

CW/CoA-2000/0011

Cranfield
UNIVERSITY

Aircraft Parameter Identification Using Matlab



G. J. Mullen

COA Report No. 0011
June 2000

College of Aeronautics
Cranfield University
Cranfield
Bedford MK43 0AL
England



1403586545

College of Aeronautics Report No.0011
June 2000

AIRCRAFT PARAMETER IDENTIFICATION USING MATLAB



G. J. Mullen
Flight Test and Dynamics Group
College of Aeronautics
Cranfield University,
Cranfield, Bedford MK43 0AL

June 2000

ISBN 1 86194 052 1

ABSTRACT

System identification techniques are routinely used in experimental stability and control studies throughout the aerospace industry. Over the years, various researchers at the College of Aeronautics have contributed to this field; most recently some of the latest methods have been employed to estimate the stability and control derivatives of a variety of aircraft types. Although the more recent investigations provide a useful insight into the capabilities and characteristics of several up-to-date methods, they have not resulted in tools which may be used on a routine basis.

Consequently, the purpose of this report is to describe a set of procedures which are straightforward to apply, and produce reasonable solutions to the type of linear parameter identification problems which are often found in aerospace work. Recordings of the short period and phugoid modes from Handley-Page Jetstream G-NFLC are used throughout as examples.

Firstly, those characteristics of the aircraft's instrumentation system which influence the quality of the signals - sample rate, antialiasing filters, time delays - are considered. This information is used in conjunction with standard signal processing techniques to ensure that the data is of sufficient quality to be used in the parameter estimation process. Next, a basic Fourier analysis and a least squares algorithm are employed to produce non-parametric and parametric models respectively. The results thus obtained are comparable to those generated using more sophisticated techniques.

In conclusion, standard signal processing methods combined with relatively simple estimation theory offer an adequate solution to the linear parameter estimation problem.

TABLE OF CONTENTS

Abstract	i
List of Contents	ii
List of Figures	iii
Nomenclature	iv
1. INTRODUCTION	1
2. SIMULATED DATA	4
3. SIGNAL CHARACTERISTICS	4
4. DATA RECORDING and PROCESSING	5
4.1 Sample Rate	6
4.2 Filtering	7
4.3 Transport Delays	8
4.4 Removing Offsets and Defining Units	8
4.5 Differentiating the States	9
4.6 Coherence	9
5. TWO BASIC IDENTIFICATION ALGORITHMS	9
5.1 A Non-Parametric Identification Technique	10
5.1.1 Basic Theory	10
5.1.2 Example	11
5.2 A Parametric Identification Technique	12
5.2.1 Basic Theory	13
5.2.2 Examples	14
5.3 Additional Comments	18
6. MAKING MORE USE OF THE BUILT-IN FUNCTIONS	18
6.1 Non-Parametric Frequency Response Identification	19
6.2 Parametric Identification Techniques	19
6.2.1 The OE Function	20
6.2.2 The PEM Function	22
7. CONCLUSIONS AND SUGGESTIONS FOR FUTURE WORK	23
REFERENCES	24
APPENDIX A: LIST OF DATA FILES	
APPENDIX B: SOFTWARE LISTINGS	
APPENDIX C: MATLAB DIARY EXAMPLES	

LIST OF FIGURES

Figure 1.	Cranfield University Jetstream and Bulldog	25
Figure 2.	Power spectral densities	26
Figure 3.	Power spectral densities, short period mode	27
Figure 4.	<i>IRS_{PtcRt}/Elevator</i> coherence, short period mode	28
Figure 5.	q/η frequency response, DFT and parametric model	28
Figure 6.	q/η frequency response, processed and unprocessed signals	29
Figure 7.	Measured and estimated short period response	30
Figure 8.	Measured and estimated short period, transport delay in pitch rate	31
Figure 9.	Estimated and measured short period and phugoid	32

NOMENCLATURE

α_{True}	True angle of attack (from <i>POSTFLT</i>)
$b_1 \cdots b_{n_b}$	Coefficients of $B(q)$ in output error structure
$B(q)$	Numerator polynomial in output error structure
$\bar{c}$	Standard mean chord
$e(t)$	Measurement noise in output error structure
EKF	Extended Kalman filter
δ	Elevator position (from <i>POSTFLT</i>)
$f_1 \cdots f_{n_f}$	Coefficients of $F(q)$ in output error structure
$F(q)$	Denominator polynomial in output error structure
FLDS	Fixed lag differentiator smoother
GM	Gauss-Markov
H	Matrix of w , q and η samples
IEKF	Iterated extended Kalman filter
IRS	Inertial reference system
$\dot{\delta}$	Pitch rate (from <i>POSTFLT</i>)
k	Harmonic number
$k_{\max}$	Maximum harmonic number
k_q	Numerator gain in elevator to pitch rate transfer function
m	Pitching moment (concise quantity)
$\dot{M}_q$	Pitching moment due to pitch rate perturbation (dimensional derivative)
MAP	Maximum a posteriori
MBFS	Modified Bryson-Frazier smoother
MSR	Modified stepwise regression
n	Sample number
nk	Number of delays in output error structure
N	Total number of samples

NOMENCLATURE (continued)

OLS	Ordinary least squares
PI	Parameter identification
PSD	Power spectral density
q	Pitch rate perturbation; polynomial as function of delay operator
q_k	Imaginary component of the j^{th} harmonic vector
r_k	Real component of the j^{th} harmonic vector
s	Laplace operator
t	Time
T_s	Sample interval
T_w	Time length of data record
$T_{\theta 1}$	Numerator factor in elevator to pitch rate transfer function
$T_{\theta 2}$	Numerator factor in elevator to pitch rate transfer function
TLS	Total least squares
u	Airspeed perturbation; system input
v	Measurement error
w	normal velocity perturbation
x_n	The n^{th} sample
x	Axial force (concise quantity)
$\hat{X}_k$	Harmonic vector corresponding to the k^{th} harmonic ($= r_k + jq_k$)
y	System output
$\underline{z}$	Vector of $\dot{w}$, $\dot{q}$ samples
z	Normal force (concise quantity)
z^{-1}	Delay operator
ZPF	Zero phase filter

NOMENCLATURE (continued)

Greek Symbols

η	Elevator perturbation
ζ_p	Phugoid damping ratio
ζ_s	Short period damping ratio
ω	Discrete frequency
$\omega_{\max}$	Maximum discrete frequency
ω_p	Phugoid undamped natural frequency
ω_s	Short period undamped natural frequency

Miscellaneous

Concise derivatives are of the form, e.g.:-

z_w	Normal force due to normal velocity perturbation, etc.
$\dot{x}$	Rate of change with respect to time.
$x(n)$	Value of a variable at time $t = t(n)$

1. INTRODUCTION

Parameter identification (PI) is a process whereby the dynamics of a system are estimated from a knowledge of the input and output time histories. PI techniques are used in support of various activities throughout the aerospace industry, some common applications being:-

- Comparing the characteristics of a real aircraft to the dynamics predicted by simulation models.
- To aid in the understanding of aerodynamic phenomena and their relationships to stability and control behaviour.
- To provide more accurate models for simulators.
- To provide improved stability augmentation systems, and on a more esoteric level to estimate the models required by adaptive flight control systems.

Several generations of researchers at the College of Aeronautics have worked in this field. In 1967 Pallister [1] investigated the lateral stability characteristics of the Shorts Belfast using a scale wind-tunnel model and a combination of static and oscillatory rigs. The work was continued the following year by Martin [2].

In the 1970's, Klein (one of the most prolific researchers in the field) contributed a great deal to the applied aspects of the parameter identification problem. In Ref. [3] the basic performance of an instrumentation system is quantified, dealing with topics such as precision, accuracy, static and dynamic calibration. In his Ph.D. thesis [4], the most up to date methods (at the time) were applied to a range of simulated and test data. The capabilities of the equation error method, output error (OE) with total least squares (TLS), maximum likelihood estimation (MLE), and Bayesian techniques were successfully applied to test data from the Basset variable stability aircraft, the HP-115 slender delta wing research aeroplane, and to simulated data from a nonlinear model of the F-100 Super Sabre.

This mix of theoretical and applied work was continued in the 1980's by Foster [5]. The MLE method was used in conjunction with a steady state Kalman filter to produce a technique for identifying stability and control derivatives in a calm or turbulent

atmosphere. The techniques thus developed were verified successfully using flight data from a Gnat jet trainer.

During the 1990's, a variety of techniques were examined by several researchers. Hoff [6], used an estimation-before-modelling method to identify models of the EMB-312 Tucano dutch roll and the University's Handley Page Jetstream phugoid from flight data. This study concentrated in two main areas; state estimation followed by parameter identification. The state estimation study evaluated the use of a standard Kalman filter with a modified Bryson-Frazier smoother (MBFS), and an iterated extended Kalman filter (IEKF) combined with a fixed lag smoother differentiator (FLDS) to recover "noise free" signal estimates. The modified stepwise regression (MSR) technique was then applied to the processed signals in an attempt to estimate the stability and control derivatives. Although a theoretically interesting thesis, only a limited success was achieved with the practical implementation on the Jetstream; e.g. in all cases bar one a positive value of $\dot{M}_q$ was obtained. It should be noted that this may be due more to the nature of the data than any deficiency in the technique, since Hoff was attempting to identify both the short period and phugoid motion from data containing only a phugoid excitation.

Another practical problem was tackled by Hinds [7] who applied the MSR technique to experimental data derived from a Hawk wind tunnel model. The thesis begins with a useful summary of the principal differences between some of the most common identification methods; ordinary least squares (OLS), TLS, Gauss-Markov (GM), maximum a posteriori (MAP) and MSR. Hinds implemented a MSR algorithm in FORTRAN and validated it using data from simulations of the F-4 Phantom and the Hawk. The experimental work was however unsuccessful, a result which appears to be due to the quality of the data.

The most recent (and arguably the most successful of the experimental studies conducted in the 1990's) attempt was conducted by Baek [8], who identified the longitudinal motion of the Jetstream in the cruise from test data. Like Hoff, Baek considered both the state estimation and parameter identification problem. A range of methods were considered for the former; a relatively complex EKF/FLDS algorithm, a much simpler zero-phase filter (ZPF), and a combined EKF/ZPF. For the parameter

estimation part, OLS, TLS and MSR were used. Broadly speaking, reasonable practical results were achieved, and the thesis provided a useful comparison of several state estimation and parameter identification techniques.

Although a serious field of study in its own right, the main role of parameter identification is to support work in other areas, e.g. aerodynamic modelling, handling qualities, etc. Therefore although a great deal of interesting work has been performed at Cranfield, it is, understandably, of a specialist nature and has not produced tools which may be employed on a routine basis to support other disciplines within the College.

Furthermore, the most recent studies have not taken into account the characteristics of their respective instrumentation systems. This is a surprising oversight, given that the quality of the sensors and any associated processing can have a drastic effect on the reasonableness of the identified models.

Consequently, the purpose of this report is to describe a relatively simple set of procedures and techniques which may be used by non-specialists to identify mathematical models from flight data. The methods are illustrated throughout using test data from Handley-Page Jetstream G-NFLC (Fig. 1) and all data analysis is performed using *MATLAB*TM, the associated toolboxes, and some customised functions.

Section 2 lists a few points which should be considered before any attempts are made to manipulate flight data. Section 3 discusses those aspects of the instrumentation system which have an impact on the quality of the identified results. Appropriate sample rates and signal processing are considered in Section 4. The application of a basic Fourier analysis technique and a simple least squares algorithm to the construction of models from flight data is discussed in Section 5, while Section 6 tackles the same problems using (more sophisticated) built-in *MATLAB*TM functions. Some conclusions and suggestions for future work are offered in Section 7.

Appendix A lists the example data files used throughout the report, the author's customised functions are contained in Appendix B, and the use of these functions is illustrated in Appendix C.

Before progressing, note that this report does not contain a detailed exposition of general system identification theory; such information has been extensively developed

elsewhere [9, 10]. Simple theory is however presented where it aids the practical understanding of the problem. Furthermore, the analyses herein are confined to linear systems, and it is assumed that the reader has a basic knowledge of flight dynamics, is familiar with the recording software *RECGEN*, the *POSTFLT* data manipulation package [11], and has some knowledge of *MATLAB*TM [12-15].

Finally, this report does not attempt to discuss parameter identification in a wider context, e.g. input requirements, the characteristics of various algorithms, how to define the correct model structure, etc. Such information is already available in the open literature [3,4,9,10,16].

2. SIMULATED DATA

Before getting to grips with flight data, it is worth trying out candidate estimation techniques on simulated data. As a bare minimum, the following points should be considered:-

- Start with perfect noise free states from a linear model. Does the chosen input excitation and identification method produce reasonable results? If not, think again, since the chances of it working on real data are extremely slim.
- Begin with the simplest practical techniques; in many cases they will produce acceptable results, even if the eventual aim is to apply a complex method.
- Once reasonable estimates are obtained with “perfect” simulated information, try corrupting the signals with noise. How are the estimates affected?
- Can the effects of noise be reduced through sensible filtering? If so, how do the estimates compare with the noise free case?

3. SIGNAL CHARACTERISTICS

As an example, consider the small perturbation state space short period description [17]:-

$$\begin{bmatrix} \dot{w} \\ \dot{q} \end{bmatrix} = \begin{bmatrix} z_w & z_q \\ m_w & m_q \end{bmatrix} \begin{bmatrix} w \\ q \end{bmatrix} + \begin{bmatrix} z_\eta \\ m_\eta \end{bmatrix} \eta \quad (1)$$

where w is the heave velocity (derived from the *POSTFLT* signal *AlphaTru*), q the pitch rate (*IRSPtcRt*) and η the elevator deflection (*Elevator*). The derivatives z_w , etc. are in

concise dimensional form. In order to estimate the stability and control derivatives from the flight data it is first necessary to consider the salient characteristics of the above measurements.

