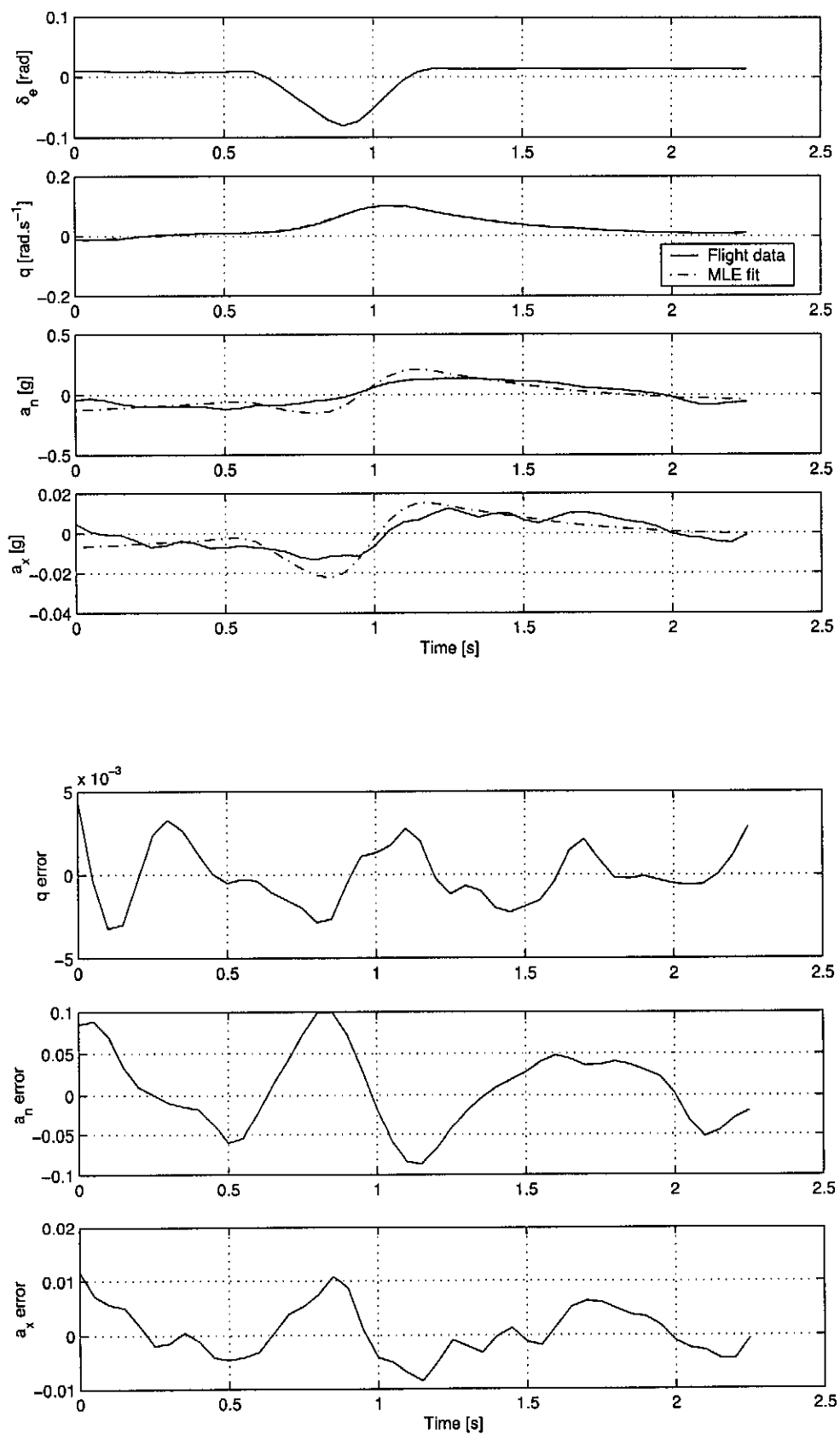


Figure B.18: Manoeuvre no.17

**Figure B.19:** Manoeuvre no.18

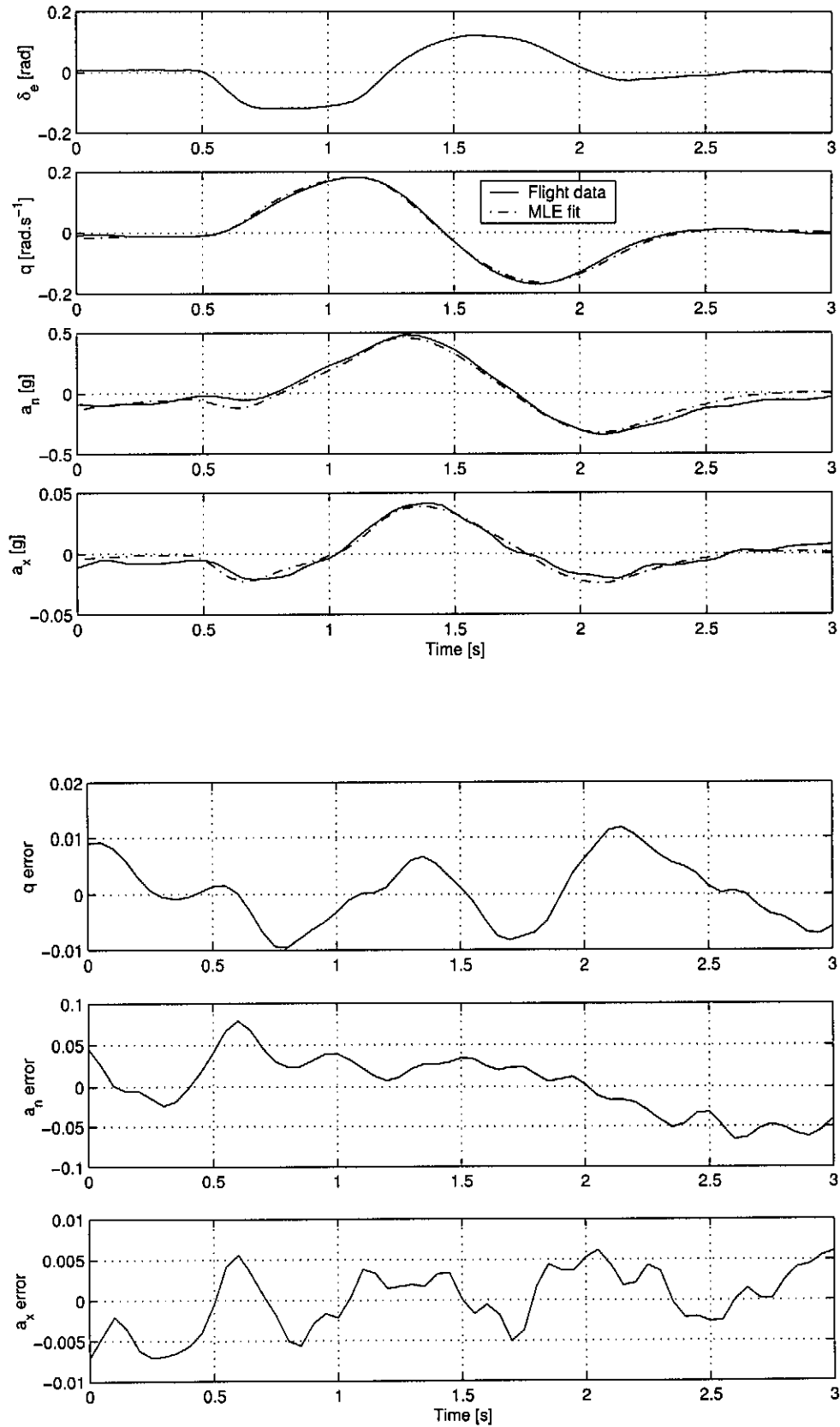


Figure B.20: Manoeuvre no.19

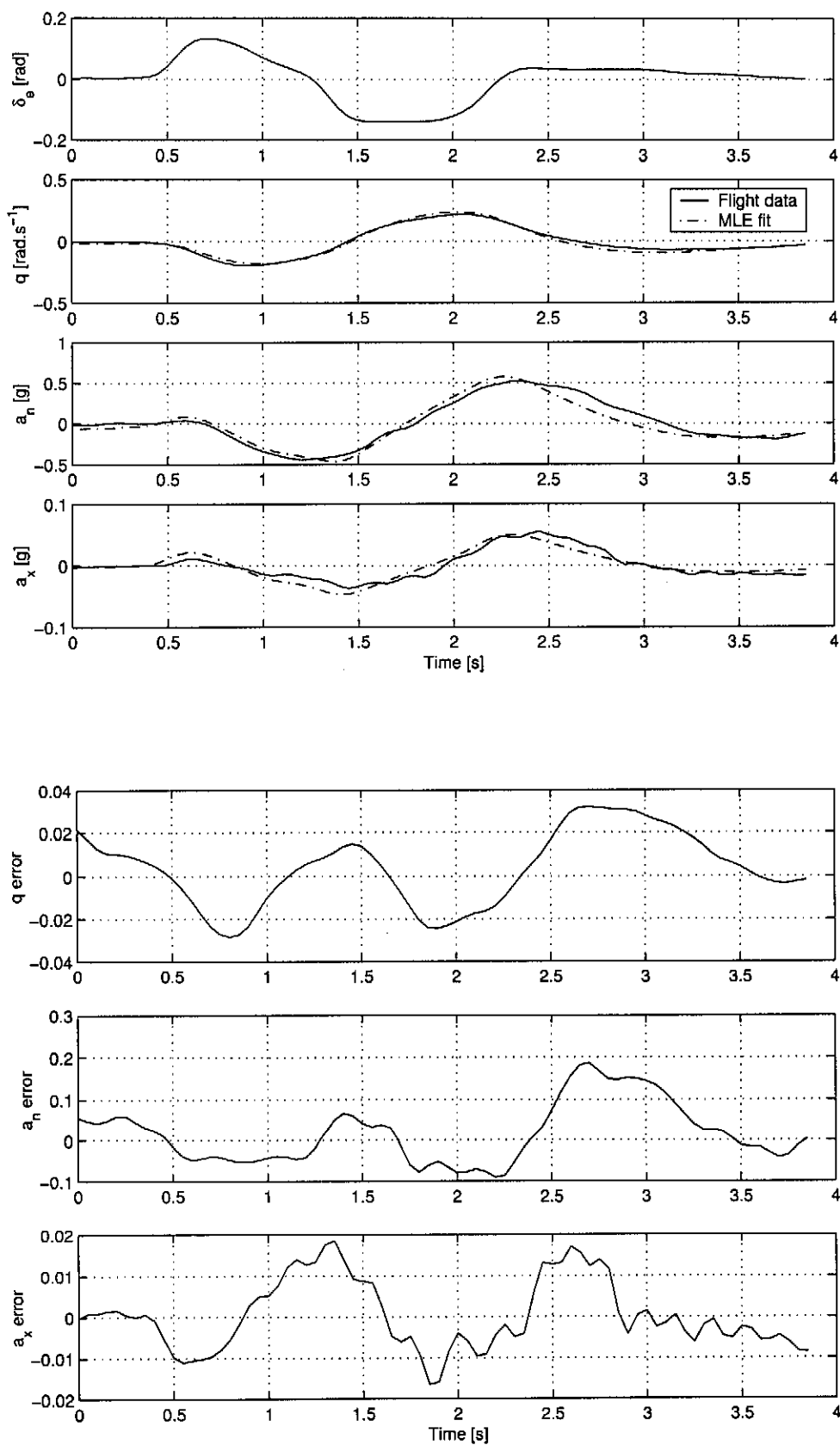
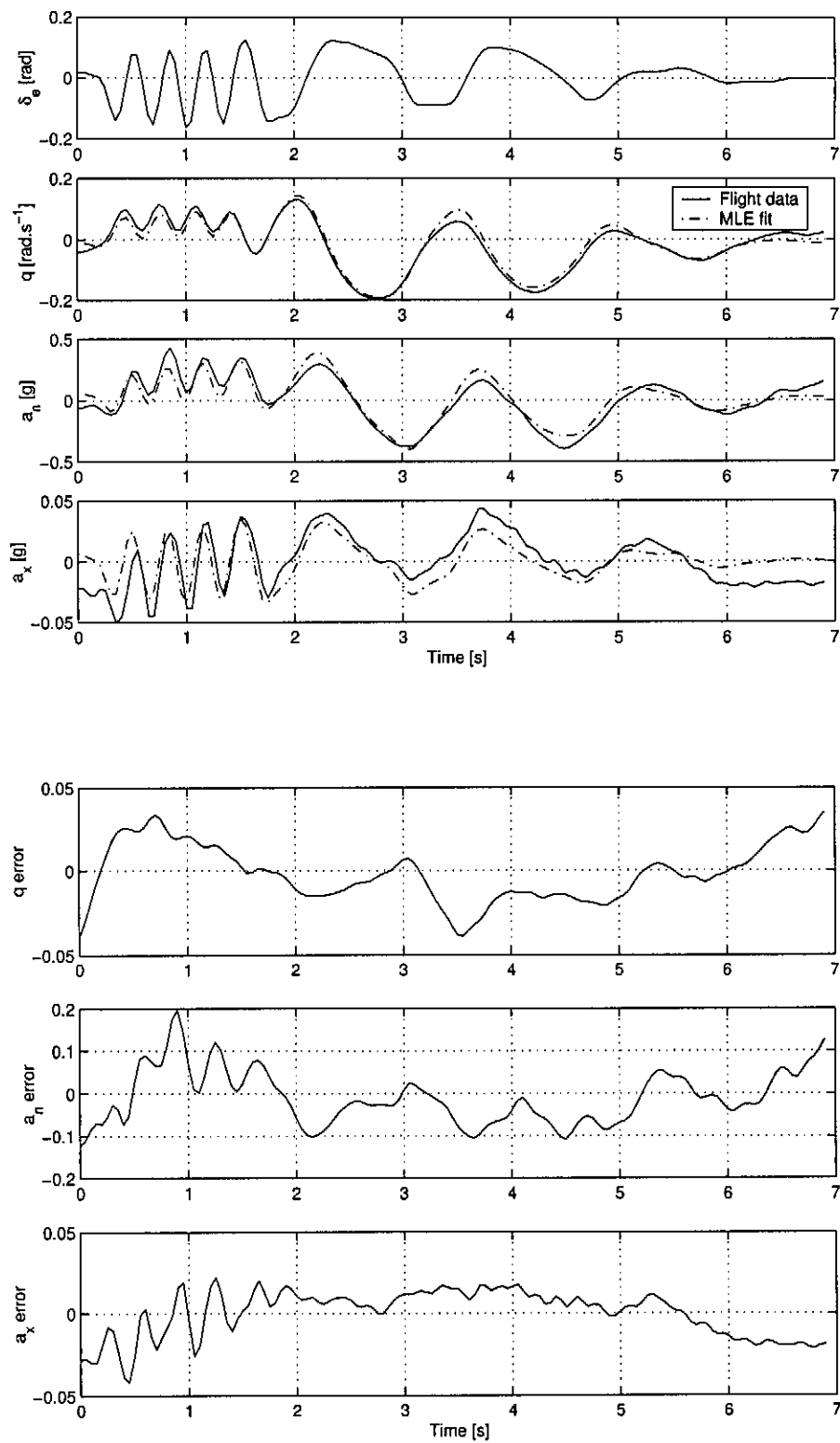


