HEATING PARAMETER ESTIMATION USING COAXIAL THERMOCOUPLE GAGES IN WIND TUN. (U) AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOOL OF ENGI... N T CAHOON UNCLASSIFIED DEC 84 AFIT/GAE/AA/84D-3 F/G 9/2
HEATING PARAMETER ESTIMATION USING COAXIAL THERMOCOUPLE GAGES IN WIND TUNNEL TEST ARTICLES

THESIS

Neil T. Cahoon
Captain, USAF

AFIT/GAE/AA/84D-3

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY
AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio
HEATING PARAMETER ESTIMATION USING COAXIAL THERMOCOUPLE GAGES IN WIND TUNNEL TEST ARTICLES

THESIS

Neil T. Cahoon
Captain, USAF

Approved for public release; distribution unlimited
HEATING PARAMETER ESTIMATION USING COAXIAL THERMOCOUPLE GAUGES IN WIND TUNNEL TEST ARTICLES

THESIS

Presented to the Faculty of the School of Engineering of the Air Force Institute of Technology Air University In Partial Fulfillment of the Requirements for the Degree of Master of Science in Aeronautical Engineering

Neil T. Cahoon, B.S.E.
Captain, USAF

December 1984

Approved for public release; distribution unlimited
Acknowledgements

I wish to express my sincere thanks and gratitude to my thesis advisor, Major James K. Hodge for his expert guidance during the course of this study. His ability to provide engineering insight ("voodoo engineering") and open the doors of understanding never ceased. I would also like to say thanks for the encouragement when the obstacles appeared overwhelming.

Also to my wife, Shirley, for the sacrifices she made during the course of this academic experience, I would like to say thanks. She provided immeasurable support and encouragement during my long hours of study.
# Table of Contents

Acknowledgments ........................................... ii
List of Figures ........................................ iv
List of Symbols ........................................ v
Abstract ................................................ vii

I. Introduction .......................................... 1
   1.1 Background ..................................... 1
   1.2 Objectives .................................... 5
   1.3 Overview ..................................... 6

II. The Thermal Model .................................. 7
   2.1 Temperature Equation ......................... 7
   2.2 Sensitivity Equations ....................... 11
   2.3 Covariance Equations ....................... 12

III. HEATEST Overview .................................. 14

IV. Results ............................................ 20
   4.1 Test Procedure ................................ 20
   4.2 Test Cases ................................... 22

V. Conclusions and Recommendations ................... 44
   5.1 Conclusions .................................. 44
   5.2 Recommendations ............................. 45

Appendix A: One-dimensional Energy Balance .......... 46
Appendix B: Sensitivity Equations .................... 52
Appendix C: HEATEST .................................. 57
Bibliography ........................................... 58
Vita .................................................... 59
# List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1 Coaxial Thermocouple Gage</td>
<td>4</td>
</tr>
<tr>
<td>3.1 HEATEST Algorithm Summary</td>
<td>15</td>
</tr>
<tr>
<td>4.1 Input Data</td>
<td>27</td>
</tr>
<tr>
<td>4.2 Discrete Data</td>
<td>28</td>
</tr>
<tr>
<td>4.3 Temp. vs Node Point, 1 iteration</td>
<td>29</td>
</tr>
<tr>
<td>4.4 Temp. vs Node Point, converged</td>
<td>30</td>
</tr>
<tr>
<td>4.5 Temp. vs Node Point, At=.5 sec</td>
<td>31</td>
</tr>
<tr>
<td>4.6 Temp. vs Node Point, At=.25 sec</td>
<td>32</td>
</tr>
<tr>
<td>4.7 Temp. vs Node Point, At=1 and .25 sec</td>
<td>33</td>
</tr>
<tr>
<td>4.8 Temp. vs Node Point, T/C length=.1 ft.</td>
<td>34</td>
</tr>
<tr>
<td>4.9 Temp. vs Node Point, T/C length=.05 ft.</td>
<td>35</td>
</tr>
<tr>
<td>4.10 Temp. vs Node Point, T/C length=.025 ft.</td>
<td>36</td>
</tr>
<tr>
<td>4.11 Temp. vs Position, T/C length=.1,.05,.025 ft.</td>
<td>37</td>
</tr>
<tr>
<td>4.12 Temp. vs Node Point, 10 sec, 6 nodes</td>
<td>38</td>
</tr>
<tr>
<td>4.13 Temp. vs Node Point, 3 sec, 6 nodes</td>
<td>39</td>
</tr>
<tr>
<td>4.14 Temp. vs Node Point, 10 sec, 12 nodes</td>
<td>40</td>
</tr>
<tr>
<td>4.15 Temp. vs Node Point, 3 sec, 12 nodes</td>
<td>41</td>
</tr>
<tr>
<td>4.16 Temp. vs Position, 6 vs 12 nodes, 10 sec.</td>
<td>42</td>
</tr>
<tr>
<td>4.17 Temp. vs Position, 6 vs 12 nodes, 3 sec.</td>
<td>43</td>
</tr>
</tbody>
</table>
List of Symbols

A, A', b, d  Coefficient Matrices (i x i)
c  Specific Heat
E  Residual Error Vector (m)
G  Kalman Gain Vector (i)
H  Thermocouple Location Matrix (m x i)
h_0  Magnitude of Heat Transfer Coefficient Ratio
h_s  Heat Transfer Coefficient Derivative
h_bar  Heat Transfer Coefficient Ratio
h_ref  Reference Heat Transfer Coefficient at Zero State
I  Identity Matrix
J_k  Conditional Information Matrix
k  Thermal Conductivity
L  Total Number of Spatial Node Points
P  Covariance Matrix (i x i)
Q  Model Error Covariance Matrix (i x i)
q  Heating Rate
R_m  Covariance for mth measurement
S  Score Vector (k)
S_i,k  Sensitivity Vectors (i) for the kth Parameter
T_ aw  Adiabatic Wall Temperature
t  Time
U  Temperature Vector (i)
<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>Spatial Coordinate</td>
</tr>
<tr>
<td>Y</td>
<td>Thermocouple Measurement Vector</td>
</tr>
<tr>
<td>α</td>
<td>Angle of Attack</td>
</tr>
<tr>
<td>ε</td>
<td>Emissivity</td>
</tr>
<tr>
<td>Θ</td>
<td>Parameter Vector</td>
</tr>
<tr>
<td>( \mu_n )</td>
<td>Measurement Vector at nth Time Point</td>
</tr>
<tr>
<td>( p )</td>
<td>Density</td>
</tr>
<tr>
<td>( σ )</td>
<td>Stefan-Boltzmann Constant</td>
</tr>
<tr>
<td>( β )</td>
<td>Transition Matrix</td>
</tr>
<tr>
<td>( β_c )</td>
<td>Scaling Parameter for Specific Heat</td>
</tr>
<tr>
<td>( β_k )</td>
<td>Scaling Parameter for Thermal Conductivity</td>
</tr>
</tbody>
</table>

**Superscripts**

- \( \) - a priori Propogation
- \( + \) - a posteriori Propogation
- \( * \) - Parameter Estimate
- \( n \) - Time Level
- \( s \) - Iteration Level
- \( T \) - Transpose

**Subscripts**

- \( i \) - Spatial Node Point
- \( k \) - Number of Model Parameter
- \( m \) - Number of Thermocouples
- \( o \) - Freestream Conditions
Abstract

A heat energy balance is applied to a coaxial thermocouple gage for parameter estimation in wind tunnel test articles. This method can significantly reduce wind tunnel test costs and time. Modifications to the data reduction technique HEATEST (HEATing ESTimation) are made. The program allows for transient test techniques to be used as well as assuming an isothermal wall. A non-linear convective heat transfer coefficient model may also be used. Data is generated to test the new program. Temperature profiles throughout the thermocouple gage were good and were compared with changes in time step, thermocouple length, and number of discrete node points. The estimation of the convective heat transfer coefficient and thermal conductivity were excellent.
I. INTRODUCTION

1.1 Background

The determination of heat transfer rates on hypersonic configurations upon reentry is important for the survival of the vehicle. The problem is that the heat rate is not a quantity which may be directly scaled from model tests in wind tunnels. However, the parameters which make up the heat rate equation (thermal conductivity and specific heat, for example) can be scaled from which the heat rate may then be calculated. Wind tunnel heat transfer measurements have traditionally used a thin walled test model fabricated with thermocouples mounted on the inside skin surface. The "thermal model" then yields a heat rate based on temperature measurement from the thermocouple. Another technique uses a coaxial thermocouple gage mounted in a thick skin model. A discussion of the two methods follows.

The Traditional Thin Skin Model

The "thermal model" of a traditional wind tunnel thin
skin model assumes that all of the heat penetrates the thin skin via conduction to a standard thermocouple gage mounted on the back face. No lateral conduction is assumed and since the emittance of the steel model is low, radiation is assumed negligible. A typical heat rate measurement data point is acquired by injecting the cooled model into the wind tunnel at a known temperature and time and by measuring the temperature at later times. The model is then removed from the tunnel, cooled, and a change in configuration is made in preparation for the next injection and subsequent data point. There are three very severe limitations associated with this technique (Ref 1). The first is the inability to acquire more than one data point during any one injection. The thermal model simply does not allow for the type of change in configuration or model attitude which can be accomplished using dynamic testing techniques (to be discussed later). Associated with this limitation is the long cooling time between each test which significantly increases the cost per data point for the overall test. A second limitation is the special thin skin model which must be fabricated, further contributing to increased test cost. Finally, the assumption of no lateral conduction through the model may in fact be a poor assumption at some critical locations with large curvature. An alternate type of gage, the coaxial thermocouple, can eliminate these limitations with an overall effect of reducing time and cost.
A New Application For An Old Thermocouple

The coaxial thermocouple gage is shown in Fig. 1.1. It consists of a constantan (a metal alloy) jacket surrounding a chromel core with a thin layer of insulation separating the two metals. The coaxial gage is mounted in a steel model thick enough so that the thermal pulse is not sensed on the backface (ie. the model wall is considered a semi-infinite slab). The thermocouple surface is formed when the gage is lightly sanded to match the contour of the model. Some gages are available with backface temperature monitoring to assure that the thermal pulse does not reach the backface in any given run so that an analytical integration of the heat equation can be used to determine the heating rate history. The backface temperature information prior to this investigation is not factored into the data reduction process, however.

Operation of the coax gage is based on uniform conduction along the gage length which would necessitate the model be made of a material with similar thermal properties (Ref 3). The thermal properties of stainless steel match very closely with the gage properties, therefore, the presence of the gage is negligible. The matching of thermal properties also enhances accuracy. A coaxial gage which is matched thermally with the model allows an isothermal wall assumption, whereas other gauges such as calorimeters and thin film gages are not thermally matched, and cause a non-isothermal wall. Measured heat transfer can be in error by
\[
\begin{align*}
\mathbf{U} & = [A] \mathbf{U} + \mathbf{b} + \mathbf{W}(t) \\
\mathbf{S}_k & = [A] \mathbf{S}_k + [d] \\
\end{align*}
\] (3-1)

These equations are solved using a tridiagonal algorithm in subroutine TPS3 for the temperature states, and subroutine SENS for the sensitivity of the temperature to the kth parameter. Propagation of the covariance, \(P\), of the temperature state at each node is accomplished by the approximate difference equation,

\[
P(t_n^-) = \phi(\Delta t) P(t_{n-1}^+) \phi^T(\Delta t) + \int_{t_{n-1}}^{t_n} \phi(t_n - \lambda) Q \phi^T(t_n - \lambda) d\lambda
\] (3-2)

where \(\phi\) is the transition matrix and where the - and + superscripts are used to denote the expected values before an update (or a priori) and updated (or a posteriori) values, respectively. This calculation is made in subroutine TPSOSP2.

A model of the temperature measurement process must be used for the Kalman filter equations. The measurement equation to identify thermocouple location is,

\[
Y(t_n) = H (U(t_n)) + \{\mu_n\}
\] (3-3)

The updated temperature is calculated by,

\[
U(t_n^+) = U(t_n^-) + G E(t_n)
\] (3-4)
time. They are found by employing a Kalman filter—a set of recursive equations that optimally combine the propagation of the model equations with measurement updates at each sample time. After the entire temperature-time (state) history has been calculated, a gradient algorithm is used to solve for best estimates of the parameters according to a maximum likelihood criterion.

The second type of estimate consists of the parameters defined in the parameter vector, \( \Theta \), as given in Equation 2-10. These parameters remain essentially constant throughout a transient maneuver profile such as a pitch sweep and are estimated based on data from the entire maneuver history. The process of estimating states and parameters is then iterated for convergence to some optimal estimate.

The method used to estimate the states and parameters is formulated from stochastic estimation theory and is known as adaptive estimation. A detailed development of the estimation equations is beyond the scope of this thesis and the reader is referred to References 7 and 8 for more detail.

The thermal model equations for temperature and sensitivity have already been written in the matrix stochastic estimation form of Equation 2-12 as,
Start

Initial Conditions

Node Point Structure

Read 1st Test Section Sample

Calculate Ref Hbar

HEATTUN

Read 1st T/C Sample

Model Error Covariance Matrix QUE

next sample

Propogate the Test Section sample

TPS3

Solve for U

TPS3

SENS

Solve for S_i,k

SENS

TPSOSP2 'a priori' Covariance 'a posteriori' TPSOSP2

Read next Test Section sample

HEATTUN

Solve for Hbar

HEATTUN

KALMAN FILTER

IC SMOOTHER

time loop

T < TSTOP?

TPS3

PARAMETER UPDATES

WRITE

iteration loop

PLOT

STOP

Figure 3.1 HEATEST Algorithm Summary
III. HEATEST OVERVIEW

The HEATEST program was originally developed to determine and model heat rates from the Space Shuttle Orbiter thermocouple data, hence most of its nomenclature references flight data samples and trajectory samples. The wind tunnel equivalence of the trajectory sample would be the test section conditions at the time of the sample (i.e., density, velocity, pressure, etc.). The flight data are the thermocouple measurements from the coaxial gages.

An algorithm summary of the HEATEST program is given in Figure 3.1. The initial conditions for the temperature profile, \( U(t) \), and the initial covariance, \( P(t) \), are specified at the start of the wind tunnel test run. Heating model initial parameters, and the initial reference values for the heating model are read in as inputs to the program. Initial sensitivities of the state are specified to be zero. The node point structure throughout the depth of the thermocouple is then calculated from the input of the length of the gage and the number of node points.

Two types of estimates must be made in order to describe the thermodynamic environment in the wind tunnel. The first type are the state estimates, which are defined by each node temperature. These state estimates are not constant since the temperature varies throughout the maneuver, and hence, must be estimated at each node point in
The \( \{ b \} \) vector is not used directly for the covariance equation, but is approximated by an error model given by,

\[
Q_{\text{error}} = [h_{\text{bar}} \cdot h_{\text{ref}}(T_{aw} - U_1) \Delta x / \sigma_k]^2
\]  \hspace{1cm} (2-15)
where $S$ is the sensitivity. The $i$ subscript identifies the node point and the second subscript identifies the particular parameter number. The sensitivity equations may also be written in the familiar form,

$$[A'][S_{i,k}] + [d] = 0 \quad (2-12)$$

The sensitivity equations are developed and summarized in Appendix B.

2.3 Covariance Equation

Propagation of the covariance of the temperature state at each node requires the equations to be of the form

$$\{\dot{U}\} = [A]\{U\} + \{b\}$$

and $$\{S_{i,k}\} = [A]\{S_k\} + \{d_k\} \quad (2-13)$$

Substituting the definitions of Equations (A-14) into Equations (2-7) and (2-8) and rearranging yields a common tridiagonal $[A]$ matrix for the above equations which is shown presently,

$$[A] = \begin{bmatrix}
-RM_i & \frac{RP_i}{RCX_i} & 0 \\
\frac{RM_i}{RCX_i} & -\left(\frac{RM_i + RP_i}{RCX_i}\right) & \frac{RP_i}{RCX_i} \\
0 & \frac{RP_i}{RCX_i} & -\left(\frac{RM_i + RP_i}{RCX_i}\right)
\end{bmatrix} \quad (2-14)$$
\[ [A'] [U_i^n] + [b] = 0 \] (2-9)

where \([A']\) is an \(n \times n\) tridiagonal matrix of material properties and \([U_i^n]\) is the \(n\)-dimensional column vector of unknown temperature at each node point for each time.

In general, the unknown parameters in this model formulation are the heat transfer coefficient intercept, \(h_0\), the slopes \(h_{a1}\) and \(h_{a2}\), and the scaling parameters for specific heat and thermal conductivity, \(\phi_c\) and \(\phi_k\), respectively. These parameters may be defined as a vector, \(\theta\), of unknown parameters for use in the system identification scheme as,

\[ \theta = \{h_0, h_{a1}, h_{a2}, \phi_c, \phi_k\}^T \] (2-10)

The primary purpose of the heating estimation program is to obtain best estimates of these parameters during transient test maneuvers. To estimate these parameters it is necessary to calculate the model sensitivity to each unknown parameter.

