Application Software for a Model Based Approach to Tool Wear Estimation

by

Kourosh Danai

and

A. Galip Ulsoy

Technical Report No. UM-MEAM-86-36

Department of Mechanical Engineering

and Applied Mechanics

University of Michigan

Ann Arbor, MI 48109-2125

Chapter 1

INTRODUCTION

The purpose of this report is to present the application software developed for a model based approach to tool wear estimation. This approach which uses an adaptive observer for tool wear estimation is based on a dynamic state model of tool wear. The application software for this approach is developed to operate on a *Digital Equipment Corporation (DEC) LSI-11/23 plus* microcomputer. Although, most of the computer code is in FORTRAN IV and can be used on other computers, certain parts of the code are exclusively written for this type of computer. The developed software can be categorized into three:

1. The program to evaluate the model, and compute its parameters for the purpose of tuning the adaptive observer (see Appendix A).

2. The program to implement the adaptive observer. This program consists of both high and low level software. The high level software is in FORTRAN IV and the low level software is in ASSEMBLY language (see Appendix B).

3. The program to plot and print the estimated results (see Appendix C).

The report describing the methodology, is organized in two parts:

1. Part 1 (Chapter 2) describes the strategy used for evaluating the model and computing the necessary data.

2. Part 2 (Chapter 3) describes the adaptive observer.

Chapter 2

MODEL EVALUATION

The design of the adaptive observer is based on a dynamic state model of tool wear. Therefore, the suitability of the model for the purpose of on-line tool wear estimation should be determined. We do this by studying the following points:

1. Simulation results using the nonlinear model.

2. Stability of the system as predicted by the model.

3. Controllability of the model as it relates to parameter estimation.

4. Observability of the wear components by cutting force measurement.

Once the suitability of the model for tool wear estimation has been verified, the adaptive observer needs to be designed. The design of the adaptive observer requires knowledge about the canonical parameters, and the observed states. Also, for the purpose of the reconstruction of the wear components from the observed states a transformation matrix **T** has to be approximated in terms of the cutting conditions, the estimated parameters, and time. This approximation is based on the true value of the transforma-

tion matrix obtained from the linearization of the nonlinear model at different operating points. The methodology used to compute the required data is discussed here.

2.1 Simulation

The simulation of the nonlinear model was performed by a simulation package developed by Leal Lauderbaugh [1]. This package uses a Runge-Kutta 4 algorithm for integration.

2.2 Linearization

In order to (i) study the stability, controllability, and observability conditions of the model; (ii) compute the canonical parameters, and states in observer form for the purpose of tuning the adaptive observer; and (iii) compute the transformation matrix as the basis for the regression analysis, the model has to be linearized. The linearization method used is the subject of this section.

The nonlinear model

$$\dot{\mathbf{x}} = \mathbf{f}'(\mathbf{x}, u, t) \tag{2.1}$$

$$y = g'(\mathbf{x}, u, t) \tag{2.2}$$

can be linearized about an operating point ($\mathbf{x}$ and u), such that

$$\dot{\mathbf{x}} = \mathbf{A}\,\mathbf{x} + \mathbf{b}\,u \tag{2.3}$$

$$y = \mathbf{c}^T\,\mathbf{x} + D\,u \tag{2.4}$$

where $\mathbf{x}$ represents the vector of state variables and u is the input variable defining the operating point. In order to obtain such a model $\mathbf{x}$ and u are divided into n' intervals

such that,

$$\triangle \mathbf{x} = \frac{\mathbf{x}}{n'} \tag{2.5}$$

$$\triangle u = \frac{u}{n'} \tag{2.6}$$

Then, the Taylor series expansion is used to linearize the nonlinear functions $\mathbf{f}'(\mathbf{x}, u, t)$ and $g'(\mathbf{x}, u, t)$ in each interval, such that

$$\mathbf{A} = \frac{1}{n'}\left[\frac{\partial \mathbf{f}'}{\partial \mathbf{x}}\bigg|_{\substack{\mathbf{x}=0\\u=0}} + \frac{\partial \mathbf{f}'}{\partial \mathbf{x}}\bigg|_{\substack{\mathbf{x}=\triangle\mathbf{x}\\u=\triangle u}} + \ldots\ldots + \frac{\partial \mathbf{f}'}{\partial \mathbf{x}}\bigg|_{\substack{\mathbf{x}=(n'-1)\triangle\mathbf{x}\\u=(n'-1)\triangle u}}\right] \tag{2.7}$$

$$\mathbf{b} = \frac{1}{n'}\left[\frac{\partial \mathbf{f}'}{\partial u}\bigg|_{\substack{\mathbf{x}=0\\u=0}} + \frac{\partial \mathbf{f}'}{\partial u}\bigg|_{\substack{\mathbf{x}=\triangle\mathbf{x}\\u=\triangle u}} + \ldots\ldots + \frac{\partial \mathbf{f}'}{\partial u}\bigg|_{\substack{\mathbf{x}=(n'-1)\triangle\mathbf{x}\\u=(n'-1)\triangle u}}\right] \tag{2.8}$$

$$\mathbf{c} = \frac{1}{n'}\left[\frac{\partial \mathbf{g}'}{\partial \mathbf{x}}\bigg|_{\substack{\mathbf{x}=0\\u=0}} + \frac{\partial \mathbf{g}'}{\partial \mathbf{x}}\bigg|_{\substack{\mathbf{x}=\triangle\mathbf{x}\\u=\triangle u}} + \ldots\ldots + \frac{\partial \mathbf{g}'}{\partial \mathbf{x}}\bigg|_{\substack{\mathbf{x}=(n'-1)\triangle\mathbf{x}\\u=(n'-1)\triangle u}}\right] \tag{2.9}$$

$$D = \frac{1}{n'}\left[\frac{\partial \mathbf{g}'}{\partial u}\bigg|_{\substack{\mathbf{x}=0\\u=0}} + \frac{\partial \mathbf{g}'}{\partial u}\bigg|_{\substack{\mathbf{x}=\triangle\mathbf{x}\\u=\triangle u}} + \ldots\ldots + \frac{\partial \mathbf{g}'}{\partial u}\bigg|_{\substack{\mathbf{x}=(n'-1)\triangle\mathbf{x}\\u=(n'-1)\triangle u}}\right] \tag{2.10}$$

where n' (the number of intervals) in Eqns. (2.5) – (2.6) is selected based on the convergence criterion of the method. For this specific nonlinear model, n' was selected to be 200. The partial derivatives in Eqns. (2.7) – (2.10) are given in [2]. Once the linear model is obtained, it can be used to study its stability condition by eigenanalysis.

2.3 Eigenvalues

Eigenvalues of a linear model can be obtained from the roots of the model characteristic equation. The characteristic equation can be obtained by using the relationship

$$det[s\mathbf{I} - \mathbf{A}] = 0 \tag{2.11}$$

If for our third order linear model (Eqns. (2.3) and (2.4)) the coefficient matrix **A** is defined as

$$\mathbf{A} = \begin{bmatrix} \alpha_{11} & \alpha_{12} & \alpha_{13} \\ \alpha_{21} & \alpha_{22} & \alpha_{23} \\ \alpha_{31} & \alpha_{32} & \alpha_{33} \end{bmatrix} \tag{2.12}$$

Then the characteristic equation of the model will have the form

$$s^3 + e_1 s^2 + e_2 s + e_3 = 0 \tag{2.13}$$

where

$$e_1 = -(\alpha_{11} + \alpha_{22} + \alpha_{33}) \tag{2.14}$$

$$e_2 = \alpha_{11}\alpha_{22} + \alpha_{11}\alpha_{33} + \alpha_{22}\alpha_{33} - \alpha_{12}\alpha_{21} - \alpha_{13}\alpha_{31} - \alpha_{23}\alpha_{32} \tag{2.15}$$

and

$$e_3 = -\alpha_{11}\alpha_{22}\alpha_{33} + \alpha_{11}\alpha_{23}\alpha_{32} + \alpha_{22}\alpha_{13}\alpha_{31} + \alpha_{33}\alpha_{12}\alpha_{21} - \alpha_{12}\alpha_{23}\alpha_{31} - \alpha_{13}\alpha_{21}\alpha_{32}. \tag{2.16}$$

The above characteristic equation (Eq. (2.13)) can then be solved to obtain the eigenvalues of the model.

2.4 Controllability and Observability

According to Takahashi [3] the linear model defined by equations (2.3) and (2.4) is controllable when,

$$Det\ \mathbf{\Phi} = Det\ [\mathbf{b}, \mathbf{A}\,\mathbf{b}, \mathbf{A}^2\mathbf{b}, ..., \mathbf{A}^{n-1}\mathbf{b}] \neq 0 \tag{2.17}$$

and is observable when

$$Det\ \mathbf{\Theta} = Det\ [\mathbf{c}, \mathbf{A}^T\mathbf{c}, (\mathbf{A}^T)^2\mathbf{c}, ..., (\mathbf{A}^T)^{n-1}\mathbf{c}] \neq 0 \tag{2.18}$$

The determinant of the controllability matrix, Φ, and the observability matrix, Θ, are calculated.

2.5 Discrete-Time Form

The continuous-time linear model defined in Eqns. (2.3) and (2.4) can be transformed into discrete-time form such that

$$\mathbf{x}(k+1) = \mathbf{P}\,\mathbf{x}(k) + \mathbf{q}\,u(k) \tag{2.19}$$

$$y(k) = \mathbf{c}^T\,\mathbf{x}(k) + D\,u(k) \tag{2.20}$$

where

$$\mathbf{P} = \exp(\mathbf{A}\,\triangle t) \tag{2.21}$$

$$\mathbf{q} = (\mathbf{P} - \mathbf{I})\,\mathbf{A}^{-1}\,\mathbf{b} \tag{2.22}$$

and $\triangle t$ is the sampling time.

For computational purposes, $\mathbf{P}$ and $\mathbf{q}$ in the above equations can be obtained by the power series expansion of the term $\exp(\mathbf{A}\,\triangle t)$ [3], such that

$$\mathbf{P} = [\mathbf{I} + (\mathbf{A}\,\triangle t) + \frac{1}{2!}(\mathbf{A}\,\triangle t)^2 + \ldots. + \frac{1}{m'!}(\mathbf{A}\,\triangle t)^{m'}] \tag{2.23}$$

and

$$\mathbf{q} = \triangle t\,[\mathbf{I} + \frac{1}{2!}(\mathbf{A}\,\triangle t) + \frac{1}{3!}(\mathbf{A}\,\triangle t)^2 + \ldots. + \frac{1}{(m'+1)!}(\mathbf{A}\,\triangle t)^{m'}]\,\mathbf{b} \tag{2.24}$$

where m' in the above equations determines the degree of accuracy of the method. Equations (2.21) and (2.22) are used to discretize the continuous-time model. For that, $m' = 100$ is used.

2.6 Canonical Parameters

In order to design the adaptive observer, the parameters of the model in observer canonical form should be computed. For that, the model should first be transformed into observer form. The observer form of the model is,

$$\mathbf{x}_0(k+1) = \tilde{\mathbf{P}}\,\mathbf{x}_0(k) + \tilde{\mathbf{q}}\,u(k) \tag{2.25}$$

$$y(k) = \tilde{\mathbf{c}}^{\mathrm{T}}\,\mathbf{x}_0(k) + D\,u(k) \tag{2.26}$$

where $\tilde{\mathbf{P}}$ and $\tilde{\mathbf{q}}$ in the above equations are defined as

$$\tilde{\mathbf{P}} = \begin{bmatrix} -a_1 & 1 & 0 \\ -a_2 & 0 & 1 \\ -a_3 & 0 & 0 \end{bmatrix}, \quad \tilde{\mathbf{q}} = \begin{Bmatrix} b_1 \\ b_2 \\ b_3 \end{Bmatrix} \tag{2.27}$$

The parameters of the model in canonical form can be obtained from the transfer function of the discrete-time model defined by Eqns. (2.17) and (2.18)). Such a transfer function will have the form

$$\frac{Y(z)}{U(z)} = \frac{b_0 + \sum_{j=1}^{m} b_j \, z^{-j}}{1 + \sum_{i=1}^{n} a_i \, z^{-i}} + D \tag{2.28}$$

where the parameters a_i and b_i are the same as in Eq. (2.25). The above transfer function can be obtained from the relationship

$$G(z) = \frac{Y(z)}{U(z)} = \mathbf{c}^T \left[z\mathbf{I} - \mathbf{P} \right]^{-1} \mathbf{q} + D. \tag{2.29}$$

Now, if $\mathbf{P}$, $\mathbf{q}$, and $\mathbf{c}^T$ are defined as

$$\mathbf{P} = \begin{bmatrix} p_{11} & p_{12} & p_{13} \\ p_{21} & p_{22} & p_{23} \\ p_{31} & p_{32} & p_{33} \end{bmatrix}, \quad \mathbf{q} = \begin{Bmatrix} q_1 \\ q_2 \\ q_3 \end{Bmatrix}, \text{ and } \mathbf{c}^T = \begin{bmatrix} c_1 & c_2 & c_3 \end{bmatrix} \tag{2.30}$$

then the a_i and b_i will have the form

$$a_1 = -(p_{11} + p_{22} + p_{33}) \tag{2.31}$$

$$a_2 = p_{11}p_{22} + p_{11}p_{33} + p_{22}p_{33} - p_{12}p_{21} - p_{13}p_{31} - p_{23}p_{32} \tag{2.32}$$

$$a_3 = -p_{11}p_{22}p_{33} + p_{11}p_{23}p_{32} + p_{22}p_{13}p_{31} + p_{33}p_{12}p_{21} - p_{12}p_{23}p_{31} - p_{13}p_{21}p_{32} \tag{2.33}$$

and

$$b_1 = c_1 q_1 + c_2 q_2 + c_3 q_3 \tag{2.34}$$

$$\begin{aligned} b_2 = {} & q_1 [- c_1 (p_{22} + p_{33}) + c_2 p_{21} + c_3 p_{31}] + \\ & q_2 [c_1 p_{12} - c_2 (p_{11} + p_{33}) + c_3 p_{32}] + \\ & q_3 [c_1 p_{13} + c_2 p_{23} - c_3 (p_{11} + p_{22})] \end{aligned} \tag{2.35}$$

$$\begin{aligned} b_3 = {} & q_1 [c_1 (p_{22} p_{33} - p_{23} p_{32}) + c_2 (p_{31} p_{23} - p_{21} p_{33}) + c_3 (p_{21} p_{32} - p_{31} p_{22})] + \\ & q_2 [c_1 (p_{32} p_{13} - p_{12} p_{33}) + c_2 (p_{11} p_{33} - p_{31} p_{13}) + c_3 (p_{31} p_{12} - p_{11} p_{32})] + \\ & q_3 [c_1 (p_{12} p_{23} - p_{13} p_{22}) + c_2 (p_{21} p_{13} - p_{23} p_{11}) + c_3 (p_{11} p_{22} - p_{21} p_{12})] \end{aligned} \tag{2.36}$$

The above equations (Eqns. (2.29) – (2.34)) are used to compute the parameters a_i and b_i for the purpose of tuning the parameter estimator.

2.7 The Transformation Matrix

The transformation matrix **T** in the relationship

$$\mathbf{x} = \mathbf{T}\,\mathbf{x}_0 \tag{2.37}$$

is required for two purposes:

1. To compute $\mathbf{x}_0$ from $\mathbf{x}$, so that the accuracy of the estimated states by the state observer can be studied. This study is important for the purpose of tuning the state observer.

2. To develop the data base used as the basis for the regression analysis to approximate the transformation matrix in terms of the cutting conditions, the estimated parameters, and time.

The transformation matrix $\mathbf{T}$ can be obtained through the observability matrices of the models in the original form and the observer form, such that

$$\mathbf{T} = \mathbf{\Theta}^{-1}\,\tilde{\mathbf{\Theta}} \tag{2.38}$$

where

$$\mathbf{\Theta} = [\,\mathbf{c}, \mathbf{P}^T\mathbf{c}, (\mathbf{P}^T)^2\mathbf{c}, ..., (\mathbf{P}^T)^{n-1}\mathbf{c}\,] \tag{2.39}$$

$$\tilde{\mathbf{\Theta}} = [\,\tilde{\mathbf{c}}, \tilde{\mathbf{P}}^T\tilde{\mathbf{c}}, (\tilde{\mathbf{P}}^T)^2\tilde{\mathbf{c}}, ..., (\tilde{\mathbf{P}}^T)^{n-1}\tilde{\mathbf{c}}\,] \tag{2.40}$$

In the above relationships, $\mathbf{P}$ and $\mathbf{c}$ are as in Eqns. (2.17) and (2.18) and $\tilde{\mathbf{P}}$ and $\tilde{\mathbf{c}}$ are as in Eqns. (2.23) and (2.24). Equation (2.36) is used to compute the transformation matrix $\mathbf{T}$. Once the transformation matrix is determined, using Eq. (2.35) the states of the model in observer form are determined.

Chapter 3

ADAPTIVE OBSERVER

The adaptive observer used here consists of the combined application of parameter estimation and state estimation. In this adaptive observer it is assumed that (i) the system parameters vary slowly as compared with the states, and (ii) the initial observation error due to poor initial parameter convergence is tolerable. This adaptive observer has the form [4],

$$\hat{\mathbf{x}}_0(k+1) = \begin{bmatrix} -\hat{a}_1 & 1 & 0 & . & 0 \\ -\hat{a}_2 & 0 & 1 & . & 0 \\ . & . & . & . & . \\ . & . & . & . & 1 \\ -\hat{a}_n & 0 & . & . & 0 \end{bmatrix} \hat{\mathbf{x}}_0(k) + \begin{Bmatrix} \hat{b}_0 \\ \hat{b}_1 \\ . \\ . \\ \hat{b}_m \end{Bmatrix} u(k) + \begin{Bmatrix} g_1 \\ g_2 \\ . \\ . \\ g_n \end{Bmatrix} [\, y(k) - \hat{y}(k) \,] \quad (3.1)$$

$$\hat{y}(k) = [\ 1 \quad 0 \quad . \quad . \quad 0\ ]\hat{\mathbf{x}}_0(k) + \hat{D}u(k) \quad (3.2)$$

where the $\hat{a}_i$ and $\hat{b}_i$ are estimated on-line by the parameter estimator.

A recursive least squares parameter estimation algorithm has the general form [4],

$$\hat{\theta}(k) = \hat{\theta}(k-1) + \frac{\mathbf{P}(k-2)\phi(k-1)}{\beta + \phi(k-1)^T\mathbf{P}(k-2)\phi(k-1)}\bar{\nu}(k) \tag{3.3}$$

$$\mathbf{P}(k-1) = \frac{1}{\beta}\left[\mathbf{P}(k-2) - \frac{\mathbf{P}(k-2)\phi(k-1)\phi(k-1)^T\mathbf{P}(k-2)}{\beta + \phi(k-1)^T\mathbf{P}(k-2)\phi(k-1)}\right] \tag{3.4}$$

where $y(k)$ is the value of the measured variable y at time $t = k\triangle t$ for $k = 0, 1, 2, \ldots\ldots$. $\mathbf{P}(k)$ is the matrix of estimation gains, β provides exponential data weighting, and $\bar{\nu}(k)$ is the parameter estimation error . $\phi(k)$ is the vector of measured (or known) variables, and $\hat{\theta}(k)$ is a vector of parameter estimates. The above algorithm recursively updates the estimated parameter vector $\hat{\theta}(k)$ defined as

$$\hat{\theta}(k) = [\ \hat{a}_1(k)\quad \hat{a}_2(k)\quad \ldots\ldots\quad \hat{a}_n(k)\quad \hat{b}_0(k)\quad \hat{b}_1(k)\quad \ldots\ldots\quad \hat{b}_m(k)\ ] \tag{3.5}$$

for any process whose equations can be written in the form,

$$y(k) = \phi(k-1)^T\,\theta(k) \tag{3.6}$$

Thus, the process model must be written in a form that is linear in the unknown parameters, which are the elements of the vector $\theta(k)$. The above algorithm can be used for different estimation methods (*equation error* method and *output error* method), which differ in defining the vector $\phi(k)$ and parameter estimation error $\bar{\nu}(k)$. These methods are discussed in the next sections.