Figure B.21: Manoeuvre no.20

**Figure B.22:** *Manoeuvre no.21*

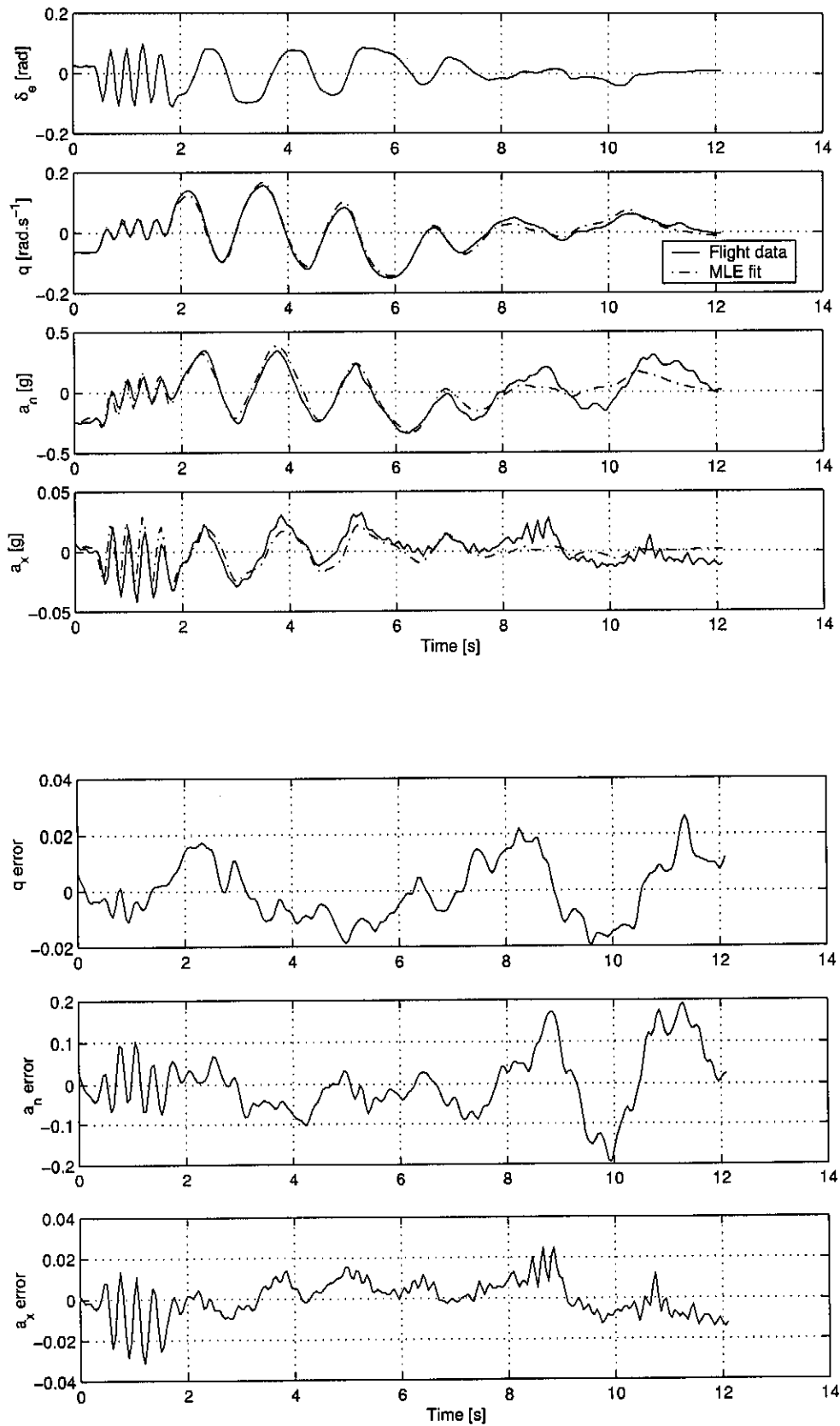


Figure B.23: *Manoeuvre no.22*

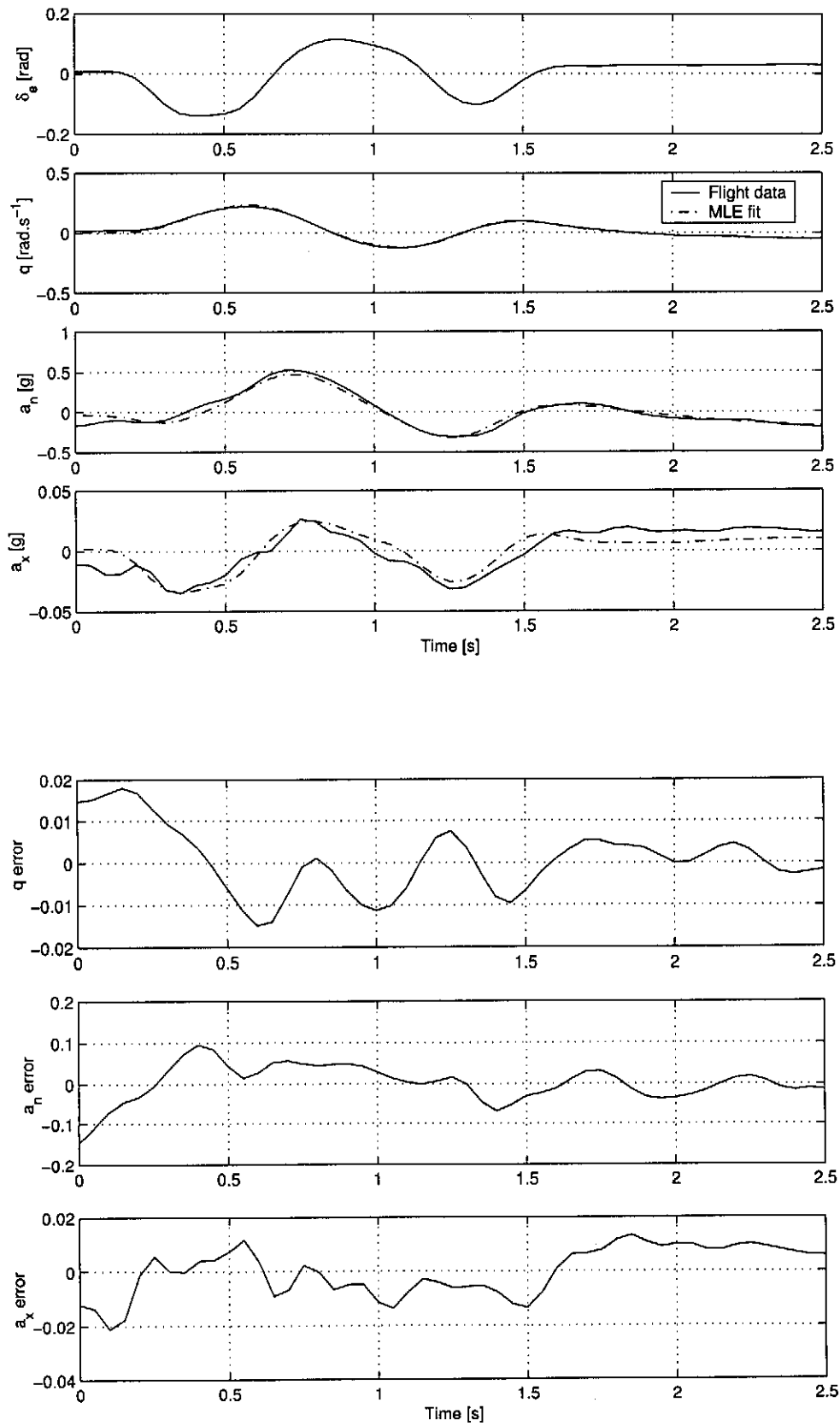
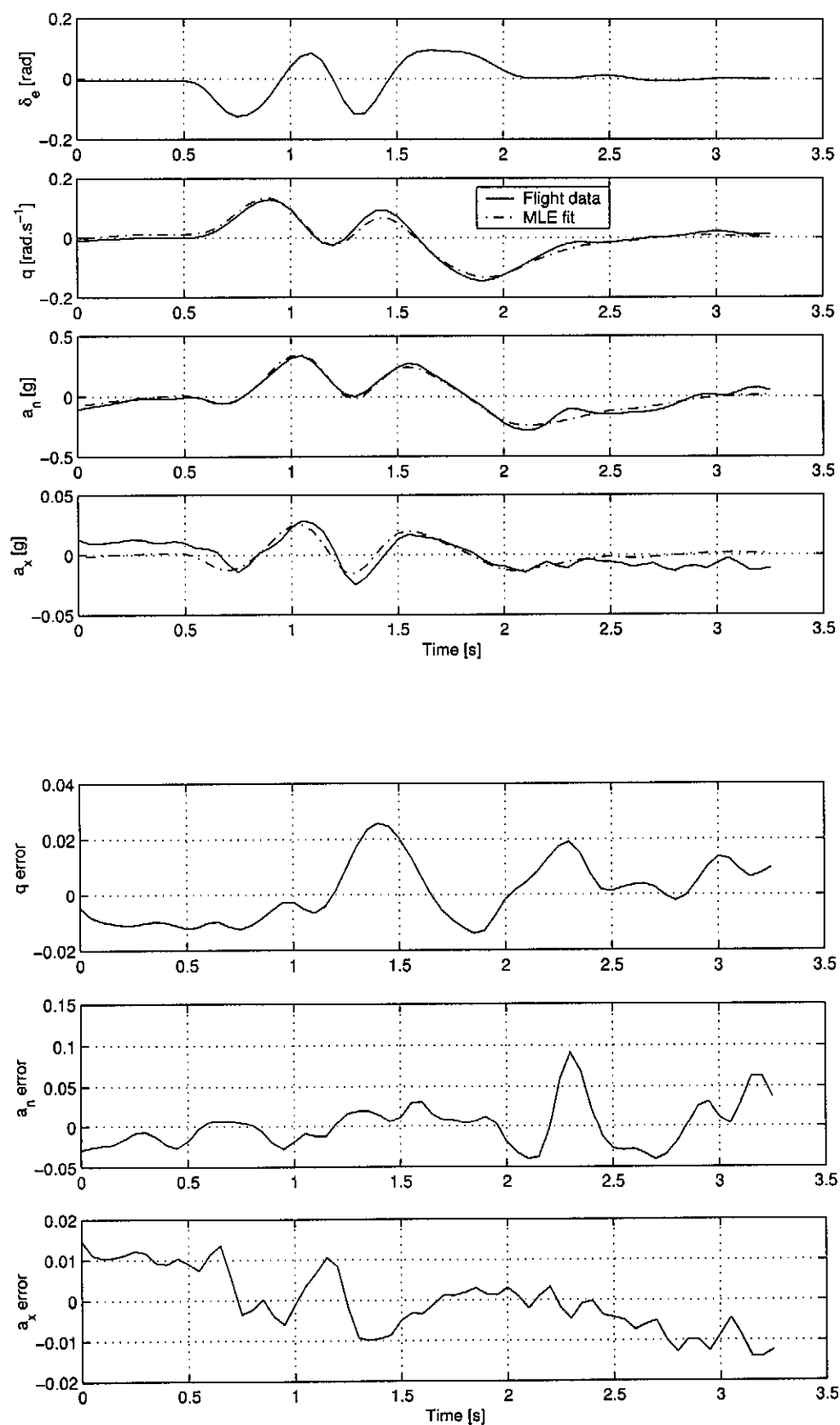


Figure B.24: Manoeuvre no.23

**Figure B.25: Manoeuvre no.24**

Appendix C

Circuit Schematics

The schematics of the circuits installed into the glider are printed on the follow pages. The circuit is split into four different schematics.