The *AlphaTru* and *Elevator* signals are obtained from separate Fisher Controls 4000 series alternating current (AC) rotary position transducers [18]. These output an analogue voltage signal which is passed through a low-pass anti-aliasing filter (single pole, cut-off frequency 100 Hz) before being digitised for use in the C30 processor which forms the basis of the instrumentation display and recording system. The rotary pick-offs are infinite resolution devices, and have a linearity of $\pm 1\%$. The dominant source of error in the position measurements they provide lies in the calibration; for a given voltage output the corresponding angular position is measured manually, resulting in a systematic error of between $1/2$ to 1 deg .

The pitch rate, *IRSPtcRt*, is available as a digital output from a Litton LTN-90 inertial reference system (IRS) [19]. Within the IRS, it is filtered by a second order Butterworth low pass filter with a cut-off frequency of 8 Hz. The signal is subject to a nominal transport delay of 50 ms (although 100 ms has been observed), is accurate to 0.1 deg/sec , and has a precision of $.004\text{ deg/sec}$.

Once digitised, all measurements are processed by the C30 processor, which has a sample rate of 500 Hz.

The overall impact of the above is discussed in the next two Sections.

4. DATA RECORDING and PROCESSING

Two problems are considered in the following paragraphs:- data recording and signal processing. The reduced order short period motion is used to illustrate the main points. In terms of data recording, attention is focused on the selection of a suitable sample rate; the signal processing paragraphs describe the steps undertaken to ensure that the states, their rates of change, and the elevator input are of suitable quality to be used in the parameter estimation process.

All analysis and processing makes use of standard *MATLAB*TM functions, [12-15]. PSD plots were constructed using “*psd*”; filter coefficients were computed using “*butter*” and

implemented using “*filtfilt*” to avoid any phase distortion effects; coherence estimates were calculated by “*cohere*”. State derivatives with respect to time were constructed using “*diff*”. The various signal processing steps are combined in functions “*pro_spl*” and “*pro_lof*” for short period and full order longitudinal motion respectively.

A more detailed treatment of the points raised in Section 4 may be found in [16].

4.1 Sample Rate

Having decided which sensors are of interest, it is then necessary to select a suitable sample rate. In theory, only a minimum acceptable value (governed by the Nyquist sampling theorem) need be considered. However, there are subsidiary considerations which will limit the final choice:- the amount of data which can be stored on a standard floppy disc, and the time delay in the *IRSPtcRt* signal.

For the short period example, the Nyquist sampling theorem dictates that the *Elevator* and *AlphaTru* signals be sampled at greater than 200 Hz. However since single pole filters are used, there may be significant noise components above 200 Hz, so the actual sampling rate should be somewhat higher, perhaps 250 Hz. If the sample rate is too low, significant aliasing may occur:- Fig. 2 shows power spectral density (PSD) plots of *AlphaTru* (file *tu151817*, Appendix A) signals recorded at 500 Hz and 50 Hz (file *th105910*). When recorded at the higher frequency, the noise components above 100 Hz are clearly visible. However, at the lower sample rate the high frequency noise has folded down into apparent noise components at approximately 3 Hz and 20 Hz.

Ideally then, an acceptable sample rate should be selected by recording *AlphaTru* and *Elevator* at ever decreasing rates, starting from the 500 Hz upper limit. Compare the PSDs at each sample rate and note the lowest acceptable value at which no aliasing occurs. This may be a time consuming process, and in addition the desired combination of sample rate and the number of channels selected may produce a file which is too large to fit in a standard floppy. A compromise is usually required.

Finally, try to select a sample rate which is a divisor of the *IRSPtcRt* transport delay; this will make it easier to remove the delay at a later stage. This is however very much a secondary requirement since one can always use the *MATLAB*TM “*interp1*” function to interpolate at a later stage.

Note that satisfying the anti-aliasing requirements of the *AlphaTru* and *Elevator* signals implicitly takes care of the *IRSPtcRt* measurement.

Individual circumstances may permit a relaxation of the rules. For example, the author mistakenly understood the cut-off frequencies of the antia-aliasing filters on the two analogue measurements to be much lower than 100 Hz; a sample rate of 50 Hz was therefore assumed to be adequate and used for all of data files discussed herein. The consequences of this error can be assessed from the PSD plots, Fig. 2. The *AlphaTru* PSDs have a similar shape below 0.5 Hz (3 rad/sec); this should be adequate to capture the main features of the airframe short period response. In contrast, the *IRSPtcRt* and *Elevator* responses show similar characteristics up to around 10 Hz (63 rad/sec); more than adequate.

4.2 Filtering

A glance through some system identification literature reveals a staggering array of candidate filtering techniques; the approach adopted here is to design low-pass filters, using standard frequency domain analysis methods. While this is sufficient for many problems, and allows the user to retain a high degree of visibility, it cannot cover every eventuality.

Fig. 3 shows a PSD plot of the unfiltered *Elevator* signal (dotted line) from a short period excitation (file *th105910*). The signal power is concentrated below 15 rad/s, say, above which significant noise components are present. Thus, as a first estimate, a low-pass Butterworth filter with a cut-off frequency of 15 rad/s would be appropriate. In order to avoid filter transients, the order should be kept as low as possible; second order is sufficient in this case.

Comparing the filtered and unfiltered PSDs, the magnitude of the noise components has been significantly reduced in the latter. The frequency domain picture is therefore adequate, but the time response should also be considered; check that the chosen filter dynamics do not cause unwanted transients.

Following similar reasoning, a third order filter with a cut-off frequency of approximately 7 rad/s was applied to the *AlphaTru* measurement; for the *IRSPtcRt* signal, the same order but a higher cut-off frequency of 20 rad/s was used. There is an additional

constraint on the filter selection here, since the rate of change of each signal with respect to time must be calculated (see equation 1). Check that the chosen filter characteristics produce acceptable derivatives.

As a general rule, aim to select a filter with as low an order, and as high a cut-off frequency as possible within the constraints of the problem.

4.3 Transport Delays

Signals derived from the LTN-90 IRS are subject to transport delays. For the *IRSPtcRt* measurement, the brochure [19] quotes a nominal value of 50 ms. Experience suggests that the actual value can vary between 50 ms and 100 ms, so some iteration is normally required.

The delays are easily removed if the sample rate, actual or interpolated, is a divisor of the delay; simply shifting the *IRSPtcRt* vector to the left by the appropriate number of samples solves the problem.

4.4 Removing Offsets and Defining Units

This study is concerned with estimating linear models about a trimmed equilibrium condition, so all initial offsets must be removed from the data; in the short period example the elevator deflection is used as a baseline. The data sample corresponding to the beginning of the pilot's elevator input is noted; the mean value of each variable between the start of the data file and this sample is then calculated. The appropriate average is then subtracted from the corresponding variable so that all signals start from a zero initial condition.

The signals are measured in a variety of units. For example, the units of w are m/s , while *IRSPtcRt* and *Elevator* are measured in units of deg/s and deg respectively. To avoid confusion at a later stage, scaling factors should be applied to ensure a consistent set of units, e.g. SI.

Another, unrelated, use of scaling is to improve the numerical conditioning of some estimation algorithms. The variables may be scaled such that the peak perturbations have a magnitude of 1. This can be helpful if the matrix $H' H$ in the least squares estimation (see equation 13) is ill conditioned. The author has applied this technique to various

Jetstream estimation problems, and although the condition number of the above matrix factor is much reduced, it makes little difference to the estimates of the stability and control derivatives. This is probably due to the fact that $H'H$ typically has a condition number in the hundreds, which is generally too low to cause serious numerical problems.

4.5 Differentiating the States

This is calculated simply as the difference in the value of the states between samples divided by the time interval. By paying careful attention to the filtering of the states, the derivative estimates thus obtained should be adequate.

4.6 Coherence

It is advisable to check the coherence of a given input/output pair. A minimum value of 0.8 is generally acceptable, although this is not a rigid rule. Since flight data at Cranfield is usually recorded on an opportunistic basis, the researcher may have to live with the available information. Regardless of this, one should check that the mode of interest has actually been excited.

A case in point is the *Elevator* to *IRSPtcRt* coherence for the file *th105910*, Fig. 4. This indicates that data above a frequency of around 15 *rad/s* should be discarded, and that at some lower frequencies the coherence is less than ideal.

5. TWO BASIC IDENTIFICATION ALGORITHMS

Although *MATLAB*TM provides a host of built-in identification algorithms of varying degrees of complexity, it is instructive to consider the problem from an elementary standpoint. The aim in doing so is not to produce algorithms which are comparable in complexity, flexibility, or speed of computation, but rather to show that useful results can be achieved with the minimum of theory. Consequently this Section describes and illustrates the use of one non-parametric and one parametric algorithm, both of which are relatively straightforward to implement.

5.1 A Non-Parametric Identification Technique

By using Fourier techniques, a set of input / output time response data may transformed directly to the frequency domain. Since the transformation does not require any knowledge of the underlying aeroplane dynamics, other than an assumption of linearity, it is known as a non-parametric estimation method.

The following discussion is based on material contained in [20], Chapters 3 and 6, which contains a useful introduction to Fourier methods and their use in practical signal processing problems.

5.1.1 Basic Theory

The discrete Fourier transform (DFT) of a time series signal is:-

$$\hat{X}_k = T_s \sum_{n=0}^{N-1} x_n e^{-j2\pi kn/N} \quad (2)$$

where T_s is the sampling interval, N is the number of samples, x_n is the n^{th} sample, and k is the harmonic number. Expressing (2) as a sum of real and imaginary components:-

$$\begin{aligned} \hat{X}_k &= T_s \sum_{n=0}^{N-1} x_n [\cos(k2\pi n/N) - j \sin(k2\pi n/N)] \\ &= T_s \sum_{n=0}^{N-1} x_n \cos(nk 2\pi/N) - j T_s \sum_{n=0}^{N-1} x_n \sin(nk 2\pi/N) \end{aligned} \quad (3)$$

Setting $k2\pi/N = d\theta$, (3) becomes:-

$$\begin{aligned} \hat{X}_k &= T_s \sum_{n=0}^{N-1} x_n \cos(nd\theta) - j T_s \sum_{n=0}^{N-1} x_n \sin(nd\theta) \\ &= r_k + j q_k \end{aligned} \quad (4)$$

If this operation were carried out for a pair of input and output signals, say elevator and pitch rate, then in each case for a given harmonic k :-

$$\begin{aligned} \hat{\eta}_k &= a_k + j b_k \\ \hat{q}_k &= c_k + j d_k \end{aligned} \quad (5)$$

and the frequency response is obtained from the ratio:-

$$\frac{\hat{q}_k}{\hat{\eta}_k} \quad (6)$$

The discrete frequency vector and the maximum harmonic required may be calculated as follows. If the length of the data record is T_w seconds, then the lowest frequency which may be calculated is $1/T_w$ Hz or $2\pi/T_w$ rad/s. Hence, if an upper frequency limit of $\omega_{\max}$ rad/s is required, then the corresponding maximum harmonic, $k_{\max}$, is obtained from:-

$$\omega_{\max} = \frac{2\pi k_{\max}}{T_w} \text{ rad/s} \quad (7)$$

Since the frequency vector only exists for discrete harmonic numbers defined by integer values of k , then increment k from zero to $k_{\max}$ in steps of 1, whereupon the discrete frequency vector is:-

$$\omega = \frac{2\pi k}{T_w} \quad (8)$$

Equations (3) through (8) are implemented in “*tfest*”, Appendix B. Note that since a DFT, not a fast Fourier transform (FFT) algorithm is used, the computation times for some data files may be excessive.

5.1.2 Example

The longitudinal dynamics of the Jetstream in the cruise will be used to demonstrate an application of the DFT approach. Data file *th111334* is a recording of the short period and phugoid motion in response to transient inputs in the cruise configuration, starting from an indicated trim airspeed of 150 kts, with the cg located at 23.5 % $\bar{c}$. The dynamics between the elevator deflection and the aeroplane response are assumed to be linear, hence transient inputs are used in preference to sinusoidal inputs. The aim is to obtain a frequency response estimate of the q/η motion.

An example of the *MATLAB*TM syntax is contained in Appendix C, but roughly speaking the steps required are as follows:-

- a) Use “*pro_lof*” to perform the signal processing.
- b) Decide on the maximum frequency required.

c) Use “*tfest*” to compute the magnitude, phase and frequency vectors.

In evaluating the results, it is helpful to have a knowledge of the expected frequency response characteristics and the associated sign conventions; see [17] for more details. For the example considered herein, one is looking for peaks in the gain curve (and associated phase changes) corresponding to the phugoid and short period motion poles. Gain and phase changes due to the numerator factors may also be visible. As an aside, it is conventional to start the low frequency phase response from 0 *deg*, this invariably requires a constant offset to be added to the phase curve.

The results are shown in Fig. 5. Considering the gain curve, the peaks due to the phugoid and short period modes are clearly seen at frequencies of $\omega_p \cong 0.15 \text{ rad/s}$ and $\omega_s \cong 3 \text{ rad/s}$ respectively; the corresponding phase changes are also quite pronounced. In addition, the effects of the $1/T_{\theta 2}$ numerator factor between ω_p and ω_s may also be discerned. Since the minimum discrete frequency is inversely proportional to the length of the time window, the steady state phase and the low frequency phase changes due to the $1/T_{\theta 1}$ numerator factor cannot be seen.

Section 4 mentioned the transport delays inherent in the IRS signals. In theory, a transport delay has unit magnitude and a phase lag proportional to the frequency in radians per second [20]. Fig. 6 illustrates the effect of neglecting these on the elevator to pitch rate frequency response estimates; if the delay is not removed, an additional phase lag is present. This could of course lead to false conclusions being drawn about the longitudinal dynamic characteristics.