2.2 Sensitivity Equations

The derivative of Equation (2-7) with respect to each parameter yields equations of the same form as Equation (2-9) from which the HEATEST program propagates the sensitivity. For example, the sensitivity of the temperature with respect to \(h_0\) would be written as follows,

\[ \frac{\partial U}{\partial \theta_1} = \frac{\partial U}{\partial h_0} = \frac{S_{h0}}{S_{i,1}} \] (2-11)
level defined by the superscript \( n \),

\[
(U_{1}^{n,s+1})^{4} = (U_{1}^{n,s})^{4} + 4(U_{1}^{n,s})^{3}(U_{1}^{n,s+1} - U_{1}^{n,s})
\]

\[
= -3(U_{1}^{n,s})^{4} + 4(U_{1}^{n,s})^{3}U_{1}^{n,s+1}
\]

(2-6)

Substituting Equation (2-6) into (2-5) yields,

\[
\rho \phi_{c}c^\Delta x \frac{U_{1}^{n}-U_{1}^{n-1}}{2\Delta t} = \frac{\phi_{k}k_{1+1/2}}{\Delta x} U_{1}^{n} + \frac{\phi_{k}k_{1+1/2}}{\Delta x} U_{2}^{n}
\]

\[- \sigma[-3(U_{1}^{n,s})^{4} + 4(U_{1}^{n,s})^{3}U_{1}^{n,s+1} - (U_{0}^{n})^{4}]
\]

\[+ [h_{0} + h_{a1}(\alpha - \alpha_{1}) + h_{a2}(\alpha - \alpha_{2})] h_{ref} (T_{aw} - U_{1}^{n,s+1})
\]

(2-7)

The model equation for the interior node points, \((i=2,imax)\), yields,

\[
\rho \phi_{c}c^\Delta x \frac{U_{i}^{n}-U_{i}^{n-1}}{\Delta t} = \frac{\phi_{k}k_{i-1/2}}{\Delta x} U_{i-1}
\]

\[- \frac{\phi_{k}(k_{i-1/2} + k_{i+1/2})}{\Delta x} U_{i}
\]

\[+ \frac{\phi_{k}k_{i+1/2}}{\Delta x} U_{i+1}
\]

(2-8)

Equations (2-7) and (2-8) can be rearranged into the familiar matrix form.
parameters other than those included in the reference heat transfer coefficient are summarized by the static transfer relation or heat transfer coefficient ratio, \( h_{\text{bar}} = \frac{h}{h_{\text{ref}}} \). Here, the ratio is assumed to be piecewise linear with respect to angle of attack as derived from Lagrange Interpolation Theory (Ref 6 and Ref therin).

\[
h_{\text{bar}} = [h_0 + h_{\alpha 1}(\alpha - \alpha_1) + h_{\alpha 2}(\alpha - \alpha_2)] \quad (2-4)
\]

where \( h_0 \) is the magnitude of the heat transfer coefficient, \( h \) at the reference conditions \( \alpha_1 \), specified by the one subscript. The heating parameters \( h_0 \), \( h_{\alpha 1} \), and \( h_{\alpha 2} \) are considered to be unknown and constant over a prescribed time period, and will be estimated by the HEATEST program. The parameters correspond to a derivative with respect to deflection angle of the model. Thus, for constant step size, the model equation at the surface node, \( i = 1 \), becomes,

\[
\frac{\rho \phi_c c \Delta x}{2} \frac{U_1^n - U_1^{n-1}}{\Delta t} = \frac{-k k_1^{1+1/2}}{\Delta x} U_1^n + \frac{k_1^{1+1/2}}{\Delta x} U_2^n - \sigma[(U_1^n)^4 - (U_0^n)^4]
\]

\[
+ [h_0 + h_{\alpha 1}(\alpha - \alpha_1) + h_{\alpha 2}(\alpha - \alpha_2)] h_{\text{ref}}(T_{aw} - U_1^n) \quad (2-5)
\]

The non-linear radiation term is quasi-linearized on an iteration level defined by the superscript \( s \) and by the time
where
\( \varepsilon \) radiative emissivity
\( \sigma \) Stefan-Boltzmann constant
\( c \) material Specific Heat
\( \rho \) material density
\( k \) Thermal Conductivity
\( \phi_c \) Specific Heat scaling parameter
\( \phi_k \) Thermal Conductivity scaling parameter

The material specific heat and thermal conductivity are both scaled by the two factors \( \phi_c \) and \( \phi_k \), respectively, hence the value for \( c \) and \( k \) will remain unchanged. The parameters \( \phi_c \) and \( \phi_k \) will be estimated by the HEATEST program. Coefficients with subscripts which are less than one or greater than \( n \) are zero. The radiation and heat rate terms are also zero except at the surface node. Equation 2-1 includes terms due to conduction from adjacent node points \( k_{i-1/2}/\Delta x_{i-1} \), surface radiation \( \varepsilon_0 u_i^4 \), and the convective transfer of energy as obtained from the heating model. The resulting system of implicit difference equations must be solved simultaneously.

The heating model for the convective transfer of energy is based on Newton's Law of Cooling.

\[ q = h(T_{aw} - T) \quad (2-2) \]

Non-dimensionalizing by a reference heat transfer coefficient, \( h_{\text{ref}} \) yields,

\[ q = h_{\text{bar}} h_{\text{ref}} (T_{aw} - U_i) \quad (2-3) \]

The dependence of the heat transfer coefficient on
II. THE THERMAL MODEL

2.1 Temperature Equations

A cross section of the one-dimensional model is given in Figure 1.1 and below as a typical coaxial thermocouple gage.

\[ q_{\text{rad}} \]

\[ q_{\text{conv}} \]

An energy balance is performed on each element. The thermal conductivity, \( k \), is taken as an average between each node. Fourier's Law of Heat Conduction throughout the gage, the Stefan-Boltzmann Law for radiation and Newton's Law of Cooling for convection on the surface face yield a system of nonlinear differential equations of the form:

\[
\left[ \left( \rho_i \delta \varepsilon C_i \Delta x_i + \rho_{i-1} \delta \varepsilon C_i-1 \Delta x_i-1 \right) / 2 \right] \left[ (U_i^n - U_{i-1}^{n-1}) \Delta t \right] \\
= \delta_k \left[ k_i-1/2 / \Delta x_i-1 \right] U_{i-1}^n \\
- \delta_k \left[ k_{i-1/2} / \Delta x_{i-1} + k_{i+1/2} / \Delta x_i+1 \right] U_i^n \\
+ \delta_k \left[ k_{i+1/2} / \Delta x_{i+1} \right] U_{i+1}^n \\
- \sigma (U_i^4 - U_{i-1}^4) + h_{\text{bar}} k_{\text{ref}} (T_{aw} - U_i) \]  \hspace{1cm} (2-1)
gage using a Kalman filter (Ref 2). The purpose of this investigation is to incorporate the appropriate thermal model equations into the HEATEST program for application to coaxial thermocouple gages as used in the wind tunnel. Two reasons for doing this are, 1) replace the analytical with the semi-infinite assumption, i.e. extend the run time or shorten the gage, and, 2) estimate thermal properties. Testing and verification of the modified program is necessary for verification of the validity of the results. Simulated data are generated by an analytical solution, and are processed for testing purposes for which the results are known.

1.3 Overview

A development of the pertinent temperature, sensitivity, and covariance equations will be developed for introduction into the HEATEST program in Chapter II. Details in format for programming may be found in Appendix A & B. Chapter III is an overview of the HEATEST algorithm and shows how the equations developed in Chapter II are utilized. Chapter IV outlines the method for testing the program and offers a discussion of the test cases made and results. Finally, conclusions about the validity of the modifications, and suggestions for further improvement are included in Chapter V.
up to 40\% because of the non-isothermal (Ref 2). The same rugged model built for pressure measurements can be used for temperature measurements as well, which would further reduce the wind tunnel costs. The data reduction technique, which uses the temperature time history, eliminates the requirement to fix the model configuration or attitude during any one run, hence dynamic testing techniques may be used similar to the flight test technique used for the Space Shuttle Orbiter (Ref 4). The model may be swept in angle of attack, for example, to determine heat rates as a function of angle of attack. All of these attributes along with a short response time and no required calibration (ref Knox) combine to yield the wind tunnel engineer a tool of marked improvement over previous methods.

1.2 OBJECTIVES

A method of analysis to identify the aerothermodynamic flight environment and update the thermal model of the engineering simulation of the Space Shuttle Orbiter was designed by the Air Force Flight Test Center. This method is in the form of a digital computer program called HEATEST (HEATing ESTimation). The program provides a correlation of the heating as well as a heat rate time history. The program integrates numerically instead of relying on some analytical assumption. It satisfies a maximum likelihood criteria for each parameter and obtains best estimates for the temperature at discrete nodes throughout the length of the
OsO-i-z z-0 Z <z o-j-i- <W ix

CHROMEL
CONSTANTAN
INSULATION

.062-

3.75

MODEL TCS
COAXIAL PROBE MOUNTED
IN A METAL WALL

FIGURE 1.1 COAXIAL THERMOCOUPLE CAGE
where
\[ G = P(t_n^-)H^T[H^T + R_m]^{-1} \]
\[ E = Y(t_n) - HU(t_n^-) \]

The updated sensitivities are calculated by,
\[ S_k(t_n^+) = [I - GH]P(t_n^-)[I - GH]^T + GR_mG^T \]  (3-5)

The updated covariance is calculated by,
\[ P(t_n^+) = [I - GH]P(t_n^-)[I - GH]^T + GR_mG^T \]  (3-6)

To alleviate the problem of imprecise initial conditions, a fixed point smoothing algorithm has been added to the HEATEST program. Details of the smoother and its effects in the adaptive estimation scheme may be found in Reference 6.

Finally, the best estimates of the parameters are then estimated at the end of a specified time segment by the gradient algorithm,
\[ \theta^* = \theta - \left[ \frac{\partial^2 F}{\partial \theta^2} \right]^{-1} \frac{\partial F}{\partial \theta} = \theta + J^{-1}S \]  (3-7)

where,
\[ J_{i,j} = \sum_{n=1}^{N} S_{i,k}(t_n^-)H^T[H^T + R_m]HS_{i,k}(t_n^-) \]
\[ S_k = \sum_{n=1}^{N} S_{i,k}(t_n^-)H^T + R_m]^{-1}[Y(t_n) - HU(t_n^-)] \]

The matrix J is an approximation for the Jacobian or conditional information matrix and is given in component
form by \( J_{i,k} \). The score vector, \( S_k \), is used to approximate the gradient of the likelihood function for a large number of time samples.

Using these equations, best estimates for the temperature time history (states) at each node can be found, as well as the deviation in temperature as provided by the covariance matrix. Also, an estimate of the parameter uncertainty is provided by the Cramer-Rao bound. The Cramer-Rao bound relates the conditional information matrix to the covariance of the parameter estimate.
IV. RESULTS

4.1 Test Procedure

To test the validity of the program modifications, a set of contrived data was generated. It's development assumes that the heat rate due to convection at the surface node is constant and equal to the heat rate due to conduction at the surface. The heat rate due to convection is given by,

\[ q = h(T_{aw} - T_w) \]

or

\[ q = h_{bar} h_{ref}(T_{aw} - T_w) \]  (4-1)

where \( h_{bar} \) is defined as in Equation A-12. The equation for the heat rate due to conduction assuming a one-dimensional, homogenous, semi-infinite solid is as follows (Ref 9),

\[ q = \frac{(\rho c k)^{1/2}}{\pi} \int_0^t \frac{dT_w(\tau)}{d\tau} \frac{d\tau}{(t-\tau)^{1/2}} \]  (4-2)

where \( t \) = time from start of heating
\( T(t) \) = surface temperature rise
\( \tau \) = dummy variable of integration

Equating Equations 4-1 and 4-2 yields,
Two different expressions for the derivative of the wall temperature with respect to time were used. The first implied a linear change in temperature with respect to time yielding a constant for \( \frac{dT_w}{dt} \) and the second expression assumes that temperature was a quadratic function of time as shown,

**Linear**

\[
T_w = bt + c
\]

\[
\frac{dT_w}{dt} = b
\]

**Quadratic**

\[
T_w = at^2 + bt + c
\]

\[
\frac{dT_w}{dt} = 2at + b
\]

Solving Equation 4-3 for \( T_{aw} \) so that \( h \) and \( q \) are constant and after making the indicated substitutions and integrating yields,

**Linear assumption**

\[
T_{aw} = T_w + \frac{2b}{h_{bar}h_{ref}} \sqrt{\frac{pckt}{\pi}}
\]  \hspace{1cm} (4-4)

**Quadratic assumption**

\[
T_{aw} = T_w + \frac{2}{h_{bar}h_{ref}} \sqrt{\frac{pckt}{\pi}} \left( b + \frac{4at}{3} \right)
\]  \hspace{1cm} (4-5)
A short computer program was written to produce a temperature-time history in the data tape format for the HEATEST program. For the above equations, $h_{\text{ref}}$ and $h_{\text{bar}}$ (the estimated parameter) were set equal to 1 and the coefficients $a$, $b$, and $c$, were selected to yield reasonable values for $T_w$.

4.2 Test Cases

The reference test case was taken to be a 10 sec. simulated wind tunnel test run using the linear data provided from the previous section. Thermocouple samples and test section samples were provided at the rate of one sample per second. The objective was to examine the rate of convergence of the temperature states and to estimate the first parameter, $h_0$. Recall from Section 4.1 that the data was generated to yield a value of one for $h_0$. Also of interest, was the validity of the model to the semi-infinite solid assumption (i.e. no change in the temperature at the back face node throughout any specified time segment).

The input data is shown in Figure 4.1 and is generated digitally depending upon a desired time step ($\Delta t$). The initial temperature throughout each gage is assigned a value of 60°F. Figure 4.2 shows the input temperature values for a $\Delta t = 1$ sec. as used in the reference test case.

Figure 4.3 identifies the temperature state at the 2 sec. (lowest curve), 6 sec. (middle curve), and 10 sec. (top curve) times following the first iteration through the
updated HEATEST program. It clearly shows that at the back face node, the semi-infinite solid assumption used to derive the data is violated. This is indicated by the change in backface (node 6) temperature with time. Note also, however, that the temperature gradient at the back nodes (between nodes 5 and 6) is zero due to the adiabatic wall assumption. It should also be pointed out that surface node temperature response to the given input was immediate with no time lag.

Figure 4.4 is similar to Figure 4.3 except the temperature states and parameter estimates have been iterated to convergence, in this case, three times. Overlaying the two figures shows no perceptible difference between them, and the data shows no variations in values until after the second decimal point. In spite of the response of the back face node which would ordinarily invalidate the test, the estimated value for $h_0$ was .99598, within .4% of the desired value of 1! Several test cases will be compared to this reference by examining changes in time step, thermocouple length, and the number of node points. Also, an examination of the ability of the program to estimate the other parameters, follows.

4.2.1 Changes in Time Step

Figures 4.4, 4.5, and 4.6 show temperature states at 2, 6, and 10 sec. for $\Delta t$ equal to 1, .5, and .25 sec., respectively. All three curves required three iterations
for convergence. The curves are all similar in shape with almost no perceptible differences. However, if the 10 sec. curves from \( \Delta t = 1 \) sec. and \( \Delta t = .25 \) sec. are overlayed as in Figure 4.7, a small difference may be noted at the backface nodes indicating that, indeed, a decrease in time step will yield a profile which will more closely approximate the model.

4.2.2 Changes in Thermocouple Length

Changes in thermocouple length offered the most dramatic changes in temperature state as can be seen in Figures 4.8, 4.9, and 4.10 where the lengths range from .1 ft., .05 ft., and .025 ft., respectively. The two longer lengths converged within three iterations while the short thermocouple length took four iterations to converge. The extra iteration is most likely due to the large deviation from the model and the large differences in temperature state from one time step to the next. Figure 4.11 compares the temperature state of each thermocouple length after the 10 sec. run. The lowest curve is associated with the longest thermocouple, and the upper curve is associated with the short gage.

4.2.3 Changes in Number of Node Points

For this comparison, a 10 sec. run with 6 node points and a 3 sec. run with 6 node points (Figures 4.12 and 4.13) will be compared with a 10 sec. run with 12 node points and
a 3 sec. run with 12 node points (Figures 4.14 and 4.15). Each of the test runs were converged by the third iteration. At both of the different run times, increasing the number of nodes yielded a solution which more closely approximates the model (ie. the semi-infinite solid at the back face node) and gave correspondingly better estimates for \( h_0 \). The comparison of temperature states is better represented by Figures 4.16 and 4.17 which directly compares 6 nodes and 12 nodes interspersed evenly throughout the .05 ft. long thermocouple for 10 sec. and 3 sec. run times, respectively.

4.2.4 Parameter Estimation

The ability of the algorithm to estimate \( h_0 \) has already been discussed. To summarize, even when the output temperature states clearly violate the semi-infinite solid assumption used in generating the data, the estimated values of \( h_0 \) remain within 15%. The 15% is a worst case number derived from the short thermocouple using a coarse grid for a long run time. An average deviation which considers all of the test cases evaluated is closer to 3%.

To estimate \( \phi_k \), an erroneous data value was given for \( K_{data} \) wherein the program iterated to a value for \( \phi_k \) which, when multiplied by the erroneous \( K_{data} \), would yield the correct value, ie. \( \phi_k K_{data} = K_{correct} \). The erroneous \( K_{data} \) which was input into the program was 14% in error of the \( K_{correct} \) value. The value provided by the program for \( \phi_k \) when multiplied by \( K_{data} \) yielded a value within .9% of
The estimate of $\phi_c$ was not as successful, however. After 6 iterations, the solution was diverging from the expected value. A suspected sign error in the $\phi_c$ sensitivity calculation is the most likely cause.

The quadratic input data was used as a comparison to the linear data to challenge the algorithm, i.e., the more complicated the input the more difficult the estimation process. No direct comparison may be made of temperature, however, since the input data is different. The parameter estimation of $h_4$ using the quadratic input data was still excellent yielding $0.1\%$, while the estimate using linear data was somewhat better at $0.05\%$. 
FIGURE 4.1

PLOT 1
INPUT DATA
LINEAR AND QUADRATIC
PLOT 1
INPUT DATA
* LINEAR
+ QUADRATIC

FIGURE 4.2
PLOT 2
10 SEC. DEL T=1
6 NODES, LINEAR
1ST ITERATION

FIGURE 4.3 TEMP VS NODE POINT HISTORY (2.6.10 SEC)
PLOT 5
10 SEC. DEL T=1
6 NODES, LINEAR
CONVERGED

FIGURE 4.4 TEMP VS NODE POINT HISTORY (2.5, 10 SEC)
Figure 4.5 Temp vs Node Point History (2.6, 10 sec)
the utility of the program to be used in wind tunnel runs of much longer duration.

5.2 Recommendations

Actual wind tunnel test data from coaxial gages needs to be analyzed by the program to instill more confidence in the results. This would require that the data tape format be modified to be compatible with the program inputs.

Also, a prescribed model for $h_{\text{bar}}$ as a function of angle of attack needs to be input to determine the ability of the program to estimate the piecewise linear derivatives, $h_{a1}$ and $h_{a2}$. The additional input data would then be the wind tunnel model angle of attack at each thermocouple sample time.

Another potential modification to the program would be to use a second order time derivative approximation as opposed to the current first order approximation. It is suspected that the increased accuracy would improve the state estimates particularly for large time gaps in thermocouple data.

The temperature state estimates might be improved by incorporating a variable grid. An exponential grid generation scheme would concentrate node points near the surface where the largest temperature gradients exist.

A final recommendation, would be to make the program more 'user friendly' and to publish documentation much like a users manual.
V. CONCLUSIONS AND RECOMMENDATIONS

5.1 Conclusions

The program works for this simplified test case of known parameters and number of node points. The ability of the program to determine the temperature state appeared very good but definite conclusions cannot be drawn without comparing this data to actual thermocouple data. The ability of the program to estimate the parameters, $h_0$ and $\phi_k$, was demonstrated extremely well. Re-examination of the $\phi_c$ equations must be made before further conclusions may be made about the ability of the program to estimate this parameter. It is theoretically possible, however.

