Cranfield UNIVERSITY

Fortran programs for aircraft parameter identification using the estimation-before-modelling technique

J.C.Hoff

COA report No.9709
July 1997

Air Vehicle Technology Group
College of Aeronautics
Cranfield University
Cranfield
Bedford MK43 0AL
England

College of Aeronautics Report No.9709
July 1997

Fortran programs for aircraft parameter identification using the estimation-before-modelling technique

J.C.Hoff

ISBN 1 871315670

£8.00

"The views expressed herein are those of the author/s alone and do not necessarily represent those of the University"

Air Vehicle Technology Group
College of Aeronautics
Cranfield University
Cranfield
Bedford MK43 0AL
England

Acknowledgement

This report describes the outcome of some research undertaken by the author whilst studying for a PhD in the College of Aeronautics in the period 1993-1996. The author is indebted to Embraer, Empresa Brasileira de Aeronautica and to the Brazilian sponsoring body CNPq for financial support.

CONTENTS Page

NOTATION

$a_{x_m}, a_{y_m}, a_{z_m}$	Measured accelerations on axes *x,y* and *z* respectively.
b_i	Bias terms of the observation model.
F(t)	Gradient of the state matrix.
g	Acceleration of gravity.
h, h_m	Height and its measured value, respectively.
$\mathbf{H}_{k/k-1}$	Gradient of the observation matrix calculated for $\hat{\mathbf{x}}_{k/k-1}$.
$\mathbf{H}_{i/i}$	Gradient of the observation matrix calculated for $\hat{\mathbf{x}}_{i/i}$
I	Unity matix
I_X, I_Y, I_Z	Moment of inertia referred to axes *x,y* and *z* respectively.
I_{XZ}, I_{YZ}	Product of inertia *xz* and *yz*, respectively.
$\mathbf{K}_k$	Matrix of Kalman filter gains.
L	Normalised roll moment
M	Normalised pitch moment.
N	Normalised yaw moment
$\mathbf{P}_{k-1/k-1}$	Filter covariance matrix at instant (sample) *k-1*.
$\mathbf{P}_{k/k-1}$	Filter covariance matrix propagated from instant *k-1* to *k*.
$\mathbf{P}_{k/k}$	Filter covariance matrix updated at instant *k*.
Q	Process noise covariance matrices.
R	Measurement noise covariance matrix.
u, v, w	Velocity components on axes *x,y* and *z*, respectively.
p, q, r	Roll, pitch and yaw body rates.
p_m, q_m, r_m	Roll, pitch and yaw measured body rates.
V, V_m	True and measured true airspeed.
$\mathbf{x}(t)$	State vector whose terms are the states of the dynamic model.
$\hat{\mathbf{x}}_{k/k}$	Estimated value of state **x** at instant (or sample) *k*.
$\hat{\mathbf{x}}_{k/k-1}$	Propagated value of state **x** from instant (sample) *k-1* to instant *k*.
$\mathbf{x}_{k-1/k}$	Estimated of state **x** from instant (or sample) *k* back to instant *k-1*.
x_1, y_1, z_1	Body axes coordinates of accelerometers package - relative to *cg*.
x_2, y_2, z_2	Body axes coordinates of pitot probe - relative to *cg*.
x_3, y_3, z_3	Body axes coordinates of incidence vane - relative to *cg*.
x_4, y_4, z_4	Body axes coordinates of sideslip vane - relative to *cg*.
α, α_m	Incidence angle and measured incidence angle, respectively.
β, β_m	Sideslip angle and measured sideslip angle, respectively.
ϕ, ϕ_m	Attitude roll angle and its measured value
θ, θ_m	Attitude pitch angle and its measured value.
Δt	Time interval between samples.

1. INTRODUCTION

This report describes five Fortran programs formulated to be used in the Estimation-Before-Modelling (E-B-M) methodology for aircraft parameter estimation.

The E-B-M technique is a two step estimation process. In the first step, as formulated in this report, the aircraft states are estimated by using Extended Kalman Filter techniques. In the second step the unknowns aerodynamic derivatives are estimated by linear regression formulated as the Stepwise Regression [1].

The five program comprise two different techniques for state estimation, two algorithms for linear regression and a fixed-lag smoother-differentiator.
The programs are identified as;

(i) IEKF.FOR.
(ii) EKFMBF.FOR
(iii) EKFDER.FOR
(iv) MSR.FOR
(v) MSRH.FOR

(i) IEKF.FOR - is a simplified Iterated Extended Kalman Filter (IEKF) for the estimation of aircraft states.

(ii) EKFMBF.FOR - is an Extended Kalman Filter (EKF) associated with the Modified Bryson-Frazier (MBF) smoother and is used for aircraft state estimation.

(iii) EKFDER.FOR - is a Fixed Lag (FL) Smoother-Differentiator formulated specifically to smooth and differentiate the output of the program IEKF.FOR.

(iv) MSR.FOR - is the Stepwise Regression program with the linear regression formulated by the normal equation solution approach.

(v) MSRH.FOR - is the Stepwise Regression program with the linear regression solution formulated by the Householder Transformation approach.

The state estimation algorithms are formulated in terms of inertial and gravitational models, therefore independent of definition of aerodynamic models. The force components X, Y, Z and the moment components L, M and N are modelled as second order Gauss-Markov.

The programs have been formulated in Microsoft Fortran 5.1 and operate in DOS.

2. METHODOLOGY

2.1 State Estimation

The programs follow the methodology described in chapter 3 of reference [2], i.e., Extended Kalman Filter (EKF) methodology. The ordinary EKF is used associated with the MBF smoother and the Iterated Extended Kalman Filter, in its simplified version (i.e., iteration in the measurement model only), is used associated with the Fixed-Lag smoother.

In formulating the EKF and IEKF algorithms the following models have been used:

(i) The dynamic model ($\dot{\mathbf{x}} = f[\mathbf{x},t]$) is composed by;

$$\begin{aligned}
\dot{u} &= rv - qw - g\sin\theta + X_1 \\
\dot{v} &= pw - ru + g\cos\theta\sin\phi + Y_1 \\
\dot{w} &= qu - pv + g\cos\theta\cos\phi + Z_1 \\
\dot{p} &= pqC_{11} + qrC_{12} + qC_{13} + L_1 + N_1C_{14} \\
\dot{q} &= prC_{21} + (r^2 - p^2)C_{22} - rC_{23} + M_1 \\
\dot{r} &= pqC_{31} + qrC_{32} + qC_{33} + L_1C_{34} + N_1 \\
\dot{\theta} &= q\cos\phi - r\sin\phi \\
\dot{\phi} &= p + q\tan\theta\sin\phi + r\tan\theta\cos\phi \\
\dot{h} &= u\sin\theta - v\cos\theta\sin\phi - w\cos\theta\cos\phi
\end{aligned}$$

with

$$\begin{aligned}
C_{11} &= [I_{XZ}(I_{ZZ} + I_{XX} - I_{YY})]/I^2 & C_{31} &= [I_{XX}(I_{XX} - I_{YY}) + I_{XZ}^2]/I^2 \\
C_{12} &= [I_{ZZ}(I_{YY} - I_{ZZ}) - I_{XZ}^2]/I^2 & C_{32} &= I_{XZ}(I_{YY} - I_{ZZ} - I_{XX})/I^2 \\
C_{13} &= I_{XZ}I_{EX}/I^2 & C_{33} &= I_{XX}I_{EX}/I^2 \\
C_{14} &= I_{XZ}/I_{XX} & C_{34} &= I_{XZ}/I_{ZZ} \\
C_{21} &= (I_{ZZ} - I_{XX})/I_{YY} & & \\
C_{22} &= I_{XZ}/I_{YY} & & \text{where} \\
C_{23} &= I_{EX}/I_y & I^2 &= (I_{XX}I_{ZZ} - I_{XZ}^2)
\end{aligned}$$

$X_1, Y_1, Z_1, \ldots, N_1$ are modelled as 2nd order Gauss-Markov, for example,

$$\begin{aligned}
\overset{\circ}{X}_1 &= X_2 + w_1 \\
\overset{\circ}{X}_2 &= X_3 + w_2\,, \qquad \text{where } w_i \text{ are noise terms.} \\
\overset{\circ}{X}_3 &= 0 + w_3
\end{aligned}$$

(ii) The measurement model ($\mathbf{z} = f_h[\mathbf{x}, t]$) is composed by;

$$
\begin{aligned}
a_{x_m} &= \frac{X}{m} + \frac{T}{m} - g\sin\theta - (r^2+q^2)x_1 + (pq-\dot{r})y_1 + (pr+\dot{q})z_1 + b_{a_x} \\
a_{y_m} &= \frac{Y}{m} + g\cos\theta\sin\phi - (p^2+r^2)y_1 + (pq+\dot{r})x_1 + (qr-\dot{p})z_1 + b_{a_y} \\
a_{z_m} &= \frac{Z}{m} + g\cos\theta\cos\phi - (p^2+q^2)z_1 + (pr-\dot{q})x_1 + (qr+\dot{p})y_1 + b_{z_m} \\
p_m &= p \\
q_m &= q + b_q \\
r_m &= r \\
V_m &= \sqrt{u_1^{\,2} + v_1^{\,2} + w_1^{\,2}} \\
\alpha_m &= \tan^{-1}\left[\frac{w}{u}\right] - \frac{x_3 . q}{V} + b_\alpha \\
\beta_m &= \sin^{-1}\left[\frac{v}{V}\right] - \frac{x_4 . r}{V} + b_\beta \\
\theta_m &= \theta + b_\theta \\
\phi_m &= \phi \\
h_m &= h
\end{aligned}
$$

where
$$
\begin{bmatrix} u_1 \\ v_1 \\ w_1 \end{bmatrix} = \begin{bmatrix} u \\ v \\ w \end{bmatrix} + \begin{bmatrix} 0 & -r & q \\ r & 0 & -p \\ -q & p & 0 \end{bmatrix} \begin{bmatrix} x_2 \\ y_2 \\ z_2 \end{bmatrix}
$$

The bias terms (b_i) are applied only to the terms most significant for the longitudinal analysis (exception of the sideslip).

(iii) The Kalman Filter Algorithms;

The Iterated Extended Kalman Filter is formulated as follows;

Given the state vector $\mathbf{x}_{k-1/k-1}$ the matrix $\mathbf{F}$ is calculated by,

$$
\mathbf{F}(t) = \left.\frac{\partial f}{\partial x}\right|_{x_{k-1/k-1}} \tag{1}
$$

$\mathbf{F}(t)$ corresponds to the derivatives of the dynamic model (f) w.r.t the states. The covariance matrix $\mathbf{P}$ is propagated from the instant t_{k-1} to t_k by;

$$
\mathbf{P}_{k/k-1} = \mathbf{\Phi}\mathbf{P}_{k-1/k-1}\mathbf{\Phi}^T + \mathbf{Q}
$$

where:

$$
\mathbf{\Phi} = \mathbf{I} + \mathbf{F}(t)\Delta t + \frac{1}{2}(\mathbf{F}(t)\Delta t)^2 \quad ,
$$

$\mathbf{Q}$ is the process noise covariance matrix and Δt is the sampling interval.

The state vector $\mathbf{x}$ is propagated from t_{k-1} to t_k by numerical integration (4th order Runge-Kutta).

$$\hat{\mathbf{x}}_{k/k-1} = \int_{t_{K-1}}^{t_K} f(\mathbf{x}(t),t)dt \qquad (2)$$

The *Kalman gain matrix* is calculated as follows:

$$\mathbf{K}_k = \mathbf{P}_{k/k-1}\mathbf{H}^T_{k/k-1}[\mathbf{H}_{k/k-1}\mathbf{P}_{k/k-1}\mathbf{H}^T_{k/k-1} + \mathbf{R}]^{-1} \quad \text{with} \quad \mathbf{H}_{k/k-1} = \left.\frac{\partial f_h}{\partial x}\right|_{\hat{\mathbf{x}}_{k/k-1}} \qquad (3)$$

$\mathbf{H}$ are the derivatives of the measurement model (f_h) w.r.t the states and $\mathbf{R}$ is the measurement noise covariance matrix.

- *State Update*;
At time t_k the updated estimate of the state vector is calculated by adding the measurement residual, appropriately weighted by the Kalman gain matrix, to the propagated state vector calculated by equation 2.

$$\hat{\mathbf{x}}^j_{i/i} = \hat{\mathbf{x}}_{i/i-1} + \mathbf{K}^j_i[z_i - f_h(\hat{\mathbf{x}}^{j-1}_{i/i}) - \mathbf{H}_{i/i}(\hat{\mathbf{x}}^{j-1}_{i/i})[\hat{\mathbf{x}}_{i/i-1} - \hat{\mathbf{x}}^{j-1}_{i/i}]] \; j=1,2 \qquad (4)$$

The matrix $\mathbf{H}$ is then recalculated for the new vector $\mathbf{x}$ and a new Kalman Gain is calculated. The vector $\mathbf{x}$ is updated again . Only two iteration are performed.

For the ordinary EKF the states are update by the following expression, replacing equation 4 above, and no iterations are performed updating the matrix $\mathbf{H}$ and the Kalman Gain;

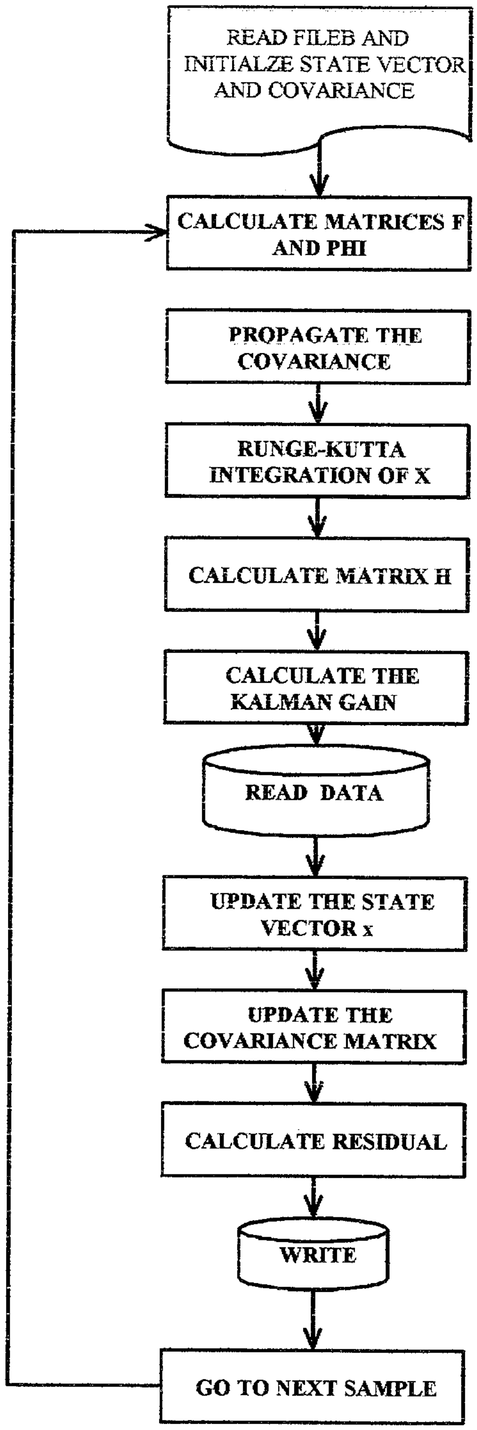
$$\mathbf{x}_{k/k} = \mathbf{x}_{k/k-1} + \mathbf{K}_k(\mathbf{z}_j - \mathbf{h}_j)$$

where z_j are the measurements and h the measurement models estimated for x_{k-1}.

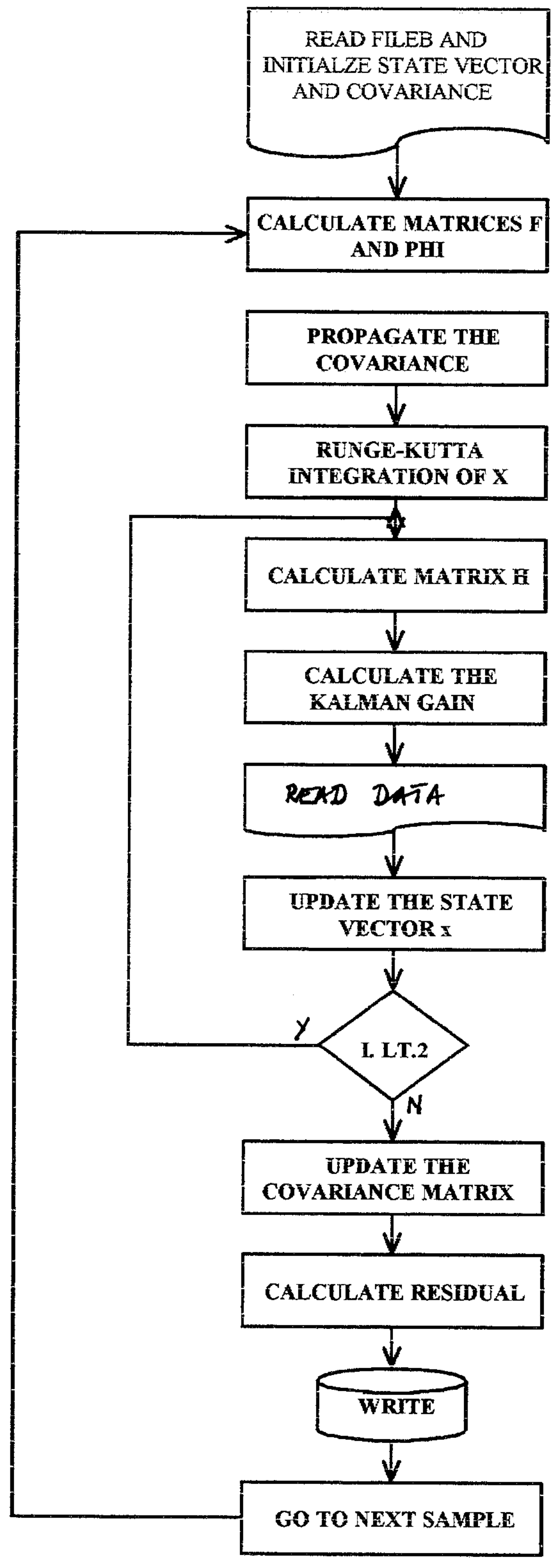
- *Covariance Matrix Update,*
The covariance matrix is updated using the latest calculated Kalman gain and $\mathbf{H}$ matrix,

$$\mathbf{P}_{k/k} = [\mathbf{I} - \mathbf{K}_k\mathbf{H}_{k/k-1}]\mathbf{P}_{k/k-1}$$

The flow diagrams below show the EKF and IEKF mechanization,



Flow diagram 1. Extended Kalman Filter mechanization



Flow diagram 2. Iterated Extended Kalman Filter mechanization

(iv) Modified Bryson-Frazier Smoother

The computational steps involved in its execution are presented below and follows Bierman [3],

- *Initialization*:

The adjoint variables, vector λ and matrix Λ, are initialized at $t = t_m$, last iteration of the EKF by,

$$\lambda_{m/m} = -\mathbf{H}^T_{m/m-1}\mathbf{D}^{-1}_{m/m-1}\mathbf{r}_m\delta_{t_m,t_0}$$
$$\Lambda_{m/m} = \mathbf{H}^T_{m/m-1}\mathbf{D}^{-1}_{m/m-1}\mathbf{H}_{m/m-1}\delta_{t_m,t_0}$$

where:

$$\mathbf{D}_{m/m-1} = [\mathbf{H}_{m/m-1}\mathbf{P}_{m/m-1}\mathbf{H}^T_{m/m-1} + \mathbf{R}]$$

$\mathbf{r}_m$ is the vector of residuals generated by the EKF, $\mathbf{H}_{m/m-1}$ is calculated by equation 3 and δ denotes the Kronecker delta function. $\delta = 0$ if t_m is not an observation time.

-*Adjoint Variables Propagation*:

The adjoint variables are evaluated at time t_k by backward integration of the following equation from time t_{k+1},

$$\begin{aligned} \dot{\lambda} &= -\mathbf{F}^T\lambda_{k+1/k+1} \\ \dot{\Lambda} &= -(\mathbf{F}^T\Lambda_{k+1/k+1}) - (\mathbf{F}^T\Lambda_{k+1/k+1})^T \end{aligned} \tag{5}$$

Here **F** is given by equation 1, however it is calculated for $\hat{x}_{k+1/k+1}$. The numerical integration of equations 5 produce $\lambda_{k/k+1}$ and $\Lambda_{k/k+1}$,

$$\lambda_{k/k+1} = \lambda_{k+1/k+1} + \int_{t_{k+1}}^{t_k} \dot{\lambda}\,dt$$

$$\Lambda_{k/k+1} = \Lambda_{k+1/k+1} + \int_{t_{k+1}}^{t_k} \dot{\Lambda}\,dt$$

- *Adjoint variables matrices update*

The matrices of adjoint variables are updated at time t_k by evaluation of the following equations,

$$\lambda_{k/k} = \lambda_{k/k+1} - \mathbf{H}^T_{k/k-1}\mathbf{D}^{-1}_{k/k-1}[\mathbf{r}_k + \mathbf{D}_{k/k-1}\mathbf{K}^T_k\lambda_{k/k+1}]$$
$$\Lambda_{k/k} = [\mathbf{I} - \mathbf{K}_k\mathbf{H}_{k/k-1}]^T\Lambda_{k/k+1}[\mathbf{I} - \mathbf{K}_k\mathbf{H}_{k/k-1}] + \mathbf{H}^T_{k/k-1}\mathbf{D}^{-1}_{k/k-1}\mathbf{H}_{k/k-1}$$

- *State vector smoothing*

The vector of smoothed state estimates is obtained by correcting the EKF filter state estimates. The vector of smoothed state variable estimates is given by,

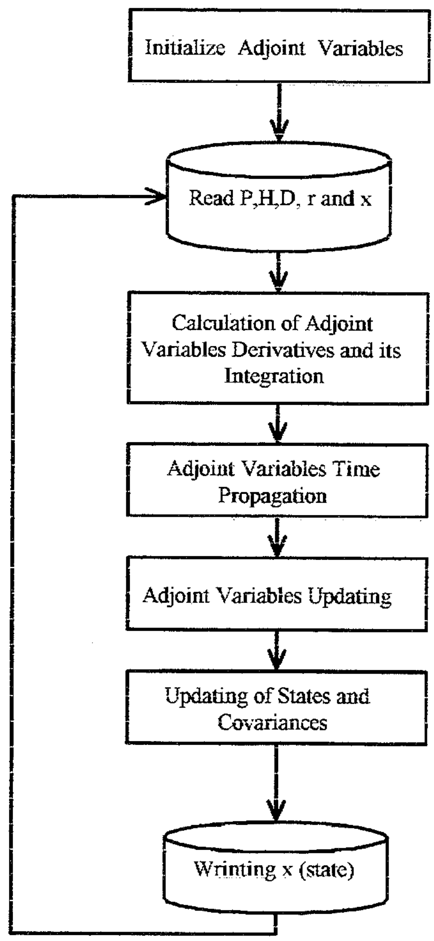
$$\hat{\mathbf{x}}_{k/k}\big|_{smoother} = \hat{\mathbf{x}}_{k/k}\big|_{EKF} - \mathbf{P}_{k/k}\big|_{EKF}\,\lambda_{k/k}$$

and the corresponding covariance matrix is given by,

$$\mathbf{P}_{k/k}\big|_{smoother} = \mathbf{P}_{k/k}\big|_{EKF} - \mathbf{P}_{k/k}\big|_{EKF}\,\Lambda_{k/k}\mathbf{P}_{k/k}\big|_{EKF}$$

In order to reduce computation time $\mathbf{P}_{k/k}, \mathbf{H}_{k/k-1}, \mathbf{D}_{k/k-1}, \mathbf{D}^{-1}_{k/k-1}, \mathbf{K}_{k}, \mathbf{r}_{k}$ and $\hat{\mathbf{x}}_{k/k}$ computed by the EKF are temporarily stored to be used by the smoother.