3..1 Equation Error Method

In the *equation error* method, the vector $\phi(k)$ and the estimation error $\bar{\nu}(k)$ are defined as

$$\phi(k-1)^T = [\ -y(k-1)\quad -y(k-2)\quad .....\quad -y(k-n)\quad u(k-1)$$

$$u(k-2)\quad .....\quad u(k-m)\ ] \tag{3.7}$$

and

$$\bar{\nu}(k) = [y(k) - \phi(k-1)^T \hat{\theta}(k-1)] \tag{3.8}$$

3..2 Output Error Method

In the parameter estimation algorithm based on the *output error* method the vector $\phi(k)$ and the estimation error $\bar{\nu}(k)$ are defined as

$$\phi(k-1)^T = [\ -\bar{y}(k-1)\quad -\bar{y}(k-2)\quad .....\quad -\bar{y}(k-n)\quad u(k-1)$$

$$u(k-2)\quad .....\quad u(k-m)\ ] \tag{3.9}$$

where

$$\bar{y}(k) = \phi(k-1)^T\, \hat{\theta}(k) \tag{3.10}$$

$$\hat{y}(k) = \phi(k-1)^T\, \hat{\theta}(k-1) \tag{3.11}$$

and the estimation error $\bar{\nu}(k)$ is defined as

$$\bar{\nu}(k) = y(k) - \hat{y}(k) + [D(q^{-1}) - 1]\,[y(k) - \bar{y}(k)]. \tag{3.12}$$

where $D(q^{-1})$ in the above equation is a fixed moving average filter defined as

$$D(q^{-1}) = 1 + d_1 q^{-1} + ... + d_l q^{-l} \tag{3.13}$$

Appendix A

This appendix presents the computer package developed for evaluating the nonlinear model for tool wear estimation, and computing the necessary data for tuning the adaptive observer. The components of this computer package are:

DISCRT: Main program which calls the appropriate subroutines.

DATA: Subroutine to collect the necessary information about the operating point for linearization, n' (see Eqns. (2.5) and (2.6)), and m' (see Eqns. (2.21) and (2.22)).

CALC: Subroutine to compute the linear model through linearization of the nonlinear model.

ROOTS: Subroutine to compute the roots of a polynomial such as shown in Eq. (2.13).

CNTRL: Subroutine to compute the controllability matrix.

OBSERV: Subroutine to compute the observability matrix.

DSCCON: Subroutine to discretize the continuous-time model.

CANON: Subroutine to compute the parameters of the model in canonical form.

TFORM: Subroutine to compute the transformation matrix.