5.2 A Parametric Identification Technique

If it is required to estimate the numerical parameters describing the aircraft motion (e.g. the stability and control derivatives, the frequency and damping of the phugoid) then parametric identification techniques must be employed. A wide variety of methods are available to achieve this [9, 10]. The method used herein is an equation error formulation; it is arguably the simplest of the parametric techniques, and an excellent description of the underlying theory with applications may be found in [21].

In many respects equation error techniques have been superseded by more sophisticated methods, although they are commonly used to provide start-up inputs to more powerful algorithms.

5.2.1 Basic Theory

As an example, consider again the state space short period description of equation (1) [17]:-

$$\begin{bmatrix} \dot{w} \\ \dot{q} \end{bmatrix} = \begin{bmatrix} z_w & z_q \\ m_w & m_q \end{bmatrix} \begin{bmatrix} w \\ q \end{bmatrix} + \begin{bmatrix} z_\eta \\ m_\eta \end{bmatrix} \eta \quad (9)$$

If the short period motion is recorded at a set of discrete intervals $t = t(0), t(1), \dots, t(n)$ and if equation (9) holds, then at each point the following must also hold:-

$$\begin{aligned} \dot{w}(0) &= w(0)z_w + q(0)z_q + \eta(0)z_\eta + v(0) \\ \dot{q}(0) &= w(0)m_w + q(0)m_q + \eta(0)m_\eta + v(0) \\ &\text{-----} \\ \dot{w}(1) &= w(1)z_w + q(1)z_q + \eta(1)z_\eta + v(1) \\ \dot{q}(1) &= w(1)m_w + q(1)m_q + \eta(1)m_\eta + v(1) \\ &\text{-----} \\ &\vdots \\ \dot{w}(n) &= w(n)z_w + q(n)z_q + \eta(n)z_\eta + v(n) \\ \dot{q}(n) &= w(n)m_w + q(n)m_q + \eta(n)m_\eta + v(n) \end{aligned} \quad (10)$$

where, for example, $w(n)$ is the value of w measured at time $t = t(n)$ and $v(n)$ is the corresponding measurement error. Since the stability and control derivatives are common at all times, (10) may be reformulated as:-

$$\begin{bmatrix} \dot{w}(0) \\ \dot{q}(0) \\ \text{-----} \\ \dot{w}(1) \\ \dot{q}(1) \\ \text{-----} \\ \vdots \\ \dot{w}(n) \\ \dot{q}(n) \end{bmatrix} = \begin{bmatrix} w(0) & q(0) & \eta(0) & 0 & 0 & 0 \\ 0 & 0 & 0 & w(0) & q(0) & \eta(0) \\ \text{-----} \\ w(1) & q(1) & \eta(1) & 0 & 0 & 0 \\ 0 & 0 & 0 & w(1) & q(1) & \eta(1) \\ \text{-----} \\ \vdots \\ w(n) & q(n) & \eta(n) & 0 & 0 & 0 \\ 0 & 0 & 0 & w(n) & q(n) & \eta(n) \end{bmatrix} \begin{bmatrix} z_w \\ z_q \\ z_\eta \\ m_w \\ m_q \\ m_\eta \end{bmatrix} + \begin{bmatrix} v(0) \\ v(0) \\ \text{-----} \\ v(1) \\ v(1) \\ \text{-----} \\ \vdots \\ v(n) \\ v(n) \\ \text{-----} \end{bmatrix} \quad (11)$$

Equation (11) is in the general form:-

$$\underline{\dot{z}} = H\underline{\theta} + \underline{v} \quad (12)$$

where $\underline{\theta}$ is the vector of unknown derivatives. The parameter vector estimate, $\hat{\underline{\theta}}$, calculated using an OLS error criterion [21] is:-

$$\hat{\underline{\theta}} = [H' H]^{-1} H' \underline{z} \quad (13)$$

Equations (11) and (13) are implemented in “*ls_sp*”, Appendix B. This function represents a very simple implementation, and as such the computation times for some problems may be of the order of several minutes. There are two main ways of overcoming this; implement a modified algorithm [21], or sub-sample the data. If choosing the latter option, ensure that the thinned data has the same PSD characteristics as the original filtered information.

5.2.2 Examples

Data file *th105910* is a short period recording from the Jetstream in the landing configuration. The q and w responses to the pilot's input are shown in Fig. 7.

An example of the *MATLAB*TM syntax is shown in Appendix C, but roughly speaking the steps required are as follows:-

- a) Process the data using “*pro_sp1*”.
- b) Calculate the parameter estimates using “*ls_sp*”.

For this data file, the resulting state and input matrices together with the standard errors (in brackets) are:-

$$\begin{bmatrix} \dot{w} \\ \dot{q} \end{bmatrix} = \begin{bmatrix} -0.8423 & (0.0051) & 53.2018 & (0.1831) \\ -0.0247 & (0.0002) & -1.5523 & (0.0071) \end{bmatrix} \begin{bmatrix} w \\ q \end{bmatrix} + \begin{bmatrix} 20.1538 & (0.3034) \\ -5.7813 & (0.0118) \end{bmatrix} \eta \quad (14)$$

From Fig. 7 it may be seen that motion predicted by equation (14) agrees closely with the measurements recorded in flight; furthermore, since the standard error estimates are significantly smaller than the corresponding derivative estimates it may be concluded that, in numerical terms, the identification process has been successful.

However, it is worth considering if the estimated derivatives have the correct sign and a sensible magnitude; a knowledge of basic flight dynamics and a rough idea of what to expect from the Jetstream simulation will provide a distinct advantage. Starting with the stability derivatives, Cook [17] states that z_w should be negative, and the aircraft simulation suggests a magnitude of around $1/sec$; equation (14) is broadly in agreement.

From simple theory, z_q should be positive (true for the flight case considered) and approximately equal to the trim airspeed. In equation (14) z_q is 53.2 m/s, or 106 kt; from the flight records the trim airspeed is 117 kt, so a reasonable agreement has been obtained. Moving on to the pitch rate equation, both m_w and m_q should be negative (they are) and their magnitudes are in broad agreement with simulated estimates.

Considering now the control derivatives, the z_η estimates are significantly in error. z_η should be negative and approximately one third of the size indicated. The error is not due to the relatively simple techniques used to obtain the state and parameter estimates, since rather more complex methods involving the use of Kalman filtering and maximum likelihood estimation [8] also produce z_η estimates which are clearly wrong. The answer lies in the physics of the problem, since z_η influences only the *initial* part of the short period response.

For example, when the elevator is deflected trailing edge up (negative), the increased download at the tail initially causes the aeroplane to sink, thus creating a small perturbation in w in the *downwards* (positive) direction; consequently z_η should be negative. Consider now what happens immediately after the aircraft sinks; the nose up pitching moment due to elevator deflection causes the angle of attack to increase. Assuming that the flight path is unchanged, the change in angle of attack increases the overall lift, and therefore causes a perturbation in w in the *upwards* (negative) direction. Therefore if the whole w time history is used to identify z_η , the estimate will have the wrong sign since it will appear that a *negative* elevator deflection produces a *negative* w perturbation, suggesting a positive z_η .

In practice, it is very difficult to identify z_η from the flight recordings since the effect is very small. Since the error in this estimate has only a small influence on the resulting short period dynamics - the numerator zero is reduced by around 13% - the identified value may be retained. Alternatively, a sensible value of z_η may be fed into the estimation process and the other derivatives calculated accordingly.

Finally, the pitching moment due to elevator, m_η should be negative (it is) and the Jetstream simulation predicts a value of around $-6/s^2$, so a reasonable agreement has been obtained.

In general, it is more useful to identify the state space equations and then convert to other forms if required. For example, from (14) the elevator to pitch rate transfer function is:-

$$\frac{q(s)}{\eta(s)} = \frac{-5.7813(s+0.9286)}{(s^2+2.3947s+2.6232)} \text{rad/s/rad, } 31.8\%\bar{c} \quad (15)$$

and for another case at the same flight condition but at a different c.g. location:-

$$\frac{q(s)}{\eta(s)} = \frac{-5.9729(s+0.8979)}{(s^2+2.6389s+4.1245)} \text{rad/s/rad, } 23.5\%\bar{c} \quad (16)$$

If the general form of the transfer function [17] is then examined:-

$$\frac{q(s)}{\eta(s)} = \frac{m_\eta(s-z_w)}{\left[s^2 - (m_q + z_w)s + (m_q z_w - m_w U_e)\right]} \equiv \frac{k_q(s+1/T_{\theta 2})}{s^2 + 2\zeta_s \omega_s s + \omega_s^2} \quad (17)$$

and compared with expressions (15) and (16) then any changes in the short period characteristics as the cg moves aft may be traced to individual derivatives or combinations thereof [22]; note that simply identifying the transfer function does not offer this degree of visibility.

As an aside, consider the effect of neglecting the time delay in the $IRSPtcRt$ signal, Fig. 8. The least squares algorithm clearly under estimates the damping and the corresponding stability derivative estimates will also be in error.

The same technique can be applied to estimating the coefficients of a full order longitudinal model [17]:-

$$\begin{bmatrix} \dot{u} \\ \dot{w} \\ \dot{q} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} x_u & x_w & x_q & x_\theta \\ z_u & z_w & z_q & z_\theta \\ m_u & m_w & m_q & m_\theta \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} u \\ w \\ q \\ \theta \end{bmatrix} + \begin{bmatrix} x_\eta \\ z_\eta \\ m_\eta \\ m_\theta \end{bmatrix} \eta \quad (18)$$

By following the derivation in Section 5.2.1, it is relatively simple to extend the results to the fourth order identification problem. The algorithm is implemented in “*ls_long*”, Appendix B.

Using the short period and phugoid excitation file *th111334* the derivatives, together with the standard errors (in brackets) are:-

$$\begin{bmatrix} \dot{u} \\ \dot{w} \\ \dot{q} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} -.05 (.01) & .18 (.03) & -.39 (1.67) & -9.43 (.3) \\ -.17 (0) & -.76 (0) & 88.08 (.16) & 1.37 (.03) \\ -0 (0) & -.07 (0) & -2.36 (0) & 0 (0) \\ 0 (0) & 0 (0) & 1 (.02) & 0 (0) \end{bmatrix} \begin{bmatrix} u \\ w \\ q \\ \theta \end{bmatrix} + \begin{bmatrix} -4.64 (3.6) \\ 62.18 (.33) \\ -9.7 (.01) \\ 0 (.04) \end{bmatrix} \eta \quad (19)$$

Note that all elements and their errors have been rounded to 2 decimal places. If all of the data points are used in the estimation process, the computation time for the above file is around 35 minutes. Clearly this is excessive, but can be reduced to around 1 minute by sub-sampling the data (by invoking “*reduce1*” within “*pro_lof*”) at 10 Hz. The parameter estimates thus obtained are identical (to two decimal places) to those of equation (19).

The estimated and predicted values of the motion variables are shown in Fig. 9. The predicted short period variables, w and q , follow the measured values closely. In contrast the phugoid variables are not as well matched, although the *character* of the estimated responses is similar to the recorded values. The reasonableness of the derivative estimates describing the phugoid may be assessed as discussed previously, but there are several factors which limit the accuracy of the predicted values.

Firstly, equation (18) models only the airframe, and does not account for the influence of the propeller/engine combination on the damping of the long period motion. As a result, the magnitude of the x_u estimate will be greater than expected from an analysis of the airframe alone, since it effectively incorporates both engine and airframe effects. In addition, the influence of the spring in the elevator control circuit is not considered; as the speed varies, the balance between the spring and the dynamic pressure varies accordingly. The elevator position will therefore be changing about its trim value, and the corresponding phugoid motion will differ from that which would follow if the elevator remained fixed at the trim position.

It may be seen from the $\dot{u}$ equation in (19) that the x_q derivative should be discarded since the standard error is significantly larger than the estimate itself. Similarly, the error

in x_η is 78% of the magnitude of the estimate. In theory x_θ should be equal to g , but the estimated value is slightly lower.

In Fig. 5 the elevator to pitch rate frequency response generated by (19) is compared to that estimated using the DFT technique discussed earlier. Overall, the two methods produce similar results, particularly at higher frequencies (above the phugoid). The (relatively small) discrepancies observed at lower frequencies may arise for several reasons:- firstly, the neglected dynamics in the parametric model; secondly the low frequency resolution in the DFT estimate depends directly on the length of the time window.

From the estimated state space model (19), the short period frequency and damping ratio are 2.83 *rad/s* and 0.55 respectively. The phugoid values are 0.12 *rad/s* and 0.27, although for the reasons discussed above, individual derivative estimates should be treated with some caution.

5.3 Additional Comments

The preceding sections demonstrate that useful results can be obtained through the application of filtering and some straightforward estimation techniques. It is possible however to simplify the parametric estimation problem further by noting a few points:-

- a) In theory it is not necessary to estimate the elements of the $\dot{\theta}$ equation.
- b) Using the integral from of equations (9) or (18) removes the requirement to calculate the rate of changes of the states, and thereby avoids the problem of working with noisy differentiated signals. Promising results have been obtained using this technique.

6. MAKING MORE USE OF THE BUILT-IN FUNCTIONS

Although the algorithms presented in Sections 4 and 5 are useful for gaining a basic working knowledge of some parameter identification techniques, they do not necessarily represent the most efficient or flexible way of achieving results.

The aim of the following Sections is therefore to illustrate the use of some built-in functions contained in the signal processing toolbox [13] and the system identification toolbox [14].

6.1 Non-Parametric Frequency Response Identification

The signal processing toolbox function “*fft*” employs a fast Fourier transform (FFT) algorithm to generate harmonic vectors from a given time series. It is therefore computationally more efficient than the straightforward DFT algorithm employed by the author’s “*tfest*” routine.

Function “*tfest1*” (Appendix B) employs “*fft*” to produce estimates of the harmonic vectors and hence the frequency response. It’s operation may be summarised as follows:-

- a) Compute the harmonic components of the input and output signals using “*fft*”.
- b) The ratio of the output to input gives the harmonic response in complex number form.
- c) Convert this to magnitude (*dB*) and phase (*deg*).
- d) Calculate the frequency vector.