- **Signal Conditioning:** The signal conditioning circuit acts has two functions. To supply the sensors with electricity and to condition the incoming signals from the sensors.
- **Pressure Sensors:** The schematic details the circuit in the pressure sensor box and the instrumentation amplifier for the dynamic pressure sensor.
- **Inertial Measurement Unit:** The connections of the angular rate sensors are detailed as well as the circuitry for the accelerometers.
- **Control Deflection Sensors:** The connection of the control deflection potentiometers to the wiring harness. The wiring harness also connects the pressure sensors and power leads to the signal conditioning circuit.

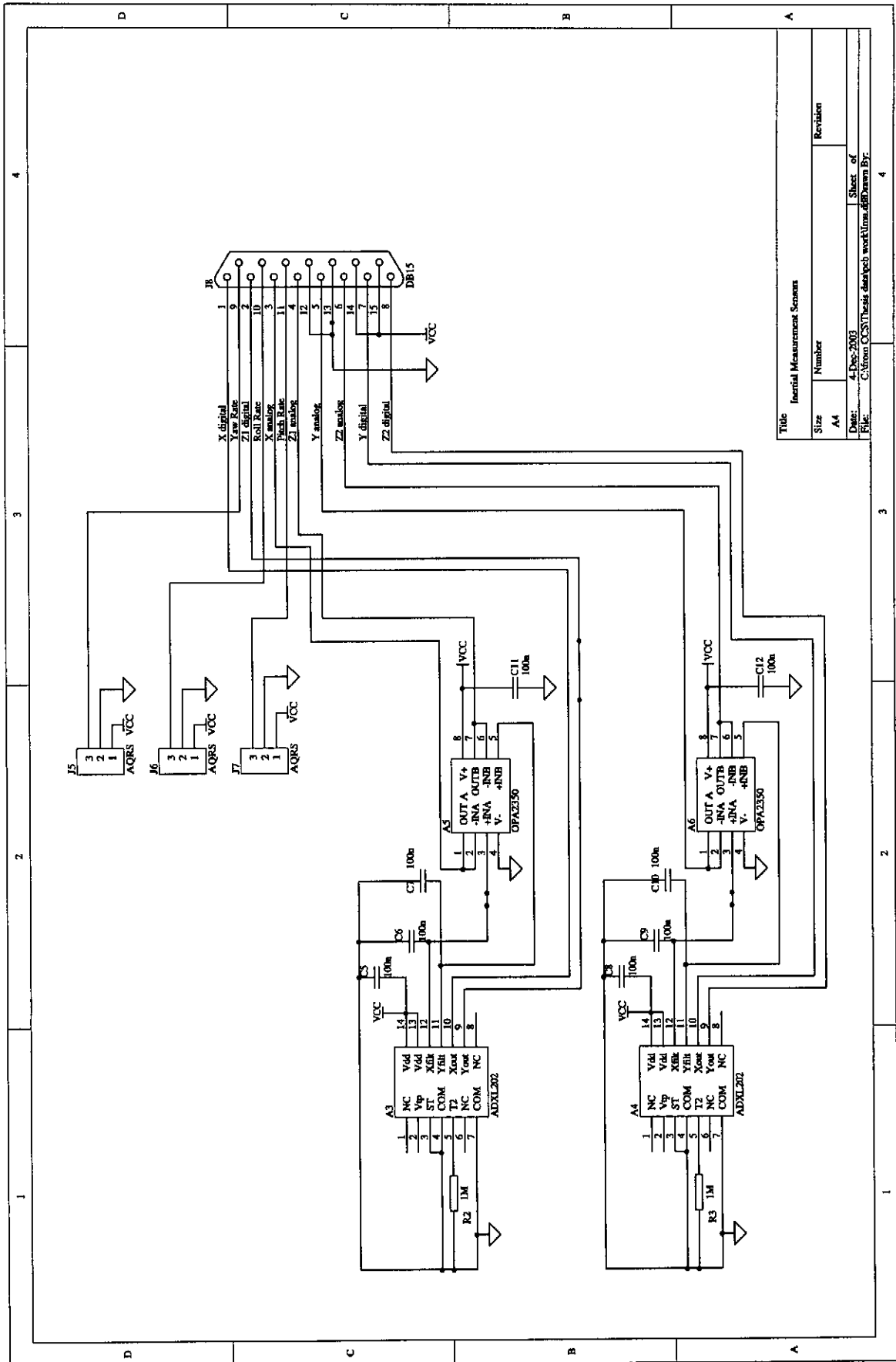


Figure C.3: Schematic for Inertial Measurement

Appendix D

Program Code

D.1 Delphi Code

Two programs were written in Delphi. The program used to capture the data from the serial port and store it on the hard drive and the program that converted the data from hexadecimal values to ADC codes. The Delphi compiler generates extra code to implement the graphical interfaces that does not contain any of the core functionality of the software, for the sake of brevity the extra code is omitted.

D.1.1 Code listing of: Seriallog.pas

```
unit Seriallog;
{ Serial data storage for glider data gathering
  Author K Browne
  Notes: Uses VaComm6 component to access serial port
}

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, ExtCtrls, VaClasses, VaComm, MPlayer;

type
  TForm1 = class(TForm)
    VaComm1: TVaComm;
    MediaPlayer1: TMediaPlayer;
    Timer1: TTimer;
    Label1: TLabel;
    procedure RxMessage(Sender: TObject; Count: Integer);
    procedure CreateForm1(Sender: TObject);
    procedure CloseForm1(Sender: TObject; var Action: TCloseAction);
    procedure Timer1Timer(Sender: TObject);
    procedure FormKeyPress(Sender: TObject; var Key: Char);

  private
    { Private declarations }
    Myfile:Text;
    S2,S1:string;
  public
    { Public declarations }
  end;

var
  Form1: TForm1;
  FStream:Text;
  overwrite:Boolean;
  lineswritten:Integer;
```

```

implementation

($R *.dfm)

procedure TForm1.RxMessage(Sender: TObject; Count: Integer);
{ This procedure receives the data from the serial port component.
  It converts the data from an ascii string to hexadecimal numbers.
  Adds a timestamp to each packet received.
  Saves the packet on the hard disk.
  Intermittently saves the data to disk to preserve the FAT.
}

var
  x: byte;

begin
  S1 := S1 + VaComm1.ReadText; // Read data from serial port
                                // and store it in string S1
  if (Ord(S1[1]) = 255) then // test for the start word
  begin
    if (Ord(S1[2]) = 255) then
    begin
      if (Length(S1) >= 42) then // check that data is available
      begin // for all channels
        for x :=1 to 42 do S2 := S2 + IntToHex(Ord(S1[x]),2);
          // Convert to hex numbers and separate channels
          S2:=timetostr(time) + ':' + S2;
          writeln(FStream,S2);
          Delete(S1,1,42);
          if lineswritten = 30000 then // open and close file
          begin
            Append(FStream);
            lineswritten:= 0;
            end;
            lineswritten:=lineswritten+1;
          end;
          S2 := '';
        end
        else if (length(S1) > 1) then S1 := '';
      end
      else S1 := '';
    end;
  end;

procedure TForm1.CreateForm1(Sender: TObject);
{ Called when the program is run.
  Activates media player for audio warning that the program
  is active.
  Also handles all other initialization.
}
VAR
error : integer;
begin
  MediaPlayer1.Open;
  MediaPlayer1.Play;
  overwrite:=true;
  Timer1.Enabled:=true;
  lineswritten:=0;
end;

procedure TForm1.CloseForm1(Sender: TObject; var Action: TCloseAction);
{ Called upon closing of the Form.
  Closes serial port and files
}
begin
  VaComm1.Close;
  if overwrite then
    CloseFile(FStream);
end;

procedure TForm1.Timer1Timer(Sender: TObject);
{ Timer component called every 10 milli-seconds at the beginning of program.
  Its function is to ensure that all initialization occurs at the correct time.
  Functions include: opening the file, starting the mediaplayer
}
begin
  if Timer1.Tag = 0 then
  begin
    Timer1.Tag :=1;
    MediaPlayer1.Play
  end;
end;

```



```

    Timer1.Interval :=1000;
    if overwrite then
        begin
            assignfile(FStream,'flight.dat');
            {$i-}
            rewrite(FStream);
            {$i+}
            Labell.Caption:=' Started';
            Labell.Visible:=true;
            end
        else
            Labell.Visible:=true;

            end
        else
            begin
                MediaPlayer1.Close;
                Timer1.Enabled:=false;
            end
        end;
    end;
end;

procedure TForm1.FormKeyPress(Sender: TObject; var Key: Char);
{ Stops files from previous sessions from being overwritten
}
begin
    overwrite:=false;
    Labell.Visible:=true;
end;
end.

```

D.1.2 Code listing of: Postflight.pas

```

unit Postflight;
{ Post-process after the data has been stored
  Turns hex into ADCcodes and outputs them in a file.
  Also allows viewing of collected data with Graph.
  Author K Browne
  Date July 2002
}
interface

uses
    Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
    Dialogs, StdCtrls;

type
    TForm1 = class(TForm)
        Memol: TMemo;
        Edit1: TEdit;
        Button1: TButton;
        Label1: TLabel;
        Button2: TButton;
        Button3: TButton;
        Label2: TLabel;
        procedure Button1Click(Sender: TObject);
        procedure FormClose(Sender: TObject; var Action: TCloseAction);
        procedure Button3Click(Sender: TObject);
        procedure Display(Sender: TObject);
        procedure FormCreate(Sender: TObject);
    private
        { Private declarations }
        flight,
        postflight:Text;
    public
        { Public declarations }
    end;

var
    Form1: TForm1;

implementation

uses formGraph;

{$R *.dfm}

```

```

procedure TForm1.Button1Click(Sender: TObject);
{Takes file flight.dat and converts it to ADCcodes.
}

var
  Chann: array[1..20] of integer;
  i:byte;
  step,samplenum,DataStart,x,V,Code,check:integer;
  s,line,newline,channel,channelx,DataMissing:string;
  LineCount :Boolean;
begin
  assignfile(flight,'flight.dat');//Setting up files
  assignfile(postflight,Edit1.Text);
  {$I-}
  rewrite(postflight);
  reset(flight);
  {$I+}
  memol.Text:='Starting';
  samplenum := 0;
  LineCount:=false;
  check:=0;
  DataMissing:='Lines with missing data:'+#13#10;
  while not EOF(flight) do
    begin
      check:=check+1; // checking for missing data
      ReadLn(flight,line);
      if not (line='') then
        begin
          if LineCount=false then // finds how many points must be ignored
            begin
              DataStart:=Pos('F',Line)+4;
              LineCount :=true;
            end; //if

          samplenum := samplenum + 1;
          s:=#13#10+IntToStr(samplenum); //
          x:=DataStart;

          for i:=1 to 19 do
            begin
              x:=x+4;

              channel:= copy(line,x,4);
              channelx:= '$'+channel;
              if channelx ='$' then
                begin
                  channelx :='$0000';
                  DataMissing :=DataMissing+' '+inttostr(check)+' column '+inttostr(i)+#13#10;
                end; //if

              step:=Round(5000*(StrToInt(channelx))/1023);// Conversion
              Chann[i]:=step;
              s:=s+' '+ IntToStr(step);//newline;
            end; //end for
          Write(postflight,s);
          GraphForm.Series1.AddY(Chann[1]);
          GraphForm.Series1.AddY(Chann[1]);
          GraphForm.Series2.AddY(Chann[2]);
          GraphForm.Series3.AddY(Chann[3]);
          GraphForm.Series4.AddY(Chann[4]);
          GraphForm.Series5.AddY(Chann[5]);
          GraphForm.Series6.AddY(Chann[6]);
          GraphForm.Series7.AddY(Chann[7]);
          GraphForm.Series8.AddY(Chann[8]);
          GraphForm.Series9.AddY(Chann[9]);
          GraphForm.Series10.AddY(Chann[10]);
          GraphForm.Series11.AddY(Chann[11]);
          GraphForm.Series12.AddY(Chann[12]);
          GraphForm.Series13.AddY(Chann[13]);
          GraphForm.Series14.AddY(Chann[14]);
          GraphForm.Series15.AddY(Chann[15]);
          GraphForm.Series16.AddY(Chann[16]);
          GraphForm.Series17.AddY(Chann[17]);
          GraphForm.Series18.AddY(Chann[18]);
          GraphForm.Series19.AddY(Chann[19]);
          GraphForm.Series20.AddY(Chann[20]);
        end; //end if
      end; //end while
      memol.Text:='Finito'; // acknowledge completion
      memol.Text:=memol.Text+#13#10+DataMissing;
    end;
end;

```

```

        closefile(flight);
        closefile(postflight);
    end;

    procedure TForm1.Button3Click(Sender: TObject);
    begin
        Close;
    end;

    procedure TForm1.Display(Sender: TObject);
    {Displays the graph. }
    begin
        GraphForm.Show;
    end;

    procedure TForm1.FormCreate(Sender: TObject);
    {Initialization on program start}
    begin
        mem01.Text:='';
    end;

end.