Below, it is presented the flow diagram of the Bryson-Frazier smoother;



Flow diagram 3. Bryson-Frazier Smoother

(v) Fixed Lag Smoother-Differentiator;

The Fixed Lag smoother-differentiator formulation follows Fioretti and Jetto [4] and reference [2].

The use of the Fixed Lag smoother requires:

- the definition of a model for the signal.
- the determination of the signal noise characteristics.
- smoothing of the modelled signal for the identified noise characteristics.

The data model is formulated as;

$$\mathbf{x}((k+1)\Delta t) = \mathbf{A}_m \mathbf{x}(k\Delta t) + \sigma_w \underline{\mathbf{w}}((k+1)\Delta t) \tag{6}$$

where

$$\mathbf{A}_m = \begin{bmatrix} 1 & \Delta t & \frac{\Delta t^2}{2!} & \cdots & \frac{\Delta t^n}{n!} \\ 0 & 1 & \Delta t & \cdots & \frac{\Delta t^{n-1}}{(n-1)!} \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{bmatrix} \tag{7}$$

with $\underline{\mathbf{w}}$ denoting a white noise sequence $\sim N(0,\underline{\mathbf{Q}})$ (Gaussian, zero mean and covariance $\underline{\mathbf{Q}}$. Δt is the sampling interval and the elements of its covariance matrix $\underline{\mathbf{Q}}$ are given by the generic expression,

$$q_{i,j} = \frac{\Delta t^{(2n+3)-(i+j)}}{(n+1-i)!(n+1-j)!(2n+3)-(i+j))}$$

The measured state component is the sampled data, thus the observation equation may be written as,

$$\mathbf{z}(k\Delta t) = \mathbf{C}\mathbf{x}(k\Delta t) + \mathbf{w}_z(k\Delta t) \tag{8}$$

Using the model formed by equations 6, 7 and 8 an ordinary Kalman Filter estimates **x** from the measurement **z** of **x** and an initial guess to the noise. The filter residuals (innovation) are calculated by:

$$r_k = z(K\Delta t) - \mathbf{C}\mathbf{A}_m \hat{\mathbf{x}}_{k-1/k-1}$$

The filter calculates and stores the residuals of the estimation given an initial value for the noise covariance. With the residuals and the **stabilized** filter covariance the theoretical and estimated autocorrelation are determined.
The theoretical autocorrelation function φ of the residuals in steady-state condition is given by;

$$\varphi(\sigma_w, \sigma_\eta, i) = [\mathbf{C}\tilde{\mathbf{P}}(\sigma_w, \sigma_\eta)\mathbf{C}^T + \sigma_\eta^2]\delta_i$$

where δ_i is the Kronecker delta function ($\delta_i = 1$ for $i = 0$, $\delta_i = 0$ for $i \neq 0$) and $\tilde{\mathbf{P}}(\sigma_w, \sigma_\eta)$ is the filter estimated covariance in steady-state condition.

The actual autocorrelation function of the innovation process is calculated by,

$$\hat{\varphi}(\sigma_w, \sigma_\eta, i) = \frac{1}{m-i}\sum_{k=1}^{m-i} r_k r_{k+i} \quad , \qquad i=0,1,\ldots,l$$

where:

- r_k $(k=1,2,\ldots,m)$ are samples of the innovation process computed with the steady Kalman filter gain.
- m is the total number of observation.
- l is the filter lag measured in iteration steps.

The determination of the noise characteristics of the signal is carried out by minimizing the error between the theoretical and the estimated autocorrelation. The cost function is given by;

$$J(\sigma_w, \sigma_\eta) = \sum_{i=0}^{l}[(\varphi(\sigma_w, \sigma_\eta, i) - \hat{\varphi}(\sigma_w, \sigma_\eta, i)]^2$$

The optimal values of the process and measurement noise σ_w and σ_η, respectively, are those that minimize the norm of J in a region R of the (σ_w, σ_η)-plane for the autocorrelation functions calculated with the steady-state gain of the Kalman filter. The cost function J is rewritten as,

$$J(\sigma_w, \sigma_\eta) = J_1(\sigma_w, \sigma_\eta) + J_2(\sigma_w, \sigma_\eta)$$

where,

$$J_1(\sigma_w, \sigma_\eta) = [\mathbf{C}\mathbf{P}(\sigma_w, \sigma_\eta)\mathbf{C}^T + \sigma_\eta^2 - \sigma_r^2]^2$$

and,

$$J_2 = \sum_{i=1}^{l}\hat{\varphi}(\sigma_w, \sigma_\eta, i)^2$$

with,

$$\sigma_r^2 = \frac{1}{m}\sum_{k=1}^{m} r_k^2$$

Fixing a value $\overline{\sigma}_\eta$ for σ_η, J_2 is minimized by varying σ_w with the help of a simple minimization algorithm. Once $\overline{\sigma}_w$ has been calculated, J_1 is minimized by calculating:

$$\sigma_w^2 = \hat{\sigma}_w^2 = \frac{\overline{\sigma}_r^2 \overline{\sigma}_w^2}{\mathbf{C}\tilde{\mathbf{P}}(\overline{\sigma}_w, \overline{\sigma}_\eta)\mathbf{C}^T + \overline{\sigma}_\eta^2}$$

The optimal pair (σ_w, σ_η) is the pair $(\hat{\sigma}_\eta, \hat{\sigma}_w)$ with $\hat{\sigma}_\eta = \frac{\overline{\sigma}_\eta}{\overline{\sigma}_w}\hat{\sigma}_w$ that minimized both $J_1(\sigma_w, \sigma_\eta)$ and $J_2(\sigma_w, \sigma_\eta)$ and consequently $J(\sigma_w, \sigma_\eta)$.

The Fixed Lag Smoother is implemented augmenting the state vector with additional states representing the values of the states in the previous instant of time. The additional states are represented by,

$$\begin{aligned} \mathbf{x}_1(i) &= \mathbf{x}(i-1) \\ \mathbf{x}_2(i) &= \mathbf{x}_1(i-1) \\ &\vdots \\ \mathbf{x}_l &= \mathbf{x}_{l-1}(i-1) \end{aligned}$$

Thus the additional vectors contains the l previous values of the state vector $\mathbf{x}(i)$ where l is the time lag in sampling intervals, resulting in the following augmented state space system:

$$\begin{bmatrix} \mathbf{x}(i) \\ \mathbf{x}_1(i) \\ \mathbf{x}_2(i) \\ \vdots \\ \mathbf{x}_l(i) \end{bmatrix} = \begin{bmatrix} \mathbf{A}_m(i/i-1) & 0 & \cdots & 0 & 0 \\ \mathbf{I} & 0 & \cdots & 0 & 0 \\ 0 & \mathbf{I} & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & \mathbf{I} & 0 \end{bmatrix} . \begin{bmatrix} \mathbf{x}(i-1) \\ \mathbf{x}_1(i-1) \\ \mathbf{x}_2(i-1) \\ \vdots \\ \mathbf{x}_l(i-1) \end{bmatrix} + \begin{bmatrix} \sigma_w^2 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \mathbf{w}_x(i-1)$$

while the corresponding output equation is,

$$\mathbf{z}(i) = \begin{bmatrix} C & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{x}(i) \\ \mathbf{x}_1(i) \\ \mathbf{x}_2(i) \\ \vdots \\ \mathbf{x}_l(i) \end{bmatrix} + \sigma_\eta(i)$$

Fixed-lag smoothing is accomplished by applying Kalman filter equations to the augmented state space model, resulting;

- covariance propagation,

$$\begin{aligned}
\mathbf{P}(i/i-1) &= \mathbf{A}\mathbf{P}(i-1/i-1)\mathbf{A}_m^T + \sigma_w^2\overline{\mathbf{Q}} \\
\mathbf{P}_1(i/i-1) &= \mathbf{P}(i-1/i-1)\mathbf{A}_m^T \\
&\vdots \\
\mathbf{P}_l(i/i-1) &= \mathbf{P}_l(i-1/i-1)\mathbf{A}_m^T
\end{aligned}$$

- Kalman gain,

$$\begin{aligned}
\mathbf{K}(i) &= \mathbf{P}(i/i-1)\mathbf{C}^T[\mathbf{C}\mathbf{P}(i/i-1)\mathbf{C}^T + \sigma_\eta^2]^{-1} \\
\mathbf{K}_1(i) &= \mathbf{P}_1(i/i-1)\mathbf{C}^T[\mathbf{C}\mathbf{P}_1(i/i-1)\mathbf{C}^T + \sigma_\eta^2]^{-1} \\
&\vdots \\
\mathbf{K}_l(i) &= \mathbf{P}_l(i/i-1)\mathbf{C}^T[\mathbf{C}\mathbf{P}_l(i/i-1)\mathbf{C}^T + \sigma_\eta^2]^{-1}
\end{aligned}$$

- state update,

$$\begin{aligned}
\hat{\mathbf{x}}(i/i) &= \hat{\mathbf{x}}(i/i-1) + \mathbf{K}_k[\mathbf{z}(i) - \mathbf{C}\hat{\mathbf{x}}(i/i-1)] \\
\hat{\mathbf{x}}(i-1/i) &= \hat{\mathbf{x}}(i-1/i-1) + \mathbf{K}_1[\mathbf{z}(i) - \mathbf{C}\hat{\mathbf{x}}(i/i-1)] \\
&\vdots \\
\hat{\mathbf{x}}(i-l/i) &= \hat{\mathbf{x}}(i-l/i-1) + \mathbf{K}_l[\mathbf{z}(i) - \mathbf{C}\hat{\mathbf{x}}(i/i-1)]
\end{aligned}$$

Here the prior estimate on the right-hand side of each equation is the previous value of the posterior estimate from the left-hand side of the equation immediately above.

- covariance update,

$$\begin{aligned}
\mathbf{P}(i/i) &= [\mathbf{I} - \mathbf{K}_k\mathbf{C}]\mathbf{P}(i/i-1) \\
\mathbf{P}_1(i/i) &= \mathbf{P}_1(i/i-1)[\mathbf{I} - \mathbf{C}^T\mathbf{K}_k^T] \\
&\vdots \\
\mathbf{P}_l(i/i) &= \mathbf{P}_l(i/i-1)[\mathbf{I} - \mathbf{C}^T\mathbf{K}_k^T]
\end{aligned}$$

The covariance matrices without subscript such as $\mathbf{P}(i/i-k)$ and $\mathbf{P}(i/i)$ and the Kalman gain matrix $\mathbf{K}_k$ are those of the standard Kalman filter algorithm applied to the original state space system while the remaining $\mathbf{K}_i$ and $\mathbf{P}_i$ are generated by the Fixed Lag smoother.

The optimal lag is defined as two or three times the dominant time constant of the Kalman filter given by;

$$l \geq \text{int}[2\,\tau_f \,/\, \Delta t]$$

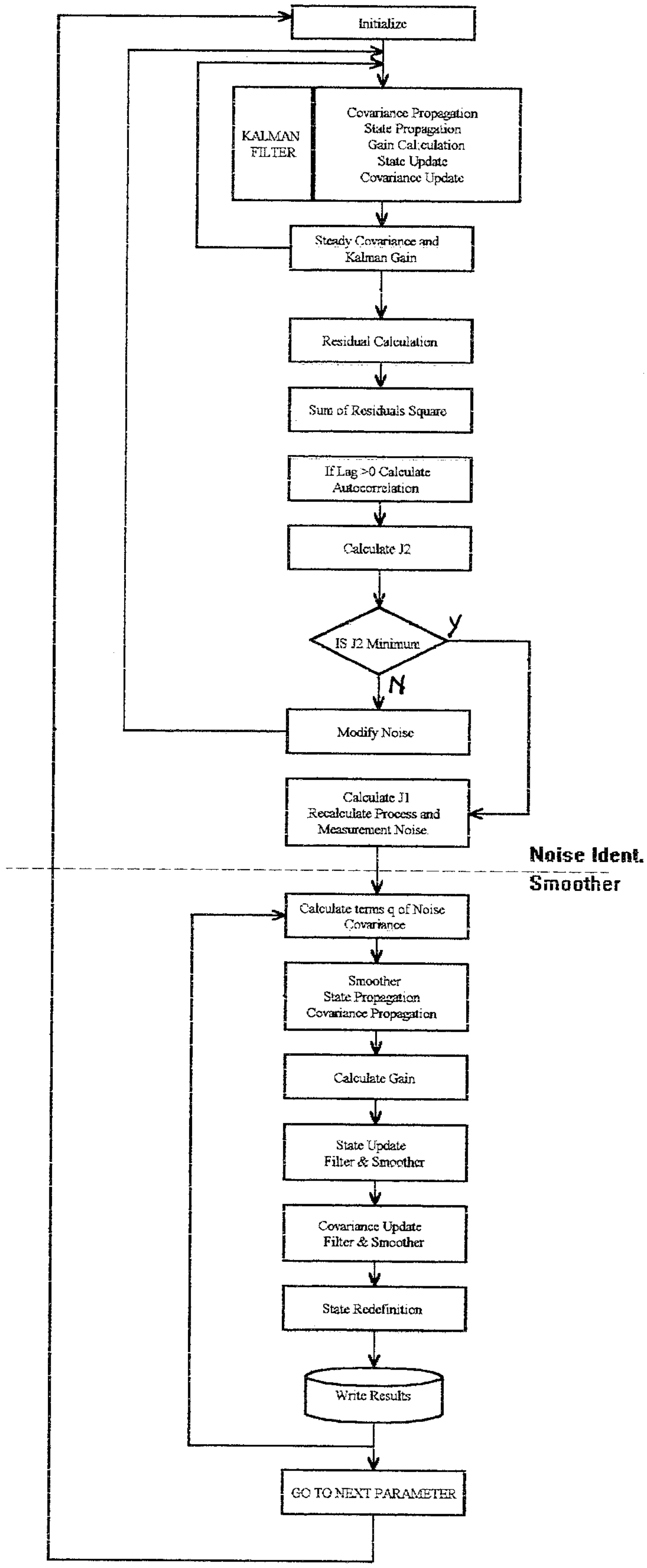
where τ_f is the dominant time constant of the filter and Δt is the sampling interval

$$\tau_f = -\Delta t \,/\, \ln \lambda_{\max}$$

$\lambda_{\max}$ is the dominant eigenvalue of the Kalman filter dynamic matrix

$$[\mathbf{A}_m - \mathbf{K}_k \mathbf{C} \mathbf{A}_m]$$

The eigenvalues are estimated outside of the program (e.g. using Matlab).
The dynamic matrix is calculated for every smoothed state and its time constant is determined in order to identify whether the assumed lag satisfy or not the ratio of 2 to 3 time constants. Next, a flow diagram of the program EKFDER.FOR is presented.



Flow Diagram 4. Program EKFDER.FOR

2.2 Linear Regression and Stepwise Regression

- Normal Equation Solution to the Linear Regression;

If **A** is the matrix of the independent variables **x** (measurements) and **y** is the vector of the dependent variable the coefficients of the expansion of **y** as a function of **x**, i.e., $y = \mathbf{A}\Theta + e$ are given by:

$$\Theta = \left[\mathbf{A}^T\mathbf{A}\right]^{-1}\mathbf{A}^T\mathbf{y} \qquad (8)$$

This is the normal equation formulation of the linear regression.

- Linear Regression Solution by Householder Transformation;

The Householder Transformation (an orthogonal transformation) is used to triangularize the matrix formed by augmenting the matrix **A** of the independent variables with the vector **y** of the dependent variable.

Let **T** be the transformation. If **A** is *n* x *m*, applying *m* transformation to the augmented matrix the resulting matrix will be triangular;

$$\mathbf{T}[\mathbf{A} \mid \mathbf{y}] = [\bar{\mathbf{A}} \mid \bar{\mathbf{y}}]$$

In this case the solution of $\mathbf{y} = \mathbf{A}\Theta + e$ may be formulated as;

$$\Theta = \bar{\mathbf{A}}^{-1}\bar{\mathbf{y}}$$

This is easily obtained because $\bar{\mathbf{A}}$ is triangular and the inversion may be obtained by back substitution.

Matrix Triangularization by Householder transformation, from Ref. [2] : The basic operation of the triangularization process using the *Householder Transformation* is the construction of the scalar s and the matrix $\tilde{\mathbf{A}}(m,n-1)$ so that,

$$\mathbf{T}_u\mathbf{A} = \begin{bmatrix} s & \\ & \tilde{\mathbf{A}} \\ 0 & \end{bmatrix}$$

where s and $\tilde{\mathbf{A}}$ are computed by;

$$s = -sgn(A(1,1))\left[\sum_{i=1}^{n}[A(i,1)]^2\right]^{1/2}$$

with

$$u_{(1)} = A(1,1) - s \quad \text{and} \quad u_{(i)} = A(i,1) \quad i = 2,3,\ldots,m$$

for the application of one elementary transformation. Generalizing for m applications, results;

$$s_j = -sgn(A(j,j))\left[\sum_{i=j}^{m}[A(i,j)]^2\right]^{1/2} \qquad \text{with}$$

$$\tilde{A}(i,j-1) = A(i,j) + \gamma u(i) \qquad u_j(i) = 0 \qquad i < j$$

$$\gamma = \beta\sum_{i=1}^{n} u_{(i)}A(i,j) \qquad u_j(i) = A(i,j) \qquad i > j$$

- Stepwise Regression Procedure

The stepwise regression procedure is a process that includes or excludes independent variables in a regression model by analysing how the variables contribute to the overall fit of the regression model. The exclusion process is carried out by the analysis of the partial 'F' test performed for each variable included in the model while the inclusion process is carried out by residual analysis, as presented in chapter 4 of Reference [2].

Once the regression coefficients were calculated by any one of the regression process above described, the following terms are calculated;

$Dy = y - \hat{y}$ — Regression residual, where $\hat{y}$ is the dependent variable values calculated by the model.

$RESS = \sum Dy * Dy$ — Residual sum of squares.

$VAR = RESS / (NAT - IVV)$ — Residual Variance, where NAT is the number of samples and IVV is the number of independent variables in the regression model.

$SB(I) = \sqrt{VAR * [\mathbf{A}^T(I,I)\mathbf{A}(I,I)]^{-1}}$ — Standard error of the regression coefficients.

$F = \dfrac{\Theta\mathbf{A}^T y - NATy_{AVG}^2}{VAR\,(IVV - 1)}$ — Regression 'F' value.

y_{AVG} is the average value of the dependent variable.

$$R^2 = \frac{F}{\frac{(NAT - IVV)}{(IVV - 1)} + F}$$ Regression correlation coefficient

The partial correlation coefficient is calculated by;

$F_p(i) = \frac{\Theta^2(i)}{SB^2(i)}$, That is, one coefficient F_p for every variable in the regression.

If the calculated value of F_p is small than a specified value (e.g. 7) then the variable is rejected from the model (only one variable is rejected, i.e., the one with smallest F_p value).

To analyse which variable, between the ones not yet included in the model, is the best candidate to be included in the next interaction, the following steps are performed;

- Calculate the regression coefficients of the actual model taken as dependent variables the independent variables not yet included in the model;

$$\Theta_{iN} = [\mathbf{A}^T\mathbf{A}]^{-1}\mathbf{A}^T\mathbf{x}_i \qquad (9)$$

- If $\hat{\mathbf{y}}_{iN}$ is the estimated value of $\mathbf{x}_i$, then the residual of the regression is given by;

$$\mathbf{z}_i = \hat{\mathbf{y}}_{iN} - \mathbf{x}_i \quad \text{Residual}$$

$$z_{AVG} = \sum z_i / NAT \quad \text{Residual average}$$

- Calculate;

$$SJJ = \sum z_i z_i \qquad \text{The residual sum of squares,}$$

$$SJY = \sum (z_i - z_{AVG})(y - \hat{y} - Dy_{AVG}) \quad \text{and} \quad RJY = \frac{SYY}{\sqrt{SYY * SJJ}}$$

The process is repeated for all variables not yet included in the model. The next variable to be included in the model will be the one presenting biggest RJY value.

3. INPUT DATA FILES.

3.1 Files for the EKFMBF.FOR and IEKF.FOR programs.

The programs use two files, FileA and FileM.

FileA contains the flight data consisting in lines of measured data containing the following parameters in free format:

a_x, - longitudinal acceleration (m/s)
a_y - lateral acceleration (m/s)
a_z - normal acceleration (m/s)
p - roll rate (rad)
q - pitch rate (rad)
r - roll rate (rad)
V - true airspeed (m/s)
α - incidence angle (rad)
β - sideslip angle (rad)
h - altitude (m)
θ - pitch attitude
ϕ - roll attitude.
Thrust (2 propellers)
Propeller Normal Force (2 propellers).

Table 3.1 below represents one data sample of FILEA.DAT

9.979670E-01	-2.394202E-03	-5.746084E-02	-6.204185E-03
-8.863121E-04	-8.181342E-04	68.656340	6.889767E-02
1.218210E-02	2085.999000	1.046959E-01	-9.683409E-03
6600.000000	400.0000000		

Table 3.1 Example of FILEA.DAT

FileM contains the initial value of the states, covariances, aircraft mass, inertias, etc.

FileM starts with the states initial values (7 values in every line);
By order the states are:

1. u - longitudinal component of the velocity
2. v - lateral component of the velocity.

