FINAL REPORT for NAG-1-2246
Control Law — Control Allocation Interaction

F/A-18 PA Simulation Test-Bed

Dr. Wayne Durham & Mark Nelson

Department of Aerospace & Ocean Engineering
Virginia Polytechnic Institute & State University
Blacksburg, Virginia 24061
1 Introduction

This report documents the first stage of research into Control Law — Control Allocation Interactions. A three-year research effort was originally proposed:

1. Create a desktop flight simulation environment under which experiments related to the open questions may be conducted.

2. Conduct research to determine which aspects of control allocation have impact upon control law design that merits further research.

3. Conduct research into those aspects of control allocation identified above, and their impacts upon control law design.

Simulation code was written utilizing the F/A-18 airframe in the power approach (PA) configuration. A dynamic inversion control law was implemented and used to drive a state-of-the-art control allocation subroutine.

2 Simulation

The airframe used was derived from the F/A-18 model already implemented in CASTLE. The airframe is not realistic, but is intended to be a test-bed for further research. The greatest area in which the test-bed simulation differs from the original airframe is in the treatment of control deflections. There are essentially two sets of control effectors:

1. The original control effectors in the F/A-18 airframe model. These are used only for initial trim and subsequent scheduling.
2. A duplicate set of control effectors that have linear effectiveness. This control set is the input to the control allocator, and the forces and moments they generate are superimposed on those of the bare airframe and original control set. Rate limits of the duplicate set are the no-load rate limits of the original controls. Position limits of the duplicate control effectors are referenced from the trim or scheduled positions of their counterparts in the original controls.

The rationale for incorporating a duplicate control set was to provide a constant, linear control effective matrix with flexibility for future variations and modifications. The control deflections are calculated for the trimmed/scheduled flight condition using the original F/A-18 nonlinear table lookups. The control deflections calculated from the allocator to produce desired moments use the control effectiveness matrix obtained from linearizing the F/A-18 aero database.

### 2.1 Simulation Code

There are six files that are used in the simulation of the airframe: Aero.f, Aeropa.f, Control.f, Constants.f, Engine.f, and Alloc.f.

#### 2.1.1 Aero.f

The aero code first calls Aeropa.f to calculate the aerodynamics of the scheduled/trimmed flight condition. The code then combines the aerodynamics from the non-linear scheduled/trimmed flight condition and from the control deflections calculated in the allocator to produce the desired moments.
2.1.2 Aeropa.f

This code is taken from the F/A-18 simulation and modified slightly to include the control positions that are used in the table lookups later in the code. This is the only code that gives the airframe F/A-18 like characteristics. All added code is at the top of the RUN section.

2.1.3 Control.f

Stick and rudder pedal commands are taken as inputs and converted into an alpha command $\alpha_{cmd}$, beta command $\beta_{cmd}$, and a roll rate command $p_{cmd}$. These commands are input to a simple dynamic inversion control law that generates desired moments for the control allocation subroutine. First, $\alpha_{cmd}$ and $\beta_{cmd}$ are converted to desired accelerations $\dot{u}_{des}$ and $\dot{v}_{des}$:

$$
\begin{align*}
\dot{u}_{cmd} &= u \tan \alpha_{cmd} \\
\dot{v}_{des} &= \lambda_u (u - u_{cmd}) \\
\dot{v}_{cmd} &= V \sin \beta_{cmd} \\
\dot{v}_{des} &= \lambda_v (v - v_{cmd})
\end{align*}
$$

Next, $\dot{w}_{des}$ and $\dot{v}_{des}$ are applied to inversions of the body-axis force equations (treating $q$ and $r$ as controls):

$$
\begin{align*}
q_{cmd} &= \frac{\dot{w}_{des} + pv - g \cos \theta \cos \phi - Z/m}{u} \\
r_{cmd} &= \frac{-\dot{v}_{des} + pu + g \cos \theta \sin \phi + Y/m}{u}
\end{align*}
$$

These two inversions are made as perfect as possible by using actual aircraft states, and the last calculated values of the body-axis forces $Y$ and $Z$ from the
aerodynamic calculations. First-order responses are specified for the desired angular accelerations,

\[ \dot{p}_{\text{des}} = \lambda_p (p - p_{\text{cmd}}) \]
\[ \dot{q}_{\text{des}} = \lambda_q (q - q_{\text{cmd}}) \]
\[ \dot{r}_{\text{des}} = \lambda_r (r - r_{\text{cmd}}) \]

Finally, the desired body-axis moments, required to obtain the desired accelerations, are calculated from inversions of the body-axis moment equations:

\[ C_{\ell_{\text{des}}} = -C_{\ell} + \frac{I_{yy} \dot{p}_{\text{des}} - I_{zz} \dot{r}_{\text{des}} + (I_{ss} - I_{yy})qr - I_{ss} pq}{q S_b} \]
\[ C_{m_{\text{des}}} = -C_{m} + \frac{I_{xx} \dot{q}_{\text{des}} + (I_{xx} - I_{yy})pr + I_{ss} (p^2 - r^2)}{q S_e} \]
\[ C_{n_{\text{des}}} = -C_{n} + \frac{-I_{xx} \dot{r}_{\text{des}} - I_{zz} \dot{p}_{\text{des}} + (I_{yy} - I_{zz})pq + I_{ss} qr}{S_b} \]

The moment coefficients \( C_{\ell}, C_{m}, \) and \( C_{n} \) are the last calculated values of the body-axis moments. Since control-generated moments are superimposed on these values, they are the moments generated by the bare-airframe plus scheduled control deflections. The trimmed flight control deflections are used to calculate moments for the current flight condition to be used in the restoring algorithm. The attained moments are calculated next using the control deflections from the last time step for comparison purposes with the desired moments. The desired moments, along with the required inputs, are input to the allocator to produce the required control deflections. The last step is to check the control deflections against the limits and reset them accordingly.

2.1.4 Constants.f

This section of code sets the model specific constants.
2.1.5 Engine.f

The engine model is taken from the Stevens & Lewis F-16 model [1].

2.1.6 Alloc.f

This code is the control allocator that produces required control deflections for desired moments. This code is explained in detail later.

3 Desktop Simulation

The F/A-18 PA model was first implemented on the UNIX-based CASTLE.

The conversion of the simulation to the desktop PC required the CASTLE offline help menu provided with the PC version of CASTLE. Some additional steps were taken to complete the compilation of the airframe. The steps are as follows:

1. The directory structure from UNIX was copied to the CASTLE airframes folder.

2. A project was created in Microsoft Studio 6.0 following the F/A-18 project already included with the PC version of CASTLE. All custom builds were set up in the same way the F/A-18 project had them set up\(^1\). The custom builds were implemented on symbols.sdf and all the FTP data files.

3. In symbols.sdf the realtime CDF section was changed to resemble the offline CASTLE help explains a different way of setting up custom builds, but did not work.

\(^1\)The offline CASTLE help explains a different way of setting up custom builds, but did not work.
F/A-18 realtime CDF section in the corresponding symbols.sdf. The reason is just a difference in structure between the PC CASTLE and UNIX CASTLE.

4 Control Allocation Algorithm

4.1 Introduction

The control allocation algorithm is a FORTRAN implementation of the bisecting, edge-searching algorithm. The theory behind the allocation code is explained in detail in [2]. Following is a step-by-step explanation of the code. Line numbers correspond to those in the attached file "Alloc.f".