The objective, to validate the use of the modified HEATEST program for use of coaxial thermocouple gages on wind tunnel test articles, has been met for the special case of this analytical model. The program is capable of determining temperature states and estimating parameters with a high degree of accuracy. A note concerning the semi-infinite slab assumption needs to be emphasized. This assumption was made only for the data generation program to yield data for which the analytical solution was known. As mentioned in Chapter 1, some coaxial gages are available with backface temperature monitoring which would also provide another temperature measurement to enhance estimation of $\phi_k$ and $\phi_c$. This feature would greatly enhance
FIGURE 4.17 TEMP VS NODE POS. (6 NODES VS 12 NODES)
FIGURE 4.16 TEMP VS NODE POS. (6 NODES VS 12 NODES)
PLOT 17
3 SEC. DEL T=.2
12 NODES, LINEAR
CONVERGED

FIGURE 4.15 TEMP VS NODE POINT (1.4, 1.6, 3.0 SEC.)
PLOT 15
10 SEC. DEL T=1
12 NODES, LINEAR
CONVERGED

FIGURE 4.14 TEMP VS NODE POINT HISTORY (2.6.10 SEC)
FIGURE 4.13 TEMP VS NODE POINT (0.4, 1.6, 3.0 SEC)
FIGURE 4.12 TEMP VS NODE POINT HISTORY (2.6.10 SEC)
FIGURE 4.11 TEMP VS POS. (T/C LENGTH=.05,.025FT)
FIGURE 4.10 TEMP VS NODE POINT (T/C LENGTH=.025FT)
Figure 4.9 Temp vs Node Point (T/C Length = .05 ft)
FIGURE 4.8 TEMP VS NODE POINT (T/C LENGTH=.1 FT.)
FIGURE 4.7 TEMP VS NODE POINT (DEL T=1 AND .25 SEC)
Figure 4.6 Temp vs Node Point History (2.6, 10 sec)
Derivation of equations using a one-dimensional energy balance formulation are given as follows,

\[ \frac{q_{\text{rad}}}{dx} = q \]

Energy in the left face

\[ = -k \frac{\partial T}{\partial x} = q_1 \]

Energy generated within the element

\[ = q \ dx = 0 \]

Change in internal energy

\[ = \rho c \left( \frac{\partial T}{\partial \tau} \right) dx \]

Energy out of right face

\[ = -k \left( \frac{\partial T}{\partial x} \right) _{x+dx} \]

\[ = -[k \frac{\partial T}{\partial x} + \frac{\partial }{\partial x} (k \frac{\partial T}{\partial x})] \]

Then, combining the above and using Fourier's Law of Heat Conduction, i.e.,

\[ \text{energy in} + \text{energy within} = \text{change in internal energy} + \text{energy out} \]

yields,

\[ -k \frac{\partial T}{\partial x} + q dx = \rho c \frac{\partial T}{\partial \tau} dx - \left[ k \frac{\partial T}{\partial x} + \frac{\partial }{\partial x} (k \frac{\partial T}{\partial x}) \right] \]

or,

\[ \rho c \frac{\partial T}{\partial \tau} = \frac{\partial }{\partial x} (k \frac{\partial T}{\partial x}) \]
or, replacing $T$ by $U$,

$$\rho c U_t = (k U_x)_x \quad (A-3)$$

or,

$$\rho c U_t = \left[ \frac{k_{i-1/2}}{\Delta x_{i-1/2}} (U_{i-1}^n - U_{i+1}^n) - \frac{k_{i+1/2}}{\Delta x_{i+1/2}} (U_i^n - U_{i+1}^n) \right] \frac{1}{\Delta x} \quad (A-4)$$

where $\Delta x$ may be written as,

$$\Delta x = \frac{\Delta x_{i+1/2} + \Delta x_{i-1/2}}{2}$$

then, writing the time gradient in first order backward difference form and expanding yields,

$$\frac{U_i^n - U_{i-1}^n}{\Delta t} = \frac{2k_{i-1/2}}{\Delta x_{i-1/2}(\Delta x_{i+1/2} + \Delta x_{i-1/2})} U_{i-1}^n$$

$$- \frac{2k_{i-1/2}}{\Delta x_{i-1/2}(\Delta x_{i+1/2} + \Delta x_{i-1/2})} U_i^n$$

$$- \frac{2k_{i+1/2}}{\Delta x_{i+1/2}(\Delta x_{i+1/2} + \Delta x_{i-1/2})} U_i^n$$

$$+ \frac{2k_{i+1/2}}{\Delta x_{i+1/2}(\Delta x_{i+1/2} + \Delta x_{i-1/2})} U_{i+1}^n \quad (A-5)$$

then, specifying equal spacing for each node point,
\[ \Delta x_{i+1/2} = \Delta x_{i-1/2} \]

and the equation becomes,

\[ \frac{U_i^n - U_{i-1}^{n-1}}{\rho \phi_c c \Delta x} \frac{\Delta t}{\Delta x^2} = \frac{k_{i-1/2}}{\Delta x^2} U_{i-1}^n \]

\[ - \left( \frac{k_{i-1/2}}{\Delta x^2} + \frac{k_{i+1/2}}{\Delta x^2} \right) U_i^n + \frac{k_{i+1/2}}{\Delta x^2} U_{i+1}^n \]

(A-6)

Now, instead of estimating \( c \) and \( k \) directly, define two scaling parameters \( \phi_c \) and \( \phi_k \) such that \( c \) and \( k \) will remain constant. These two parameters are estimated by the HEATEST program.

\[ \frac{U_i^n - U_{i-1}^{n-1}}{\rho \phi_c c \Delta x} \frac{\Delta t}{\Delta x} = \phi_k \frac{k_{i-1/2}}{\Delta x} U_{i-1}^n \]

\[ - \phi_k \left( \frac{k_{i-1/2}}{\Delta x} + \frac{k_{i+1/2}}{\Delta x} \right) U_i^n + \phi_k \frac{k_{i+1/2}}{\Delta x} U_{i+1}^n \]

(A-7)

This equation is applicable at all interior (\( i \neq 1, i \neq i_{\text{max}} \)) points.

For the back face, assuming a semi-infinite solid, \( i = i_{\text{max}} \) and the equation becomes,

\[ \frac{\rho \phi_c c \Delta x}{2} \frac{U_L^n - U_{L-1}^{n-1}}{\Delta t} = \phi_k \frac{k_{L-1/2}}{\Delta x} U_{L-1}^n - \phi_k \frac{k_{L-1/2}}{\Delta x} U_L^n \]

(A-8)
For the front face, \( (i = 1) \), the effects of radiation away from and convection toward the solid surface must be accounted for.

The radiation is modeled using the Stefan-Boltzmann Law,

\[
q = e \sigma (U_1^4 - U_\infty^4)
\]  
(A-9)

where
- \( e \) radiative emissivity
- \( \sigma \) Stefan-Boltzmann constant
- \( U_1 \) Temperature \(^\circ\text{R}\)

The convective transfer of energy is modeled using Newton's Law of Cooling,

\[
q = h(T_{aw} - T_w)
\]  
(A-10)

Non-dimensionalizing by a reference heat transfer coefficient, \( h_{ref} \) yields,

\[
q = h_bar h_{ref} (T_{aw} - U_1)
\]  
(A-11)

where, \( h_bar \) = convective heat transfer coefficient ratio
\( T_{aw} \) = adiabatic wall temp of test article

The dependance of the heat transfer coefficient on parameters other than those included in the reference heat transfer coefficient are summarized by the static transfer relation or heat transfer coefficient ratio, \( h/h_{ref} \). Here, the ratio is assumed to be piecewise linear with respect to angle of attack as derived from Lagrange Interpolation Theory (Ref 6),
\[ h_{\text{bar}} = \frac{h}{h_{\text{ref}}} = \left[ h_0 + h_{a1}(\alpha - \alpha_1) + h_{a2}(\alpha - \alpha_2) \right] \]  

(A-12)

where \( h_0 \) is the magnitude of the heat transfer coefficient, \( h \), at the reference condition specified at \( \alpha_1 \). Combining Equations A-7, A-9, A-11, and A-12 and evaluating at node one yields,

\[
\frac{\rho \phi c \Delta x}{2} \frac{U_1^n - U_1^{n-1}}{\Delta t} = -\frac{\phi k_k^{k+1/2}}{\Delta x} U_1^n \\
+ \frac{\phi k_k^{k+1/2}}{\Delta x} U_2^n - \varepsilon \sigma [(U_1^n)^4 - (U_\infty^n)^4] \\
+ [h_0 + h_{a1}(\alpha - \alpha_1) + h_{a2}(\alpha - \alpha_2)] h_{\text{ref}}(T_{aw} - U_1^n)
\]

(A-13)

Using the quasi-linearization as developed in Equation 2-6, the resultant form for determining the temperature time history at each node point is given in Equations 2-7 and 2-8.

The matrix form for the equations may be found after defining the following,

\[
\begin{align*}
RCX_i &= \rho \phi c \Delta x \\
RCX_1 &= \frac{RCX_1}{2} \\
RCX_L &= \frac{RCX_L}{2} \\
RM_i &= \frac{\phi k_k^{k+1/2}}{\Delta x} \\
RM_1 &= 0 \\
RP_i &= \frac{\phi k_k^{k+1/2}}{\Delta x} \\
RP_L &= 0
\end{align*}
\]

50
\[
BBB_1 = \frac{RCX_1}{\Delta t} + RM_1 + RP_1 + 4\varepsilon \sigma (U_1^n s^n_1)^3 + h_{\text{bar ref}}
\]

\[
BBB_i = \frac{RCX_i}{\Delta t} + RM_i + RP_i
\]

Then, using Equations A-14 in Equations 2-7 and 2-8 yields the matrix form of Equation 2-9,

\[
[A][U_i^n] + \{b\} = 0
\]

where,

\[
[A] = \begin{bmatrix}
-1 & RP_i/BBB_i & 0 \\
RM_i/BBB_i & -1 & 0 \\
0 & RM_i/BBB_i & -1
\end{bmatrix}
\]

and,

\[
\{b\} = \begin{cases}
\varepsilon \sigma [(3U_1^n s^n_1)^4 + U_0^4] + h_{\text{bar ref}} Taw + \frac{RCX_1 U_1^n}{\Delta t} & BB_i \\
\frac{RCX_i U_i^n}{\Delta t} & BB_i \\
\frac{RCX_L U_L^n}{\Delta t} & BB_L
\end{cases}
\]

(A-16)
APPENDIX B

The derivation of the sensitivity equations. The derivative of Equation 2-7 with respect to each parameter yields equations from which the HEATEST program propagates the sensitivity. A vector of parameters is formed and the sensitivity notation is as shown,

\[ \Theta = [h_0, h_{a1}, h_{a2}, \rho_c, \rho_k]^T \quad S_{i,k} = \frac{\partial U}{\partial \Theta_k} \]

Parameter no
Node point

Defining, \( h_{\text{bar}} = [h_0+h_{a1}(\alpha-a_1)+(h_{a2}(\alpha-a_2)) \), the sensitivity equations at node one are,

\[ \Theta_1: \quad \frac{\rho \rho_c c \Delta x}{2} \frac{S_{1,1}^n-S_{1,1}^{n-1}}{\Delta t} = -\frac{\phi_k(k_i+1/2)}{\Delta x} S_{1,1}^n \]

\[ + \frac{\phi_k(k_i+1/2)}{\Delta x} S_{2,1}^n - 4 \varepsilon \sigma (U_1^n)^3 S_{1,1}^n + h_{\text{ref}}(T_{aw}-U_1^n) \]

\[ - S_{1,1}^n h_{\text{ref}} h_{\text{bar}} \]

\( (B-1) \)

\[ \Theta_2: \quad \frac{\rho \rho_c c \Delta x}{2} \frac{S_{1,2}^n-S_{1,2}^{n-1}}{\Delta t} = -\frac{\phi_k(k_i+1/2)}{\Delta x} S_{1,2}^n \]

\[ + \frac{\phi_k(k_i+1/2)}{\Delta x} S_{2,2}^n - 4 \varepsilon \sigma (U_1^n)^3 S_{1,2}^n \]

\[ + (\alpha-a_1) h_{\text{ref}}(T_{aw}-U_1^n) - S_{1,2}^n h_{\text{ref}} h_{\text{bar}} \]

\( (B-2) \)
The sensitivity equations at the interior node points are as
follows.

\[ \theta_1, \ i = 1, 2, 3 \]

\[
\frac{\rho \phi_c c \Delta x}{\Delta t} S_{i,k}^{n} - S_{i,k}^{n-1} = \frac{\phi_k k_{i-1/2}}{\Delta x} S_{i-1,k}^{1/2}
\]

\[- \frac{\phi_k (k_{i-1/2} + k_{i+1/2})}{\Delta x} S_{i,k}^{1/2}
\]

\[ + \frac{\phi_k k_{i+1/2}}{\Delta x} S_{i+1,k}^{1/2} \]

\[ (B-6) \]

\[ \theta_4: \]

\[
\frac{S_{i,4}^{n} - S_{i,4}^{n-1}}{\Delta t} = \frac{\phi_k k_{i-1/2}}{\rho \phi_c c \Delta x^2} S_{i-1,4}^{n-1} - \frac{\phi_k k_{i-1/2}}{\rho \phi_c^2 c \Delta x^2} U_{i-1}^{n-1}
\]

\[- \frac{\phi_k (k_{i-1/2} + k_{i+1/2})}{\rho \phi_c c \Delta x} S_{i,4}^{n}
\]

\[ + \frac{\phi_k (k_{i-1/2} + k_{i+1/2})}{\rho \phi_c^2 c \Delta x} U_{i}^{n} + \frac{\phi_k k_{i+1/2}}{\rho \phi_c c \Delta x^2} S_{i+1,4}^{n}
\]

\[- \frac{\phi_k k_{i+1/2}}{\rho \phi_c^2 c \Delta x^2} U_{i+1}^{n} \]

\[ (B-7) \]

\[ \theta_5: \]

\[
\frac{\rho \phi_c c \Delta x}{\Delta t} S_{i,5}^{n} - S_{i,5}^{n-1} = \frac{\phi_k k_{i-1/2}}{\Delta x} S_{i-1,5}
\]

\[- \frac{\phi_k (k_{i-1/2} + k_{i+1/2})}{\Delta x} S_{i,5} \]

54
\[ + \frac{\phi_{k+1/2}}{\Delta x} S_{i+1,5} + \frac{k_{i-1/2}}{\Delta x} U_{i-1} \]

\[- \frac{(k_{i-1/2} + k_{i+1/2})}{\Delta x} U_i + \frac{k_{i+1/2}}{\Delta x} U_{i+1} \]

(B-8)

The backface equations are of the same form as the node 1 equations without the convection and radiation terms.

If Equations A-14 are used to reduce the equations to the form of Equation 2-9, the sensitivity equations become,

\[ [A'] [S_{i,k}] + [d_k] = 0 \]  

(B-9)

where the [A] matrix for the sensitivity equations is the same as the [A] matrix for the temperature equations, A-15.

The \( d \) vectors for each parameter are listed as follows,

\( (d)_1 = \left\{ \begin{array}{c}
\left( \frac{h_{\text{ref}}(T_{aw}-U_1) + \frac{RCX_1}{\Delta t} S_{1,1} \cdot n^{-1}}{BBB_1} \right) \\
\left( \frac{RCX_i S_{i,1} \cdot n^{-1}}{\Delta t} \right) / BBB_i \\
\end{array} \right\} \)

\( (d)_2 = \left\{ \begin{array}{c}
\left( \frac{(\alpha-\alpha 1)h_{\text{ref}}(T_{aw}-U_1) + \frac{RCX_1}{\Delta t} S_{1,2} \cdot n^{-1}}{BBB_1} \right) \\
\left( \frac{RCX_i S_{i,2} \cdot n^{-1}}{\Delta t} \right) / BBB_i \\
\end{array} \right\} \)

55
\[ (a-a2)h_{ref}(T_{aw}-U_1) + \frac{RCX_i S_{1,3}}{\Delta t} + \frac{RCX_i S_{i,4}}{\Delta t} \] /BBB

\[ \{ \begin{array}{l}
(\frac{RCX_i S_{i,3}}{\Delta t} + \frac{RCX_i S_{i,4}}{\Delta t}) /BBB \\

\{ \begin{array}{l}
(\frac{-RP_1(U_2-U_1) + \sigma(U_{i+1}^4-U_i^4) - \frac{RCX_i S_{i,4}}{\Delta t}}{\frac{\rho_c}{\rho_c}})

\{ \begin{array}{l}
(\frac{-RM_i(U_{i-1}-U_i) + RP_1(U_i-U_{i+1}) + \frac{RCX_i S_{i,4}}{\Delta t}}{\frac{\rho_c}{\rho_c}})

\{ \begin{array}{l}
(\frac{RP_1(U_2-U_1) + RCX_i S_{i,5}}{\Delta t}) /BBB

\{ \begin{array}{l}
(\frac{-RM_i(U_{i-1}-U_i) + (RM_i + RP_1)U_i + RP_i(U_{i+1}) + RCX_i S_{i,5}}{\Delta t}) /BBB

(B-10)
APPENDIX C

The HEATEST program follows.
DATA LABELX/SHLABEL/
DATA DTIP/0/
DATA ERALOW/.5/
DATA KAS, KAF, IPE/1, 1, 1/
DATA THDEP/.05/
DATA NPTSS/.5/
DATA TRAD/0./

CALL INPUT
CALL NODE POINT STRUCTURE

C ENTER OUTER/PARAMETER ESTIMATION ITERATION LOOP

READ(5, 3990) ITRSK, ITCSK, NPTPC, IIC
3990 FORMAT(8(8X, I2))
READ(5, 4001) INTERV, IFXFLG
4001 FORMAT(6X, I4, 9X, L1, 10X, 7L1)
NDEL = (TSTOP - TSTART) / INTERV
TSTOPF = TSTOP
4 TSTOP = TSTART + DELS
IF(IFXFLG) THEN
1 READ(5, 4011) END = 4024) NRPIITER, IFX, FAVTO, TSTOP, KFPT, NPTPC
4011 FORMAT(4X, I1, I5, 1811, 10X, 7L1, 10X, F10.5, 5X, 11, 8X, 12)
1 READ(5, 4012) END = 4034) FREAD, READ
4012 FORMAT(3X, 13L1, 13F8.4)
1 DO 4016 II = 1, 5
4016 IF(FREAD((II)) = IFXFLG) THEN
1 READ(5, 4013) END = 4034) FREAD, READ
1 DO 4017 II = 1, 2
4017 IF(FREAD((II)) = IFXFLG) THEN
1 GO TO 4033
GO TO 4033
4034 IFXFLG = .FALSE.
97 TSTOP = TSTOPF
98 CONTINUE
199 END IF
100 IF((TSTOP, GT. TSTOPF) TSTOP = TSTOPF
101 IFXSUM = 0
102 DO 30 I = 1, NPAR
30 IFXSUM = IFXSUM + IFX(I)
104 NRPI = NRPI + 1
105 DO 198 ITPRAM = 1, NRPI
106 KA = 1
107 KA = 1
108 REWIND 3
109 REWIND 4
110 REWIND 10
111 REWIND 13
112 CALL ZERO(5, NPAR, 1)
SUBROUTINE TPS3 (DTT) UPAUG1 11
COMMON /CCTMNT/NTCT UPAUG1 12
COMMON /COMTUN/T,TAM,ALPHA,H,V,RHO,P,TEMP,C,TRAD,RHOG, OCT10 3
&TSINK,XFT,DEL,PD1 COMTUN 3
COMMON /CHEAT/Q,T,REF,TM,M1,RENS,HBAR,HEF CP0CT09 5
COMMON/COSP/NPTSS,USI(6),PHI(6,6),NPT,PC(8,6),RR, UPAUG1 6
&QD(1,0),QD(1,1),QD(1,2),R(0,6),A(6,8),RCK(8),R(6),RM(8) UPOCT09 5
COMMON/CSENS/SUSI(6,5),UMI(8) UPAUG1 7
COMMON /ICTPS2/TINIT(1),ERALOW,E UPAUG1 15
COMMON /CDX/DX(1) UPAUG1 3
REAL M3 UPOCT09 8
DIMENSION AAAA(6),CC CCC(6),DDDD(6),G(6),W(6) FTPS3 19
EQUIVALENCE (QD(1,1),AAAA(1)),QD(1,2),CCCC(1),QD(1,3),DDDD(1)) FTPS3 20
&QD(1,5),G(1),QOT(1,1),W(1)) FTPS3 20
DATA SIG/4.781E-13/ UPAUG1 18
DATA MIT/2/ UPAUG1 17
C SHIFT STORAGE FTPS3 38
DO 460 I=1,NPTSS FTPS3 39
460 UM(I)=USI(1) FTPS3 40
C C FORM TRIAGONAL MATRIX FTPS3 46
C DO 511 I=2,NPTSS FTPS3 47
C BBB=RCX(I)/DTT+RP(I)+RM(I) FTPS3 48
AAA(I)=RM(1)/BBB FTPS3 49
CC C(1)=RP(1)/BBB FTPS3 50
DDDD(1)=RCX(I)+UMI(1)/DTT/BBB FTPS3 60
C CONTINUE FTPS3 51
511 CONTINUE FTPS3 72
C TRIAGONAL SOLUTION FTPS3 73
DO 540 M=1,MIT FTPS3 74
C I=1 FTPS3 79
BBB=RCX(I)/DTT+RP(I)+RM(I)+.E*S I+(USI(1)+460.)*3.+ UPAUG1 18
*HBAR+HREF FTPS3 79
AAA(I)=RM(1)/BBB FTPS3 80
CC C(1)=RP(1)/BBB FTPS3 81
DDDD(1)=RCX(I)+UMI(1)/DTT+E*S I+(3*USI(1)+TRAD+4)+HBAR+HREF* UPAUG1 22
&TA M(I)/BBB FTPS3 104
G(I)+DDDD(1) UPAUG1 24
W(I)=CCC(1) FTPS3 25
DO 520 I=2,NPTSS FTPS3 110
510 WI=CCC(I)/(1.+AAA(I)+W(I-1)) FTPS3 115
520 G(I)=(DDDD(I)+AAA(I)+G(I-1))/(1.+AAA(I)+W(I-1)) FTPS3 118
UNEW +G(NPTSS) FTPS3 118
UERMX=ABS(UNEW-USI(NPTSS)) FTPS3 119
USI(NPTSS)=UNEW FTPS3 120
DO 530 L=2,NPTSS FTPS3 121
I=NPTSS-L+1 FTPS3 122
UERM=ABS(UNEW+USI(I+1)) FTPS3 123
500 UERM=ABS(UNEW-USI(I)) FTPS3 124
UERMX=AMAX1(UERMX,UERM) FTPS3 125
USI(I)=UERM FTPS3 126
530 CONTINUE FTPS3 127
540 CONTINUE FTPS3 128
IF(UERMX. LT.ERALOW. AND. M. GE. 5) GO TO 550 FTPS3 129
SUBROUTINE MAKEA