3. w - normal component of the velocity.
4. p - roll rate
5. q - pitch rate
6. r - roll rate
7. θ - pitch attitude
8. ϕ - roll attitude.
9. h - altitude
10. X1 - corresponds to (T+X)/mass
11. X2 - internal variable (Gauss Markov model) that may be set equal to zero.
12. X3 - internal variable (Gauss-Markov model) that may be set equal to zero.
13. Y1 - is the normalised Lateral Force.
14. Y2 - internal variable (Gauss-Markov model) that may be set equal to zero.
15. Y3 - internal variable (Gauss-Markov model) that may be set equal to zero.
16. Z1 - is the normalised Normal Force.
17. Z2 - internal variable (Gauss-Markov model) that may be set equal to zero.
18. Z3 - internal variable (Gauss-Markov model) that may be set equal to zero.
19. L1 - is the normalised Roll Moment.
20. L2 - internal variable (Gauss-Markov model) that may be set equal to zero.
21. L3 - internal variable (Gauss-Markov model) that may be set equal to zero.
22. M1- is the normalised Pitch Moment.
23. M2 - internal variable (Gauss-Markov model) that may be set equal to zero.
24 M3 - internal variable (Gauss-Markov model) that may be set equal to zero.
25 Z1 - is the normalised Normal Force.
26 Z2 - internal variable (Gauss-Markov model) that may be set equal to zero.
27 Z3 - internal variable (Gauss-Markov model) that may be set equal to zero.
28. Normal acceleration bias
29. Incidence bias.
30. Sideslip bias
31. Pitch rate bias
32. Pitch Attitude bias
33. Longitudinal acceleration bias.

Note that the numbers above represent also the order of the states in the state vector, as used in the programs.

After the state initial values the file contains the covariance of the measurements in the following order (6 values every line);

1. a_x, 2. a_y, 3. a_z, 4. p, 5. q, 6. r
7. V, 8. Incidence, 9. Sideslip, 10. Altitude, 11. Pitch attitude, 12. Attitude Roll.

After that, the file contains the Process Noise of the state variables in the same order as for the initial values of the states and in the same format. After that the file contains the covariance of the initial values of the state variables (same order and format).
The last line of the file contains:

Number of samples, sample interval, CG%, Ix, Iy, Iz, Ixz, Engine z-arm to CG.

Table 3.2 shows an example of FILEB.DAT

```
68.54  0  4.64  -6.20E-03  -8.86E-04  -8.18E-04  1.04E-01
-9.68E-03  2085  .94  0  0  0  0
0    -9.999   0  0  0  0  0
0    0     0  0  0  0   0
0    0     0  0.34  0.22
2.1E-04  2.9E-04  1.5E-04  1.0E-08  7.0E-08  2.0E-07
1.3E-03  1.7E-06  8.0E-07   0.67  8.0E-07  1.0E-08
0.075  0.05  0.075  0.00001  0.00001  0.00001  0.000001
0.00001  0.75  0.025  0.025  0.01  0.025   0.025
0.001  0.025  0.025  0.01  0.02  0.01   0.001
0.02   0.01  0.001  0.02  0.01  0.001   0.0000000000001
.00000000000001 .0000000000001 .0000000000001 .0000000000001 .00000000000001
0.1  0.1  0.1  0.1  0.1  0.1  0.1
0.1  0.1  0.1  0.1  0.1  0.1  0.1
0.1  0.1  0.1  0.1  0.1  0.1  0.1
0.1  0.1  0.1  0.1  0.1  0.1  0.5
0.25  0.25  0.25  0.25  5.0
1965  0.04  25.59  26682.  42657.  54217.  3304.  0.0633
```

Table 3.2 Example of FILEB.DAT

This file in fact may have a non fixed format because it depends of the number of states in the program, mainly the bias terms effectively included in the measurement model.

3.2 Files for the EKFDER.FOR program.

The program EKFDER.FOR needs two data files, FILEC and FILED.

FILEC.DAT:

FILEC.DAT: contains the information relative to the parameters that are going to be smoothed-differentiated.

FILEC first line contains the initial values of the measurement and process noise, the last in general formulated as a large number (e.g. 5000 is a good number).
The next four lines contain a 4x4 initial covariance matrix, reflecting and initial guess to the covariances.
It is followed by a line containing the total number of samples in the file, the number of parameter in each sample, the sampling interval, the lag (in sampling intervals) to be used in the smoother, the aircraft MASS and the aircraft inertia Iy.
Next lines contain the parameters to be smoothed-differentiated. Table 3.3 below presents a typical format of FILEC.DAT

```
 0.05   5000.
 0.01  0.1  0.1  0.1    !
 0.1   0.1  0.1  0.1    !  May be used
 0.1   0.1  0.1  0.5    !  for any data.
 0.1   0.1  0.5  1.0    !
 2015  9   0.04  15  5334.  52600.
 69.00         4.551100     -4.6222E-03    7.742999E-02   !
 -6.011E-03   -3.834E-01    2.2265E-03     5.555777E-03   !
 -1.666E-01                                               !
 69.10         4.581100     -4.6322E-03    7.752909E-02   !
 -6.111E-03   -3.854E-01    2.2765E-03     5.565787E-03   !
 -1.766E-01                                               !  A total of 2015
     .                                                    !  samples.
     .                                                    !
     .                                                    !
 69.10         4.581100     -4.6322E-03    7.752909E-02   !
 -6.111E-03   -3.854E-01    2.2765E-03     5.565787E-03   !
 -1.766E-01                                               !
```

Table 3.3 FILEC.DAT

FILED.DAT

Contains the elevator angle in *rd* (or any other control input). It is not used in the program for calculation purposes. It is used only to inlude the input in the regression file to be used in the linear regression. Note that if more inputs are added, the program EKFDER has to be revised on order to reflect the additional parameters.

File example:

```
0.020
0.021
0.021
  .
  .
  .
0.001
```

3.3 Data File of the Regression Programs MSR.FOR and MSRH.FOR.

One common file is required by the regression programes MSR.FOR and MSRH.FOR.
The file contains lines of samples taken in the time interval. Each line contains the independent variables plus the dependent variables. The file first line contains the total number of sample, the number of independent variables and the umber of dependent variables. The maximum number of independent variables is 11 and the maximum number of dependent variables is 3. The maximum number of samples is 2000.

Table 3.4 below presents a typical format of the data file;

```
2000   11   3
    68.562580       4.853940    4.829005E-03   7.069317E-02      ! 11 independent
  -1.054134E-01   4.619305E-03    4700.827000      23.560730     ! variables plus 3
  -8.899439E-02  -2.015400E-02    4155.000000    -535.373500     ! dependent
  -52571.430000    -328.451300                                   ! variables.
    68.562680       4.847060    3.998138E-03   7.072155E-02
  -1.642646E-01   3.966332E-03    4700.841000      23.493990
  -8.909377E-02  -2.142626E-02    4156.000000    -517.561400
  -52637.130000    -253.778700
    68.562870       4.839545    3.109287E-03   7.072821E-02
  -2.095646E-01   3.879622E-03    4700.867000      23.421190
  -8.929249E-02  -2.291300E-02    4157.000000    -483.274600
  -52673.050000    -221.317600
    68.563100       4.830721    2.182843E-03   7.071401E-02
  -2.468214E-01   3.965823E-03    4700.899000      23.335870
  -8.919313E-02  -2.311341E-02    4158.000000    -445.931600
  -52681.290000    -224.798200
```

```
 68.563120       4.818213   1.296546E-03   7.067891E-02
-2.895375E-01   2.652711E-03   4700.901000      23.215180
-8.859694E-02  -2.075625E-02   4159.000000   -437.393500
-52668.930000    -241.696900
 .
 .
 .
 68.521100       4.810721   2.082843E-03   7.061401E-02
-2.468214E-01   3.965823E-03   4700.899000      23.345870
-8.919313E-02  -2.311341E-02   4158.000000   -445.931600
-52691.390000    -225.698200

                         Total of 2000 sample
```

Table 3.4 Regression data file.

2000 x 12 represents the maximum matrix size (64 K) that can be handled by a PC using DOS.

4. PROGRAM LISTINGS.

The program listings contain notation of the main variables used in the programs.