4.2 Subroutine DA3

4.2.1 Diagnostics

The sections of code that depend on the DIAGS flag are debugging tools that can be used to dump several relevant variables. Because a great volume of output is generated, the DIAGS flag should be used sparingly.

4.2.2 Code Description

Lines 0126–0146: Array CSPHI is a table of sines and cosines of angles, beginning at 45° and proceeding through 20 bisections.

Lines 0191–0208: The desired moments are checked for zero length, and a vector of zero control deflections returned if they are; otherwise the
vector is normalized.

**Lines 0210–0219:** The initial rotation is performed using the transformation generated by subroutine DCGEN to align the desired moment direction \( y_3 \) with the \( y_1 \) axis. Subroutine DCGEN is an implementation of the method described in [2, Section 5.1]. Lines 0212–0219 perform the matrix multiplication, \( B_3 = TB_{3..y} \).

**Lines 0221–0231:** The controls that generate the moment with the maximum \( y_1 \) component are found by examining the sign of the first row of \( B \) and setting the control to its maximum or minimum, depending on that sign. The controls are first set to \( \pm 1 \) (object notation) and then set to their actual limits by subroutine SETU.

**Lines 0233–0243:** This section of code was added to deal with the finite precision of computer math. The variable TOL is a distance in moment space that is related to the smallest bisection angle to be used, at the distance from the origin of the vertex just determined (maximum \( y_1 \) component). TOL is used in subsequent code to resolve near-zero numbers.

**Lines 0264–0265:** Subroutine R20 solves the 2-D problem for the projection onto the current \( y_1-y_2 \) plane. R20 returns the object-notation vector of controls of the intersecting edge in variable \( U1 \), the control that defines that edge in variable \( IU \), and a \( \pm 1 \) value in variable INFRONT that is +1 if the edge is in front of, and −1 if it is behind the \( y_1-y_2 \) plane.

The three variables TEMP2, TEMP3, and TEMP4 contain respectively
the sorted list of controls (ITHETA) with an additional zero between the two controls at the ends of the intersecting edge, the number of vertices in the list (NANGS), and the index of the position in the list of the additional zero (INDX). Finally, the logical variable ISVERTEX signals that the desired moment points directly at a vertex.

**Lines 0274-0293:** This section of code has no counterpart in reference [2]. It was added during debugging and found to improve the success rate of the algorithm (decrease the number of estimations required). The most recently found edges that were in front (Last In Front, LIF) and behind (Last In Back, LIB) are saved. Theoretically the last two edges found will be LIF and LIB, but in some cases they were not.

**Lines 0295-0299:** If R20 reports a vertex in ISVERTEX, the controls that determine that vertex, and the saturation of the desired moment, are calculated by a call to DOVERT, and the subroutine is exited.

**Lines 0304-0322:** This section of code initializes several variables, including the rotation matrix T22.

**Lines 0333-0510:** This is the main loop, in which the 2-D problem is repeatedly solved for different rotations about the $y_1$ axis.

**Lines 0335-0342:** Used during debugging, retained for possible future use.

**Lines 0344-0349:** Rotation about $y_1$. B1 is the operative $B$ matrix throughout. Code performs operation $T \cdot B$.
Lines 0360–0364: The last returned values of ITHETA, NANGS, and INDX are assigned to those variables to be saved when TEMP2, TEMP3, and TEMP4 are overwritten by R20.

Lines 0366–0367: Call to R20 to solve the 2-D problem for the current orientation of B1 about the y1-axis.

Lines 0376–0395: The edge identified by R20 is assigned to LIF or LIB according to the sign of the variable INFRONT.

Lines 0397–0401: Another vertex check.

Lines 0411–0495: Executed when the most recent and the prior edges differ in sign of their y3 component, as indicated by the variables INFRONT and WASINF. This section of code is the implementation of the description given in [2, Section 5.3]. Through line 0436 the code is doing housekeeping and (possibly) diagnostics.

Lines 0438–0457: This section reflects a subtlety in the implementation of the algorithm not described in [2]. The prior edge was identified using a different B matrix than the most recent edge. All relevant information regarding the prior edge is contained in the saved variables ITHETA, NANGS, and INDX. At lines 0456–0457 a call is made to subroutine GETEDGE, which is also called as the last step of subroutine R20.

**Lines 0486–0493:** Check the last two edges identified to see if they comprise the solution facet. If they do not, the LIF and LIB edges are checked. Both sets of edges are checked using subroutine **ISFACET**, described below. Output from **ISFACET** consists of the logical ISOK, numbers of the two defining controls in IUOUT and JUOUT, and controls (in object notation) at three vertices of the facet as columns of the array U123.

**Lines 0518–0519:** If the variable ISOK is false, the correct facet has not been determined and the maximum number of bisections has been performed. One last check of LIF and LIB is performed.

**Lines 0520–0574:** If ISOK is true, the solution is calculated. Otherwise (lines 0572–0573) the solution is estimated.

**Lines 0521–0565:** A straightforward implementation of [2, Equations (13) and (14)]. $M_{123}$ is the matrix $[\ddot{e}_{3,1}(v_0^x - v_0^y)(v_0^x - v_0^z)]$ in [2, Equation (13)]; variables AA, BB, and CC correspond to $\alpha_3$, $C_{1,2}$, and $C_{1,3}$ respectively; and MTEMP is $v_0^x$. The variable UDA is the same as $u^a$ in [2, Equation (14)], except that it has been scaled as necessary.

**Lines 0572–0573:** The estimator is called.
4.3 Subroutine DCGEN

This subroutine is a straightforward implementation of the initial transformation algorithm described in [2, Section 5.1].

**Lines 0789–0798:** The desired moments are normalized using double precision math.

**Lines 0810–0824:** If one or more of the leading components of the normalized moment vector are zero, the size of the problem is reduced.

**Lines 0829–0833:** The first row of the transformation matrix is set to the normalized desired moments.

**Lines 0837–0850:** The remaining terms are calculated in the three nested do-loops in [2, Equation (4)].

**Lines 0858–0868:** The last section of DCGEN ensures that the determinant of the transformation matrix is +1.

4.4 Subroutine R20

To find the edge that the desired moments direction is pointing to, the subroutine R20 is implemented. The theory behind this subroutine is in [2, Section 5.2.2]. All calculations in this subroutine are done in the $y_1$-$y_2$ plane.

**Lines 0928–0961:** The $y_2$ component of the point with the maximum $y_1$ component (UMAX in object notation, XUMAX in control notation) is calculated to determine its sign. The desired moment is checked to see
if its direction points towards a vertex of the attainable moment subset. If it is a vertex the subroutine is exited and the allocation carries on.

**Lines 0976–0992:** Implementation of [2, Step 1, Page 20]. The array THETA is the needed part of the set $\mathcal{L}_\phi$, and ITHETA that of $\mathcal{L}_u$. Once the angle is found, $\pi$ is added or subtracted from it if the absolute value is greater than $\pi/2$ and depending on the sign of the angle. In this way, the angles of just the vertices with positive $g_1$ components are generated.

**Lines 0998–1010:** The angles are sorted in a clockwise or counterclockwise manner starting with the vertex that has the largest $g_2$ component. The manner in which they are sorted depends on the sign of the $g_2$ component of the maximum vertex, recorded in SY.