56       RCI(I) = RHOG + PHIC + ZP + DX(1) * .5
57       RP(I) = PHIK + Z/DX(1)
58       RM(I) = 0.
59       A(1,1) = RP(I)/RCX(I)
60       A(1,2) = RP(I)/RCX(I)
61       ALIN = - (4. * E * SIG + (USI(1) + 460.) + 3. * HBAR + HREF) / RCX(I)
62       END IF
63       511 CONTINUE
64       A(1,1) = A(1,1) + ALIN
65       C
66       C SYSTEM MATRIX COMPLETED
67       C
68       RETURN
69       END
SUBROUTINE MAKEA  74/865  OPT=0 ROUND= A/ S/ M/ D/ 05  FNT 5.1-587  84/11/19. 13. 14. 29  PAGE 1

SUBROUTINE MAKEA
COMMON /CPLCL/IPT, PFRAC
COMMON /COMTUN/T, TM, ALPHA, H, V, RHO, P, TEMP, C, TRAD, RHOG,
& TO, TSNK, XPT, DEL, POEL COMMON /CHEAT/O, TS, OREF, TW, M, RENS, HBAR, HREF
COMMON /ICTPS2/TINIT(1), ERALOW, E
COMMON /COSP/NPTSS, UIS(6), PHI(6, 8), NPT, PC(6, 8), RR,
& OD, OD, ODEC(6, 8), QUE(6, 8), A(6, 8), RCX(6, 8), RP(8, 8)
COMMON /CP/CPC/PCP
LOGICAL FAUTO
DIMENSION FAUTO(7)
COMMON /DIMOP/HO, HALF(2), PHIC, PHIK, ZP, Z, ALPHI(2), KA(S, 5), & CIF(5, 8), KAF, IFX(5), ACC(5), IFXSUM, NPAR, DALPHI(2)
EQUIVALENCE (HO, OP(5))
COMMON /CKF/K(6), S1(6, 8), J1I(6, 8), TC(2), NODES(2), KTQPT
REAL K, J1
COMMON /CX/DX(1)
REAL M1
COMMON /CCLUD/NTCT
DATA SIG/4.781E-13/
DATA E/3/.
DATA HREF/1/.
DATA RHOG/17.1060393/.
DATA ZP/3.233477/.
DATA Z/3.054E-3/.
C SET UP LINEARIZED SYSTEM MATRIX, A
C CALL ZERO (A(I, 1), NPTSS, NPTSS)
DO 511 I=1, NPTSS
C CURRENT PASS TEMPERATURES
C FORM MATRIX
C I=NPTS
IF (I.EQ. NPTSS) THEN
   RCX(I)=RHOG*PHIC*ZP+DX(1)*.5
   RM(I)=PHIK*Z/DX(I)
   RP(I)=0.
   A(I, I-1)=RM(I)/RCX(I)
   A(I, I)=RM(I)/RCX(I)
C BLOCK B INTERIOR POINTS
ELSE IF ((I.GT. 1).AND.(I.LT. NPTSS)) THEN
   RCX(I)=RHOG*PHIC*ZP+DX(1)
   RP(I)=PHIK*Z/DX(I)
   RM(I)=PHIK*Z/DX(I)
   A(I, I-1)=RM(I)/RCX(I)
   A(I, I)=RM(I)/RCX(I)
   A(I, I+1)=RP(I)/RCX(I)
   A(I, I+1)=A(I, I+1)-A(I, I+1)
C SURFACE NODE I=1
ELSE IF (I.EQ. 1) THEN
C WHILE I.EQ.1 ADD BOUNDARY NOISE DUE TO HEAT TERMS
56     IF (I.EQ.1) THEN
57         QUE(I,J) = QER*QER*SDME*SQRT(SDBN**2+SDME**2)*R(I,J)
58     ELSE
59         QUE(I,J) = QER*QER*SDME**2*R(I,J)
60     END IF
61
62     C END WHILE
63     C
64     221 QUE(J,I) = QUE(I,J)
65     222 CONTINUE
66     C
67     RETURN
68     END
SUBROUTINE QUEMAT    74/885  OPT=O, ROUND=  A/  S/ M/ -DS  FYN 5.1:587  84/11/19  13.14.29  PAGE 1

DO = LONG/ -OT, ARG = COMMON/ -FIXED, CS = USER/ -FIXED, DB = TB/ -SB/ -SL/ ER/ -ID/ -PM0/ -ST, PL = 5000
FTN5, I, ANSI = O, L=OUTS, 0=-S/ -A.

1  SUBROUTINE QUEMAT
   COMMON /CONST/ XP(13)
   COMMON /IFICENT, IPRINT
   COMMON /IFLAN/ IFLAN, IPRINT
   COMMON /COMMT/T,AW1, ALPHA, H, V, RHO, P, TEMP, C, TRAD, RHOG
   &TD, TSINK, KF, DEL, FDEL
   COMMON /CTYPE/ NTCT
   COMMON /CDX/ DX(1)
   COMMON /COS, NPTSS, USI(6), PHI(6,6), NPT, PC(6,6), RR
   &QD(6,6), QOT(6,6), QUE(6,6), A(6,6), RCK(6,6), RPK(6,6), RM(6)
   COMMON /CSNS, SUI(6,6), UM1(6)
   COMMON /CKF/S(6,6), J(6,6), TCI(2), NODES(2), KF0PT
   REAL K, J1
   COMMON /CICST/ TR, SDIC, SDME, SDMEA, SDBN
   COMMON /CHEAT/ Q, TS, QREF, TW, M1, RENS, HBAR, HREF
   REAL M, M1
   DIMENSION R(6,6)
   EQUIVALENCE (QD(1,1), R(1,1))
   FQUE = 19
   FQUE = 18
   XR = SDME
   C MATRIX OF SPACIAL CORRELATIONS, R
   C SPACIAL CORRELATIONS IN RSI MUST ALLOW FOR VARIABLE NODE STRUCTURE
   DO 200 I = 1,NPTSS
      DO 280 J = 1,NPTSS
         X = SUMR*TR
         XP = ABS(XP)
         IF (I .EQ. 1) XP(J) = XP*TR
         IF (XP .GT. 100.) XP = 100.
         R(I,J) = EXP(-XP)
         IF (ABS(RP(J)), LT, 1.E-8) GO TO 200
         SUMR = SUMR + R(J,J)
      END
      PRINT*, 'XP1(', I, ',', J, ')', NPTSS
   END
   C MODEL ERROR MATRIX, QME
   RC = SQRT(ABS(RCK(1) + RCK(2)) + .5))
   QER = (HBAR*HREF*(TAW1-USI(1)) / (RM(1) + RPK(1))/2)**2
   DO 221 I = 1, NPTSS
   END
   C WHILE I.EQ.1 ADD BOUNDARY NOISE DUE TO HEAT TERMS
   IF (I .EQ. 1) THEN
      QME(I,1) = (SDBN+QER)**2 + (SDME*QER)**2/2
   ELSE
      QME(I,1) = (SDME*QER)**2
   END IF
   C END WHILE
   IF (I .EQ. NPTSS) GO TO 222
   IP1 = I + 1
   DO 221 J = IP1, NPTSS
C READ CALCOMP PLOT SPECIFICATIONS

C READ(5,100)IPLT
C READ(5,2000)TSCLAE,TMIN,TAUL
C READ(5,2000)YSCLAE,YMIN,YAXL
C READ(5,2000)ASCLAE,A.MIN,A.AXL
C 1000 FORMAT(10X,12,7(3X,12))
C READ TIMES
C C START-STOP TIMES / PRINT TIME STEP
C READ(5,2000)TSTART,TSTOP,DTPENT
C DATA FOR I.C. SMOOTHER
C READ(5,100)SMIC,TSMTH
C 100 FORMAT(6X,L1,8X,F10.8)
C # OF ITERATIONS/FIX=O FIXES PARAMETER/AUTO FLAG FOR REFERENCE
C READ(5,2010)NPITER,IFX,FAUTO,KFOPT
C C NEWTON-RAPHSON ACCELERATION PARAMETERS(0)ACC=2
C READ(5,2015)ACC
C C HEATING MODEL INITIAL PARAMETERS
C READ(5,2020)(QP(II),II=1,NPAR)
C C INITIAL REFERENCE VALUES FOR HEATING MODEL
C READ(5,2030)ALPH
C 2010 FORMAT(5X,12,3X,511,5X,7L1,8X,11)
C 2015 FORMAT(5X,8F5.2,5X,8F5.2)
C 2020 FORMAT(10X,8F8.4/20X,5F8.4)
C 2030 FORMAT(10X,7F8.4)
C RETURN
C END
SUBROUTINE IC
  LOGICAL IFICENT, IFPLLOT, IFPRINT
  COMMON /CFLAG/IFICENT, IFPLLOT, IFPRINT
  COMMON /CTCMNT/NTCT
  COMMON /CDX/DX(1)
  COMMON /COSP/NPTSS, USI(6), PHI(6,6), NPT, PC(6,6), RR,
  &OQ(6,6), QD(6,6), QUE(6,6), A(6,6), RCX(6), RP(6), RM(6)
  COMMON /ICTPS2/INIT(1), ERALOW, E
  COMMONE/CSENS/SUSI(6,5), UH(6)
  COMMON/CX/R(6,6), SI(6,6), J1(6,6), TC(2), NODES(2), KFOP
  REAL K1
  COMMON /CICSTAT/TR, SDIC, SDME, SDMEA, SDDBN
  COMMON /CHEAT/Q, TS, QREF, TW, M1, RENS, HBAR, HREF
  REAL M1
  DIMENSION R(6,6)
  EQUIVALENCE (QD(1,1), R(1,1))
  INITIAL SENSITIVIES
  C
  MATRIX OF SPACIAL CORRELATIONS, R
  C
  SPACIAL CORRELATIONS IN RSI MUST ALLOW FOR VARIABLE NODE STRUCTURE
  DO 200 I=1,NPTSS
    SUMR=0.
  DO 200 J=1,NPTSS
    XP=SUMR/TR.
    XP=ABS(XP)
    IF(XP.GT.100.) XP=100.
    R(I,J)=EXP(-XP)
    IF(ABS(RP(J)).LT.1.E-8) GO TO 200
  SUMR+=SUMR+RCX(J)/RP(J)
  200 R(J,I)=R(I,J)
  C
  COVARIANCE MATRIX OF TEMP ICS, PC
  C
  DO 210 I = 1, NPTSS
    PC(I,I) = (SDIC*USI(I))*2.
  DO 210 J = 1, NPTSS
    IF (I.EQ.NPTSS) GO TO 211
  IF (I.EQ.NPTSS) GO TO 211
  IP1 = I+1
  PC(I,J) = USI(I)*USI(J)*SDIC*2.*R(I,J)
  210 PC(J,I) = PC(I,J)
  211 CONTINUE
  RETURN
END
56 C READ CALCOMP PLOT SPECIFICATIONS
57 C READ (5,1000) PLOT
58 READ(5,1000) TSCALE,TMIN,TAXL
59 READ(5,2000)YSYAL,PMIN,YAXL
60 READ(5,2000)ASCAL,PMIN,AXAXL
61 1000 FORMAT(10X,12,7(3X,12))
62 C FINPUT 129
63 C READ TIMES
64 C START-STOP TIMES / PRINT TIME STEP
65 READ(5,2000) TSTART,TSTOP,DPENT
66 C DATA FOR I.C. SMOOTHER
67 READ(5,100) SMIC,TSNTH
68 100 FORMAT(8X,L1,8X,F10.5)
69 C # OF ITERATIONS/FIX=0 FIXES PARAMETER/AUTO FLAG FOR REFERENCE
70 READ(5,2010) NRITER,IFX,FAUTO,KFOPT
71 C NEWTON-RAPHSON ACCELERATION PARAMETERS(0)ACC12)
72 READ(5,2015) ACC
73 C HEATING MODEL INITIAL PARAMETERS
74 READ(5,2020) (QP(I)),II=1,NPAR)
75 C INITIAL REFERENCE VALUES FOR HEATING MODEL
76 READ(5,2030) ALPH
77 2010 FORMAT(5X,12,3X,5I1,5X,7L1,8X,I1)
78 2015 FORMAT(5X,8F5.2/5X,8F5.2)
79 2020 FORMAT(10X,8F8.4/26X,5F8.4)
80 2030 FORMAT(10X,7F8.4)
81 RETURN
82 END
SUBROUTINE INPUT

1 SUBROUTINE INPUT
2 LOGICAL IFICIENT, IFPLOT, IFPRINT
3 CHARACTER *30 VEH, FLTD, TMANV, CTPT
4 COMMON /CFLAG/IFICIENT, IFPLOT, IFPRINT
5 COMMON /CTMNT/NTCT
6 COMMON /CTIME/TSTART, TSTOP, DTPENT, NRPITER, ITPRAM
7 COMMON /ICTPS/2TINIT(1), ERALOW, E
8 COMMON /CDX/DX(1)
9 LOGICAL FAUTO
10 DIMENSION FAUTO(7)
11 DIMENSION QP(5)
12 COMMON /CPARAM/HO, HALF(2), PHIC, PHIK, ZP, Z, ALPH(2), KA, S(5),
13 &GF(5, S), KAF, IFX(5), ACC(5), IFXSUM, NPAR, DALPH(2)
14 EQUIVALENCE (HO, QP(1))
15 COMMON /CKF/K(8), S(8), J(8, 8), TC(2), NODES(2), KFOPT
16 REAL K, J1
17 COMMON /CSMTH/ UICSM(8), PICS(8, 6), UAP(8), PAP(8, 8),
18 &SMIC, TSMTH, W(8, 8)
19 LOGICAL SMIC
20 COMMON /COM/VEH, FLTD, TMANV, CTPT
21 COMMON /CDSMTR/SD, SDME, SDBN
22 COMMON /CFPLOT/IFPLOT, TSCALE, TMIN, TAXL, YSCALE, YMIX, YAXL, ASCALE, AMIX
23 *
24 C READ VEHICLE/MANEUVER UNIQUES
25 C READ(5, 3000)VEH
26 C FORMAT(A30)
27 C READ(5, 3000)FLTD
28 C READ(5, 3000)TMANV
29 C READ(5, 3000)CTPT
30 C READ(5, 3000)NTCT
31 C IFICIENT = T ICS FROM DISK / F CONSTANT ICS FROM CARDS
32 C IFPLOT = T CREATE CALCOMP / F NO PLOT FILE
33 C IFPRINT = T PRINT TIME SERIES / F NO TEMP TIME SERIES OUTPUT
34 C READ(5, 4000)IFICIENT
35 C FORMAT(8X, L1))
36 C READ(5, 4000)IFPLOT
37 C READ(5, 4000)IFPRINT
38 C FORMAT(10X, F10.5, 3(5X, F10.5))
39 C READ IN INITIAL TEMPERATURES (USED IF IFICIENT = F)
40 C READ(5, 2000)INIT
41 C INITIAL COVARIANCE AND ERROR STATISTICS
42 C READ(5, 2001)TR, SD, SDME, SDMA
43 C READ(5, 2001)SDBN
44 C 2001 FORMAT(10X, F10.5, 3(5X, F10.5))
PRINT *, 'T,USI=', T, T, TEST, (USI(II), II=1,NPTSS)
REWIND 3
TSTART=TSTOP
GO TO 4
END IF
REWIND 3
WRITE(3) USI, PC, SUSI
PRINT *, 'T,USI=', T, T, TEST, (USI(II), II=1,NPTSS)
998 IF(IFPLOT) CALL FPLOT
C PLOT CALCOMP FILE FROM TAPE12
C STOP 'END-OF-FILE ENCOUNTERED ON INPUT TAPE IN HEATEST'
999 STOP 'END-OF-FILE ENCOUNTERED ON INPUT TAPE IN HEATEST'
END
PROGRAM HEATEST 74/855 OPT=0. ROUND= A/5. N/-D. DS FYT 5.1+587 84/11/19. 13. 14. 29 PAGE 6