```

D.2 Matlab Scripts

The Matlab scripts listed here from three different groups. The code is split into preprocessing, lateral motion estimation and longitudinal motion estimation.

D.2.1 Preprocessing

Script:newdata.m

```

% First step data manipulation
% m-file to facilitate the data capture needed for identification
% saves data in mat-file and adds extra information
% Can call filtering with user input.p

clc
%--- setup structure
%-----
FTdata.TestName = input('Test name ? ','s');
FTdata.date     = date;
FTdata.Time     = input('Time of Test ? ');
FTdata.Temp     = input('Air Temperature at time of test ? ');
FTdata.WindV    = input('Wind velocity ? ');
FTdata.FrontPMass = input('Front Pilot Mass ? ');
FTdata.RearPMass = input('Rear Pilot Mass ? ');
FTdata.Notes    = input('Any additional notes ? ','s');
FTdata.History  = 'Put counts in mat and add info';
FTdata.Data     = load(input('Flight Test datafile name ? ','s'));

%saving data in structure
%-----
save(FTdata.TestName,'FTdata');

% Calling the next step.
%-----
if input('Do other processing y/n? ','s')== 'y'
    datafilt(FTdata.TestName,'hamfilt10.mat',1,5); %filtering
    mansplit([FTdata.TestName 'flt']); % manoeuvre indexing
end

Function:datafilt.m

% This script filters all the data
%-----

function datafilt(testdata,filtermat,p,q);

```

```

if exist('FTdata')==0 % test if data is in memory
    load(testdata);
end

load(filtermat)
%--- load hamfilt10 10Hz 500 order hamming filter -6dB at 10Hz
%--- p=1;
%--- q=5; %Down sample from 100Hz to 20Hz
FTdata.Data=resample(FTdata.Data,p,q,Bfilt);% Downsampled and filtered

%--- Sorting data into channels
%-----
FTdata.PsrAltitude = FTdata.Data(:,2); % General data
FTdata.DiffPsr     = FTdata.Data(:,15);
FTdata.Strain      = FTdata.Data(:,12);
FTdata.Trim        = FTdata.Data(:,16);
FTdata.Airbrakes   = FTdata.Data(:,18);
FTdata.Pulse       = FTdata.Data(:,20);

FTdata.Lateral.Aileron = FTdata.Data(:,3); % lateral data
FTdata.Lateral.RollRate = FTdata.Data(:,14);
FTdata.Lateral.Rudder  = FTdata.Data(:,19);
FTdata.Lateral.YawRate = FTdata.Data(:,11);
FTdata.Lateral.Yaccel  = FTdata.Data(:,6);

FTdata.Longitudinal.Elavator = FTdata.Data(:,17); %longitudinal data
FTdata.Longitudinal.PitchRate = FTdata.Data(:,10);
FTdata.Longitudinal.Xaccel  = FTdata.Data(:,4);
FTdata.Longitudinal.Zaccel  = FTdata.Data(:,8);

FTdata=rmfield(FTdata,'Data'); %tidy up

FTdata.History=[FTdata.History 'Filtered with' filtermat ',channels sorted '];

%--- Save data
%-----
save([FTdata.TestName 'FLT'],'FTdata');
disp(' Data filtered');
return

Function:mansplit.m

% Load data and plot with zoom and grid user must then search
% all manoeuvres and enter the starting and end indexes
%-----
function mansplit(filename);
if exist('FTdata')==0
    load([filename]);
end

%--- Longitudinal Manoeuvres
%-----
disp(' Longitudinal manoeuvres:');
figure;
plot(FTdata.Longitudinal.Elavator );
zoom; grid; title('Elavator plot');

man=[];
mstart=1;
mend=0;
while mstart ~= mend
    mstart=input('Manoeuvre Start Index ');
    shg;
    mend =input('Manoeuvre End Index ');
    shg

    man=[man;[mstart mend]];
end
FTdata.Longitudinal.Manoeuvre = man;
clf

%--- Lateral Manoeuvres
%-----
plot(FTdata.Lateral.Aileron);
hold on;
plot(FTdata.Lateral.Rudder,'r');
%plot(FTdata.Pulse,'g');

```

```

legend('Aileron','Rudder','Pulse',0)
grid;zoom on;
%man={mstart mend};

if length(man)==1
    man=[];
else
    man=man(1:length(man)-1,:);
end

disp(' Lateral manoeuvres:');

mstart = 1;
mend = 0;
man=[];
while mstart ~= mend
    mstart=input('Manoeuvre Start Index ');
    shg;
    mend =input('Manoeuvre End Index ');
    shg;
    man=[man;{mstart mend}];
end

if length(man)==1
    man=[];
else
    man=man(1:length(man)-1,:);
end

FTdata.Lateral.Manoeuvre = man;

save(FTdata.TestName 'SPLT','FTdata');
disp('Manoeuvres saved')

if input('Plot Manoeuvres ? y/n ','s')== 'y'
    plotmans(FTdata.TestName 'SplT');
end
return

```

D.2.2 Lateral Estimation

Function: K13_P2SS_lat4.m

```

function [A,phi,gam,C,D,g,x0,dt,rowinq,B]=K13_P2SS_Lat4(par)
% P2SS function for ZS-GHB Lateral Motion
% Linearized Equations of Motion used from
% Application of parameter estimation to aircraft
% stability and control, Maine and Iliff
% -----
global S bb c...
    m...
    K1 K2 K3 K4 ...

% Unpack coefficients from p-vector
% -----
C_Y_beta = par(1);
C_Y_dr = par(2);
C_l_beta = par(3);
C_l_p = par(4);
C_l_r = par(5);
C_l_da = par(6);
C_l_dr = par(7);
C_n_beta = par(8);
C_n_p = par(9);
C_n_r = par(10);
C_n_da = par(11);
C_n_dr = par(12);
C_a_Sa = par(13);
C_a = par(14);
C_Y_Bb = par(15);
C_l_bias = par(16);
C_n_bias = par(17);
z_ay = par(18);
x_ay = par(19);
y_ay = par(20);

```

```

C_Y_bias = par(21);
Vel       = par(22);
dynPress  = par(23);
X0_1     = par(24);
X0_2     = par(25);
X0_3     = par(26);
X0_4     = par(27);
X0_5     = par(28);
Ixx      = par(29);
Iyy      = par(30);
Izz      = par(31);
Ixz      = par(32);
Proc_n1  = par(33);
Proc_n2  = par(34);
Proc_n3  = par(35);
Proc_n4  = par(36);
C1       = par(37);
C2       = par(38);

%Constants
%-----
dt = 0.05; %sample time
g = 9.81; %Gravitational acceleration

% Define State-Space matrices
% -----

%--- state equation P xdot = Q x + R u
P = [ 1, 0, 0, 0
      0, 1, 0, 0
      0, 0, Ixx, -Ixz
      0, 0, -Ixz, Izz ];

Q = [ K1*C_Y_beta, 0, C_a*Sa, -C_a
      0, 1, 0, 0
      K3*C_l_beta, 0, K3*C_l_p*bb/(2*Vel), K3*C_l_r*bb/(2*Vel)
      K3*C_n_beta, 0, K3*C_n_p*bb/(2*Vel), K3*C_n_r*bb/(2*Vel) ];

R = [ 0, K1*C_Y_dr, K1*C_Y_Bb, 1, 0, 0, 0, 0
      0, 0, 0, 0, 1, 0, 0, 0
      K3*C_l_da, K3*C_l_dr, K3*C_l_bias, 0, 0, 1, 0, 0
      K3*C_n_da, K3*C_n_dr, K3*C_n_bias, 0, 0, 0, 1, 0];

%--- convert into state equation xdot = A x + B u
Pinv = inv(P);
A = Pinv * Q;
B = Pinv * R;

%--- output equation y = C x + D u

C4 = [K4*C_Y_beta, 0, 0, 0] - z_ay/g*A(3,:) + x_ay/g*A(4,:);
D4 = {0, K4*C_Y_dr, K4*C_Y_bias, 0, 0, 0, 0 -y_ay/g} ...
      - z_ay/g*B(3,:) + x_ay/g*B(4,:);

C = [ 0, 0, C1, 0
      0, 0, 0, C2
      C4 ];

D = [ 0, 0, 0, 0, 0, 0, 0, 0
      0, 0, 0, 0, 0, 0, 0, 0
      D4 ];

% process noise parameters
% -----
q = [ Proc_n1, 0, 0, 0
      0, Proc_n2, 0, 0
      0, 0, Proc_n3, 0
      0, 0, 0, Proc_n4];

%--- rows in Q in which the parameter occur
rowinq=zeros(1,length(par));
rowinq(27:30)=[1 2 3 4];

```

```

% initial state [beta, phi, p, r]
% -----
x0 = [X0_1 X0_2 X0_3 X0_4];

% discretize
% -----

[phi,gam] = c2d_mm(A,B,dt); % C2D_MM == C2D

%-----

Script: oelatnewalt1123_1.m

% Script to setup and run the MML E algo
% Estimating the Lateral Motion derivatives.
% For the flight stored in '11_23_flight_1SPLT.mat'
% Single manoeuvre estimation
% -----
format compact,clc
clear

% Constants
% -----
global S C bb g m K1 K2 K3 K4
S = 17.5; % Wing surface area - m2
c = 1.09; % Mean Aerodynamic chord - m
bb = 16; % Wing span - m
g = 9.81; % Gravitational acceleration - ms-2
m = 400; % Mass kg
airdensity = 1.2; % [kg/m^3]
dt = 0.05; % downsampled datafilt.m ,data sampled at 20 hz

% name of P2SS
% -----
p2ssnam='K13_P2SS_Lat4'; % name of P2SS

% load data and calibrate
% -----
if exist('FTdata')== 0
    load('11_23_flight_1SPLT.mat');
    p_bias =452.25; % biases are for the first flight only
    q_bias =455.0531;
    r_bias =526.0582;
    man = FTdata.Lateral.Manoevre;
    Kp = 32.87/3.07*0.00488*pi/180; % Gyro scale factors
    Kq = 32.85/3.06*0.00488*pi/180;
    Kr = 32.92/3.15*0.00488*pi/180;

    p_all = (Kp * (FTdata.Lateral.RollRate - p_bias));
    q_all = (Kq * (FTdata.Longitudinal.PitchRate - q_bias));
    r_all = (Kr * (FTdata.Lateral.YawRate - r_bias));

    FTdata.DiffPsr=((1023-FTdata.DiffPsr)*555.5*0.00488/2.531);
    FTdata.Lateral.Yaccel=0.00488/2.3225*(FTdata.Lateral.Yaccel-574)*9.81/9.81;

    Aileron_all = -0.0012523*FTdata.Lateral.Aileron+ 0.62615;
    Rudder_all = 0.0014279*FTdata.Lateral.Rudder-0.74364;
end

ns=0;

% starting values for the parameters
% -----
C_Y_beta = 0.44742; %par(1);
C_Y_dr = 0.13371; %par(2);
C_l_beta = 0.021365; %par(3);
C_l_p = -0.20803; %par(4);
C_l_r = 0.080346; %par(5);
C_l_da = -0.059528; %par(6);
C_l_dr = -0.0056791; %par(7);
C_n_beta = -0.044277; %par(8);
C_n_p = -0.034271; %par(9);
C_n_r = -0.089348; %par(10);
C_n_da = 0.0075093; %par(11);
C_n_dr = -0.031594; %par(12);
C_a_Sa = 0.2328 ; %par(13);
C_a = -1; %par(14)
C_Y_Bb = -0.0015; %par(15);