For comparison purposes, calculate the elevator to pitch rate frequency response from file *th111334* using both “*tfest1*” and “*tfest*”. The former is substantially faster, but both give identical numerical results (after applying the appropriate phase offsets) .

Alternatively, the system identification toolbox function “*etfe*” will give similar results. This function calculates the gain and phase directly, and also allows the user to perform a smoothing operation on the estimates. However, the character of the achieved frequency responses are *very* sensitive to the value of this smoothing parameter. In addition, although the frequency, gain and phase are estimated, they must be extracted using another function “*getfff*”. In the author’s opinion, there is little advantage to be gained from the combined use of these implementations.

6.2 Parametric Identification Techniques

The system identification toolbox contains a large selection of algorithms for obtaining parametric models from test data. Depending on the algorithm used, the model parameters can be returned in discrete polynomial, continuous polynomial, discrete state-

space, or continuous state-space forms. The default output argument of these functions is the vector estimate of unknown parameters, $\hat{\theta}$, or theta format in *MATLAB*TM parlance.

The following paragraphs consider only a small subset which have been shown to give reasonable practical results. Mathematical discussions are included to aid the user in understanding how to gain the most from the functions, but the implementation of the algorithms themselves is not considered. Example listings of the syntax are included in Appendix C.

6.2.1 The OE Function

The “oe” function [x] models a discrete-time input/output relationship of the form:-

$$y(t) = \frac{B(q)}{F(q)} u(t - nk) + e(t) \quad (20)$$

where $y(t)$ is the output signal, $u(t)$ the input, nk the number of delays, and $e(t)$ the noise. $B(q)$ and $F(q)$ are the polynomials whose coefficients are to be estimated. If the noise and the delays can be effectively removed using the techniques discussed previously, then (20) reduces to:-

$$y(t) = \frac{B(q)}{F(q)} u(t) \quad (21)$$

Expanding the polynomials:-

$$\begin{aligned} B(q) &= b_1 + b_2 q^{-1} + \dots + b_{nb} q^{-nb+1} \\ F(q) &= 1 + f_1 q^{-1} + \dots + f_{nf} q^{-nf} \end{aligned} \quad (22)$$

and substituting (22) into (21) and cross multiplying:-

$$y(t) + f_1 q^{-1} y(t) + \dots + f_{nf} q^{-nf} y(t) = b_1 u(t) + b_2 q^{-1} u(t) + \dots + b_{nb} q^{-nb+1} u(t) \quad (23)$$

Since q^{-1} etc., are just delay operators, expression (23) may be written:-

$$y(t) + f_1 y(t-1) + \dots + f_{nf} y(t-nf) = b_1 u(t) + b_2 u(t-1) + \dots + b_{nb} u(t-nb+1) \quad (24)$$

The coefficients $[f_1 \dots f_{nf}]$, $[b_1 \dots b_{nb}]$ are estimated by an iterative procedure, the basic *MATLAB* syntax being:-

$$th = oe(z, nn, maxiter, tol, lim, maxsize, T) \quad (25)$$

where $z = [y \ u]$ and $nn = [nb \ nf \ nk]$. To see how these are formulated, consider the problem of estimating the short period dynamics:-

$$\frac{q(s)}{\eta(s)} = \frac{k_q(s + 1/T_{\theta 2})}{s^2 + 2\zeta_s \omega_s s + \omega_s^2} \quad (26)$$

In this case, $z = [q \ \eta]$, the matrix of pitch rate and elevator samples, but it is not immediately obvious how nn is derived from (26). The relationship becomes clear if the continuous transfer function is converted to discrete-time form (via a Tustin transformation, for example):-

$$\frac{q(z)}{\eta(z)} = \frac{Az^{-2} + Bz^{-1} + C}{Dz^{-2} + Ez^{-1} + F} \quad (27)$$

Cross multiplying equation (27):-

$$\begin{aligned} Dz^{-2}q(z) + Ez^{-1}q(z) + Fq(z) &= Az^{-2}\eta(z) + Bz^{-1}\eta(z) + C\eta(z) \\ \therefore \frac{D}{F}z^{-2}q(z) + \frac{E}{F}z^{-1}q(z) + q(z) &= \frac{A}{F}z^{-2}\eta(z) + \frac{B}{F}z^{-1}\eta(z) + \frac{C}{F}\eta(z) \end{aligned} \quad (28)$$

Now since $z^{-n}q(z)$ is simply $q(t-n)$, equation (28) can be rewritten:-

$$\frac{D}{F}q(t-2) + \frac{E}{F}q(t-1) + q(t) = \frac{A}{F}\eta(t-2) + \frac{B}{F}\eta(t-1) + \frac{C}{F}\eta(t) \quad (29)$$

The components of the input argument can be deduced by comparing equations (29) and (24), from whence $nn = [3 \ 2 \ 0]$. The remaining variables are effectively tuning parameters, and their use is illustrated in Appendix C. Consequently, invoking “*oe*” with the appropriate input arguments returns the estimates of the discrete polynomial coefficients in theta format.

To examine the quality of the estimates, use the “*present*” command. This displays the coefficients together with their standard deviations. *NB:-* the standard deviations should be significantly smaller than the estimates; if they are comparable or greater, think again.

To convert from theta format to discrete transfer function form, invoke “*th2tf*”. Do not simply copy the values output by “*present*” since these have been rounded off.

To convert the discrete transfer function to the continuous form of equation (26), use the control systems toolbox “*d2cm*” function. This may result in a (spurious) second

order numerator term. This is a numerical effect caused by the transformation, and the coefficient is usually much smaller than those of the constant or first order terms. As such it may be discarded.

As an example, consider the short period excitation *th105910* discussed in Section 5.2.2. Following the above procedure, the estimated elevator to pitch rate transfer function is:-

$$\frac{q(s)}{\eta(s)} = \frac{-5.7217(s+1.2142)}{s^2 + 2.5556s + 3.1301} \quad (31)$$

which appears to be a reasonable match with the previous approximation of equation (15).

6.2.2 The PEM Function

The “*pem*” function may be used to estimate the parameters of a continuous time state space model; it is essentially a multivariable implementation of “*oe*”. Although the *MATLAB*TM syntax is straightforward, it is somewhat lengthy; see Appendix C for details. Generally speaking, the steps required are as follows:

- a) Define the unknown parameters (in this case the stability and control derivatives) in state space form.
- b) Define the initial conditions and the initial parameter estimates; the latter may be obtained from the Jetstream simulation or from “*ls_sp*”.
- c) Construct the matrix of measured states and inputs.
- d) Using the results of a) through c), convert the parameter estimation problem to theta format, and set the sample interval.
- e) Invoke “*pem*” to compute the updated parameter estimates.

Using file *th105910*, the estimated state and input matrices are:-

$$\begin{bmatrix} \dot{w} \\ \dot{q} \end{bmatrix} = \begin{bmatrix} -0.9722 & 56.1436 \\ -0.0262 & -1.5222 \end{bmatrix} \begin{bmatrix} w \\ q \end{bmatrix} + \begin{bmatrix} 19.8583 \\ -5.8092 \end{bmatrix} \eta \quad (31)$$

The above are comparable to the estimates obtained by the much simpler “*ls_sp*” function, equation (14). As far as the user is concerned, the justification for using a more complex method is that the resulting parameter estimates should have better statistical properties, e.g. lower standard deviations and a reduced bias. That may be the case, but

note that the standard errors in equation (14) are already relatively small. In addition, the value of z_η predicted by “*pem*” is no more realistic than that calculated by “*ls_sp*”.

In making the above comparison, the aim is not to discourage the use of more sophisticated (or even new!) techniques, since they can often encompass many more aspects of the problem than the author’s functions are able to. Rather, the aim is to show that the combination of basic signal processing and elementary identification theory can produce reasonable practical results which are of comparable quality to those obtained by more complex means.

7. CONCLUSIONS AND SUGGESTIONS FOR FUTURE WORK

Recent research in the field of parameter identification (PI) at the College of Aeronautics has produced some interesting work, but has not resulted in tools which may be used as a matter of course for the solution of linear PI problems. Using stability and control data recorded from Jetstream G-NFLC as an example, it is shown that the application of elementary theory can produce practical tools which may be used by non-specialists. More specifically:-

- The use of power spectral density methods in conjunction with zero-phase filtering results in signals which are of sufficient quality to use in the estimation process.
- A basic discrete Fourier transform algorithm produces reasonable frequency response estimates from flight data.
- A simple least squares equation error implementation produces acceptable parametric models from flight data.
- Parametric and non-parametric models can be realised more quickly using built-in *MATLABTM* functions.
- As far as the linear PI problem is concerned, all of the required tools are available commercially.

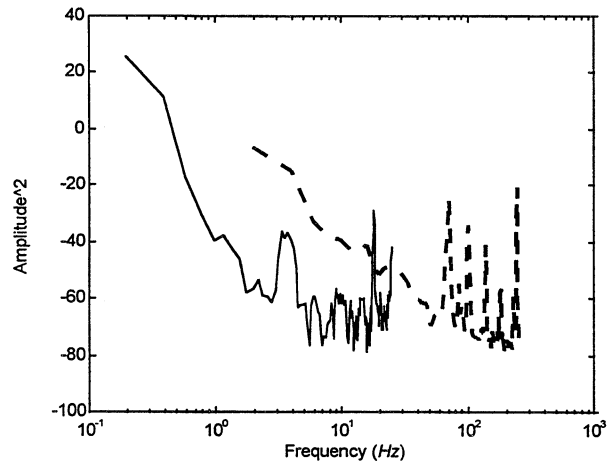
Concerning the last point, the algorithms implemented in [14] are derived directly from the theory presented in [10]. In the author’s opinion, the simplest route to realising a flexible, powerful tool which may be used on a daily basis is to investigate and document the applicability of the functions contained in the system identification toolbox.

REFERENCES

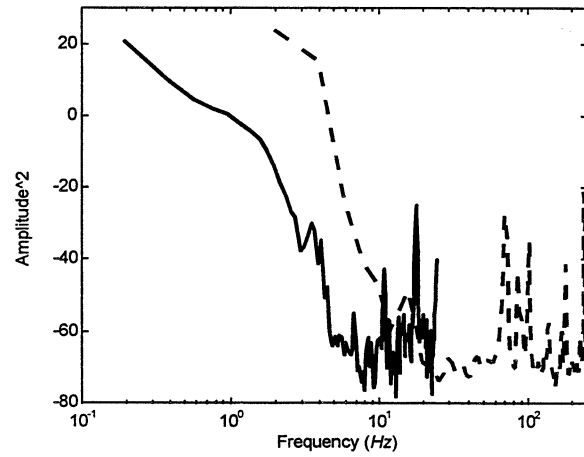
1. Pallister, K. *Investigation of the Lateral Stability Characteristics of Aeroplanes with Bluff Upswept Rear Fuselages*. College of Aeronautics DCAe thesis, 1967.
2. Martin, C.A. *Further Investigation of the Lateral Stability Characteristics of an Aircraft with a Bluff Upswept Rear Fuselage*. College of Aeronautics DCAe thesis, 1968.
3. Klein, V. *Evaluation of the Basic Performance Characteristics of an Instrumentation System*. Cranfield Institute of Technology, College of Aeronautics Report, Aero. 21, 1973.
4. Klein, V. *Parameter Identification Applied to Aircraft*. Cranfield Institute of Technology, College of Aeronautics, PhD thesis, 1973.
5. Foster G.W. *The Identification of Aircraft Stability and Control Parameters in Turbulence*. Cranfield Institute of Technology, College of Aeronautics, PhD thesis, 1982.
6. Hoff, J.C. *Aircraft Parameter Estimation by Estimation-Before-Modelling Technique*. Cranfield University, College of Aeronautics, PhD thesis, 1995.
7. Hinds, H.A. *The Application of a Modified Stepwise Regression Method to the Estimation of Aircraft Stability and Control Derivatives*. Cranfield University, College of Aeronautics, PhD thesis, 1996.
8. Baek, Y-H. *An Experimental Review of Some Aircraft Parameter Identification Techniques*. Cranfield University, College of Aeronautics, PhD thesis, 1998.
9. Soderstrom, T. & Stoica, P. *System Identification*. Prentice-Hall, 1994.
10. Ljung, L. *System Identification - Theory for the User*. Prentice-Hall, 1987.
11. Stevens, J.R.A. *Data Recording and POSTFLT Analysis Software*, User Guide, Version 2.0. Cranfield Aerospace Ltd., 1995.
12. *MATLABTM* Version 4.2c.1.
13. *MATLABTM* Signal Processing Toolbox, Version 3.0b.
14. *MATLABTM* System Identification Toolbox, Version 3.0a.
15. *MATLABTM* Control Systems Toolbox, Version 3.0b.
16. Maine, R.E. & Iliff, K.W. *Identification of Dynamic Systems - Applications to Aircraft, Part 1: The Output Error Approach*. AGARDograph No. 300, Vol. 3, 1986.
17. Cook, M.V. *Flight Dynamics Principles*. Arnold, 1997.
18. Anon. *Fisher Controls A.C. and D.C. Pick-Offs*. Technical Brochure, Fisher Controls, Lewisham, England.
19. Anon. *LTN-90 Software Requirements Document*. Revision 3, Litton Aero Products, 1989.
20. Golten, J. *Understanding Signals and Systems*. McGraw Hill, 1997.
21. Mendel, J.M. *Discrete Techniques of Parameter Estimation: The Equation Error Formulation*. Marcel Dekker Inc., 1973.
22. Mullen, G.J. *Investigation of Handley-Page Jetstream Handling During Landing*. Submitted to The Aeronautical Journal.