SUBPRO: A package of subroutine for matrix and vector manipulation.

```
C This program
C (1) Asks for the operating points at which the nonlinear
C     model is to be linearized (SUBROUTINE DATA).
C (2) Linearizes the model about the selected operating point
C     (SUBROUTINE CALC).
C (3) Calculates the eigenvalues (SUBROUTINE ROOTS), and
C     determinants of the controllability (SUBROUTINE CNTRL)
C     and observability matrices (SUBROUTINE OBSERV) of the
C     linearized model.
C (4) Transforms the model into discrete time form (SUBROUTINE DSCCON).
C (5) Calculates the eigenvalues of the discrete-time form (SUBROUTINE ROOTS).
C (6) Calculates the observer canonical form of the model (SUBROUTINE CANON).
C (7) Calculates the transformation matrix for the transformation
C     (SUBROUTINE TFORM).
C
C
C
       IMPLICIT REAL*8 (A-H,O-Z)
C
       REAL*8 L0
C
       REAL*4 Y,S1,S2,S3,S4,S5,S6
C
       DIMENSION Q(14),P(7),X(3),A(3,3),B(3),C(3),PD(3,3),AC(3),
     $ QD(3,3),QDB(3),AA(3,3),ACAN(3),BCAN(3),PHAT(3,3),QHAT(3),
     $ DOBSV(3,3),C1(3),PDC(3),PDT(3,3),CNOBSV(3,3),T(3,3),
     $ TT(3,3),GG(3,3),ROOTR(3),ROOTI(3),CTRL(3,3),OBSV(3,3),
     $ AT(3,3),B1(3),AB(3),ALPHA(3),BETA(3)
C
       COMMON/INOUT/IN,IOUT
       COMMON/COEF/Q,P,L0,V,F1,D,RAKE,TEMP1,TEMP2,FORCE
C
       DATA Y/'Y'/
       DATA IN/5/,IOUT/7/
C
C      CALL ASSIGN(5,'TT:')
C       CALL ASSIGN(7,'TT:')
C       CALL ASSIGN(6,'LP:')
C
C
C Now collect the necessary data
C
C
 100   CALL DATA(X,N,DELT,MACK)
C
C
C Compute the matrices A,B,C,D
C
C
       CALL CALC(N,A,B,C,DD,X,MACK)
C
C Now compute the eigenvalues of the continuous-time system
C
       CALL ROOTS(A,N,ROOTR,ROOTI)
C
C Check the controllability of the system
C
       WRITE(IOUT,201)
 201   FORMAT(1X,'Would you like to check the controllability',
     $  ' and observability of the model Y/N?!')
       READ(IN,300) S1
```

```
 300    FORMAT(A2)
        IF(S1.NE.Y) GOTO 101
C
        CALL CNTRL(N,A,B,B1,AB,CTRL)
        CALL DETER(N,CTRL,DET)
C
C Now check the observability of the system
C
        CALL OBSERV(N,A,C,OBSV,C1,AC,AT)
        CALL DETER(N,OBSV,DET)
C
 101    WRITE(IOUT,230)
 230    FORMAT(1X,'Would you like to compute the transfer',
     $  ' function in the s-plane, Y/N?!')
          READ(IN,300) S6
        IF (S6.NE.Y) GOTO 102
C
         CALL CANON(N,A,B,C,ALPHA,BETA,PHAT,QHAT)

C
C Compute the matrices P and Q for the discrete time system
C
 102     WRITE(IOUT,202)
 202     FORMAT(2X,'Would you like to transform the model',
     $     ' into discrete-time,  Y/N?!')
         READ(IN,300) S2
         IF(S2.NE.Y) GOTO 900
C
        CALL DSCCON(A,B,N,PD,QD,GG,AA,QDB,DELT)
C
C COMPUTE THE ROOTS OF THE DISCRT-TIME SYSTEM
C
        CALL ROOTS(PD,N,ROOTR,ROOTI)
C
C Now we can compute the a's and b's of the canonical form
C
        WRITE(IOUT,203)
 203    FORMAT(2X,'Would you like to transform the model',
     $ ' into canonical form,  Y/N?!')
        READ(IN,300) S3
        IF(S3.NE.Y) GOTO 900
C
        CALL CANON(N,PD,QDB,C,ACAN,BCAN,PHAT,QHAT)
C
C
C Now we can compute the transformation matrix
C
        WRITE(IOUT,204)
 204    FORMAT(2X,'Would you like to compute the transformation',
     $ ' matrix,  Y/N?!')
        READ(IN,300) S4
        IF(S4.NE.Y) GOTO 900
C
C
        CALL TFORM(N,PD,C,DOBSV,C1,PDC,PDT,PHAT,CNOBSV,T,TT,MT,NT,X,XBAR)
C
 900    WRITE(IOUT,330)
 330    FORMAT(//,2X,'Would you like to try another set of input',
     $ 2X,' Y/N?!',3X)
        READ(IN,300) S5
        IF(S5.EQ.Y) GOTO 100
```

```
C
      STOP
      END
```

```
      SUBROUTINE DATA(X,N,DELT,MACK)
C
C This subprogram collects the input data
C
      IMPLICIT REAL*8 (A-H,O-Z)
      REAL*8 L0
      REAL*4 Y,S1
C
      DIMENSION Q(14),P(7),X(3)
C
      COMMON/INOUT/IN,IOUT
      COMMON/COEF/Q,P,L0,V,F1,D,RAKE,TEMP1,TEMP2,FORCE
C
      DATA Y/'Y'/
C
      WRITE(IOUT,100)
 100  FORMAT(5X,'Enter the cutting speed, in m/min,',5X)
      READ (IN,101) V
 101  FORMAT(F10.5)
C
      WRITE(IOUT,102)
 102  FORMAT(5X,'Enter the feed rate, in mm/rev,',5X)
      READ(IN,101) F1
C
      WRITE(IOUT,103)
 103  FORMAT(5X,'Enter the depth of cut, in mm,',5X)
      READ(IN,101) D
C
C      WRITE(IOUT,106)
C 106  FORMAT(5X,'Enter the rake angle, in degrees',5X)
C      READ(IN,101) RAKE
C      RAKE=RAKE*(2.0D0*3.141593)/360.0D0
      RAKE=10.0D0*(2.0D0*3.141593)/360.0D0
C
      N=3
C
      WRITE(IOUT,107)
 107  FORMAT(5X,'Enter the width of flank wear ',
     $ ' due to abrasion, in mm',5X)
      READ(IN,101) X(1)
C
      WRITE(IOUT,109)
 109  FORMAT(5X,'Enter the width of flank wear',
     $ ' due to diffusion, in mm,',5X)
      READ(IN,101) X(2)
C
      WRITE(IOUT,110)
 110  FORMAT(5X,'Enter the depth of crater wear,',
     $ 'in mm,')
      READ(IN,101) X(3)
C
      WRITE(IOUT,117)
 117  FORMAT(5X,'Enter the number of steps for linearization')
      READ(IN,118) MACK
 118  FORMAT(I3)
C
 116  WRITE(IOUT,111)
 111  FORMAT(5X,'Enter the time increment for simulation,',
     $ 'in min.',3X)
      READ(IN,112) DELT
 112  FORMAT(F8.6)
```

```
C
      WRITE (6,113) V,F1,D,X(1),X(2),X(3),MACK,DELT
 113  FORMAT(10X,'v = ',G15.8,/,10X,'f = ',G15.8,/,
     $ 9X,' d = ',G15.8,/,7X,' wf1 = ',G15.8,/,
     $ 7X,' wf2 = ',G15.8,/,8X,' wc = ',G15.8,/,
     $ 6X,' MACK = ',I3,/,6X,' DELT = ',G15.8,//)
C
      RETURN
      END
```

```
SUBROUTINE CALC(N,A,B,C,DD,X,MACK)
      C
      C
      C This subprogram computes the coefficient  matrices of the linear model
      C
      C
            IMPLICIT REAL*8 (A-H,O-Z)
            REAL*8 L0
      C
            DIMENSION A(N,N),B(N),C(N),Q(14),P(7),X(N)
      C
            COMMON/INOUT/IN,IOUT
            COMMON/COEF/Q,P,L0,V,F1,D,RAKE,TEMP1,TEMP2,FORCE
      C
      C
      C
      C
      C
            DO 2 L=1,N
            DO 2 M=1,N
            A(L,M)=0.0D0
            B(L)=0.0D0
       2      CONTINUE
      C
            G1=V/L0
            G2=Q(1)*DCOS(RAKE)/F1
      C
            A(1,1)=G1*(-1.0D0+G2*Q(13))
            A(1,2)=G1*G2*Q(13)
            A(1,3)=-G1*G2*Q(14)/D
            B(1)=G1*G2*(Q(9)*F1**(P(7)-1.0D0)*(1.D0-Q(10)*RAKE)-
           $      Q(11)/F1-Q(12)*V/F1)
      C
            RACK=DBLE(FLOAT(MACK))+1.0D0
            DO 20 I=0,MACK
            BACK=DBLE(FLOAT(I))
            VAR1=BACK*X(1)/RACK
            VAR2=BACK*X(2)/RACK
            VAR3=BACK*X(3)/RACK
            VAR4=BACK*F1/RACK
      C
      C
            IF(VAR4.GT.0.02D0) GOTO 201
            FORCE=0.0D0
            TEMP1=0.0D0
            TEMP2=0.0D0
            GOTO 202
      201   FORCE=(Q(9)*VAR4**P(7)*(1.D0-Q(10)*RAKE)-Q(11)-Q(12)*V)*D+
           $        Q(13)*D*(VAR1+VAR2)-Q(14)*VAR3
      C
            TEMP1=Q(6)*V**P(1)*VAR4**P(2)+Q(7)*(VAR1+VAR2)**P(3)
      C
              TEMP2=Q(8)*FORCE*V**P(4)*VAR4**P(5)*D**P(6)
      C
       202  G3=273.0D0+TEMP1
            G4=273.0D0+TEMP2
            G5=DEXP(-Q(3)/G3)
            G6=Q(2)*DSQRT(V)
            G7=Q(4)*V
            G8=DEXP(-Q(5)/G4)
            IF(VAR4.GT.0.001D0) GOTO 203
```

```
       G9=0.0D0
       G10=0.0D0
       G11=0.0D0
       GOTO 204
 203   G9=Q(8)*V**P(4)*VAR4**P(5)*D**P(6)
        G10=Q(9)*P(7)*VAR4**(P(7)-1.D0)*(1.0D0-Q(10)*RAKE)*D
       G11=Q(8)*FORCE*V**P(4)*P(5)*VAR4**(P(5)-1.D0)*D**P(6)
C
 204   A(2,1)=A(2,1)+(G6*Q(3)*Q(7)*P(3)*(VAR1+VAR2)**(P(3)-1.0D0)*G5/
     $           (G3*G3))/RACK
       A(2,2)=A(2,1)
       A(2,3)=0.0D0
       IF(VAR4.GT.0.02D0) GOTO 205
       B(2)=0.0D0
       GOTO 206
 205   B(2)=B(2)+(G6*G5*Q(3)*Q(6)*V**P(1)*P(2)*VAR4**(P(2)-1.0D0)/
     $       (G3*G3))/RACK
C
 206   A(3,1)=A(3,1)+(G7*G8*(Q(13)*D+(FORCE*Q(5)*Q(13)*D*G9)/
     $         (G4*G4)))/RACK
       A(3,2)=A(3,1)
       A(3,3)=A(3,3)+(G7*G8*(-Q(14)-(FORCE*Q(5)*Q(14)*G9)/
     $         (G4*G4)))/RACK
       B(3)=B(3)+(G7*G8*(G10+(FORCE*Q(5)*(G9*G10+G11))/(G4*G4)))/RACK
C
C       CFLANK=(A(2,1)*VAR1+A(2,2)*VAR2+A(2,3)*VAR3+B(2)*VAR4)/
C    $          (BACK+1.0D0)
C       CCRATE=(A(3,1)*VAR1+A(3,2)*VAR2+A(3,3)*VAR3+B(3)*VAR4)/
C    $            (BACK+1.0D0)
C
 20       CONTINUE
C
       AFLANK=WF1(X(1),X(2),X(3),F1)
       DFLANK=WF2(X(1),X(2),X(3),F1)
       DCRATE=WC(X(1),X(2),X(3),F1)
       CAFLAN=(A(1,1)*X(1)+A(1,2)*X(2)+A(1,3)*X(3)+B(1)*F1)
       CFLANK=(A(2,1)*X(1)+A(2,2)*X(2)+A(2,3)*X(3)+B(2)*F1)
       CCRATE=(A(3,1)*X(1)+A(3,2)*X(2)+A(3,3)*X(3)+B(3)*F1)
       WRITE(6,305) AFLANK,CAFLAN,DFLANK,CFLANK,DCRATE,CCRATE
 305   FORMAT(' ACTUAL ABRASIVE FLANK RATE= ',G16.8,10X,
     $         'COMPUTED ABRASIVE FLANK RATE= ',G16.8,/,
     $         ' ACTUAL DIFFUSIVE FLANK RATE=',G16.8,10X,
     $         'COMPUTED DIFFUSIVE FLANK RATE=',G16.8,/,
     $         ' ACTUAL CRATER RATE=',G16.8,19X,
     $         'COMPUTED CRATER RATE=',G16.8)
C
C Record the components of the C matrix
C
       C(1)=Q(13)*D
       C(2)=Q(13)*D
       C(3)=-Q(14)
C
       DD=(Q(9)*F1**(P(7)-1.0D0)*(1.D0-Q(10)*RAKE)-Q(11)/F1-Q(12)*V/
     $     F1)*D
C
       WRITE(6,160)
 160   FORMAT(///,28X,'THE A MATRIX',//)
       DO 15 I=1,N
 15    WRITE(6,161) (A(I,J),J=1,N)
 161   FORMAT(3(5X,G16.8))
C
```

```
      WRITE(6,162)
 162  FORMAT(///,28X,'THE B VECTOR',//)
      DO 16 I=1,N
 16   WRITE(6,163) B(I)
 163  FORMAT(27X,G16.8)
C
      WRITE(6,164)
 164  FORMAT(///,28X,'THE C VECTOR',//)
      WRITE(6,165) (C(I),I=1,3)
 165  FORMAT(3(5X,G16.8))
C
C
      WRITE(6,168)
 168  FORMAT(///,25X,'THE D COMPONENT',//)
      WRITE(6,169) DD
 169  FORMAT(25X,G16.8)
C
      RETURN
      END
```

```
      SUBROUTINE ROOTS(A,N,ROOTR,ROOTI)
C
C  PROGRAM FOR REAL AND COMPLEX ROOTS OF A REAL POLYNOMIAL
C
      IMPLICIT REAL*8 (A-H,O-Z)
C
      DIMENSION A(N,N),CON(4),W(4),ROOTR(3),ROOTI(3)
C
      COMMON/INOUT/IN,IOUT
C
C
C
      CON(1)=-A(1,1)*A(2,2)*A(3,3)+A(1,1)*A(2,3)*A(3,2)+
     $          A(2,2)*A(1,3)*A(3,1)+A(3,3)*A(1,2)*A(2,1)-
     $          A(1,2)*A(2,3)*A(3,1)-A(1,3)*A(2,1)*A(3,2)
C
      CON(2)= A(1,1)*A(2,2)+A(1,1)*A(3,3)+A(2,2)*A(3,3)-
     $          A(1,2)*A(2,1)-A(1,3)*A(3,1)-A(2,3)*A(3,2)
C
      CON(3)=-(A(1,1)+A(2,2)+A(3,3))
C
       CON(4)=1.0D0
C
      CALL POLRT(CON,W,3,ROOTR,ROOTI,IER)
C
      IF(IER-1) 90,60,70
C
 60      WRITE(6,65)
 65      FORMAT(////34H ORDER OF POLYNOMIAL LESS THAN ONE)
      GO TO 100
C
 70      IF(IER-3) 75,80,78
C
 75      WRITE(6,77)
 77      FORMAT(////36H ORDER OF POLYNOMIAL GREATER THAN 36)
      GOTO 100
C
 78      WRITE(6,79)
 79      FORMAT(////31H HIGH ORDER COEFFICIENT IS ZERO)
      GOTO 100
C
 80      WRITE(6,85)
 85      FORMAT(////50H UNABLE TO DETERMINE ROOT. THOSE ALREADY FOUND ARE)
      GOTO 100
C
 90      WRITE(6,95)
 95      FORMAT(////5X,9HREAL ROOT,6X,12HCOMPLEX ROOT//)
C
       DO 96 I=1,3
 96      WRITE(6,97)ROOTR(I),ROOTI(I)
 97      FORMAT(2E16.7)
C
 100   RETURN
        END
C
C     ..................................................................
C
C       SUBROUTINE POLRT
C
C       PURPOSE
C          COMPUTES THE REAL AND COMPLEX ROOTS OF A REAL POLYNOMIAL
```

```
C
C       USAGE
C          CALL POLRT(XCOF,COF,M,ROOTR,ROOTI,IER)
C
C       DESCRIPTION OF PARAMETERS
C          XCOF -VECTOR OF M+1 COEFFICIENTS OF THE POLYNOMIAL
C                ORDERED FROM SMALLEST TO LARGEST POWER
C          COF  -WORKING VECTOR OF LENGTH M+1
C          M    -ORDER OF POLYNOMIAL
C          ROOTR-RESULTANT VECTOR OF LENGTH M CONTAINING REAL ROOTS
C                OF THE POLYNOMIAL
C          ROOTI-RESULTANT VECTOR OF LENGTH M CONTAINING THE
C                CORRESPONDING IMAGINARY ROOTS OF THE POLYNOMIAL
C          IER  -ERROR CODE WHERE
C                IER=0  NO ERROR
C                IER=1  M LESS THAN ONE
C                IER=2  M GREATER THAN 36
C                IER=3  UNABLE TO DETERMINE ROOT WITH 500 INTERATIONS
C                       ON 5 STARTING VALUES
C                IER=4  HIGH ORDER COEFFICIENT IS ZERO
C
C       REMARKS
C          LIMITED TO 36TH ORDER POLYNOMIAL OR LESS.
C          FLOATING POINT OVERFLOW MAY OCCUR FOR HIGH ORDER
C          POLYNOMIALS BUT WILL NOT AFFECT THE ACCURACY OF THE RESULTS.
C
C       SUBROUTINES AND FUNCTION SUBPROGRAMS REQUIRED
C          NONE
C
C       METHOD
C          NEWTON-RAPHSON ITERATIVE TECHNIQUE.  THE FINAL ITERATIONS
C          ON EACH ROOT ARE PERFORMED USING THE ORIGINAL POLYNOMIAL
C          RATHER THAN THE REDUCED POLYNOMIAL TO AVOID ACCUMULATED
C          ERRORS IN THE REDUCED POLYNOMIAL.
C
C     ..................................................................
C
      SUBROUTINE POLRT(XCOF,COF,M,ROOTR,ROOTI,IER)
      DIMENSION XCOF(1),COF(1),ROOTR(1),ROOTI(1)
      DOUBLE PRECISION XO,YO,X,Y,XPR,YPR,UX,UY,V,YT,XT,U,XT2,YT2,SUMSQ,
     1 DX,DY,TEMP,ALPHA,DABS
C
C        ..................................................................
C
C        IF A DOUBLE PRECISION VERSION OF THIS ROUTINE IS DESIRED, THE
C        C IN COLUMN 1 SHOULD BE REMOVED FROM THE DOUBLE PRECISION
C        STATEMENT WHICH FOLLOWS.
C
         DOUBLE PRECISION XCOF,COF,ROOTR,ROOTI
C
C        THE C MUST ALSO BE REMOVED FROM DOUBLE PRECISION STATEMENTS
C        APPEARING IN OTHER ROUTINES USED IN CONJUNCTION WITH THIS
C        ROUTINE.
C        THE DOUBLE PRECISION VERSION MAY BE MODIFIED BY CHANGING THE
C        CONSTANT IN STATEMENT 78 TO 1.0D-12 AND IN STATEMENT 122 TO
C        1.0D-10.  THIS WILL PROVIDE HIGHER PRECISION RESULTS AT THE
C        COST OF EXECUTION TIME
C
C        ..................................................................
C
      IFIT=0
```

```
      N=M
      IER=0
      IF(XCOF(N+1))10,25,10
   10 IF(N) 15,15,32
C
C       SET ERROR CODE TO 1
C
   15 IER=1
   20 RETURN
C
C     SET ERROR CODE TO 4
C
   25 IER=4
      GO TO 20
C
C       SET ERROR CODE TO 2
C
   30 IER=2
      GO TO 20
   32 IF(N-36) 35,35,30
   35 NX=N
      NXX=N+1
      N2=1
      KJ1 = N+1
      DO 40 L=1,KJ1
      MT=KJ1-L+1
   40 COF(MT)=XCOF(L)
C
C       SET INITIAL VALUES
C
   45 XO=.00500101
      YO=0.01000101
C
C       ZERO INITIAL VALUE COUNTER
C
      IN=0
   50 X=XO
C
C       INCREMENT INITIAL VALUES AND COUNTER
C
      XO=-10.0*YO
      YO=-10.0*X
C
C       SET X AND Y TO CURRENT VALUE
C
      X=XO
      Y=YO
      IN=IN+1
      GO TO 59
   55 IFIT=1
      XPR=X
      YPR=Y
C
C       EVALUATE POLYNOMIAL AND DERIVATIVES
C
   59 ICT=0
   60 UX=0.0
      UY=0.0
      V =0.0
      YT=0.0
      XT=1.0
```

```
      U=COF(N+1)
      IF(U) 65,130,65
   65 DO 70 I=1,N
      L =N-I+1
      TEMP=COF(L)
      XT2=X*XT-Y*YT
      YT2=X*YT+Y*XT
      U=U+TEMP*XT2
      V=V+TEMP*YT2
      FI=I
      UX=UX+FI*XT*TEMP
      UY=UY-FI*YT*TEMP
      XT=XT2
   70 YT=YT2
      SUMSQ=UX*UX+UY*UY
      IF(SUMSQ) 75,110,75
   75 DX=(V*UY-U*UX)/SUMSQ
      X=X+DX
      DY=-(U*UY+V*UX)/SUMSQ
      Y=Y+DY
   78 IF(DABS(DY)+DABS(DX)-1.0D-12) 100,80,80
C
C       STEP ITERATION COUNTER
C
   80 ICT=ICT+1
      IF(ICT-500) 60,85,85
   85 IF(IFIT)100,90,100
   90 IF(IN-5) 50,95,95
C
C       SET ERROR CODE TO 3
C
   95 IER=3
      GO TO 20
  100 DO 105 L=1,NXX
      MT=KJ1-L+1
      TEMP=XCOF(MT)
      XCOF(MT)=COF(L)
  105 COF(L)=TEMP
      ITEMP=N
      N=NX
      NX=ITEMP
      IF(IFIT) 120,55,120
  110 IF(IFIT) 115,50,115
  115 X=XPR
      Y=YPR
  120 IFIT=0
  122 IF(DABS(Y)-1.0D-4*DABS(X)) 135,125,125
  125 ALPHA=X+X
      SUMSQ=X*X+Y*Y
      N=N-2
      GO TO 140
  130 X=0.0
      NX=NX-1
      NXX=NXX-1
  135 Y=0.0
      SUMSQ=0.0
      ALPHA=X
      N=N-1
  140 COF(2)=COF(2)+ALPHA*COF(1)
  145 DO 150 L=2,N
  150 COF(L+1)=COF(L+1)+ALPHA*COF(L)-SUMSQ*COF(L-1)
```

```
  155  ROOTI(N2)=Y
       ROOTR(N2)=X
       N2=N2+1
       IF(SUMSQ) 160,165,160
  160  Y=-Y
       SUMSQ=0.0
       GO TO 155
  165  IF(N) 20,20,45
       END
```

```
      SUBROUTINE CNTRL(N,A,B,B1,AB,CTRL)
C
C
C This subprogram computes the controllability matrix
C
C
      IMPLICIT REAL*8 (A-H,O-Z)
C
      DIMENSION A(N,N),B(N),CTRL(N,N),B1(N),AB(N)
C
C
C
      DO 4 I=1,N
      B1(I)=0.0
 4    B1(I)=B1(I)+B(I)
C
      N1=0
C
      DO 5 I=1,N
      CALL PLACE(N,N1,B1,CTRL)
      CALL MATVEC(N,N,A,B1,AB)
      DO 13 J=1,N
      B1(J)=0.0
 13   B1(J)=B1(J)+AB(J)
 5    CONTINUE
C
      WRITE(6,129)
 129  FORMAT(///,20X,'THE CONTROLLABILITY MATRIX',//)
C
      DO 6 I=1,N
 6    WRITE(6,130) (CTRL(I,J),J=1,N)
 130  FORMAT(3(5X,G16.8))
C
      RETURN
      END
```

```
      SUBROUTINE OBSERV(N,A,C,OBSV,C1,AC,AT)
C
C
C
C This subprogram computes the observabitily matrix
C
C
      IMPLICIT REAL*8 (A-H,O-Z)
C
      DIMENSION A(N,N),C(N),OBSV(N,N),C1(N),AC(N),AT(N,N)
C
      COMMON/INOUT/IN,IOUT
C
      DO 7 I=1,N
      C1(I)=0.0
 7    C1(I)=C1(I)+C(I)
C
      CALL TRANS(N,N,MAT,NAT,A,AT)
C
      N1=0
C
      DO 8 I=1,N
      CALL PLACE(N,N1,C1,OBSV)
      CALL MATVEC(MAT,NAT,AT,C1,AC)
      DO 8 K=1,N
      C1(K)=0.0
 8    C1(K)=C1(K)+AC(K)
C
      WRITE(6,131)
 131  FORMAT(//,20X,'THE OBSERVABILITY MATRIX',//)
C
      DO 10 I=1,N
 10   WRITE(6,133) (OBSV(I,J),J=1,N)
 133  FORMAT(3(5X,G16.8))
C
      CALL SINVAL(OBSV,D)
C
      RETURN
      END
```

```
      SUBROUTINE DSCCON(A,B,N,PD,QD,GG,AA,QDB,DELT)
C
C
C This subprogram computes the matrices P and Q for the
C discrete time linear system
C
C
      IMPLICIT REAL*8 (A-H,O-Z)
C
      DIMENSION A(N,N),B(N),PD(N,N),QD(N,N),QDB(N),
     $ AA(N,N),GG(N,N)
C
      COMMON/INOUT/IN,IOUT
C
C Find the largest component of A
C
C       U=DABS(A(1,1))
C       DO 22 I=1,N
C       DO 22 J=1,N
C       V=DABS(A(I,J))
C       IF(U.LT.V) U=V
C 22    CONTINUE

C
C Now find the value of Qmin
C
C       KMIN=0
C 408   KMIN=KMIN+1
C       GMIN=FLOAT(KMIN)
C       GMA=FLOAT(N)
C       VAL=(GMA)*U*(DELT)
C       CHECK=1.D00/DGAMMA(KMIN)*VAL**(KMIN)*DEXP(VAL)
C       IF(CHECK.GT.0.001) GOTO 408
        KMIN=100
        WRITE(6,180)KMIN
 180    FORMAT(//,25X,'number of iterations = ',1X,I3,//)
C
        DO 23 I=1,N
        DO 23 J=1,N
        PD(I,J)=0.0
        QD(I,J)=0.0
        GG(I,J)=0.0
        IF(I.NE.J) GOTO 90
        PD(I,J)=1.0
        QD(I,J)=1.0
        GG(I,J)=1.0
 90     CONTINUE
 23     CONTINUE
C
C
        DO 31 M=1,KMIN
        CALL MATMUL(N,N,N,N,GG,A,AA)
        DO 24 I=1,N
        DO 24 J=1,N
        AA(I,J)=AA(I,J)*DELT/M
        PD(I,J)=PD(I,J)+AA(I,J)
        QD(I,J)=QD(I,J)+AA(I,J)/(M+1)
        GG(I,J)=AA(I,J)
 24     CONTINUE
 31     CONTINUE
C
```

```
C
        DO 25 I=1,N
        DO 25 J=1,N
 25     QD(I,J)=QD(I,J)*DELT
        CALL MATVEC(N,N,QD,B,QDB)
C
        WRITE(6,101)
 101    FORMAT(//,25X,' THE P MATRIX',//)
        DO 26 I=1,N
 26     WRITE(6,102)(PD(I,J),J=1,N)
 102    FORMAT(3(G16.8,10X))
C
        WRITE(6,105)
 105    FORMAT(//,25X,' THE Q MATRIX',//)
        DO 27 I=1,N
 27     WRITE(6,103) QDB(I)
 103    FORMAT(25X,G16.8)
C
C
        RETURN
        END
```

```
      SUBROUTINE CANON(N,PD,QDB,C,ACAN,BCAN,PHAT,QHAT)
C
C  THIS SUBPROGRAM CALCULATES THE PARAMETERS OF THE
C  CANONICAL FORM
C
      IMPLICIT REAL*8 (A-H,O-Z)
C
      DIMENSION PD(N,N),QDB(N),ACAN(N),BCAN(N),PHAT(N,N),
     $ QHAT(N),C(N)
C
      COMMON/INOUT/IN,IOUT
C
C
C  FIND THE a's FIRST
C
C
C
      ACAN(1)=-(PD(1,1)+PD(2,2)+PD(3,3))
C
      ACAN(2)= PD(1,1)*PD(2,2)+PD(1,1)*PD(3,3)+PD(2,2)*PD(3,3)-
     $         PD(1,2)*PD(2,1)-PD(1,3)*PD(3,1)-PD(2,3)*PD(3,2)
C
      ACAN(3)=-PD(1,1)*PD(2,2)*PD(3,3)+PD(1,1)*PD(2,3)*PD(3,2)+
     $         PD(2,2)*PD(1,3)*PD(3,1)+PD(3,3)*PD(1,2)*PD(2,1)-
     $         PD(1,2)*PD(2,3)*PD(3,1)-PD(1,3)*PD(2,1)*PD(3,2)
C
C NOW LET'S FIND THE b's
C
      BCAN(1)= C(1)*QDB(1)+C(2)*QDB(2)+C(3)*QDB(3)
C
      BCAN(2)=QDB(1)*(-C(1)*(PD(2,2)+PD(3,3))+C(2)*PD(2,1)+
     $        C(3)*PD(3,1))+QDB(2)*(C(1)*PD(1,2)-C(2)*(PD(1,1)+
     $        PD(3,3))+C(3)*PD(3,2))+QDB(3)*(C(1)*PD(1,3)+C(2)*
     $        PD(2,3)-C(3)*(PD(1,1)+PD(2,2)))
C
      BCAN(3)= QDB(1)*(C(1)*(PD(2,2)*PD(3,3)-PD(2,3)*PD(3,2))+
     $                 C(2)*(PD(3,1)*PD(2,3)-PD(2,1)*PD(3,3))+
     $                 C(3)*(PD(2,1)*PD(3,2)-PD(3,1)*PD(2,2)))+
     $          QDB(2)*(C(1)*(PD(3,2)*PD(1,3)-PD(1,2)*PD(3,3))+
     $                 C(2)*(PD(1,1)*PD(3,3)-PD(3,1)*PD(1,3))+
     $                 C(3)*(PD(3,1)*PD(1,2)-PD(1,1)*PD(3,2)))+
     $          QDB(3)*(C(1)*(PD(1,2)*PD(2,3)-PD(1,3)*PD(2,2))+
     $                 C(2)*(PD(2,1)*PD(1,3)-PD(2,3)*PD(1,1))+
     $                 C(3)*(PD(1,1)*PD(2,2)-PD(2,1)*PD(1,2)))
C
C
C Now write the a's and b's
C
       WRITE(6,150)(I,ACAN(I),I=1,3)
 150   FORMAT(//,3(5X,' ALPHA(',I1,')=',1X,G20.12))
C
       SUMA=1.0D0+ACAN(1)+ACAN(2)+ACAN(3)
       WRITE(6,159) SUMA
 159   FORMAT(/,2X,'1 + SUM OF ALPHA = ',G16.8,/)
C
       WRITE(6,151)(I,BCAN(I),I=1,3)
 151   FORMAT(//,3(6X,' BETA(',I1,')=',1X,G20.12))
C
       SUMB=BCAN(1)+BCAN(2)+BCAN(3)
       WRITE(6,160) SUMB
 160   FORMAT(/,2X,'SUM OF BETA = ',G16.8,/)
```

```
C
C FORM THE CANONICAL MATRICES PHAT AND QHAT
C
        DO 50 I=1,N
        DO 51 J=1,N
        PHAT(I,J)=0.0D0
        IF(J.EQ.1) PHAT(I,J)=-ACAN(I)
        L=I+1
        IF(J.EQ.L) PHAT(I,J)=1.0D0
 51     CONTINUE
 50     CONTINUE
C
        DO 55 I=1,N
        QHAT(I)=BCAN(I)
 55     CONTINUE
C
C       WRITE(6,120)
C 120   FORMAT(//,35X,' THE "PHAT" MATRIX',//)
C
C       DO 52 I=1,N
C 52    WRITE(6,121)(PHAT(I,J),J=1,N)
C 121   FORMAT(3(10X,G16.8))
C
C       WRITE(6,123)
C 123   FORMAT(//,35X,' THE QHAT VECTOR',//)
C       DO 53 I=1,N
C 53    WRITE(6,124) QHAT(I)
C 124   FORMAT(35X,G16.8)
C
C
       RETURN
       END
```

```
      SUBROUTINE TFORM(N,PD,C,DOBSV,C1,PDC,PDT,PHAT,
     $                 CNOBSV,T,TT,MT,NT,X,XBAR)
C
C
C THIS SUBPROGRAM COMPUTES THE TRANSFORMATION MATRIX
C
C
      IMPLICIT REAL*8 (A-H,O-Z)
C
      DIMENSION PD(N,N),C(N),DOBSV(N,N),C1(N),PDC(N),
     $ PDT(N,N),PHAT(N,N),CNOBSV(N,N),T(N,N),TT(N,N),
     $ X(N),XBAR(N)
C
      COMMON/INOUT/IN,IOUT
C
C NOW COMPUTE W TRANSPOSE
C
      CALL OBSERV(N,PD,C,DOBSV,C1,PDC,PDT)
C
C INVERSE W TRANSPOSE
C
      CALL MINV(DOBSV,N,DETER,C1,PDC)
C
C NOW COMPUTE W-HAT TRANSPOSE
C
      C(1)=1.0D0
      DO 70 I=2,N
      C(I)=0.D0
 70   CONTINUE
C
      CALL OBSERV(N,PHAT,C,CNOBSV,C1,PDC,PDT)
C
      CALL MATMUL(N,N,N,N,CNOBSV,DOBSV,TT)
C
C
C TRANSPOSE THE TRANSFORMATION MATRIX
C
C
      CALL TRANS(N,N,MT,NT,TT,T)
C
      WRITE(6,430)
 430  FORMAT(//,30X,'THE TRANSFORMATION MATRIX',//)
C
      DO 81 I=1,MT
 81   WRITE(6,432)(T(I,J),J=1,NT)
 432  FORMAT(3(8X,G20.12))
C
C NOW CHECK THE TRANSFORMATION MATRIX
C
      CALL MATMUL(N,N,MT,NT,PD,T,TT)
      CALL MINV(T,N,DETER,C1,PDC)
C
       WRITE(6,437)
 437  FORMAT(//,30X,'INVERSE OF THE TRANSFORMATION MATRIX',//)
      DO 83 I=1,MT
 83   WRITE(6,432)(T(I,J),J=1,NT)
C
C      CALL MATMUL(MT,NT,N,NT,T,TT,PHAT)
C
C      WRITE(6,435)
C 435  FORMAT(//,25X,'THE PHAT MATRIX FROM THE TRANSFORMATION',//)
```

```
C         DO 82 I=1,N
C 82    WRITE(6,436)(PHAT(I,J),J=1,N)
C 436   FORMAT(3(10X,G16.8))
C
C
        CALL MATVEC(N,N,T,X,XBAR)
C
        WRITE(6,438)
 438    ORMAT(//,25X,'THE TRANSFORMED STATES',//)
        WRITE(6,439) (XBAR(J),J=1,N)
 439    FORMAT(30X,G16.8)
C
        RETURN
        END
```

```
      SUBROUTINE MATMUL(MA,NA,MB,NB,A,B,AB)
C
C
C This subprogram multiplies the two matrices and puts the
C results in matrix AB
C
C
      DOUBLE PRECISION A,B,AB
C
      DIMENSION A(MA,NA),B(MB,NB),AB(MA,NB)
C
      DO 1 I=1,MA
      DO 1 J=1,NB
      AB(I,J)=0.0D0
      DO 1 K=1,NA
 1    AB(I,J)=AB(I,J)+A(I,K)*B(K,J)
C
      RETURN
      END
C
C
      SUBROUTINE MATVEC(MA,NA,A,V,W)
C
C MULTIPLIES THE MATRIX A WITH THE VECTOR V
C AND RETURNS THE RESULTS IN VECTOR W
C
      DOUBLE PRECISION A,V,W
C
      DIMENSION A(MA,NA),V(NA),W(MA)
C
C
      DO 10 I=1,MA
      W(I)=0.0D0
      DO 10 J=1,NA
      W(I)=W(I)+A(I,J)*V(J)
 10   CONTINUE
      RETURN
      END
C
C
      SUBROUTINE TRANS(MA,NA,MAT,NAT,A,AT)
C
C
C This subroutine transforms A and puts the result
C in AT
C
      DOUBLE PRECISION A,AT
C
      DIMENSION A(MA,NA),AT(NA,MA)
C
      MAT=NA
      NAT=MA
C
      DO 2 I=1,MAT
      DO 2 J=1,NAT
      AT(I,J)=0.0D0
 2    AT(I,J)=AT(I,J)+A(J,I)
C
      RETURN
      END
C
```

```
C
C
C
C
      SUBROUTINE MATINV(MA,NA,N,A)
C
C
C
C Replaces matrix A by its inverse
C
      DOUBLE PRECISION A,DUM

C
      DIMENSION A(MA,NA)
C
      DO 1 I=1,N
      DUM=A(I,I)
      DO 3 J=1,N
 3    A(I,J)=A(I,J)/DUM
      A(I,I)=1.0D0/DUM
      DO 4 J=1,N
      IF(I.EQ.J) GOTO 4
      DUM=A(J,I)
      DO 5 K=1,N
 5    A(J,K)=A(J,K)-A(I,K)*DUM
      A(J,I)=-A(I,I)*DUM
 4    CONTINUE
 1    CONTINUE
C
      RETURN
      END
C
C
C
C
C
      SUBROUTINE PLACE(N,N1,B,CHECK)
C
C
      DOUBLE PRECISION B,CHECK
C
      DIMENSION B(N),CHECK(N,N)
C
      N1=N1+1
C
      DO 3 J=1,N
      K=N1
      CHECK(J,K)=0.0D0
 3    CHECK(J,K)=CHECK(J,K)+B(J)
C
      RETURN
      END
C
C
C
C
      BLOCK DATA
C
C
C This subprogram initializes the COMMON block
C
```

```
C
      IMPLICIT REAL*8 (A-H,O-Z)
      REAL*8 L0
C
      DIMENSION Q(14),P(7)
C
      COMMON/COEF/Q,P,L0,V,F1,D,RAKE,TEMP1,TEMP2,FORCE
C
      DATA Q/5.2D-5,20.0D0,8000.0D0,5.0D0,22000.0D0,72.0D0,
     $ 2500.0D0,0.056D0,1960.0D0,0.57D0,86.0D0,0.1D0,500.0D0,
     $ 2000.0D0/
C
      DATA P/0.40D0,0.60D0,1.45D0,0.45D0,-0.55D0,-0.95D0,0.76D0/
C
      DATA L0/500.0D0/
C
      END
C
C
C
C
      FUNCTION DGAMMA(N)
C
C
C  THIS FUNCTION CALCULATES THE FACTORIAL OF N
C
C
      DOUBLE PRECISION DGAMMA
C
      DGAMMA=1.D0
      DO 1 I=1,N
      DGAMMA=DGAMMA*I
 1      CONTINUE
C
      RETURN
      END
C
C
C
C
      SUBROUTINE DETER(N,A,DET)
C
C  THIS SUBROUTINE COMPUTES THE DETERMINANT OF 3 BY 3
C  MATRIX
C
      DOUBLE PRECISION A,DET
C
      DIMENSION A(N,N)
C
      DET= A(1,1)*(A(2,2)*A(3,3)-A(3,2)*A(2,3))-
     $       A(1,2)*(A(2,1)*A(3,3)-A(2,3)*A(3,1))+
     $         A(1,3)*(A(2,1)*A(3,2)-A(2,2)*A(3,1))
C
      WRITE(6,105) DET
 105   FORMAT(///,5X,'THE DETERMINANT OF THE ABOVE MATRIX IS ',
     $ 2X,G16.8)
C
      RETURN
      END
C
C
```

```
      FUNCTION WF1(X1,X2,X3,FEED)
C
      IMPLICIT REAL*8 (A-H,O-Z)
      REAL*8 L0
C
      DIMENSION Q(14),P(7)
C
      COMMON/COEF/Q,P,L0,V,F1,D,RAKE,TEMP1,TEMP2,FORCE
C
      FORCED=0.0D0
      IF(FEED.NE.0.0D0) GOTO 201
      GOTO 202
 201  FORCED=(Q(9)*FEED**P(7)*(1.D0-Q(10)*RAKE)-Q(11)-Q(12)*V)*
     $ D+Q(13)*D*(X1+X2)-Q(14)*X3
C
 202  WF1=(V/L0)*(-X1+Q(1)*DCOS(RAKE)*FORCED/(FEED*D))
C
C
C
      RETURN
      END
C
C
      FUNCTION WF2(X1,X2,X3,FEED)
C
      IMPLICIT REAL*8 (A-H,O-Z)
      REAL*8 L0
C
      DIMENSION Q(14),P(7)
C
      COMMON/COEF/Q,P,L0,V,F1,D,RAKE,TEMP1,TEMP2,FORCE
C
      FTEMP= Q(6)*V**P(1)*FEED**P(2)+Q(7)*(X1+X2)**P(3)
C
      WF2= Q(2)*DSQRT(V)*DEXP(-Q(3)/(273.D0+FTEMP))
C
C      WRITE(6,201) FTEMP,WF2
C 201 FORMAT(' FTEMP= ',G16.8,2X,'WF2= ',G16.8)
      RETURN
      END
C
C
C
C
      FUNCTION WC(X1,X2,X3,FEED)
C
      IMPLICIT REAL*8 (A-H,O-Z)
      REAL*8 L0
C
      DIMENSION Q(14),P(7)
C
      COMMON/COEF/Q,P,L0,V,F1,D,RAKE,TEMP1,TEMP2,FORCE
C
      IF(FEED.NE.0.0D0) GOTO 100
      CTEMP=0.0D0
      FORCED=0.0D0
      GOTO 101
 100  FORCED=(Q(9)*FEED**P(7)*(1.D0-Q(10)*RAKE)-Q(11)-Q(12)*V)*
     $ D+Q(13)*D*(X1+X2)-Q(14)*X3
C
      CTEMP=Q(8)*FORCED*V**P(4)*FEED**P(5)*D**P(6)
```

```
C
 101   WC=Q(4)*FORCED*V*DEXP(-Q(5)/(273.D0+CTEMP))
C
C       WRITE(6,201) FORCED,CTEMP,WC
C 201  FORMAT(' FORCED= ',G16.8,2X,'CTEMP= ',G16.8,2X,'WC= ',G16.8)
C
       RETURN
       END
C
C
C
C     ..................................................................
C
C        SUBROUTINE MINV
C
C        PURPOSE
C           INVERT A MATRIX
C
C        USAGE
C           CALL MINV(A,N,D,L,M)
C
C        DESCRIPTION OF PARAMETERS
C           A - INPUT MATRIX, DESTROYED IN COMPUTATION AND REPLACED BY
C               RESULTANT INVERSE.
C           N - ORDER OF MATRIX A
C           D - RESULTANT DETERMINANT
C           L - WORK VECTOR OF LENGTH N
C           M - WORK VECTOR OF LENGTH N
C
C        REMARKS
C           MATRIX A MUST BE A GENERAL MATRIX
C
C        SUBROUTINES AND FUNCTION SUBPROGRAMS REQUIRED
C           NONE
C
C        METHOD
C           THE STANDARD GAUSS-JORDAN METHOD IS USED. THE DETERMINANT
C           IS ALSO CALCULATED. A DETERMINANT OF ZERO INDICATES THAT
C           THE MATRIX IS SINGULAR.
C
C     ..................................................................
C
      SUBROUTINE MINV(A,N,D,L,M)
      DIMENSION A(1),L(1),M(1)
C
C        .....................................................
C
C        IF A DOUBLE PRECISION VERSION OF THIS ROUTINE IS DESIRED, THE
C        C IN COLUMN 1 SHOULD BE REMOVED FROM THE DOUBLE PRECISION
C        STATEMENT WHICH FOLLOWS.
C
      DOUBLE PRECISION A,D,BIGA,HOLD,DABS
C
C        THE C MUST ALSO BE REMOVED FROM DOUBLE PRECISION STATEMENTS
C        APPEARING IN OTHER ROUTINES USED IN CONJUNCTION WITH THIS
C        ROUTINE.
C
C        THE DOUBLE PRECISION VERSION OF THIS SUBROUTINE MUST ALSO
C        CONTAIN DOUBLE PRECISION FORTRAN FUNCTIONS.  ABS IN STATEMENT
C        10 MUST BE CHANGED TO DABS.
C
```

```
C        ..........................................................
C
C        SEARCH FOR LARGEST ELEMENT
C
      D=1.0
      NK=-N
      DO 80 K=1,N
      NK=NK+N
      L(K)=K
      M(K)=K
      KK=NK+K
      BIGA=A(KK)
      DO 20 J=K,N
      IZ=N*(J-1)
      DO 20 I=K,N
      IJ=IZ+I
   10 IF(DABS(BIGA)- DABS(A(IJ))) 15,20,20
   15 BIGA=A(IJ)
      L(K)=I
      M(K)=J
   20 CONTINUE
C
C        INTERCHANGE ROWS
C
      J=L(K)
      IF(J-K) 35,35,25
   25 KI=K-N
      DO 30 I=1,N
      KI=KI+N
      HOLD=-A(KI)
      JI=KI-K+J
      A(KI)=A(JI)
   30 A(JI) =HOLD
C
C        INTERCHANGE COLUMNS
C
   35 I=M(K)
      IF(I-K) 45,45,38
   38 JP=N*(I-1)
      DO 40 J=1,N
      JK=NK+J
      JI=JP+J
      HOLD=-A(JK)
      A(JK)=A(JI)
   40 A(JI) =HOLD
C
C        DIVIDE COLUMN BY MINUS PIVOT (VALUE OF PIVOT ELEMENT IS
C        CONTAINED IN BIGA)
C
   45 IF(BIGA) 48,46,48
   46 D=0.0
      RETURN
   48 DO 55 I=1,N
      IF(I-K) 50,55,50
   50 IK=NK+I
      A(IK)=A(IK)/(-BIGA)
   55 CONTINUE
C
C        REDUCE MATRIX
C
      DO 65 I=1,N
```

```
      IK=NK+I
      HOLD=A(IK)
      IJ=I-N
      DO 65 J=1,N
      IJ=IJ+N
      IF(I-K) 60,65,60
   60 IF(J-K) 62,65,62
   62 KJ=IJ-I+K
      A(IJ)=HOLD*A(KJ)+A(IJ)
   65 CONTINUE
C
C        DIVIDE ROW BY PIVOT
C
      KJ=K-N
      DO 75 J=1,N
      KJ=KJ+N
      IF(J-K) 70,75,70
   70 A(KJ)=A(KJ)/BIGA
   75 CONTINUE
C
C        PRODUCT OF PIVOTS
C
      D=D*BIGA
C
C        REPLACE PIVOT BY RECIPROCAL
C
      A(KK)=1.0/BIGA
   80 CONTINUE
C
C        FINAL ROW AND COLUMN INTERCHANGE
C
      K=N
  100 K=(K-1)
      IF(K) 150,150,105
  105 I=L(K)
      IF(I-K) 120,120,108
  108 JQ=N*(K-1)
      JR=N*(I-1)
      DO 110 J=1,N
      JK=JQ+J
      HOLD=A(JK)
      JI=JR+J
      A(JK)=-A(JI)
  110 A(JI) =HOLD
  120 J=M(K)
      IF(J-K) 100,100,125
  125 KI=K-N
      DO 130 I=1,N
      KI=KI+N
      HOLD=A(KI)
      JI=KI-K+J
      A(KI)=-A(JI)
  130 A(JI) =HOLD
      GO TO 100
  150 RETURN
      END
```