```

```

C_l_bias = -0.0016;    %par(16);
C_n_bias = 0.0008;    %par(17);
z_ay     = -0.376;    %par(18);
x_ay     = -0.839;    %par(19);
y_ay     = 0          ; %par(20);
C_Y_bias = -0.0044;    %par(21);
Vel      = 22.2      ; %par(22);
DynPress = 12.7778;    %par(23);
X0_1     = -0.0065;    %par(24);
X0_2     = 0         ; %par(25);
X0_3     = 0.0196 ;    %par(26);
X0_4     = -0.0000;    %par(27);
X0_5     = 0.15      ; %par(28);
Ixx      = 962.36 ;    %par(29);
Iyy      = 3027.1 ;    %par(30);
Izz      = 3732.2 ;    %par(31);
Ixz      = -0.7955;    %par(32);
Proc_n1  = ns       ; %par(33);
Proc_n2  = ns       ; %par(34);
Proc_n3  = ns       ; %par(35);
Proc_n4  = ns       ; %par(36);
C1       = 1       ; %par(37);
C2       = 1       ; %par(38);

% extracting Manoeuvre from data
%-----
index=9;    % select manoeuvre
mstart = man(index,1);
mend   = man(index,2);

% make uydata
%-----
global uydata twofcramer wapriori

ndp=mend-mstart + 1; % number of data points

dynpress = (FTdata.DiffPar(mstart:mend));
V        = abs(sqrt(2.*dynpress/airdensity)); % using the starting velocity
pii      = p_all(mstart:mend);
qi       = q_all(mstart:mend);
ri       = r_all(mstart:mend);
Aileron  = Aileron_all(mstart:mend);
Rudder   = Rudder_all(mstart:mend);
Yaccel   = FTdata.Lateral.Yaccel(mstart:mend);
Yaccel   = Yaccel -mean(Yaccel);

% Manoeuvre specific constants
K1 = dynpress(1)*S/(m*mean(V));
K2 = g/mean(V);
K3 = mean(dynpress)*S*bb;
K4 = mean(dynpress)*S/(m*g);
Vel=mean(V);

% Generating phi and theta
phil=zeros(ndp+1,1);
thetal=phil;
phil_0=X0_2;
thetal_0=X0_5;
phil(1)=phil_0;
thetal(1)=thetal_0;

for index=1:ndp
    theta_dot = qi.*cos(phil(index))-ri.*sin(phil(index));
    phi_dot   = pii + (qi.*sin(phil(index))+ri.*cos(phil(index))).*tan(thetal(index));
    phil     = phil(index) + phi_dot*dt;
    thetal   = thetal(index) + theta_dot*dt;
end

phil=(phil(1); phil(1:ndp-1))-mean(phil);

thetal={thetal(1);thetal(1:ndp-1)};

%-- creating extra inputs
bias      = ones(ndp,1);
u4       = K2.*sin(phil).*cos(thetal);
u5       = (ri.*sin(phil)+qi.*cos(phil)).*tan(thetal);

```



```

u6      = (qi.*ri*(Iyy-Izz)+pii.*Ixz);
u7      = (pii.*qi*(Ixx-Iyy)-qi.*ri.*Ixz);
u8      = (pii.^2-ri.^2);

u=[Aileron Rudder bias u4 u5 u6 u7 u8]; % all he inputs into system

%-- 20H delay correction
u(1:ndp-1)=0.5*(u(1:ndp-1)+u(2:ndp)); % Use mean of input over sample period

uydata=[u pii ri Yaccel]; %inputs and outputs

% Organising start parameters
%-----
pstart=[C_Y_beta, C_Y_dr, C_l_beta, C_l_p, C_l_r, C_l_da, C_l_dr, C_n_beta, C_n_p, C_n_r, C_n_da, ...
        C_n_dr, C_a_Sa, C_a, C_Y_Bb, C_l_bias, C_n_bias, z_ay, x_ay, y_ay, C_Y_bias,...
        Vel, DynPress, X0_1, X0_2, X0_3, X0_4, X0_5, Ixx, Iyy, Izz, Ixz,...
        Proc_n1, Proc_n2, Proc_n3, Proc_n4 C1 C2 A1 ];

% Parameters to be estimated
%-----
ap=[1 3 4 5 8 9 10];
abp=[29 31 32 ];
bp=[2 6 7 12 ];
cp=[18 19 20];
dp=[21];
x0p=[26 27];
N=[33 34 35 36];

%--- for estimating the parameters
% pidq =[15 16 17 21 x0p 14 ];
% pidm =[ap bp 15 16 17 x0p 21 13 11];
% pidf =[ap bp 11] ;

%--- for simulation of weighted mean system
pidq =[15 16 17 21 x0p 14 ];
pidm =[15 16 17 x0p 21 13 14];
pidf =[15 16 17 x0p 13 14] ;

% setup and run MMLE
%-----

p0 = pstart;
pert = 0.001;
pref=p0;
linesearch = 1;
opt=[1 60 10 10 0.000001 0.0001 0.001 0.5];
mmle

% Plot results
% -----
t=[0:ndp-1]*.05';
y=uydata(:,9:11);
x0=[X0_1 X0_2 X0_3 X0_4];
[A,phi,gam,C,D,q,x0,dt,rowinq,B]=K13_P2SS_Lat4(pfin);
yest=dlsim_mm(phi,gam,C,D,u,x0);

figure

subplot(414);
h=plot(t,[y(:,3) yest(:,3)]); set(h(2),'Col','r','linestyle','-.');
grid; ylabel('a_y [g] ');
xlabel('Time [s]');

subplot(413);
h=plot(t,[y(:,2) yest(:,2) ]); set(h(2),'Col','r','linestyle','-.');
grid;ylabel('r [rad.s^-1]');

subplot(411);
h=plot(t,[Aileron Rudder ]); set(h(2),'Col','r','linestyle','-.');
grid;ylabel('Input [rad]');
legend('\delta_a','\delta_r',4)

subplot(412);
h=plot(t,[y(:,1) yest(:,1)]); set(h(2),'Col','r','linestyle','-.');
grid; ylabel('p [rad.s^-1]');

```

```

legend('Flight data','MLE fit',4)

set(gcf, 'Papertype', 'A4');
set(gcf, 'PaperOrientation', 'portrait');
set(gcf, 'PaperUnits', 'inches')
set(gcf, 'InvertHardcopy', 'on')

inov=yest(:,1:3);
figure
subplot(311);

h=plot(t, [inov(:,3) ]);
%axis([0 max(t) -0.5 0.5]);
grid;

ylabel('Yaccel inovations');

subplot(312);

h=plot(t, [inov(:,2) ]);
%axis([0 max(t) -0.5 0.5]);
grid; ylabel('YawRate inovations');

subplot(313);

h=plot(t, [inov(:,1) ]);
%axis([0 max(t) -0.5 0.5]);
grid; ylabel('RollRate inovations');

set(gcf, 'Papertype', 'A4');
set(gcf, 'PaperOrientation', 'landscape');
set(gcf, 'PaperUnits', 'inches')

myfiguresize = [0.2500 0.2500 11.1929 7.7677];
set(gcf, 'PaperPosition', myfiguresize);

error=mean(abs(inov))

%-----
Scriptoelatall1g.m

% Script to setup and run the MMLE algo
% Estimating the Lateral Motion derivatives.
% For all The flights
% Stores results in user inpute file
%-----
format compact,cic
clear

% Constants
%-----
global S C bb g m K1 K2 K3 K4
S = 17.5; % Wing surface area - m2
c = 1.09; % Mean Aerodynamic chord - m
bb = 16; % Wing span - m
g = 9.81; % Gravitational acceleration - ms-2
m = 400; % Mass kg

% name of P2SS
%-----
p2snam='K13_P2SS_Lat4'; % name of P2SS
%---name of results file
ressave=input('filename for results : ','s');

%-----
% Start with the first flight
%-----
% load data and calibrate
%-----
load('11_23_flight_1SPLT.mat');
p_bias =452.25; % biases ar for the first flight only
q_bias =455.0531;
r_bias =526.0582;
man = FTdata.Lateral.Manceuvre;
Kp = 32.87/3.07*0.00488*pi/180; % Gyro scale factors
Kq = 32.85/3.06*0.00488*pi/180;
Kr = 32.92/3.15*0.00488*pi/180;

```

```

p_all = (Kp * (FTdata.Lateral.RollRate - p_bias));
q_all = (Kq * (FTdata.Longitudinal.PitchRate - q_bias));
r_all = (Kr * (FTdata.Lateral.YawRate - r_bias));

FTdata.DiffPsr=((1023-FTdata.DiffPsr)*555.5*0.00488/2.531);
FTdata.Lateral.Yaccel=0.00488/2.3225*(FTdata.Lateral.Yaccel-574)*9.81/9.81;

Aileron_all = -0.0012523*FTdata.Lateral.Aileron+ 0.62615;
Rudder_all = 0.0014279*FTdata.Lateral.Rudder-0.74364;

ns=0;

% starting values for the parameters
%-----
C_Y_beta = 0.44742; %par(1);
C_Y_dr = 0.13371; %par(2);
C_l_beta = 0.021365; %par(3);
C_l_p = -0.20803; %par(4);
C_l_r = 0.080346; %par(5);
C_l_da = -0.059528; %par(6);
C_l_dr = -0.0056791; %par(7);
C_n_beta = -0.044277; %par(8);
C_n_p = -0.034271; %par(9);
C_n_r = -0.089348; %par(10);
C_n_da = 0.0075093; %par(11);
C_n_dr = -0.031594; %par(12);
C_a_Sa = 0.2328 ; %par(13);
C_a = -1; %par(14);
C_Y_Bb = -0.0015; %par(15);
C_l_bias = -0.0016; %par(16);
C_n_bias = 0.0008; %par(17);
z_ay = -0.376; %par(18);
x_ay = -0.839; %par(19);
y_ay = 0 ; %par(20);
C_Y_bias = -0.0044; %par(21);
Vel = 22.2 ; %par(22);
DynPress = 12.7778; %par(23);
X0_1 = -0.0065; %par(24);
X0_2 = 0 ; %par(25);
X0_3 = 0.0196 ; %par(26);
X0_4 = -0.0000; %par(27);
X0_5 = 0.15 ; %par(28);
Ixx = 962.36 ; %par(29);
Iyy = 3027.1 ; %par(30);
Izz = 3732.2 ; %par(31);
Ixz = -0.7955; %par(32);
Proc_n1 = ns ; %par(33);
Proc_n2 = ns ; %par(34);
Proc_n3 = ns ; %par(35);
Proc_n4 = ns ; %par(36);
C1 = 1 ; %par(37);
C2 = 1 ; %par(38);

% extracting Manoeuvre from data
%-----
for manindex=1:length(man)
mstart = man(manindex,1);
mend = man(manindex,2);
ndp=mend-mstart + 1; % number of data points

% make uydata
%-----
global uydata twoframer wapriori

ndp=mend-mstart + 1; % number of data points

dynpress = (FTdata.DiffPsr(mstart:mend));
V = abs(sqrt(2.*dynpress/airdensity)); % using the starting velocity
pii = p_all(mstart:mend);
qi = q_all(mstart:mend);
ri = r_all(mstart:mend);
Aileron = Aileron_all(mstart:mend);
Rudder = Rudder_all(mstart:mend);
Yaccel = FTdata.Lateral.Yaccel(mstart:mend);
Yaccel = Yaccel -mean(Yaccel);

```

```

% Manoeuvre specific constants
K1 = dynpress(1)*S/(m*mean(V));
K2 = g/mean(V);
K3 = mean(dynpress)*S*bb;
K4 = mean(dynpress)*S/(m*g);
Vel=mean(V);

% Generating phi and theta
phil=zeros(ndp+1,1);
thetal=phil;
phil_0=X0_2;
thetal_0=X0_5;
phil(1)=phil_0;
thetal(1)=thetal_0;

for index=1:ndp
    theta_dot = qi.*cos(phil(index))-ri.*sin(phil(index));
    phi_dot   = pii + (qi.*sin(phil(index))+ri.*cos(phil(index))).*tan(thetal(index));
    phil      = phil(index) + phi_dot*dt;
    thetal    = thetal(index) + theta_dot*dt;
end

phil=[phil(1); phil(1:ndp-1)]-mean(phil);

thetal=[thetal(1);thetal(1:ndp-1)];

%-- creating extra inputs
bias      = ones(ndp,1);
u4        = K2.*sin(phil).*cos(thetal);
u5        = (ri.*sin(phil)+qi.*cos(phil)).*tan(thetal);
u6        = (qi.*ri*(Iyy-Izz)+pii.*Ixz);
u7        = (pii.*qi*(Ixx-Iyy)-qi.*ri*Ixz);
u8        = (pii.^2-ri.^2);

u=[Aileron Rudder bias u4 u5 u6 u7 u8]; % all he inputs into system

%-- ZOH delay correction
u(1:ndp-1)=0.5*(u(1:ndp-1)+u(2:ndp)); % Use mean of input over sample period

uydata=[u pii ri Yaccel]; %inputs and outputs

% Organising start parameters
%-----
pstart=[C_Y_beta, C_Y_dr, C_l_beta, C_l_p, C_l_r, C_l_da, C_l_dr, C_n_beta, C_n_p, C_n_r, C_n_da, ...
        C_n_dr, C_a_Sa, C_a, C_Y_Bb, C_l_bias, C_n_bias, z_ay, x_ay, y_ay, C_Y_bias,...
        Vel, DynPress, X0_1, X0_2, X0_3, X0_4, X0_5, Ixx, Iyy, Izz, Ixz,...
        Proc_n1, Proc_n2, Proc_n3, Proc_n4 C1 C2 A1 ];

% Parameters to be estimated
%-----
ap=[1 3 4 5 8 9 10];
abp=[29 31 32 ];
bp=[2 6 7 12 ];
cp=[18 19 20];
dp=[21];
x0p=[26 27];
N=[33 34 35 36];

%--- for estimating the parameters
% pidq =[15 16 17 21 x0p 14 ];
% pidm ={ap bp 15 16 17 x0p 21 13 11};
% pidf ={ap bp 11} ;

%--- for simulation of weighted mean system
pidq =[15 16 17 21 x0p 14 ];
pidm =[15 16 17 x0p 21 13 14];
pidf =[15 16 17 x0p 13 14] ;

% setup and run MMLE
%-----

p0 = pstart;
pert = 0.001;
pref=p0;
linesearch = 1;