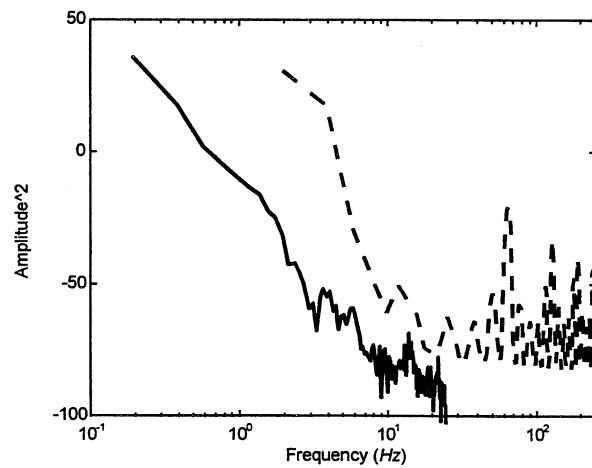
Figure 1. Cranfield University Jetstream and Bulldog



(a) *AlphaTru*

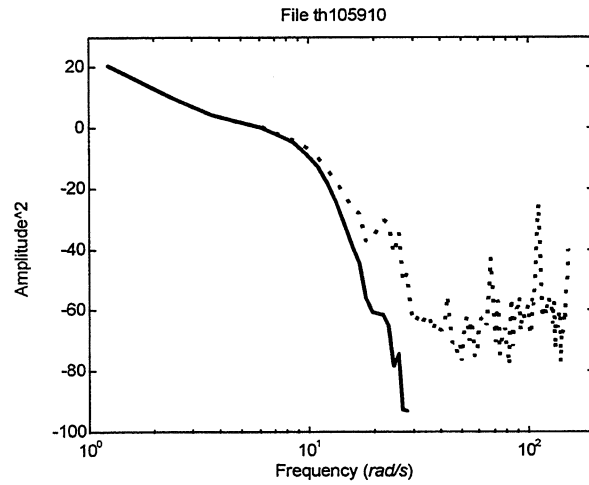


(b) *Elevator*

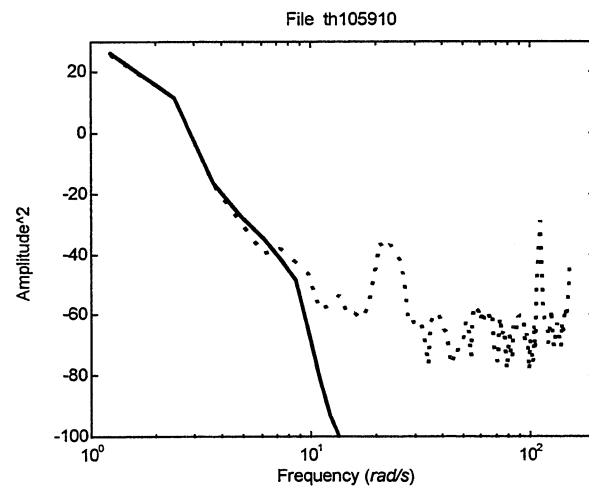


(c) *IRSPTcRt*

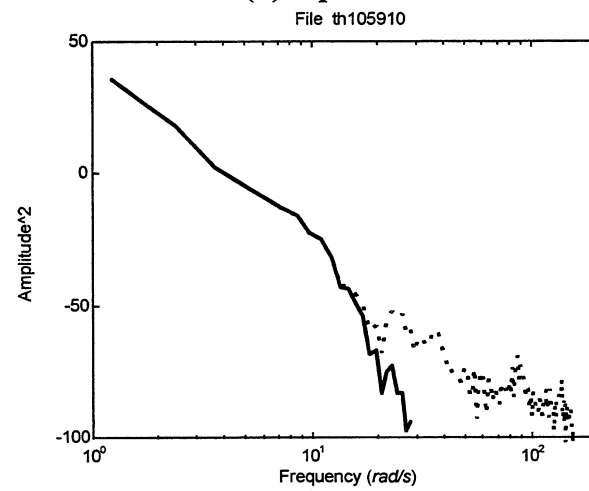
**Figure (2). Power spectral densities
50 Hz sample rate(-), 500 Hz sample rate(--)**



(a) Elevator



(b) AlphaTru



(c) IRSPTcRt

**Figure 3. Power spectral densities, short period mode
filtered(-), unfiltered(--)**

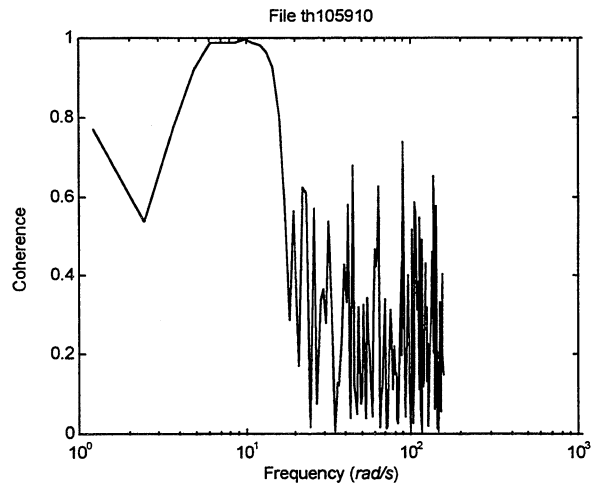


Figure 4. *IRSPtcRt* / *Elevator* coherence, short period mode

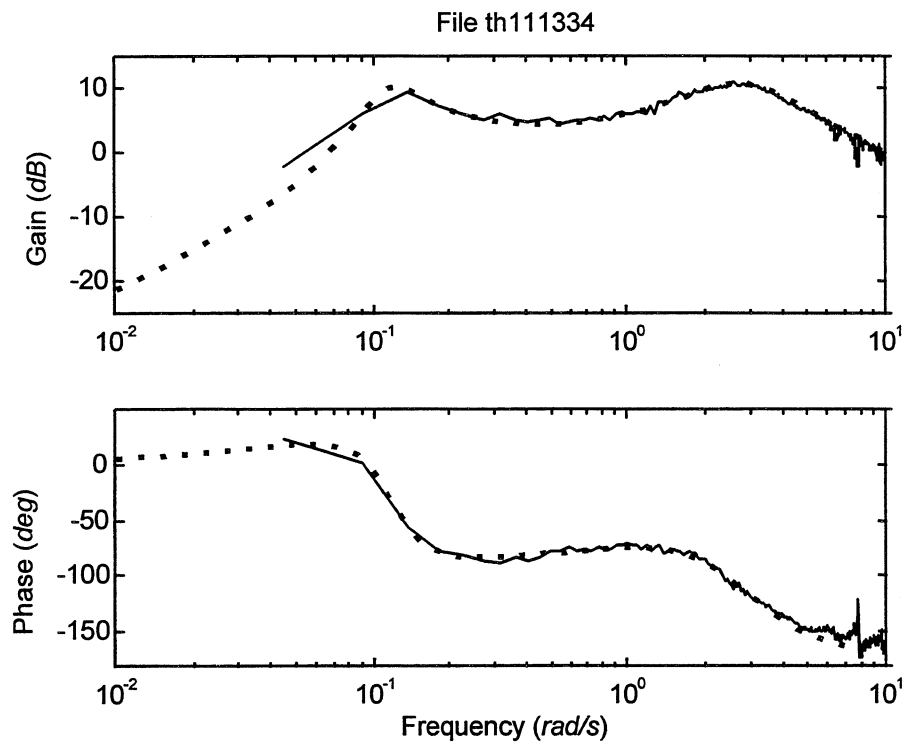


**Figure 5. q/η frequency response estimates
discrete Fourier transform(-), parametric model(--)**

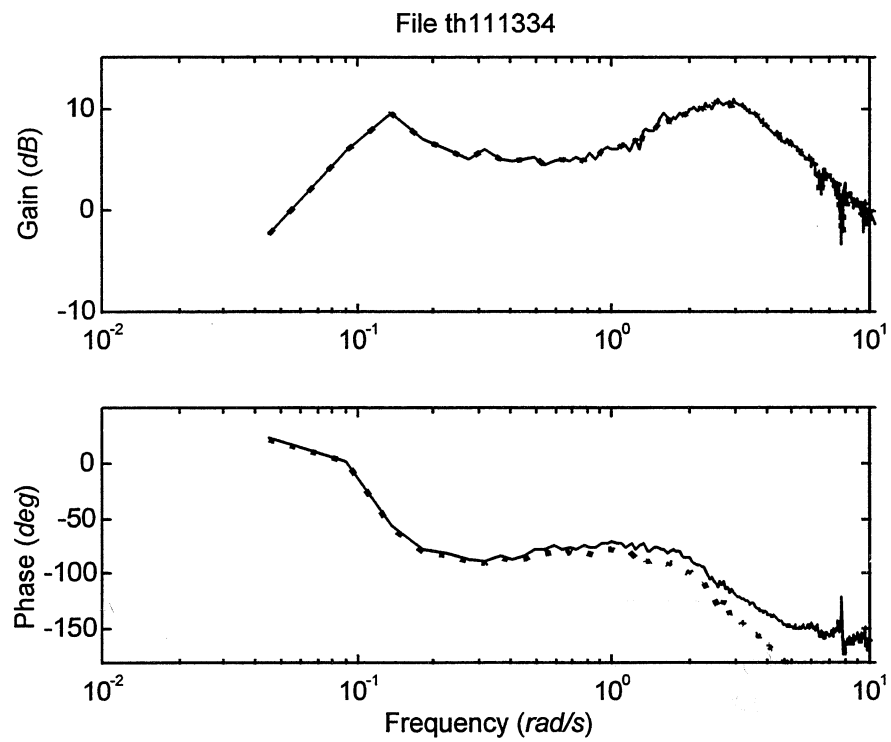
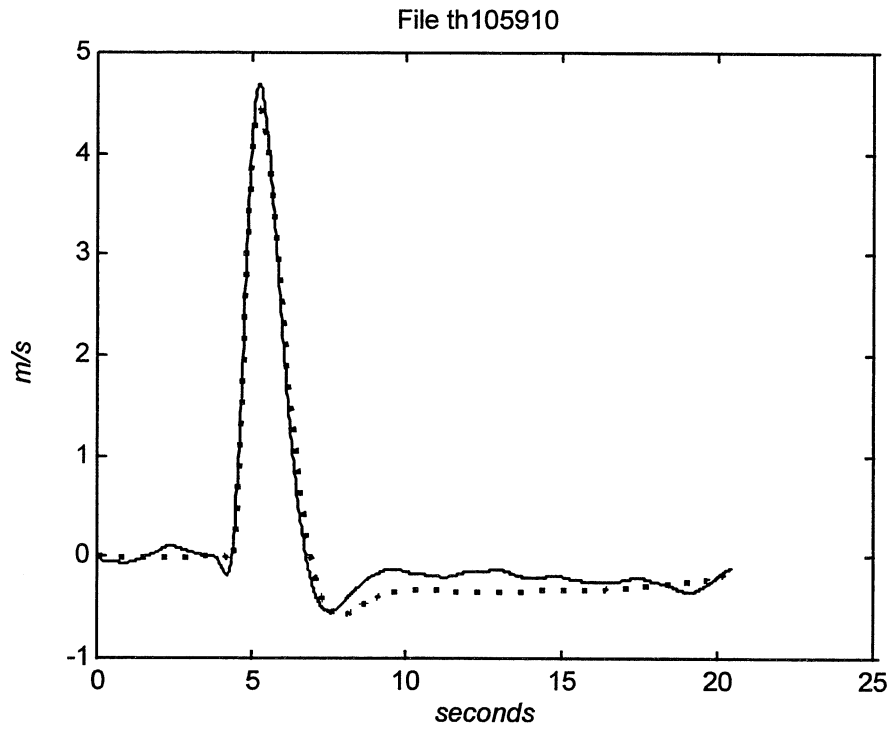
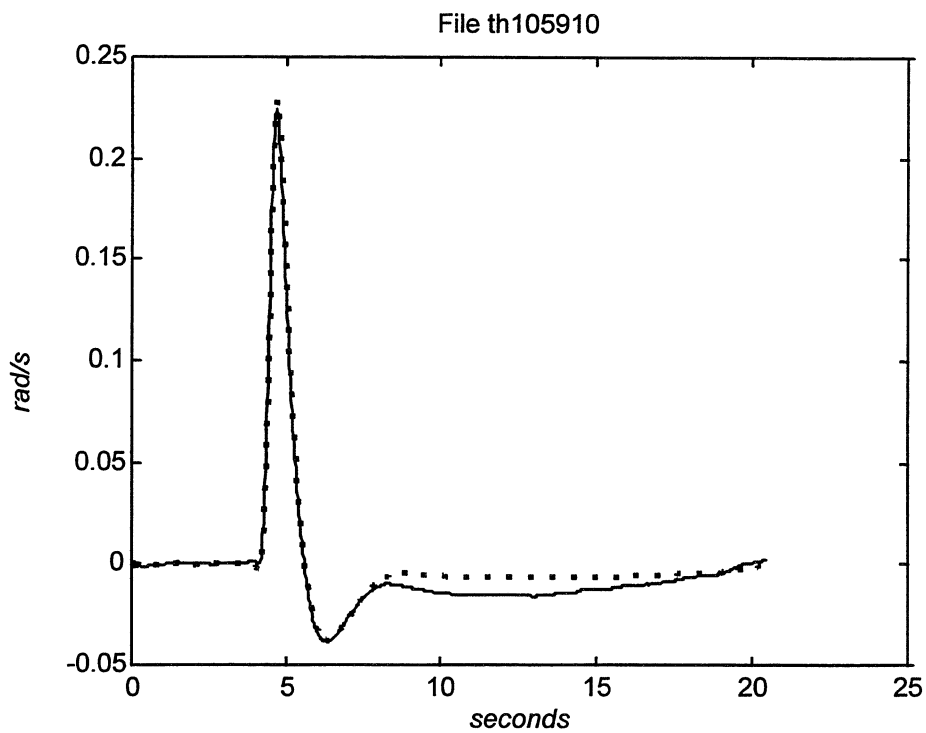