Appendix B

This appendix presents the software developed for designing the adaptive observer. This software, which is prepared for different stages of the design process, can be categorized as:

1. The computer package to design the parameter estimation algorithms for a linear model. The components of this package are:

   LST: Main program which calls different parameter estimation algorithms to operate on a linear model.

   DPAR1: A double-precision least squares parameter estimation algorithm using a diagonal gain matrix and parameter constraints.

   DPAR2: A double-precision least squares parameter estimation algorithm using a full gain matrix which can be reset periodically.

   DPAR3: A double-precision least squares parameter estimation algorithm using a full gain matrix which is updated with a constant trace.

   DPAR4: A double-precision least squares parameter estimation algorithm with U-D factorization of the gain matrix.

2. The computer package to apply the adaptive observer to the nonlinear model. For the simulation of the nonlinear model the simulation package SIMULA [1] is used. The components of this package are:

   LSTNON: Main program to define the nonlinear model and call the necessary subroutines containing the adaptive observer.

   PARAM1: A single-precision least squares parameter estimation routine using a diagonal gain matrix and parameter constraints.

   STATE3: A linear third order state observer.

   BACK: Subroutine to approximate the transformation matrix from the equations obtained from regression analysis (see [2]), and transform the observed states into estimated wear components.

3. The computer package to implement parameter and state estimation techniques on the measured variable. The components of this package are:

   SMAIN: Main program to estimate the parameters of the system on-line.

   SAMPLE: Main program to sample the measured data and store in a data file.

   ESTIM: Main program to test the tool wear estimation method on the recorded data.

   SINPUT: Subroutine to collect the necessary data for sampling, such as: (i) sampling period; (ii) the input value; (iii) the voltage range on the analog to digital converter (ADC); (iv) the channel of the ADC; (v) the sampling gain.

   OPAR21: An output error estimation routine for a first order model.

KDADC: Main program to test the ADC board.

ADC: Subroutine to operate the ADC (in ASSembly).

SSUBS: A package of subroutines to compute the ADC gain, compute the real value of the sampled variable, and reset the real time clock.