**Lines 1013–1025:** A zero is inserted in THETA and ITHETA to mark the point at which the angle changes sign.

**Lines 1034–1036:** THETA, ITHETA, and NANGS (the number of angles generated) are sent to subroutine GETEDGE to finish the solution to the 2-D problem. Subroutine GETEDGE is provided separately so that it could be called independently from DA3, as described above.

### 4.5 Subroutine GETEDGE

This subroutine is part of the explanation in [2, Section 5.2.2].

**Lines 1090–1127:** The first loop in this subroutine is looking for a sign change in the $y_2$ component between ordered vertices. Since the vertices
were sorted in the manner described, the solution edge will be the first one encountered in traversing the edges starting with the first vertex. The list is stepped through in the proper direction by the index \(IX = IX - SY\).

The previous \(y_2\) value is stored before the next \(y_2\) value is calculated. This new value is compared to the previous one determining whether the edge crosses the \(y_1\) axis. If the do loop continues, \(U2\) is set to the next vertex by changing the sign of the control that is defining the current edge. The index is updated accordingly with the sign of \(y_2\) and the process starts again until the edge is found. The do loop is exited when a new point is found that has a different sign than the point before.

**Lines 1129–1181:** This section deals with possible failure of the previous loop to find an edge, as indicated by \((SY.EQ.SSY)\). The starting values of relevant variables are restored, and the vertex list is traversed in the opposite direction. The first loop should always find the proper edge when \texttt{GETEDGE} is called from \texttt{R20}, but the first loop may fail when called from within \texttt{DA3}. The list is traversed in the opposite direction by the index \(IX = IX + SY\). Implementation of this section of code was the reason for inserting a zero in the ordered list of vertices.

**Lines 1190–1200:** One or the other of the previous two loops will have identified \(U2\) (a vertex in object notation) and \(JU\) (the number of the control that defines the edge). \(U2\) is converted to control notation using the subroutine \texttt{SETU}. The third row of the \(B\) matrix is applied to the two
vertices that define the solution edge to determine the $y_3$ component in moment space at the point where the $y_2$ component of the edge is zero. If the $y_3$ component is positive, the edge is described as “in front”, whereas if the $y_3$ component is negative, the edge is “behind” the line defined by the direction of the desired moments $\ell_3$.

**Lines 1202–1217:** A final vertex check is made and the subroutine is exited.

### 4.6 Subroutine DOVERT

**Lines 693–728:** If it was determined that the desired moments points directly to a vertex the subroutine **DOVERT** is called. **DOVERT** uses the maximum or minimum controls that make up the vertex and calculates the total moment from there, scaling it appropriately. The allocator subroutine is then exited and the simulation carries on. This case is rare during simulation, but may occur.

### 4.7 Subroutine EST

The theory behind the estimator subroutine is explained in [2, Section 5.4.2].

**Lines 0604–629** The subroutine starts with the last two edges that the allocator had found and creates a facet by setting the appropriate control to $-1$ or $+1$. **SETU** is used to assign actual control limits to these points which are then put into moment space using the control effectiveness matrix.

**Lines 631–669** An interpolation is then made with the estimated facet vertices to determine the solution.
The moments are calculated using the estimated control positions and then scaled with the saturation limits.

### 4.8 Subroutine ISFACET

The subroutine is used to test the facet found by DA3. The subroutine uses the two defining controls from DA3 to find a facet from scratch that these two controls define. This algorithm is the subject of reference [4].

**Lines 1236–1251:** Zeros are set in the appropriate positions of the vertex arrays so that two edges are defined for the facet. The dimension of the union (see [2, Section 4.2]) of the two edges is determined. If the union is not two dimensional, then the edges can not form a facet; ISOK is set to false and the subroutine is exited.

**Lines 1253–1320:** For the two dimensional case the routine begins to calculate from scratch the facet that is determined by the two defining controls. The method used is completely independent of the edge-searching method and is explained in [4].

**Lines 1255–1287:** This section of code was lifted from earlier FORTRAN implementations of the facet-searching allocation method described in reference [4]. The facet defined by the two controls is in the variable TESTFACET.

**Lines 1291–1311:** The facet TESTFACET is compared with the object OBJ that was generated by R20. If they are different, the
facet opposite TESTFACET (also generated by the same two controls) is tested (lines 1300-1311).

**Lines 1322–1336:** If the facet just found is the same facet as the one that was found from the allocator, then U123, which is the matrix whose columns correspond to controls that generate three of the vertices that make up the solution facet, is assembled and returned.

### 4.9 Miscellaneous Subroutines

#### 4.9.1 MINNORM

The purpose of the minimum-norm restoring solution is to keep the controls as close to their trimmed control position as possible. The usual minimum-norm solution keeps the controls as close to zero as possible, however, in this application the zero position is redefined as the trimmed/scheduled control positions.

**Lines 1496–1531:** The subroutine is started by finding the total control position for the current time step and calculating the total attained moment.

**Lines 1533–1554:** If the control limits are zero, the routine is returned and no restoring takes place. Otherwise, the difference between the pseudo-inverse solution redefined at the trim condition, and the controls given by the allocation routine are used to find a delta control position that will drive the controls towards the trimmed position. This delta control position is scaled according to the control limits and a new restored control position is returned.

For more information on control restoring, refer to Bolling [3, Ch. 4]
4.9.2 SORTC

Lines 1351–1492: A sorting subroutine downloaded from the National Institute of Standards and Technology (NIST) GAMS (Guide to Available Mathematical Software) at http://gams.nist.gov/. This particular algorithm was chosen for its efficiency, and for the fact that it returns a sorted index vector along with the sorted vector.

4.9.3 INVMAT3


5 Verification Data

Sample runs are included to verify the airframe. The four tests cases used are a trimmed flight condition, a step in the longitudinal stick, and step doublets in the lateral stick and rudder pedals. The MANGEN command in CASTLE was implemented to produce the desired stick commands. Complete MATLAB files of the four cases are attached as trim_dec11.mat, long_dec11.mat, lat_dec11.mat, and dir_dec11.mat.

The plots include selected states of the airframe along with the trimmed/scheduled control positions and the allocated control positions.

Figure 1 shows the time histories of the six global controls in a trimmed flight condition at 8.1 degrees angle of attack, 1200 ft, and 231.52 ft/sec. These settings
are the default when the airframe is loaded. Some settling of the controls to achieve steady state is noted.

Figure 2 shows time histories for a step input in longitudinal stick of 2.5 inches aft from center. The airframe was initialized to the trim conditions described above and the stick step implemented at time = 1 sec for 1 second.

Figure 3 shows the time histories for a step douplet in lateral stick. The lateral stick was driven right 2 inches from center at time = 1 sec for 1 second and then left of center 2 inches for 1 second.