4.1 Program EKFMBF.FOR

```
C    ********** P R O G R A M   E K F M B F . F O R *********
C                              Hoff Aug/94
C                         REV MAR/95, REV APR/P5
C
C    Estimation of aircraft states u,v,w,p,q,r,X,Y,Z,L,M,N
C    alpha,beta using an Extended Kalman Filter and a Modified
C    Bryson-Frazier Smoother applied to an inertial and
C    gravitational model.
C
C    Bias terms to Airspeed, Incidence, Sideslip and Pitch Rate.
C
C    Needs two data files
c   (i) File with initial conditions, process and measurement
C       noise covariances, CG position, aircraft Inertias,
C       Mass and engine z arm to CG.
C    (ii) File with flight data: Ax, Ay, Az, p, q, r, True airpeed,
C        Alpha, Beta, Altitude, theta, Phi, Thrust, Propeller
C        Normal Force (metric units, angles in rad.).
C
C    NAT    Number of samples in the data file
C    CG     Aircraft CG %
C    RR     Covariance matrix of the measurements
C    QQ     Process noise covariance
C    PU     Initial covariance of the estimates
C    DT     Sampling interval
C    Ix, Iy,... Moment of Inertia
C    MASS   AIRCRAFT MASS
C    X(I)   MATRIX OF UPDATED STATES
C    XM(I)  MATRIX OF TIME PROPAGATED STATES
C    PM(I,J) TIME PROPAGATED COVARIANCE MATRIX
C    PU(I,J) UPDATED COVARIANCE MATRIX
C    KK(I,J) KALMAN GAIN MATRIX
C    hh(I)  VECTOR OF MEASUREMT MODELS
C    ZZ(I)  VECTOR OF MEASUREMENTS
C    DH(I)  RESIDUAL
C    F(I,J) GRADIENT OF THE DYNAMIC MODEL
C    H(I,J) GRADIENT OF THE MEASUREMENT MODEL
C    LAM,LAMBM,LAND SMOOTHER ADJOINT VARIABLE IN ITS DIF. FORMATS
C    AA,AAM  SMOOTHER ADJOINT VARIABLE IN ITS DIF. FORMATS
C    PD,RK,KH AUXILIARY MATRICES
C    WORK,IDENT AUXILIARY MATRICES OF INVERSION ROUTINE
C    Lze    ENGINE z ARM RELATIVE TO THE CG.

      REAL*8 H(12,31),PM(31,31),QQ(31),hh(12),DH(12),
     +     LAM(31),LAMBM(31),LAMD(31)
      REAL*4 X(31),XM(31),ZZ(12),RR(12),Ix,Iy,Iz,Ixz,I2,Lze,MASS
      REAL*8 WORK[ALLOCATABLE] (:,:),   RK[ALLOCATABLE] (:,:),
     +       PD[ALLOCATABLE] (:,:),   F[ALLOCATABLE] (:,:),
     +       KK[ALLOCATABLE] (:,:), HPHT1[ALLOCATABLE] (:,:),
```

```
     +     HPHT[ALLOCATABLE] (:,:),    KH[ALLOCATABLE] (:,:),
     +       PU[ALLOCATABLE] (:,:), IDENT[ALLOCATABLE] (:,:),
     +       AA[ALLOCATABLE] (:,:),   AAM[ALLOCATABLE] (:,:)
      INTEGER*2 IN,K1,K4
      INTEGER*4 MM,K2,K3
      OPEN(UNIT=2,FILE='phu809A.DAT',STATUS='OLD')
      OPEN(UNIT=6,FILE='FILEC.DAT',STATUS='OLD')
      OPEN(UNIT=3,FILE='PMBFD1.DAT')
      OPEN(UNIT=4,FILE='PMBFD2.DAT')
c      OPEN(UNIT=5,FILE='PMBFD3.DAT')
      OPEN(UNIT=7,FILE='PMBFE3.DAT')
      OPEN(UNIT=9,FILE='PMBFE1.DAT')
      OPEN(UNIT=10,FILE='PMBFE2.DAT')
      OPEN(UNIT=8,ACCESS='DIRECT',FILE='DDAT8.DAT',
     +    FORM='UNFORMATTED',RECL=248,STATUS='NEW')
      OPEN(UNIT=12,ACCESS='DIRECT',FILE='DDAT1.DAT',
     +    FORM='UNFORMATTED',RECL=224,STATUS='NEW')
      OPEN(UNIT=13,ACCESS='DIRECT',FILE='DDAT2.DAT',
     +    FORM='UNFORMATTED',RECL=96,STATUS='NEW')
      OPEN(UNIT=14,ACCESS='DIRECT',FILE='DDAT3.DAT',
     +    FORM='UNFORMATTED',RECL=96,STATUS='NEW')
      OPEN(UNIT=15,ACCESS='DIRECT',FILE='DDAT4.DAT',
     +    FORM='UNFORMATTED',RECL=96,STATUS='NEW')
      OPEN(UNIT=16,ACCESS='DIRECT',FILE='DDAT5.DAT',
     +    FORM='UNFORMATTED',RECL=96,STATUS='NEW')
      OPEN(UNIT=17,ACCESS='DIRECT',FILE='DDAT6.DAT',
     +    FORM='UNFORMATTED',RECL=248,STATUS='NEW')
      OPEN(UNIT=18,ACCESS='DIRECT',FILE='DDAT7.DAT',
     +    FORM='UNFORMATTED',RECL=248,STATUS='NEW')
      He=0.0
      G=9.80665
      MM=0
      K1=0
      K2=0
      K3=0
      K4=0

C... INITIALIZATION - Matrix in a data file with all initial values
C                obtained averaging the stabilization
      READ(6,*) (X(K),K=1,7)       !
      READ(6,*) (X(K),K=8,14)      !
      READ(6,*) (X(K),K=15,21)     ! STATES INITIAL VALUES
      READ(6,*) (X(K),K=22,28)     !
      READ(6,*) (X(K),K=29,31)     !
C
      READ(6,*) (RR(I),I=1,6)      ! MEASUR. NOISE COVAR. MATRIX
      READ(6,*) (RR(I),I=7,12)     ! ASSUMED DIAGONAL
C
      READ(6,*) (QQ(J),J=1,7)      !
      READ(6,*) (QQ(J),J=8,14)     ! PROCESS NOISE COVAR. MATRIX
      READ(6,*) (QQ(J),J=15,21)    ! ASSUMED DIAGONAL
```

```
      READ(6,*) (QQ(J),J=22,28)    !
      READ(6,*) (QQ(J),J=29,31)    !
C
      ALLOCATE (PU(31,31))

C     INITIAL COVAR. MATRIX
      DO J=1,31
       DO I=1,31
        PU(I,J)=0.D0              ! ZEROING THE COVARIANCE MATRIX
       ENDDO
      ENDDO
      jk=0
      DO I=1,4
       READ(6,*) (PU(J,J),J=1+jk,7+jk)
       jk=jk+7
      ENDDO
       READ(6,*) (PU(J,J),J=29,31)

      READ(6,*) NAT,DT,CG,Ix,Iy,Iz,Ixz,MASS,Lze
C
      Xv = 7.1325 + (CG-30)*1.717/100. ! POSITION INCID./SIDES. VANE
      Zv = 0.57                    !  "         "
      D = SQRT(Xv**2 + Zv**2)        ! Vane approx. dist to cg.
      X1 = Xv-0.35                   ! PITOT POSIT. X  m
      Y1 = 0.                      ! "      "  Y  m
      Z1 = 0.57                    ! "      "  Z  m (APPROX.)
      X0 = 1.717*(CG-85.3)/100.       ! IRS position x (Average)
      Y0 = -.584                   !      "   y
      Z0 = 0.236                   !      "   z
c           Valid for the Jetstream G-AXUI
c
      I2=Ix*Iz-Ixz*Ixz
      A11=Ixz*(Iz+Ix-Iy)/I2
      A12=(Iz*(Iy-Iz)-Ixz*Ixz)/I2
      A13=Ixz*He/I2
      A14=Ixz/Ix                   !
      A21=(Iz-Ix)/Iy               ! DYNAMIC MODEL CONSTANTS.
      A22=Ixz/Iy                   !
      A23=He/Iy
      A31=(Ix*(Ix-Iy)+Ixz*Ixz)/I2
      A32=(Ixz*(Iy-Iz-Ix))/I2
      A33=Ix*He/I2
      A34=Ixz/Iz

      DO ISAMPLE=1,NAT    !<<<<<<< MAIN LOOP >>>>>>>>>>>>>>>>>>>>>>>
      write(*,*) 'SAMPLE No.',ISAMPLE

      ALLOCATE (F(31,31))
C
C***  F MATRIX   F=df/dx  **********************************
C
```

```
      DO I=1,31
       DO J=1,31
        F(I,J)=0.D0   ! Zeroing F
       ENDDO
       DO IJ=1,12
        H(IJ,I)=0.D0  ! Zeroing H
       ENDDO
       XM(I)=X(I)     ! Updating XM
      ENDDO

      COST=COS(X(7))
      SINT=SIN(X(7))
      COSP=COS(X(8))
      SINP=SIN(X(8))
      TANT=SINT/COST
C...
      F(1,2) =  X(6)
      F(1,3) = -X(5)
      F(1,5) = -X(3)
      F(1,6) =  X(2)
      F(1,7) = -G*COST
      F(1,10) = 1.
C...
      F(2,1) = -X(6)
      F(2,3) =  X(4)
      F(2,4) =  X(3)
      F(2,6) = -X(1)
      F(2,7) = -G*SINT*SINP
      F(2,8) =  G*COST*COSP
      F(2,13) = 1.
C...
      F(3,1) =  X(5)
      F(3,2) = -X(4)
      F(3,4) = -X(2)
      F(3,5) =  X(1)
      F(3,7) = -G*SINT*COSP
      F(3,8) = -G*COST*SINP
      F(3,16) = 1.
C...
      F(4,4) =  X(5)*A11
      F(4,5) =  X(4)*A11 + X(6)*A12 + A13
      F(4,6) =  X(5)*A12
      F(4,19) = 1.
      F(4,25) = A14
C...
      F(5,4) =  X(6)*A21-2.*X(4)*A22
      F(5,6) =  X(4)*A21+2.*X(6)*A22-A23
      F(5,22) = 1.
C...
      F(6,4) =  X(5)*A31
      F(6,5) =  X(4)*A31+X(6)*A32+A33
```

```
      F(6,6) =  X(5)*A32
      F(6,19) = A34
      F(6,25) = 1.
C...
      F(7,5) =  COSP
      F(7,6) = -SINP
      F(7,8) = -X(5)*SINP-X(6)*COSP
C...
      F(8,4) = 1.
      F(8,5) = TANT*SINP
      F(8,6) = TANT*COSP
      SECTH2 = 1. + TANT*TANT
      F(8,7) = X(5)*SINP*SECTH2 + X(6)*COSP*SECTH2
      F(8,8) = X(5)*TANT*COSP - X(6)*TANT*SINP
C...
      F(9,1) =   SINT
      F(9,2) = - COST*SINP
      F(9,3) = - COST*COSP
      F(9,7) = -(-X(1)*COST-X(2)*SINT*SINP-X(3)*SINT*COSP)
      F(9,8) = -(X(2)*COST*COSP-X(3)*COST*SINP)

C.....gauss-markov models
      F(10,11) = 1.
      F(11,12) = 1.

      F(13,14) = 1.
      F(14,15) = 1.

      F(16,17) = 1.
      F(17,18) = 1.

      F(19,20) = 1.
      F(20,21) = 1.

      F(22,23) = 1.
      F(23,24) = 1.

      F(25,26) = 1.
      F(26,27) = 1.
C
C**   Covariance Time Propagation
C
      DO I=1,31
       DO J=1,31
        FDT = F(I,J)*DT
        FDT2 = FDT*FDT
        F(I,J)=FDT+0.5*FDT2
       ENDDO
        F(I,I) = F(I,I) + 1.0
      ENDDO
```

```
      ALLOCATE (PD(31,31))   ! PD the derivative of P

      DO I=1,31
       DO J=1,31
        PD(I,J)=0.D0
       DO K=1,31
        PD(I,J)=PD(I,J)+PU(I,K)*F(J,K)     ! P*FT
       ENDDO
       ENDDO
      ENDDO

      DO I=1,31
       DO J=1,31
        PM(I,J)=0.D0
        DO K=1,31
         PM(I,J)=PM(I,J)+F(I,K)*PD(K,J)    ! P=F*P*FT
        ENDDO
       ENDDO
        PM(I,I) = PM(I,I) + QQ(I)          ! PM=F*PU*FT+Q
       ENDDO

      DEALLOCATE(F,PD,PU)
      ALLOCATE (RK(4,21))
C
C      RUNGE-KUTTA INTEGRATION - DYNAMIC SYSTEM - time propagation
      XK=2.
      DO K=1,4
      SINT=SIN(X(7))
      COST=COS(X(7))
      SINP=SIN(X(8))
      COSP=COS(X(8))
      TANT=SINT/COST
      RK(K,1)= DT*(X(6)*X(2)-X(5)*X(3)-G*SINT + X(10))
      RK(K,2)= DT*(X(4)*X(3)-X(6)*X(1)+G*COST*SINP + X(13))
      RK(K,3)= DT*(X(5)*X(1)-X(4)*X(2)+G*COST*COSP + X(16))
      RK(K,4)= DT*(X(4)*X(5)*A11+X(5)*X(6)*A12+X(5)*A13+X(19)
     +        +A14*X(25))
      RK(K,5)= DT*(X(4)*X(6)*A21+(X(6)*X(6)-X(4)*X(4))*A22
     +        - X(6)*A23 + X(22))
      RK(K,6)= DT*(X(4)*X(5)*A31+X(5)*X(6)*A32+X(5)*A33
     +        + X(25)+A34*X(19))
      RK(K,7)= DT*(X(5)*COSP - X(6)*SINP)
      RK(K,8)= DT*(X(4)+X(5)*TANT*SINP + X(6)*TANT*COSP)
      RK(K,9)=-DT*(-X(1)*SINT+X(2)*COST*SINP+X(3)*COST*COSP)
      RK(K,10)=X(11)*DT        !
      RK(K,11)=X(12)*dt        !
      RK(K,12)=X(14)*DT        !
      RK(K,13)=X(15)*dt        !
      RK(K,14)=X(17)*DT        !
      RK(K,15)=X(18)*dt        ! GAUSS-MARKOV PARAMETERS
      RK(K,16)=X(20)*DT        !
```

```
      RK(K,17)=X(21)*dt       !
      RK(K,18)=X(23)*DT       !
      RK(K,19)=X(24)*dt       !
      RK(K,20)=X(26)*DT       !
      RK(K,21)=X(27)*dt       !
C...
      IF(K.GE.3) XK=1.
      X(1)=X(1)+RK(K,1)/XK
      X(2)=X(2)+RK(K,2)/XK
      X(3)=X(3)+RK(K,3)/XK
      X(4)=X(4)+RK(K,4)/XK
      X(5)=X(5)+RK(K,5)/XK
      X(6)=X(6)+RK(K,6)/XK
      X(7)=X(7)+RK(K,7)/XK
      X(8)=X(8)+RK(K,8)/XK

      X(10)=X(10)+RK(K,10)/XK
      X(11)=X(11)+RK(K,11)/XK
      X(13)=X(13)+RK(K,12)/XK
      X(14)=X(14)+RK(K,13)/XK
      X(16)=X(16)+RK(K,14)/XK
      X(17)=X(17)+RK(K,15)/XK
      X(19)=X(19)+RK(K,16)/XK
      X(20)=X(20)+RK(K,17)/XK
      X(22)=X(22)+RK(K,18)/XK
      X(23)=X(23)+RK(K,19)/XK
      X(25)=X(25)+RK(K,20)/XK
      X(26)=X(26)+RK(K,21)/XK
      ENDDO
C     State Estimate Propagation, x(-) calculation
      XM(1)= XM(1) + RK(1,1)/6.+RK(2,1)/3.+RK(3,1)/3.+RK(4,1)/6.
      XM(2)= XM(2) + RK(1,2)/6.+RK(2,2)/3.+RK(3,2)/3.+RK(4,2)/6.
      XM(3)= XM(3) + RK(1,3)/6.+RK(2,3)/3.+RK(3,3)/3.+RK(4,3)/6.
      XM(4)= XM(4) + RK(1,4)/6.+RK(2,4)/3.+RK(3,4)/3.+RK(4,4)/6.
      XM(5)= XM(5) + RK(1,5)/6.+RK(2,5)/3.+RK(3,5)/3.+RK(4,5)/6.
      XM(6)= XM(6) + RK(1,6)/6.+RK(2,6)/3.+RK(3,6)/3.+RK(4,6)/6.
      XM(7)= XM(7) + RK(1,7)/6.+RK(2,7)/3.+RK(3,7)/3.+RK(4,7)/6.
      XM(8)= XM(8) + RK(1,8)/6.+RK(2,8)/3.+RK(3,8)/3.+RK(4,8)/6.
      XM(9)= XM(9) + RK(1,9)/6.+RK(2,9)/3.+RK(3,9)/3.+RK(4,9)/6.
      XM(10)=XM(10)+RK(1,10)/6.+RK(2,10)/3.+RK(3,10)/3.+RK(4,10)/6.
      XM(11)=XM(11)+RK(1,11)/6.+RK(2,11)/3.+RK(3,11)/3.+RK(4,11)/6.
      XM(13)=XM(13)+RK(1,12)/6.+RK(2,12)/3.+RK(3,12)/3.+RK(4,12)/6.
      XM(14)=XM(14)+RK(1,13)/6.+RK(2,13)/3.+RK(3,13)/3.+RK(4,13)/6.
      XM(16)=XM(16)+RK(1,14)/6.+RK(2,14)/3.+RK(3,14)/3.+RK(4,14)/6.
      XM(17)=XM(17)+RK(1,15)/6.+RK(2,15)/3.+RK(3,15)/3.+RK(4,15)/6.
      XM(19)=XM(19)+RK(1,16)/6.+RK(2,16)/3.+RK(3,16)/3.+RK(4,16)/6.
      XM(20)=XM(20)+RK(1,17)/6.+RK(2,17)/3.+RK(3,17)/3.+RK(4,17)/6.
      XM(22)=XM(22)+RK(1,18)/6.+RK(2,18)/3.+RK(3,18)/3.+RK(4,18)/6.
      XM(23)=XM(23)+RK(1,19)/6.+RK(2,19)/3.+RK(3,19)/3.+RK(4,19)/6.
      XM(25)=XM(25)+RK(1,20)/6.+RK(2,20)/3.+RK(3,20)/3.+RK(4,20)/6.
      XM(26)=XM(26)+RK(1,21)/6.+RK(2,21)/3.+RK(3,21)/3.+RK(4,21)/6.
```

```
C...
      DEALLOCATE(RK)

C     DETERMINATION OF  H=dh/dx  .............................
C
      H(1,10) = 1.
C...
      H(2,13) = 1.
C...
      H(3,7)  =  G*SIN(XM(7))*COS(XM(8))
      H(3,8)  =  G*COS(XM(7))*SIN(XM(8))
      H(3,16) = -1.
C...
      H(4,4)  = 1.0D0
C...
      H(5,5)  = 1.0D0
      H(5,31) = 1.0D0
C...
      H(6,6)  = 1.0D0
C...
      C1 = XM(1)-XM(6)*Y1+XM(5)*Z1              ! u
      C2 = XM(2)+XM(6)*X1-XM(4)*Z1              ! v
      C3 = XM(3)-XM(5)*X1+XM(4)*Y1              ! w
      C4 = SQRT(C1*C1 + C2*C2 + C3*C3)          ! V  (Airspeed)
      H(7,1) = C1/C4
      H(7,2) = C2/C4
      H(7,3) = C3/C4
      H(7,4) = (-Z1*C2+Y1*C3)/C4
      H(7,5) = (Z1*C1-X1*C3)/C4
      H(7,6) = (-Y1*C1+X1*C2)/C4
      H(7,28) = 1.0D0
C...
      C5 = 1. + (XM(3)/XM(1))**2
      C6 = 1./C5
      Vt=SQRT(XM(1)*XM(1)+XM(2)*XM(2)+XM(3)*XM(3))
      Vt3=Vt*Vt*Vt
      H(8,1) = -C6*XM(3)/(XM(1)**2) + d*XM(5)*XM(1)/Vt3
      H(8,2) = d*XM(5)*XM(2)/Vt3
      H(8,3) = C6/XM(1) + d*XM(5)*XM(3)/Vt3
      H(8,5) = -D/Vt
      H(8,29) = 1.0D0
C...
      C7 = 1./SQRT(1.-(XM(2)/Vt)**2)
      H(9,1) = -C7*XM(2)*XM(1)/Vt3
      H(9,2) =  C7/Vt-C7*XM(2)*XM(2)/Vt3
      H(9,3) = -C7*XM(2)*XM(3)/Vt3
      H(9,30) = 1.0D0
C...
      H(10,9)  = 1.0D0
C...
      H(11,7)  = 1.0D0
```

```
C...
    H(12,8)  = 1.0D0
C...
    K1=K1+1
    WRITE(12,REC=K1) H(1,10),H(2,13),H(3,7),H(3,8),H(3,16),H(4,4),
   + H(5,5),H(5,31),H(6,6),H(7,1),H(7,2),H(7,3),H(7,4),H(7,5),
   + H(7,6),H(7,28),H(8,1),H(8,2),H(8,3),H(8,5),H(8,29),H(9,1),
   + H(9,2),H(9,3),H(9,30),H(10,9),H(11,7),H(12,8)
C
C**   GAIN CALCULATION
C
    ALLOCATE (PD(31,12))
    DO I=1,31
     DO J=1,12
      PD(I,J)=0.D0              ! PARTIAL PRODUCT PD=P*HT
      DO K=1,31
       PD(I,J)=PD(I,J) + PM(I,K)*H(J,K)
      ENDDO
     ENDDO
    ENDDO

    IN=12
    ALLOCATE (HPHT(IN,IN))
    ALLOCATE (HPHT1(IN,IN))

    DO I=1,12
     DO J=1,12
      HPHT(I,J)=0.D0            ! PARTIAL PRODUCT HPHT=H*P*HT
      HPHT1(I,J)=0.D0
      DO K=1,31
       HPHT(I,J)=HPHT(I,J)+H(I,K)*PD(K,J)
      ENDDO
     ENDDO
       HPHT(I,I) = HPHT(I,I) + RR(I)        ! HPHT + RR
    ENDDO

C...  MATRIX INVERSION

    ALLOCATE (WORK(IN,2*IN))
    ALLOCATE (IDENT(IN,IN))
    CALL INVMAT(HPHT,IN,IN,HPHT1,WORK,IDENT)   ! INVERSE

    DO I=1,12
    K2=K2+1
    WRITE(13,REC=K2) (HPHT(I,N),N=1,12)
    WRITE(14,REC=K2) (HPHT1(I,N),N=1,12)
    ENDDO

    DEALLOCATE (HPHT,WORK,IDENT)
    ALLOCATE (KK(31,12))
C....................................................
```

```
      DO I=1,31
       DO J=1,12
        KK(I,J)=0.D0
         DO K=1,12
          KK(I,J)=KK(I,J)+PD(I,K)*HPHT1(K,J)  ! GAIN MATRIX
         ENDDO
        ENDDO
       ENDDO

      DO I=1,31
       K3=K3+1
       WRITE(15,REC=K3) (KK(I,J),J=1,12)      ! STORE GAIN
      ENDDO

      DEALLOCATE (PD,HPHT1)
C
      READ(2,*) (ZZ(K),K=1,12),Thrust,FN  ! MEASUR. READING
      pdot= XM(4)*XM(5)*A11+XM(5)*XM(6)*A12+XM(5)*A13+XM(19)+A14*XM(25)
      qdot=  XM(4)*XM(6)*A21+(XM(6)*XM(6)-XM(4)*XM(4))*A22- XM(6)*A23
     +       +XM(22)
      rdot=  XM(4)*XM((5)*A31+XM(5)*XM(6)*A32+XM(5)*A3+ XM(25)+A34*XM(19)
      Dax = -(ZZ(6)**2+ZZ(5)**2)*X0+(ZZ(4)*ZZ(5)-rdot)*Y0
     +     +(ZZ(4)*ZZ(6)+qdot)*Z0       ! ax correction to cg
      Day = -(ZZ(4)**2+ZZ(6)**2)*Y0+(ZZ(4)*ZZ(5)+rdot)*X0
     +     +(ZZ(5)*ZZ(6)-pdot)*Z0       ! ay correction to cg
      Daz = -(ZZ(4)**2+ZZ(5)**2)*Z0+(ZZ(4)*ZZ(6)-qdot)*X0
     +     +(ZZ(5)*ZZ(6)+pdot)*Y0       ! az correction to cg.
c
      hh(1)=XM(10) - Dax
      hh(2)=XM(13) - Day
      hh(3)=-XM(16)-G*COS(XM(7))*COS(XM(8)) - Daz
      hh(4)=XM(4)
      hh(5)=XM(5) + XM(31)
      hh(6)=XM(6)
      hh(7)= C4 + XM(28)                        ! V
      hh(8)= ATAN(XM(3)/XM(1)) - D*XM(5)/Vt + XM(29)   ! ALPHA
      hh(9)= ASIN(XM(2)/Vt) + XM(30)                 ! BETA
      hh(10)=XM(9)                          ! ALT
      hh(11)=XM(7)                          ! THETA
      hh(12)=XM(8)                          ! PHI
C...  STATES UPDATE AFTER MEASUREMENT  -  X(+) Calculation
C
      DO I=1,31
       X(I)=0.0
      DO J=1,12
       DH(J)=ZZ(J)-hh(J)         ! FILTER RESIDUAL
       X(I)=X(I)+KK(I,J)*DH(J)
      ENDDO
       X(I)=X(I)+XM(I)           ! STATE UPDATE
      ENDDO
      K4=K4+1
```

```
      WRITE(16,REC=K4) (DH(K),K=1,12)
      WRITE(18,REC=K4) (X(K),K=1,31)
C
C... COVARIANCE UPDATE  - P(+) Calculation
C
      ALLOCATE (KH(31,31))

      DO I=1,31
       DO J=1,31
        KH(I,J)=0.0
        DO K=1,12
         KH(I,J)=KH(I,J)-KK(I,K)*H(K,J)  ! Calculation of K*H
        ENDDO
       ENDDO
        KH(I,I)=1.+KH(I,I)             ! Calculation of I-K*H
      ENDDO

      ALLOCATE (PU(31,31))

      DO I=1,31
       DO J=1,31
        PU(I,J)=0.D0
        DO K=1,31
         PU(I,J)=PU(I,J)+KH(I,K)*PM(K,J)
        ENDDO
       ENDDO
       MM=MM+1
       WRITE(17,REC=MM) (PU(I,LL),LL=1,31)
      ENDDO
C
      VV=SQRT(X(1)**2+X(2)**2+X(3)**2)
      RESV=ZZ(7)-(VV+X(28))            ! V RESIDUAL
      RESA=ZZ(8)-(ATAN(X(3)/X(1))+X(29))  ! INCIDENCE RESIDUAL
      RESB=ZZ(9)-(ASIN(X(2)/VV)+D*X(5)/Vt-X(29)) ! SIDESLIP RESIDUAL
      RESQ=ZZ(5)-X(31)                  ! PITCH RATE RESIDUAL
      WRITE(9,*) X(1),X(3),X(5),X(7)      ! STATES
      WRITE(10,*) X(10),X(16),X(22),X(8)  ! STATES
      WRITE(7,*) X(28),X(29),X(30),X(31)  ! BIAS
c     WRITE(5,*) RESV,RESA,RESB,RESQ      ! RESIDUALS
      DEALLOCATE (KH,KK)
      ENDDO  !<<<<<<<<<<< END OF EKF MAIN LOOP >>>>>>>>>>>>>
C
C... MOD. BRYSON-FRAZIER SMOOTHER..............................
C
      ALLOCATE (F(31,31))
      ALLOCATE (AA(31,31))
      ALLOCATE (HPHT1(12,12))
      ALLOCATE (PD(31,12))
C     Smoother Initialization
      K2=(NAT-1)*12
      DO I=1,12
```

```
 K2=K2+1
 READ(14,REC=K2) (HPHT1(I,K),K=1,12)
ENDDO
DO I=1,31
 DO J=1,12
  PD(I,J)=0.0
  DO K=1,12
   PD(I,J)=PD(I,J)-H(K,I)*HPHT1(K,J)   ! -HT*D1
  ENDDO
 ENDDO
ENDDO
DO I=1,31
 LAM(I)=0.0
  DO K=1,12
   LAM(I)=LAM(I)+PD(I,K)*DH(K)        ! INITIAL LAMBDA
  ENDDO
ENDDO
DO I=1,31
 DO J=1,31
  AA(I,J)=0.0
   DO K=1,12
    AA(I,J)=AA(I,J)-PD(I,K)*H(K,J)    ! INITIAL AA
   ENDDO
 ENDDO
ENDDO
DEALLOCATE (PD,HPHT1)
K1=K1-1
K4=K4-1
WRITE(8,REC=NAT) (X(K),K=1,31)
C
DO ISAM=NAT-1,1,-1   ! .... BACKWARD SMOOTHER ............
WRITE(*,*) 'SMOOTHER - ITERATION',ISAM
ALLOCATE (HPHT1(12,12))

K2=(ISAM-1)*12
DO I=1,12
 K2=K2+1
 READ(14,REC=K2) (HPHT1(I,K),K=1,12)      ! READ HPHT1
ENDDO

DO I=1,31              !  ZEROING  'F' AND 'H'
 DO K=1,31
  F(I,K)=0.D0
 ENDDO
 DO J=1,12
  H(J,I)=0.D0
 ENDDO
ENDDO
READ(12,REC=K1) H(1,10),H(2,13),H(3,7),H(3,8),H(3,16),H(4,4),
+ H(5,5),H(5,31),H(6,6),H(7,1),H(7,2),H(7,3),H(7,4),H(7,5),
+ H(7,6),H(7,28),H(8,1),H(8,2),H(8,3),H(8,5),H(8,29),H(9,1),
```

```
     + H(9,2),H(9,3),H(9,30),H(10,9),H(11,7),H(12,8)
      K1=K1-1

      ALLOCATE (PD(31,12))

      DO I=1,31
       DO J=1,12
        PD(I,J)=0.0
        DO K=1,12
         PD(I,J)=PD(I,J)-H(K,I)*HPHT1(K,J)  ! -HT*D1
        ENDDO
       ENDDO
      ENDDO

      DEALLOCATE (HPHT1)
C..............................................
      COST=COS(X(7))
      SINT=SIN(X(7))
      COSP=COS(X(8))
      SINP=SIN(X(8))
      TANT=SINT/COST
      F(1,2) =  X(6)
      F(1,3) = -X(5)
      F(1,5) = -X(3)
      F(1,6) =  X(2)
      F(1,7) = -G*COST
      F(1,10) = 1.0D0
C...
      F(2,1) = -X(6)
      F(2,3) =  X(4)
      F(2,4) =  X(3)
      F(2,6) = -X(1)
      F(2,7) = -G*SINT*SINP
      F(2,8) =  G*COST*COSP
      F(2,13) = 1.0D0
C...
      F(3,1) =  X(5)
      F(3,2) = -X(4)
      F(3,4) = -X(2)
      F(3,5) =  X(1)
      F(3,7) = -G*SINT*COSP
      F(3,8) = -G*COST*SINP
      F(3,16) = 1.0D0
C...
      F(4,4) =  X(5)*A11
      F(4,5) =  X(4)*A11 + X(6)*A12 + A13
      F(4,6) =  X(5)*A12
      F(4,19) = 1.0D0
      F(4,25) = A14
C...
      F(5,4) =  X(6)*A21-2.*X(4)*A22
```

```
      F(5,6) =  X(4)*A21+2.*X(6)*A22-A23
      F(5,22) = 1.0D0
C...
      F(6,4) =  X(5)*A31
      F(6,5) =  X(4)*A31+X(6)*A32+A33
      F(6,6) =  X(5)*A32
      F(6,19) = A34
      F(6,25) = 1.0D0
C...
      F(7,5) =  COSP
      F(7,6) = -SINP
      F(7,8) = -X(5)*SINP-X(6)*COSP
C...
      F(8,4) = 1.
      F(8,5) = TANT*SINP
      F(8,6) = TANT*COSP
      SECTH2 = 1. + TANT*TANT
      F(8,7) = X(5)*SINP*SECTH2 + X(6)*COSP*SECTH2
      F(8,8) = X(5)*TANT*COSP - X(6)*TANT*SINP
C...
      F(9,1) =   SINT
      F(9,2) = - COST*SINP
      F(9,3) = - COST*COSP
      F(9,7) = -(-X(1)*COST-X(2)*SINT*SINP-X(3)*SINT*COSP)
      F(9,8) = -(X(2)*COST*COSP-X(3)*COST*SINP)
C...
      F(10,11) = 1.0D0
      F(11,12) = 1.0D0
      F(13,14) = 1.0D0
      F(14,15) = 1.0D0
      F(16,17) = 1.0D0
      F(17,18) = 1.0D0
      F(19,20) = 1.0D0
      F(20,21) = 1.0D0
      F(22,23) = 1.0D0
      F(23,24) = 1.0D0
      F(25,26) = 1.0D0
      F(26,27) = 1.0D0

      READ(18,REC=K4) (X(K),K=1,31)    !.............

      READ(16,REC=K4) (DH(K),K=1,12)
      K4=K4-1
C
C   ADJOINT VARIABLES DERIVATIVES CALCULATION
C
      DO I=1,31
       LAMD(I)=0.0
      DO K=1,31
       LAMD(I)=LAMD(I)-F(K,I)*LAM(K)          ! LAMD=-FT*LAM
      ENDDO
```

```
      ENDDO
      DO I=1,31
       DO J=1,31
        PM(I,J)=0.D0
        DO K=1,31
         PM(I,J)=PM(I,J)-F(K,I)*AA(K,J)       ! -(FT*A)
        ENDDO
       ENDDO
      ENDDO

      DO I=1,31
       DO J=1,31
        PM(I,J)=PM(I,J)+PM(J,I)             ! AAD=-(FT*A)-(FT*A)T
       ENDDO
      ENDDO

      DEALLOCATE (F)
      ALLOCATE (AAM(31,31))
C
C... ADJOINT VARIABLES INTEGRATION
C
      DO I=1,31
       LAMBM(I)=LAM(I)+LAMD(I)*(-DT)     ! SIMPLIFIED INTEGRAT.
       DO J=1,31
        AAM(I,J)=AA(I,J)+PM(I,J)*(-DT)   ! SIMPLIFIED INTEGRAT.
       ENDDO
      ENDDO

      DEALLOCATE (AA)
      ALLOCATE (KK(31,12))
      ALLOCATE (HPHT(12,12))
C...
      K3=(ISAM-1)*31
      DO I=1,31
      K3=K3+1
      READ(15,REC=K3) (KK(I,J),J=1,12)    ! READ KK  - GAIN
      ENDDO

      K2=(ISAM-1)*12
      DO I=1,12
      K2=K2+1
      READ(13,REC=K2) (HPHT(I,J),J=1,12)  ! READ HPHT
      ENDDO
C
C... ADJOINT VARIABLES UPDATING
C
      DO I=1,12
       hh(I)=0.D0
       DO K=1,31
        hh(I)=hh(I)+KK(K,I)*LAMBM(K)      ! KKT*LAMB
       ENDDO
```

```
      ENDDO
      DO I=1,12
       ZZ(I)=0.D0
      DO K=1,12
       ZZ(I)=ZZ(I)+HPHT(I,K)*hh(K)       ! D*KKT*LAMB
      ENDDO
       ZZ(I)=ZZ(I)+DH(I)               ! (DH+D*KKT*LAMBM)
      ENDDO

      DO I=1,31
        DLAMB=0.0
      DO K=1,12
        DLAMB=DLAMB+PD(I,K)*ZZ(K)
      ENDDO
        LAM(I)=LAMBM(I)+DLAMB             ! LAMBDA UPDATE
      ENDDO

C...
      ALLOCATE (F(31,31))

      DO I=1,31
       DO J=1,31
        F(I,J)=0.D0
         DO K=1,12
          F(I,J)=F(I,J)- PD(I,K)*H(K,J)       ! HT*D1*H
         ENDDO
        ENDDO
      ENDDO

      ALLOCATE (KH(31,31))
      DEALLOCATE (PD)

      DO I=1,31
       DO J=1,31
        KH(I,J)=0.D0
         DO K=1,12
          KH(I,J)=KH(I,J)-KK(I,K)*H(K,J)
         ENDDO
       ENDDO
        KH(I,I)=1.0+KH(I,I)               ! I-KH CALCULATION
      ENDDO

      DEALLOCATE (KK,HPHT)
      ALLOCATE (PD(31,31))

      DO I=1,31
       DO J=1,31
        PD(I,J)=0.D0
         DO K=1,31
          PD(I,J)=PD(I,J)+AAM(I,K)*KH(K,J)   ! AAM*(I-KH)
         ENDDO
```

```
      ENDDO
     ENDDO

     ALLOCATE (AA(31,31))

     DO I=1,31
      DO J=1,31
       AA(I,J)=0.D0
        DO K=1,31
         AA(I,J)=AA(I,J)+KH(K,I)*PD(K,J)     ! (I-KH)T*A*(I-KH)
        ENDDO
         AA(I,J)=AA(I,J)+F(I,J)             ! FINAL A UPDATE
      ENDDO
     ENDDO

     DEALLOCATE (KH)
C
C... STATE SMOOTHING, COVARIANCE UPDATE .......................
C
     MM=(ISAM-1)*31
     DO L=1,31
      MM=MM+1
      READ(17,REC=MM) (PU(L,J),J=1,31)      ! READ PU (COVARIANCE)
     ENDDO

     DO I=1,31
      DDS=0.0
       DO K=1,31
        DDS=DDS+PU(I,K)*LAMBM(K)
       ENDDO
       XM(I)=X(I)-DDS                       ! SMOOTHED STATE
     ENDDO

     DO I=1,31
      DO J=1,31
       PD(I,J)=0.D0
        DO K=1,31
         PD(I,J)=PD(I,J)+AAM(I,K)*PU(K,J)    !  A*PU
        ENDDO
      ENDDO
     ENDDO
     DO I=1,31
      DO J=1,31
       F(I,J)=0.D0
       DO K=1,31
        F(I,J)=F(I,J)+PU(I,K)*PD(K,J)        !  PU*A*PU
       ENDDO
        PM(I,J)=PU(I,J)-F(I,J)               ! COVARIACE - SMOOTHER
      ENDDO
     ENDDO
     WRITE(8,REC=ISAM)(XM(I),I=1,31)
```

```
      DEALLOCATE (PD,AAM)
      ENDDO ! <<<<<<<<<<<<<<< END OF SMOOTHER >>>>>>>>>>>>>>>>>
C
C     READING, REORD. AND PRINTING THE RESULTS
C
      DO I=1,NAT
       READ(8,REC=I) (X(K),K=1,31)
       X10=(X(10)-Thrust/MASS)*MASS         ! X FORCE
       X16=(X(16)-FN/MASS)*MASS             ! Z FORCE
       X22=(X(22)-Lze*Thrust/Iy-1.94*FN/Iy)*Iy ! PITCH MOMENT
       WRITE(3,*) X(1),X(3),X(5),X(7)       ! STATES
       WRITE(4,*) X10,X16,X22,X(8)          ! STATES
C      WRITE(5,*) X(28),X(29),X(30),X(31)   ! BIAS
      ENDDO
      CLOSE(4)
      CLOSE(6)
      CLOSE(8)
      CLOSE(12)
      CLOSE(13)
      CLOSE(14)
      CLOSE(15)
      CLOSE(16)
      CLOSE(17)
      CLOSE(18)
      STOP
      END
C>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
C
      SUBROUTINE INVMAT(A,IAR,IAC,AINV,WORK,IDENT [REFERENCE])
C
C     Matrix inversion - max xx*xx matrix
c
      INTEGER*2 IAR,IAC
      REAL*8 A(IAR,IAC),WORK(IAR,2*IAC),AINV(IAR,IAC),IDENT(IAR,IAC)
      REAL*8 WKDIV,WKMULT
C
C ... N = NUMBER OF ROWS    (I)
C ... M = NUMBER OF COLUMNS (J)
C ... N = M OR CANNOT INVERT THE MATRIX A
C
      N = IAR
      M = IAC
C
C     TO CREATE THE APPROPRIATE IDENTITY MATRIX In=IDENT(N,M)

      DO 20 I=1,N
       DO 10 J=1,M
         IDENT(I,J)=0.0
 10    CONTINUE
         IDENT(I,I)=1.0
 20 CONTINUE
```

```
C ... TO ADJOIN THE A AND IDENT MATRICES

      MDASH=2*M
      DO 40 I=1,N
       DO 30 J=1,M
          WORK(I,J)=A(I,J)
          WORK(I,M+J)=IDENT(I,J)
30       CONTINUE
40     CONTINUE

C ... TO MAKE WORK(1,1)=1.0

      WKDIV=WORK(1,1)

      DO 50 J=1,MDASH
        WORK(1,J)=WORK(1,J)/WKDIV
50     CONTINUE

C ...  TO MAKE ZEROS BELOW DIAGONAL OF LHS OF MATRIX WORK

      DO 90 I=2,N
        DO 70 K=I,N
          WKMULT=WORK(K,I-1)
          DO 60 J=1,MDASH
            WORK(K,J)=WORK(K,J)-(WKMULT*WORK((I-1),J))
60           CONTINUE
70         CONTINUE
          WKDIV=WORK(I,I)
          DO 80 J=I,MDASH
            WORK(I,J)=WORK(I,J)/WKDIV
80           CONTINUE
90     CONTINUE

C ... TO GET THE UPPER LHS TO ZEROS

      DO 130 K=N,2,-1

        DO 120 I=1,K-1
          WKMULT=WORK(I,K)

          DO 110 J=1,MDASH
            WORK(I,J)=WORK(I,J)-(WKMULT*WORK(K,J))
110           CONTINUE
120         CONTINUE
130     CONTINUE

C ... TO EXTRACT INVERSE MATRIX ON RHS AND
C ... MULTIPLY BY ORIGINAL TO SEE ACCURACY OF IDENTITY

      DO 150 I=1,N
```

```
      DO 140 J=M+1,MDASH
        AINV(I,J-M)=WORK(I,J)
140     CONTINUE
150   CONTINUE
      RETURN
      END
```