1 284 IF(NLAB_EQ.O)GO TO 120
1 285 24 CONTINUE
1 286 GO TO 100
1 287 END IF
1 288 C
1 289 C PROPAGATION TO TSTOP
1 290 C
1 291 120 DELT=TSTOP-TTEST
1 292 CALL TPS3(Delt)
1 293 TTEST=TSTOP
1 294 C WRITE SMOOTHED I.CS TO TAPE3
1 295 IF(SMIC)THEN
1 296 REWIND (3)
1 297 WRITE(3) UICSM,PICSM,SUSI
1 298 REWIND (3)
1 299 IIC=1
1 300 ENDIF
1 301 C
1 302 C EXIT TEMPERATURE/STATE ESTIMATION LOOP
1 303 C
1 304 IF(ITPRAM.LE.NRPTER.OR.NRPTER.EQ.0) THEN
1 305 C
1 306 C UPDATE PARAMETER ESTIMATES - LIST RESULTS
1 307 C
1 308 CALL PAREST
1 309 IT=1
1 310 DO 197 II=1,NPAR
1 311 IF(IFX(I).EQ.0)THEN
1 312 CI(I)=0.
1 313 GO TO 197
1 314 END IF
1 315 CI(I)=SORT(ABS(CIF(IT,IT)))
1 316 IT=IT+1
1 317 197 CONTINUE
1 318 C
1 319 3079 FORMAT(//1X,'ITER' ,T18, 'HALPH1', T30, 'HALPH2', T42, 'PHIC',
1 320 &T54, 'PHIK', T66, 'ZP', T78, 'Z', T88, 'ALPHA1', T100, 
1 321 &'ALPHA2', T112, 'ALPHA3'/78, 'QBCA', T18, 'OGLOGE', T30, 'QDELE', T42
1 322 &,'QDELDF', T54, 'QMAC', T88, 'PHIKB'/'1X,'(CRAMER-RAO BOUND))'
1 323 WRITE(6,3080)ITPRAM,(QP(I),I=1,10),(CI(I),I=1,10),
1 324 &OP(I),I=1,10),(CI(I),I=11,18)
1 325 3080 FORMAT(1X,12,2X,10(F8.5,4X)/5X,10(1X,'(',E8.2,''),1X)/
1 326 &/5X,8(F8.5,4X)/5X,6(1X,'(',E8.2,''),1X))
1 327 WRITE(6,3081)AERROR
1 328 3081 FORMAT(1X,'AVERAGE ERROR=','E12.5/)
1 329 C
1 330 END IF
1 331 198 CONTINUE
1 332 IIIC=1
1 333 C
1 334 C IF(SMIC)GO TO 988
1 335 C
1 336 IF(TSTOP .LT.TSTOP-1. -E.0) THEN
1 337 C RESET INITIAL CONDITIONS WITH FOLLOWING DATA TO DISK
1 338 C
1 339 REWIND 3
1 340 WRITE(3)USI,PC,SUSI
DO 500 IM=1,NPTSS
   UICSM(IM)=US1(IM)
   500 VEN(IM)=$RT(A,B(C(IM),IM))
   PRINT, 'UICSM(IM),IM+1,NPTSS'
   PRINT, 'VAR(IM),IM+1,NPTSS'
   ICOUNT=0
C PROPAGATION TO TRAJECTORY SAMPLE TIME/TIMES
    100 IF(T.LT.TUPD1.AND.T.LT.TSTOP)THEN
    1 DELT=T-TLEST
    2 CALL TPS3(DELT)
    3 CALL SENS(DELT)
    4 CALL TPSOSP2(DELT)
    5 TLEST=T
    6 DTP=DTP+DEL
    7 WHEN DTP.GE.DTPE Then WRITE TEMP/STATE ESTIMATES TO TAPE10/TAPE12
    8 IF((DTP.GE.DTPE).AND.(I:PRAM.GT.NRPITER)) THEN
    9 IF(IFPRINT)WRITE(80),TLEST,US1(TC(HBAR,HREF,ALPHA,T0)
   10 IF(IFPRINT)WRITE(10),TLEST,US1(1),EQUI,US1(NODES(1)),Q,
   11 &QREF,ALPHA,BETA,RENS,DELE,DELBF,WN)
   12 IF(IFPRINT)WRITE(12,3055)TLEST,US1(1),EQUI,US1(NODES(1)),TC(HBAR)
   13 &ALPHA,BETA,RENS,DELE,DELBF,WN,QN
   14 3055 FORMAT(991,1)
   15 TRITE=TLEST
   16 END IF
C READ NEXT TRAJECTORY SAMPLE
   DO 22 II=1,ITRUSK
   1 READ(4,END=899)NLAB,T,FLT1(I),I=1,NLAB
   2 IF(NLAB.EQ.0)T=TSTOP
   22 CONTINUE
C CALL HEATUN
   CALL HEATUN
   GO TO 100
C END IF
C PROPAGATION TO THERMOCouple SAMPLE TIME/TIMES
C IF(TUPD1.LE.TSTOP)THEN
C DLT=TUPD1-TLEST
C CALL TPS3(DELT)
C CALL SENS(DELT)
C CALL TPSOSP2(DELT)
C TLEST=TUPD1
C DTP=DTP+DEL
C IF(TUPD1.GE.T)THEN
C DO 23 II=1,ITRUSK
C READ(4,END=899)NLAB,T,FLT1(I),I=1,NLAB
C IF(NLAB.EQ.0)T=TSTOP
C 23 CONTINUE
C CALL HEATUN
C END IF
C KALMAN UPDATES
C
CALL ZERO(CIF,NPAR,NPAR)
IF(IIC.EQ.0) CALL ZERO(SUSI,NPTSS,NPAR)
C READ C & B FILE LABEL ON TRAJ TAPE
C READ(4)C,LIC,ITAIL,TITLE,ITEST,FLT,DFLT,DREQ,DCOM
33 IF(EQF(4).NE.0) CALL EXIT
120 IF(C .NE. CLAB) CALL EXIT
121 READ(4) LABEL,NRSECRET,NREM,(REMARK(I),I=1,NREM),NLAB,
122 *(PLAB1(I),PLAB2(I),NUM(I),I=1,NLAB)
123 IF (LABEL .NE. LABELX) CALL EXIT
124 C READ C & B FILE LABEL ON THE T/C MEAS TAPE
125 C READ(13)C,LIC,ITAIL,TITLE,ITEST,FLT,DFLT,DREQ,DCOM
126 44 IF(EQF(13).NE.0) CALL EXIT
127 IF(C .NE. CLAB) CALL EXIT
128 READ(13) LABEL,NRSECRET,NREM,(REMARK(I),I=1,NREM),NLAB,
129 *(PLAB1(I),PLAB2(I),NUM(I),I=1,NLAB)
130 IF (LABEL .NE. LABELX) CALL EXIT
131 C SET TEMPERATURE INITIAL CONDITIONS
132 IF(IIC.EQ.0) THEN
133 C INITIAL TEMPERATURES
134 1 139 DO 403 I=1,NPTSS
135 403 USI(I)=TINIT(I)
136 140 IF(IIFINCNT) THEN
141 READ (3) USI
142 END IF
143 END IF
144 IF(IIC.NE.0) THEN
145 REMIND 3
146 READ(3)USI,PC,SUSI
147 REMIND 3
148 END IF
150 C INITIALIZE SMOOTHER
151 C INITIALIZE PARAMETERS AT TSTART
152 C INITIALIZE PARAMETERS AT TSTART
153 DTP=0.
154 TLEST=TSTART
155 C READ FIRST TRAJECTORY SAMPLE
156 10 READ(4,END=999)NLAB,T,(FLT(I),I=1,NLAB)
157 IF(NLAB.EQ.0) GO TO 999
158 IF(T.LT.TSTART) GO TO 10
159 C CALCULATE REFERENCE HEATING
160 CALL HEATLN
161 C READ FIRST THERMOCOUPLE SAMPLES AND LOCAL PRESSURE
162 20 READ(13,END=999)NLAB,TUPDT,(TC(I),I=1,NTCT)
163 IF(NLAB.EQ.0) GO TO 999
164 IF(TUPDT.LT.TSTART) GO TO 20
165 C INITIALIZE THERMAL PROPERTIES/A MATRIX
166 CALL MAKEA
167 IF(IIC .EQ. 0) CALL IC
168 CALL QUEMAT
169 C
IF(UERMX.GT.ERALOW)WRITE(6,1000)ERALOW,N,T,TERMX,USI(1)
1000 FORMAT(1X,15HMAX ERROR TEMP..E12.6,2X,12.2X,4(E12.6,1X))
550 CONTINUE
570 RETURN
END
SUBROUTINE SEN5(DDT)
COMMON/COMUNIT/T,TAW1,ALPHA,H,V,RHO,P,TEMP,C,TRAD,RHOG,
&TS,TSMK,XFT,DEL,PDEL
COMMON /CHEAT/Q,TS,QREF,TM,MI,RENS,HBAR,HREF
COMMON/GOPT/PNPTSS,USI(6),PHI(6),NPT,PC(6),RR,
&QD(6),QDT(6),UX(6),A(6),RCX(6),RP(6),RM(6)
COMMON /ICTPS2/TINIT(1),ERALW,E
COMMON /CDS/IX(1)
COMMON/CSEN/SUSI(8),UM1(8)
LOGICAL FAUTO
DIMENSION FAUTO(7)
DIMENSION QP(5)
COMMON/CPAR/HQ,HQ2,HQ,F(HQ),HALF(2),PHIC,PHIK,ZP,Z,ALPH(2),KA,S(5),
&IFX(5),KAF,IXF(5),ACC(5),IFXSUM,NPAR,DALPH(2)
EQUIVALENCE (HQ,QP(1))
REAL M1
DIMENSION AA(8),BB(B8),CC(8),DD(6),AAA(8),CCC(6),DDD(8),W(6),G(6)
EQUIVALENCE (QD(1),1),AA(1),QD(1,2),BB(1),QD(1,3),CC(1))
&QD(1,4),DD(1),QD(1,5),AAA(1))
EQUIVALENCE (QD(1,1),CCC(1)),QD(1,2),DDD(1),(QD(1,3),W(1)),
&QD(1,4),G(1))
DATA SIG/4.78E-13/
DATA E/13/
DATA NPTS/40/
C
IF (DDT.EQ.0.0) GO TO 999
C BACKWARD-DIFFERENCE FORMULATION OF DIFF. EQS.
C
C SET UP TRIDIAGONAL MATRIX (COMMON TERMS ONLY)
DO 520 I=1,NPPTSS
BB(I)=RCX(I)-D/T GG+RP(I)
AA(I)=RM(I)
CC(I)=RP(I)
DD(I)=RCX(I)
520 CONTINUE
C
C I=1 SENSITIVITY FOR EACH PARAMETER
DO 530 IP=1,NPAR
IF(IFX(IP).EQ.0) GO TO 530
DO 531 IP=1,NPPTSS
530 CONTINUE
C
C SENSITIVITY FOR SPECIFIC HEAT FACTOR PHIC
SUBROUTINE SEN5(DT)
COMMON/COMUNIT/T,TAW1,ALPHA,H,V,RHO,P,TEMP,C,TRAD,RHOG,
&TS,TSMK,XFT,DEL,PDEL
COMMON /CHEAT/Q,TS,QREF,TM,MI,RENS,HBAR,HREF
COMMON/GOPT/PNPTSS,USI(6),PHI(6),NPT,PC(6),RR,
&QD(6),QDT(6),UX(6),A(6),RCX(6),RP(6),RM(6)
COMMON /ICTPS2/TINIT(1),ERALW,E
COMMON /CDS/IX(1)
COMMON/CSEN/SUSI(8),UM1(8)
LOGICAL FAUTO
DIMENSION FAUTO(7)
DIMENSION QP(5)
COMMON/CPAR/HQ,HQ2,HQ,F(HQ),HALF(2),PHIC,PHIK,ZP,Z,ALPH(2),KA,S(5),
&IFX(5),KAF,IXF(5),ACC(5),IFXSUM,NPAR,DALPH(2)
EQUIVALENCE (HQ,QP(1))
REAL M1
DIMENSION AA(8),BB(B8),CC(8),DD(6),AAA(8),CCC(6),DDD(8),W(6),G(6)
EQUIVALENCE (QD(1),1),AA(1),QD(1,2),BB(1),QD(1,3),CC(1))
&QD(1,4),DD(1),QD(1,5),AAA(1))
EQUIVALENCE (QD(1,1),CCC(1)),QD(1,2),DDD(1),(QD(1,3),W(1)),
&QD(1,4),G(1))
DATA SIG/4.78E-13/
DATA E/13/
DATA NPTS/40/
C
IF (DDT.EQ.0.0) GO TO 999
C BACKWARD-DIFFERENCE FORMULATION OF DIFF. EQS.
C
C SET UP TRIDIAGONAL MATRIX (COMMON TERMS ONLY)
DO 520 I=1,NPPTSS
BB(I)=RCX(I)-D/T GG+RP(I)
AA(I)=RM(I)
CC(I)=RP(I)
DD(I)=RCX(I)
520 CONTINUE
C
C I=1 SENSITIVITY FOR EACH PARAMETER
DO 530 IP=1,NPAR
IF(IFX(IP).EQ.0) GO TO 530
DO 531 IP=1,NPPTSS
530 CONTINUE
C
C SENSITIVITY FOR SPECIFIC HEAT FACTOR PHIC
SUBROUTINE SEN5(DT)
COMMON/COMUNIT/T,TAW1,ALPHA,H,V,RHO,P,TEMP,C,TRAD,RHOG,
&TS,TSMK,XFT,DEL,PDEL
COMMON /CHEAT/Q,TS,QREF,TM,MI,RENS,HBAR,HREF
COMMON/GOPT/PNPTSS,USI(6),PHI(6),NPT,PC(6),RR,
&QD(6),QDT(6),UX(6),A(6),RCX(6),RP(6),RM(6)
COMMON /ICTPS2/TINIT(1),ERALW,E
COMMON /CDS/IX(1)
COMMON/CSEN/SUSI(8),UM1(8)
LOGICAL FAUTO
DIMENSION FAUTO(7)
DIMENSION QP(5)
COMMON/CPAR/HQ,HQ2,HQ,F(HQ),HALF(2),PHIC,PHIK,ZP,Z,ALPH(2),KA,S(5),
&IFX(5),KAF,IXF(5),ACC(5),IFXSUM,NPAR,DALPH(2)
EQUIVALENCE (HQ,QP(1))
REAL M1
DIMENSION AA(8),BB(B8),CC(8),DD(6),AAA(8),CCC(6),DDD(8),W(6),G(6)
EQUIVALENCE (QD(1),1),AA(1),QD(1,2),BB(1),QD(1,3),CC(1))
&QD(1,4),DD(1),QD(1,5),AAA(1))
EQUIVALENCE (QD(1,1),CCC(1)),QD(1,2),DDD(1),(QD(1,3),W(1)),
&QD(1,4),G(1))
DATA SIG/4.78E-13/
DATA E/13/
DATA NPTS/40/
C
IF (DDT.EQ.0.0) GO TO 999
C BACKWARD-DIFFERENCE FORMULATION OF DIFF. EQS.
C
C SET UP TRIDIAGONAL MATRIX (COMMON TERMS ONLY)
DO 520 I=1,NPPTSS
BB(I)=RCX(I)-D/T GG+RP(I)
AA(I)=RM(I)
CC(I)=RP(I)
DD(I)=RCX(I)
520 CONTINUE
C
C I=1 SENSITIVITY FOR EACH PARAMETER
DO 530 IP=1,NPAR
IF(IFX(IP).EQ.0) GO TO 530
DO 531 IP=1,NPPTSS
530 CONTINUE
C
C SENSITIVITY FOR SPECIFIC HEAT FACTOR PHIC
SUBROUTINE SENS

IF(IP.EQ.4) THEN
  DDD(I)=DDD(I)-RP(I)+(USI(2)-USI(I))-(HBAR*HREF*(TAM1-
  AUSI(I)))*E*SIG*(((USI(I)+480.)***4-TRAD**4)/PHIC
  DO 533 I=2,NPTSS-1
  533  DDD(I)=DDD(I)-(RM(I)*(USI(I-1)-USI(I))+RP(I)*(USI(I)-USI(1)
  END IF

C SENSITIVITY FOR CONDUCTIVITY FACTOR PHIK

IF(IP.EQ.0) THEN
  DDD(I)=DDD(I)-RP(I)*(USI(2)-USI(I))/PHIK
  DO 534 I=2,NPTSS-1
  534  DDD(I)=DDD(I)+((RM(I)*USI(I)-1)/(RM(I)*RP(I)))*USI(I)+RP(I)
  END IF

C TRIDIAGONAL SOLUTION

C ELIMINATION STEP

G(I)=DDD(I)
  W(I)=CC(I)
  DO 536 I=2,NPTSS
  W(I)=CC(I)/(1.+AAA(I)*W(I-1))
  536  G(I)=DDD(I)+AAA(I)*G(I-1)/(1.+AAA(I)*W(I-1))

C BACKWARD SUBSTITUTION

SUSI(NPTSS,IP)=G(NPTSS)
  DO 537 L=2,NPTSS
  SUSI(I,IP)=G(I)-W(I)*SUSI(I+1,IP)
  537  CONTINUE

530 CONTINUE

999 RETURN

END
SUBROUTINE PAREST
"LOGICAL FAUTO"
"DIMENSION FAUTO(7)"
"DIMENSION SP(5)"
"COMMON/SPARE/SP,H(2),PHIC,PHIK,ZP,Z,ALPH(2),KA,S(5),"&"CIF(5,5),KAF,IFX(5),ACC(5),IFXSUM,NPAR,DALPH(2),"EQUIVALENCE (HO,QP(1))"
"COMMON/ICFSPSC/TINIT1,ERALOW,E common/CDX/DX(1)"
"COMMON/CMTIME/TSTART,TSTOP,Treator,NRPITER,ITPRAM"
"COMMON/MAIN2/IMA2"
"DIMENSION CIF1(6,6)"
"DATA KIN,KOUT/5,6/"
"IMA2=16"
"NR=IFXSUM"
"IF(NR.EQ.1) THEN"
"CIF(1,1)=1./CIF(1,1)"
"GO TO 20"
"END IF"
"C"
"C INVERT CONDITIONAL INFORMATION MATRIX,CIF"
"CALL GMINV(NR,NR,CIF1,CIF,MN,MR,1)"
"DO 15 IR=1,NR"
"DO 15 IC=1,NR"
"CIF(IR,IC)=CIF(I,IC)"
"IT=1"
"DO 29 IP=1,NPAR"
"IF(IFX(IP).EQ.0) GO TO 29"
"JT=1"
"DO 30 JP=1,NPAR"
"IF(IFX(JP).EQ.0) GO TO 29"
"QP(IP)-QP(IP)+CIF(IT,JT)*S(JT)"
"JT=JT+1"
"28 CONTINUE"
"IT=IT+1"
"29 CONTINUE"
"RETURN"
"END"
SUBROUTINE KF

DO 90 I=1,NPTSS
DO 5 ITT=1,NCT
IF(I.EQ.NODES(IITT))THEN
  NODE=NODES(IITT)
  UMEAS=TC(IITT)
  GO TO 10
END IF
CONTINUE
GO TO 98

ERROR=UMEAS-UAP(NODE)

IF(KAF.EQ.1) AERROR=ERROR
AERROR=((KAF-1)*AERROR+ERROR)/KAF
KAF=KAF+1

SCORE RUNNING SUMS FOR JACOBIAN OF LIKELIHOOD FN. S.
AND CONDITIONAL INFORMATION MATRIX, CIF

R=(SDMEA+UMEAS)**2.