Figure 4 shows the time histories for a step douplet in rudder pedals. The pedals were driven right 2 inches from center at time = 1 sec for 1 second and then left of center 2 inches for 1 second.
Figure 1: Global Control Deflections, Trimmed Flight (Degrees)
Figure 2: Longitudinal Stick Step Input
Figure 3: Lateral Stick Step Doublet
Figure 4: Lateral Rudder Pedal Step Doublet
References


0001 C***********************************************************************
0002 C
0003 C TITLE: DA3
0004 C
0005 C
0006 C
0007 C FUNCTION: 3 Moment Control Allocator
0008 C    Direct Allocation for the 3 objective problem
0009 C    using bisecting edge searching algorithm
0010 C
0011 C
0012 C
0013 C
0014 C DESIGNED BY: Bull Durham
0015 C
0016 C CODED BY: Kevin Scalera
0017 C
0018 C MAINTAINED BY: VPI SIMULATIONS
0019 C
0020 C
0021 C
0022 C MODIFICATION HISTORY:
0023 C
0024 C DATE PURPOSE BY
0025 C ===== ======== ===
0026 C
0027 C
0028 C
0029 C
0030 C GLOSSARY
0031 C ===========
0032 C
0033 C ASSIGNMENTS:
0034 C
0035 C NONE
0036 C
0037 C
0038 C
0039 C INPUTS:
0040 C
0041 C IMODE Sim. mode: -2=init,-1=reset,0=hold,1=ru -------- --------
0042 C
0043 C
0044 C
0045 C OUTPUTS:
0046 C
0047 C NONE
0048 C
0049 C
0050 C
0051 C LOCALLS:
0052 C
0053 C NONE
0054 C
0055 C
0056 C
0057 C OTHER LOCALLS:
SUBROUTINE DA3(UDA, SAT, IERR,
    & B, MDES, U_MIN, U_MAX, M, NBI, TIME, DIAGS)

IMPLICIT NONE

** Parameters

** INPUTS:

INTEGER*4 IMODE

** OTHER LOCALS:

BYTE CONPAR, CTLBUF

LOGICAL*4 DIAGS, DIDSWITCH, INITIALIZED, ISOK, ISVERTEX, STUCK

INTEGER*4 I, I_COUNT, IERR, INDX, I_THETA(21), IU, IUOUT

INTEGER*4 IＵTEMP(20), I_LIB, I_LIF, J, JU, JUOUT, K, M

INTEGER*4 MAXSTEPS, NA_NGS, NBI, NMu_X, STEPS, SY, TEMP2(21)

INTEGER*4 TEMP3, TEMP4, U1(20), U123(20,3), U2(20), U_MAX(20)

INTEGER*4 U_LIB(20), U_LIF(20)

REAL*4 MINV(3,3), MTEMP(3), MXMAX(3), COSPHI, CSPHI(2,20)

REAL*4 ABC(3), NORM, PI, SAT, SINPHI, DET, AA, INF前线, T(3,3)

REAL*4 T22(2,2), BB, BTEMP(2), CC, TIME, TOL, TOLANG, B(3,20)

REAL*4 M123(3,3), MAXNORM, UDA(20), BI(3,20), MD(3), MDES(3)

REAL*4 U_MAX(20), U_MIN(20), WASINF, XU123(20,3), XUMAX(20)

REAL*4 XUTEMP(20), Y

** EQUIVALENCE:

EQUIVALENCE( CONPAR(1), IMODE)
DATA SECTION
DATA INITIALIZED/.FALSE./
DATA PI/3.141592653589793/
DATA CSPHI/
DATA COSINES AND SINES OF BISECTION ANGLE
C
C INITIALIZATION SECTION
IF( (IMODE.LE.-2) .OR. .NOT.INITIALIZED ) THEN
ENDIF
C
C RESET SECTION
C
C RUN SECTION
IF (DIAGS) THEN

WRITE(*,'(A6)', ) ' Entering DA3'
WRITE(*,'(A6)', ) ' Calling arguments'
WRITE(*,'(A6,6E13.6)') ' *DA3* B(I,1) = ', (B(1,I), I=1,M)
WRITE(*,'(A6,6E13.6)') ' *DA3* MDES = ', (MDES(I), I=1,M)
WRITE(*,'(A6,6E13.6)') ' *DA3* U_MIN = ', (U_MIN(I), I=1,M)
WRITE(*,'(A6,6E13.6)') ' *DA3* U_MAX = ', (U_MAX(I), I=1,M)
WRITE(*,'(A6,6E13.6)') ' *DA3* TIME = ', TIME

ENDIF

INFRONT = 1.0

NORM = 0.0
DO I = 1,3
NORM = NORM + MDES(I)*MDES(I)
ENDDO

IF (NORM .EQ. 0.0) THEN
IERR = 0
SAT = 0.0
DO I = 1,M
UDA(I) = 0.0
ENDDO
RETURN
ENDIF

NORM = SQRT(NORM)
DO I = 1,3
MD(I) = MDES(I)/NORM
ENDDO

CALL DCGEN(T, MD)

DO I = 1,3
DO J = 1,M
DO K = 1,3
BI(I,J) = BI(I,J) + T(I,K)*B(K,J)
ENDDO
ENDDO

DO I = 1,M
IF (BI(I,I).EQ.0.) THEN
UMAX(I) = 0
ELSEIF (BI(I,I).LT.0.0) THEN
UMAX(I) = -1
ELSE
UMAX(I) = 1
ENDIF
ENDDO
ENDDO
CALL SETU(XUMAX, UMAX, U_MIN, U_MAX, M)
TOLANG = CSPHI(2, MIN(20, 2*NBI))
DO I=1,3
MXMAX(I) = 0.
DO J=1,M
MXMAX(I) = MXMAX(I) + B1(I,J) * XUMAX(J)
ENDDO
ENDDO
MAXNORM = SQRT(MXMAX(I)*MXMAX(I))
& + MXMAX(2)*MXMAX(2)
& + MXMAX(3)*MXMAX(3))
TOL = MAXNORM*TOLANG
IF (DIAGS) THEN
WRITE(*,'(A30, E18.6)') 'First call to R20'
WRITE(*,'(A30, E18.6)') 'Preliminary Calcs'
WRITE(*,'(A30, E18.6)') 'DA1* NORM = ', NORM
WRITE(*,'(A30, E18.6)') 'DA1* XD = ', (MD(I), I=1,3)
WRITE(*,'(A30, E18.6)') 'DA1* T(1, :) = ', (T(1,I), I=1,3)
WRITE(*,'(A30, E18.6)') 'DA1* T(2, :) = ', (T(2,I), I=1,3)
WRITE(*,'(A30, E18.6)') 'DA1* T(3, :) = ', (T(3,I), I=1,3)
WRITE(*,'(A30, E18.6)') 'DA1* BI(1,I) = ', (B1(1,I), I=1,M)
WRITE(*,'(A30, E18.6)') 'DA1* BI(2,I) = ', (B1(2,I), I=1,M)
WRITE(*,'(A30, E18.6)') 'DA1* BI(3,I) = ', (B1(3,I), I=1,M)
WRITE(*,'(A30, E18.6)') 'DA1* UMAX = ', (XUMAX(I), I=1,M)
WRITE(*,'(A30, E18.6)') 'DA1* MXMAX = ', (MXMAX(I), I=1,M)
WRITE(*,'(A30, E18.6)') 'DA1* TOLANG = ', TOLANG
WRITE(*,'(A30, E18.6)') 'DA1* MAXNORM = ', MAXNORM
WRITE(*,'(A30, E18.6)') 'DA1* TOL = ', TOL
ENDIF
ENDIF
CALL R20(U1, IU, INFRONT, TEMP2, TEMP3, TEMP4, ISVERTEX,
& B1, UMAX, XUMAX, U_MIN, U_MAX, TOL, M, DIAGS)
IF (DIAGS) THEN
WRITE(*,'(A30)') ' After last R20'
WRITE(*,'(A30, THETA)') 'DA3* TEMP2 (ITHETA) = ', (TEMP2(I), I=1, TEMP3+1)
WRITE(*,'(A30)') 'DA3* TEMP3 (NANGS) = ', TEMP3
WRITE(*,'(A30)') 'DA3* TEMP4 (INDX) = ', TEMP4
ENDIF
IF (IU.NE.0) THEN
IF (INFRONT.EQ.1.) THEN
DO I=1,M
U_LIF(I) = U1(I)
ENDDO
I_LIF = IU
ELSEIF (INFRONT.EQ.-1.) THEN
DO I=1,M
U_LIB(I) = U1(I)
ENDDO
I_LIB = IU
ENDIF
IF (DIAGS) THEN
    WRITE(*,'(/A50)') ' After 1st LIF/LIB'
    WRITE(*,'(A30,6I13)') ' *DA3* U_LIF = ', (U_LIF(I), I=1,M)
    WRITE(*,'(A30,13)') ' *DA3* I_LIF = ', I_LIF
    WRITE(*,'(A30,6I13)') ' *DA3* U_LIB = ', (U_LIB(I), I=1,M)
    WRITE(*,'(A30,13)') ' *DA3* I_LIB = ', I_LIB
ENDIF