```

```

opt=[1 60 10 10 0.000001 0.0001 0.001 0.5];
mmle

results1=results; % shuffle results around

% save results of this estimation run
%-----
    if manindex==1
        [r,c]=size(results1);
        results_all=zeros(r,c,length(man));
        pfinal = zeros(length(pfin'),length(man));

    end

    results_all(:, :,manindex)=results1;
    pfinal(:,manindex) = pfin';

end % end of for loop

% Save Results of the flight
%-----
    save(ressave,'results_all','pfinal');

%-----
% Start of the Second flight
%-----
% load data and calibrate
%-----

load('11_23_flight_2SPLT.mat');
p_bias =452.6810; % biases ar for the second flight only
q_bias =455.5346;
r_bias =526.2214;
man = FTdata.Lateral.Manoevre;

Kp = 32.87/3.07*0.00488*pi/180; % Gyro scale factors
Kq = 32.85/3.06*0.00488*pi/180;
Kr = 32.92/4.06*0.00488*pi/180;

p_all = (Kp * (FTdata.Lateral.RollRate - p_bias));
q_all = (Kq * (FTdata.Longitudinal.FitchRate - q_bias));
r_all = (Kr * (FTdata.Lateral.YawRate - r_bias));

FTdata.DiffPsr=(1023-FTdata.DiffPsr)*555.5*0.00488/2.531);
Ya_bias=460;
FTdata.Lateral.Yaccel=0.00488/2.3225*(FTdata.Lateral.Yaccel- Ya_bias)*9.81/9.81;
Aileron_all = -0.0012523*FTdata.Lateral.Aileron+ 0.62615;
Rudder_all = 0.0014279*FTdata.Lateral.Rudder-0.74364;

% extracting Manoeuvre from data
%-----
for manindex=1:length(man)
mstart = man(manindex,1);
mend = man(manindex,2);
ndp=mend-mstart + 1; % number of data points

% make uydata
%-----
global uydata twofcramer wapriori

ndp=mend-mstart + 1; % number of data points

dynpress = (FTdata.DiffPsr(mstart:mend));
V = abs(sqrt(2.*dynpress/airdensity)); % using the starting velocity
pil = p_all(mstart:mend);
qi = q_all(mstart:mend);
ri = r_all(mstart:mend);
Aileron = Aileron_all(mstart:mend);
Rudder = Rudder_all(mstart:mend);
Yaccel = FTdata.Lateral.Yaccel(mstart:mend);
Yaccel = Yaccel -mean(Yaccel);

% Manoeuvre specific constants

```

```

K1 = dynpress(1)*S/(m*mean(V));
K2 = g/mean(V);
K3 = mean(dynpress)*S*bb;
K4 = mean(dynpress)*S/(m*g);
Vel=mean(V);

% Generating phi and theta
phil=zeros(ndp+1,1);
thetal=phil;
phil_0=X0_2;
thetal_0=X0_5;
phil(1)=phil_0;
thetal(1)=thetal_0;

for index=1:ndp
    theta_dot = qi.*cos(phil(index))-ri.*sin(phil(index));
    phi_dot   = pii + (qi.*sin(phil(index))+ri.*cos(phil(index))).*tan(thetal(index));
    phil      = phil(index) + phi_dot*dt;
    thetal    = thetal(index) + theta_dot*dt;
end

phil=[phil(1); phil(1:ndp-1)]-mean(phil);

thetal=(thetal(1);thetal(1:ndp-1));

%-- creating extra inputs
bias      = ones(ndp,1);
u4        = K2.*sin(phil).*cos(thetal);
u5        = (ri.*sin(phil)+qi.*cos(phil)).*tan(thetal);
u6        = (qi.*ri*(Iyy-Izz)+pii.*Ixz);
u7        = (pii.*qi*(Ixx-Iyy)-qi.*ri.*Ixz);
u8        = (pii.^2-ri.^2);

u=[Aileron Rudder bias u4 u5 u6 u7 u8]; % all he inputs into system

%-- ZOH delay correction
u(1:ndp-1)=0.5*(u(1:ndp-1)+u(2:ndp)); % Use mean of input over sample period

uydata=[u pii ri Yacell]; %inputs and outputs

% Organising start parameters
%-----
pstart=[C_Y_beta, C_Y_dr, C_l_beta, C_l_p, C_l_r, C_l_da, C_l_dr, C_n_beta, C_n_p, C_n_r, C_n_da, ...
        C_n_dr, C_a_Sa, C_a, C_Y_Bb, C_l_bias, C_n_bias, z_ay, x_ay, y_ay, C_Y_bias,...
        Vel, DynPress, X0_1, X0_2, X0_3, X0_4, X0_5, Ixx, Iyy, Izz, Ixz,...
        Proc_n1, Proc_n2, Proc_n3, Proc_n4 C1 C2 A1 ];

% Parameters to be estimated
%-----
ap=[1 3 4 5 8 9 10];
abp=[29 31 32 ];
bp=[2 6 7 12 ];
cp=[18 19 20];
dp=[21];
x0p=[26 27];
N=[33 34 35 36];

%--- for estimating the parameters
% pidq = [15 16 17 21 x0p 14 ];
% pidm = [ap bp 15 16 17 x0p 21 13 11];
% pidf = [ap bp 11] ;

%--- for simulation of weighted mean system
pidq = [15 16 17 21 x0p 14 ];
pidm = [15 16 17 x0p 21 13 14];
pidf = [15 16 17 x0p 13 14] ;

% setup and run MMLE
%-----
p0 = pstart;
pert = 0.001;
pref=p0;
linesearch = 1;
opt=[1 60 10 10 0.000001 0.0001 0.001 0.5];

```

```

munle
    results1=results;

% Save Results
%-----
    if manindex==1
        depth=size(results_all,3);
        end
        results_all(:, :, manindex+depth)=results1;
        pfinal(:, manindex+depth)= pfin';

end % end of for loop

% Save Results of the second flight
%-----
    save(ressave, 'results_all', 'pfinal');

%=====
% Start of the Second flight
%=====
% load data and calibrate
%-----

    load('16_08_2003SPLT.mat');
    p_bias =453.5682; % biases ar for the third flight only
    q_bias =455.2555;
    r_bias =527.3622;
    man = FTdata.Lateral.Manoevre;
Kp = 32.87/3.07*0.00488*pi/180; % Gyro scale factors
Kq = 32.85/3.06*0.00488*pi/180;
Kr = 32.92/4.06*0.00488*pi/180;

p_all = (Kp * (FTdata.Lateral.RollRate - p_bias));
q_all = (Kq * (FTdata.Longitudinal.PitchRate - q_bias));
r_all = (Kr * (FTdata.Lateral.YawRate - r_bias));

    FTdata.DiffPsr=(1023-FTdata.DiffPsr)*555.5*0.00488/2.531;
    Ya_bias=479.9747;
    FTdata.Lateral.Yaccel=0.00488/2.3225*(FTdata.Lateral.Yaccel-479.974)*9.81/9.81

% extracting Manoeuvre from data
%-----
for manindex=1:length(man)
mstart = man(manindex,1);
mend = man(manindex,2);
ndp=mend-mstart + 1; % number of data points

% make uydata
%-----
global uydata twofcramer wapriori

ndp=mend-mstart + 1; % number of data points

dynpress = (FTdata.DiffPsr(mstart:mend));
V = abs(sqrt(2.*dynpress/airdensity)); % using the starting velocity
pii = p_all(mstart:mend);
qi = q_all(mstart:mend);
ri = r_all(mstart:mend);
Aileron = Aileron_all(mstart:mend);
Rudder = Rudder_all(mstart:mend);
Yaccel = FTdata.Lateral.Yaccel(mstart:mend);
Yaccel = Yaccel -mean(Yaccel);

% Manoeuvre specific constants
K1 = dynpress(1)*S/(m*mean(V));
K2 = g/mean(V);
K3 = mean(dynpress)*S*bb;
K4 = mean(dynpress)*S/(m*g);
Vel=mean(V);

% Generating phi and theta
phi1=zeros(ndp+1,1);
theta1=phi1;

```

```

phil_0=X0_2;
thetal_0=X0_5;
phil(1)=phil_0;
thetal(1)=thetal_0;

for index=1:ndp
    theta_dot = qi.*cos(phil(index))-ri.*sin(phil(index));
    phi_dot   = pii + (qi.*sin(phil(index))+ri.*cos(phil(index))).*tan(thetal(index));
    phil      = phil(index) + phi_dot*dt;
    thetal    = thetal(index) + theta_dot*dt;
end

phil=[phil(1); phil(1:ndp-1)]-mean(phil);

thetal=[thetal(1);thetal(1:ndp-1)];

%-- creating extra inputs
bias      = ones(ndp,1);
u4        = X2.*sin(phil).*cos(thetal);
u5        = (ri.*sin(phil)+qi.*cos(phil)).*tan(thetal);
u6        = (qi.*ri*(Iyy-Izz)+pii.*Ixz);
u7        = (pii.*qi*(Ixx-Iyy)-qi.*ri*Ixz);
u8        = (pii.^2-ri.^2);

u=[Aileron Rudder bias u4 u5 u6 u7 u8]; % all he inputs into system

%-- ZOH delay correction
u(1:ndp-1)=0.5*(u(1:ndp-1)+u(2:ndp)); % Use mean of input over sample period

uydata=[u pii ri Yaccel]; %inputs and outputs

% Organising start parameters
%-----
pstart=[C_Y_beta, C_Y_dr, C_l_beta, C_l_p, C_l_r, C_l_da, C_l_dr, C_n_beta, C_n_p, C_n_r, C_n_da, ...
        C_n_dr, C_a_Sa, C_a, C_Y_Sb, C_l_bias, C_n_bias, z_ay, x_ay, y_ay, C_Y_bias,...
        Vel, DynPress, X0_1, X0_2, X0_3, X0_4, X0_5, Ixx, Iyy, Izz, Ixz,...
        Proc_n1, Proc_n2, Proc_n3, Proc_n4 C1 C2 A1 ];

% Parameters to be estimated
%-----
ap=[1 3 4 5 8 9 10];
abp=[29 31 32 ];
bp=[2 6 7 12 ];
cp=[18 19 20];
dp=[21];
x0p=[26 27];
N=[33 34 35 36];

%--- for estimating the parameters
% pidq =[15 16 17 21 x0p 14 ];
% pidm =[ap bp 15 16 17 x0p 21 13 11];
% pidf =[ap bp 11] ;

%--- for simulation of weighted mean system
pidq =[15 16 17 21 x0p 14 ];
pidm =[15 16 17 x0p 21 13 14];
pidf =[15 16 17 x0p 13 14] ;

% setup and run MMLE
%-----

p0 = pstart;
pert = 0.001;
pref=p0;
linesearch = 1;
opt=[1 60 10 10 0.000001 0.0001 0.001 0.5];
mmle
    results1=results;

% Save Results
%-----
if manindex==1
    depth=size(results_all,3);
    end
results_all(:, :,manindex+depth)=results1;
pfinal(:,manindex+depth)= pfin';

```



```

end % end of for loop

% Save Results of the third flight
%-----
save(ressave,'results_all','pfinal');

%=====

```

D.2.3 Longitudinal Estimation

Function:K13LongP2SS3.m

```

function [A,phi,gam,C,D,q,x0,dt,rowing,B]=K13LongP2SS3(p)
% P2SS function for ZS-GHB Longitudinal Motion
% Linearized Equations of Motion used from
% Application of parameter estimation to aircraft
% stability and control, Maine and Iliff
%-----

% ----- Unpack coefficients from p-vector
C_N_alpha = p(1);
C_N_delta = p(2);
C_A_alpha = p(3);
C_A_delta = p(4);
C_m_alpha = p(5);
C_m_delta = p(6);
C_m_q = p(7);
C_N_bias = p(8);
C_A_bias = p(9);
C_m_bias = p(10);
C_L_bias = p(11);
alphadot_bias = p(12);
m = p(13);
V = p(14);
dynPress = p(15);
Xan = p(16);
Proc_n1 = p(17);
Proc_n2 = p(18);
alpha_IC = p(19);
q_IC = p(20);
d_e_delay = p(21);
Iy = p(22);
K_alpha = p(23);
Zax = p(24);
Yan = p(25);
outselect = p(26);
C_q_bias = p(27);

% Constants
global K1 K2 K3
S = 17.5; % Wing surface area - m2
chord = 1.09; % Mean Aerodynamic chord - m
g = 9.81; % Gravitational acceleration
m = 400; % Mass kg
dt = 0.05;

% Define State-Space matrices
%-----
% state equation xdot = Ax + Bu

A = [ K1*C_N_alpha 1 ;
      K2*C_m_alpha K2*C_m_q*chord/2/V ];

B = [K1*C_N_delta, 1, K1*(C_N_bias+alphadot_bias),0,0,0,0,
      K2*C_m_delta, 0, K2*C_m_bias, 0,0,0,0,0];

% output equation y = C x + D u
C3 = [K3*C_N_alpha, 0] + Xan/g * A(2,:);
C4 = [-K3*C_A_alpha, 0] + Zax/g * A(2,:);

C = [0,1;C3;C4];

D3 = [K3*C_N_delta, 0, K3*C_N_bias, Zax/g,Zax/g, 0, -Yan/g, 0];
D4 = [-K3*C_A_delta, 0, -K3*C_A_bias,Xan/g, 0, Xan/g, 0, -Yan/g];