DO 28 KO=1,NPAR
  S1(KO)=SUSI(NODE,KO)+IFX(KO)*ERROR/(PC(NODE,NODE)+R)
  S2(KO)=SUSI(NODE,L)+IFX(L)*SUSI(NODE,L)/(PC(NODE,NODE)+R)
  DO 25 L=1,NPAR
    IFX(KO,L)=S1(KO)+S2(L)+IFX(KO,L)
    IFX(L,KO)=S1(L)+S2(KO)+IFX(L,KO)
  DO 25 L=1,NPAR
  DO 28 KO=1,NPAR
SUBROUTINE KF
56 26 CONTINUE
57 IT = IT + 1
58 DO 29 IP = 1, NPAR
59 IF (IFX(IP).EQ.0) GO TO 29
60 S(IT) = S(IT) + S(IT)
61 J = J + 1
62 DO 28 JP = 1, NPAR
63 IF (IFX(JP).EQ.0) GO TO 28
64 CIF(IT, JP) = D(IJ(IP, JP) + CIF(IT, JP))
65 J = J + 1
66 28 CONTINUE
67 IT = IT + 1
68 29 CONTINUE
69 C COMPUTE KALMAN GAIN, K
70 C DO 30 IK = 1, NPTSS
71 C 30 K(IK) = PAP(IK, NODE)/(PAP(NODE, NODE) + R)
72 C IF KOPT = 1 UPDATE
73 C IF KOPT = 2 UPDATE EXCEPT ON LAST ITERATION(ITPRAM-NRPITER)
74 C IF KOPT = 3 DO NOT UPDATE
75 C IF KOPT = 4 UPDATE COVARIANCE AND SENSITIVITY ONLY
76 C IF KOPT = 5 ONLY UPDATE TEMP ON LAST ITERATION
77 C IF KOPT = 6 ONLY UPDATE ON LAST ITERATION
78 C GO TO 101, 102, 103, 104, 101, 101)KFPOT
79 C 101 IF (ITPRAM.GT.0NPITER) GO TO 103
80 C 102 IF (ITPRAMNRPITER) GO TO 103
81 C STATE UPDATE
82 C 103 CONTINUE
83 C DO 40 IO = 1, NPTSS
84 C 40 USI(IO) = USI(IO) + K(IO)*(UMEAS-UAP(NODE))
85 C SENSITIVITY UPDATE
86 C 104 CONTINUE
87 C DO 35 IP = 1, NPAR
88 C 35 SUSI(L, IP) = SUSL(IP, L) - K(L)*SUSI(NODE, IP)
89 C COVARIANCE UPDATE, PC - JOSEPH FORM
90 C NPTSS NPTPC
91 C CALL ZERO(QD(1,1), NPTSS, NPTSS)
92 C DO 50 IC = 1, NPTSS
93 C 50 QD(IC, IC) = QD(IC, NODE) * QD(IC, IC)
94 C CALL MAT4(NPTSS, NPTSS, PC(1,1), QD(1,1), QDT(1,1))
95 C CALL MAT4(NPTSS, 1, R, K(1), QD(1,1))
96 C DO 55 IPC = 1, NPTSS
97 C 55 PC(IPC, JPC) = QDT(IPC, JPC) + QD(IPC, JPC)
98 C NPTSS = INP
99 C CONTINUE
100 C RETURN
SUBROUTINE FPSM(START)  HAROLD 39
COMMON /CONST/ XP(13)  HAROLD 40
COMMON /CTCMHT/ NTCT  OCT 10  13
COMMON /CSMTH/ UICSM(6), PICSM(6,6), UAP(6), PAP(6,6),  UPDCT09 1
&SMIC,TSMTH,W(6,6)  UPDCT09 2
6  LOGICAL SMIC  FPMHT  4
7 COMMON/COSP/NPTSS, USI(6), PHI(6,6), NPT, PC(6,6), RR,  UPDCT09 6
&OQ(6,6), QT(6,6), QUE(6,6), A(6,6), RCX(6,6), RP(6), RM(6)  UPDCT09 5
9 COMMON/CK(6), S(6,6), J1(6,6), TC2() NOD(2), KFDPT  UPDCT09 8
10 REAL K, J1, J2  FDPT  3
11 DIMENSION RINV(6,6), HTR(6,6), SPF(6,6), GAIN(6)  UPDCT09 3
&WRK(I,6), WRK2(6,6)  UPDCT09 4
13 COMMON /MAINI/NPT, IPVT(6), WORK(6)  OCT 10  4
14 IPN=6  OCT 10  5
15 C  FPSMIC  8
16 C THIS ROUTINE IS A FIXED POINT SMOOTHER ALGORITHM  FPSMIC  9
17 C CALL ZERO(SFP, NPTSS, NPTSS)  HAROLD 42
18 DO 10 I=1, NTCT  HAROLD 43
10 SFN(NODES(I), NODES(I))=1./(RR+TC(I))**2.  HAROLD 44
C  FPSMIC  25
22 C FORM I-SP AND FIND PHII  FPSMIC  26
23 CALL WMUL(SFP, PC, NPTSS, NPTSS, WRK1)  FPSMIC  27
24 DO 30 I=1, NPTSS  FPSMIC  28
25 DO 30 J=1, NPTSS  FPSMIC  29
26 WRK1(I, J)=WRK1(I, J)  FPSMIC  30
27 IF(I.EQ. J) WRK1(I, J)=1.0+WRK1(I, J)  FPSMIC  31
30 WRK2(I, J)=PHI(J, I)  HAROLD 45
29 DO 35 I=1, NTPTSS  HAROLD 46
30 DO 35 J=1, NPTSS  HAROLD 47
35 PHI(I, J)=WRK2(I, J)  HAROLD 48
C  FPSMIC  33
33 C FORM W=W+PHII*(I-SP)  FPSMIC  34
34 CALL WMUL(PHI, WRK1, NPTSS, NPTSS, NPTSS, WRK2)  CHUCK1  4
35 CALL WMUL(W, WRK2, NPTSS, NPTSS, WRK1)  FPSMIC  36
36 DO 40 I=1, NPTSS  FPSMIC  37
37 DO 40 J=1, NPTSS  FPSMIC  38
40 W(I, J)=WRK1(I, J)  FPSMIC  39
C  FPSMIC  40
40 C SOLVE FOR COVARIANCE -- P=P-W*(S+PAP*S+ S)^TRAN  FPSMIC  41
41 CALL WMUL(PAP, SFP, NPTSS, NPTSS, NPTSS, WRK1)  FPSMIC  42
42 CALL WMUL(SFP, WRK1, NPTSS, NPTSS, NPTSS, WRK2)  FPSMIC  43
43 DO 50 I=1, NPTSS  FPSMIC  44
44 DO 50 J=1, NPTSS  FPSMIC  45
50 WRK1(I, J)=WRK2(I, J)  FPSMIC  48
46 CALL TRI(NPTSS, WRK1, W, PHI, WRK2, NPTSS)  HAROLD 49
47 DO 60 I=1, NPTSS  FPSMIC  48
48 DO 60 J=1, NPTSS  FPSMIC  49
60 PICSM(I, J)=PICSM(I, J)-WRK2(I, J)  FPSMIC  50
C  FPSMIC  51
50 C SOLVE FOR SMOOThED STATE (SCALAR UPDATES)  FPSMIC  52
52 DO 150 I=1, NPTSS  FPSMIC  53
53 DO 110 ITT=1, NTCT  FPSMIC  54
54 IF(I.EQ. NODES(ITT)) THEN  FPSMIC  55
55 NODE=NODES(ITT)  FPSMIC  56
SUBROUTINE FPSM
74/055 OPT=O, ROUND= A/ S/ W/- D/ S FYN S.1:587 84/11/19. 13 14.29 PAGE 2

1 58 UMEAS=TC(ITT)
1 57 GO TO 120
1 58 ENDIF
59 110 CONTINUE
60 150 GO TO 150
61 120 R=(R*R+UMEAS)**2.
62 C
63 C COMPUTE GAIN
64 DO 121 IJ=1,NPTSS
65 121 GAIN(IJ)=W(IJ,NODE)/R
66 C
67 C UPDATE
68 DO 125 IO=1,NPTSS
69 TCONST=TSTART*XP1(IO)
70 IF(TSMTH.GT. TCONST) GO TO 125
71 125 CONTINUE
72 CONTINUE
73 RETURN
74 END
SUBROUTINE MULT (X,Y,N,Z,N2)
DIMENSION X(N2,N2),Y(N2,N2),Z(N2,N2)
DO 20 I=1,N
DO 20 J=1,N
DO 20 K=1,N
20 Z(I,J)=Z(I,J)+X(I,K)*Y(K,J)
RETURN
END
SUBROUTINE MEXP(N, SUB1, TIME, SUB2, Q, QT, N2)
DIMENSION SUB1(N2,N2), SUB2(N2,N2)
C
MULTIPLY ELEMENTS OF SUB1 BY TIME
DO 102 I=1,N
DO 102 J=1,N
SUB1(I,J)=SUB1(I,J)*TIME
102 SUB2(I,J)=SUB1(I,J)
C
GENERATE IDENTITY MATRIX FOR INPUT Q, FOR HQR
DO 30 I=1,N
DO 30 J=1,N
Q(I,J)=0
30 IF(I.EQ.J) Q(I,J)=1.
C
CALL HQR(N, SUB2, Q, IERR, N2)
C
MATRIX SUB2 HAS BEEN DESTROYED
C
Q IS NOW AN ORTHOGONAL TRANSFORMATION MATRIX
DO 40 I=1,N
DO 40 J=1,N
40 QT(I,J)=Q(J,I)
C
QT IS NOW THE TRANSPOSE AND THE INVERSE, OF Q
CALL MULTI(QT, SUB2, N, SUB1, N2)
C
SUB1 NOW CONTAINS THE TRIANGULAR MATRIX QT*A*Q
DO 50 I=1,N
50 SUB2(I,J)=0.
C
CALL FUNCT(1, N, SUB1, SUB2, N2)
C
SUB2 NOW HOLDS EXP(A*TIME) IN TRIANGULAR FORM
CALL MULTI(SUB2, QT, N, SUB1, N2)
CALL MULTI(QT, SUB1, N, SUB2, N2)
C
SUB2 NOW HOLDS EXP(A*TIME) IN ORIGINAL BASIS FORM
RETURN
END
SUBROUTINE FUNCT(R,S,T,F,MM)
DIMENSION T(MM,MM),F(MM,MM)
INTEGER R,S
REAL EXP
DO 10 I=R,S

C THE IF-BLOCK GIVES 14-DIGIT ACCURACY WITHOUT UNDERFLOW
IF( T(I,I) .LT. -43 ) THEN
  F(I,I)=0.
ELSE
  F(I,I)=EXP( T(I,I) )
END IF
10 CONTINUE

C PROCESS THE KTH SUPERDIAGONAL
N=S-R+1
NN=N-1

C NN = NUMBER OF SUPERDIAGONALS IN THE BLOCK
IF(NN .EQ. 0) RETURN
DO 13 K=1,NN
LL=S-K

DO 12 I=R,LL
   DIFF=T(I,I)-T(I+K,I+K)
   IF(ABS(DIFF) .EQ. 0.0) GO TO 14
   G=T(I,I+K)*F(I,I)-F(I+K,I+K)
   KK=K-1
   IF(KK .EQ. 0) GO TO 12
   DO 11 M=1, KK
      12 F(I,I+K)=G/DIFF
   13 CONTINUE
RETURN
14 MM=MM
RETURN
END
SUBROUTINE HQR(IGH,H,Z,IERR,N2)
INTEGER I,J,K,L,M,N,EN,II, JJ, LL, MM, NA, NM, NN, N2,
     & H, ITS, LOW, MP2, EM2, IERR, MING
REAL H(N2,N2), Z(N2,N2)
REAL P, Q, R, S, T, W, X, Y, RA, SA, VI, VR, ZZ, NORM
REAL MACHEP, SQRT, ABS, SIGN, REAL, AIMAG
LOGICAL NOTLAS
COMPLEX Z3, CMPLX

MACHEP IS A PARAMETER THAT SPECIFIES PRECISION
MACHEP = 0.00000000000001
NM = IGH
N' = IGH
LOW = 1
IERR = 0
NORM = 0

COMPUTE MATRIX NORM
DO 50 I = 1, N
    DO 40 J = K, N
        40 NORM = NORM + ABS(H(I,J))
    K = I
    50 CONTINUE
    EN = IGH
    1 = 0.0
    0
    28
    29
    30
    31
    32
    33
    34
    35
    36
    37
    38
    39
    40
    41
    42
    43
    44
    45
    46
    47
    48
    49
    50
    51
    52
    53
    54
    55
    56

    100 X = H(EN,EN)
    100 IF(X.EQ.0) GO TO 270
    100 Y = H(NA,NA)
    100 W = H(EN,NA) * H(NA,EN)
    100 IF(L.EQ.0) GO TO 280
    100 IF(LS.EQ.30) GO TO 1000
    100 IF(LS.EQ.10 .AND. ITS .NE. 20) GO TO 130
    100 *** FORM EXCEPTIONAL SHIFT ***
        100 T = X
        120 H(I,I) = H(I,I) - X
        120 S = ABS(H(EN,NA)) + ABS(H(NA,ENM2))
        120 X = 0.75 * S
    120 DO 120 I = LOW, EN
    120 H(I,I) = H(I,I) - X
    120 S = ABS(H(EN,NA)) + ABS(H(NA,ENM2))
    120 X = 0.75 * S
    120 DO 120 I = LOW, EN
*** LOOK FOR TWO CONSECUTIVE SMALL SUB-DIAGONAL ELEMENTS***

170 ITS = ITS + 1

*** FOR M = EN-2 STEP -1 UNTIL L DO ***

DO 140 MM = L, ENM2
    M = ENM2 + L - MM
    ZZ = H(M,M)
    R = X - ZZ
S = Y - ZZ
    P = (R*S - W) / H(M+1,M) + H(M,M+1)
    Q = H(M+1,M+1) - ZZ - R - S
    R = H(M+2,M+1)
    S = ABS(P) + ABS(Q) + ABS(R)
    P = P/S
    Q = Q/S
    R = R/S
    IF(M.EQ.L) GO TO 150
    IF(ABS(H(M,M-1)) + ABS(Q) + ABS(R)) .LE. MACHEP*ABS(P)
        Y = ABS(H(M-1,M-1)) + ABS(ZZ) + ABS(H(M+1,M+1))) GO TO 150

140 CONTINUE

150 MP2 = M + 2

DO 160 I = MP2, EN
    H(I,I-2) = 0.0
    IF(I.EQ.MP2) GO TO 160
    H(I,I-2) = 0.0

160 CONTINUE

C * DOUBLE OR STEP INVOLVING ROWS L TO EN AND COLUMNS M TO EN *

DO 280 K = M, NA
    NOTLAS = K .NE. NA
    IF(K .EQ. M) GO TO 170
    P = H(K,K-1)
    Q = H(K+1,K-1)
    R = 0.0
    IF(NOTLAS) R = H(K+2,K-1)
    X = ABS(P) + ABS(Q) + ABS(R)
    IF(X .EQ. 0.0) GO TO 260
    P = P/X
    Q = Q/X
    R = R/X

170 S = SIGN(SORT(P*P+Q*Q+R*R),P)
    IF(K.EQ.M) GO TO 180
    H(K,K-1) = -S*X
    GO TO 190

180 IF(K .EQ. M) H(K,K-1) = -H(K,K-1)

190 P = P + S
    X = P/S
    Y = Q/S
    ZZ = R/S
    Q = Q/P
    R = R/P

*** ROW MODIFICATION ***

10 210 J = K, N
    P = H(K,J) + Q = H(K+1,J)
    IF(NOT. NOTLAS) GO TO 200
    P = P + R * H(K+2,J)
    H(K+2,J) = H(K+2,J) - P * ZZ

H(K,J) = P * ZZ

200 GO TO 210

END
SUBROUTINE HQR

200 H(K+1,J) = H(K+1,J) - P * Y
201 H(K,J) = H(K,J) - P * X

210 CONTINUE

J = MIN0(EN,K+3)

C *** COLUMN MODIFICATION ***

DO 230 I = 1, J

P = X * H(I,K) + Y = H(I,K+1)
IF ( NOT .NOT. NOTLAS) GO TO 220

212 P = P + ZZ = H(I,K+2)

213 H(I,K+1) = H(I,K+1) - P * Q

214 H(I,K) = H(I,K) - P

215 CONTINUE

C *** ACCUMULATE TRANSFORMATIONS ***

DO 250 I = LOW, IGH

P = X * Z(I,K) + Y * Z(I,K+1)
IF ( NOT .NOT. NOTLAS) GO TO 240

220 P = P + ZZ = Z(I,K+2)

221 Z(I,K+1) = Z(I,K+1) - P * R

222 Z(I,K) = Z(I,K) - P

223 CONTINUE

C *** ONE ROOT FOUND ***

270 H(EN,EN) = X + T

271 EN = NA

272 GO TO 80

C *** TWO ROOTS FOUND ***

280 P = (Y - X)/ 2.0

281 Q = P + P + W

282 ZZ = SQRT(ABS(Q))

283 H(EN,EN) = X + T

284 X = H(EN,EN)

285 H(NA,NA) = Y + T

286 IF (Q .LT. 0.0) GO TO 320

C *** REAL PAIR ***

290 ZZ = P + SIGN(ZZ,P)

291 X = H(EN,NA)

292 S = ABS(X) + ABS(ZZ)

293 P = X / S

294 Q = ZZ / S

295 R = SQRT(P * P + Q * Q)

296 P = P / R

297 Q = Q / R

C *** ROW MODIFICATION ***

DO 280 J = NA, N

281 ZZ = H(NA,J)

282 H(NA,J) = Q * ZZ + P * H(EN,J)

283 H(EN,J) = Q * H(EN,J) - P * ZZ

284 CONTINUE

C *** COLUMN MODIFICATION ***

DO 300 I = 1, EN

290 ZZ = H(I,NA)

291 H(I,NA) = Q * ZZ + P * H(I,EN)

292 H(I,EN) = Q * H(I,EN) - P * ZZ

300 CONTINUE
SUBROUTINE HQR

170 C *** ACCUMULATE TRANSFORMATIONS ***
171 DO 310 I = LOW, IGH
172 ZZ = Z(I,NA)
173 Z(I,NA) = Q*ZZ + P*Z(I,EN)
174 Z(I,EN) = Q*Z(I,EN) - P*ZZ
175 CONTINUE
176 GO TO 330
177 C *** COMPLEX PAIR ***
178 CONTINUE
179 GO TO 330
180 GO TO 330
181 C * SET ERROR - NO CONVERGENCE TO EIGENVALUE AFTER 30 ITERATIONS
182 1000 IERR = EN
183 1001 RETURN
184 END
SUBROUTINE MULT2(N, X, Y, Z, N2)

C Computes Z = X * Y

DIMENSION X(N2, N2), Y(N2, N2), Z(N2, N2)

DO 20 I = 1, N

DO 20 J = 1, N

Z(I, J) = 0.