4.2 Program IEKF.FOR

```
C    ************ P R O G R A M   I E K F.FOR ******************
C                        Hoff March/94
C                        REV. NOV/94, APR/95
C
C    Estimation of aircraft states u,v,w,p,q,r,X,Y,Z,L,M,N
C    alpha,beta using an Iterated Extended  Kalman  Filter
C    applied to an inertial and gravitational model.
C
C    Bias terms to Ax, Az, q, alpha, Beta, Theta.
C
C    Needs two data files
c   (i) File with initial conditions, process and measurement
C       noise covariances, CG position, aircraft Inertias,
C       Mass and engine z arm to CG.
C   (ii) File with flight data: Ax, Ay, Az, p, q, r, True airpeed,
C        Alpha, Beta, Altitude, theta, Phi, Thrust, Propeller
C        Normal Force (metric units, angles in rad.).
C    Notation:
C    NAT   - total number of samples
C    DT    - sample interval in seconds.
C    CG    - aircraft CG (%).
C    X(i)  - updated state variable
C    XM(i) - time propagated state variable
C    PM(i) - time propagated covariance matrix
C    PU(i) - updated covariance matrix
C    RR(i) - measurement noise covarariance matrix
C    QQ(i) - process noise covariance matrix
C    H(i,j) - gradient of the observation model matrix
C    F(i,j) - gradient of the dynamic model matrix
C    KK(i,j) - Kalman gain matrix
C    HPHT - matrix product of H*PM*H'
C    HPHT1 - inverse of HPHT
C    Work, Ident, PD - auxiliary matices
C    RK - auxiliary matrix of Runge-Kutta integration
C    KH,VV,PD - auxiliary matrix
C    MASS - Aircraft mass
C    Lze - Engine z coordinate relative to CG.

      REAL*8 H(12,33),PM(33,33),RR(12),QQ(33),hh(12),ZZ(14),XM(33)
      REAL*4 X(33),V(33),Ix,Iy,Iz,Ixz,I2,MASS,Lze
      REAL*8 WORK[ALLOCATABLE] (:,:),   RK[ALLOCATABLE] (:,:),
     +       PD[ALLOCATABLE] (:,:),    F[ALLOCATABLE] (:,:),
     +       KK[ALLOCATABLE] (:,:), HPHT1[ALLOCATABLE] (:,:),
     +     HPHT[ALLOCATABLE] (:,:),   KH[ALLOCATABLE] (:,:),
     +       PU[ALLOCATABLE] (:,:), IDENT[ALLOCATABLE] (:,:)
      INTEGER*2 IN

      OPEN(UNIT=4,FILE='PHU809A.DAT',STATUS='OLD')
```

```
      OPEN(UNIT=6,FILE='PHU809M.DAT',STATUS='OLD')
      OPEN(UNIT=5,FILE='IEKFDAT.DAT')
      OPEN(UNIT=7,FILE='IEKDAT1.DAT')
      OPEN(UNIT=8,FILE='IEKDAT2.DAT')
      OPEN(UNIT=9,FILE='IEKDAT3.DAT')
      OPEN(UNIT=10,FILE='IEKDAT4.DAT')
      OPEN(UNIT=11,FILE='IEKDAT5.DAT')
      OPEN(UNIT=12,FILE='IEKDAT6.DAT')
      He=0.0
      g=9.80665

C...  INITIALIZATION - Matrix in a data file with all initial values
C                obtained averaging the stabilization
      READ(6,*) (X(K),K=1,7)    !
      READ(6,*) (X(K),K=8,14)   !
      READ(6,*) (X(K),K=15,21)  ! STATES INITIALIZATION
      READ(6,*) (X(K),K=22,28)  !
      READ(6,*) (X(k),k=29,33)  !
C
      READ(6,*) (RR(I),I=1,6)   ! MEASUR. NOISE COVAR. MATRIX
      READ(6,*) (RR(I),I=7,12)  ! ASSUMED DIAGONAL
C
      READ(6,*) (QQ(J),J=1,7)   !
      READ(6,*) (QQ(J),J=8,14)  ! PROCESS NOISE COVAR. MATRIX
      READ(6,*) (QQ(J),J=15,21) ! ASSUMED DIAGONAL
      READ(6,*) (QQ(J),J=22,28) !
      READ(6,*) (QQ(J),J=29,33) !
C
      ALLOCATE (PU(33,33))

C   INITIAL COVAR. MATRIX
      DO J=1,33
       DO I=1,33
        PU(I,J)=0.D0      ! Zeroing
       ENDDO
      ENDDO
      jk=0
      DO I=1,4
       READ(6,*) (PU(J,J),J=1+jk,7+jk) ! Reading initial cov. matrix
       jk=jk+7
      ENDDO
       READ(6,*) (PU(J,J),J=29,33)

      READ(6,*) NAT,DT,CG,Ix,Iy,Iz,Ixz,MASS,Lze
C
      Xv = 7.1325 + (CG-30)*1.717/100. ! POSITION INCID./SIDES. VANE
      Zv = 0.57
      D = SQRT(Xv**2 + Zv**2)        ! Vane approx. dist to cg.
      X1 = Xv-0.35                ! PITOT POSIT. X  m
      Y1 = 0.                    ! "      "  Y  m
      Z1 =  0.57                 ! "      "  Z  m (APPROX.)
```

```
      X0 = 1.717*(CG-85.3)/100.        ! IRS position x (average)
      Y0 = -.584                 !    "    y
      Z0 = 0.236                 !    "    z
c               Valid for the Jetstream G-AXUI
c
      I2=Ix*Iz-Ixz*Ixz
      A11=Ixz*(Iz+Ix-Iy)/I2
      A12=(Iz*(Iy-Iz)-Ixz*Ixz)/I2
      A13=Ixz*He/I2
      A14=Ixz/Ix
      A21=(Iz-Ix)/Iy
      A22=Ixz/Iy
      A23=He/Iy
      A31=(Ix*(Ix-Iy)+Ixz*Ixz)/I2
      A32=(Ixz*(Iy-Iz-Ix))/I2
      A33=Ix*He/I2
      A34=Ixz/Iz
C
      DO ISAMPLE=1,NAT    !<<<<<<< MAIN LOOP >>>>>>>>>>>>>>>>>>>>>>
      WRITE(*,*) 'SAMPLE No.',ISAMPLE

      ALLOCATE (F(33,33))
C
C***  F MATRIX   F=df/dx  **********************************
C
      DO I=1,33
       DO J=1,33
        F(I,J)=0.D0      ! Zeroing F
       ENDDO
       DO IJ=1,12
        H(IJ,I)=0.D0     ! Zeroing H
       ENDDO
       XM(I)=X(I)       ! Updating XM
      ENDDO

      COST=COS(X(7))
      SINT=SIN(X(7))
      COSP=COS(X(8))
      SINP=SIN(X(8))
      TANT=SINT/COST
C
C...  F calculation
C
      F(1,2) =  X(6)
      F(1,3) = -X(5)
      F(1,5) = -X(3)
      F(1,6) =  X(2)
      F(1,7) = -G*COST
      F(1,10) = 1.
C...
      F(2,1) = -X(6)
```

```
      F(2,3) =  X(4)
      F(2,4) =  X(3)
      F(2,6) = -X(1)
      F(2,7) = -G*SINT*SINP
      F(2,8) =  G*COST*COSP
      F(2,13) = 1.
C...
      F(3,1) =  X(5)
      F(3,2) = -X(4)
      F(3,4) = -X(2)
      F(3,5) =  X(1)
      F(3,7) = -G*SINT*COSP
      F(3,8) = -G*COST*SINP
      F(3,16) = 1.
C...
      F(4,4) =  X(5)*A11
      F(4,5) =  X(4)*A11 + X(6)*A12 + A13
      F(4,6) =  X(5)*A12
      F(4,19) = 1.
      F(4,25) = A14
C...
      F(5,4) =  X(6)*A21-2.*X(4)*A22
      F(5,6) =  X(4)*A21+2.*X(6)*A22-A23
      F(5,22) = 1.
C...
      F(6,4) =  X(5)*A31
      F(6,5) =  X(4)*A31+X(6)*A32+A33
      F(6,6) =  X(5)*A32
      F(6,19) = A34
      F(6,25) = 1.
C...
      F(7,5) =  COSP
      F(7,6) = -SINP
      F(7,8) = -X(5)*SINP-X(6)*COSP
C...
      F(8,4) = 1.
      F(8,5) = TANT*SINP
      F(8,6) = TANT*COSP
      SECTH2 = 1. + TANT*TANT
      F(8,7) = X(5)*SINP*SECTH2 + X(6)*COSP*SECTH2
      F(8,8) = X(5)*TANT*COSP - X(6)*TANT*SINP
C...
      F(9,1) =   SINT
      F(9,2) = - COST*SINP
      F(9,3) = - COST*COSP
      F(9,7) = -(-X(1)*COST-X(2)*SINT*SINP-X(3)*SINT*COSP)
      F(9,8) = -(X(2)*COST*COSP-X(3)*COST*SINP)
C.....gauss-markov models
      F(10,11) = 1.
      F(11,12) = 1.
```

```
      F(13,14) = 1.
      F(14,15) = 1.

      F(16,17) = 1.
      F(17,18) = 1.

      F(19,20) = 1.
      F(20,21) = 1.

      F(22,23) = 1.
      F(23,24) = 1.

      F(25,26) = 1.
      F(26,27) = 1.
C
C**   Gradient of the dynamic model.
C
      DO I=1,33
       DO J=1,33
        FDT=   F(I,J)*DT
        FDT2=  FDT*FDT
        F(I,J)= FDT+0.5*FDT2
       ENDDO
        F(I,I) = F(I,I) + 1.0
      ENDDO
C
C...  Covariance time propagation

      ALLOCATE (PD(33,33))

      DO I=1,33
       DO J=1,33
         PD(I,J)=0.D0
        DO K=1,33
         PD(I,J)=PD(I,J)+PU(I,K)*F(J,K)     ! P*FT
        ENDDO
       ENDDO
      ENDDO

      DO I=1,33
       DO J=1,33
        PM(I,J)=0.D0
        DO K=1,33
         PM(I,J)=PM(I,J)+F(I,K)*PD(K,J)    ! P=F*P*FT
        ENDDO
       ENDDO
        PM(I,I) = PM(I,I) + QQ(I)         ! PM=F*PU*FT+Q
      ENDDO

      DEALLOCATE(F,PD,PU)
      ALLOCATE (RK(4,21))
```

```
C
C    RUNGE-KUTTA INTEGRATION - DYNAMIC SYSTEM - time propagation
      XK=2.
      DO K=1,4
      SINT=SIN(X(7))
      COST=COS(X(7))
      SINP=SIN(X(8))
      COSP=COS(X(8))
      TANT=SINT/COST
      RK(K,1)= DT*(X(6)*X(2)-X(5)*X(3)-G*SINT + X(10))
      RK(K,2)= DT*(X(4)*X(3)-X(6)*X(1)+G*COST*SINP + X(13))
      RK(K,3)= DT*(X(5)*X(1)-X(4)*X(2)+G*COST*COSP + X(16))
      RK(K,4)= DT*(X(4)*X(5)*A11+X(5)*X(6)*A12+X(5)*A13+X(19)
     +        +A14*X(25))
      RK(K,5)= DT*(X(4)*X(6)*A21+(X(6)*X(6)-X(4)*X(4))*A22
     +        - X(6)*A23 + X(22))
      RK(K,6)= DT*(X(4)*X(5)*A31+X(5)*X(6)*A32+X(5)*A33
     +        + X(25)+A34*X(19))
      RK(K,7)= DT*(X(5)*COSP - X(6)*SINP)
      RK(K,8)= DT*(X(4)+X(5)*TANT*SINP + X(6)*TANT*COSP)
      RK(K,9)=-DT*(-X(1)*SINT+X(2)*COST*SINP+X(3)*COST*COSP)
      RK(K,10)=X(11)*DT        !
      RK(K,11)=X(12)*dt        !
      RK(K,12)=X(14)*DT        !
      RK(K,13)=X(15)*dt        !
      RK(K,14)=X(17)*DT        !
      RK(K,15)=X(18)*dt        ! GAUSS-MARKOV PARAMETERS
      RK(K,16)=X(20)*DT        !
      RK(K,17)=X(21)*dt        !
      RK(K,18)=X(23)*DT        !
      RK(K,19)=X(24)*dt        !
      RK(K,20)=X(26)*DT        !
      RK(K,21)=X(27)*dt        !
C...
      IF(K.GE.3) XK=1.
      X(1)=X(1)+RK(K,1)/XK
      X(2)=X(2)+RK(K,2)/XK
      X(3)=X(3)+RK(K,3)/XK
      X(4)=X(4)+RK(K,4)/XK
      X(5)=X(5)+RK(K,5)/XK
      X(6)=X(6)+RK(K,6)/XK
      X(7)=X(7)+RK(K,7)/XK
      X(8)=X(8)+RK(K,8)/XK

      X(10)=X(10)+RK(K,10)/XK
      X(11)=X(11)+RK(K,11)/XK
      X(13)=X(13)+RK(K,12)/XK
      X(14)=X(14)+RK(K,13)/XK
      X(16)=X(16)+RK(K,14)/XK
      X(17)=X(17)+RK(K,15)/XK
      X(19)=X(19)+RK(K,16)/XK
```

```
      X(20)=X(20)+RK(K,17)/XK
      X(22)=X(22)+RK(K,18)/XK
      X(23)=X(23)+RK(K,19)/XK
      X(25)=X(25)+RK(K,20)/XK
      X(26)=X(26)+RK(K,21)/XK
      ENDDO
C     State Estimate Propagation,  x(-) calculation
      XM(1)= XM(1) + RK(1,1)/6.+RK(2,1)/3.+RK(3,1)/3.+RK(4,1)/6.
      XM(2)= XM(2) + RK(1,2)/6.+RK(2,2)/3.+RK(3,2)/3.+RK(4,2)/6.
      XM(3)= XM(3) + RK(1,3)/6.+RK(2,3)/3.+RK(3,3)/3.+RK(4,3)/6.
      XM(4)= XM(4) + RK(1,4)/6.+RK(2,4)/3.+RK(3,4)/3.+RK(4,4)/6.
      XM(5)= XM(5) + RK(1,5)/6.+RK(2,5)/3.+RK(3,5)/3.+RK(4,5)/6.
      XM(6)= XM(6) + RK(1,6)/6.+RK(2,6)/3.+RK(3,6)/3.+RK(4,6)/6.
      XM(7)= XM(7) + RK(1,7)/6.+RK(2,7)/3.+RK(3,7)/3.+RK(4,7)/6.
      XM(8)= XM(8) + RK(1,8)/6.+RK(2,8)/3.+RK(3,8)/3.+RK(4,8)/6.
      XM(9)= XM(9) + RK(1,9)/6.+RK(2,9)/3.+RK(3,9)/3.+RK(4,9)/6.
      XM(10)=XM(10)+RK(1,10)/6.+RK(2,10)/3.+RK(3,10)/3.+RK(4,10)/6.
      XM(11)=XM(11)+RK(1,11)/6.+RK(2,11)/3.+RK(3,11)/3.+RK(4,11)/6.
      XM(13)=XM(13)+RK(1,12)/6.+RK(2,12)/3.+RK(3,12)/3.+RK(4,12)/6.
      XM(14)=XM(14)+RK(1,13)/6.+RK(2,13)/3.+RK(3,13)/3.+RK(4,13)/6.
      XM(16)=XM(16)+RK(1,14)/6.+RK(2,14)/3.+RK(3,14)/3.+RK(4,14)/6.
      XM(17)=XM(17)+RK(1,15)/6.+RK(2,15)/3.+RK(3,15)/3.+RK(4,15)/6.
      XM(19)=XM(19)+RK(1,16)/6.+RK(2,16)/3.+RK(3,16)/3.+RK(4,16)/6.
      XM(20)=XM(20)+RK(1,17)/6.+RK(2,17)/3.+RK(3,17)/3.+RK(4,17)/6.
      XM(22)=XM(22)+RK(1,18)/6.+RK(2,18)/3.+RK(3,18)/3.+RK(4,18)/6.
      XM(23)=XM(23)+RK(1,19)/6.+RK(2,19)/3.+RK(3,19)/3.+RK(4,19)/6.
      XM(25)=XM(25)+RK(1,20)/6.+RK(2,20)/3.+RK(3,20)/3.+RK(4,20)/6.
      XM(26)=XM(26)+RK(1,21)/6.+RK(2,21)/3.+RK(3,21)/3.+RK(4,21)/6.
C...
      DEALLOCATE(RK)
      DO L=1,33
       X(L)=XM(L)
      ENDDO

      DO ITER=1,2  ! >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
C     DETERMINATION OF  H=dh/dx  ..............................
C
      H(1,10) = 1.
      H(1,33) = 1.
C...
      H(2,13) = 1.
C...
      H(3,7)  =  G*SIN(X(7))*COS(X(8))
      H(3,8)  =  G*COS(X(7))*SIN(X(8))
      H(3,16) = -1.
      H(3,28)=1.
C...
      H(4,4)  = 1.
C...
      H(5,5)  = 1.
      H(5,31) = 1.
```

```
C...
      H(6,6)  = 1.
C...
      C1 = X(1)-X(6)*Y1+X(5)*Z1             ! u
      C2 = X(2)+X(6)*X1-X(4)*Z1             ! v
      C3 = X(3)-X(5)*X1+X(4)*Y1             ! w
      C4 = SQRT(C1*C1 + C2*C2 + C3*C3)         ! V  (Airspeed)
      H(7,1) = C1/C4
      H(7,2) = C2/C4
      H(7,3) = C3/C4
      H(7,4) = (-Z1*C2+Y1*C3)/C4
      H(7,5) = (Z1*C1-X1*C3)/C4
      H(7,6) = (-Y1*C1+X1*C2)/C4
C...
      C5 = 1. + (X(3)/X(1))**2
      C6 = 1./C5
      Vt=SQRT(X(1)*X(1)+X(2)*X(2)+X(3)*X(3))
      Vt3=Vt*Vt*Vt
      H(8,1) = -C6*X(3)/(X(1)*X(1)) + d*X(5)*X(1)/Vt3
      H(8,2) = d*X(5)*X(2)/Vt3
      H(8,3) = C6/X(1) + D*X(5)*X(3)/Vt3
      H(8,5) =- d/Vt
      H(8,29) = 1.0
C...
      C7 = 1./SQRT(1.-(X(2)/Vt)**2)
      H(9,1) = -C7*X(2)*X(1)/Vt3
      H(9,2) =  C7/Vt-C7*X(2)*X(2)/Vt3
      H(9,3) = -C7*X(2)*X(3)/Vt3
      H(9,30) = 1.
C...
      H(10,9)  = 1.
C...
      H(11,7)  = 1.
      H(11,32) = 1.
C...
      H(12,8)  = 1.
C
C**   GAIN CALCULATION
C
      ALLOCATE (PD(33,12))
      DO I=1,33
       DO J=1,12
        PD(I,J)=0.D0                ! PARTIAL PRODUCT PD=P*HT
        DO K=1,33
         PD(I,J)=PD(I,J) + PM(I,K)*H(J,K)
        ENDDO
       ENDDO
      ENDDO

      IN=12
      ALLOCATE (HPHT(IN,IN))
```

```
      ALLOCATE (HPHT1(IN,IN))

      DO I=1,12
       DO J=1,12
        HPHT(I,J)=0.D0           ! PARTIAL PRODUCT HPHT=H*P*HT
        HPHT1(I,J)=0.D0
        DO K=1,33
         HPHT(I,J)=HPHT(I,J)+H(I,K)*PD(K,J)
        ENDDO
       ENDDO
        HPHT(I,I) = HPHT(I,I) + RR(I)          ! HPHT + RR
      ENDDO

C...  MATRIX INVERSION

      ALLOCATE (WORK(IN,2*IN))
      ALLOCATE (IDENT(IN,IN))
      CALL INVMAT(HPHT,IN,IN,HPHT1,WORK,IDENT)   ! INVERSE
      DEALLOCATE (HPHT,WORK,IDENT)
      ALLOCATE (KK(33,12))
C
      DO I=1,33
       DO J=1,12
        KK(I,J)=0.D0
        DO K=1,12
          KK(I,J)=KK(I,J)+PD(I,K)*HPHT1(K,J)    ! GAIN MATRIX
        ENDDO
       ENDDO
      ENDDO
      DEALLOCATE (PD,HPHT1)
      IF(ITER.EQ.1) THEN
      pdot= X(4)*X(5)*A11+X(5)*X(6)*A12+X(5)*A13+X(19)+A14*X(25)
      qdot=  X(4)*X(6)*A21+(X(6)*X(6)-X(4)*X(4))*A22- X(6)*A23 + X(22)
      rdot=  X(4)*X((5)*A31+X(5)*X(6)*A32+X(5)*A3+ X(25)+A34*X(19)
       READ(4,*) (ZZ(L),L=1,12),Thrust,FN   ! MEASUREMENTS READING
       Dax =-(zz(6)**2+zz(5)**2)*X0+(zz(4)*zz(5)-rdot)*Y0
     +      +(zz(4)*zz(6)+qdot)*Z0       ! ax correction to cg
       Day =-(zz(4)**2+zz(6)**2)*Y0+(zz(4)*zz(5)+rdot)*X0
     +      +(zz(5)*zz(6)-pdot)*Z0       ! ay correction to cg
       Daz =-(zz(4)**2+zz(5)**2)*Z0+(zz(4)*zz(6)-qdot)*X0
     +      +(zz(5)*zz(6)+pdot)*Y0       ! az correction to cg.
       ZZ(1)=ZZ(1)-Dax
       ZZ(2)=ZZ(2)-DaY
       ZZ(3)=ZZ(3)-DaZ
      ENDIF
      hh(1)=X(10) +  X(33)                  ! Ax
      hh(2)=X(13)                          ! Ay
      hh(3)=-X(16)-G*COS(X(7))*COS(X(8)) + X(28)      ! Az
      hh(4)=X(4)                           ! p
      hh(5)=X(5) + X(31)                     ! q
      hh(6)=X(6)                           ! r
```

```
      hh(7)= C4 ! + X(29)                     ! V
      hh(8)= ATAN(X(3)/X(1)) - D*X(5)/Vt + X(29)       ! ALPHA
      hh(9)= ASIN(X(2)/Vt) + X(30)                 ! BETA
      hh(10)=X(9)                      ! ALT
      hh(11)=X(7)+x(32)                    ! THETA
      hh(12)=X(8)                      ! PHI
C
C...  STATES UPDATE AFTER MEASUREMENT  -  X(+) Calculation
C
     DO I=1,12
      V(I)=0.0
     DO K=1,33
      V(I)=V(I)+H(I,K)*(XM(K)-X(K))
     ENDDO
     ENDDO
C
     DO I=1,33
      DDX=0.0
     DO J=1,12
       DDX=DDX+KK(I,J)*(ZZ(J)-hh(J)-V(J))
     ENDDO
      X(I)=DDX+XM(I)            ! STATES UPDATING
     ENDDO
     IF(ITER.EQ.1) DEALLOCATE (KK)
     ENDDO  !  >>>>>>  END OF IEKF ITERATION   >>>>>>>>>>
C
C...  COVARIANCE UPDATE   -  P(+) Calculation
C
     ALLOCATE (KH(33,33))

     DO I=1,33
      DO J=1,33
       KH(I,J)=0.0
       DO K=1,12
        KH(I,J)=KH(I,J)+KK(I,K)*H(K,J)  ! Calculation of K*H
       ENDDO
        KH(I,J)=-KH(I,J)
       IF(I.EQ.J) KH(I,J)=1.+KH(I,J)   ! Calculation of I-K*H
      ENDDO
     ENDDO

     ALLOCATE (PU(33,33))

     DO I=1,33
      DO J=1,33
       PU(I,J)=0.D0
      DO K=1,33
       PU(I,J)=PU(I,J)+KH(I,K)*PM(K,J)  ! Updated covariance
      ENDDO
     ENDDO
    ENDDO
```

```
C
C... Filter residuals
      vvv=sqrt(x(1)**2+x(2)**2+x(3)**2)
      resv=zz(7)-vvv ! Residual of V
      resa=zz(8)-ATAN(x(3)/X(1))-x(29)+D*x(5)/vvv ! Res. of alpha
      resq=zz(5)-x(5)-X(31) ! Residual of q
      resb=ZZ(9)-ASIN(X(2)/VVV)-X(30) ! Residual of sideslip
      DEALLOCATE (KH,KK)
      ALFA=ATAN(X(3)/X(1))
      BETA=ASIN(X(2)/VVV)
      resaz=zz(3)-(-X(16)-G*COS(X(7))*COS(X(8))+X(28)) ! Res. of Az
C
C... Removing the thrust and moments due to thrust/Prop Normal Force
      X10=X(10)-Thrust/MASS
      X16=X(16)-FN/MASS
      X22=X(22)-Lze*Thrust/Iy-1.94*FN/Iy
C
C... Printing the Data
C
C     WRITE(5 = U,W,Q,THETA,V,PHI,X,Z,M ..........long. analysis
      WRITE(5,*) X(1),X(3),X(5),X(7),X(2),X(8),X10,X16,X22
      WRITE(7,*) X(1),X(3),X(5),X(7)
      WRITE(8,*) X(10),X(16),X(22),X(9)
      WRITE(9,1234) x(28),x(29),x(30),X(31),X(32)
      write(10,1234) resv,resa,RESB,resq,resaz,x(33)
 1234 FORMAT(6F12.7)
      WRITE(11,1234) PU(28,28),PU(30,30),PU(31,31),PU(32,32)
      WRITE(12,1234) PU(33,33),PU(10,10),PU(16,16),PU(22,22)
C
      ENDDO     !<<<<<<<<< MAIN LOOP >>>>>>>>>>
      CLOSE(6)
      CLOSE(7)
      CLOSE(8)
      CLOSE(9)
      CLOSE(10)
      CLOSE(11)
      CLOSE(12)
      CLOSE(14)
      STOP
      END
C>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
C
      SUBROUTINE INVMAT(A,IAR,IAC,AINV,WORK,IDENT [REFERENCE])
C
C     Matrix inversion - max xx*xx matrix
c
      INTEGER*2 IAR,IAC
      REAL*8 A(IAR,IAC),WORK(IAR,2*IAC),AINV(IAR,IAC),IDENT(IAR,IAC)
      REAL*8 WKDIV,WKMULT
C
C ... N = NUMBER OF ROWS    (I)
```

```
C ... M = NUMBER OF COLUMNS (J)
C ... N = M OR CANNOT INVERT THE MATRIX A
C
      N = IAR
      M = IAC
C
C   TO CREATE THE APPROPRIATE IDENTITY MATRIX In=IDENT(N,M)

      DO 20 I=1,N
        DO 10 J=1,M
          IDENT(I,J)=0.0
  10    CONTINUE
          IDENT(I,I)=1.0
  20 CONTINUE

C ... TO ADJOIN THE A AND IDENT MATRICES

      MDASH=2*M
      DO 40 I=1,N
       DO 30 J=1,M
          WORK(I,J)=A(I,J)
          WORK(I,M+J)=IDENT(I,J)
30      CONTINUE
40    CONTINUE

C ... TO MAKE WORK(1,1)=1.0

      WKDIV=WORK(1,1)

      DO 50 J=1,MDASH
        WORK(1,J)=WORK(1,J)/WKDIV
50    CONTINUE

C ...  TO MAKE ZEROS BELOW DIAGONAL OF LHS OF MATRIX WORK

      DO 90 I=2,N
        DO 70 K=I,N
          WKMULT=WORK(K,I-1)
          DO 60 J=1,MDASH
            WORK(K,J)=WORK(K,J)-(WKMULT*WORK((I-1),J))
60          CONTINUE
70        CONTINUE
          WKDIV=WORK(I,I)
          DO 80 J=I,MDASH
            WORK(I,J)=WORK(I,J)/WKDIV
80          CONTINUE
90    CONTINUE

C ... TO GET THE UPPER LHS TO ZEROS

      DO 130 K=N,2,-1
```

```
      DO 120 I=1,K-1
        WKMULT=WORK(I,K)

        DO 110 J=1,MDASH
          WORK(I,J)=WORK(I,J)-(WKMULT*WORK(K,J))
110       CONTINUE
120     CONTINUE
130   CONTINUE

C ... TO EXTRACT INVERSE MATRIX ON RHS AND
C ... MULTIPLY BY ORIGINAL TO SEE ACCURACY OF IDENTITY

      DO 150 I=1,N
        DO 140 J=M+1,MDASH
          AINV(I,J-M)=WORK(I,J)
140     CONTINUE
150   CONTINUE
      RETURN
      END
```