```

```

D = {0,0,C_q_bias,0,0,0,0,0; D3; D4};
D(2,:)=D(2,:) + Xan/g * B(2,:); % +Zax/g
D(3,:)=D(3,:) + Zax/g * B(2,:);

%--- selects different outputs outputs
switch outselect
case 0
    C=C; %--- all
    D=D;
case 1
    C=C(1,:); %--- pitchrate only
    D=D(1,:);
case 2
    C=[C(1,:);C(3,:)]; %-- pitchrate and axial accel
    D=[D(1,:);D(3,:)];
case 3
    C=C(3,:); %--- axial accel
    D=D(3,:);
case 4
    C=C(2,:); %--- Normal Acceleration
    D=D(2,:);
case 5
    C=[C(1,:);C(2,:)]; % pitchrate and Normal Acceleration
    D=[D(1,:);D(2,:)];
otherwise
    C=C; %--- all
    D=D;
end

% process noise parameters
% -----
q=[ p(17) 0; 0, p(18)];

%--- rows in Q in which the parameter occur
rowinq=zeros(1,length(p));
rowinq(17:18)=[1 2];

% initial state [alpha,q]
% -----
x0=p(19:20);

% discretize
% -----

[phi,gam] = c2d_nm(A,B,dt); % C2D_MM == C2D

%=====

Script: oelong1608.m

% Script to setup and run the MMLE algo
% Estimating the Longitudinal Motion derivatives.
% For the flight stored in '16_08_2003SPLT.mat'
% Single manoeuvre estimation
%-----

format compact,clc
clear

% Constants
%-----
S = 17.5; % Wing surface area - m2
chord = 1.09; % Mean aerodynamic chord - m
m = 400; % maneuver weight (constant)
Iy = 3027.1; % Y moment of inertia
g = 9.81; % Gravitational acceleration - ms-2
dt = 0.05; % downsampled datafilt.m ,data sampled at 20 hz

% name of P2SS
%-----
p2ssnam = 'K13LongP2SS3';

% load data and calibrate
%-----
if exist('FTdata')== 0
    load('16_08_2003SPLT.mat');
    p_bias =453.5682; % biases are for this particular flight only
    q_bias =455.2555;

```

```

r_bias =527.3622;
man = FTdata.Longitudinal.Manoeuvre;

Kp = 32.87/3.07*0.00488*pi/180; % Gyro scale factors
Kq = 32.85/3.06*0.00488*pi/180;
Kr = 32.92/3.15*0.00488*pi/180;

p_all = (Kp * (FTdata.Lateral.RollRate - p_bias));
q_all = (Kq * (FTdata.Longitudinal.PitchRate - q_bias));
r_all = (Kr * (FTdata.Lateral.YawRate - r_bias));

FTdata.DiffPsr = ((1023-FTdata.DiffPsr)*555.5*0.00488/2.531);

a_x_all = 0.00488/4.11*(FTdata.Longitudinal.Xaccel-436.7471)*9.81/9.81;
a_z_all = 0.00488/1.8*(FTdata.Longitudinal.Zaccel-320.6194)*9.81/9.81;
Elavator_all=-0.001027*FTdata.Longitudinal.Elavator+0.29077;

end

% starting values for the parameters
%-----
p=zeros(1,24);
p(1) = 6.1464 ;Name(1)='C_N_alpha';
p(2) = 0.3494 ;Name(2)='C_N_delta';
p(3) = -0.0170 ;Name(3)='C_A_alpha';
p(4) = 0.0134 ;Name(4)='C_A_delta';
p(5) = -0.669 ;Name(5)='C_m_alpha';
p(6) = -3.3918 ;Name(6)='C_m_delta';
p(7) = -30.5183 ;Name(7)=' C_m_q';
p(8) = 1.182 ;Name(8)=' C_N_bias';
p(9) = -0.0162 ;Name(9)=' C_A_bias';
p(10) = 0.1252 ;Name(10)=' C_m_bias';
p(11)= 0 ;Name(11)=' C_L_bias';
p(12)= -0.4854 ;Name(12)='alphadot_b';
p(13)= m ;Name(14)=' mass';
p(14)= 24 ;Name(14)=' V';
p(15)= 3 ;Name(15)=' dynPress';
p(16)= -0.839 ;Name(16)=' Xan';
p(17)= 0 ;Name(17)=' Proc_n1';
p(18)= 0 ;Name(18)=' Proc_n2';
p(19)= 0.1 ;Name(19)=' alpha_IC';
p(20)= 0 ;Name(20)=' q_IC';
p(21)= 0 ;Name(21)='d_e_delay';
p(22)= Iy ;Name(22)=' Iy';
p(23)= 3 ;Name(23)=' K_alpha';
p(24)= -0.37 ;Name(24)=' Zax';
p(25)= 0 ;Name(25)=' Yax';
p(26)=0 ;Name(26)='output select';
p(27)= -0.1116 ;Name(27)=' C_q_bias';

ptrue= p;

% extracting Manoeuvre from data
%-----
index=4;
mstart = man(index,1);
mend = man(index,2);

% make uydata
%-----
global uydata twoframer wapriori K1 K2 K3

airdensity = 1.2;% intelligent thumb suck values
dynpress = FTdata.DiffPsr(mstart:mend);
dyn=dynpress;
V = abs(sqrt(2.*dynpress/airdensity));%*ones(ndp,1); % using the starting velocity
pii = p_all(mstart:mend);
qi = -q_all(mstart:mend);%sign correction
ri = r_all(mstart:mend);
Elavator = Elavator_all(mstart:mend);
a_x = a_x_all(mstart:mend);
a_x = a_x - mean(a_x);
a_z = -a_z_all(mstart:mend); %a_z is opposite to normal
a_z = a_z - mean(a_z);

%--- Manoeuvre specific constants
dynpress=mean(dynpress);

```

```

V=mean(V);
K1 = -dynpress*S/m/V;
K2 = dynpress*S*chord/Iy;
K3 = dynpress*S/m/g;

%--- Generating phi, theta, alpha, r_dot, p_dot
phil=zeros(ndp+1,1);
thetal=phil;
phil_0=0;
thetal_0=0;
alpha(1)=p0(19);
phil(1)=phil_0;
thetal(1)=thetal_0;
p_dot=phil;
r_dot=phil;

for index=1:ndp
    theta_dot = qi.*cos(phil(index))-ri.*sin(phil(index));
    phi_dot   = pii + (qi.*sin(phil(index))+ri.*cos(phil(index))).*tan(thetal(index));
    phil      = phil(index) + phi_dot*dt;
    thetal    = thetal(index) + theta_dot*dt;
    alpha_dot = K1*(p(1)+p(12))+qi+g/V*(cos(phil(index))*cos(thetal(index))*cos(alpha(index))+sin(thetal(index))*sin(alpha(index)));
    alpha     = alpha(index) + alpha_dot*dt;
    if index ~= 1
        p_dot(index) = (pii(index)-pii(index-1))/dt;
        r_dot(index) = (ri(index)-ri(index-1))/dt;
    else
        p_dot(index) = (pii(index+1)-pii(index))/dt;
        r_dot(index) = (ri(index+1)-ri(index))/dt;
    end
end

phil=(phil(1); phil(1:ndp-1))-mean(phil);
thetal=[thetal(1);thetal(1:ndp-1)];
p_dot=p_dot(1:ndp);
r_dot=r_dot(1:ndp);

%--- Generating inputs
u1 = Elevator;
u2=g./V.*(cos(phil).*cos(thetal).*cos(alpha)+sin(thetal).*sin(alpha));
u3 = ones(ndp,1);
u4 = qi.^2;
u5 = pii.^2;
u6 = ri.^2;
u7 = p_dot;
u8 = r_dot;
u = [u1,-u2,u3,u4,-u5,-u6,u7,u8];

%--- ZOH delay correction
u(1:ndp-1)=0.5*(u(1:ndp-1)+u(2:ndp)); % Use mean of input over sample period
                                     % to compensate for ZOH half sample delay

y=(qi,a_z,a_x);
uydata = [u y]; % Pack into uydata
p(26)=0;

% setup and run MMLE
%-----
p0=p;
%--- for estimation of all derivatives
%pidq = [ 7 8 9 10 12 19 20];
%pidm = [ 1 2 3 4 5 6 7 8 9 10 12 27 ];
%pidf = [ 1 2 3 4 5 6 7 ];
%--- for estimation of rest with alpha derivatives fixed
%pidq = [ 7 8 9 10 12 19 20];
%pidm = [ 2 4 6 7 8 9 10 12 27 ];
%pidf = [ 2 4 6 7 ];
%--- for simulation of weighted mean biases and initial conditions
pidq = [ 8 9 10 12 19 20];
pidm = [ 8 9 10 12 27 ];
pidf = [ 8 9 10 ];

opt=[1 50 40 10 0.0001 0.0001 0.0001 0.5];
mmle

% Plotting the results
%-----

```

```

figure
t=[0:ndp-1]*.05';
subplot(411);
h=plot(t,[u1 ]);
grid;ylabel('\delta_e [rad]');
subplot(412);
h=plot(t,[y(:,1) yest(:,1) ]); set(h(2),'Col','r','linestyle','-');
grid; ylabel('q [rad.s^-1] ');
legend('Flight data','Weighted Mean fit',4);

subplot(413);
h=plot(t,[y(:,2) yest(:,2) ]); set(h(2),'Col','r','linestyle','-');
grid; ylabel('a_n [g]');

subplot(414);
h=plot(t,[y(:,3) yest(:,3) ]); set(h(2),'Col','r','linestyle','-');
grid; ylabel('a_x [g]');
xlabel('Time [s]');

set(gcf, 'Papertype', 'A4');
set(gcf,'PaperOrientation','portrait');
set(gcf,'PaperUnits','inches')
set(gcf,'InvertHardcopy','on')

```

```

figure
inov=y-yest;
subplot(311);
h=plot(t,[inov(:,1) ]);
%axis([0 max(t) -0.5 0.5]);
grid; ylabel('qi inovations');
subplot(312);
h=plot(t,[inov(:,2) ]);
%axis([0 max(t) -0.5 0.5]);
grid; ylabel('a_z inovations');

subplot(313);
h=plot(t,[inov(:,3) ]);
%axis([0 max(t) -0.5 0.5]);
grid; ylabel('a_x inovations');

```

```

shg
error=mean(abs(inov))

```

```

%=====

```

```

Scriptoelongall4.m

```

```

% Script to setup and run the MMLE algo
% Estimating the Longitudinal Motion derivatives.
% For the flight stored in '16_08_2003SFLT.mat'
% Single manoeuvre estimation
%-----

```

```

format compact,clc
clear

```

```

% Constants
%-----

```

```

% P2SS filename and results savefile
%-----

```

```

p2ssnam = 'K13LongP2SS3';
ressave=input('filename for results : ','s');

```

```

% Constants
%-----

```

```

S = 17.5; % Wing surface area - m2
chord = 1.09; % Mean Aerodynamic chord - m
m = 400; % maneuver weight (constant)
Iy = 3027.1; % Y moment of inertia
g = 9.81; % Gravitational acceleration - ms-2
dt = 0.05; % downsampled datafilt.m ,data sampled at 20 hz