There are also routines to operate the reat time clock and the digital I/O port which are discussed in detail in [5].

```
C This program checks the precision of the estimation algorithm
C
C            by
C           Kourosh Danai
C
       IMPLICIT REAL*8 (A-H,O-Z)
       REAL*4 BIT
C
       DIMENSION THETA(6),PHI(6),FGAIN(6,6)
       DIMENSION BIT(6)
C
       REAL*4 YES,ANSWER
       DATA YES/'Y'/
C
C        INTEGER HOUR,MINUTE,SECOND,MILSEC
C
C ********** DELETE THIS SET OF CONTROL EQUNS. AND ENTER YOUR'S ****
C
C  NOW SET UP THE MATRICES
C
C **********   DETERMINE THE VALUE OF N , DCVAL  **********************
C
       N=6
       DCVAL=0.2D0
C
C ***************************************************************
C
       CALL ASSIGN(1,'KDLIN.DAT')
       CALL ASSIGN(5,'TT:')
       CALL ASSIGN(6,'TT:')
       CALL ASSIGN(7,'LP:')
C
C       MINUTE=0
C       SECOND=0
C       MILSEC=0
C       CALL STRCLK
C
C
C SET THE THETA AND GAIN MATRICES
C
C
 500   M=0
        WRITE(6,121)
 121   FORMAT(' ENTER     S1',/)
       READ(5,122) S1
 122   FORMAT(G10.5)
C
       WRITE(6,123)
 123   FORMAT(' ENTER S2',/)
       READ(5,122) S2
C
       Y=0.0D0
       YPAST1=0.0D0
       YPAST2=0.0D0
       YPAST3=0.0D0
       UPAST1=0.0D0
       UPAST2=0.0D0
       UPAST3=0.0D0
       I1=0
       I2=0
C
```

```
C
      DO 10 I=1,N
 10   PHI(I)=0.0D0
C
      DO 11 I=1,N
      DO 11 J=1,N
      FGAIN(I,J)=0.0D0
      FGAIN(I,I)=1000.0D0
C      IF(I.LE.3) FGAIN(I,I)=100.0D0
 11     CONTINUE
C
      THETA(1)=-2.80D0
      THETA(2)= 2.80D0
      THETA(3)=-1.0D0
      THETA(4)= 8.0D0
      THETA(5)=-18.0D0
      THETA(6)= 7.0D0
C
      A1=-2.98292630780D0
      A2= 2.96584835305D0
      A3=-0.982922045614D0
C
      B1= 10.5360079769D0
      B2=-21.0711820226D0
      B3= 10.5351746106D0
C
      WRITE(6,125) A1,A2,A3,B1,B2,B3
D      WRITE(7,125) A1,A2,A3,B1,B2,B3
 125  FORMAT(/,' TRUE PARAMATERS=',3(2X,G16.8),/,17X,3(2X,G16.8),///)
C
C  LET'S CALCULATE E(K+1)
C
 230  Y= A1*YPAST1+A2*YPAST2+A3*YPAST3+
     $      B1*UPAST1+B2*UPAST2+B3*UPAST3
C
C
C
      YPAST3=YPAST2
      YPAST2=YPAST1
      YPAST1=-Y
C
      CALL DPAR4(M,S1,S2,Y,U,FGAIN,PHI,THETA,ERROR)
C
C LET'S CALCULATE THE NEW INPUT
C
       IF(M.LT.5) GOTO 200
      U=DCVAL
      K=M/5
      I=K/2
      J=I*2
      IF(J.EQ.K) U=DCVAL*(1.50D0)
      L=I/2
      J=L*4
      IF(J.EQ.K) U=DCVAL*(0.50D0)
      GOTO 350
 200  U=DCVAL
 350  UPAST3=UPAST2
       UPAST2=UPAST1
      UPAST1=U
C
C  NOW UPDATE THE PHI MATRIX
```

```
C
        YTEST=Y/S1
       UTEST=U*S2
          CALL SETUP(N,PHI,PHI,YTEST,UTEST)
C
C  STOP THE CLOCK
C
C        CALL STPCLK
C        WRITE(IOUT,100) MINUTE,SECOND,MILSEC
C100    FORMAT(' TIME IS: ',I2,'.',I2,' . ',I5)
C
C PREDICT THE OUTPUT
C
       DO 28 I=1,6
       BIT(I)=SNGL(THETA(I))
 28       CONTINUE
       WRITE(1) BIT(1),BIT(2),BIT(3),BIT(4),BIT(5),BIT(6)

C
       IF(M.LE.200) GOTO 230
C
       WRITE(6,128)
 128   FORMAT(' Would you like to rerun the program ?!, Y/N?',/)
       READ(5,129) ANSWER
 129   FORMAT(A2)
       IF(ANSWER.EQ.YES) GOTO 500
C
       CLOSE(UNIT=1)
C
        STOP
       END
```

```
      SUBROUTINE DPAR1(MSTEP,S1,S2,Y,U,FGAIN,PHI,THETA,ERROR)
C
C This is a double precision least squares parameter estimation routine
C using a diagonal gain matrix and parameter constraints
C
C ******************************************************
      IMPLICIT REAL*8 (A-H,O-Z)
C ******************************************************
      COMMON/INOUT/IN,IOUT
C
      DIMENSION THETA(6),PHI(6),FGAIN(6,6)
      DIMENSION BETA(6),C(6),V(6)
      DIMENSION PAT1(6),PAT2(6),PAT3(6)
C
C ******************************************************
      REAL*8 LAMDA1,LAMDA2
C ******************************************************
      DATA LAMDA1/0.95D0/,LAMDA2/1.0D0/
C
      N=6
C
      DO 10 I=1,3
      PAT3(I)=THETA(I)*S1
 10      CONTINUE
      DO 11 I=4,6
      PAT3(I)=THETA(I)/S2
 11      CONTINUE
C
C  LET'S CALCULATE E(K+1)
C
       MSTEP=MSTEP+1
       CALL VVSCAL(N,PAT3,PHI,Z1)
      ERROR=Y-Z1
C
C  UPDATE D,PAT3
C
      DO 30 J=1,N
      PAT1(J)=FGAIN(J,J)*PHI(J)
 30      CONTINUE
C
      DO 31 J=1,N
      K=J-1
      IF(K.EQ.0) GOTO 500
      BETA(J)=BETA(K)+PHI(J)*PAT1(J)
      GOTO 501
 500  BETA(J)=LAMDA1+PHI(J)*PAT1(J)
 501  V(J)=PAT1(J)
 31      CONTINUE
C
C Update the gain matrix
C
      DO 32 J=1,N
      FGAIN(J,J)=FGAIN(J,J)-PAT1(J)*PAT1(J)/BETA(N)
 32      CONTINUE
C
      DO 33 I=1,N
      PAT2(I)=V(I)/BETA(N)
 33      CONTINUE
C
C  UPDATE THETA
C
```

```
       DO 13 I=1,N
        PAT3(I)=PAT3(I)+PAT2(I)*ERROR
 13       CONTINUE
C
C Impose the constraints
C
       GOTO 352
       DSET1=-3.10D0*S1
       USET1=-2.90D0*S1
       IF(PAT3(1).LT.DSET1.OR.PAT3(1).GT.USET1) PAT3(1)=-3.0D0*S1
       DSET2=2.90D0*S1
       USET2=3.10D0*S1
       IF(PAT3(2).LT.DSET2.OR.PAT3(2).GT.USET2) PAT3(2)=3.0D0*S1
       DSET3=-1.20D0*S1
       USET3=-0.80D0*S1
       IF(PAT3(3).LT.DSET3.OR.PAT3(3).GT.USET3) PAT3(3)=-1.0D0*S1
       USET5=-1.5D0*PAT3(4)
       DSET5=-2.5D0*PAT3(4)
       IF(PAT3(5).LT.DSET5.OR.PAT3(5).GT.USET5) PAT3(5)=-2.0D0*PAT3(4)
       USET6=1.5D0*PAT3(4)
       DSET6=0.5D0*PAT3(4)
       IF(PAT3(6).LT.DSET6.OR.PAT3(6).GT.USET6) PAT3(6)=PAT3(4)
C
C  RESCALE THE THETA VECTOR
C
 352   DO 21 J=1,3
 21       THETA(J)=PAT3(J)/S1
       DO 22 I=4,6
 22       THETA(I)=PAT3(I)*S2
C
C PRINT OUT THE VALUES
C
       WRITE(6,101) MSTEP,Y,ERROR,U
D       WRITE(7,101) MSTEP,Y,ERROR,U
 101   FORMAT(,///,' MSTEP=',I4,/,' Y= ',G16.8,10X,'ERROR=',G16.8,
     $  ' U= ',G16.8)
C
       WRITE(6,102) (THETA(K),K=1,6)
D       WRITE(7,102) (THETA(K),K=1,6)
 102   FORMAT(' THETA=',3(1X,G16.8),/,7X,3(1X,G16.8),//)
C
        RETURN
       END
```

```
      SUBROUTINE DPAR2(MSTEP,S1,S2,Y,U,FGAIN,PHI,THETA,ERROR)
C
C This subroutine is a LST with covariance resetting and forgetting factor
C estimation algorithm
C
C            by
C           Kourosh Danai
C
C ****************************************************************
      IMPLICIT REAL*8 (A-H,O-Z)
C ****************************************************************
C
      DIMENSION THETA(6),PHI(6),FGAIN(6,6)
      DIMENSION PAT1(6),PAT2(6),PAT3(6),PAT4(6),PAT5(6,6)
C
C ****************************************************************
      REAL*8 LAMDA1,LAMDA2
C ****************************************************************
      DATA LAMDA1/0.95D0/,LAMDA2/1.0D0/
C
      N=6
C
       DO 10 I=1,3
      PAT3(I)=THETA(I)*S1
 10      CONTINUE
      DO 11 I=4,6
      PAT3(I)=THETA(I)/S2
  11  CONTINUE
C
C  LET'S CALCULATE E(K+1)
C
       MSTEP=MSTEP+1
C
      CALL VVSCAL(N,PAT3,PHI,Z1)
      CALL VECMAT(N,N,PHI,FGAIN,PAT1)
      CALL VVSCAL(N,PAT1,PHI,Z2)
      ERROR=(Y-Z1)/(1.0+Z2)
C
C  PRINT OUT THE STUFF
C
C
C
C  UPDATE THETA
C
      CALL MATVEC(N,N,FGAIN,PHI,PAT2)
      DO 13 I=1,N
 13   PAT3(I)=PAT3(I)+PAT2(I)*ERROR
C
C  UPDATE FGAIN
C
      CALL VVMAT(N,N,PAT2,PAT1,PAT5)
      DO 14 I=1,N
      DO 14 J=1,N
      FGAIN(I,J)=(FGAIN(I,J)-PAT5(I,J)/(LAMDA1/LAMDA2+Z2))/LAMDA1
 14      CONTINUE
C
C
C RESET THE GAIN MATRIX
C
      GOTO 520
      KGAIN=M/10
```

```
       IGAIN=KGAIN/2
       JGAIN=IGAIN*2
       IF(JGAIN.NE.KGAIN) GOTO 520
       DO 24 L=1,N
       DO 24 J=1,N
       FGAIN(L,J)=0.0D0
       FGAIN(L,L)=1000.0D0
 24       CONTINUE
C
C  NOW BACK TRANSFER THE VALUES
C
 520   DO 21 J=1,3
       THETA(J)=PAT3(J)/S1
 21       CONTINUE
       DO 22 K=4,6
       THETA(K)=PAT3(K)*S2
 22       CONTINUE
C
       WRITE(6,102) MSTEP,Y,ERROR,(PHI(I),I=1,6)
D       WRITE(7,102) MSTEP,Y,ERROR,(PHI(I),I=1,6)
 102   FORMAT(' M=',I4,/,' Y= ',G16.8,10X,'ERROR=',G16.8,/,
     $  ' PHI = ',3(2X,G16.8),/,7X,3(2X,G16.8),/)
C
       WRITE(6,104) (THETA(K),K=1,6)
D       WRITE(7,104) (THETA(K),K=1,6)
 104   FORMAT(2X,'THETA =',3(1X,G16.8),/,9X,3(1X,G16.8),//)
C
       RETURN
       END
```

```
      SUBROUTINE DPAR3(MSTEP,S1,S2,Y,U,FGAIN,PHI,THETA,ERROR)
C
C This subroutine is a double precision constant trace LST
C parameter estimation algorithm
C
C            by
C           Kourosh Danai
C
C ************************************************************
      IMPLICIT REAL*8 (A-H,O-Z)
C ************************************************************
C
      DIMENSION THETA(6),PHI(6),FGAIN(6,6)
      DIMENSION PAT1(6),PAT2(6),PAT3(6),PAT4(6),PAT5(6,6)
C
C ************************************************************
      REAL*8 LAMDA1
C ************************************************************
      DATA LAMDA1/1.0D0/,ALPHA/0.8D0/,DTRACE/10000.0D0/
C
      N=6
C
      DO 10 I=1,3
      PAT3(I)=THETA(I)*S1
 10      CONTINUE
      DO 11 I=4,6
      PAT3(I)=THETA(I)/S2
 11      CONTINUE
C
C  LET'S CALCULATE E(K+1)
C
      MSTEP=MSTEP+1
C
      CALL VVSCAL(N,PAT3,PHI,Z1)
      CALL VECMAT(N,N,PHI,FGAIN,PAT1)
      CALL VVSCAL(N,PAT1,PHI,Z2)
      ERROR=(Y-Z1)/(1.0D0+Z2)
C
C  UPDATE THETA
C
      CALL MATVEC(N,N,FGAIN,PHI,PAT2)
      DO 13 I=1,N
 13   PAT3(I)=PAT3(I)+PAT2(I)*ERROR
C
C  UPDATE FGAIN
C
      CALL VVMAT(N,N,PAT2,PAT1,PAT5)
      DO 14 I=1,N
      DO 14 J=1,N
      FGAIN(I,J)=FGAIN(I,J)-PAT5(I,J)/(ALPHA+Z2)
 14      CONTINUE
C
C
C RESET THE GAIN MATRIX
C
      GOTO 520
      KGAIN=M/10
      IGAIN=KGAIN/2
      JGAIN=IGAIN*2
      IF(JGAIN.NE.KGAIN) GOTO 520
      DO 24 L=1,N
```

```
       DO 24 J=1,N
       FGAIN(L,J)=0.0D0
       FGAIN(L,L)=1.0D0
 24       CONTINUE
C
 520   TRACE=0.0D0
       DO 18 I=1,N
       TRACE=TRACE+FGAIN(I,I)
 18       CONTINUE
C
       LAMDA1=TRACE/DTRACE
       IF(LAMDA1.LE.0.0D0) LAMDA1=1.0D0/DTRACE
       IF(LAMDA1.GT.1.0D0) LAMDA1=1.0D0
       DO 19 I=1,N
       DO 19 J=1,N
       FGAIN(I,J)=FGAIN(I,J)/LAMDA1
 19       CONTINUE
C
C  NOW BACK TRANSFER THE VALUES
C
       DO 21 J=1,3
       THETA(J)=PAT3(J)/S1
 21       CONTINUE
       DO 22 K=4,6
       THETA(K)=PAT3(K)*S2
 22       CONTINUE
C
       WRITE(6,102) MSTEP,Y,ERROR,(PHI(I),I=1,6)
D       WRITE(7,102) MSTEP,Y,ERROR,(PHI(I),I=1,6)
 102   FORMAT(' M=',I4,/,' Y= ',G16.8,10X,'ERROR=',G16.8,/,
     $  ' PHI = ',3(2X,G16.8),/,7X,3(2X,G16.8),/)
C
       WRITE(6,103) (THETA(K),K=1,6)
D       WRITE(7,103) (THETA(K),K=1,6)
 103   FORMAT(2X,'THETA =',3(1X,G16.8),/,9X,3(1X,G16.8),//)
C
       RETURN
       END
```

```
      SUBROUTINE DPAR4(MSTEP,S1,S2,Y,U,FGAIN,PHI,THETA,ERROR)
C
C This subprogram is a double precision LST routine with
C U-D decomposition of the gain matrix
C
C            by KOUROSH DANAI
C
C ***************************************************************
      IMPLICIT REAL*8 (A-H,O-Z)
C ***************************************************************
C
      DIMENSION THETA(6),PHI(6),FGAIN(6,6)
      DIMENSION WNEW(6,6),WPAST(6,6)
      DIMENSION BETA(6),WT(6,6),C(6),V(6)
      DIMENSION PAT1(6),PAT2(6),PAT3(6),PAT4(6)
C
C ***************************************************************
      REAL*8 LAMDA1,LAMDA2
C ***************************************************************
      DATA LAMDA1/0.95D0/,LAMDA2/1.0D0/
C
      N=6
C
      DO 10 I=1,3
      PAT4(I)=THETA(I)*S1
 10      CONTINUE
      DO 11 I=4,6
      PAT4(I)=THETA(I)/S2
 11      CONTINUE
C
      IF(MSTEP.GT.0) GOTO 240
      DO 12 J=1,N
      DO 12 K=1,N
      IF(J-K) 201,202,203
 201  WPAST(J,K)=0.0D0
      WNEW(J,K)=0.0D0
      GOTO 12
 202  WPAST(J,K)=1.0D0
      WNEW(J,K)=1.0D0
      GOTO 12
 203  WPAST(J,K)=0.0D0
      WNEW(J,K)=0.0D0
 12      CONTINUE
C
C  LET'S CALCULATE E(K+1)
C
 240  MSTEP=MSTEP+1
       CALL VVSCAL(N,PAT4,PHI,Z1)
      ERROR=Y-Z1
C
C  UPDATE U,D,PAT3
C
      CALL TRANS(N,N,WPAST,WT)
      CALL MATVEC(N,N,WT,PHI,PAT1)
      CALL MATVEC(N,N,FGAIN,PAT1,PAT2)
      DO 30 J=1,N
      K=J-1
      IF(K.EQ.0) GOTO 500
      BETA(J)=BETA(K)+PAT1(J)*PAT2(J)
       FGAIN(J,J)=BETA(K)*FGAIN(J,J)/(BETA(J)*LAMDA1)
      C(J)=-PAT1(J)/BETA(K)
```

```
       GOTO 501
 500   BETA(J)=LAMDA1+PAT1(J)*PAT2(J)
        FGAIN(J,J)=LAMDA1*FGAIN(J,J)/(BETA(J)*LAMDA1)
       C(J)=-PAT1(J)/LAMDA1
 501   V(J)=PAT2(J)
C
       IF(K.EQ.0) GOTO 30
C
        DO 31 I=1,K
       WNEW(I,J)=WPAST(I,J)+V(I)*C(J)
        V(I)=V(I)+WPAST(I,J)*V(J)
 31       CONTINUE
 30       CONTINUE
C
C Update the U matrix
C
       DO 32 I=1,N
       PAT3(I)=V(I)/BETA(N)
       DO 32 J=1,N
       WPAST(I,J)=WNEW(I,J)
 32       CONTINUE
C
C  UPDATE THETA
C
       DO 13 I=1,N
 13    PAT4(I)=PAT4(I)+PAT3(I)*ERROR
C
C  RESCALE THE THETA VECTOR
C
       DO 21 J=1,3
 21       THETA(J)=PAT4(J)/S1
       DO 22 I=4,6
 22       THETA(I)=PAT4(I)*S2
C
C PRINT OUT THE VALUES
C
       WRITE(6,101) MSTEP,Y,ERROR,U
D       WRITE(7,101) MSTEP,Y,ERROR,U
 101   FORMAT(,///,' MSTEP=',I4,/,' Y= ',G16.8,10X,'ERROR=',G16.8,
     $ ' U= ',G16.8)
C
C       WRITE(6,110) (PHI(J),J=1,6)
C       WRITE(7,110) (PHI(J),J=1,6)
C 110  FORMAT(' PHI=',3(2X,G16.8),/,5X,3(2X,G16.8),/)
C
      WRITE(6,102) (THETA(K),K=1,6)
D      WRITE(7,102) (THETA(K),K=1,6)
 102  FORMAT(' THETA=',3(1X,G16.8),/,7X,3(1X,G16.8),//)
C
       RETURN
      END
```

```
C    General Description: This is an adaptive observer routine
C                            for the nonlinear model
C
C    External References: SIMULA
C
C             by KOUROSH DANAI
C
C  This program simulates the tool- wear nonlinear model
C
C Declarations
C
     LOGICAL*1 ANSW,NO,YES
C
C ********** CHANGE DIMENSIONS HERE *********
C
C THE DIMENSION OF X AND XP =NEQN
C THE DIMENSION OF IXC =NCEQN
C THE DIMENSION OF Y,IYC AND IREF =NOUT
C THE DIMENSION OF U AND IUC =NIN
C THE DIMENSION OF IUC  =NOUT
C THE DIMENSION OF WORK = 5*NEQN
C
       REAL*4 U(1)
       DIMENSION X(8),XP(8),Y(1),IXC(9),IYC(1),IUC(1),IREF(1)
       DIMENSION WORK(40)
C
C Common blocks
C
       COMMON/INOUT/IN,IOUT
       COMMON/ADCDAC/ADCGAN,IADCOF,IMXADC,DACGAN,IDACOF,DACMIN,DACMAX
       COMMON/SIZE/NEQN,NIN,NOUT,NCEQN
       COMMON/CLOCK/H,TEND,TPRINT,DELT,DELDST,DELAY
       COMMON/INDEX/ICNT,IX,IY,IU,INXC,INYC,INUC,INDV
       COMMON/MXMN/DVMAX,DVMIN,RIDVMX,RIDVMN
C
C Data statments
C
       CALL ASSIGN(5,'TT:')
       CALL ASSIGN(6,'TT:')
       CALL ASSIGN(7,'LP:')
       DATA IN/5/,IOUT/6/
C
C
C  set up constants for simulation
C
       NO='N'
       YES='Y'
C
C ********** CHANGE SIZES HERE **********
C
C      NIN=# OF PLANT INPUTS
C
       NIN=1
C
C NOUT=# OF PLANT OUTPUTS
C
       NOUT=1
C
C NEQUN= # OF PLANT STATE EQUATIONS
C
       NEQN=8
```

```
C
C NCEQN= # OF CONTROLLER STATE EQUATIONS
C
      NCEQN=9
C
C ********** EDITING IN MAIN COMPLETE *********
C
 10       CALL SIMULA(X,XP,Y,U,IXC,IYC,IUC,IREF,WORK)
C
C Check for a repeat
C
9990    WRITE(IOUT,9991)
9991    FORMAT(' Do you wish to repeat the calculations? (y/n)')
        READ(IN,9992)ANSW
9992    FORMAT(A1)
        IF(ANSW.EQ.YES)GO TO 10
        IF(ANSW.NE.NO)GO TO 9990
        STOP
        END
C
      SUBROUTINE F(T,X,XP,U)
C ********** DO NOT CHANGE THESE DIMENSIONS *********
      REAL*4 T,X(1),XP(1),U(1)
C ********** DELETE THIS SET OF PLANT EQUNS. AND ENTER YOUR'S ****
C
       COMMON/INOUT/IN,IOUT
C
C ******      PUT IN THE VALUE OF THE INPUTS   **********
C
      IF(T.GE.0.01) GOTO 200
C
      A0=500.0
      A1=5.2E-05
      A2=20.0
      A3=8000.0
      A4=5.0
      A5=22000.0
      A6=1960.0
      A7=0.57