4.3 Program EKFDER.FOR

```
C    ************ P R O G R A M   E K F D E R ***************
C                          Hoff August/94
C                          Rev. A Feb/95
C
C   Estimation of noise covariance, smoothed states and state
C   derivatives using a Kalman-like Filter approach.
C
C   NAT     Number of samples in the data file
C   NP      Number of parameters in the data file
C   LAG     Smoother lag (in sample intervals)
C   SV      Initial estimate of measurem. noise
C   SIGW    Initial estimate of process noise
C   DT      Time interval between samples
C   ZZ(i.k) Matrix of data to be analised or smoother output
C   PM      Time Propagated Covariance Matrix
C   X       Updated State Matrix
C   XM      Time Propagated State Matrix
C   PU      Updated Covarinace Matrix
C   A       Data Model Matrix
C   Q       Data Model Noise Covariance Matrix
C   C       Output Matrix
C   CA      Kalman Filter Dynamic Matrix (to deter. time constant)
C   KK      Gain Matrix
C   SV      Measurement Noise
C   SIGW    Process Noise
C   KH      Auxiliary Matrix
C   FF1,FF2 Cost Functions
C   LAMBDA  Autocorrelation
C   Files  - AILRUD: Aileron and Rudder position.
C          PPLODER1 : Noise statistics and Kalman dynam. matrices,
C          PPLODER2 : Temporary data storage.
C          REGRES: Regression data file - pre-fixed format.
C          EKFDERDA: Contains the data to be smoothed. Initial
C                state values, Covariance matrix, constants
C                and the data composed by: u,w,q,theta,v,
C                phi,X,Z,M (from IEFK program).

      REAL*8 FF1,FF2,LAMBDA(30),DZ(2015)
      REAL*4 X(4,31),XM(4,1),C(4),d(4,4),CA(4),KCA(4,4),XN(4,31)
      REAL*4 QQ(4,4),ZZ(2015,20),PUI(4,4),A(4,4),Q(4,4),P(4,31)
      REAL*4 PU(4,4,31),PM(4,4,31),KK(4,31),KH(4,4),KHT(4,4),MASS,
     +      Iy,MT
      OPEN(UNIT=6,FILE='EKFDERDA.DAT',STATUS='OLD')
      OPEN(UNIT=7,FILE='PPLODER1.DAT')
      OPEN(UNIT=8,FILE='PPLODER2.DAT')
      OPEN(UNIT=9,FILE='ETAT805.DAT',STATUS='OLD')
      OPEN(UNIT=10,FILE='REGRES.DAT')
      OPEN(UNIT=11,FILE='FILUWQTH.DAT')
      OPEN(UNIT=12,FILE='FILUWQD.DAT')
```

```
      OPEN(UNIT=13,FILE='FILXZM.DAT')

C... INITIALIZATION - Initial value of measurement covariance
C               and an initial covariance matrix
      write(*,*) 'READING DATA'
      READ(6,*) SV,SIGW
C... INITIAL COVAR. MATRIX
      DO I=1,4
       READ(6,*) (PUI(I,J),J=1,4) ! COVAR. INITIAL VALUES
      ENDDO
      READ(6,*) NAT,NP,DT,LAG,MASS,Iy
      NAT34=0.75*NAT
      IF(LAG.GT.30) LAG=30
      DO J=1,NAT
       READ(6,*) (ZZ(J,K),K=1,NP)   ! Reading the data
      ENDDO
C...
      A(1,1) = 1.0          !
      A(1,2) = DT           !
      A(1,3) = DT*DT/2.     !
      A(1,4) = DT*DT*DT/6.  !
      A(2,1) = 0            !
      A(2,2) = 1.           !
      A(2,3) = DT           !
      A(2,4) = DT*DT/2.     ! DATA MODEL (3 DERIVATIVES)
      A(3,1) = 0.           !
      A(3,2) = 0.           !
      A(3,3) = 1.0          !
      A(3,4) = DT           !
      A(4,1) = 0.           !
      A(4,2) = 0.           !
      A(4,3) = 0.           !
      A(4,4) = 1.0          !
C...
      C(1)=1.               !
      C(2)=0.               ! OUTPUT MATRIX
      C(3)=0.               !
      C(4)=0.               !
C...
      Q(1,1) = (DT**7)/252.  !
      Q(1,2) = (DT**6)/72.   !
      Q(1,3) = (DT**5)/30.   !
      Q(1,4) = (DT**4)/24.   !
      Q(2,1) = (DT**6)/72.   !
      Q(2,2) = (DT**5)/20.   !
      Q(2,3) = (DT**4)/8.    !
      Q(2,4) = (DT**3)/6.    !
      Q(3,1) = (DT**5)/30.   ! DATA MODEL COVAR. MATRIX
      Q(3,2) = (DT**4)/8.    !
      Q(3,3) = (DT**3)/3.    !
      Q(3,4) = (DT**2)/2.    !
```

```
    Q(4,1) = (DT**4)/24.     !
    Q(4,2) = (DT**3)/6.      !
    Q(4,3) = (DT**2)/2.      !
    Q(4,4) = DT              !
    JK=1
C...
 777 DSIG=SIGW/5.
    IC=0
    SX=-1.
    FF20=1.
    DO WHILE (SIGW.GT.0)     ! LOOP1 DETERMIN. OF SIGW & SIGV
 888 IC=IC+1
    SIGW2=SIGW*SIGW
    write(*,*) 'sigw',sigw
    DO I=1,4
     DO L=1,4
      QQ(I,L)=SIGW2*Q(I,L)
      PU(I,L,1)=PUI(I,L)
     ENDDO
    ENDDO
    X(1,1)=ZZ(1,JK)            !
    X(2,1)=(ZZ(2,JK)-ZZ(1,JK))/DT  ! Initialization of X
    X(3,1)=0.                  !
    X(4,1)=0.                  !
C...
    DO ISAMPLE=1,NAT     !....Kalman Filter Loop............

C... STATE AND COVARIANCE PROPAGATION
    DO I=1,4
     XM(I,1)=0.0
      DO K=1,4
       XM(I,1)=XM(I,1)+A(I,K)*X(K,1)     ! STATE PROPAGATION
      ENDDO
    ENDDO
    DO I=1,4
     DO J=1,4
      P(I,J)=0.0
       DO K=1,4
        P(I,J)=P(I,J)+A(I,K)*PU(K,J,1)   ! AT*P
       ENDDO
     ENDDO
    ENDDO
    DO I=1,4
     DO J=1,4
       PM(I,J,1)=0.0
      DO K=1,4                      ! AT*P*A
       PM(I,J,1)=PM(I,J,1)+P(I,K)*A(J,K)   ! COVARIANCE PROPAGATION
      ENDDO
       PM(I,J,1)=PM(I,J,1)+QQ(I,J)        ! AT*P*A+Q
     ENDDO
    ENDDO
```

```
C
C**  GAIN CALCULATION
C
      DO I=1,4
       P(I,1)=0.0               ! PARTIAL PRODUCT P=P*HT
       DO K=1,4
        P(I,1)=P(I,1) + PM(I,K,1)*C(K)
       ENDDO
      ENDDO
      HPHT=0.0                  ! PARTIAL PRODUCT HPHT=H*P*HT
      DO K=1,4
       HPHT=HPHT+C(K)*P(K,1)
      ENDDO
       HPHT=HPHT+SV*SV              ! HPHT + SV2

C... MATRIX INVERSION
      HPHT1=1.0/HPHT
C... GAIN
      DO I=1,4
       KK(I,1)=P(I,1)*HPHT1         ! GAIN MATRIX
      ENDDO
C
C... STATES UPDATE AFTER MEASUREMENT  -  X(+) Calculation
      DO I=1,4
       X(I,1)=XM(I,1)+KK(I,1)*(ZZ(ISAMPLE,JK)-XM(1,1)) ! STATE UPDATE
      ENDDO

C... COVARIANCE UPDATE  -  P(+) Calculation
      DO I=1,4
       DO J=1,4
         KH(I,J)=-KK(I,1)*C(J)      ! Calculation of K*H
       ENDDO
        KH(I,I)=1.+KH(I,I)          ! Calculation of I-K*H
      ENDDO

      DO I=1,4
       DO J=1,4
        PU(I,J,1)=0.0
        DO K=1,4
         PU(I,J,1)=PU(I,J,1)+KH(I,K)*PM(K,J,1) ! Covar. Update
        ENDDO
       ENDDO
      ENDDO
      IF(ISAMPLE.EQ.NAT34) THEN
       WRITE(7,*) JK
       WRITE(7,*) '3*NAT/4 COVAR. & GAIN',PU(1,1,1),KK(1,1)
      ENDIF
C
      ENDDO      !<<<<<< END OF K-F FILTER LOOP  >>>>>>>>>
C
      WRITE(7,*) ' FINAL  COVAR. & GAIN',PU(1,1,1),KK(1,1)
```

```
      SIGMAR2=0.0
      FF1=0
      FF2=0
      X(1,1)=ZZ(1,JK)
      X(2,1)=(ZZ(2,JK)-ZZ(1,JK))/DT
      X(3,1)=0
      X(4,1)=0
C..
      DO ISAM=1,NAT      !  LOOP - AUTOCORRELATION CALC.
      DO I=1,4
       XM(I,1)=0.0
       DO K=1,4
        XM(I,1)=XM(I,1)+A(I,K)*X(K,1)    ! State Propagation
       ENDDO
      ENDDO
      DZ(ISAM)=ZZ(ISAM,JK)-XM(1,1)      ! Residual using K and P
      DO K=1,4                    ! from steady-state K-F
       X(K,1)=XM(K,1)+KK(K,1)*DZ(ISAM)   ! above - Need to certify s-s
      ENDDO
       SIGMAR2=SIGMAR2+1./(FLOAT(NAT))*DZ(ISAM)**2
      ENDDO
C...............
      IF(LAG.GT.0) THEN
       DO N=1,LAG
       LAMBDA(N)=0.0
       DO ISAM=N+1,NAT
        LAMBDA(N)=LAMBDA(N)+1./(FLOAT(NAT-N))*(DZ(ISAM)*DZ(ISAM-N))
       ENDDO
        FF2=FF2+LAMBDA(N)*LAMBDA(N)
       ENDDO
      ELSE
       FF2=SIGMAR2*SIGMAR2
      ENDIF
      FF1=(PU(1,1,1)+SV*SV-SIGMAR2)**2
C...
      SIGWB=SQRT((SIGMAR2*SIGW*SIGW)/(PU(1,1,1)+SV*SV))
      XLAM=SV/SIGW
      SIGVB=XLAM*SIGWB
      IF(IC.GE.2) THEN          !
       IF(FF2.LT.FF20) THEN        !
        IF(SIGW.LT.SIG0) THEN     !
         SX=-1.              !
        ELSE                 !
         SX=1.               !
        ENDIF                !
       ELSE                  !
        IF(SIGW.LT.SIG0) THEN     !
         DSIG=DSIG/2.          !  Determination of minimum
         SX=1.               !  FF2. Search for direction
        ELSE                 !  and step. Non-efficient !
         DSIG=DSIG/2.          !
```

```
      SX=-1.              !
     ENDIF                !
    ENDIF                 !
   ENDIF                  !
    SIG0=SIGW               !
    SIGW=SIGW+SX*DSIG          !
    IF(SIGW.LE.0.) THEN        !
     SIGW=(SIGW+SIG0)/2.       !
     DSIG=SIGW/2.            !
    ENDIF
    IF(ABS((FF2-FF20)/FF20).LT.0.00001) GO TO 999
    FF20=FF2
    IF(ABS(SIGW-SIG0).LT.1.0E-04) GO TO 999
   ENDDO     !<<<<<<<<< LAM LOOP
 999 CONTINUE
   WRITE(7,112) SV,SIGW,SIGMAR2,FF1,FF2
 112 FORMAT(T05,'SV=',E12.6,2X,',  SW'E12.5,' SIGR2=',E14.7/
  +      T05,'FF1=',E14.7,'  FF2=',E14.7)
   WRITE(7,114) JK,SIGVB,SIGWB
 114 FORMAT(T05,'VARIABLE=',I2,'  SIGVB=',E12.7,'  SIGWB=',E12.7)
   WRITE(*,114) JK,SIGVB,SIGWB
   SIGW=SIGWB
   SV=SIGVB
   IF(IC.LT.9999) THEN
    DSIG=SIGWB/100.
    IC=9999
    GO TO 888
   ENDIF
C...  END OF PROCESS AND MEASUREMENT NOISE DETERMINATION
C
C...  CHARACTERISTIC MATRIX
   do l=1,4
     ca(l)=0
    do k=1,4
     ca(l)=ca(l)+c(k)*a(k,l)
    enddo
   enddo
   do I=1,4
    do j=1,4
     kca(i,j)=0
     d(i,j)=0
     kca(i,j)=kk(i,1)*ca(j)
     d(i,j)=a(i,j)-kca(i,j)  ! system matrix to determine
    enddo              ! dominant eigenvalue.
   enddo
    write(7,*) 'MATRIX d, PARAMETER',JK
   do i=1,4
     write(7,*) (d(i,k),k=1,4)
   enddo
   write(7,*)
C
```

```
C.....SM0OTHER >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
C
      DO L=2,LAG+1
       DO I=1,4
        X(I,L)=0.
        DO J=1,4
         PU(I,J,L)=PU(I,J,1)
        ENDDO
       ENDDO
      ENDDO
C...
      SV2=SV*SV
      SIGW2=SIGW*SIGW
      DO I=1,4
       DO L=1,4
        QQ(I,L)=SIGW2*Q(I,L)
       ENDDO
      ENDDO

C >>>>>>>>>>>>>>>>>>>>>> MAIN LOOP <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
      X(1,1)=ZZ(1,JK)
      X(2,1)=(ZZ(2,JK)-ZZ(1,JK))/DT  ! Initial estim. of derivative
      X(3,1)=0
      X(4,1)=0

      DO ISAMPLE=1,NAT
C...  STATE AND COVARIANCE PROPAGATION
      DO I=1,4
       XM(I,1)=0.0
        DO K=1,4
         XM(I,1)=XM(I,1)+A(I,K)*X(K,1)  ! STATE PROPAGATION XM(K/K-1)
        ENDDO
      ENDDO
C...
      DO L=1,LAG+1
       LL=1
       IF(L.GT.2) LL=L-1
      DO I=1,4
       DO J=1,4
        PM(I,J,L)=0.0
         DO K=1,4
          PM(I,J,L)=PM(I,J,L)+PU(I,K,LL)*A(J,K)
         ENDDO
       ENDDO
      ENDDO
      ENDDO
C...
      DO I=1,4
       DO J=1,4
         DDP=0.0
        DO K=1,4
```

```
      DDP=DDP+A(I,K)*PM(K,J,1)        ! COVAR. PROPAGATION
     ENDDO
      PM(I,J,1)=DDP+QQ(I,J)         ! PM(K/K-1)
    ENDDO
   ENDDO
C
C**  GAIN CALCULATION
C
   DO LL=1,LAG+1
   DO I=1,4
    P(I,LL)=0.0              ! PARTIAL PRODUCT P=P*HT
    DO K=1,4
     P(I,LL)=P(I,LL) + PM(I,K,LL)*C(K)
    ENDDO
   ENDDO
   ENDDO
   HPHT=0.0                  ! PARTIAL PRODUCT HPHT=H*P*HT
   DO K=1,4
    HPHT=HPHT+C(K)*P(K,1)
   ENDDO
   HPHT=HPHT+SV2              ! HPHT + SV2
   HPHT1=1.0/HPHT             ! INVERSION
C...
   DO LL=1,LAG+1
   DO I=1,4
     KK(I,LL)=P(I,LL)*HPHT1      ! GAIN MATRIX - KALMAN
   ENDDO
   ENDDO
C
C... STATES UPDATE AFTER MEASUREMENT  -  X(+) Calculation
C
   DZZ=ZZ(ISAMPLE,JK)-XM(1,1)
   IF(LAG.GT.0) THEN
    DO LL=2,LAG+1
     DO I=1,4
      XN(I,LL)=X(I,LL-1)+KK(I,LL)*DZZ ! STATE UPDATE - SMOOTHER
     ENDDO
    ENDDO
   ENDIF

   DO I=1,4
    X(I,1)=XM(I,1)+KK(I,1)*DZZ       ! STATE UPDATE - K.FILTER
   ENDDO
C...
   DO I=1,4
    DO J=1,4
     KH(I,J)=-KK(I,1)*C(J)
     KHT(I,J)=-C(I)*KK(J,1)
    ENDDO
     KH(I,I)=1. + KH(I,I)
     KHT(I,I)=1. + KHT(I,I)          ! I-K*H
```

```
    ENDDO
 C...
    DO I=1,4
     DO J=1,4
      PU(I,J,1)=0.
       DO K=1,4
        PU(I,J,1)=PU(I,J,1)+KH(I,K)*PM(K,J,1) ! COVAR.UP. P(K/K)
       ENDDO
     ENDDO
    ENDDO

    IF(LAG.GT.0) THEN
    DO LL=2,LAG+1
    DO I=1,4
     DO J=1,4
      PU(I,J,LL)=0.
       DO K=1,4
        PU(I,J,LL)=PU(I,J,LL)+PM(I,K,LL)*KHT(K,J) ! COV.UP.P(K/K)
       ENDDO
      ENDDO
       X(I,LL)=XN(I,LL)          ! STATE REDEFINITION ???
     ENDDO
    ENDDO
    ENDIF
c    WRITE(7,*) (X(K,1),K=1,4)
    IF(ISAMPLE.GE.LAG) WRITE(8,*) (X(K,LAG+1),K=1,2) ! State and 1st
     IF(ISAMPLE.EQ.NAT) THEN                 ! derivative
      DO K=LAG,2,-1                     ! stored (only)
       WRITE(8,*) X(1,K),X(2,K)              !
      ENDDO
     ENDIF
    ENDDO
C>>>>>>>>>>>>>>>>>>>>> MAIN LOOP F-L END <<<<<<<<<<<<<<<<<<<<<<<<<<<<

    JK=jk+1
    IC=0
    SIGW= 1000.   ! Arbitrary number
    IF(JK-1.LT.NP) GO TO 777   ! GO TO NEXT PARAMETER
    REWIND 8
C
C   WRITING THE REGRESSION FILE FOR LATERAL ANALYSIS
C
    DO I=1,NP
    PRINT *,'READING PARAM.',I
     K=2*I
     DO J=1,NAT
      READ(8,*) ZZ(J,K-1),ZZ(J,K)  ! READ STATE AND ITS DERIVATIVE
     ENDDO
    ENDDO
    PRINT *,'WRITING FINAL FILE - REGRES.DAT'
    DO I=1,NAT
```

```
      READ(9,*) ELEV,THRX          ! READ ELEVATOR, THRUST
      u2=ZZ(I,1)**2
      w2=ZZ(I,3)**2
      X=ZZ(I,13)*MASS   ! X Force non normalised
      Z=-ZZ(I,15)*MASS  ! Z Force "     "
      MT=ZZ(I,17)*Iy    ! M Moment "    "
c     write   u,w,q,theta,wdot,udot,u2,w2,eta,qdot,shp,x,z,m
      WRITE(10,*) ZZ(I,1),ZZ(I,3),ZZ(I,5),ZZ(I,7),ZZ(I,4),ZZ(I,2),
     +       u2,w2,ELEV,ZZ(I,6),THRX,X,Z,MT ! Final regression
                                     ! data file.
      WRITE(11,*) ZZ(I,1),ZZ(I,3),ZZ(I,5),ZZ(I,7)
      WRITE(12,*) ZZ(I,2),ZZ(I,4),ZZ(I,6)
      WRITE(13,*) X,Z,MT
      ENDDO
      CLOSE(6)
      CLOSE(7)
      CLOSE(8)
      STOP
      END
```