IF (ISVERTEX) THEN
    WRITE(*,'*') ' TIME = ', TIME, ' FIRST CALL TO R20'
    CALL DOVERT(UDA,SAT,U1,B,U_MIN,U_MAX,M,NORM)
    RETURN
ENDIF

C 1st rotation about x-axis
IF (M.GE.8) THEN
    ICOUNT = 1
ELSE
    ICOUNT = 2
ENDIF
COSPHI = CPHI(1,ICOUNT)
SINPHI = INFRONT*COSPHI(2,ICOUNT)
T22(I,1) = COSPHI
T22(I,2) = -SINPHI
T22(2,1) = SINPHI
T22(2,2) = COSPHI

MAXSTEPS = 2*INT(ABS(PI/ASIN(SINPHI)))

DO WHILE ((ICOUNT.LT.NMAX) .AND. (.NOT.ISOK))

STUCK = .FALSE.

STEPS = STEPS+I
IF (STEPS.GE.MAXSTEPS) THEN
    STUCK = .TRUE.
ELSE
    ISOK = .FALSE.
 ELSE
    NMAX = NBI + 1
ENDIF

DIDSWITCH = .FALSE.

TIME = TIME + 1

IF (DIAGS) THEN
    WRITE(*,'(A30)') ' Before Main Loop'
    WRITE(*,'(A30,E10.6)') ' *DA3* COSPHI = ', COSPHI
    WRITE(*,'(A30,E10.6)') ' *DA3* SINPHI = ', SINPHI
    WRITE(*,'(A30,13)') ' *DA3* MAXSTEPS = ', MAXSTEPS
ENDIF

MAIN LOOP ***************************************
DO WHILE ((ICOUNT.LT.NMAX) .AND. (.NOT.ISOK))

STEPS = STEPS+1
IF (STEPS.GE.MAXSTEPS) THEN
    STUCK = .TRUE.
ELSE
    ISOK = .FALSE.
 ELSE
    NMAX = NBI + 1
ENDIF

DIDSWITCH = .FALSE.

TIME = TIME + 1

IF (DIAGS) THEN
    WRITE(*,'(A30)') ' Before Main Loop'
    WRITE(*,'(A30,E10.6)') ' *DA3* COSPHI = ', COSPHI
    WRITE(*,'(A30,E10.6)') ' *DA3* SINPHI = ', SINPHI
    WRITE(*,'(A30,13)') ' *DA3* MAXSTEPS = ', MAXSTEPS
ENDIF

ENDIF
DO J = 1,M
  BTEMP(1) = T22(1,1)*BI(2,J) + T22(1,2)*BI(3,J)
  BTEMP(2) = T22(2,1)*BI(2,J) + T22(2,2)*BI(3,J)
  BI(2,J) = BTEMP(1)
  BI(3,J) = BTEMP(2)
ENDDO

IF (DIAGS) THEN
  WRITE(*,'(/A30,)') ' In DA3 DO WHILE'
  WRITE(*,'(A30,)') ' *DA3* ICOUNT = ', ICOUNT
  WRITE(*,'(A30,)') ' *DA3* STEPS = ', STEPS
  WRITE(*,'(A30,)') ' *DA3* MAXSTEPS = ', MAXSTEPS
  WRITE(*,'(A30,)') ' *DA3* B1(2,:)= ', (BI(2,I),I=1,M)
  WRITE(*,'(A30,)') ' *DA3* B1(3,:)= ', (BI(3,I),I=1,M)
ENDIF

NANGS = TEMP3
INDX = TEMP4
DO I=1,21
  ITHETA(I) = TEMP2(I)
ENDDO

CALL R20(UI,IU, INFRONT, TEMP2, TEMP3, TEMP4, ISVERTEX, IF (IU.NE.0) THEN
  IF (INFRONT.EQ.1.) THEN
    DO I=1,M
      U_LIF(I) = U1(I)
    ENDDO
    I_LIF = IU
  ELSEIF (INFRONT.EQ.-1.) THEN
    DO I=1,M
      U_LIB(I) = UI(I)
    ENDDO
    I_LIB = IU
 ENDIF
ENDIF

IF (ISVERTEX) THEN
  WRITE(*,*) ' [X£ : ' TIME, ALL
  CALL DOVERT(UDA,SAT,UI,BI,U_MIN,U_MAX,M,NORM)
ENDIF

IF (IU.NE.0) THEN
  IF (INFRONT.EQ.1.) THEN
    DO I=1,M
      U_LIF(I) = U1(I)
    ENDDO
    I_LIF = IU
  ELSEIF (INFRONT.EQ.-1.) THEN
    DO I=1,M
      U_LIB(I) = UI(I)
    ENDDO
    I_LIB = IU
 ENDIF
ENDIF

IF (DIAGS) THEN
  WRITE(*,'(/A30,)') ' After Loop R20 LIF LIF'
  WRITE(*,'(A30,)') ' *DA3* U_LIF = ', (U_LIF(I),I=1,M)
  WRITE(*,'(A30,)') ' *DA3* I_LIF = ', I_LIF
  WRITE(*,'(A30,)') ' *DA3* U_LIB = ', (U_LIB(I),I=1,M)
  WRITE(*,'(A30,)') ' *DA3* I_LIB = ', I_LIB
ENDIF