      A8=86.0
      A9=0.1
      A10=500.0
       A11=2000.0
      A12=72.0
      A13=2500.0
      A14=0.056
C
      Z1= 0.76
      Z2= 0.4
      Z3= 0.6
      Z4= 1.45
      Z5= 0.45
      Z6=-0.55
      Z7=-0.95
      GAM=0.1745
      V=250.
       D=4.0
C
C INPUTS ARE AS FOLLOWS
C
C       U(1)= f (FEED)
```

```
C
 200   FEED=U(1)
C
C THE EQUATIONS ARE AS FOLLOWS
C
C       X(1)=WF1
C       X(2)=WF2
C       X(3)=WC
C       X(4)=WF
C       X(5)=F
C       X(6)=THETA1
C       X(7)=THETA2
C
       XP(1)= (V/A0)*(-X(1)+A1*X(5)*COS(GAM)/(FEED*D))
       XP(2)= A2*SQRT(V)*EXP(-A3/(273.+X(6)))
       XP(3)= A4*X(5)*V*EXP(-A5/(273.+X(7)))
       XP(4)= 0.0
       X(4) = X(1)+X(2)
       XP(5)= 0.0
       X(5) = (A6*(FEED**Z1)*(1.-A7*GAM)-A8-A9*V)*D+
     $            A10*D*X(4)-A11*X(3)
       XP(6)= 0.0
       X(6) = A12*(V**Z2)*(FEED**Z3)+A13*(X(4)**Z4)
       XP(7)= 0.0
       X(7) = A14*X(5)*(V**Z5)*(FEED**Z6)*(D**Z7)
       XP(8)=0.0
       X(8) = A10*D*X(4)-A11*X(3)
C
C
        RETURN
       END
C
C
C
       SUBROUTINE CONTRL(T,IXC,IYC,Y,IUC,U,IREF)
       COMMON/INOUT/IN,IOUT
       COMMON/CLKBLK/HOUR,MINUTE,SECOND,MILSEC
C ********** DO NOT CHANGE THESE DIMENSIONS *********
       DIMENSION IXC(1),IYC(1),IUC(1),IREF(1),Y(1),U(1)
C
C ********** SUBSTITUTE THE VALUE OF N **************
C
       DIMENSION THETA(6),PHI(6),FGAIN(6,6),XBAR(3)
       DIMENSION TRANS(3,3),XHAT(3),XOLD(3)
C
C WHERE N IN THE ABOVE EQUATIONS IS 2*NEQN
C
       INTEGER HOUR,MINUTE,SECOND,MILSEC
       INTEGER DCVAL
C
C ********** DELETE THIS SET OF CONTROL EQUNS. AND ENTER YOUR'S ****
C
C  NOW SET UP THE MATRICES
C
C **********   DETERMINE THE VALUE OF N , DCVAL  **********************
C
       N=6
       DCVAL=2048+120
C
C ******************************************************************
C
```

```
C       MINUTE=0
C       SECOND=0
C       MILSEC=0
C       CALL STRCLK
C
C
C  SET THE THETA AND GAIN MATRICES
C
       IF(T.GE.0.05) GOTO 230
       MSTEP=0
       S1=10.0
       S2=1.0
       DO 10 I=1,N
       PHI(I)=0.0
       THETA(I)=(FLOAT(IXC(I)))
       K=N+I
       XBAR(I)=(FLOAT(IXC(K)))
 10       CONTINUE
C
       DO 11 J=1,3
       XHAT(J)=0.0
       XOLD(J)=0.0
 11       CONTINUE
C
       DO 12 I=1,N
       DO 12 J=1,N
       FGAIN(I,J)=0.0
       FGAIN(I,I)=1000.0
       IF (I.LE.3) FGAIN(I,I)=1.0
 12       CONTINUE
C
 230   CALL PARAM1(MSTEP,S1,S2,Y(1),U(1),FGAIN,PHI,THETA,ERROR)
C
       CALL STATE(MSTEP,Y(1),U(1),THETA,XBAR)
C
       GOTO 221
       IF(T.LT.0.15) GOTO 221
       V=150.0
       D=2.0
       CALL BACK(T,V,U(1),D,THETA(4),XBAR,TRANS,XHAT)
C
C       DO 21 J=1,3
C       IF(XHAT(J).LT.0.0.OR.XHAT(J).GT.0.5) XHAT(J)=XOLD(J)
C       XOLD(J)=XHAT(J)
C 21   CONTINUE
C
C PRINT OUT THE VALUES
C
C       ICOUNT=MSTEP/40
C       JCOUNT=ICOUNT*40
C
       WRITE(IOUT,205)
C       IF(JCOUNT.EQ.MSTEP) WRITE(7,205)
 205   FORMAT(//,25X,'THE TRANSFORMED STATES',//)
       WRITE(IOUT,206) (XHAT(J),J=1,3)
C       IF(JCOUNT.EQ.MSTEP) WRITE(7,206) (XHAT(J),J=1,3)
 206   FORMAT(2X,3(5X,G16.8),/)
C
C LET'S CALCULATE THE NEW INPUT
C
 221   IF(T.LT.100.0) GOTO 200
```

```
        IUC(1)=DCVAL
        K=IFIX(T/0.2)
        I=K/4
        J=I*4
        IF(J.EQ.K) IUC(1)=DCVAL+10
        L=I/2
        J=L*8
        IF(J.EQ.K) IUC(1)=DCVAL-10
        GOTO 350
  200   IUC(1)=DCVAL
  350   U(1)=(FLOAT(IUC(1)-2048))*0.0025
C
C  NOW UPDATE THE PHI MATRIX
C
         YTEST=Y(1)/S1
        UTEST=U(1)*S2
         CALL SETUP(N,PHI,PHI,YTEST,UTEST)
C
C  NOW BACK TRANSFER THE VALUES
C
         DO 15 I=1,N
        IXC(I)=IFIX(THETA(I)*100.)
 15        CONTINUE
C
        DO 18 J=1,3
        I=J+N
        IXC(I)=IFIX(XBAR(J))
 18        CONTINUE
C
C  STOP THE CLOCK
C
C        CALL STPCLK
C        WRITE(IOUT,100) MINUTE,SECOND,MILSEC
C100    FORMAT(' TIME IS: ',I2,'.',I2,' . ',I5)
C
C PREDICT THE OUTPUT
C
         RETURN
        END
C
        SUBROUTINE YEQUNS(T,X,Y,U)
C ********** DO NOT CHANGE THESE DIMENSIONS *********
        DIMENSION X(1),Y(1),U(1)
C ********** DELETE THIS SET OF EQUNS. AND ENTER YOUR'S ****
C
C
C GENERATE THE NOISE
C
C        IF(T.GT.0.05) GOTO 500
C
C        I1=0
C        I2=0
C
C 500   XNOISE=RAN(I1,I2)*50.0
C
        XNOISE=0.0
        Y(1)=X(8)+XNOISE
C
        RETURN
        END
```

```
      SUBROUTINE PARAM1(MSTEP,S1,S2,Y,U,FGAIN,PHI,THETA,ERROR)
C
C This is a single precision least squares parameter estimation routine
C using a diagonal gain matrix and parameter constraints
C
      COMMON/INOUT/IN,IOUT
C
      DIMENSION THETA(6),PHI(6),FGAIN(6,6)
      DIMENSION BETA(6),C(6),V(6)
      DIMENSION PAT1(6),PAT2(6),PAT3(6)
C
      REAL*4 LAMDA1,LAMDA2
      DATA LAMDA1/0.95/,LAMDA2/1.0/
C
      N=6
C
      DO 10 I=1,3
      PAT3(I)=THETA(I)*S1
 10      CONTINUE
      DO 11 I=4,6
      PAT3(I)=THETA(I)/S2
 11      CONTINUE
C
C  LET'S CALCULATE E(K+1)
C
       MSTEP=MSTEP+1
       CALL VVSCAL(N,PAT3,PHI,Z1)
      ERROR=Y-Z1
C
C  UPDATE D,PAT3
C
      DO 30 J=1,N
      PAT1(J)=FGAIN(J,J)*PHI(J)
 30      CONTINUE
C
      DO 31 J=1,N
      K=J-1
      IF(K.EQ.0) GOTO 500
      BETA(J)=BETA(K)+PHI(J)*PAT1(J)
      GOTO 501
 500  BETA(J)=LAMDA1+PHI(J)*PAT1(J)
 501  V(J)=PAT1(J)
 31      CONTINUE
C
C Update the gain matrix
C
      DO 32 J=1,N
      FGAIN(J,J)=FGAIN(J,J)-PAT1(J)*PAT1(J)/BETA(N)
 32      CONTINUE
C
      DO 33 I=1,N
      PAT2(I)=V(I)/BETA(N)
 33      CONTINUE
C
C  UPDATE THETA
C
      DO 13 I=1,N
       PAT3(I)=PAT3(I)+PAT2(I)*ERROR
 13   CONTINUE
C
C Impose the constraints
```

```
C
C       IF(IFLAG.EQ.0)     GOTO 352
       DSET1=-3.10*S1
       USET1=-2.90*S1
       IF(PAT3(1).LT.DSET1.OR.PAT3(1).GT.USET1) PAT3(1)=-3.0*S1
       DSET2=2.90*S1
       USET2=3.10*S1
       IF(PAT3(2).LT.DSET2.OR.PAT3(2).GT.USET2) PAT3(2)= 3.0*S1
       DSET3=-1.20*S1
       USET3=-0.80*S1
       IF(PAT3(3).LT.DSET3.OR.PAT3(3).GT.USET3) PAT3(3)=-1.0*S1
       USET5=-1.5*PAT3(4)
       DSET5=-2.5*PAT3(4)
       IF(PAT3(5).LT.DSET5.OR.PAT3(5).GT.USET5) PAT3(5)=-2.0*PAT3(4)
       USET6=1.5*PAT3(4)
       DSET6=0.5*PAT3(4)
       IF(PAT3(6).LT.DSET6.OR.PAT3(6).GT.USET6) PAT3(6)=PAT3(4)
C
C  RESCALE THE THETA VECTOR
C
 352   DO 21 J=1,3
 21       THETA(J)=PAT3(J)/S1
       DO 22 I=4,6
 22       THETA(I)=PAT3(I)*S2
C
C PRINT OUT THE VALUES
C
       ICOUNT=MSTEP/20
       JCOUNT=ICOUNT*20
       WRITE(6,101) MSTEP,Y,ERROR,U
       IF(JCOUNT.EQ.MSTEP) WRITE(7,101) MSTEP,Y,ERROR,U
 101   FORMAT(,///,' MSTEP=',I4,/,' Y= ',G16.8,10X,'ERROR=',G16.8,
     $ ' U= ',G16.8)
C
       WRITE(6,102) (THETA(K),K=1,6)
       IF(JCOUNT.EQ.MSTEP) WRITE(7,102) (THETA(K),K=1,6)
 102   FORMAT(' THETA=',3(1X,G16.8),/,7X,3(1X,G16.8),//)
C
        RETURN
       END
```

```
      SUBROUTINE STATE(MSTEP,Y,U,THETA,XBAR)
C
C This a state observer with fixed gain values
C
      COMMON/INOUT/IN,IOUT
      DIMENSION XBAR(3),THETA(6),VEC2(3),GAIN(3),E(3,3),F(3)
C
      GAIN(1)=-THETA(1)
      GAIN(2)=-THETA(2)
      GAIN(3)=-THETA(3)
C
       E(1,1)=-THETA(1)
      E(1,2)=1.0
      E(1,3)=0.0
      E(2,1)=-THETA(2)
      E(2,2)=0.0
      E(2,3)=1.0
      E(3,1)=-THETA(3)
      E(3,2)=0.0
      E(3,3)=0.0
C
       F(1)= THETA(4)
      F(2)= THETA(5)
      F(3)= THETA(6)
C
       ERROR=YPAST-XBAR(1)
C
      CALL MATVEC(3,3,E,XBAR,VEC2)
C
      DO 3 J=1,3
      XBAR(J)=VEC2(J)+F(J)*U+GAIN(J)*ERROR
 3      CONTINUE
C
C PRINT OUT THE VALUES
C
      ICOUNT=MSTEP/20
      JCOUNT=ICOUNT*20
C
      WRITE(IOUT,102) ERROR,(XBAR(I),I=1,3)
      IF(JCOUNT.EQ.MSTEP) WRITE(7,102) ERROR,(XBAR(I),I=1,3)
 102  FORMAT(' ERROR=',G16.8,/,' XBAR=',3(2X,G15.8),/)
      YPAST=Y
C
      RETURN
      END
```

```
      SUBROUTINE BACK(TIME,V,F,D,BETA1,XBAR,T,X)
C
C THIS SUBPROGRAM COMPUTES THE TRANSFORMATION MATRIX
C AND TRANSFORMS THE ESTIMATED STATES INTO ORIGINAL STATES
C
      DIMENSION T(3,3),XBAR(3),X(3),B(3),C(3)
C
      COMMON/INOUT/IN,IOUT
C
C NOW COMPUTE THE T MATRIX
C
      T(1,1)= 500.0*D
      T(1,2)= 500.0*D
      T(1,3)=-2000.0
C
      T(2,1)=-1000.0*D-(1.783380E-06*(TIME**0.964060)*
     $          (V**1.38290)*(F**5.20270)*(D**(-1.46500))*
     $          (BETA1**2.60360))
      T(2,2)=(-995.0+0.050*(V-100.0))*D-(2.763535E-06*
     $          (V**2.00580)*(F**0.154940)*(D**1.00960)*
     $          (BETA1**(-0.0177230)))
      T(2,3)=(3980.0-0.20*(V-100.0))+(7.497504E-02*
     $          (TIME**0.322780)*(V**(0.68170))*(F**3.45220)*
     $          (D**(-2.02130))*(BETA1**2.12400))
C
      T(3,1)=500.0*D+(2.545777E-11*(TIME**0.982830)*
     $            (V**3.50660)*(F**3.00210)*(D**(0.700920))*
     $          (BETA1**0.401560))
      T(3,2)=(495.00-0.050*(V-100.0))*D+(2.714236E-06*
     $            (TIME**0.00127560)*(V**2.00880)*(F**0.154460)*
     $          (D**1.01130)*(BETA1**(-0.0193010)))
      T(3,3)=(-1980.0+0.20*(V-100.0))-(7.386619E-02*
     $          (TIME**0.320170)*(V**(0.681920))*(F**3.43090)*
     $          (D**(-2.00870))*(BETA1**2.10970))
C
C       WRITE(7,201)
C 201   FORMAT(//,30X,'INVERSE OF THE TRANSFORMATION MATRIX',//)
C       DO 10 I=1,3
C 10    WRITE(7,202)(T(I,J),J=1,3)
C 202   FORMAT(3(8X,G20.12))
C
C COMPUTE THE TRANSFORMATION MATRIX
C
      CALL MATINV(3,3,3,T)
C
C     WRITE(7,203)
C 203 FORMAT(//,30X,'THE TRANSFORMATION MATRIX',//)
C
C     DO 11 I=1,3
C 11  WRITE(7,204)(T(I,J),J=1,3)
C 204 FORMAT(3(8X,G20.12))
C
C NOW CHECK THE TRANSFORMATION MATRIX
C
      CALL MATVEC(3,3,T,XBAR,X)
C
C       WRITE(7,205)
C 205  FORMAT(//,25X,'THE TRANSFORMED STATES',//)
C       WRITE(7,206) (X(J),J=1,3)
C 206  FORMAT(2X,3(5X,G16.8),/)
C
```

```
        RETURN
        END
```

```
C
C This program tests the on-line tool wear estimation method
C on an actual system.
C
C
C            by
C           Kourosh Danai
C
      INTEGER RANGE,CHANNL,GAIN,VALUE
      INTEGER HOUR,MINUTE,SECOND,MILSEC
      REAL*4 YES,ANSWER
C
      DIMENSION THETA(6),PHI(6),FGAIN(6,6)
C
      COMMON/INOUT/IN,IOUT
      COMMON/CLKBLK/HOUR,MINUTE,SECOND,MILSEC
C
      DATA YES/'Y'/
      DATA IN/5/,IOUT/6/
C
      CALL ASSIGN(3,'DL0:KDDAT.DAT')
      CALL ASSIGN(5,'TT:')
      CALL ASSIGN(6,'TT:')
      CALL ASSIGN(7,'LP:')
C
      CALL SINPUT(DELT,U,VOLTAG,CHANNL,SCALE,S1,S2)
C
      M=0
      N=6
C

C  SET THE THETA AND GAIN MATRICES
C
      DO 1 I=1,N
 1      THETA(I)=0.0
C
      DO 2 I=1,N
      DO 2 J=1,N
      FGAIN(I,J)=0.0
      FGAIN(I,I)=1000.0
      IF(I.LE.3) FGAIN(I,I)=1.0
 2      CONTINUE
C
C determine the gain for the ADC board
C
      CALL KDADC(VOLTAG,CHANNL,GAIN)
C
      GOTO 10
 20      CALL STPCLK
       WRITE(IOUT,101) M
 101  FORMAT(' Would you like to CONTINUE the program? Y/N?!',
     $ '    MSTEP= ',I3,/,)
       READ(IN,102) ANSW
 102  FORMAT(A2)
      IF(ANSW.NE.YES) GOTO 40
C
 10      CALL DRVRED(1,VALUE)
      IF(VALUE.EQ.-2) GOTO 10
C
C Set the flag for the constraints
C
```

```
       IFLAG=1
C
C Reset the PHI vector and FGAIN matrix
C
       DO 3 I=1,N
       PHI(I)=0.0
 3       CONTINUE
C
       YTEST=0.0
       UTEST=U*S2
       CALL SETUP(N,PHI,PHI,YTEST,UTEST)
C
       DO 4 I=1,N
       DO 4 J=1,N
       FGAIN(I,J)=0.0
       FGAIN(I,I)=1000.0
       IF(I.LE.3) FGAIN(I,I)=1.0
 4       CONTINUE
C
       CALL SETTIM(0,0,0,0)
       CALL ADC(GAIN,IVALUE)
       CALL FACTOR(SCALE,VOLTAG,IVALUE,OUTPUT)
       WRITE(IOUT,120) OUTPUT
 120   FORMAT(' FORCE= ',G15.8)
C
       DCVAL=OUTPUT
       CALL SETTIM(0,0,0,0)
C
       CALL STRCLK
C
 30       IF(SECOND.LT.DELT) GOTO 30
       CALL SETTIM(0,0,0,0)
       CALL ADC(GAIN,IVALUE)
       CALL FACTOR(SCALE,VOLTAG,IVALUE,OUTPUT)
       Y=OUTPUT-DCVAL
C
       CALL PARAM1(M,S1,S2,Y,U,FGAIN,PHI,THETA,ERROR,IFLAG)
D       IFLAG=0
C
C Store the variables
C
        WRITE(3) M,Y,U,(THETA(J),J=1,6),ERROR
C
C Now update the PHI vector
C
        YTEST=Y/S1
       UTEST=U*S2
          CALL SETUP(N,PHI,PHI,YTEST,UTEST)
C
       WRITE(IOUT,100) MINUTE,SECOND,MILSEC
 100   FORMAT(' TIME IS: ',I2,'.',I2,' . ',I5)
C
       CALL DRVRED(1,VALUE)
       IF(VALUE.EQ.-2) GOTO 20
       GOTO 30
C
C  STOP THE CLOCK
C
 40       WRITE(6,103)
 103   FORMAT(' Would you like to print out the values?!, Y/N?',/)
       READ(5,102) ANSW
```

```
      IF(ANSW.NE.YES) GOTO 70
C
      WRITE(7,104)
 104  FORMAT(1X,'Mstep',7X,'Output',9X,'Input',/,'Alpha1',9X,
     $  'Alpha2',9X,'Alpha3',9X,'Beta1',10X,'Beta2',10X,'Beta3',10X,
     $  'ERROR'//)
C
      REWIND 3
      DO 5 J=1,M
      READ(3) MSTEP,Y,U,(THETA(I),I=1,6),ERROR
      WRITE(7,105) MSTEP,Y,U,(THETA(I),I=1,6),ERROR
 105  FORMAT(1X,I4,7X,2(1X,G14.8),/,7(1X,G14.8))
 5      CONTINUE
C
 70      CLOSE(UNIT=3)
      STOP
      END
```

```
C
C This program samples the forces one axis at a time
C
C
C            by
C           Kourosh Danai
C
       INTEGER RANGE,CHANNL,VALUE,DELT
       INTEGER GAIN1,GAIN2,GAIN3
       INTEGER CH1,CH2,CH3
       INTEGER HOUR,MINUTE,SECOND,MILSEC
       REAL*4 YES,ANSWER
C
       COMMON/INOUT/IN,IOUT
       COMMON/CLKBLK/HOUR,MINUTE,SECOND,MILSEC
C
       DATA YES/'Y'/
       DATA IN/5/,IOUT/6/
C
       CALL ASSIGN(1,'DL0:KDX1.DAT')
D       CALL ASSIGN(2,'DL0:Y1.DAT')
D       CALL ASSIGN(3,'DL0:Z1.DAT')
       CALL ASSIGN(5,'TT:')
       CALL ASSIGN(6,'TT:')
       CALL ASSIGN(7,'LP:')
C
       CALL SINPUT(DELT,U,VOLTAG,CHANNL,SCALE,S1,S2)
C
       WRITE(IOUT,505) DELT
 505   FORMAT(' DELT= ',I3)

       M=0
       CH1=1
D       CH2=2
D       CH3=3
C
C determine the gain for the ADC board
C
       CALL KDADC(VOLTAG,CH1,GAIN1)
D       CALL KDADC(VOLTAG,CH2,GAIN2)
D       CALL KDADC(VOLTAG,CH3,GAIN3)
C
       GOTO 10
 20       CALL STPCLK
          WRITE(IOUT,101) M
 101   FORMAT(' Would you like to CONTINUE the program? Y/N?!',
     $ '    MSTEP= ',I3,/,)
          READ(IN,102) ANSW
 102   FORMAT(A2)
       IF(ANSW.NE.YES) GOTO 40
C
 10       CALL DRVRED(1,VALUE)
       IF(VALUE.EQ.-2) GOTO 10
C
       CALL SETTIM(0,0,0,0)
       CALL ADC(GAIN1,IVAL1)
D       CALL ADC(GAIN2,IVAL2)
D       CALL ADC(GAIN3,IVAL3)
C
       CALL FACTOR(SCALE,VOLTAG,IVAL1,OUT1)
D       CALL FACTOR(SCALE,VOLTAG,IVAL2,OUT2)
```

```
D        CALL FACTOR(SCALE,VOLTAG,IVAL3,OUT3)
C
        WRITE(IOUT,120) OUT1
D        WRITE(IOUT,120) OUT1,OUT2,OUT3
 120    FORMAT(' XFORCE= ',G16.8//)
D 120   FORMAT(' XFORCE= ',G15.8,2X,'YFORCE= ',G16.8,2X,'ZFORCE=',G16.8//)
C
        DCVAL1=OUT1
D        DCVAL2=OUT2
D        DCVAL3=OUT3
C
        CALL SETTIM(0,0,0,0)
C
        CALL STRCLK
C
 30        IF(MILSEC.LT.DELT) GOTO 30
        CALL SETTIM(0,0,0,0)
C
        CALL ADC(GAIN1,IVAL1)
D        CALL ADC(GAIN2,IVAL2)
D        CALL ADC(GAIN3,IVAL3)
C
        CALL FACTOR(SCALE,VOLTAG,IVAL1,OUT1)
D        CALL FACTOR(SCALE,VOLTAG,IVAL2,OUT2)
D        CALL FACTOR(SCALE,VOLTAG,IVAL3,OUT3)
C
D        X=OUT1
        X=OUT1-DCVAL1
D        Y=OUT2-DCVAL2
D        Z=OUT3-DCVAL3

C
C Store the variables
C
        M=M+1
         WRITE(1) M,X
D         WRITE(2) M,Y
D        WRITE(3) M,Z
C
        WRITE(IOUT,107) M,X
 107    FORMAT(' M= ',I3,5X,' FORCE= ',G16.8)
C
D        WRITE(IOUT,100) MINUTE,SECOND,MILSEC
D 100   FORMAT(' TIME IS: ',I2,'.',I2,' . ',I5)
C
        CALL DRVRED(1,VALUE)
        IF(VALUE.EQ.-2) GOTO 20
        GOTO 30
C
C  STOP THE CLOCK
C
 40        WRITE(6,103)