4.4 Program MSR.FOR

```
C***  PROGRAM MOD. STEPWISE REGRESSION - MSR.FOR  *****
C
C                        HOFF JUN/93
C                        REV. DEC/93,DEC/94
C
C   NAT = NUMBER OF SAMPLES OF EACH VARIABLE
C   NV  = MAXIMUM NUMBER OF INDEPENDENT VARIABLES
C   IVV = ACTUAL NUMBER OF VARIABLES IN THE MODEL
C   NN  = ACTUAL NUMBER OF VARIABLES IN THE DATA ARCHIVE
C   ISTAT(IV) = DEFINE STATUS OF THE VARIABLE IN THE MODEL
C          IF.EQ.-1 IS NEGLETED.
C   ISTATU(I) = VARIABLE NUMBER
C   X(I,J) = INDEPENDENT VARIABLES READ FROM FLIGHT DATA
C   XWORK(I,J) = THE X(s) ACTUALY USED BY THE MODEL
C   Y(I)  = DEPENDENT VARIABLE - FROM FLIGHT DATA
C   YN(I)  = NEW DEPENDENT VARIABLE FOR TESTING NEW X TO ENTER
C         NEXT ITERATION.
C   YHAT  = ESTIMATED Y FROM THE REGRESSION MODEL
c   FT    = PARTIAL Fp TESTE
C   XTRX  = MATRIX PRODUCT OF X TRANSPOSE TIMES X
C   XTRY  = MATRIX PRODUCT OF X TRANSPOSE TIMES MATRIX Y
C   B     = VECTOR OF REGRESSION COEFFICIENTS
C   WORK, IDENT = AUXILIARY MATRICES
C   DY     = REGRESSION RESIDUAL
C   SB    = ESTIMATED STANDARD ERROR
C   VAR    = RESIDUAL VARIANCE
C
   REAL*4 X(2000,12),Y(2000,3),SB(12),Z(12),FP(12),DY(2000),
  +     XWORK(2000,12),XTRY(12),B(12),YN(2000)
   REAL*4 YHAT(2000),YNHAT(2000),BTXTRY
   REAL*8  XTRX[ALLOCATABLE] (:,:),  XTRXI[ALLOCATABLE] (:,:),
  +     IDENT[ALLOCATABLE] (:,:),   WORK[ALLOCATABLE] (:,:)

   REAL*8 SYY,SJY,SJJ,YAVER,MODRJY,RJY,ZAV,RMAX,DYAV,
  +     FMIN,FPMIN,R2,F,VAR,RESS
   INTEGER*2 ISTAT(11),ISTATU(12),I,J,K,L,M,N,IN,NAT,NV,NN,IV,
  +       IVV,ITER,NEWVAR,IT
   CHARACTER*1 ICHAR
   CHARACTER*12 ARQ
   CHARACTER*6 IMOD(12)/
  *'  Y =',' B0 + ','B1*X1+','B2*X2+','B3*X3+',
  * 'B4*X4+','B5*X5+','B6*X6+','B7*X7+','B8*X8+','B9*X9+',
  * 'B10X10'/
   LOGICAL PEND
   FMIN=5.
   IN=1
   IOLD=12
   PEND=.FALSE.
C
```

```
C*** DATA READING ***
C
     PRINT *,'ENTER DATA FILE NAME - USE .DAT'
     READ(*,777) ARQ
 777 FORMAT(A12)
     OPEN(UNIT=6,FILE=ARQ,STATUS='OLD')
     OPEN(UNIT=8,FILE='MSROUT',STATUS='NEW')
     OPEN(UNIT=9,FILE='LSINIT.DAT')
     WRITE(*,*) '* READING DATA FILE *'
     READ(6,*)NAT,NN,NY
     IF(NAT.GT.2000) NAT=2000
     NV=NN+1
     DO I=1,NAT
      X(I,1)=1.
      READ(6,*)(X(I,J+1),J=1,NN),(Y(I,K),K=1,NY)
     ENDDO
     CLOSE(UNIT=6)
C
C*** READING THE MODEL ***
C
     DO I=1,NV
      ISTAT(I)=-1
     ENDDO
 888 PRINT *,'THERE IS/ARE',NY,' DEPENDENT VARIABLES IN THE FILE'
     PRINT *,'CHOOSE ONE TO BE USED - TYPE VARIABLE NUMBER'
     READ *,IY
     PRINT *,'ENTER THE VARIABLES TO BE INCLUDED IN THE MODEL'
     PRINT *,'TYPE 1ST VARIABLE NUMBER, I2,'
     READ *,IV
     ISTAT(IV+1)=0
  10 PRINT *,'ENTER NEXT VAR. NUMBER, I2, - TO STOP ENTER 99'
     READ *,IV
     IF(IV.EQ.12) THEN
      PRINT *,'MAX. No. OF VARIABLES EXCEEDED'
      IV=99
     ENDIF
     IF(IV.NE.99) THEN
      ISTAT(IV+1)=0
      GO TO 10
     ENDIF
     PRINT *,' OK - ALL VARIABLES NOW ENTERED'
C
C*** Y AVERAGE
C
     YAVER=0.0
     DO M=1,NAT
      YAVER=YAVER+Y(M,IY)
     ENDDO
     YAVER=YAVER/FLOAT(NAT)
C
C<<<<<< REGRESSION PROCESS >>>>>>>>>>>>>>>>>>>>>>>>
```

```
C
    NEWVAR=-3
    ITER=0

 999 CONTINUE
    IVV=0
    DO M=1,NV
     IF(ISTAT(M).EQ.0) THEN
      IVV=IVV+1
      ISTATU(IVV)=M-1
     ENDIF
    ENDDO
    ITER=ITER+1

C***  PRINTING THE MODEL

    PRINT *,' '
    WRITE(*,199) ITER
 199 FORMAT('******ITERATION No. ',I2,' ***************************')
    WRITE(8,200)
    WRITE(*,201)
 200 FORMAT(/T05,'*** REGRESSION MODEL:'/)
 201 FORMAT(/T05,'REGRESSION MODEL:')
    WRITE(8,205) IMOD(1),(IMOD(ISTATU(L)+2),L=1,IVV)
    WRITE(*,205) IMOD(1),(IMOD(ISTATU(L)+2),L=1,IVV)
 205 FORMAT(T02,11A6)

    IF(PEND) GO TO 1111
    IF(ITER.GT.2) GO TO 1111
    DO N=1,NAT
     DO L=1,IVV
      XWORK(N,L)=X(N,ISTATU(L)+1)
     ENDDO
    ENDDO
C
    ALLOCATE (XTRX(IVV,IVV))
    DO I=1,IVV
     DO J=1,IVV
      XTRX(I,J)=0
      DO K=1,NAT
       XTRX(I,J)=XTRX(I,J)+XWORK(K,I)*XWORK(K,J)  ! XT*X
      ENDDO
     ENDDO
    ENDDO
C
    ALLOCATE (WORK(IVV,2*IVV))
    ALLOCATE (IDENT(IVV,IVV))
    ALLOCATE (XTRXI(IVV,IVV))
    CALL INVMAT(XTRX,IVV,IVV,XTRXI,WORK,IDENT) ! INVERSE XTRX
C
    DO I=1,IVV
```

```
      XTRY(I)=0
      DO J=1,NAT
       XTRY(I)=XTRY(I)+XWORK(J,I)*Y(J,IY)      ! XT*Y
      ENDDO
     ENDDO
C
     DO I=1,IVV
      B(I)=0
     DO J=1,IVV
      B(I)=B(I)+XTRXI(I,J)*XTRY(J)  ! ESTIMATED COEFFICIENTS
      ENDDO
     ENDDO
C
C***  STATISTICS ***
C
     DO I=1,NAT
      YHAT(I)=0
      DO J=1,IVV
       YHAT(I)=YHAT(I)+XWORK(I,J)*B(J)  ! IS THE Y ESTIMATED
      ENDDO
     ENDDO
C
      BTXTRY=0
     DO I=1,IVV
      BTXTRY=BTXTRY+B(I)*XTRY(I)
     ENDDO
C
     DYAV=0.0
     RESS=0.0
     DO L=1,NAT
      DY(L)=Y(L,IY)-YHAT(L)              ! RESIDUE
      RESS = RESS + DY(L)*DY(L)           ! RESIDUAL SUM SQUARES
      DYAV=DYAV + DY(L)
     ENDDO
      DYAV=DYAV/NAT
      SYY=0.0
     DO L=1,NAT
      SYY=SYY+(DY(L)-DYAV)**2
     ENDDO
C
     VAR=RESS/FLOAT(NAT-IVV)              ! RESIDUAL VARIANCE
C
     DO K=1,IVV
      SB(K)=SQRT(VAR*XTRXI(K,K))          ! ESTIMATED STD ERROR
     ENDDO

     F=(BTXTRY-NAT*(YAVER**2))/(VAR*(IVV-1))  ! F VALUE

     R2=F/((NAT-IVV)/(IVV-1)+F)           ! CORRELATION COEF.
C
C    COVARIANCE MATRIX
```

```
      IF(ITER.EQ.1) THEN
       WRITE(9,*) (B(K),K=1,IVV)
        DO I=1,IVV
         DO J=1,IVV
          XTRX(I,J)=VAR*XTRXI(I,J)
         ENDDO
          WRITE(9,*) (XTRX(I,K),K=1,IVV)     ! FOR FIRST MODEL ONLY
        ENDDO
      ENDIF
C
C *** PRINTING THE SIGNIFICATIVE PARAMETERS
C
      WRITE(8,*) ' '
      DO K=1,IVV
       WRITE(8,210) ISTATU(K),B(K),SB(K)
  210  FORMAT(T05,'VARIABLE  X',I2,'  COEF.Bj = ',F14.5,'  STD ERROR',
     * E12.6)
      ENDDO
      WRITE(8,215) R2,F,RESS,VAR
  215 FORMAT( /T05,'CORRELATION COEF. "R2".... = ',F10.6/
     *        T05,'"F"  COEFFICIENT.......... = ',E12.6/
     *        T05,'RESIDUAL SUM OF SQUARES... = ',E12.6/
     *        T05,'RESIDUAL VARIANCE......... = ',E12.6)
C
C<<<<<<  VARIABLE TO BE REJECTED  >>>>>>>>>>>>>>>>>>>>>>>>
C    THE NULL CASE :

      IF(ITER.EQ.1.AND.F.LT.FMIN) THEN
       PRINT *,' ALL B(J)=0 - REGRESSION ABORTED'
       WRITE(8,220)
  220  FORMAT(/T05,'* REGRESSION ABORTED: F LOWER THAN Fmin *')
       GO TO 1111
      ENDIF
      WRITE(8,222)
  222 FORMAT(/T05,'PARTIAL CORRELATION COEFFICIENTS'/)
C
C***  PARTIAL TEST - FP -- VARIABLE TO BE REJECTED
C
      IT=0
      FPMIN=FMIN
      DO J=1,IVV
       FP(J)=(B(J)**2)/(SB(J)**2)
        IF(FP(J).LT.FPMIN) THEN
         FPMIN=FP(J)
         IT=ISTATU(J)+1         ! THE LAST WILL BE THE REJECTED
        ENDIF
         WRITE(8,224) ISTATU(J),FP(J)
  224    FORMAT(T05,'FP(',I2,')  ........ = ',E10.4)
      ENDDO
      IF(IT.NE.0) THEN
       PRINT *,'VARIABLE',IT-1,' WILL BE REJECTED'
```

```
      PRINT *,'DO YOU WANT TO HOLD VAR.',IT-1,' IN THE REGRESSION'
      CALL SREAD(ICHAR)
       IF(ICHAR.EQ.'S') GO TO 1111
       IF(ICHAR.EQ.'Y') THEN
        PRINT *,'VARIABLE',IT-1,' HELD'
        WRITE(8,225) IT-1
 225    FORMAT(/T05,'VARIABLE ',I2,' HELD IN THE REGRESSION'/)
        IT=0
        GO TO 230
       ENDIF
      IF(IT.EQ.NEWVAR-1) THEN
       PRINT *,' * LAST INTRODUCED VARIABLE WAS REJECTED *'
       WRITE(8,226) IT-1
 226   FORMAT(/T05,'LAST INTRODUCED VARIABLE WAS REJECTED..X',I2/)
       ISTAT(IT)=-2
       GO TO 230
      ELSE
       PRINT *,' ONE VARIABLE REJECTED ',IT-1
       WRITE(8,228) IT-1
 228   FORMAT(/T05,'ONE VARIABLE REJECTED....X',I2/)
       PRINT *,' REPROCESS WITHOUT THE REJECTED VARIABLE ? '
       CALL SREAD(ICHAR)
       IF(ICHAR.EQ.'S') GO TO 1111
       ISTAT(IT)=-2            ! RESET STATUS VARIABLE TO REJECT
        IF(ICHAR.EQ.'Y') THEN
         DEALLOCATE(XTRX,XTRXI,WORK,IDENT)
         WRITE(8,*) '   NEW REGRESSION WITHOUT REJEC. VARIABLE'
         WRITE(8,*) ''
         GO TO 999
        ENDIF
      ENDIF
     ELSE
      PRINT *,' NO VARIABLE REJECTED'
      WRITE(8,229)
 229  FORMAT(/T05,'NO VARIABLE REJECTED')
      IF(IVV.EQ.NV) THEN
       WRITE(*,*) ' NO MORE VARIABLES TO BE INCLUDED'
       GO TO 1111
      ENDIF
     ENDIF
C
C<<<<< IDENTIFICATION NEW VARIABLE TO INCLUDE IN THE MODEL >>>>>>
C
 230 WRITE(8,231)
 231 FORMAT(//T05,'ANALYSIS OF NEW VARIABLES '/)
    NEWVAR=-3
    RMAX=0.0
C
    DO L=1,NV
     IF(ISTAT(L).EQ.0.OR.ISTAT(L).EQ.-2) GO TO 1000
     DO J=1,NAT
```

```
     YN(J)=X(J,L)     ! NEW INDEPENDENT VARIABLE
    ENDDO

C   REGRESSION
    DO J=1,IVV
     XTRY(J)=0
     DO I=1,NAT
      XTRY(J)=XTRY(J)+XWORK(I,J)*YN(I)     ! XT*Y (NEW Y)
     ENDDO
    ENDDO
    DO I=1,IVV
     B(I)=0
     DO J=1,IVV
      B(I)=B(I)+XTRXI(I,J)*XTRY(J)        ! REGRES. COEFF.
     ENDDO
    ENDDO
    DO I=1,NAT
     YNHAT(I)=0
     DO J=1,IVV
      YNHAT(I)=YNHAT(I)+XWORK(I,J)*B(J)     ! NEW Y ESTIMATE
     ENDDO
    ENDDO

    ZAV=0.0
    DO I=1,NAT
     Z(I)=YN(I)-YNHAT(I)                 ! RESIDUE
     ZAV=Z(I)+ZAV                       ! AVERAGE RESIDUE
    ENDDO
    ZAV=ZAV/FLOAT(NAT)
    SJY=0.0
    SJJ=0.0
    DO I=1,NAT
     SJY=SJY+(Z(I)-ZAV)*((Y(I,IY)-YHAT(I))-DYAV)
     SJJ=SJJ+(Z(I)-ZAV)**2
    ENDDO
    RJY=0.0
    IF((SYY*SJJ).NE.0.0) RJY=SJY/SQRT(SYY*SJJ)
    MODRJY=DABS(RJY)
    WRITE(8,232) L-1,MODRJY
232   FORMAT(T05,'VARIABLE   X',I2,'.... RJY=',E12.6)
    IF(MODRJY.GT.RMAX) THEN        ! NEW VARIABLE SELECTION
     RMAX=MODRJY
     NEWVAR=L-1                 ! IDENTIFY THE CHOSEN VARIABLE
    ENDIF
 1000 CONTINUE
    ENDDO  ! END OF NEW VARIABLE CHOICE - RETURN TO THE LOOP

    IF(NEWVAR.EQ.-3) THEN
     WRITE(8,235)
 235 FORMAT(/T05,'NO MORE VARIABLES TO BE INCLUDED - PROGRAM END'//
   +        T05,'FINAL MODEL:')
```

```
      PEND=.TRUE.
      GO TO 999
     ENDIF

     DEALLOCATE(XTRX,XTRXI,WORK,IDENT)

     IF(ISTAT(IT).EQ.-2)ISTAT(IT)=-1   ! RESET STAT. REJEC. VAR
     ISTAT(NEWVAR+1)=0                 ! RESET STATUS NEW VARIABLE
     WRITE(8,240) NEWVAR
     WRITE(*,240) NEWVAR
 240 FORMAT(/T05,'THE NEW BEST VARIABLE IS: X',I2/)
     DO J=1,NV
      IF(ISTAT(J).EQ.-2) ISTAT(J)=-1
     ENDDO
     GO TO 999            ! TRY A NEW REGRESSION
1111 CONTINUE
     PRINT *,' ** TRY A NEW INDEPENDENT VARIABLE ? - Y/N **'
     CALL SREAD(ICHAR)
     IF(ICHAR.EQ.'Y') GO TO 888
     CLOSE(UNIT=8)
     STOP
     END
C
C>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
C
     SUBROUTINE INVMAT(A,IAR,IAC,AINV,WORK,IDENT [REFERENCE])
C
C    Matrix inversion - max xx*xx matrix
c
     INTEGER*2 IAR,IAC
     REAL*8 A(IAR,IAC),WORK(IAR,2*IAC),AINV(IAR,IAC),IDENT(IAR,IAC)
     REAL*8 WKDIV,WKMULT
C
C ... N = NUMBER OF ROWS    (I)
C ... M = NUMBER OF COLUMNS (J)
C ... N = M OR CANNOT INVERT THE MATRIX A
C
     N = IAR
     M = IAC
C
C    TO CREATE THE APPROPRIATE IDENTITY MATRIX In=IDENT(N,M)

     DO 20 I=1,N
       DO 10 J=1,M
          IDENT(I,J)=0.0
  10   CONTINUE
          IDENT(I,I)=1.0
  20 CONTINUE

C ... TO ADJOIN THE A AND IDENT MATRICES
```

```
      MDASH=2*M
      DO 40 I=1,N
       DO 30 J=1,M
         WORK(I,J)=A(I,J)
         WORK(I,M+J)=IDENT(I,J)
30      CONTINUE
40    CONTINUE

C ... TO MAKE WORK(1,1)=1.0

      WKDIV=WORK(1,1)

      DO 50 J=1,MDASH
       WORK(1,J)=WORK(1,J)/WKDIV
50    CONTINUE

C ...  TO MAKE ZEROS BELOW DIAGONAL OF LHS OF MATRIX WORK

      DO 90 I=2,N
       DO 70 K=I,N
        WKMULT=WORK(K,I-1)
        DO 60 J=1,MDASH
         WORK(K,J)=WORK(K,J)-(WKMULT*WORK((I-1),J))
60       CONTINUE
70      CONTINUE
        WKDIV=WORK(I,I)
        DO 80 J=I,MDASH
         WORK(I,J)=WORK(I,J)/WKDIV
80       CONTINUE
90    CONTINUE

C ... TO GET THE UPPER LHS TO ZEROS

      DO 130 K=N,2,-1

       DO 120 I=1,K-1
        WKMULT=WORK(I,K)

        DO 110 J=1,MDASH
         WORK(I,J)=WORK(I,J)-(WKMULT*WORK(K,J))
110      CONTINUE
120     CONTINUE
130   CONTINUE

C ... TO EXTRACT INVERSE MATRIX ON RHS AND
C ... MULTIPLY BY ORIGINAL TO SEE ACCURACY OF IDENTITY

      DO 150 I=1,N
       DO 140 J=M+1,MDASH
        AINV(I,J-M)=WORK(I,J)
140     CONTINUE
```

```
150  CONTINUE
     RETURN
     END
C>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
     SUBROUTINE SREAD(CHAR)
     CHARACTER*1 CHAR
     CHAR=' '
     DO WHILE ((CHAR.NE.'N').AND.(CHAR.NE.'Y').AND.(CHAR.NE.'S'))
       WRITE(*,'(A)') ' ENTER Y OR N, OR S TO STOP:'
       READ(*,'(A)') CHAR
     ENDDO
     RETURN
     END
```