20 Z(I, J) = Z(I, J) + X(I, K) * Y(J, K)

RETURN

END
SUBROUTINE TRI(N,Q,X,Z,W,Z,W2)
  C COMPUTES Z=XQX^T
  DIMENSION Q(N2,N2),X(N2,N2),W(N2,N2),Z(N2,N2)
  CALL MULT(X,Q,W,Z2)
  C X=Q IS STORED IN W
  CALL MULT2(N,W,X,Z2)
  C Z=W*X^T
  RETURN
END
SUBROUTINE SGFFA(A, LDA, N, IPVT, INFO)
INTEGER LDA, N, IPVT(1), INFO
REAL A(LDA, 1)

C SGEFA FACTORS A REAL MATRIX BY GAUSSIAN ELIMINATION.
C ON ENTRY:
C A: THE MATRIX TO BE FACTORED
C LDA: THE LEADING DIMENSION OF THE ARRAY A
C N: THE ORDER OF THE ARRAY A
C A: AN UPPER TRIANGULAR MATRIX AND THE MULTIPLIERS
C WHICH WERE USED TO OBTAIN IT.
C IPVT: AN INTEGER VECTOR OF PIVOT INDICES
C INFO: = 0 NORMAL VALUE.
C = K IF U(K,K).EQ. 0.0 THIS IS NOT AN ERROR
C CONDITION FOR SGEFA, BUT INDICATES THAT
C SGEDI WILL DIVIDE BY ZERO WHEN CALLED.
C THIS IS FROM LINPACK USER'S GUIDE, VERSION 08/14/78
C REAL T
INTEGER ISMAX, J, K, KP1, L, NM1

C GAUSSIAN ELIMINATION WITH PARTIAL PIVOTING
C INFO = 0
C NM1 = N-1
C IF(NM1.LT. 1.) GO TO 70
C DO 60 K=1,NM1
C KP1 = K+1

C FIND L = PIVOT INDEX
C L = ISMAX(N-K+1, A(K,K), 1)+K-1
C IPVT(K) = L

C ZERO PIVOT IMPLIES THIS COLUMN IS TRIANGULARIZED
C IF(A(L,K).EQ. 0.0E0) GO TO 40

C INTERCHANGE IF NECESSARY
C IF(L .EQ. K) GO TO 10
C T = A(L,K)
C A(L,K) = A(K,K)
C A(K,K) = T
C 10 CONTINUE

C COMPUTE MULTIPLIERS
C T = -1.0E0/A(K,K)
C CALL SSCAL(N-K,T,A(K+1,K),1)

C ROW ELIMINATION WITH COLUMN INDEXING
C DO 30 J=KP1,N
C T = A(L,J)
C IF(L .EQ. K) GO TO 20
C A(L,J) = A(K,J)
C 20 CONTINUE
CALL SAXPY(N-K,T,A(K+1,K),1,A(K+1,J),1)  SGEFA 57
30 CONTINUE  SGEFA 58
GO TO 50  SGEFA 59
40 CONTINUE  SGEFA 60
INFO= K  SGEFA 61
50 CONTINUE  SGEFA 62
60 CONTINUE  SGEFA 63
70 CONTINUE  SGEFA 64
IPVT(N)= M  SGEFA 65
IF(A(N,N) .EQ. 0.000) INFO= N  SGEFA 66
RETURN  SGEFA 67
END  SGEFA 68
SUBROUTINE SGEDI(A,LDA,N,IPVT,WORK)

INTEGER LDA,N,IPVT(1)
REAL A(LDA,1),WORK(1)

C
C SGEDI COMPUTES INVERSE OF MATRIX A USING
C FACTORS COMPUTED BY SGFA.

C ON ENTRY:
A: THE OUTPUT FROM SGFA, REAL(LDA,N)
LDA: THE LEADING DIMENSION OF ARRAY A
N: THE ORDER OF MATRIX A
IPVT: THE PIVOT VECTOR FROM SGFA, INTEGER(N)
WORK: WORK VECTOR, CONTENTS DESTROYED, REAL(N)

C ON RETURN:
A: INVERSE OF THE ORIGINAL MATRIX
ERROR CONDITION: A DIVISION BY ZERO WILL OCCUR IF THE
INPUT FACTOR CONTAINS A ZERO ON THE DIAGONAL.
IT WILL NOT OCCUR IF SGFA HAS SET INFO=0

C THIS IS FROM LINPACK USER'S GUIDE, VERSION 08/14/78
REAL T
INTEGER I,J,K,KB, KP1,L,NM1

C
C COMPUTE INVERSE

DO 100 K=1,N
A(K,K)= 1.0E0/A(K,K)
T= -A(K,K)
CALL SCAL(K-1,T,A(1,K),1)
KP1= K+1
IF(N .LT. KP1) GO TO 80
DO 30 J=KP1,N
T= A(K,J)
A(K,J)= 0.0E0
CALL SAXPY(K,T,A(1,K),1,A(1,J),1)

80 CONTINUE
90 CONTINUE
100 CONTINUE

C
C FORM INVERSE(U)=INVERSE(L)

NM1= N-1
IF(NM1 .LT. 1) GO TO 140
DO 130 KB=1,NM1
K= N-KB
KP1= K+1
DO 120 I=KP1,N
WORK(I)= A(I,K)
C A(K,K)= 0.0E0
110 CONTINUE
120 CONTINUE
L= IPVT(K)
FUNCTION ISAMAX    74/855  OPT=O,ROUND= A/ S/ M/-D,-DS  FTN 5.1-587  84/11/15  13.14.29  PAGE 1

DO=-LONG/-DT,ARG=-COMMON/-FIXED,CS= USER/-FIXED,DS= -TB/-SB/-SL/ ER/-ID/- PMD/-ST,PL=5000
FTNS.I,ANSI=O,L=OUTS.LO=S/-A.

1    INTEGER FUNCTION ISAMAX(N,SX,INCX)    ISAMAX  2
2    C ISAMAX FINDS INDEX OF ELEMENT WITH MAX. ABSOLUTE VALUE.    ISAMAX  3
3    C LINPACK USER'S GUIDE, VERSION 03/11/78    ISAMAX  4
4    REAL SX(1),SMAX    ISAMAX  5
5    INTEGER I,INCX,IX,N    ISAMAX  6
6    ISAMAX= 0    ISAMAX  7
7    IF(N.LT. 1) RETURN    ISAMAX  8
8    ISAMAX= 1    ISAMAX  9
9    IF(N.EQ. 1) RETURN    ISAMAX 10
10   IF(INCX.EQ. 1) GO TO 20    ISAMAX 11
11   C    ISAMAX 12
12   C CODE FOR INCREMENT NOT EQUAL TO 1    ISAMAX 13
13   IX= 1    ISAMAX 14
14   SMAX= ABS(SX(1))    ISAMAX 15
15   IX= IX+INCX    ISAMAX 16
16   DO 10 I=2,N    ISAMAX 17
17   IF(ABS(SX(IX)).LE.SMAX) GO TO 5    ISAMAX 18
18   ISAMAX= I    ISAMAX 19
19   SMAX= ABS(SX(IX))    ISAMAX 20
20   5 IX= IX+INCX    ISAMAX 21
21   10 CONTINUE    ISAMAX 22
22   RETURN    ISAMAX 23
23   C    ISAMAX 24
24   C CODE FOR INCREMENT EQUAL TO 1    ISAMAX 25
25   20 SMAX= ABS(SX(1))    ISAMAX 26
26   DO 30 I=2,N    ISAMAX 27
27   IF(ABS(SX(I)).LE.SMAX) GO TO 30    ISAMAX 28
28   ISAMAX= I    ISAMAX 29
29   SMAX= ABS(SX(I))    ISAMAX 30
30   30 CONTINUE    ISAMAX 31
31   RETURN    ISAMAX 32
32   END
SUBROUTINE SAXPY(N, SA, SX, INCX, SY, INCY)
C CONSTANT TIMES A VECTOR PLUS A VECTOR
C USES UNROLLED LOOP FOR INCREMENTS=1.
C FROM LINPACK USER'S GUIDE, VERSION 03/11/78
REAL SX(1), SY(1), SA
INTEGER I, INCX, INCY, IX, IY, M, MP1, N
IF(N .LE. 0) RETURN
IF(SA .EQ. 0.0) RETURN
IF(INCX .EQ. 1 .AND. INCY .EQ. 1) GO TO 20
C CODE FOR UNEQUAL INCREMENTS OR FOR
C EQUAL INCREMENTS NOT EQUAL TO 1
IX = 1
IY = 1
IF(INCX.LT.0) IX = ((N+1)*INCX +1
IF(INCY.LT.0) IY = ((N+1)*INCY +1
DO 10 I = 1, N
SY(I) = SY(I) + SA*SX(IX)
IX = IX + INCX
IY = IY + INCY
10 CONTINUE
RETURN
C CODE FOR BOTH INCREMENTS EQUAL TO 1
C CLEAN-UP LOOP
20 M = MOD(N,4)
IF(M .EQ. 0) GO TO 40
DO 30 I = 1, N
SY(I) = SY(I) + SA*SX(I)
30 CONTINUE
IF(N .LT. 4) RETURN
40 MP1 = M + 1
DO 50 I = MP1, N, 4
SY(I) = SY(I) + SA*SX(I)
50 CONTINUE
RETURN
END
SUBROUTINE SSCALL(N,SA,SX,INCX)
C SCALES A VECTOR BY A CONSTANT.
C USES UNROLLED LOOPS FOR INCREMENT EQUAL TO 1.
C LINPACK USER'S GUIDE, VERSION 03/11/78
C
REAL SA,SX(I)
INTEGER I,INCX,M,MP1,N,NINCX
IF(N.LE.0) RETURN
IF(INCX.EQ.1) GO TO 20
CODE FOR INCREMENT NOT EQUAL TO 1
NINCX= N*INCX
DO 10 I=1,NINCX,INCX
SXI= SA*SXI
10 CONTINUE
RETURN
CODE FOR INCREMENT EQUAL TO 1.
CLEAN-UP LOOP
20 M= MOD(N,B)
IF(N.EQ.0) GO TO 40
DO 30 I=1,M
SXI= SA*SXI
30 CONTINUE
IF(N.LT.5) RETURN
40 MP1= M+1
DO 50 I=MP1,N,B
SXI= SA*SXI
50 CONTINUE
RETURN
END
SUBROUTINE SSSAP(N, SX, INCX, SY, INCY)

C INTERCHANGES TWO VECTORS.
C USES UNROLLED LOOPS FOR INCREMENTS EQUAL TO 1.
C LINPACK USER'S GUIDE, VERSION 03/11/78

REAL SX(1), SY(1), STEM
INTEGER I, IX, INCX, IX, IY, M, MP1, N
IF(N.LE.0) RETURN
IF(INCX.EQ.1 .AND. INCY.EQ.1) GO TO 20

C CODE FOR UNEQUAL INCREMENTS OR EQUAL INCREMENTS NOT EQUAL TO 1
C
IX = 1
IY = 1
IF(INCX.LT.0) IX = (-N+1)*INCX+1
IF(INCY.LT.0) IY = (-N+1)*INCY+1
DO 10 I = 1, N
STEM = SX(I)
SX(I) = SY(IY)
SY(IY) = STEM
IY = IY+INCY
10 CONTINUE
RETURN

C CODE FOR BOTH INCREMENTS EQUAL TO 1.
C
CLEAN-UP LOOP
20 M = MOD(N, 3)
IF(M.EQ.0) GO TO 40
DO 30 I = 1, M
STEM = SX(I)
SX(I) = SY(I)
SY(I) = STEM
30 CONTINUE
IF(N.LT.3) RETURN

40 MP1 = M+1
DO 50 I = MP1, N, 3
STEM = SX(I)
SX(I) = SY(I)
SY(I) = STEM
50 CONTINUE
RETURN

END
SUBROUTINE FLINE(M, TSCALE, TMIN, I, ASCALE, AMIN, AYL)

DIMENSION U(12)

EQUIVALENCE (T, U(1))

DATA UNIT/12/

REWIND UNIT

II=0

100 READ (UNIT, 1000, END=190) U

1000 FORMAT (8E13.7)

XO=(U(M)-TMIN)/TSCALE

YO=(U(I)-AMIN)/ASCALE+AYL

II=II+1

IF (II.EQ.1) CALL PLOT(XO, YO, 3)

CALL PLOT(XO, YO, 2)

GO TO 100

190 RETURN

END
SUBROUTINE FLINES(M,TSCALE,TMIN,I,ASCALE,AMIN,AYL,HT,ISkip,NCHAR)    FLINES  2
  DIMENSION U(12)                        FLINES  3
  EQUIVALENCE (T,U(1))                   FLINES  4
  DATA IUNIT/12/                        FLINES  5
  REWIND IUNIT                          FLINES  6
  II=0                                  FLINES  7
100 READ(IUNIT,1000,END=190)U            FLINES  8
1000 FORMAT(BE13.7)                     FLINES  9
     XO=(U(M)-TMIN)/TSCALE               FLINES 10
     YO=(U(1)-AMIN)/ASCALE+AYL           FLINES 11
     II=II+1                            FLINES 12
     III=((II/ISkip))*ISkip             FLINES 13
     IF(II.NE.III) GO TO 100             FLINES 14
     CALL SYMBOL(XO,YO,HT,NCHAR,0.,-1)   FLINES 15
150 GO TO 100                           FLINES 16
190 CONTINUE                            FLINES 17
170 RETURN                              FLINES 18
180 END                                  FLINES 19
C START PLOT SEQUENCE
CALL PLOT(4.,5.,-3)
C YLAB="T2(DEC F)"
TLAB="TIME(SEC)"
CALL AXIS(0.,0.,TLAB,-10,TAXL,0.,TMIN,TSCALE)
CALL AXIS(0.,0.,YLAB,10,YAXL,90.,YMIN,YScale)
C 4 IN CALL POINTS TO 4TH VARIABLE IN READ,Y
CALL FLINE(1.,TSCALE,TMIN,4.,YScale,YMIN,0.)
C 5 POINTS TO Z
CALL FLINES(1.,TSCALE,TMIN,5.,YScale,YMIN,0.,HT,1.,3)
C PLOT DEPENDENT VARIABLE
ALAB="ALPHA(DEG)"
CALL AXIS(0.,AYL,ALAB,10,AAXL,90.,AMIN,AScale)
C 6 POINTS TO A
CALL FLINE(1.,TSCALE,TMIN,6.,AScale,AMIN,AYL)
C NEXT PLOT SEQUENCE
AXO=TAXL+2.
CALL PLOT(AXO,0.,-3)
C YLAB="Q/QRef"
YAXL=10.
YScale=.5.
YMIN=0.
IFOX(I)=IFX(4)
DO 32 I=2,8
IFOX(I)=IFX(I+9)
CONTINUE
DO 200 I=1,6
IF(IFOX(I).EQ.0)GO TO 200
DATA XL/"ALPHA(DEG)"/,"LOG(RE)"/,"DELE(DEG)"/"DELBF(DEG)"/"MACH"/
DATA XLS/6.,4.,4.,5.,5./
DATA XSC/5.,1.,5.,5.,5./
DATA XM/20.,-3.,5.,-10.,0.,0./
CALL AXIS(0.,0.,YLAB,10.,YAXL,90.,YMIN,YScale)
CALL AXIS(0.,0.,XL(I),-10.,XLL(I),0.,XM(I),XSC(I))
C I+5 POINTS TO ALPHA / 12 POINTS TO Q/QRef
C 200 CONTINUE
IPT=I+5
CALL FLINE(IPT,XSC(I),XM(I),12.,YScale,YMIN,0.)
CALL PLOT(AXO,0.,-3)
CONTINUE
CALL PLOT"N"
RETURN
SUBROUTINE FPLT

COMMON/CPFLT/iplot, TSCLAE, TMIN, TAXL, YSCALE, YMIN, YAXL,
$ASCAL.E, AMIN, AAXL
DIMENSION XL(6), XXL(6), YSL(6), XSL(6)
DIMENSION IFOX(6)
DIMENSION DIYM(1024)
LOGICAL FAUTO
DIMENSION FAUTO(7)
DIMENSION GP(5)
COMMON/CPARAG/2, HALF(2), PHIC, PHIK, ZP, Z, ALPH(2), KA, S(5),
& CIF(5, 5), KAF, IFLX(5), ACC(5), IFXSUM, NPAR, DALPH(2)
EQUIVALENCE (HO, GP(1))
C INITIALIZE PLOTS AND WRITE PLOT FILE TO UNIT 2
CALL PLTOS(DUM, 1024, 2)
CALL FACTOR(.787403)
DATA IUNIT/12/
REWIND IUNIT
C FIND MAX AND MINS FOR SCALING
DATA THN, TMX, YMN, YMX/1.07, 0., 5000., -400.0/
DATA AMN, AMX/25, 45.0/
C READ T, USI1, EQU, USI2, UMEAS, ALPHE
READ(IUNIT, 1000, END=190) T, U, V, Z, A, B, R, DE, DB, DM, QN
100 FORMAT(8E13.7)
THN=AMIN(T,THN)
TMX-AMAX(1,THN)
YMN=AMIN(1,YMN, U, V, Z)
YMX-AMAX(1,YMX, U, V, Z)
AMN=AMIN(AMN, A)
AMX-AMAX(1, AMX, A)
GO TO 100
CONTINUE
190 CONTINUE
IF(IPLT.GT.0)GO TO 195
C DEFAULT TIME AXIS LENGTH = 4 INCHES
TAXL=4.
TSCLAE=IFIX(((TMX-TMN)/TAXL)+.999)
TMIN=THN
C DEFAULT Y AXIS LENGTH = 4 INCHES
TAXL=4.
DYMN=25.
YSCALE=1-DYMNFIFX((YMNM-YNM)/DYMN/YAXL+1.999)
YMN=YNM=YSCALE*IFX(1, YMN/SCALE)
C DEFAULT A AXIS LENGTH = 2 INCHES
AAXL=2.
DAMN=5
ASCAL.E=DAMN*IFX(AAXL, DAMN/AAXL)+1.999
AMN=ASCAL.E*IFX(1, AMN/ASCAL.E)
CONTINUE
195 CONTINUE
C C SCALE USING INPUT TSCLAE ONLY
C PUT IN NEGATIVE OR ZERO FOR TMIN AND TAXL
IF(TAXL.GT.0)GO TO 198
TMIN=THN
TAXL=IFIX((TMX-TMN)/TSCLAE+.999)
CONTINUE
198 CONTINUE
C
SUBROUTINE IJUL(X,Y,N1,N2,N3,Z)
DIMENSION X(1),Y(1),Z(1)
COMMON/MAIN1/NDIM
NEND3+NDIM=N2
NEND2+NDIM=N2
DO 1 I=1,N1
      DO 1 J=1,NEND3,NDIM
 1     TM=0.
     K=I
     KK=J-I
5      KK=KK+1
2      TM=TM+X(K)*Y(KK)
8      END
9      IF(K.LE.NEND2) GO TO 5
15     1 Z(J)=TM
16     RETURN
17     END
SUBROUTINE MAT4 (N1,N2,X,Y,Z)

C Z=XY* X=X" IS N2XN2, Y IS N1XN2, Z IS N1XN1

DIMENSION X(1), Y(1), Z(1)

COMMON /MAIN1/ MDIM

CALL MMUL (Y,X,N1,N2,N2,Z)

N2=N2+NDIM

DO 3 I=1,N1

IM1=I-1

II=IM1+NDIM

J=I+II

DO 2 J=1,N1

TEMP=0.