Figure 6. q/η frequency response estimates
processed measurements(-), unprocessed(--)

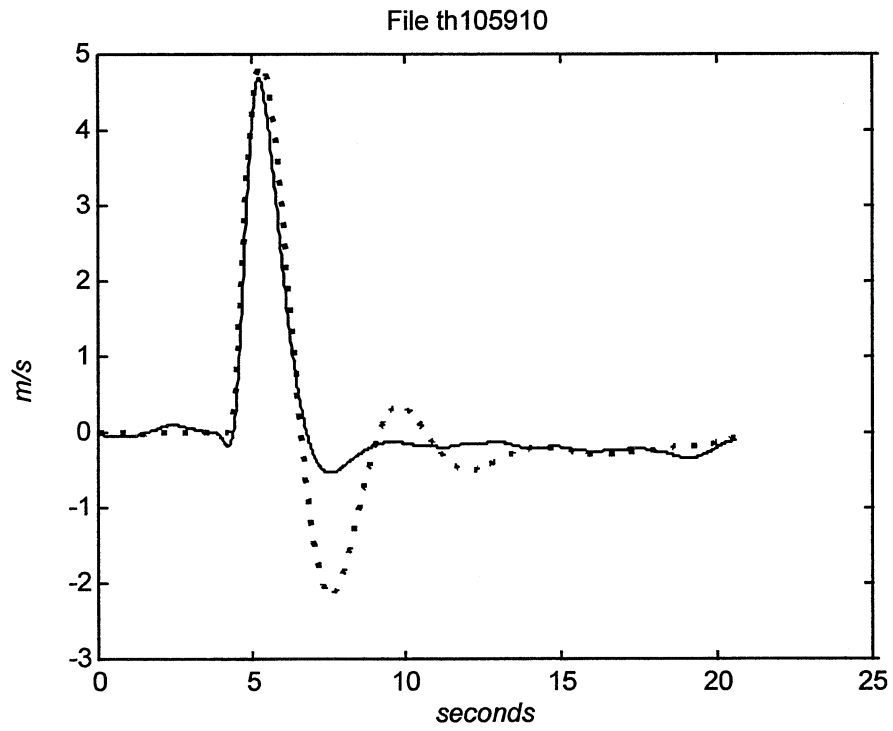


(a) w perturbation

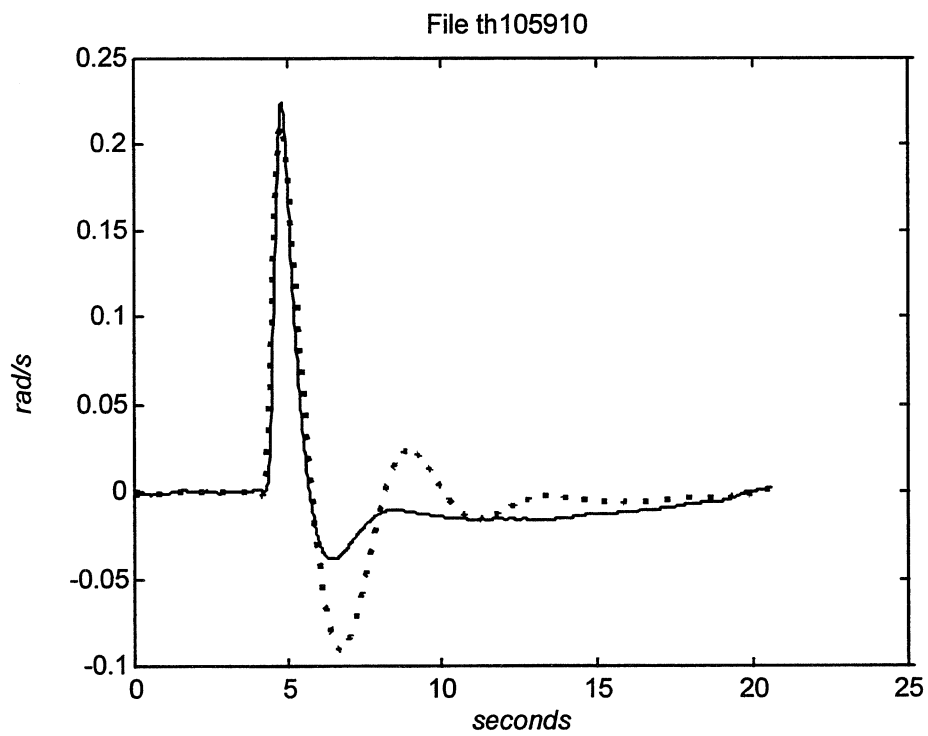


(b) q perturbation

Figure 7. Measured(-) and estimated(--) short period response



(a) w perturbation



(b) q perturbation

Figure 8. Measured(-) and estimated(--) short period response with transport delay in pitch rate

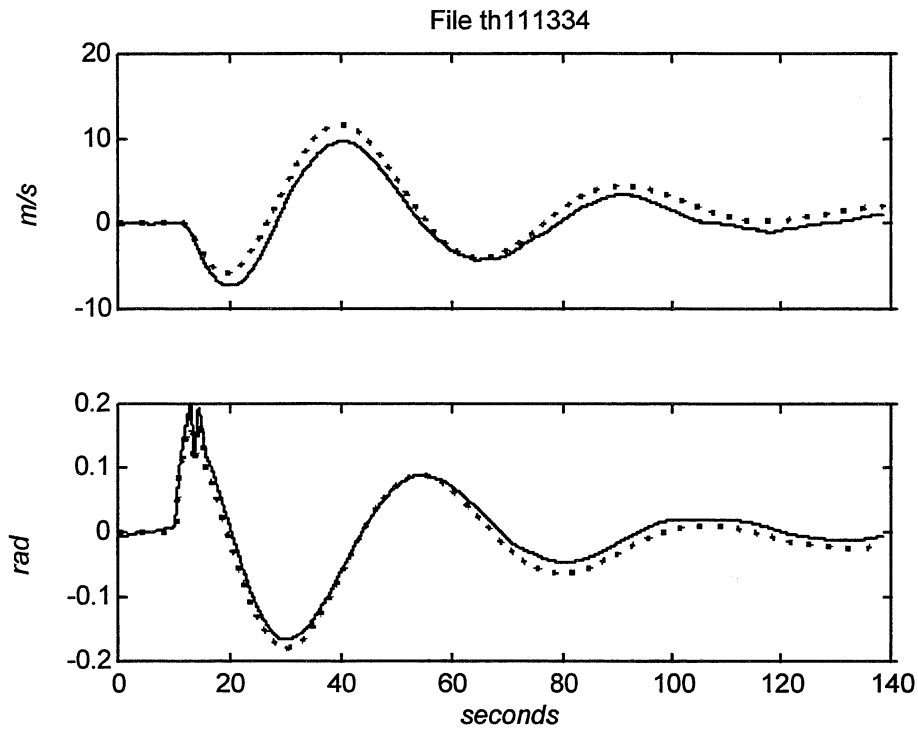


Figure 9(a): Estimated(--) and measured(-) airspeed and pitch attitude perturbations

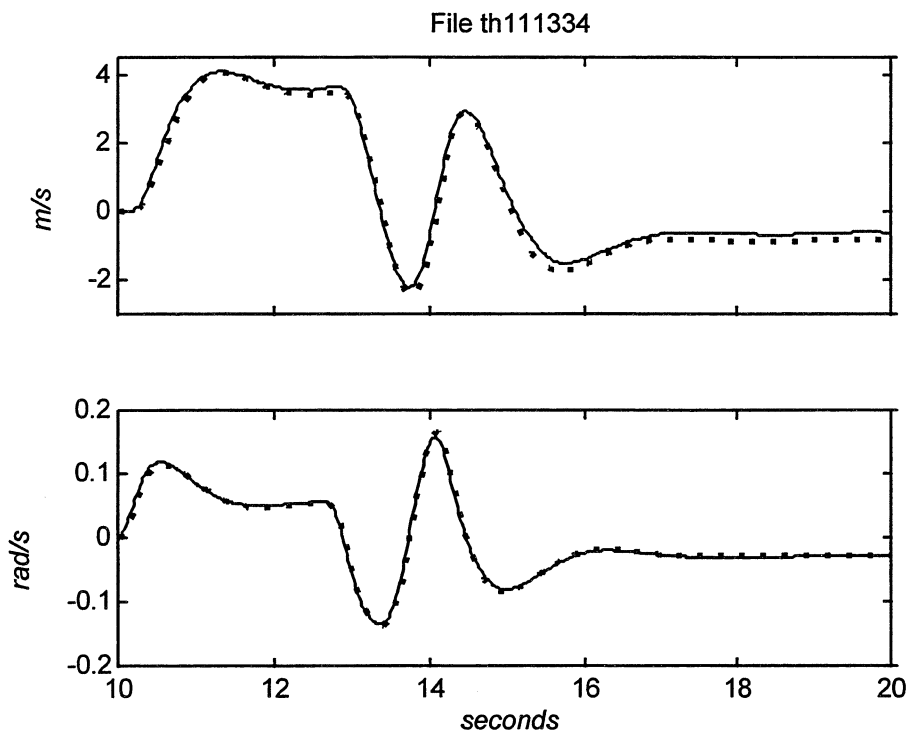


Figure 9(b): Estimated(--) and measured(-) vertical velocity and pitch rate perturbations

Appendix A

List of Data Files

File	Comment	Sample Rate (Hz)	Configuration	Aircraft Mass (kg)	cg (% $\bar{c}$)
tu151817	short period	500	150 <i>kts</i> , cruise	-	-
tu151910	short period	500	150 <i>kts</i> , cruise	-	-
th105910	short period	50	115 <i>kts</i> , landing	-	31.8
th111334	short period + phugoid	50	150 <i>kts</i> , cruise	-	23.5

The incomplete table above is a prime example of bad practice; if you forget to record mass and cg information, it might cause headaches at a later stage.

Appendix B

Software Listings

<i>reduce1.m</i>	Sub-samples a data set.
<i>pro_sp1.m</i>	Short period signal processing.
<i>pro_lof.m</i>	Short period and phugoid signal processing.
<i>tfest.m</i>	Frequency response estimation using a discrete Fourier transform.
<i>tfest1.m</i>	Frequency response estimation using a fast Fourier transform.
<i>ls_sp.m</i>	Short period motion stability and control derivative estimation using simple least squares algorithm.
<i>ls_long.m</i>	Short period and phugoid motion stability and control derivative estimation using a simple least squares algorithm.


```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% reduce1: Sub-samples a data set. The data set must have been passed %
% through an appropriate low-pass filter before sub-sampling. %
%
% INPUTS: Data file. %
%
% OUPUTS: Reduced data file. %
%
% SYNTAX: [shorty]=reduce(data); %
%
% AUTHOR: G.J. Mullen for the NFLC, Cranfield. %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [shorto] = reduce1(data)

for i = 1:5:(length(data)), % At present selects every 5th point.
    new = data(i,:);
    short = [short; new];
end
%
if nargin==1
    shorto = short;
end
%
```


Pro_sp1.m

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% pro_sp1 :Short period response, processes data from POSTFLT.
% Removes time delays, converts to m/s, rad, rad/s.
% Removes steady offsets.
% filters and differentiates signals.
%
% INPUTS : Time,AlphaTru,IRSptcRt,Elevator,EAS,Sample No.
%
% OUTPUTS: wdot, qdot, w, q, eta.
%
% SYNTAX : [t,wdot,qdot,w,q,eta] =
%           pro_sp1(Time,AlphaTru,IRSptcRt,Elevator,EAS,SamNu)
%
% AUTHOR : G.J. Mullen for the NFLC, Cranfield.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [to,wdoto,qdoto,wo,qo,etao] = pro_sp1(Time,AlphaTru,IRSptcRt,Elevator,EAS,SamNu)

% Filter variables.

[b,a] = butter(3,0.0464);
AlphaTru = filtfilt(b,a,AlphaTru);

[b,a] = butter(2,0.095);
Elevator = filtfilt(b,a,Elevator);

[b,a] = butter(3,0.127);
IRSptcRt = filtfilt(b,a,IRSptcRt);

% Convert all variables to SI units and calculate trim airspeed.

```

```

AlphaTru = (pi/180)*AlphaTru;
IRSptcRt = (pi/180)*IRSptcRt;
Elevator = (pi/180)*Elevator;
EAS = 0.51444*EAS;

Uo = mean(EAS(1:SamNu)); % Trim airspeed

% Remove initial offsets.

Elevator = Elevator - mean(Elevator(1:SamNu));
AlphaTru = AlphaTru - mean(AlphaTru(1:SamNu));
IRSptcRt = IRSptcRt - mean(IRSptcRt(1:SamNu));

% Construct w

w = Uo*tan(AlphaTru);

% Remove time delays.

l = length(Time);
IRSptcRt = IRSptcRt(7:l);
Elevator = Elevator(1:(l-6));
Time = Time(1:(l-6));
w = w(1:(l-6));

% Calculate rate of change of signals, and remove any initial offsets.

wdot = diff(w)./diff(Time);
wdot = [0; wdot];
wdot = wdot - mean(wdot(1:SamNu));

```

```

qdot = diff(IRSPTcRt)./diff(Time);
qdot = [0; qdot];
qdot = qdot - mean(qdot(1:SamNu));

% Divide each value by maximum

%wdot      = wdot./max(abs(wdot));
%qdot      = qdot./max(abs(qdot));
%w         = w./max(abs(w));
%q         = q./max(abs(q));
%Elevator = Elevator./max(abs(Elevator));

if nargout==6
    to = Time;
    wdoto = wdot;
    qdoto = qdot;
    wo = w;
    qo = IRSPTcRt;
    etao = Elevator;
end

```



```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% pro_lof :Short period and phugoid response, processes data from POSTFLT.
%
% Removes time delay, converts to m/s, rad, rad/s.
%
% Removes steady offsets.
%
% filters and differentiates signals.
%
%
% INPUTS : Time, EAS, AlphaTru, IRSptcRt, IRSPitch, Elevator, Sample No.
%
%
% OUTPUTS: t, udot, wdot, qdot, thdot, u, w, q, theta, eta.
%
%
% SYNTAX : [t,udot,wdot,qdot,thdot,u,w,q,theta,eta] =
%           pro_lof(Time,EAS,AlphaTru,IRSptcRt,IRSPitch,Elevator,SamNu)
%
%
% AUTHOR : G.J. Mullen for the NFLC, Cranfield.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```

```

function [to,udoto,wдото,qдото,thдото,uo,wo,qo,thetao,etao] = pro_lof(Time,EAS,AlphaTru,IRSptc
Rt,IRSPitch,Elevator,SamNu)

```

```

% Filter measurements.