ENDIF
0400  RETURN  
0401  ENDIF  
0402  
0403  IF (DIAGS) THEN  
0404  WRITE(*, '(A50)', ' Before testing reversal'  
0405  WRITE(*, '(A30,F14.6)') ' CA3* INFRONT = ', INFRONT  
0406  WRITE(*, '(A30,F14.6)') ' CA3* WASINF = ', WASINF  
0407  ENDIF  
0408  
0409  DIDSWITCH = .FALSE.  
0410  IF (INFRONT.NE.WASINF) THEN  
0411     IF (DIAGS) THEN  
0412        WRITE(*, '(A50)', ' Reversing'  
0413        WRITE(*, '(A30,I3)') ' CA3* Steps taken = ', STEPS  
0414        WRITE(*, '(A30,F14.6)') ' CA3* Angle = ', 180.*ASIN(SINPHI)/PI  
0415        WRITE(*, '(A30,I3)') ' CA3* MAXSTEPS = ', MAXSTEPS  
0416        WRITE(*, '(A30,L3)') ' CA3* STUCK = ', STUCK  
0417     ENDIF  
0418  ENDIF  
0419  
0420  ICOUNT = ICOUNT+1  
0421  C Bisection and next transformation  
0422  COSPHI = CSPHI(1,ICOUNT)  
0423  SINPHI = INFRONT*CSPHI(2,ICOUNT)  
0424  T22(1,1) = COSPHI  
0425  T22(1,2) = -SINPHI  
0426  T22(2,1) = SINPHI  
0427  T22(2,2) = COSPHI  
0428  MAXSTEPS = 2*INT(ABS(PI/ASIN(SINPHI)))  
0429  IF (DIAGS) THEN  
0430     WRITE(*, '(A50)', ' Bisection and next transformation'  
0431     WRITE(*, '(A30,E13.6)') ' CA3* COSPHI = ', COSPHI  
0432     WRITE(*, '(A30,E13.6)') ' CA3* SINPHI = ', SINPHI  
0433     WRITE(*, '(A30,I3)') ' CA3* MAXSTEPS = ', MAXSTEPS  
0434  ENDIF  
0435  
0436  STEPS = 0  
0437  C Check last edge with new Bl  
0438  Y = 0.0  
0439  DO I = 1,M  
0440     Y = Y + BI(2,1)*XUMAX(I)  
0441  ENDDO  
0442  SY = 1  
0443  IF (Y.LT.0.0) SY = -1  
0444  IF (DIAGS) THEN  
0445     WRITE(*, '(A50)', ' Before GE TEDGE'  
0446     WRITE(*, '(A30,E13.6)') ' CA3* Bl(2,1) = ', (BI(2,1),I=1,M)  
0447     WRITE(*, '(A30,E13.6)') ' CA3* Bl(3,1) = ', (BI(3,1),I=1,M)  
0448     WRITE(*, '(A30,E13.6)') ' CA3* UM A X = ', (UMAX(I),I=1,M)  
0449     WRITE(*, '(A30,E13.6)') ' CA3* Y = ', Y  
0450     WRITE(*, '(A30,L3)') ' CA3* SY = ', SY  
0451     WRITE(*, '(A30,I3)') ' CA3* ITHETA = ', (ITHETA(I),I=1,NANGS+1)  
0452     WRITE(*, '(A30,I3)') ' CA3* NANGS = ', NANGS  
0453  ENDIF  
0454  
0455  CALL GETEDGE(U2, JU, INFRONT, ISVERTEX,
& B1, UMAX, Y, SY, ITHETA, NANGS, U_MIN, U_MAX, INDX, TOL, M, DIAGS)
0459 IF (JU.NE.0) THEN
0460   IF (INFRONT.EQ.1.) THEN
0461     DO I=1,M
0462       U_LIF(I) = U2(I)
0463     ENDDO
0464     I_LIF = JU
0465   ELSEIF (INFRONT.EQ.-1.) THEN
0466     DO I=1,M
0467       U_LIB(I) = U2(I)
0468     ENDDO
0469     I_LIB = JU
0470   ENDIF
0471 IF (DIAGS) THEN
0472   WRITE(*, '(A30,13)') ' After SETEDGE LIF/LIB'
0473   WRITE(*, '(A30,613)') ' *DA3* U_LIF = ', (U_LIF(I), I=1,M)
0474   WRITE(*, '(A30,613)') ' *DA3* I_LIF = ', I_LIF
0475   WRITE(*, '(A30,613)') ' *DA3* U_LIB = ', (U_LIB(I), I=1,M)
0476   WRITE(*, '(A30,23)') ' *DA3* I_LIB = ', I_LIB
0477 ENDIF
0478 ENDIF
0479 IF (ISVERTEX) THEN
0480   WRITE(*, '(A30,13)') ' TIME = ', TIME, ' FOL GETEDGE'
0481   CALL DOVERT(UDA,SAT,U2,B,U_MIN,U_MAX,M,NORM)
0482   RETURN
0483 ENDIF
0484 IF (JU.NE.0) THEN
0485   CALL ISFACET(ISOK, IUOUT, JUOUT, U123, IU, JU, U1, U2, B1, M, TOL)
0486   IF (.NOT.ISOK) CALL ISFACET(ISOK, IUOUT, JUOUT, U123, I_LIF, I_LIB, U_LIF, U_LIB, B1, M, TOL)
0487 ELSE
0488     ISOK = .FALSE.
0489 ENDIF
0490 IF ((ICOUNT.EQ.NMAX).AND.(.NOT.ISOK) .AND.(.NOT.DIDSWITCH)) THEN
0491     NMAX = NMAX+1
0492 ENDIF
0493 IF (STUCK) THEN
0494     WRITE(*, '(A30,13)') ' Stuck in DA3, exiting'
0495     WRITE(*, '(A30,613)') ' TIME = ', TIME
0496     RETURN
0497 ENDIF
0498 C Must leave on a switch
0499 C IF (((ICOUNT.EQ.NMAX).AND.(.NOT.ISOK).AND.(.NOT.DIDSWITCH)) THEN
0500 C NMAX = NMAX+1
0501 C ENDIF
0502 IF (STUCK) THEN
0503     WRITE(*, '(A30,13)') ' Stuck in DA3, exiting'
0504     WRITE(*, '(A30,613)') ' TIME = ', TIME
0505     RETURN
0506 ENDIF
0507 ENDDO ! End of do while statement
0508 C END MAIN LOOP ********************************
0509 C
IF (DIAGS) THEN
  WRITE(*, '(AB0)') ' Exited from B3'
ENDIF

IF (.NOT.ISOK) CALL ISFACET(ISOK, IUOUT, JUOUT, U123, & I_LIF, I_LIB, U_LIF, U_LIB, B1, M, TOL)
IF (ISOK) THEN
  DO I=1,3
    DO J=1,M
      IUTEMP(J)=U123(J,I)
    ENDDO
    CALL SETU(XUTEMP, IUTEMP, U_MIN, U_MAX, M)
    DO J=1,M
      XU123(J,I)=XUTEMP(J)
    ENDDO
  ENDDO

  DO I=1,3
    DO J=1,3
      MI23(I,J)=0.
      DO K=I,M
        MI23(I,J)=MI23(I,J)+B(I,K)*XUI23(K,I)
      ENDDO
    ENDDO
  ENDDO

  DO I=1,3
    DO J=2,3
      MI23(I,J)=MI23(I,I)-MI23(I,J)
    ENDDO
    MTEMP(I)=MI23(I,I)
    MI23(I,I)=MDES(I)
  ENDDO
  CALL INVMAT3(MI23,MINV,DET)
  DO I=1,3
    ABC(I)=0.
    DO J=1,3
      ABC(I)=ABC(I)+MINV(I,J)*MTEMP(J)
    ENDDO
  ENDDO
  AA=ABC(1)
  BB=ABC(2)
  CC=ABC(3)
  SAT=1./AA
  IF (AA.LT.1.) AA = 1.
  DO I=1,M
    UDA(I) = (XU123(I,1) + BB*(XU123(I,2)-XU123(I,1)) + CC*(XU123(I,3)-XU123(I,1)))/AA
  ENDDO