KK=J

DO 1 K=I,N2,NDIM

TEMP=TEMP+Y(K)*Z(KK)

1 KK=KK+NDIM

Z(JJ)=TEMP

2 JJ=JJ+NDIM

JJ=I

K=II+1

KK=II+IM1

DO 3 J=K,KK

Z(JJ)=Z(J)

JJ=JJ+NDIM

3 CONTINUE

RETURN

END
FUNCTION XNORM (N,A)
C
COMPUTES AN APPROXIMATION TO NORM OF A-- NOT A BOUND
DIMENSION A(1)
COMMON /MAIN1/ NDIM
NDIM=NDIM+1
NN=N+NDIM
C1=0.
TR=A(1)
IF (N.EQ.1) GO TO 4
I=2
DO 2 II=NDIM1,NN,NDIM

J=II
DO 1 JJ=I,II,NDIM
C1=C1+ABS(A(J)*A(JJ))
1 J=J+1
TR=TR+A(J)
2 I=I+1
TR=TR/FLOAT(N)
DO 3 II=1,NN,NDIM1
3 C1=C1+(A(II)-TR)**2
4 XNORM=ABS(TR)+SQRT(C1)
RETURN
END
<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
SUBROUTINE SGTS(L(N,C,D,E,B,INFO))
INTEGER N,INFO
REAL C(1),E(1),B(1),D(1)
C
SGTS(L GIVEN A GENERAL TRIDIAGONAL MATRIX AND A RIGHT HAND SIDE WILL FIND THE SOLUTION - SEE THE LINPACK USER'S GUIDE
C
INTEGER K, KB, KP1, NM1, NM2
REAL T
INFO=0
C(1)=D(1)
NM1=N-1
IF(NM1.LT.1)GO TO 40
D(1)=E(1)
E(1)=0.0E0
E(N)=0.0E0
DO 30 K=1,NM1
   IF(ABS(C(KP1)).LT.ABS(C(K)))GO TO 10
20 T=C(KP1)
   C(KP1)=C(K)
   C(K)=T
   B(KP1)=B(K)
   E(KP1)=E(K)
   E(K)=T
   T=B(KP1)
   B(KP1)=B(K)
   C(KP1)=C(K)
   T=E(KP1)
   E(KP1)=E(K)
   E(K)=T
   T=B(KP1)
   B(KP1)=B(K)
10 CONTINUE
   IF(C(K).NE.0.0E0)GO TO 20
   INFO=K
20 GO TO 100
30 CONTINUE
   T=C(KP1)/C(K)
   C(KP1)=D(KP1)+T*D(K)
   D(KP1)=E(KP1)+T*E(K)
   E(KP1)=0.0E0
   B(KP1)=B(KP1)+T*B(K)
40 CONTINUE
   IF(C(N).NE.0.0E0)GO TO 50
   INFO=N
50 GO TO 90
   NM2=N-2
   NM1=N-1
   IF(N.ME.1)GO TO 60
60 B(KP1)=B(KP1)+T*B(K)
70 INFO=N
   GO TO 90
   NM2=KB+1
   IF(NM2.EQ.1)GO TO 80
   IF(NM2.EQ.2)GO TO 70
   D=KB+1
   B(KP1)=B(KP1)+T*B(K)
   C(KP1)=C(KP1)+T*C(K)
   T=B(KP1)
   B(KP1)=B(KP1)+T*B(K)
   C(KP1)=C(KP1)+T*C(K)
   T=B(KP1)
   B(KP1)=B(KP1)+T*B(K)
   C(KP1)=C(KP1)+T*C(K)
   T=B(KP1)
   B(KP1)=B(KP1)+T*B(K)
   C(KP1)=C(KP1)+T*C(K)
   T=B(KP1)
   B(KP1)=B(KP1)+T*B(K)
   C(KP1)=C(KP1)+T*C(K)
   T=B(KP1)
   B(KP1)=B(KP1)+T*B(K)
   C(KP1)=C(KP1)+T*C(K)
   T=B(KP1)
   B(KP1)=B(KP1)+T*B(K)
   C(KP1)=C(KP1)+T*C(K)
   T=B(KP1)
   B(KP1)=B(KP1)+T*B(K)
   C(KP1)=C(KP1)+T*C(K)
   T=B(KP1)
   B(KP1)=B(KP1)+T*B(K)
   C(KP1)=C(KP1)+T*C(K)
   T=B(KP1)
   B(KP1)=B(KP1)+T*B(K)
   C(KP1)=C(KP1)+T*C(K)
   T=B(KP1)
   B(KP1)=B(KP1)+T*B(K)
   C(KP1)=C(KP1)+T*C(K)
   T=B(KP1)
   B(KP1)=B(KP1)+T*B(K)
   C(KP1)=C(KP1)+T*C(K)
   T=B(KP1)
   B(KP1)=B(KP1)+T*B(K)
   C(KP1)=C(KP1)+T*C(K)
   T=B(KP1)
   B(KP1)=B(KP1)+T*B(K)
   C(KP1)=C(KP1)+T*C(K)
   T=B(KP1)
   B(KP1)=B(KP1)+T*B(K)
   C(KP1)=C(KP1)+T*C(K)
   T=B(KP1)
   B(KP1)=B(KP1)+T*B(K)
   C(KP1)=C(KP1)+T*C(K)
   T=B(KP1)
   B(KP1)=B(KP1)+T*B(K)
   C(KP1)=C(KP1)+T*C(K)
   T=B(KP1)
   B(KP1)=B(KP1)+T*B(K)
   C(KP1)=C(KP1)+T*C(K)
   T=B(KP1)
   B(KP1)=B(KP1)+T*B(K)
   C(KP1)=C(KP1)+T*C(K)
   T=B(KP1)
   B(KP1)=B(KP1)+T*B(K)
   C(KP1)=C(KP1)+T*C(K)
   T=B(KP1)
   B(KP1)=B(KP1)+T*B(K)
   C(KP1)=C(KP1)+T*C(K)
   T=B(KP1)
   B(KP1)=B(KP1)+T*B(K)
   C(KP1)=C(KP1)+T*C(K)
   T=B(KP1)
   B(KP1)=B(KP1)+T*B(K)
   C(KP1)=C(KP1)+T*C(K)
   T=B(KP1)
   B(KP1)=B(KP1)+T*B(K)
   C(KP1)=C(KP1)+T*C(K)
   T=B(KP1)
   B(KP1)=B(KP1)+T*B(K)
   C(KP1)=C(KP1)+T*C(K)
   T=B(KP1)
   B(KP1)=B(KP1)+T*B(K)
   C(KP1)=C(KP1)+T*C(K)
   T=B(KP1)
   B(KP1)=B(KP1)+T*B(K)
   C(KP1)=C(KP1)+T*C(K)
   T=B(KP1)
   B(KP1)=B(KP1)+T*B(K)
   C(KP1)=C(KP1)+T*C(K)
   T=B(KP1)
   B(KP1)=B(KP1)+T*B(K)
   C(KP1)=C(KP1)+T*C(K)
   T=B(KP1)
   B(KP1)=B(KP1)+T*B(K)
   C(KP1)=C(KP1)+T*C(K)
   T=B(KP1)
   B(KP1)=B(KP1)+T*B(K)
   C(KP1)=C(KP1)+T*C(K)
   T=B(KP1)
   B(KP1)=B(KP1)+T*B(K)
   C(KP1)=C(KP1)+T*C(K)
   T=B(KP1)
   B(KP1)=B(KP1)+T*B(K)
   C(KP1)=C(KP1)+T*C(K)
   T=B(KP1)
   B(KP1)=B(KP1)+T*B(K)
   C(KP1)=C(KP1)+T*C(K)
   T=B(KP1)
   B(KP1)=B(KP1)+T*B(K)
   C(KP1)=C(KP1)+T*C(K)
   T=B(KP1)
   B(KP1)=B(KP1)+T*B(K)
   C(KP1)=C(KP1)+T*C(K)
   T=B(KP1)
   B(KP1)=B(KP1)+T*B(K)
   C(KP1)=C(KP1)+T*C(K)
   T=B(KP1)
   B(KP1)=B(KP1)+T*B(K)
   C(KP1)=C(KP1)+T*C(K)
   T=B(KP1)
   B(KP1)=B(KP1)+T*B(K)
   C(KP1)=C(KP1)+T*C(K)
   T=B(KP1)
   B(KP1)=B(KP1)+T*B(K)
   C(KP1)=C(KP1)+T*C(K)
   T=B(KP1)
   B(KP1)=B(KP1)+T*B(K)
   C(KP1)=C(KP1)+T*C(K)
   T=B(KP1)
   B(KP1)=B(KP1)+T*B(K)
   C(KP1)=C(KP1)+T*C(K)
   T=B(KP1)
   B(KP1)=B(KP1)+T*B(K)
   C(KP1)=C(KP1)+T*C(K)
   T=B(KP1)
   B(KP1)=B(KP1)+T*B(K)
   C(KP1)=C(KP1)+T*C(K)
   T=B(KP1)
   B(KP1)=B(KP1)+T*B(K)
   C(KP1)=C(KP1)+T*C(K)
   T=B(KP1)
   B(KP1)=B(KP1)+T*B(K)
   C(KP1)=C(KP1)+T*C(K)
   T=B(KP1)
   B(KP1)=B(KP1)+T*B(K)
   C(KP1)=C(KP1)+T*C(K)
   T=B(KP1)
   B(KP1)=B(KP1)+T*B(K)
   C(KP1)=C(KP1)+T*C(K)
   T=B(KP1)
   B(KP1)=B(KP1)+T*B(K)
   C(KP1)=C(KP1)+T*C(K)
   T=B(KP1)
   B(KP1)=B(KP1)+T*B(K)
   C(KP1)=C(KP1)+T*C(K)
   T=B(KP1)
   B(KP1)=B(KP1)+T*B(K)
   C(KP1)=C(KP1)+T*C(K)
   T=B(KP1)
   B(KP1)=B(KP1)+T*B(K)
   C(KP1)=C(KP1)+T*C(K)
   T=B(KP1)
   B(KP1)=B(KP1)+T*B(K)
   C(KP1)=C(KP1)+T*C(K)
   T=B(KP1)
   B(KP1)=B(KP1)+T*B(K)
   C(KP1)=C(KP1)+T*C(K)
   T=B(KP1)
   B(KP1)=B(KP1)+T*B(K)
   C(KP1)=C(KP1)+T*C(K)
   T=B(KP1)
   B(KP1)=B(KP1)+T*B(K)
   C(KP1)=C(KP1)+T*C(K)
   T=B(KP1)
   B(KP1)=B(KP1)+T*B(K)
   C(KP1)=C(KP1)+T*C(K)
   T=B(KP1)
   B(KP1)=B(KP1)+T*B(K)
   C(KP1)=C(KP1)+T*C(K)
   T=B(KP1)
   B(KP1)=B(KP1)+T*B(K)
   C(KP1)=C(KP1)+T*C(K)
   T=B(KP1)
   B(KP1)=B(KP1)+T*B(K)
   C(KP1)=C(KP1)+T*C(K)
   T=B(KP1)
   B(KP1)=B(KP1)+T*B(K)
   C(KP1)=C(KP1)+T*C(K)
   T=B(KP1)
   B(KP1)
SUBROUTINE INTEG(N,A,T,QUE,PHI,QD,QDT,N2)

DIMENSION A(N2,N2),QUE(N2,N2),PHI(N2,N2),QD(N2,N2),QDT(N2,N2)

T2 = T*0.5

CALL MXEP(N,A(1,1),T2,PHI(1,1),QD(1,1),QDT(1,1),N2)

CALL TRN(N,QUE(1,1),PHI(1,1),QD(1,1),QDT(1,1),N2)

CALL MSCALE(N,N,QDT(1,1),4,0,QDT(1,1))

CALL MUL(PHI(1,1),PHI(1,1),N,A(1,1),N2)

CALL TRN(N,QUE(1,1),A(1,1),QD(1,1),PHI(1,1),N2)

DO 10 I=1,N

DO 10 J=1,N

QDT(I,J)= QUE(I,J)+ QDT(I,J)+ PHI(I,J)

T8=T/8.0

CALL MSCALE(N,N,QDT(1,1),T8,QDT(1,1))

RETURN

END
SUBROUTINE HEATUN

COMMON/CEAT/Q,TS,QREF,TW,M1,RENS,HBAR,HREF
COMMON/COMTUN/T,TAW1,ALPHA,H,V,RHO,P,TEMP,C,TRAD,ROHOG,
&TO,TSINK,XFT,DEL,PDEL
LOGICAL FAUTO
DIMENSION FAUTO(7)

COMMON/QP(5)
COMMON/CPARAM/HO,HALF(2),PHIC,PHIK,ZP,Z,ALPH(2),KA,S(5),
&CF(5),KAF,IFX(5),ACC(5),IFXSUM,NNPAR,DALPH(2)
EQUIVALENCE (HO,QP(1))
COMMON/ICTPS2/TINIT(1),ERALOW,E
COMMON/CDX/DX(1)
DATA HREF/1./

CHECK FOR ALPHA SEGMENT

IF(ALPHA,LT,ALPH(2)) THEN
   DALPH(1)=ALPHA
   DALPH(2)=ALPH(2)
ENDIF

IF(ALPHA,GE,ALPH(2)) THEN
   DALPH(1)=ALPH(1)
   DALPH(2)=ALPH(2)
ENDIF

HBAR=HO+HALF(1)*(DALPH(1)-ALPH(1))+HALF(2)*(DALPH(2)-ALPH(2))
RETURN
END
SUBROUTINE TPSOSP2 (DT)
COMMON/COSP/NPTSS,USI(B),PHI(B,B),NPT,PC(B,B),RR,
&QD(B,B),QDT(B,B),QUE(B,B),A(B,B),RCX(B),RP(B),RM(B)
COMMON/COPT/N,IOA(B),IOB(B),IOC(B),IOFACE(B),IOD(B)
COMMON/CPC/NPTPC
COMMON /MAIN1/INP,IPVT(B),WORK(B)
INP=6
IF(DT.LE.0.) GO TO 999
T+DT
CALL INTEG(NPTPC,A,T,QUE,PHI,QD,QDT,NPTSS)
CALL SGF(A(1,1),NPTSS,NPTPC,IPVT(1))
CALL SGD(A(1,1),NPTSS,NPTPC,IPVT(1),WORK(1))
CALL MAT4(NPTPC,NPTPC,QD(1,1),A(1,1),QDT(1,1))
DO 20 I=1,NPTPC
DO 20 J=1,NPTPC
PHI(I,J)=A(I,J)
20 PC(I,J)=QDT(I,J)+QD(I,J)
999 RETURN
END
SUBROUTINE EQUATE (NR, NC, A, B)

DIMENSION A(1), B(1)

COMMON /MAIN1/ NDIM

NN=NC+NDIM

NR1=NR-1

DO 1 J=1, NN, NDIM

II=J+NR1

DO 1 ID=J, II

A(IJ)=B(IJ)

1 CONTINUE

RETURN

END
SUBROUTINE GMINV (NR,NC,A,U,MR,NT)  
DIMENSION A(1), U(I), S(J)  
COMMON /MAIN2/ NDIM  
COMMON /INOU/ KIN,KOUT  
NDIM1=NDIM+1  
TOL=1.E-14  
ADV=1.E-24  
NR=NC  
NRN1+NR-1  
TOL1=0.  
JJ=1  
DO 1 J=1,NC  
S(J)=DOT(NR,A(JJ),A(JJ))  
1 IF (S(J).GT.TOL1) TOL1=S(J)  
JJ=JJ+NDIM  
TOL1=ADV+TOL1  
ADV=TOL1  
JJ=1  
DO 14 J=1,NC  
FAC=S(J)  
JM(J)=J+1  
JR(J)=J+NRN1  
JCM=JJ+1  
DO 2 I=J,JCM  
2 U(I)=0.  
U(JCM)=1.0  
IF (J.EQ.1) GO TO 5  
MAX=1  
DO 28 K=1,MAX  
DO 30 L=1,1  
L=L+1  
MAX=MAX+1  
30 IF (S(K).EQ.1.0) GO TO 3  
TEMP=DOT(NR,A(JJ),A(KK))  
CALL VADD (K,TEMP,U(JJ),U(KK))  
4 KK=KK+NDIM  
DD 4 L=1.2  
KK=KK+1  
DO 38 K=1,MAX  
IF (S(K).EQ.0.) GO TO 4  
TEMP=DOT(NR,A(JJ),A(KK))  
CALL VADD (NR,TEMP,A(JJ),A(KK))  
CALL VADD (K,TEMP,U(JJ),U(KK))  
4 KK=KK+NDIM  
TOL1=TOL+ADV  
FAC=DOT(NR,A(JJ),A(JJ))  
5 IF (FAC.GT.TOL1) GO TO 9  
DD 8 I=J,JRM  
6 A(I)=0.  
S(JJ)=0.  
KK=1  
IF (S(K).EQ.0.) KK=KK+NDIM  
IF (S(K).EQ.0.) GO TO 8  
DO 7 K=1,MAX  
TEMP=DOT(K,U(KK),U(JJ))  
CALL VADD (NR,TEMP,A(JJ),A(KK))  
7 KK=KK+NDIM  
8 FAC=DOT(J,U(JJ),U(JJ))
SUBROUTINE VADD
DIMENSION A(I), B(I)
DO 1 I=1,N
1 A(I)=A(I)+C1*B(I)
RETURN
END
SUBROUTINE VADD (N, C1, A, B)
DIMENSION A(1), B(1)

DO 1 I=1, N
1 A(I)=A(I)+C1*B(I)
RETURN
END
BIBLIOGRAPHY


Neil Thomas Cahoon was born on 31 July 1954 in St. Paul, Minnesota, the son of Thomas C. and Eleanor I. Cahoon. He graduated from Henry Sibley Senior High School in 1972 and the United States Air Force Academy in 1976 from which he received a Bachelor's Degree in Aeronautical Engineering. He completed pilot training and received his wings in September 1977 whereupon he was assigned as an RF4C pilot to the 67th Tactical Reconnaissance Wing, Bergstrom AFB, Texas until May 1981. He was then assigned as an RF4C instructor pilot and later as a flight examiner to the 363rd Tactical Fighter Wing, Shaw AFB, South Carolina until entering the school of Engineering, Air Force Institute of Technology, in May 1983.

permanent address: 3530 E. Marshal Gulch
Tuscon, Arizona 85718
Title: Heating Parameter Estimation Using Coaxial Thermocouple Gages in Wind Tunnel Test Articles

Thesis Advisor: James K. Hodge, Major, USAF
A heat energy balance is applied to a coaxial thermocouple gage for parameter estimation in wind tunnel test articles. This method can significantly reduce wind tunnel test costs and time. Modifications to the data reduction program HEATEST (HEATing ESTimation) are made. The program allows for transient test techniques to be used as well as assuming an isothermal wall. A non-linear convective heat transfer coefficient model may also be used. Data is generated to test the new program. Temperature profiles throughout the thermocouple gage were good and were compared with changes in time step, thermocouple length, and number of discrete node points. The estimation of the convective heat transfer coefficient and thermal conductivity were excellent.