```

```

[b,a] = butter(2,0.127);
Elevator = filtfilt(b,a,Elevator);
% NOTE: Filter characteristics
% are not unique. Modify to suit
% specific requirements.

```

```

[b,a] = butter(2,0.127);
IRSptcRt = filtfilt(b,a,IRSptcRt);

```

```

[b,a] = butter(2,0.127);
AlphaTru = filtfilt(b,a,AlphaTru);

```

```

[b,a] = butter(2,0.127);

```

```

EAS = filtfilt(b,a,EAS);

% Convert all variables to SI units and construct Uo.

AlphaTru = (pi/180)*AlphaTru;
IRSPtcRt = (pi/180)*IRSPtcRt;
IRSPitch = (pi/180)*IRSPitch;
Elevator = (pi/180)*Elevator;
EAS      = 0.51444*EAS;
Uo       = mean(EAS(1:SamNu)); % Trim airspeed

% Subtract Initial offsets.

Elevator = Elevator - mean(Elevator(1:SamNu));
AlphaTru = AlphaTru - mean(AlphaTru(1:SamNu));
IRSPtcRt = IRSPtcRt - mean(IRSPtcRt(1:SamNu));
IRSPitch = IRSPitch - mean(IRSPitch(1:SamNu));
EAS      = EAS - mean(EAS(1:SamNu));

% Construct w

w = Uo*tan(AlphaTru);

% Remove time delays.

l = length(Time);
IRSPitch = IRSPitch(6:(l-1)); % VALID FOR 50 Hz SAMPLE RATE ONLY.
IRSPtcRt = IRSPtcRt(7:1); % MUST BE MODIFIED BY USER.
Elevator = Elevator(1:(l-6));
Time     = Time(1:(l-6));
w        = w(1:(l-6));

```

```

EAS      = EAS(1:(1-6));

% Calculate rate of change and filter if necessary.

udot     = diff(EAS)./diff(Time);
udot     = [0; udot];
[b,a]    = butter(2,0.016);
udot     = filtfilt(b,a,udot);
udot     = udot - mean(udot(1:SamNu));

wdot     = diff(w)./diff(Time);
wdot     = [0; wdot];
wdot     = wdot - mean(wdot(1:SamNu));

qdot     = diff(IRSptcRt)./diff(Time);
qdot     = [0; qdot];
qdot     = qdot - mean(qdot(1:SamNu));

thdot    = IRSptcRt; % Small perturbation model

% Sub sample for every 5th element of the variables.

%IRSPitch = reduce1(IRSpitch);
%IRSptcRt = reduce1(IRSptcRt);
%Elevator = reduce1(Elevator);
%w        = reduce1(w);
%EAS      = reduce1(EAS);

%udot     = reduce1(udot);
%wdot     = reduce1(wdot);
%qdot     = reduce1(qdot);

```

```
%thdot = reduce1(thdot);  
%Time = reduce1(Time);  
  
if nargout==10  
    to = Time;  
    udoto = udot;  
    wdoto = wdot;  
    qdoto = qdot;  
    thdoto = thdot;  
    uo = EAS;  
    wo = w;  
    qo = IRSptcRt;  
    thetao = IRSPitch;  
    etao = Elevator;  
  
end
```

Tfest.m

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% tfest : Estimates a frequency response using the discrete
%         Fourier transform of output/input data.
%         Assumes all input data has been processed by
%         "pro_spl" or "pro_lof".
%
% INPUTS : Input (u), output (y), time (t),
%         sample interval (Ts, seconds)
%         highest desired frequency (fmax, rad/s).
%
% OUTPUTS : Magnitude (dB), phase (deg), frequency (rad/s).
%
% SYNTAX : [mag,phase,freq] = tfest(u,y,t,Ts,fmax);
%
% AUTHOR : G.J. Mullen for the NFLC, Cranfield.
%
% COMMENTS :See Ch.6 of "Understanding Signals & Systems"
%           by Golten for background theory.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [mago,phaseo,freqo] = tfest(u,y,t,Ts,fmax)

if exist('re_in')
    clear re_in
end
if exist('im_in')
    clear im_in
end
if exist('re_out')
    clear re_out
end

```

```

if exist('im_out')
    clear im_out
end
if exist('fr_input')
    clear fr_input
end
if exist('fr_output')
    clear fr_output
end

% Calculate number of samples
N = length(u);

% Calculate maximum harmonic and discrete frequency vector.
k_max = fmax*t(N)/(2*pi);
k_max = ceil(k_max);

k     = 0:1:k_max;
freq  = (2*pi*k)/t(N);

clear k % Avoids confusion below.

% Compute real and imaginary components for each harmonic of the input data.
% Outer loop deals with harmonic number. MATLAB must start from
% first element, zeroth element not allowed.

for k = 0:k_max
    dth = (k*2*pi)/N;
    re_in(k+1) = 0;

```

```

    im_in(k+1) = 0;

% Inner loop sums for each sample value. Cannot sum from n = 0 to (N-1)
% because the zeroth element is not allowed in MATLAB.

    for n = 0:(N-1)
        ang = n*dth;
        re_in(k+1) = re_in(k+1) + u(n+1)*cos(ang);
        im_in(k+1) = im_in(k+1) - u(n+1)*sin(ang);
    end % of inner loop.

    re_in(k+1) = re_in(k+1)*Ts;
    im_in(k+1) = im_in(k+1)*Ts;
end % of outer loop.

re_in = re_in';
im_in = im_in';

% Frequency domain representation of input.

fr_input = re_in+i*im_in;

% REPEAT FOR OUTPUT DATA.

for k = 0:k_max
    dth = (k*2*pi)/N;
    re_out(k+1) = 0;
    im_out(k+1) = 0;

for n = 0:(N-1)
    ang = n*dth;

```

Tfest.m

```

re_out(k+1) = re_out(k+1) + y(n+1)*cos(ang);
im_out(k+1) = im_out(k+1) - y(n+1)*sin(ang);
end % of inner loop.

```

```

re_out(k+1) = re_out(k+1)*Ts;
im_out(k+1) = im_out(k+1)*Ts;
end % of outer loop.

```

```

re_out = re_out';
im_out = im_out';

```

```

% Frequency domain representation of output.

```

```

fr_output = re_out+i*im_out;

```

```

% Frequency response estimate.

```

```

fr_resp = fr_output./fr_input;
mag = 20*log10(abs(fr_resp));
phase = (180/pi)*unwrap(angle(fr_resp));% Check phase using your knowledge

```

```

% of the problem. Constant offset must
% be added to make low frequency 0 deg.

```

```

if nargout==3
    mago = mag;
    phaseo = phase;
    freqo = freq;
end

```

Tfest1.m

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% tfest1 : Estimates a frequency response using the fast
%          Fourier transform of output/input data.
%          Assumes all input data has been processed by
%          "pro_spl" or "pro_lof".
%
% INPUTS : Input (u), output (y), time (t).
%
% OUTPUTS : Magnitude(dB), phase(deg), frequency (rad/s).
%
% SYNTAX : [mag,phase,freq] = tfest1(u,y,t);
%
% AUTHOR : G.J. Mullen for the NFLC, Cranfield.
%
% COMMENT: Uses the signal processing toolbox function "fft"
%          %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [mago,phaseo,freqo] = tfest1(u,y,t)

% Compute harmonic components of the input and output. Then the ratio.

ip = fft(u);
op = fft(y);

fresp = op./ip;

% Calculate magnitude and phase.

mag = 20*log10(abs(fresp));
phase = (180/pi)*unwrap(angle(fresp)); % Check phase using your knowledge
                                         % of the problem. Constant offset must

```

Tfest1.m

% be added to make low frequency 0 deg.

% Now the frequency vector.

k = 0:1:(length(u)-1); % The harmonic numbers
freq = (2*k*pi)/max(t); % Frequencies in rad/s.

if nargout==3
 mago = mag;
 phaseo = phase;
 freqo = freq;
end

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% ls_sp : Estimates a reduced order state space
%         short period model of the form
%
%         | wdot | = | zw   zq | | w | + | zeta | eta
%         | qdot | = | mw   mq | | q |   | meta |
%
%         using a simple least squares algorithm.
%         Assumes all data processed by "pro_sp1".
%
% INPUTS : Heave velocity and rate of change (w, wdot),
%          pitch rate and rate of change (q, qdot),
%          elevator (eta) and time (t).
%
% OUTPUTS : Stability and control derivatives and
%          error estimates.
%
% SYNTAX : [A,B,C,D,seA,seB] = ls_sp(wdot,qdot,w,q,eta,t)
%
% AUTHOR : G.J. Mullen for the NFLC, Cranfield
%
% COMMENTS: For background theory see J.M. Mendel,
%          "Discrete Techniques of Parameter Estimation",
%          Marcel Dekker, Inc 1973.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [Ao,Bo,Co,Do,seAo,seBo] = ls_sp(wdot,qdot,w,q,eta,t)

% Form the Z and H matrices.

n = 2*length(wdot)-1;

```

```

N = length(wdot);

for i = 1:2:n
    Z(i) = wdot(.5*(i+1));
    Z(i+1) = qdot(.5*(i+1));

    H(i,1:6) = [w(.5*(i+1)) q(.5*(i+1)) eta(.5*(i+1)) 0 0 0];
    H(i+1,1:6) = [0 0 0 w(.5*(i+1)) q(.5*(i+1)) eta(.5*(i+1))];
end

% Make Z a column vector.

Z = Z';

% Calculate the parameter estimates.

theta = inv(H'*H)*H'*Z;

f = inv(H'*H); % Used later in standard error estimates.

zw = theta(1);
zq = theta(2);
mw = theta(4);
mq = theta(5);
zeta = theta(3);
meta = theta(6);

A = [zw zq; mw mq];
B = [zeta; meta];
C = eye(2);
D = zeros(2,1);

```

```

% Output frequency and damping

damp(A)

% Estimate the standard error for each derivative.

[y,x] = lsim(A,B,C,D,eta,t);      % predict the outputs from estimated model.
figure(1),plot(t,w,t,y(:,1),'--g') % compare estimated and actual w.
xlabel('seconds'),ylabel('m/s'),title('Measured(-) and Predicted(--g) w')
figure(2),plot(t,q,t,y(:,2),'--g') % compare estimated and actual q.
xlabel('seconds'),ylabel('rad/s'),title('Measured(-) and Predicted(--g) q')

sqe_w = [y(:,1) - w].*[y(:,1) - w]; % square of errors for w.
ssqe_w = sum(ssqe_w);               % sum of square of errors for w.
sigma_w =sqrt((ssqe_w)/(N-3));      % standard deviation of w equation.

sqe_q = [y(:,2) - q].*[y(:,2) - q];
size(sqe_q);
ssqe_q = sum(ssqe_q);
sigma_q =sqrt((ssqe_q)/(N-3));

for j = 1:6
    g(j,j) = sqrt(f(j,j));
end

g = diag(g);

sew = sigma_w * g(1:3);
seq = sigma_q * g(1:3);

```

% Put error components in the same form as the A, B matrices.