4.5 Program MSRH.FOR

```
C*** PROGRAM MOD. STEPWISE REGRESSION *******************
C
C   NEW VERSION WITH HOUSEHOLDER TRANSFORMATION
C
C                  HOFF, AUGUST/1993, Rev. Fev.95
C
C   NAT = NUMBER OF SAMPLES OF EACH VARIABLE
C   NV  = MAXIMUM NUMBER OF INDEPENDENT VARIABLES
C   IVV = ACTUAL NUMBER OF VARIABLES IN THE MODEL
C   NN  = ACTUAL NUMBER OF VARIABLES IN THE DATA FILE (MAX=11)
C   NY  = NUMBER OF DEPENDENT VARIABLES IN THE DATA FILE
C       NY MAX. EQUAL TO 3.
C   ISTAT(IV) = DEFINE STATUS OF THE VARIABLE IN THE MODEL
C           IF.EQ.-1 IS NEGLETED.
C   ISTATU(I) = VARIABLE NUMBER
C   X(I,J) = INDEPENDENT VARIABLES READ FROM FLIGHT DATA
C   XWORK(I,J) = THE X(s) ACTUALY USED BY THE MODEL AND AUGMENTED
C            MATRIX.
C   Y(I,k)  = DEPENDENT VARIABLE - FROM FLIGHT DATA
C   YHAT   = ESTIMATED Y VALUE (BY THE REGRESSION MODEL).
C   B    = REGRESSION COEFFICIENTS
C   DY   =  REGRESSION RESIDUAL
C   SB   =  ESTIMATED STANDARD ERROR
C   VAR  =  RESIDUAL VARIANCE
C   U    = AUXILIARY MATRIX

   REAL*4 X(2000,12),Y(2000,3),Z(12),SB(12),FP(12),DY(2000),
  +     XWORK(2000,15),YHAT(2000),XHAT(2000),V(2000),
  +     U(12,12),COV(12,12),B(12),XTRY(12)

   REAL*8 SYY,SJY,SJJ,YAVER,MODRJY,RJY,ZAV,RMAX,DYAV,
  +     FMIN,FPMIN,R2,F,VAR,RESS
   INTEGER*2 ISTAT(12),ISTATU(12),I,J,K,L,M,N,NAT,NV,NN,IV,
  +       IVV,ITER,NEWVAR,IT
   CHARACTER*1 ICHAR
   CHARACTER*12 ARQ
   CHARACTER*7 IMOD(13)/
  *'  Y =',' B0 + ',' B1*X1+',' B2*X2+',' B3*X3+',
  * ' B4*X4+',' B5*X5+',' B6*X6+',' B7*X7+',' B8*X8+',' B9*X9+',
  * 'B10X10+',' B11X11'/
   LOGICAL PEND
   FMIN=5.
   PEND=.FALSE.
C
C*** DATA READING ***
C
   PRINT *,'ENTER DATA FILE NAME'
   READ(*,777) ARQ
```

```
 777 FORMAT(A12)
     OPEN(UNIT=6,FILE=ARQ,STATUS='OLD')
     OPEN(UNIT=8,FILE='MSROUT',STATUS='NEW')
     WRITE(8,777) ARQ
     WRITE(*,*) '* READING DATA FILE *'
     READ(6,*)NAT,NN,NY
     IF(NAT.GT.2000) NAT=2000
     NV=NN+1
     DO I=1,NAT
      X(I,1)=1.
      READ(6,*) (X(I,J+1),J=1,NN),(Y(I,K),K=1,NY)
     ENDDO
     CLOSE(UNIT=6)
C
C***  READING THE MODEL ***
C
     DO I=1,NV
      ISTAT(I)=-1
     ENDDO
 888  PRINT *,'THERE IS/ARE',NY,' DEPENDENT VARIABLES IN THE FILE'
     PRINT *,'CHOOSE ONE TO BE USED - TYPE VARIABLE NUMBER'
     READ *,IY
     PRINT *,'ENTER THE VARIABLES TO BE INCLUDED IN THE MODEL'
     PRINT *,'TYPE 1ST VARIABLE NUMBER, I2,'
     READ *,IV
     ISTAT(IV+1)=0
  10 PRINT *,'ENTER NEXT VAR. NUMBER, I2, - TO STOP ENTER 99'
     READ *,IV
     IF(IV.EQ.12) THEN
      PRINT *,'MAX. No. OF VARIABLES EXCEEDED'
      IV=99
     ENDIF
     IF(IV.NE.99) THEN
      ISTAT(IV+1)=0
      GO TO 10
     ENDIF
     PRINT *,' OK - ALL VARIABLES NOW ENTERED'
C
C***  Y AVERAGE
C
     YAVER=0.0
     DO M=1,NAT
      YAVER=YAVER+Y(M,IY)
     ENDDO
     YAVER=YAVER/FLOAT(NAT)
C
C<<<<<< REGRESSION PROCESS >>>>>>>>>>>>>>>>>>>>>>>
C
     NEWVAR=-3
     ITER=0
```

```
 999 CONTINUE
    IVV=0
    DO M=1,NV
     IF(ISTAT(M).EQ.0) THEN
      IVV=IVV+1
      ISTATU(IVV)=M-1
     ENDIF
    ENDDO
     MM=IVV
    DO M=1,NV
     IF(ISTAT(M).NE.0) THEN
      MM=MM+1
      ISTATU(MM)=M-1
     ENDIF
    ENDDO
    ITER=ITER+1

C***  PRINTING THE MODEL

    PRINT *,' '
    WRITE(*,199) ITER
 199 FORMAT('******ITERATION No. ',I2,' ************************')
    WRITE(8,200) IY
    WRITE(*,201) IY
 200 FORMAT(/T05,'REGRESSION MODEL FOR INDEPENDENT VARIABLE :',I2,//)
 201 FORMAT(/T05,'REGRESSION MODEL FOR INDEPENDENT VARIABLE :',I2,)
    WRITE(8,205) IMOD(1),(IMOD(ISTATU(L)+2),L=1,IVV)
    WRITE(*,205) IMOD(1),(IMOD(ISTATU(L)+2),L=1,IVV)
 205 FORMAT(T02,11A7)

    IF(PEND) GO TO 1111
    IF(ITER.GT.50) GO TO 1111

    DO N=1,NAT
     DO L=1,IVV
      XWORK(N,L)=X(N,ISTATU(L)+1)      ! X OF WORK
     ENDDO
      XWORK(N,IVV+1)=Y(N,IY)          ! AUGMENT XWORK WITH Y
      DO K=IVV+1,NV
       XWORK(N,K+1)=X(N,ISTATU(K)+1)    ! AUGMENT WITH REMAINING X
      ENDDO
    ENDDO
C
    DO J=1,IVV
     XTRY(J)=0.0
      DO I=1,NAT
       XTRY(J)=XTRY(J)+XWORK(I,J)*Y(I,IY)    ! XT*Y
      ENDDO
    ENDDO
C
    ZZ=0.
```

```
      DO 40 J=1,IVV+1
      SIG=ZZ
      DO 11 I=J,NAT
      V(I)=XWORK(I,J)
      XWORK(I,J)=ZZ
   11 SIG=SIG+V(I)**2
      IF(SIG.LE.ZZ) GO TO 40
      SIG=SQRT(SIG)
      IF(V(J).GT.ZZ) SIG=-SIG
      XWORK(J,J)=SIG
      V(J)=V(J)-SIG
      SIG=1./(SIG*V(J))
      DO 30 K=J+1,NV+1
      ALF=ZZ
      DO 20 I=J,NAT
   20 ALF=ALF+XWORK(I,K)*V(I)
      ALF=ALF*SIG
      DO 30 I=J,NAT
   30 XWORK(I,K)=XWORK(I,K)+ALF*V(I)
   40 CONTINUE
C     REMOVE THE TRANSFORMED Y (Z) FROM THE A MATRIX
       DO L=1,IVV
        DO I=1,IVV
        U(L,I)=ZZ
        ENDDO
       ENDDO
C     ---- INVERSION OF 'A' PRODUCING 'U'-------
       U(1,1)=1./XWORK(1,1)
      DO 60 L=2,IVV
       U(L,L)=1./XWORK(L,L)
       JM1=L-1
      DO 60 K=1,JM1
       SUM=0.0
      DO 50 I=K,JM1
   50  SUM=SUM-U(K,I)*XWORK(I,L)
   60  U(K,L)=SUM*U(L,L)
C     ----- SOLUTION OF  X=A**-1 * Z  OR B=U*Z
      DO I=1,IVV
       B(I)=0.0
       COV(I,I)=0.0
       DO L=1,IVV
        B(I)=B(I)+U(I,L)*XWORK(L,IVV+1)  ! REGRESSION COEFF.
        COV(I,I)=COV(I,I)+U(I,L)*U(I,L)  ! COV. MATRIX MAIN DIAG.
       ENDDO
      ENDDO
C
C***  STATISTICS ***
C
      DO L=1,IVV
       DO N=1,NAT
        XWORK(N,L)=X(N,ISTATU(L)+1)          ! XWORK REDEFINITION
```

```
     ENDDO
    ENDDO
C
    DO I=1,NAT
     YHAT(I)=0
     DO J=1,IVV
      YHAT(I)=YHAT(I)+XWORK(I,J)*B(J)     ! Y ESTIMATED
     ENDDO
    ENDDO
C
    BTXTRY=0
    DO J=1,IVV
     BTXTRY=BTXTRY+B(J)*XTRY(J)
    ENDDO
C
    DYAV=0.0
    RESS=0.0
    DO L=1,NAT
     DY(L)=Y(L,IY)-YHAT(L)              ! RESIDUE
     RESS = RESS + DY(L)*DY(L)            ! RESIDUAL SUM SQUARES
     DYAV=DYAV + DY(L)
    ENDDO
     DYAV=DYAV/NAT
     SYY=0.0
    DO L=1,NAT
     SYY=SYY+(DY(L)-DYAV)**2
    ENDDO
C
    VAR=RESS/(NAT-IVV)                  ! RESIDUAL VARIANCE

C
    DO K=1,IVV
     SB(K)=SQRT(VAR*COV(K,K))            ! ESTIMATED STD ERROR
    ENDDO

    F=(BTXTRY-NAT*(YAVER**2))/(VAR*(IVV-1))  ! F VALUE

    R2=F/((NAT-IVV)/(IVV-1)+F)           ! CORRELATION COEF.

C
C *** PRINTING THE SIGNIFICANT PARAMETERS/STATISTICS
C
    WRITE(8,*) ''
    WRITE(8,*) ''
    DO K=1,IVV
     WRITE(8,210) ISTATU(K),B(K),SB(K)
 210  FORMAT(T05,'VARIABLE  X',I2,'  COEF.Bj = ',F14.5,'  STD ERROR',
   * E12.6)
    ENDDO
    WRITE(8,215) R2,F,RESS,VAR
 215 FORMAT(//T05,'CORRELATION COEF. "R2".... = ',F10.6/
```

```
  *       T05,'"F"  COEFFICIENT.......... = ',E12.6/
  *       T05,'RESIDUAL SUM OF SQUARES... = ',E12.6/
  *       T05,'RESIDUAL VARIANCE......... = ',E12.6//)

C
C<<<<<< VARIABLE TO BE REJECTED >>>>>>>>>>>>>>>>>>>>>>>>>
C   THE NULL CASE :

      IF(ITER.EQ.1.AND.F.LT.FMIN) THEN
       PRINT *,' ALL B(J)=0 - REGRESSION ABORTED'
       WRITE(8,220)
 220  FORMAT(/T05,'* REGRESSION ABORTED: F LOWER THAN Fmin *')
       GO TO 1111
      ENDIF
      WRITE(8,222)
 222 FORMAT(/T05,'PARTIAL CORRELATION COEFFICIENTS'/)
C
C***  PARTIAL TEST - FP -- VARIABLE TO BE REJECTED
C
      IT=0
      FPMIN=FMIN
      DO J=1,IVV
        FP(J)=(B(J)**2)/(SB(J)**2)
        IF(FP(J).LT.FPMIN) THEN
         FPMIN=FP(J)
         IT=ISTATU(J)+1          ! THE LAST WILL BE THE REJECTED
        ENDIF
         WRITE(8,224) ISTATU(J),FP(J)
 224    FORMAT(T05,'FP(',I2,')  ........ = ',E10.4)
       ENDDO
       IF(IT.NE.0) THEN
        PRINT *,'VARIABLE',IT-1,' WILL BE REJECTED'
        PRINT *,'DO YOU WANT TO HOLD VAR.',IT-1,' IN THE REGRESSION'
        CALL SREAD(ICHAR)
         IF(ICHAR.EQ.'S') GO TO 1111
         IF(ICHAR.EQ.'Y') THEN
          PRINT *,'VARIABLE',IT-1,' HELD'
          WRITE(8,225) IT-1
 225      FORMAT(/T05,'VARIABLE ',I2,' HELD IN THE REGRESSION'/)
          IT=0
          GO TO 230
         ENDIF
        IF(IT.EQ.NEWVAR-1) THEN
         PRINT *,' * LAST INTRODUCED VARIABLE WAS REJECTED *'
         WRITE(8,226) IT-1
 226     FORMAT(/T05,'LAST INTRODUCED VARIABLE WAS REJECTED..X',I2/)
         ISTAT(IT)=-2
         GO TO 230
        ELSE
         PRINT *,' ONE VARIABLE REJECTED ',IT-1
         WRITE(8,228) IT-1
```

```
 228   FORMAT(//T05,'ONE VARIABLE REJECTED....X',I2/)
       PRINT *,' REPROCESS WITHOUT THE REJECTED VARIABLE ? '
       CALL SREAD(ICHAR)
        IF(ICHAR.EQ.'S') GO TO 1111
         ISTAT(IT)=-2        ! RESET STATUS VARIABLE TO REJECT
        IF(ICHAR.EQ.'Y') THEN
         WRITE(8,*) '   NEW REGRESSION WITHOUT REJEC. VARIABLE'
         WRITE(8,*) ' '
         GO TO 999
        ENDIF
      ENDIF
     ELSE
      PRINT *,' NO VARIABLE REJECTED'
      WRITE(8,229)
 229  FORMAT(/T05,'NO VARIABLE REJECTED')
     ENDIF
C
C<<<<< IDENTIFICATION NEW VARIABLE TO BE INCLUDED TO THE MODEL >>>
C
 230  WRITE(8,231)
 231  FORMAT(//T05,'ANALYSIS OF NEW VARIABLES '/)
    NEWVAR=-3
    RMAX=0.0
C
C   REGRESSION FOR VARIABLES NO INCLUDED PRESENT MODEL
C   SOLUTION OF  X=A**-1 * Z  OR B=U*Z
    DO L=IVV+2,NV+1
    DO I=1,IVV
     B(I)=0.0
     DO J=1,IVV
      B(I)=B(I)+U(I,J)*XWORK(J,L)       !   REGRES. COEFF.
     ENDDO
    ENDDO

    DO I=1,NAT
     XHAT(I)=0
      DO J=1,IVV
       XHAT(I)=XHAT(I)+B(J)*XWORK(I,J)   ! NEW Y ESTIMATES
      ENDDO
    ENDDO

    ZAV=0.0
    DO I=1,NAT
     Z(I)=X(I,ISTATU(L-1)+1)-XHAT(I)     ! RESIDUE
     ZAV=Z(I)+ZAV                        ! AVERAGE RESIDUE
    ENDDO
    ZAV=ZAV/FLOAT(NAT)
    SJY=0.0
    SJJ=0.0
    DO I=1,NAT
     SJY=SJY+(Z(I)-ZAV)*((Y(I,IY)-YHAT(I))-DYAV)
```

```
      SJJ=SJJ+(Z(I)-ZAV)**2
      ENDDO
      RJY=0.0
      IF((SYY*SJJ).NE.0.0) RJY=SJY/SQRT(SYY*SJJ)
      MODRJY=DABS(RJY)
      WRITE(8,232) ISTATU(L-1),MODRJY
 232  FORMAT(T05,'VARIABLE  X',I2,'.... RJY=',E12.6)
      IF(MODRJY.GT.RMAX) THEN
       RMAX=MODRJY
       NEWVAR=ISTATU(L-1)        ! IDENTIFY THE CHOSEN VARIABLE
      ENDIF
 1000 CONTINUE
      ENDDO  ! END OF NEW VARIABLE CHOICE - RETURN TO THE LOOP

      IF(NEWVAR.EQ.-3) THEN
       WRITE(8,235)
 235  FORMAT(/T05,'NO MORE VARIABLES TO BE INCLUDED - PROGRAM END'//
     +       T05,'FINAL MODEL:')
       PEND=.TRUE.
       GO TO 999
      ENDIF

      IF(ISTAT(IT).EQ.-2)ISTAT(IT)=-1  ! RESET STAT. REJEC. VAR
      ISTAT(NEWVAR+1)=0               ! RESET STATUS NEW VARIABLE
      WRITE(8,240) NEWVAR
      WRITE(*,241) NEWVAR
 240  FORMAT(/T05,'THE NEW BEST VARIABLE IS: X',I2/)
 241  format(/T05,'THE NEW BEST VARIABLE IS:',I2)
      DO J=1,NV
       IF(ISTAT(J).EQ.-2) ISTAT(J)=-1
      ENDDO
      GO TO 999                  ! TRY A NEW REGRESSION
 1111 CONTINUE
      PRINT *,'TRY A NEW INDEPENDENT VARIABLE ?'
      CALL SREAD(ICHAR)
      IF(ICHAR.EQ.'Y') GO TO 888
      CLOSE(UNIT=8)
      STOP
      END
C>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
      SUBROUTINE SREAD(CHAR)
      CHARACTER*1 CHAR
      CHAR=' '
      DO WHILE ((CHAR.NE.'N').AND.(CHAR.NE.'Y').AND.(CHAR.NE.'S'))
       WRITE(*,'(A)') ' ENTER Y OR N, OR S TO STOP:'
       READ(*,'(A)') CHAR
      ENDDO
      RETURN
      END
```

REFERENCES

1. Draper N.R. and Smith H. *Applied Regression Analysis*, John Wiley & Sons Inc., 1966.

2. Hoff, J.C. *Aircraft Parameter Estimation by Estimation-Before-Modelling Technique.* PhD Thesis. College of Aeronautics, Cranfield University, September 1995.

3. Bierman, G.J. *Fixed Interval Smoothing With Discrete Measurements.* International Journal of Control, Vol. 18, No. 1, 1973, pp. 65-75.

4. Fioretti, S. and Jetto, L. *Low a Priori Statistical Information Model for Optimal Smoothing and Differentiation of Noise Signals.* International Journal of Adaptative Control and Signal Processing, Vol. 8, pp. 305-320, John Wiley & Sons 1994.