```

```

% Parameters to be estimated
%-----

```

```

%--- for estimation of all derivatives

```

```

%pidq = [ 7 8 9 10 12 19 20];
%pidm = [ 1 2 3 4 5 6 7 8 9 10 12 27 ];
%pidf = [ 1 2 3 4 5 6 7 ];
%--- for estimation of rest with alpha derivatives fixed
%pidq = [ 7 8 9 10 12 19 20];
%pidm = [ 2 4 6 7 8 9 10 12 27 ];
%pidf = [ 2 4 6 7 ];

% starting values for the parameters
%-----
p=zeros(1,24);
p(1) = 6.1464 ;Name{1}='C_N_alpha';
p(2) = 0.3494 ;Name{2}='C_N_delta';
p(3) = -0.0170 ;Name{3}='C_A_alpha';
p(4) = 0.0134 ;Name{4}='C_A_delta';
p(5) = -0.669 ;Name{5}='C_m_alpha';
p(6) = -3.3918 ;Name{6}='C_m_delta';
p(7) = -30.5183 ;Name{7}=' C_m_q';
p(8) = 1.182 ;Name{8}=' C_N_bias';
p(9) = -0.0162 ;Name{9}=' C_A_bias';
p(10)= 0.1252 ;Name{10}=' C_m_bias';
p(11)= 0 ;Name{11}=' C_L_bias';
p(12)= -0.4854 ;Name{12}=' alphasdot_b';
p(13)= m ;Name{14}=' mass';
p(14)= 24 ;Name{14}=' V';
p(15)= 3 ;Name{15}=' dynPress';
p(16)= -0.839 ;Name{16}=' Xan';
p(17)= 0 ;Name{17}=' Proc_n1';
p(18)= 0 ;Name{18}=' Proc_n2';
p(19)= 0.1 ;Name{19}=' alpha_IC';
p(20)= 0 ;Name{20}=' q_IC';
p(21)= 0 ;Name{21}=' d_e_delay';
p(22)= Iy ;Name{22}=' Iy';
p(23)= 3 ;Name{23}=' K_alpha';
p(24)= -0.37 ;Name{24}=' Zax';
p(25)= 0 ;Name{25}=' Yax';
p(26)=0 ;Name{26}='output select';
p(27)= -0.1116 ;Name{27} = 'C_q_bias' ;

%=====
% Starting with First flight
%=====
% load data and calibrate
%-----
load('11_23_flight_1SPLT.mat');
p_bias =452.25; % biases ar for the first flight only
q_bias =455.0531;
r_bias =526.0582;
man = FTdata.Longitudinal.Manoevre;

Kp = 32.87/3.07*0.00488*pi/180; % Gyro scale factors
Kq = 32.85/3.06*0.00488*pi/180;
Kr = 32.92/3.15*0.00488*pi/180;

p_all = (Kp * (FTdata.Lateral.RollRate - p_bias));
q_all = (Kq * (FTdata.Longitudinal.PitchRate - r_bias));
r_all = (Kr * (FTdata.Lateral.YawRate - q_bias));

FTdata.DiffPsr = ((1023-FTdata.DiffPsr)*555.5*0.00488/2.531);
FTdata.Lateral.Yaccel = 0.00488/4.065*(FTdata.Lateral.Yaccel-574)*9.81;
FTdata.Longitudinal.Xaccel =0.00488/4.11*(FTdata.Longitudinal.Xaccel-416.5)*9.81/9.81;
FTdata.Longitudinal.Zaccel =0.00488/1.8*(FTdata.Longitudinal.Zaccel-294.4)*9.81/9.81;

a_x_all = 0.00488/4.11*(FTdata.Longitudinal.Xaccel-436.7471)*9.81/9.81;
a_z_all = 0.00488/1.8*(FTdata.Longitudinal.Zaccel-320.6194)*9.81/9.81;
Elavator_all=-0.001027*FTdata.Longitudinal.Elavator+0.29077;

% extracting Manoeuvre from data
%-----
for manindex = 1:length(man)

mstart = man(manindex,1);
mend = man(manindex,2);

ptrue= p;

```

```

% make uydata
%-----
global uydata twoframer wapriori K1 K2 K3

airdensity = 1.2;% intelligent thumb suck values
dynpress = FTdata.DiffPsr(mstart:mend);
dyn=dynpress;
V = abs(sqrt(2.*dynpress/airdensity));%*ones(ndp,1); % using the starting velocity
pii = p_all(mstart:mend);
qi = -q_all(mstart:mend);%sign correction
ri = r_all(mstart:mend);
Elavator = Elavator_all(mstart:mend);
a_x = a_x_all(mstart:mend);
a_x = a_x - mean(a_x);
a_z = -a_z_all(mstart:mend); %a_z is opposite to normal
a_z = a_z - mean(a_z);

%--- Manoeuvre specific constants
dynpress=mean(dynpress);
V=mean(V);
K1 = -dynpress*S/m/V;
K2 = dynpress*S*chord/Iy;
K3 = dynpress*S/m/g;

%--- Generating phi,theta,alpha,r_dot,p_dot
phil=zeros(ndp+1,1);
thetal=phil;
phil_0=0;
thetal_0=0;
alphal(1)=p0(19);
phil(1)=phil_0;
thetal(1)=thetal_0;
p_dot=phil;
r_dot=phil;

for index=1:ndp
    theta_dot = qi.*cos(phil(index))-ri.*sin(phil(index));
    phi_dot = pii + (qi.*sin(phil(index))+ri.*cos(phil(index))).*tan(thetal(index));
    phil = phil(index) + phi_dot*dt;
    thetal = thetal(index) + theta_dot*dt;
    alpha_dot = K1*(p(1)+p(12))+qi+g/V*(cos(phil(index))*cos(thetal(index))*cos(alphal(index))+sin(thetal(index))*sin(alphal(index)));
    alphal = alphal(index) + alpha_dot*dt;
    if index ~= 1
        p_dot(index) = (pii(index)-pii(index-1))/dt;
        r_dot(index) = (ri(index)-ri(index-1))/dt;
    else
        p_dot(index) = (pii(index+1)-pii(index))/dt;
        r_dot(index) = (ri(index+1)-ri(index))/dt;
    end
end

phil=[phil(1); phil(1:ndp-1)]-mean(phil);
thetal=[thetal(1);thetal(1:ndp-1)];
p_dot=p_dot(1:ndp);
r_dot=r_dot(1:ndp);

%--- Generating inputs
u1 = Elevator;
u2=g./V.*(cos(phil).*cos(thetal).*cos(alphal)+sin(thetal).*sin(alphal));
u3 = ones(ndp,1);
u4 = qi.^2;
u5 = pii.^2;
u6 = ri.^2;
u7 = p_dot;
u8 = r_dot;
u = [u1,-u2,u3,u4,-u5,-u6,u7,u8];

%--- ZOH delay correction
u(1:ndp-1)=0.5*(u(1:ndp-1)+u(2:ndp)); % Use mean of input over sample period
% to compensate for ZOH half sample delay

y=(qi,a_z,a_x);
uydata = [u y]; % Pack into uydata
p(26)=0;

% setup and run MMLE
%-----
p0=p;

```

```

opt=(1 50 40 10 0.0001 0.0001 0.0001 0.5);
mmle

% Save results for this manoeuvre
%-----
if manindex==1
    [r,c]=size(results);
    results_all=zeros(r,c,length(man));
    pfinal = zeros(length(pfin'),length(man));

end

results_all(:,:,manindex)=results;
pfinal(:,manindex)= pfin';

end % end of for loop

% Save results for this flight
%-----
save(ressave,'results_all','pfinal');

clear FTdata man

%=====
% Starting with second flight
%=====
% load data and calibrate
%-----

clear FTdata man
load('11_23_flight_2SPLT.mat');
load(ressave);
p_bias =452.7162; % biases ar for the first flight only
q_bias =455.5136;
r_bias =526.1985;
man = FTdata.Longitudinal.Manoevre;

Kp = 32.87/3.07*0.00488*pi/180; % Gyro scale factors
Kq = 32.85/3.06*0.00488*pi/180;
Kr = 32.92/3.15*0.00488*pi/180;

p_all = (Kp * (FTdata.Lateral.RollRate - p_bias));
q_all = (Kq * (FTdata.Longitudinal.PitchRate - q_bias));
r_all = (Kr * (FTdata.Lateral.YawRate - r_bias));

FTdata.DiffPsr = ((1023-FTdata.DiffPsr)*555.5*0.00488/2.531);
FTdata.Lateral.Yaccel = 0.00488/4.065*(FTdata.Lateral.Yaccel-470)*9.81;
FTdata.Longitudinal.Xaccel =0.00488/4.11*(FTdata.Longitudinal.Xaccel-432.5083)*9.81/9.81;
FTdata.Longitudinal.Zaccel =0.00488/1.8*(FTdata.Longitudinal.Zaccel-320.9335)*9.81/9.81;

a_x_all = 0.00488/4.11*(FTdata.Longitudinal.Xaccel-436.7471)*9.81/9.81;
a_z_all = 0.00488/1.8*(FTdata.Longitudinal.Zaccel-320.6194)*9.81/9.81;
Elavator_all=-0.001027*FTdata.Longitudinal.Elavator+0.29077;

% extracting Manoeuvre from data
%-----
for manindex = 1:length(man)

    mstart = man(manindex,1);
    mend = man(manindex,2);

ptrue= p;
% make uydata
%-----
global uydata twofcramer wapriori K1 K2 K3

airdensity = 1.2;% intelligent thumb suck values
dynpress = FTdata.DiffPsr(mstart:mend);
dyn=dynpress;
V = abs(sqrt(2.*dynpress/airdensity));%*ones(ndp,1); % using the starting velocity
pii = p_all(mstart:mend);
qi = -q_all(mstart:mend);%sign correction
ri = r_all(mstart:mend);
Elavator = Elavator_all(mstart:mend);
a_x = a_x_all(mstart:mend);
a_x = a_x - mean(a_x);
a_z = -a_z_all(mstart:mend); %a_z is opposite to normal

```



```

a_z = a_z - mean(a_z);

%--- Manoeuvre specific constants
dynpress=mean(dynpress);
V=mean(V);
K1 = -dynpress*S/m/V;
K2 = dynpress*S*chord/Iy;
K3 = dynpress*S/m/g;

%--- Generating phi,theta,alpha,r_dot,p_dot
phil=zeros(ndp+1,1);
thetal=phil;
phil_0=0;
thetal_0=0;
alphan(1)=p0(19);
phil(1)=phil_0;
thetal(1)=thetal_0;
p_dot=phil;
r_dot=phil;

for index=1:ndp
    theta_dot = qi.*cos(phil(index))-ri.*sin(phil(index));
    phi_dot   = pii + (qi.*sin(phil(index))+ri.*cos(phil(index))).*tan(thetal(index));
    phil      = phil(index) + phi_dot*dt;
    thetal    = thetal(index) + theta_dot*dt;
    alpha_dot = K1*(p(1)+p(12))+qi+g/V*(cos(phil(index))*cos(thetal(index))*cos(alphan(index))+sin(thetal(index))*sin(alphan(index)));
    alphan    = alphan(index) + alpha_dot*dt;
    if index ~= 1
        p_dot(index) = (pii(index)-pii(index-1))/dt;
        r_dot(index) = (ri(index)-ri(index-1))/dt;
    else
        p_dot(index) = (pii(index+1)-pii(index))/dt;
        r_dot(index) = (ri(index+1)-ri(index))/dt;
    end
end

phil=[phil(1); phil(1:ndp-1)]-mean(phil);
thetal=[thetal(1);thetal(1:ndp-1)];
p_dot=p_dot(1:ndp);
r_dot=r_dot(1:ndp);

%--- Generating inputs
u1 = Elevator;
u2=g./V.*(cos(phil).*cos(thetal).*cos(alphan)+sin(thetal).*sin(alphan));
u3 = ones(ndp,1);
u4 = qi.^2;
u5 = pii.^2;
u6 = ri.^2;
u7 = p_dot;
u8 = r_dot;
u = [u1,-u2,u3,u4,-u5,-u6,u7,u8];

%--- ZOH delay correction
u(1:ndp-1)=0.5*(u(1:ndp-1)+u(2:ndp)); % Use mean of input over sample period
% to compensate for ZOH half sample delay

y=(qi,a_z,a_x);
uydata = [u y]; % Pack into uydata
p(26)=0;

% setup and run MMLE
%-----
p0=p;
opt=[1 50 40 10 0.0001 0.0001 0.0001 0.5];
mmle

% Save results for this manoeuvre
%-----
if manindex==1
    depth=size(results_all,3);
    end
results_all(:, :,manindex+depth)=results;
pfinal(:,manindex+depth)= pfin';

end % end of for loop

% Save results for this flight
%-----

```

```

    alpha1 = alpha(index) + alpha_dot*dt;
    if index ~= 1
        p_dot(index) = (pii(index)-pii(index-1))/dt;
        r_dot(index) = (ri(index)-ri(index-1))/dt;
    else
        p_dot(index) = (pii(index+1)-pii(index))/dt;
        r_dot(index) = (ri(index+1)-ri(index))/dt;
    end
end

phil=[phil(1); phil(1:ndp-1)]-mean(phil);
thetal=[thetal(1);thetal(1:ndp-1)];
p_dot=p_dot(1:ndp);
r_dot=r_dot(1:ndp);

%--- Generating inputs
u1 = Elevator;
u2=g./V.*(cos(phil).*cos(thetal).*cos(alpha)+sin(thetal).*sin(alpha));
u3 = ones(ndp,1);
u4 = qi.^2;
u5 = pii.^2;
u6 = ri.^2;
u7 = p_dot;
u8 = r_dot;
u = [u1,-u2,u3,u4,-u5,-u6,u7,u8];

%--- ZOH delay correction
u(1:ndp-1)=0.5*(u(1:ndp-1)+u(2:ndp)); % Use mean of input over sample period
                                         % to compensate for ZOH half sample delay

y=[qi,a_z,a_x];
uydata = [u y]; % Pack into uydata
p(26)=0;

% setup and run MMLE
%-----
p0=p;
opt=[1 50 40 10 0.0001 0.0001 0.0001 0.5];
mmle

% Save results for this manoeuvre
%-----
    if manindex==1
        depth=size(results_all,3);
        end
    results_all(:, :,manindex+depth)=results;
    pfinal(:,manindex+depth)= pfin';

end % end of for loop

% Save results for this flight
%-----
save(ressave,'results_all','pfinal');

%-----

```