 103    FORMAT(' Would you like to print out the values?!, Y/N?',/)
        READ(5,102) ANSW
        IF(ANSW.NE.YES) GOTO 70
C
        WRITE(7,104)
 104    FORMAT(1X,'Mstep',7X,'XFORCE'//)

D 104   FORMAT(1X,'Mstep',7X,'XFORCE',9X,'YFORCE',9X,'ZFORCE'//)
C
```

```
      REWIND 1
D      REWIND 2
D      REWIND 3
C
      DO 5 J=1,M
      READ(1) MSTEP,X
D      READ(2) MSTEP,Y
D      READ(3) MSTEP,Z
      WRITE(7,105) MSTEP,X
D      WRITE(7,105) MSTEP,X,Y,Z
 105  FORMAT(1X,I4,7X,G14.8)
D 105 FORMAT(1X,I4,7X,3(1X,G14.8))
 5      CONTINUE
C
 70       CLOSE(UNIT=1)
D       CLOSE(UNIT=2)
D       CLOSE(UNIT=3)
C
       STOP
       END
```

```
C
C This program tests the on-line tool wear estimation method
C on the collected data.
C
C
C            by
C           Kourosh Danai
C
      REAL*4 YES,ANSWER
C
      DIMENSION THETA(2),PHI(2),FGAIN(2,2)
C
      COMMON/INOUT/IN,IOUT
C
      DATA YES/'Y'/
      DATA IN/5/,IOUT/6/
C
      CALL ASSIGN(1,'OPARX2.DAT')
      CALL ASSIGN(2,'KDX2.DAT')
C
      CALL ASSIGN(5,'TT:')
      CALL ASSIGN(6,'TT:')
      CALL ASSIGN(7,'LP:')
C
      S1=10.0
      S2=1.0
      U = .254
      MVAL=84
      N=2
      E=0.0
C
C  SET THE THETA VECTOR AND GAIN MATRIX
C
      DO 1 I=1,N
      THETA(I)=0.0
      PHI(I)=0.0
 1      CONTINUE
C
      DO 2 I=1,N
      DO 2 J=1,N
      FGAIN(I,J)=0.0
      FGAIN(I,I)=1000.0
      IF(I.EQ.1) FGAIN(I,I)=1.0
 2      CONTINUE
C
      REWIND 2
C
      DO 11 K=1,MVAL
      READ(2) MSTEP,Y
      IF(MSTEP.GE.17) Y = Y + 21.88
      IF(MSTEP.GE.33) Y = Y + 55.88
      IF(MSTEP.GE.50) Y = Y + 21.49
      IF(MSTEP.GE.63) Y = Y + 13.28
      IF(MSTEP.GE.50) Y = Y + 23.05
C      WRITE(IOUT,120) Y
C 120  FORMAT(' FORCE= ',G15.8)
C
      CALL OPAR21(MSTEP,S1,S2,Y,U,FGAIN,PHI,THETA,ERROR,E)
D      IFLAG=0
      CALL STATE1(Y,U,THETA,XBAR)
      X = XBAR/200.0
```

```
       WRITE(IOUT,109) X
 109   FORMAT(' The estimated state=',G16.8,/)
C
C Store the variables
C
        WRITE(1) MSTEP,Y,U,(THETA(J),J=1,2),ERROR,X
C
C Now update the PHI vector
C
        YTEST=Y/S1
       UTEST=U*S2
          CALL SETUP(N,PHI,PHI,YTEST,UTEST)
 11       CONTINUE
C
       CLOSE(UNIT=1)
        CLOSE(UNIT=2)
C
       STOP
       END
```

```
      SUBROUTINE SINPUT(DELT,U,VOLTAG,CHANNL,SCALE,S1,S2)
C
C Subroutine SINPUT is designed to display various sampling parameters for the
C user, and to prompt him for new ones.  The parameters are passed back to the
C calling program through the common blocks.
C
      INTEGER   DELT,CHANNL,RANGE
C
      COMMON/INOUT/IN,IOUT
C
C Increment the parameter flag. The parameter flag is used to tell the pro-
C gram whether this is the user's first run through the program.  If it is
C the first time through the program, all the sampling parameters will be in-
C itialized by the subroutine SINPUT.  On subsequent passes through the pro-
C gram, the sampling parameters that the user entered during the previous pass
C will be retained.  Thus it will be easier for the user if he wants to change
C only one parameter per pass.
C
      PARFLG = PARFLG + 1
C
C Check to see if this is the first time the user is passing through SINPUT,
C and branch appropriately.
C
      IF ( PARFLG .GT. 1 ) GO TO 9
C
C Since this is the user's first pass through the program, initialize cut-
C ting parameters. Note that the ADCGAN factor has units of volts/unit number.
C
      DELT   = 500
      U      = 0.01
      RANGE  = 4
      CHANNL = 1
      SCALE  = 1250.0
      S1=10.0
      S2=1.0
      U=U*25.4
C
C
C Display a header.
C
 9    IF(RANGE.EQ.1) VOLTAG=10.24
       IF(RANGE.EQ.2) VOLTAG=5.12
       IF(RANGE.EQ.3) VOLTAG=2.56
       IF(RANGE.EQ.4) VOLTAG=1.28
      WRITE(IOUT,101)
 101  FORMAT('1'/'1'/'1'/
     + 20X, 'TOOL WEAR ESTIMATION PROGRAM' )
C
C Display the present value of all the sampling parameters.
C
      WRITE(IOUT,102) DELT,U,VOLTAG,CHANNL,SCALE,S1,S2
 102  FORMAT( ////// 16X, 'SAMPLING PARAMETERS DISPLAY:'    //
     +  2X,' 1) Sampling period is :                              ',
     +                    I7,' (milsec)' /
     +  2X,' 2) Feed input is :                                   ',
     +                    G10.4,' (mm)' /
     +  2X,' 3) Output range of the ADC channel is :              ',
     +                    G10.4,' (volts)' /
     +  2X,' 4) Channel no. on the ADC board is :                 ',
     +                    I7,' (unitless)' /
     +  2X,' 5) Scale factor of the charge amplifier is :         ',
```

```
     +                    G15.8,' (Newtons/Volt)' /
     +  2X,' 6) Force dividing factor for estimation :          ',
     +                    G7.2,' (unitless)' /
     +  2X,' 7) Input multiplication factor for estimation is :  ',
     +                    G7.2,' (seconds)' /////)
C
C Ask the user if he wants to change any of the present values of the
C sampling parameters.
C
       WRITE(IOUT,103)
 103   FORMAT( ' To change any of the parameter values, enter the',
     + ' number of the parameter.'/
     + ' Otherwise RETURN to continue : ' )
       READ(IN,104,ERR=9) IFLAG
 104   FORMAT(I2)
C
       IF ( IFLAG .LT. 0 .OR. IFLAG .GT. 7 ) GO TO 9
C
       IF ( IFLAG .EQ. 0 ) GO TO 5000
C
       GO TO ( 301, 302, 303, 304, 305, 306, 307 ) IFLAG
C
 301   ASSIGN 301 TO I
       WRITE  ( IOUT, 201 )
 201   FORMAT ( '1'/'1'/'1'/ '$Enter a new value for the',
     + ' sampling period : ' )
       READ   ( IN, 202, ERR=990 ) DELT
 202   FORMAT ( I2 )
       GO TO 9
C
 302   ASSIGN 302 TO I
       WRITE  ( IOUT, 203 )
 203   FORMAT ( '1'/'1'/'1'/ '$Enter a new value for the',
     + ' feed input (in/rev) : ' )
       READ   ( IN, 204, ERR=990 ) U
 204   FORMAT(G8.4)
       U=U*25.4
       GO TO 9
C
 303   ASSIGN 303 TO I
       WRITE  ( IOUT, 205 )
 205    FORMAT ( '1'/'1'/'1'/' Enter a new value for the',
     +  ' voltage range on the ADC board : ',/,
     +  '    1.    +/- 10.24 V',/,
     +  '    2.    +/- 5.12 V',/,
     +  '    3.    +/- 2.56 V',/,
     +  '    4.    +/- 1.28 V',/,
     +  ' Please enter 1, 2, 3, or 4',/)
       READ   ( IN, 202, ERR=990 ) RANGE
       GO TO 9
C
 304    ASSIGN 304 TO I
       WRITE  ( IOUT, 206 )
 206   FORMAT ( '1'/'1'/'1'/ '$Enter a new value for the',
     +  ' channel No. : ' )
       READ ( IN, 202, ERR=990 ) CHANNL
       GO TO 9
C
305       ASSIGN 305 TO I
       WRITE  ( IOUT, 207 )
207       FORMAT ( '1'/'1'/'1'/ '$Enter a new value for the',
```

```
     +  ' amplifier scale factor : ' )
        READ    ( IN, 204, ERR=990 ) SCALE
        GO TO 9
C
 306     ASSIGN 306 TO I
        WRITE   ( IOUT, 208 )
 208    FORMAT ( '1'/'1'/'1'/ '$Enter a new value for the',
     +  ' dividing factor : ' )
        READ    ( IN, 204, ERR=990 ) S1
        GO TO 9
C
 307    ASSIGN 307 TO I
        WRITE   ( IOUT, 209 )
 209    FORMAT ( '1'/'1'/'1'/ '$Enter a new value for the',
     +  ' multiplication factor : ' )
        READ    ( IN, 204, ERR=990 ) S2
        GO TO 9
C
C
 990    WRITE(IOUT,991)
 991    FORMAT(' ***** ERROR ***** Invalid entry, try again',/,' ')
        GO TO I
C
C Return to the calling program.
C
 5000   RETURN
        END
```

```
      SUBROUTINE OPAR21(MSTEP,S1,S2,Y,U,FGAIN,PHI,THETA,ERROR,E)
C
C    General Description: This is an output error estimation
C                         routine for a first order model
C
C              by
C            Kourosh Danai
C
C
      COMMON/INOUT/IN,IOUT
C
      DIMENSION THETA(2),PHI(2),FGAIN(2,2)
      DIMENSION PAT1(2),PAT2(2),PAT3(2),PAT4(2),PAT5(2,2)
C
      REAL*4 LAMDA1,LAMDA2
      DATA LAMDA1/1./,LAMDA2/1./
      DATA C1/1.0/
C
      N=2
C
       PAT3(1) = THETA(1)*S1
       PAT3(2) = THETA(2)/S2
C
C      E(1)=0.0
C      E(2)=0.0
C
       CALL VVSCAL(N,PAT3,PHI,Z1)
      CALL VECMAT(N,N,PHI,FGAIN,PAT1)
      CALL VVSCAL(N,PAT1,PHI,Z2)
      EDOT=Y-Z1
C
      ERROR= EDOT+C1*E
      ERROR= ERROR/(1.+Z2)
C
C  UPDATE THETA
C
      DO 13 I=1,N
      CALL MATVEC(N,N,FGAIN,PHI,PAT2)
 13   PAT3(I)=PAT3(I)+PAT2(I)*ERROR
C
C COMPUTE THE PRIOR ERROR TERMS
C
      CALL VVSCAL(N,PAT3,PHI,Z3)
      E=Y-Z3
C
C  UPDATE FGAIN
C
      CALL VVMAT(N,N,PAT2,PAT1,PAT5)
       DO 14 I=1,N
      DO 14 J=1,N
      FGAIN(I,J)=(FGAIN(I,J)-PAT5(I,J)/(LAMDA1/LAMDA2+Z2))/LAMDA1
 14      CONTINUE
C
C  NOW BACK TRANSFER THE VALUES
C
      THETA(1)=PAT3(1)/S1
      THETA(2)=PAT3(2)*S2
C
C Print the stuff out
C
      WRITE(IOUT,102) MSTEP,Y,ERROR,PHI(1),PHI(2)
```

```
 102   FORMAT(' M= ' ,I4,/,' Y= ',G16.8,' ERROR= ',G16.8,/,
     $ ' PHI = ',2(2X,G16.8),/)
C
       WRITE(IOUT,103) THETA(1),THETA(2)
 103   FORMAT(2X,' THETA =',2(2X,G16.8),//)
C
       RETURN
       END
```

```
       SUBROUTINE STATE1(Y,U,THETA,XBAR)
C
C This a state observer with fixed gain values
C
       COMMON/INOUT/IN,IOUT
       DIMENSION THETA(2)
C
       GAIN=-THETA(1)
C
        E=-THETA(1)
C
       F= THETA(2)
C
        ERROR=YPAST-XBAR
C
       VEC2 = E*XBAR
C
       XBAR = VEC2 + F * U + GAIN*ERROR
C
       WRITE(6,102) ERROR, XBAR
D       WRITE(7,102) ERROR, XBAR
 102   FORMAT(' ERROR=',G16.8,/,' XBAR=',2X,G15.8,/)
       YPAST=Y
C
       RETURN
       END
```

```
C       ANALOG TO DIGITAL SOURCE CODE
        INTEGER IADC,ICH,IGAIN
C
        REAL*4 Y,S1
        DATA IN/5/,IOUT/6/
        DATA Y/'Y'/
C
        CALL ASSIGN(5,'TT:')
        CALL ASSIGN(6,'TT:')
C
 155    WRITE(IOUT,110)
 110    FORMAT(3X,'WOULD YOU LIKE TO READ THE I/O PORT',/,
     $ ' REGISTER, Y/N?!',/)
        READ(IN,123) S1
 123    FORMAT(A2)
C
        IF(S1.NE.Y) GOTO 10
C
        CALL DRVRED(1,NUMBER)
        WRITE(IOUT,124) NUMBER
 124    FORMAT(3X,' THE I/O PORT NO. IS =',I6,/)
        GOTO 155
C
 10     WRITE(IOUT,111)
 111    FORMAT(3X,'>>>ANALOG RANGE:',/,
        Z3X,'1. +/-10 V',/,
        Z3X,'2. +/-5 V',/,
        Z3X,'3. +/-2.5 V',/,
        Z3X,'4. +/-1.25 V',/,
        Z3X,'>>>PLEASE ENTER 1,2,3, OR 4?')
        READ(IN,211)IC
 211    FORMAT(I2)
        GOTO(310,320,330,340),IC
        GOTO 10
 310    IGAIN=0
        GOTO 20
 320    IGAIN=4
        GOTO 20
 330    IGAIN=8
        GOTO 20
 340    IGAIN=12
C
 20     WRITE(IOUT,121)
 121    FORMAT(3X,'>>>PLEASE SPECIFY THE CHANNEL #?0-7')
        READ(IN,211)IC
        IF(IC .EQ. 0)IC=8
        GOTO(410,420,430,440,450,460,470,480),IC
        GOTO 20
 410    ICH=2**8
        GOTO 30
 420    ICH=2**9
        GOTO 30
 430    ICH=2**9+2**8
        GOTO 30
 440    ICH=2**10
        GOTO 30
 450    ICH=2**10+2**8
        GOTO 30
 460    ICH=2**10+2**9
        GOTO 30
 470    ICH=2**10+2**9+2**8
```

```
       GOTO 30
 480   ICH=2**11
C
 30       IVAL=ICH+IGAIN
 40       CALL ADC(IVAL,IADC)
       WRITE(IOUT,131)IADC
131       FORMAT(3X,'ANALOG TO DIGITAL VALUE=',I8,/,
       Z3X,'>>WOULD YOU LIKE TO:',/,
       Z3X,'     1. CALL ADC AGAIN;',/,
       Z3X,'     2. REDEFINE CHANNEL AND ANALOG RANGE;',/,
       Z3X,'     3. QUIT...?',/,
       Z3X,'     ...PLEASE ENTER 1,2,3...')
       READ(IN,211)IC
       GOTO(40,10,50),IC
50       CONTINUE
       STOP
       END
```

```
        .TITLE ADC
        .GLOBL ADC
;
;
;
ADC:    MOV     (R5)+,R0
        MOV     @(R5)+,@#170400
;
        INC     @#170400
3$:     TSTB    @#170400
        BPL     3$
        MOV     @#170402,@(R5)+
        RTS     PC
        .END
```

```
      SUBROUTINE KDADC(VOLTAG,CHANNL,GAIN)
C
C This program computes the gain for the ADC.MAC
C
       INTEGER RANGE,CHANNL,GAIN
C
      COMMON /INOUT/IN,IOUT
C
      IF(VOLTAG.EQ.1.28) IRANGE=12
      IF(VOLTAG.EQ.2.56) IRANGE=8
      IF(VOLTAG.EQ.5.12) IRANGE=4
      IF(VOLTAG.EQ.10.24) IRANGE=0
C
      IF(CHANNL.EQ.0) IC=2**11
      IF(CHANNL.EQ.1) IC=2**8
      IF(CHANNL.EQ.2) IC=2**9
      IF(CHANNL.EQ.3) IC=2**9+2**8
      IF(CHANNL.EQ.4) IC=2**10
      IF(CHANNL.EQ.5) IC=2**10+2**8
      IF(CHANNL.EQ.6) IC=2**10+2**9
      IF(CHANNL.EQ.7) IC=2**10+2**9+2**8
C
      GAIN=IC+IRANGE
C
      RETURN
      END
C
C
C
C
      SUBROUTINE FACTOR (SCALE,VOLTAG,VALUE,OUTPUT)
C
C This subroutine computes the real force
C
      INTEGER VALUE
C
D      VOLT=(FLOAT(VALUE-2048))*VOLTAG/2048
      VOLT=(FLOAT(VALUE))*VOLTAG/4095
      OUTPUT=VOLT*SCALE
C
      RETURN
      END
C
C
C
C
      SUBROUTINE SETTIM(HVAL,MINVAL,SECVAL,MILVAL)
C
C This subroutine sets the clock
C
      INTEGER HVAL,MINVAL,SECVAL,MILVAL
      INTEGER HOUR,MINUTE,SECOND,MILSEC
C
      COMMON/CLKBLK/HOUR,MINUTE,SECOND,MILSEC
C
      HOUR=HVAL
      MINUTE=MINVAL
      SECOND=SECVAL
      MILSEC=MILVAL
C
      RETURN
```

```
      END
C
C
C
C
C
      SUBROUTINE MATMUL(MA,NA,MB,NB,A,B,AB)
C
C
C  THIS SUBPROGRAM MULTIPLIES THE TWO MATRICES AND PUTS THE
C  RESULTS IN MATRIX AB
C
C
C
      DIMENSION A(MA,NA),B(MB,NB),AB(MA,NB)
C
      DO 1 I=1,MA
      DO 1 J=1,NB
      AB(I,J)=0.0
      DO 1 K=1,NA
 1    AB(I,J)=AB(I,J)+A(I,K)*B(K,J)
      RETURN
      END
C
C
C
C
      SUBROUTINE MATVEC(MA,NA,A,V,W)
C
C
C  THIS SUBPROGRAM MULTIPLIES THE MATRIX A WITH THE VECTOR
C  V AND RETURNS THE RESULT IN VECTOR W
C
C
C
      DIMENSION A(MA,NA),V(NA),W(MA)
C
      DO 2 I=1,MA
2      W(I)=0.0
      DO 3 I=1,MA
      DO 3 J=1,NA
      W(I)=W(I)+A(I,J)*V(J)
 3      CONTINUE
      RETURN
      END
C
C
C
C
C
C
      SUBROUTINE VECMAT(MA,NA,V,A,W)
C
C
C  THIS SUBPROGRAM MULTIPLIES THE VECTOR V BY MATRIX A AND
C  PUTS THE RESULT IN W
C
      DIMENSION V(MA),A(MA,NA),W(NA)
C
C
      DO 3 J=1,NA
```

```
       W(J)=0.0
       DO 3 I=1,MA
       W(J)=W(J)+V(I)*A(I,J)
 3       CONTINUE
       RETURN
       END
C
C
C
C
       SUBROUTINE VVSCAL(N,V,W,Z)
C
C
C  THIS SUBROUTINE MULTIPLIES THE TWO VECTORS AND PUTS THE
C  RESULTS IN THE SCALAR Z
C
C
       DIMENSION V(N),W(N)
C
       Z=0.0
       DO 4 I=1,N
       Z=Z+V(I)*W(I)
 4        CONTINUE
       RETURN
       END
C
C
C
C
       SUBROUTINE VVMAT(NV,NW,V,W,A)
C
C
C THIS SUBROUTINE MULTIPLIES THE TWO VECTORS V AND W
C AND PUTS THE RESULT IN MATRIX A
C
C
       DIMENSION V(NV),W(NW),A(NV,NW)
C
C
       DO 6 I=1,NV
       DO 6 J=1,NW
       A(I,J)=V(I)*W(J)
 6     CONTINUE
       RETURN
       END
C
C
C
C
       SUBROUTINE SETUP(N,OLD,FRESH,Y,U)
C
       DIMENSION OLD(N),FRESH(N)
C
       I=N
       L=N/2+1
 50    J=I-1
       FRESH(I)=OLD(J)
       IF(I.EQ.L) FRESH(I)=U
       I=I-1
       IF(I.EQ.1) GOTO 60
       GOTO 50
```

```
 60     FRESH(I)=-Y
        RETURN
        END
C
C
C
C
C
        SUBROUTINE MATINV(MA,NA,N,A)
C
C
C
C Replaces matrix A by its inverse
C
        DIMENSION A(MA,NA)
C
        DO 1 I=1,N
        DUM=A(I,I)
        DO 3 J=1,N
 3      A(I,J)=A(I,J)/DUM
        A(I,I)=1.0/DUM
        DO 4 J=1,N
        IF(I.EQ.J) GOTO 4
        DUM=A(J,I)
        DO 5 K=1,N
 5      A(J,K)=A(J,K)-A(I,K)*DUM
        A(J,I)=-A(I,I)*DUM
 4      CONTINUE
 1      CONTINUE
C
        RETURN
        END
C
C
C
C
C
C
        FUNCTION DGAMMA(N)
C
C
C  THIS FUNCTION CALCULATES THE FACTORIAL OF N
C
C
        DGAMMA=1.D0
        DO 1 I=1,N
        DGAMMA=DGAMMA*I
 1        CONTINUE
C
        RETURN
        END
C
C
C
        SUBROUTINE TRANS(MA,NA,A,AT)
C
C
C This subroutine transforms A and puts the result
C in AT
C
C
```

```
      DIMENSION A(MA,NA),AT(NA,MA)
C
      MAT=NA
      NAT=MA
C
      DO 2 I=1,MAT
      DO 2 J=1,NAT
      AT(I,J)=0.0
 2    AT(I,J)=AT(I,J)+A(J,I)
C
      RETURN
      END
```