  IERR = 0
C Call estimate subroutine to estimate solution if facet not found
C
ELSE
    CALL EST(UDA, SAT, IERR,
           & U_LIF, I_LIF, U_LIB, I_LIB, BI, U_MIN, U_MAX, NORM, M)
ENDIF
RETURN
END

C..............................................................................
SUBROUTINE EST(UDA, SAT, IERR,
                & UI, IU, U2, JU, B, U_MIN, U_MAX, NORM, M)
IMPLICIT NONE

C INPUTS
REAL*4 B(3,20), U_MAX(20), U_MIN(20), NORM
INTEGER*4 UI(20), IU, U2(20), JU, M

C OUTPUTS
REAL*4 SAT, UDA(20)
INTEGER*4 IERR

C LOCALS
REAL*4 XUI(20), XU2(20), XU3(20), XU4(20), XMOM(3)
REAL*4 UPPER1(3), UPPER2(3), LOWER1(3), LOWER2(3), XNORM
REAL*4 XK1, XK2, XK3, XVI(3), XV2(3), XWI(20), XW2(20)
INTEGER*4 U3(20), U4(20)

C OTHER LOCALS
INTEGER*4 I, J, K

IERR = 1
U1(IU) = -1
U2(JU) = -1
DO I=1,M
    U3(I) = U1(I)
    U4(I) = U2(I)
ENDDO
U3(IU) = 1
U4(JU) = 1

CALL SETU(XU1, U1, U_MIN, U_MAX, M)
CALL SETU(XU2, U2, U_MIN, U_MAX, M)
CALL SETU(XU3, U3, U_MIN, U_MAX, M)
CALL SETU(XU4, U4, U_MIN, U_MAX, M)

DO I=1,3
    LOWER1(I) = 0.
    LOWER2(I) = 0.
    UPPER1(I) = 0.
    UPPER2(I) = 0.
    DO J=1,M
        LOWER1(I) = LOWER1(I)*B(I,J)*XU1(J)
        LOWER2(I) = LOWER2(I)*B(I,J)*XU2(J)
        UPPER1(I) = UPPER1(I)*B(I,J)*XU3(J)
        UPPER2(I) = UPPER2(I)*B(I,J)*XU4(J)
    ENDDO
ENDDO
ENDDO
ENDDO
IF (LOWER1(2).NE.UPPER1(2)) THEN
  XK1 = LOWER1(2)/(LOWER1(2)-UPPER1(2))
ELSE
  XK1 = 0.
ENDIF
IF (LOWER2(2).NE.UPPER2(2)) THEN
  XK2 = LOWER2(2)/(LOWER2(2)-UPPER2(2))
ELSE
  XK2 = 0.
ENDIF
DO I=i,3
  XV1(I) = XK1*UPPER1(I)+(1.-XK1)*LOWER1(I)
  XV2(I) = XK2*UPPER2(I)+(1.-XK2)*LOWER2(I)
ENDDO
IF (XV2(3).NE.XV1(3)) THEN
  XK3 = XV2(3)/(XV2(3)-XV1(3))
ELSE
  XK3 = 0.
ENDIF
DO I=I,M
  XWI(I) = XK1*XU3(I)+(1.-XK1)*XU1(I)
  XW2(I) = XK2*XU4(I)+(1.-XK2)*XU2(I)
ENDDO
DO I=I,M
  UDA(I) = XK3*XWI(I)+(1.-XK3)*XW2(I)
ENDDO
DO I=I,3
  XMOM(I) = 0.
  DO J=I,M
    XMOM(I) = XMOM(I) + B(I,J)*UDA(J)
  ENDDO
ENDDO
XNORM = SQRT(XMOM(1)*XMOM(1)
& + XMOM(2)*XMOM(2)
& + XMOM(3)*XMOM(3))
IF (XNORM.NE.0.) THEN
  SAT = NORM/XNORM
  XNORM = SAT
ELSE
  SAT = 0.
ENDIF
IF (XNORM.GT.1.) XNORM = 1.
DO I=I,M
SUBROUTINE DOVERT(UDA, SAT, 
& UI, B, U_MIN, U_MAX, M, NORM)
IMPLICIT NONE
REAL*4 UDA(20), SAT, U_MIN(20), U_MAX(20), B(3,20), NORM
INTEGER*4 UI(20), M
REAL*4 XMOM(3), XNORM
INTEGER*4 I, J
WRITE(*,*) ' V_ZRI'EX
CALL SETU(UDA,UI,U_MIN,U_MAX,M)
DO 1=1,3
XMOM(I) = 0.
DO J=I,M
XMOM(I) = XMOM(I)+B(I,J)*UDA(J)
ENDDO
ENDDO
XNORM = SQRT(XMOM(1)*XMOM(1)
& +XMOM(2)*XMOM(2)
& +XMOM(3)*XMOM(3))
SAT = NORM/XNORM
XNORM = SAT
IF (XNORM.GT.1.) XNORM = 1.
DO I = 1,M
UDA(I) = XNORM*UDA(I)
ENDDO
RETURN
END