```
seA = [sew(1) sew(2); seq(1) seq(2)];  
seB = [sew(3); seq(3)];
```

```
if nargin==6  
    Ao = A;  
    Bo = B;  
    Co = C;  
    Do = D;  
    seAo = seA;  
    seBo = seB;  
  
end
```

ls_long.m

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% ls_long : Estimates the A and B matrices of the longitudinal
% state space model
%
%      | udot | | xu xw xq xth | | u | | xeta |
%      | wdot | | zu zw zq zth | | w | | zeta | eta
%      | qdot | = | mu mw mq mth | | q | + | meta |
%      | thdot | | 0 0 1 0 | | th | | 0 |
%
% using a simple least squares algorithm. Assumes all data processed
% by "pro_lof".
%
% INPUTS : Airspeed and rate of change (u, udot),
%          heave velocity and rate of change (w, wdot),
%          pitch rate and rate of change (q, qdot),
%          pitch attitude (th), elevator (eta), time (t).
%
% OUTPUTS : Stability and control derivatives and
%          error estimates.
%
% SYNTAX : [A,B,C,D,seA,seB] = ls_long(udot,wdot,qdot,thdot,u,w,q,th,eta,t)
%
% AUTHOR : G.J. Mullen for the NFLC, Cranfield
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [Ao,Bo,Co,Do,seA,seB] = ls_long(udot,wdot,qdot,thdot,u,w,q,th,eta,t)

% Form the Z and H matrices.

if exist('Z')
    clear Z

```

```

end

if exist('H')
    clear H
end

n = 4*length(u)-3;
N = length(t);

for i = 1:4:n
    Z(i) = udot(.25*(i+3));
    Z(i+1) = wdot(.25*(i+3));
    Z(i+2) = qdot(.25*(i+3));
    Z(i+3) = thdot(.25*(i+3));

    H(i,1:20) = [u(.25*(i+3)) w(.25*(i+3)) q(.25*(i+3)) th(.25*(i+3)) eta(.25*(i+3)) zer
os(1,15)];
    H(i+1,1:20) = [zeros(1,5) u(.25*(i+3)) w(.25*(i+3)) q(.25*(i+3)) th(.25*(i+3)) eta(.25
*(i+3)) zeros(1,10)];
    H(i+2,1:20) = [zeros(1,10) u(.25*(i+3)) w(.25*(i+3)) q(.25*(i+3)) th(.25*(i+3)) eta(.2
5*(i+3)) zeros(1,5)];
    H(i+3,1:20) = [zeros(1,15) u(.25*(i+3)) w(.25*(i+3)) q(.25*(i+3)) th(.25*(i+3)) eta(.2
5*(i+3))];
end

% Make Z a column vector.

Z = Z';
cond(H);

```

```
% Calculate the parameter estimates and form the A and B matrices.
```

```
thetak = inv(H'*H)*H'*Z;
```

```
f = inv(H'*H); % Used later in standard error estimates.  
size(f)
```

```
% Calculate the standard error in the parameter estimates.
```

```
xu = thetak(1);  
xw = thetak(2);  
xq = thetak(3);  
xth = thetak(4);  
xeta = thetak(5);  
zu = thetak(6);  
zw = thetak(7);  
zq = thetak(8);  
zth = thetak(9);  
zeta = thetak(10);  
mu = thetak(11);  
mw = thetak(12);  
mq = thetak(13);  
mth = thetak(14);  
meta = thetak(15);
```

```
A = [xu xw xq xth; zu zw zq zth; mu mw mq mth; thetak(16) thetak(17) thetak(18) thetak(19)];  
B = [xeta; zeta; meta; thetak(20)];  
C = eye(4);  
D = zeros(4,1);
```

```
% Estimate the standard error for each derivative.
```

```

[y,x] = lsim(A,B,C,D,eta,t);
figure(1),plot(t,u,t,y(:,1),'--g')
xlabel('seconds'),ylabel('m/s'),title('Measured(-) and Predicted(--g) u')

figure(2),plot(t,w,t,y(:,2),'--g')
xlabel('seconds'),ylabel('m/s'),title('Measured(-) and Predicted(--g) w')

figure(3),plot(t,q,t,y(:,3),'--g')
xlabel('seconds'),ylabel('rad/s'),title('Measured(-) and Predicted(--g) q')

figure(4),plot(t,th,t,y(:,4),'--g')
xlabel('seconds'),ylabel('rad'),title('Measured(-) and Predicted(--g) theta')

sqe_u = [y(:,1) - u].*[y(:,1) - u]; % square of errors for u.
ssqe_u = sum(sqe_u); % sum of square of errors for u.
sigma_u = sqrt((ssqe_u)/(N-5)); % standard deviation of u equation.

sqe_w = [y(:,2) - w].*[y(:,2) - w]; % square of errors for w.
ssqe_w = sum(sqe_w); % sum of square of errors for w.
sigma_w = sqrt((ssqe_w)/(N-5)); % standard deviation of w equation.

sqe_q = [y(:,3) - q].*[y(:,3) - q]; % square of errors for q.
ssqe_q = sum(sqe_q); % sum of square of errors for q.
sigma_q = sqrt((ssqe_q)/(N-5)); % standard deviation of q equation.

sqe_th = [y(:,4) - th].*[y(:,4) - th]; % square of errors for th.
ssqe_th = sum(sqe_th); % sum of square of errors for th.
sigma_th = sqrt((ssqe_th)/(N-5)); % standard deviation of th equation.

for j = 1:20

```

```

        g(j,j) = sqrt(f(j,j));
    end

    g = diag(g)

    seu = sigma_u*g(1:5);
    sew = sigma_w*g(1:5);
    seq = sigma_q*g(1:5);
    seth = sigma_th*g(1:5);

    % Put error estimates in the same form as the A, B matrices.

    seA = [seu(1) seu(2) seu(3) seu(4); sew(1) sew(2) sew(3) sew(4); seq(1) seq(2) seq(3) seq(4);
    seth(1) seth(2) seth(3) seth(4)];
    seB = [seu(5); sew(5); seq(5); seth(5)];

    if nargout==6
        Ao = A;
        Bo = B;
        Co = C;
        Do = D;
        seAo = seA;
        seBo = seB;
    end
end

```


Appendix C

MATLAB Diary Examples


```

Commands to get started: intro, demo, help
Commands for more information: help, whatsnew, info, subscribe

» help diary

DIARY Save the text of a MATLAB session.
DIARY file_name causes a copy of all subsequent terminal input
and most of the resulting output to be written on the named
file. DIARY OFF suspends it. DIARY ON turns it back on.
DIARY, by itself, toggles the diary state.

» diary c:\work\sysid\dmvc
» % Load file th111334, saved in MATLAB format within POSTFLT.
» load c:\work\j_dat\1999\parid\th111334
»
» % Plot the elevator response to determine the time at which the pilot
» % makes the input.
» figure(1),plot(Time,Elevator)
»
» % Looks like approximately 10 seconds, which at a sample interval
» % of .02 seconds corresponds to 10/0.02 = 500 samples. Use this
» % information as input to "pro_lof". Like the standard functions, I've
» % incorporated some help information, eg:
» help pro_lof

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
pro_lof : Short period and phugoid response, processes data from POSTFLT.
          Removes time delay, converts to m/s, rad, rad/s.
          Removes steady offsets.
          filters and differentiates signals.

INPUTS : Time, EAS, AlphaTru, IRSptcRt, IRSPitch, Elevator, Sample No.

OUTPUTS: t, udot, wdot, qdot, thdot, u, w, q, theta, eta.

SYNTAX : [t,udot,wdot,qdot,thdot,u,w,q,theta,eta] =
          pro_lof(Time,EAS,AlphaTru,IRSptcRt,IRSPitch,Elevator,SamNu)

AUTHOR : G.J. Mullen for the NFLC, Cranfield.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

» % The processed variables are:
» [t,udot,wdot,qdot,thdot,u,w,q,theta,eta] = pro_lof(Time,EAS,AlphaTru,IRSptcRt,IRSPitch,Elevator,500);
»
» % Use "ttest" to generate a frequency response.
» help ttest

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
ttest : Estimates a frequency response using the discrete %
        Fourier transform of output/input data. %

```

```

Assumes all input data has been processed by
"pro_sp1" or "pro_lof".

INPUTS : Input (u), output (y), time (t),
sample interval (Ts, seconds)
highest desired frequency (fmax, rad/s).

OUTPUTS : Magnitude (dB), phase (deg), frequency (rad/s).

SYNTAX : [mag,phase,freq] = tfest(u,y,t,Ts,fmax);

AUTHOR : G.J. Mullen for the NFLC, Cranfield.

COMMENTS : See Ch.6 of "Understanding Signals & Systems"
by Golten for background theory.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
» % Say a maximum frequency of 10 rad/s is required. As the sample interval
» % is 0.02:
» [mag,phase,freq] = tfest(eta,q,0.02,10);
??? Input argument fmax is undefined.

Error in ==> c:\work\conmat\mfile\tfest.m
On line 48 ==> k_max = fmax*(N)/(2*pi);

» [mag,phase,freq] = tfest(eta,q,t,0.02,10);
»
» % In the above example, "tfest" takes quite a while to crunch the numbers.
» % Try the same thing using "tfest1", which is much quicker, but gives
» % identical results.
» help tfest1

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
tfest1 : Estimates a frequency response using the fast
Fourier transform of output/input data.
Assumes all input data has been processed by
"pro_sp1" or "pro_lof".

INPUTS : Input (u), output (y), time (t).

OUTPUTS : Magnitude (dB), phase (deg), frequency (rad/s).

SYNTAX : [mag,phase,freq] = tfest1(u,y,t);

AUTHOR : G.J. Mullen for the NFLC, Cranfield.

COMMENT: Uses the signal processing toolbox function "fft"
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
» [mag1,phase1,freq1] = tfest1(eta,q,t);

```

```

» clear
» % For an example of identifying parametric models, consider the short period
» % excitation th105910.
» load c:\work\j_dat\1999\parid\th105910
»
» % By examining the Elevator response as before, it looks like the pilot
» % makes an input at sample number 210. Use this as input to "pro_sp1".
» help pro_sp1

#####
pro_sp1 :Short period response, processes data from POSTFLR.
Removes time delays, converts to m/s, rad, rad/s.
Removes steady offsets.
filters and differentiates signals.
#####
INPUTS : Time, AlphaTru, IRSptcRt, Elevator, EAS, Sample No.
OUTPUTS: wdot, qdot, w, q, eta.
SYNTAX : [t, wdot, qdot, w, q, eta] =
          pro_sp1(Time, AlphaTru, IRSptcRt, Elevator, EAS, SamNu)
AUTHOR : G.J. Mullen for the NFLC, Cranfield.
#####
» [t, wdot, qdot, w, q, eta] = pro_sp1(Time, AlphaTru, IRSptcRt, Elevator, EAS, 210);
»
» % Estimate the A, B matrices using "ls_sp".
» help ls_sp

#####
ls_sp : Estimates a reduced order state space
       short period model of the form
       | wdot | = | zw | | w | + | zeta | eta
       | qdot | = | mw | | mq | | q | | meta |
       using a simple least squares algorithm.
       Assumes all data processed by "pro_sp1".
#####
INPUTS : Heave velocity and rate of change (w, wdot),
         pitch rate and rate of change (q, qdot),
         elevator (eta) and time (t).
OUTPUTS : Stability and control derivatives and
         error estimates.
SYNTAX : [A, B, C, D, seA, seB] = ls_sp(wdot, qdot, w, q, eta, t)

```

```

AUTHOR :   G.J. Mullen for the NFLC, Cranfield
%
%
COMMENTS:  For background theory see J.M. Mendel,
%
%          "Discrete Techniques of Parameter Estimation",
%          Marcel Dekker, Inc 1973.
%
%          %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
» [A,B,C,D,seA,seB] = ls_sp(wdot,qdot,w,q,eta,t)
Eigenvalue      Damping      Freq. (rad/sec)
-1.1973 + 1.0907i  0.7393      1.6196
-1.1973 - 1.0907i  0.7393      1.6196

```

A =

```

-0.8423    53.2018
-0.0247   -1.5523

```

B =

```

20.1538
-5.7813

```

C =

```

1    0
0    1

```

D =

```

0
0

```

seA =

```

0.0051    0.1831
0.0002    0.0071

```

seB =

```

0.3034
0.0118

```

```

» % The elevator to pitch rate transfer function can be extracted
» % from the state space form.

```

```

» [nqeta1,dqeta1] = ss2tf(A,B,C(2,:),D(2,:),1)
nqeta1 =

```

```

0 -5.7813 -5.3682

```

```

dqeta1 =

```

```

1.0000 2.3947 2.6232

```

```

» % The system identification toolbox function "oe" can be used to
» % estimate discrete-time transfer functions. These may then be
» % converted to continuous time if desired. Using the same data file
» % as an example, the steps required are as follows:
»

```

```

» z = [q eta];
» nn = [3 2 0];
» th = oe(z,nn,100,0.01,1.6,4096,1);

```

```

Current loss: 0.00002416 Previous loss: 0.00002416

```

```

Current th prev. th gn-dir
-0.1714 -0.1704 -0.0020
0.2286 0.2264 0.0043
-0.0599 -0.0587 -0.0023
-1.9490 -1.9490 0.0000
0.9502 0.9502 0.0000

```

```

Norm of gn-vector: 0.005337

```

```

» % The input arguments 100, 0.01 etc. are (check the manual): the max. number
» % of iterations, the tolerance, a penalty on large errors, max. number of
» % elements which may be contained in any intermediate matrix calculation
» % and the sampling interval.
»

```

```

» % Set the sample interval to 0.02;
»

```

```

» th = sett(th,0.02);
»

```

```

» % View the parameter estimates.
» present(th)

```

```

This matrix was created by the command OE on 7/5 2000 at 14:51

```

```

Loss fcn: 0.00002422 Akaike's FPE: 0.00002446 Sampling interval 0.02

```

```

The polynomial coefficients and their standard deviations are

```

```

B =

```

```

-0.1714 0.2286 -0.0599
0.0135 0.0276 0.0142

```

```

F =
    1.0000   -1.9490    0.9502
         0    0.0007    0.0007

» % Note that the standard deviations are significantly smaller than
» % the corresponding parameter estimates, so it looks OK.
»
» % Convert from theta format to a discrete-time transfer function, then to
» % a continuous transfer-function.
»
» [dnum,dden] = th2tf(th);
» [qnum2,qden2] = d2cm(dnum,dden,0.02,'tustin')

qnum2 =
   -0.1180   -5.7217   -6.9471

qden2 =
    1.0000    2.5556    3.1301

» % Note the spurious second order term in the numerator which should be
» % much smaller than the others. Remove it.
» qnum2 = qnum2(2:3);
» qnum2, qden2

qnum2 =
   -5.7217   -6.9471

qden2 =
    1.0000    2.5556    3.1301

» % Which is broadly comparable with the estimate from "ls_sp".
Current loss: 1.237e-007 Previous loss: 1.237e-007
Current th prev. th gn-dir.
   -0.9746   -0.9745   -0.0001
   56.1346   56.1413   -0.0066
   -0.0262   -0.0262    0.0000
   -1.5201   -1.5200   -0.0001
   19.9925   19.9858    0.0067
   -5.8175   -5.8162   -0.0013

Norm of gn-vector: 0.009556

```

```
» % Convert this to state-space form:
```

```
» [A1,B1,C1,D1] = th2ss(model)
```

```
A1 =
```

```
-0.9746  56.1346  
-0.0262 -1.5201
```

```
B1 =
```

```
19.9925  
-5.8175
```

```
C1 =
```

```
1  0  
0  1
```

```
D1 =
```

```
0  
0
```

```
» % A1, etc. agree closely with the output of "ls_sp"
```

```
» diary off
```

```
»
```