Appendix C

This appendix presents a set of routines to plot and print data stored in a data file. These routines are based on a plotting package developed by Leal Lauderbaugh [1] for a *Digital Equipment Corporation (DEC) LSI-11/23 plus* microcomputer. The components of this package are:

DISPLY: Main program to call different subroutines to plot and print the stored data.

PLOT: Subroutine to plot the selected variable on a Tecktronix 4006-1 graghics terminal.

HRDCPY: Subroutine to generate a hard copy of the plot on an xy recorder.

PRNT: Subroutine to print out the data on either a printer or video terminal (should be modified for different data files).

CHOICE: Subrutine to select the data for plotting (should be modified for different data files).

```
C General Description:
C
C This package is used to print and plot different variables
C
C External References:
C
C
C    PLOT: routine to plot the sampled forces on the graphics terminal
C       CHOICE: routine to select the variable to be plotted
C    HRDCPY: routine to plot the sampled forces on the hardcopy unit
C    PRNT: routine to print out the sampled forces
C
       INTEGER   OPTION, POINTS
C
       DIMENSION PARAM(1000)
C
       COMMON /INOUT/  IN, IOUT
       COMMON /ACQBLK/ POINTS,PARAM
C
       DATA   IN /5/, IOUT /6/, PARAM/1000*0.0/
C
C Assign devices
C
       CALL ASSIGN(5,'TT:')
       CALL ASSIGN(6,'TT:')
C
C
       WRITE(IOUT,321)
 321   FORMAT(' Enter the no. of points',/)
       READ(IN,322) POINTS
 322   FORMAT (I3)
C
C Display the main menu.
C
 10       WRITE  ( IOUT, 20 )
   20  FORMAT ( '1', 35X, 'MAIN MENU' //////
     +  22X,'1) Plot the values on the terminal' /
     +  22X,'2) Print out the forces' /
     +  22X,'3) Quit' /////// )
C
C Prompt user for menu option.
C
       WRITE  ( IOUT, 30 )
   30  FORMAT ( '$Choose option ( 1, 2, 3 ) : ' )
       READ   ( IN, 40, ERR = 10 ) OPTION
   40  FORMAT ( I2 )
C
C Check to see that the user chose a valid option.
C
       IF ( OPTION .GT. 3 .OR. OPTION .LT. 1 ) GO TO 10
C
C Branch to carry out the option requested
C
       GO TO (1000,2000,3000) OPTION
C
C Plot the desired force.
C
 1000    CALL PLOT
       GO TO 10
C
C Print out the forces.
```

```
C
 2000     CALL PRNT
       GO TO 10
C
C Leave the program.
C
 3000     STOP
       END
```

```
      SUBROUTINE PLOT
C
C General Description:
C
C This subroutine plots out the desired parameter.
C The user chooses which parameter he wants to print out via the plot-
C ting menu.  PLTFRC uses the LEALPT plotting package. The plot is sent to
C the Tektronix 4006-1 graphics terminal.
C
      COMMON /INOUT/ IN, IOUT
C
      COMMON /MXMN/   DVMAX, DVMIN, RIDVMX, RIDVMN
C
C       DVMAX:   REAL. The maximum value of the dependent variable.
C       DVMIN:   REAL. The minimum value of the dependent variable.
C       RIDVMX:  REAL. The maximum value of the independent variable.
C       RIDVMN:  REAL. The minimum value of the independent variable.
C
      COMMON /RESBLK/ DV, RIDV, LOWPT, HIGHPT
C
C       DV:     REAL.     An array of dimension 1000.  Stores the de-
C                pendent variable array for use in the plot-
C                ting package.
C       RIDV:   REAL.     An array of dimension 1000.  Stores the in-
C                dependent variable array for use in the plot-
C                ting package.  In this subroutine, a plot-
C                ting index, from 0 up to the number of
C                points plotted, in increments of one, is
C                stored as the dependent variable.
C       LOWPT:  INTEGER. The lowest data point to plot, in terms of
C                              the numerical value of the independent
C                              variable RIDV.  LOWPT and HIGHPT are used
C                              for "windowing" operations with the plotting
C                              routines, i.e. the user can look at all of
C                              his data first, then pick a smaller portion
C                              of the independent axis to look at.
C       HIGHPT: INTEGER. The highest data point to plot, in terms of
C                              the numerical value of the independent
C                              variable RIDV.  LOWPT and HIGHPT are used
C                              for "windowing" operations with the plotting
C                              routines, i.e. the user can look at all of
C                              his data first, then pick a smaller portion
C                              of the independent axis to look at.
C
C
C External References:
C
C
C    AXES:    routine to draw the axes of the plot
C    AXTYPE:    routine to choose which axes to draw, i.e. which quadrants
C    DRAW:    routine to draw a line on the plot from the current position
C               to the coordinates specified as arguments
C    MAXMIN: routine to find the maximum and minimum values of the
C               dependent and independent variable arrays
C    MOVE:    routine to move from the current screen position to the posi-
C               tion specified in the routine arguments
C    PAGE:    routine to clear the screen of the graphics terminal
C    SCALE:    routine to calculate the scale factors and offsets for the
C               plotting routines
C
C
```

```
C
C
C Declaration statements.
C
     COMMON /ACQBLK/ POINTS,PARAM
C
     INTEGER LOWPT, HIGHPT,OPTION,POINTS
     LOGICAL*1 ANSWER,Y
C
        DATA  Y/'Y'/
C
C Dimension statements.
C
        DIMENSION  DV ( 1000 ), RIDV ( 1000 ), PARAM(1000)
C
C Initialize the plotting parameters.
C
        LOWPT  = 1
        HIGHPT = POINTS
C
C Display a header.
C
    10     WRITE(IOUT,12)
    12     FORMAT('1'/'1'/'1'/
      +  29X, 'PLOTTING PROGRAM' )
C
C Display the plotting menu.
C
         WRITE  ( IOUT, 20 )
    20    FORMAT( '1', 32X, 'PLOTTING MENU' ///
      +  12X,'1) Select a new variable to Plot on the graphics terminal',
      +  ,/,
      +  12X,'2) Repeat the previus plot',/,
      +  12X,'3) Clear the screen of the graphics terminal',/,
      +  12X,'4) Change the plotting window',/,
      +  12X,'5) Return to the main menu ' ///// )
C
C Prompt user for menu option.
C
        WRITE  ( IOUT, 30 )
    30    FORMAT ( '$Please choose option ( 1, 2, 3, 4, 5 ) : ' )
        READ   ( IN, 40, ERR = 10 ) OPTION
    40    FORMAT ( I2 )
C
C Check to see that the user chose a valid option.
C
        IF ( OPTION .GT. 5 .OR. OPTION .LT. 1 ) GO TO 10
C
C Branch to carry out the option requested
C
        GO TO ( 1000, 2000, 3000, 4000, 5000 ) OPTION
C
C Put the force numbers that the user wants to print out into a one
C dimensional array.
C
  1000     CALL CHOICE
        LOWPT=1
        HIGHPT = POINTS
C
  444   DO 100   I = LOWPT, HIGHPT
C
```

```
       DV    ( (I - LOWPT) + 1 ) = PARAM(I)
       RIDV ( (I - LOWPT) + 1 ) = (I - LOWPT)
C
  100  CONTINUE
C
C Find the maximum and minimum values of the plotting arrays DV, and RIDV
C for use in scaling the graphs.
C
 1010  CALL MAXMIN ( (HIGHPT - LOWPT), DV, RIDV )
C
C Find the axes type.
C
 1020  CALL AXTYPE ( IQUAD )
C
C Find the scale for the plot.
C
       CALL SCALE( IQUAD, DVSCL, IDVOFF, RIDVSC, IIDVOF )
C
C
C See if the scale need be changed
C
       WRITE  ( IOUT, 200 ) DVMAX, DVMIN, RIDVMX, RIDVMN
 200   FORMAT ( '1'/'1'/'1'/
     +  ' The maximum value of the dependent variable is ', G11.4, /,
     +  ' The minimum value of the dependent variable is ', G11.4, /,
     +  ' The maximum value of the indepent variable is ',G11.4,/,
     +  ' The minimum value of the indepent variable is ',G11.4,/,
     +  ,/,' Do you wish to make any changes?  Y/N ?')
C
       READ(IN,101) ANSWER
 101    FORMAT(A2)
       IF(ANSWER.EQ.Y) GOTO 329
       GOTO 333
C
 329   WRITE(IOUT,520)
 520   FORMAT(' Input the new DVMAX,DVMIN,RIDVMX,RIDVMN')
       READ(IN,530)  DVMAX,DVMIN,RIDVMX,RIDVMN
 530   FORMAT(4(G10.4))
       GOTO 1020
C
C Draw the axes.
C
 333   CALL AXES ( IQUAD )
C
C Plot the forces.
C
 555    DO 400  K = LOWPT, HIGHPT
C
       IX = IFIX ( RIDVSC * RIDV ( (K - LOWPT) + 1 ) ) + IIDVOF
       IY = IFIX ( DVSCL  * DV   ( (K - LOWPT) + 1 ) ) + IDVOFF
C
       IF ( K .NE. LOWPT ) GO TO 390
       CALL MOVE ( IX, IY )
       GO TO 400
C
  390    CALL DRAW ( IX, IY )
C
  400    CONTINUE
C
 4099    WRITE  ( IOUT, 4100 )
 4100    FORMAT ( /'1'/'1'/'1'/ '$Press RETURN to continue: ' )
```

```
      READ   ( IN, 4200 ) ANSWER
 4200 FORMAT ( A1 )
C
C See if a hardcopy of the plot is needed.
C
      WRITE (IOUT,203)
 203  FORMAT( /'1'/'1'/'1'/ ' Would you like to get a hardcopy of'
     $ ' this plot ?! Y/N?!',/)
      READ (IN,101) ANSWER
      IF(ANSWER.EQ.Y) CALL HRDCPY
      GO TO 10
C
C Repeat the plot
C
 2000  GOTO 333
C
C Clear the plotting screen.
C
 3000  CALL PAGE
      GO TO 10
C
C Prompt the user for new values of LOWPT and HIGHPT,
C which define the plotting window.
C
 4000  WRITE  ( IOUT, 5100 ) POINTS, LOWPT, HIGHPT
 5100  FORMAT ( '1'/'1'/'1'/
     +  ' The original data sample ranges from point 1 to point ',
     +  I5, '.' //
     +  ' The plotting window currently extends from data',
     +  ' point ', I5, ' to data point ', I5, '.' /'1'/'1'///
     +    '$Please enter the lowest data point to plot: ' )
C
      READ   ( IN, 5200 ) LOWPT
 5200 FORMAT ( I5 )
C
      WRITE  ( IOUT, 5300 )
 5300 FORMAT ( // '$Please enter the highest data point to plot: ' )
C
      READ   ( IN, 5200 ) HIGHPT
C
C Replot the variable
C
      GO TO 444
C
C Return to the calling program.
C
 5000   RETURN
      END
```

```
      SUBROUTINE HRDCPY
C
C General Description:
C
C This subroutine gives a hard copy of the plot generated by subroutine PLOT
C HRDCPY uses the LEALPT plotting package.  The plot is sent to the
C XY Recorder ( Hardcopy unit ).
C
      COMMON /INOUT/ IN, IOUT
      COMMON /MXMN/   DVMAX, DVMIN, RIDVMX, RIDVMN
      COMMON /RESBLK/ DV, RIDV, LOWPT, HIGHPT
C
       INTEGER LOWPT,HIGHPT
C
      DIMENSION DV(1000), RIDV(1000), PARAM(1000)
C
      COMMON /ACQBLK/ POINTS,PARAM
C
C Initialize the hardcopy unit.
C
 50   CALL PLTINT
C
C Determine the axis type
C
      CALL AXTYPE (IQUAD)
C
C Find the scale for the plot
C
      CALL SCALE (IQUAD,DVSCL,IDVOFF,RIDVSC,IIDVOF)
C
C Draw the axes.
C
      CALL PAXES ( IQUAD )
C
C Plot the forces.
C
      DO 400  K = LOWPT, HIGHPT
C
      IX = IFIX ( RIDVSC * RIDV ( (K - LOWPT) + 1 ) ) + IIDVOF
      IY = IFIX ( DVSCL  * DV   ( (K - LOWPT) + 1 ) ) + IDVOFF
C
      CALL PLOTER ( IX, IY )
C
 400  CONTINUE
C
      CALL PMOVER(0,0)
C
C Print out the maximum and the minimum values of the force.
C
      WRITE  ( IOUT, 200 ) DVMAX, DVMIN
 200  FORMAT ( '1'/'1'/'1'/
     + ' The maximum value of the variable is ', G11.4, /,
     + ' The minimum value of the variable is ', G11.4,  )
C
C Pause for a minute to allow the user to observe the maximum and minimum
C values of the chosen force.
C
      WRITE  ( IOUT, 4100 )
 4100 FORMAT ( /'1' /'1' / '$Press RETURN to continue: ' )
      READ   ( IN, 4200 ) ANSWER
 4200 FORMAT ( A1 )
```

```
C
C Return to the calling program.
C
        RETURN
       END
C
C
```

```
      SUBROUTINE PRNT
C
C General Description:
C
C This routine is used to print out the data from a data file.
C The  user can choose whether he wants the data sent to the video terminal
C or the line printer.
C
      INTEGER   POINTS,OPTION
      LOGICAL*1 ANSWER
C
      COMMON /INOUT/  IN, IOUT
      COMMON /ACQBLK/ POINTS,PARAM
C
      DIMENSION PARAM(1000)
C
      CALL ASSIGN(1,'KDX2.DAT')
      CALL ASSIGN(2,'KDY2.DAT')
      CALL ASSIGN(3,'KDZ2.DAT')
      CALL ASSIGN(7,'LP:')
C
C Display a header.
C
 10       REWIND 1
      REWIND 2
      REWIND 3
C
         WRITE(IOUT,12)
   12    FORMAT('1'/'1'/'1'/
     +  29X,'PLOTTING PROGRAM' )
C
C Display the printing menu.
C
         WRITE  ( IOUT, 20 )
   20    FORMAT( '1'/
     +  33X, 'PRINTING MENU' ///
     +    22X,'1) Send values to the printer' /
     +  22X,'2) Send values to the screen'  /
     +  22X,'3) Return to the main menu ' //////// )
C
C Prompt user for menu option.
C
      WRITE  ( IOUT, 30 )
   30   FORMAT ( '$Please choose option ( 1, 2, 3 ) : ' )
      READ   ( IN, 32, ERR = 10 ) OPTION
   32   FORMAT ( I2 )
C
C Check to see that the user chose a valid option.
C
      IF ( OPTION .GT. 3 .OR. OPTION .LT. 1 ) GO TO 10
C
C Branch to carry out the option requested
C
      GO TO ( 40, 50, 1000 ) OPTION
C
C Since the user wants to send the force numbers to the printer,
C reassign the output device.
C
  40   IOUT = 7
C
C Print a heading on the top of the page.
```

```
C
   50     WRITE  ( IOUT, 100 ) POINTS
  100     FORMAT ( '1'/ '1'/ '1'/
     +     17X, 'DATA ACQUISTION RESULTS' ///
     +   ' Number of samples: ', I3, ' (unitless)'
     +   ' Force values given in Newtons' ///
     +   ' SAMPLE  #', 10X, 'FX', 15X, 'FY', 15X, 'FZ', /// )
C
C Loop to print out the results.
C
       DO 3000  J = 1, POINTS
C
C Print out the results.
C
       READ (1) M,FX
       READ (2) M,FY
       READ (3) M,FZ
C
       WRITE  ( IOUT, 2000 ) M, FX, FY, FZ
 2000    FORMAT ( 2X, I3, 3 ( 2X, G16.4 ) )
C
C Continue the looping.
C
 3000    CONTINUE
C
C If the force values were sent to the screen, pause a moment to allow the
C user to view the last of the values printed.
C
       IF ( OPTION .NE. 2 ) GO TO 600
       WRITE  ( IOUT, 5910 )
 5910    FORMAT ( / / / '$Press RETURN to continue: ' )
       READ   ( IN, 5920 ) ANSWER
 5920   FORMAT ( A1 )
C
C Reassign the output device to be the screen just in case it was changed
C to the line printer above.
C
  600    IOUT = 6
C
C Return to the printing menu.
C
       GO TO 10
C
C Return to the calling program.
C
 1000    CLOSE(UNIT=1)
         CLOSE(UNIT=2)
         CLOSE(UNIT=3)
C
         RETURN
         END
```

```
      SUBROUTINE CHOICE
C
C General Description:
C
C This routine selects the appropriate variable for plotting.
C
      CALL ASSIGN(1,'KDX2.DAT')
      CALL ASSIGN(2,'KDY2.DAT')
      CALL ASSIGN(3,'KDZ2.DAT')
C
      INTEGER   POINTS,OPTION
      LOGICAL*1 ANSWER
C
      COMMON /INOUT/  IN, IOUT
      COMMON /ACQBLK/ POINTS,PARAM
C
      DIMENSION PARAM(1000)
C
C Display the selection menu.
C
   10    WRITE  ( IOUT, 20 )
   20    FORMAT( '1'/
     +  33X, 'SELECTION MENU' ///
     +    22X,'1) Select FX' /
     +  22X,'2) Select Fy'  /
     +  22X,'3) Select Fz' //////// )
C
C Prompt user for menu option.
C
      WRITE  ( IOUT, 30 )
   30 FORMAT ( '$Please choose option ( 1, 2, 3 ) : ' )
      READ   ( IN, 32, ERR = 10 ) OPTION
   32 FORMAT ( I2 )
C
C Check to see that the user chose a valid option.
C
      IF ( OPTION .GT. 3 .OR. OPTION .LT. 1 ) GO TO 10
C
C Branch to carry out the option requested
C
      GO TO ( 40, 50, 60 ) OPTION
C
C Since the user wants to send the force numbers to the printer,
C reassign the output device.
C
 40      REWIND 1
      DO 21 J = 1, POINTS
       READ (1) MSTEP, FX
D      FX = FX + 780.65
      IF(MSTEP.GE.17) FX = FX + 21.88
      IF(MSTEP.GE.33) FX = FX + 55.88
      IF(MSTEP.GE.50) FX = FX + 21.49
      IF(MSTEP.GE.63) FX = FX + 13.28
      IF(MSTEP.GE.73) FX = FX + 23.05
C
      PARAM(J) = FX
 21      CONTINUE
      GOTO 1000
C
 50      REWIND 2
      DO 22 J = 1, POINTS
```

```
        READ (2) MSTEP, FY
D       FY = FY + 217.24
       IF(MSTEP.GE.17) FY = FY - 1.96
       IF(MSTEP.GE.33) FY = FY + 37.51
       IF(MSTEP.GE.49) FY = FY + 17.97
       IF(MSTEP.GE.62) FY = FY + 18.37
       IF(MSTEP.GE.72) FY = FY + 17.97
C
       PARAM(J) = FY
 22       CONTINUE
       GOTO 2000
C
 60       REWIND 3
       DO 23 J = 1, POINTS
        READ (3) MSTEP, FZ
D       FZ = FZ + 343.83
       IF(MSTEP.GE.17) FZ = FZ - 8.99
       IF(MSTEP.GE.33) FZ = FZ + 58.60
       IF(MSTEP.GE.49) FZ = FZ + 22.27
       IF(MSTEP.GE.62) FZ = FZ + 12.90
       IF(MSTEP.GE.72) FZ = FZ + 19.93
C
       PARAM(J) = FZ
 23       CONTINUE
       GOTO 3000
C
 1000  CLOSE(UNIT=1)
       GOTO 5000
C
 2000  CLOSE(UNIT=2)
       GOTO 5000
C
 3000  CLOSE(UNIT=3)
       GOTO 5000
C
 5000  RETURN
       END
```

Bibliography

[1] Lauderbaugh, L. K., and Ulsoy, A. G., "SIMULA – Digital Controller Simulation Package," Technical Report No. UM-MEAM-84-22, Department of Mechanical Emgineering and Applied Mechanics, University of Michigan, Ann Arbor, Michigan.

[2] K. Danai and A. G. Ulsoy 1985 in *Sensors and Controls for Manufacturing* 137-148. A Dynamic State Model for On-Line Tool Wear Estimation in Turning.

[3] Takahashi, Y., Rabins, M. J., and Auslander, D. M., *Control*, Addison-Wesley Publishing Co., Inc., Reading, Mass. 1972.

[4] G. C. Goodwin and K. S. Sin 1984 *Adaptive Filtering, Prediction, and Control* Englwood Cliffs, New Jersey: Prentice-Hall.

[5] Rasmussen, F., Lauderbaugh, L. K., and Ulsoy, A. G., "RELTIM – A Library of Routines Used in Real Time Machine Control Applications," Technical Report No. UM-MEAM-84-24, Department of Mechanical Emgineering and Applied Mechanics, University of Michigan, Ann Arbor, Michigan.