C Zero out the output matrix
SUBROUTINE INVMAT3(MATIN, MATOUT, DET)
IMPLICIT NONE
INTEGER*4 I, J
REAL*4 DET, MATIN(3,3), MATOUT(3,3)
C Zero out the output matrix
DO I = 1,3
DO J = 1,3
0742      MATOUT(I,J) = 0.0
0743      ENDDO
0744      ENDDO
0745 C Calculate the determinant of the input matrix
0746 C
0748      DET = MATIN(1,1) * MATIN(2,2) * MATIN(3,3) &
0749          + MATIN(1,2) * MATIN(2,3) * MATIN(3,1) &
0750          + MATIN(1,3) * MATIN(2,1) * MATIN(3,2) &
0751          - MATIN(1,3) * MATIN(2,2) * MATIN(3,1) &
0752          - MATIN(1,2) * MATIN(2,1) * MATIN(3,3) &
0753          - MATIN(1,1) * MATIN(2,3) * MATIN(3,2)
0754 C
0755 C Find the matrix inverse
0756 C
0758      IF (DET .NE. 0.0) THEN
0759          MATOUT(1,1) = (MATIN(2,2) * MATIN(3,3) - MATIN(2,3) * MATIN(3,2)) / DET &
0760          MATOUT(1,2) = -(MATIN(1,2) * MATIN(3,3) - MATIN(1,3) * MATIN(3,2)) / DET &
0761          MATOUT(1,3) = (MATIN(1,2) * MATIN(2,3) - MATIN(1,3) * MATIN(2,2)) / DET &
0762          MATOUT(2,1) = -(MATIN(2,1) * MATIN(3,3) - MATIN(2,3) * MATIN(3,1)) / DET &
0763          MATOUT(2,2) = (MATIN(1,1) * MATIN(3,3) - MATIN(1,3) * MATIN(3,2)) / DET &
0764          MATOUT(2,3) = -(MATIN(1,1) * MATIN(2,3) - MATIN(1,3) * MATIN(2,2)) / DET &
0765          MATOUT(3,1) = (MATIN(2,1) * MATIN(3,2) - MATIN(2,2) * MATIN(3,1)) / DET &
0766          MATOUT(3,2) = -(MATIN(1,1) * MATIN(2,3) - MATIN(1,3) * MATIN(2,2)) / DET &
0767          MATOUT(3,3) = (MATIN(1,1) * MATIN(2,2) - MATIN(1,2) * MATIN(2,1)) / DET
0768      ENDF
0769
0770
0771      RETURN
0772      END
0773
0774 C-----------------------------
0775
0776      SUBROUTINE DCGEN(T, MD)
0777      IMPLICIT NONE
0778      REAL*4 MD(3)
0779      REAL*4 T(3,3)
0780      INTEGER*4 MLOCAL
0781      REAL*8 V(3), VLEN(3), VNORM, XDOT_DC
0782      REAL*4 DETNUM, AMIN_DC
0783      INTEGER*4 J, JCOL, KCOL, I, MOM_FLAG, IROW
0784      REAL*8 DTOL
0785      REAL*8 VDOT
0786 C
0787 C Calculate the norm of the moments
0788 C
0789      DTOL = 1.D-5
0790      VNORM = DSQRT(dble(MD(1)) * dble(MD(1)) &
0791          + dble(MD(2)) * dble(MD(2)) &
0792          + dble(MD(3)) * dble(MD(3)))
0793 C
0794 C Normalize the desired moments
0795 C
0796      DO I = 1, 3
0797          V(I) = dble(MD(I)) / VNORM
0798      ENDDO
0799 C
0800 C Zero out the transformation matrix
0801 C
0802 DO I = 1,3
0803 DO J = 1,3
0804 T(I,J) = 0.0
0805 ENDDO
0806 ENDDO
0807 C
0808 C Check to see if V(I),3 to 1 is approx equal to zero => reduce size of problem
0809 C
0810 DO I = 1,3,-1
0811 IF (ABS(V(I)) .LE. DTOL) THEN
0812 T(I,I) = 1.0
0813 ELSE
0814 GOTO 5
0815 ENDF
0816 ENDDO
0817 C
0818 C T(I,1) = 1.0 or -1.0 for rotation about x-axis (depends on direction of rot.)
0819 C
0820 5 IF (I .EQ. 1) THEN
0821 T(I,I) = 1.0
0822 IF (dble(MD(1)) .LT.0.0D0) T(I,I) =-1.0
0823 RETURN
0824 ENDF
0825 C
0826 C Set the 1st row of T equal to the normalized desired moments and
0827 C calculate the square of each of these values
0828 C
0829 MLOCAL = I
0830 DO I = 1,3
0831 T(1,I) = V(I)
0832 VLEN(I) = V(I)*V(I)
0833 ENDDO
0834 C
0835 C Developing orthogonal tranformation with V as 1st row
0836 C
0837 DO JCOL = 1,MLOCAL-1
0838 IROW = MLOCAL + 1 - JCOL
0839 T(IROW,JCOL) = sngl(DSQRT(1.0D0 - VLEN(JCOL)))
0840 DO KCOL = 1,MLOCAL
0841 XDOT_DC = T(I,JCOL)*T(I,KCOL)
0842 IF (IROW .NE. MLOCAL) THEN
0843 DO I = IROW+1,MLOCAL
0844 XDOT_DC = XDOT_DC + T(I,JCOL)*T(I,KCOL)
0845 ENDDO
0846 ENDF
0847 T(IROW,KCOL) = -XDOT_DC/T(IROW,JCOL)
0848 VLEN(KCOL) = VLEN(KCOL) + dble(T(IROW,KCOL))*dble(T(IROW,KCOL))
0849 ENDDO
0850 ENDDO
0851 C
0852 C Tricky stuff here!
0853 C
0854 DETNUM = int(mod(MLOCAL,4)/2.0)
0855 C
C Necessary to do, but not easy to explain

IF (DETNUM.EQ.0.0) THEN
  IF (T(I,MLOCAL) .LT. 0.0) THEN
    T(2,MLOCAL-I) = -T(2,MLOCAL-I)
    T(2,MLOCAL) = -T(2,MLOCAL)
  ENDIF
ELSE
  IF (T(I,MLOCAL) .GT. 0.0) THEN
    T(2,MLOCAL-I) = -T(2,MLOCAL-I)
    T(2,MLOCAL) = -T(2,MLOCAL)
  ENDIF
ENDIF
RETURN
END

SUBROUTINE SETU(XU_SETU, IU_SETU, U_MIN, U_MAX, M)

IMPLICIT NONE
REAL*4 U_MAX(20), U_MIN(20)
INTEGER*4 IMODE, M
INTEGER*4 I, IU_SETU(20)
REAL*4 XU_SETU(20)

DO I = 1,M
  IF (IU_SETU(I) .EQ. 1) THEN
    XU_SETU(I) = U_MAX(I)
  ELSEIF (IU_SETU(I) .EQ.-1) THEN
    XU_SETU(I) = U_MIN(I)
  ELSE
    XU_SETU(I) = 0.
  ENDIF
ENDDO
RETURN
END

SUBROUTINE R20(U1, IU, INFRONT, ITHETA, NANGS, INDEX, ISVERTEX, & B1, U_MAX, XU_MAX, U_MIN, U_MAX, TOL, M, DIAGS)
IMPLICIT NONE
REAL*4 U_MIN(20), U_MAX(20)
INTEGER*4 IMODE, ITHETA(21), IU, U_MAX(20), M, N, NANGS, SY
INTEGER*4 U1(20)
REAL*4 PIOVR2, B1(3,20), PI, XU(20), Y, XU_MAX(20)
REAL*4 INFRONT, TOL
REAL*4 THETA(21)
LOGICAL*4 ISVERTEX, DIAGS
INTEGER*4 INDEX, SY
INTEGER*4 I, J
REAL*4 ANG, YPREV, XU(20), K, Z, Z1, Z2
DATA PIOVR2/1.570796326794897/
DATA PI/3.141592653589793/

IF (DIAGS) THEN
WRITE(*, '(/AS0)') ' Entering R2O'
WRITE(*, '(/AS0)') ' Calling arguments'
WRITE(*, '(/AS0,6F18.6)') ' *PO0 * B1(1,i) = ', (B1(1,i), I=1,M)
WRITE(*, '(/AS0,6E16.8)') ' *PO0 * B1(2,i) = ', (B1(2,i), I=1,M)
WRITE(*, '(/AS0,6E16.8)') ' *PO0 * B1(3,i) = ', (B1(3,i), I=1,M)
WRITE(*, '(/AS0,6E16.8)') ' *PO0 * XUMAX = ', (XUMAX(I), I=1,M)
WRITE(*, '(/AS0,6F14.6)') ' *PO0 * XUMAX = ', (XUMAX(I), I=1,M)
ENDIF

C Calculate Y
Y = 0.0
DO I = 1,M
UI(I) = UMAX(I)
Y = Y + B1(2,I)*XUMAX(I)
ENDDO

C VERTEX CHECK
ISVERTEX = .FALSE.
IF (ABS(Y).LT.TOL) THEN
Y = 0.
Z = 0.
DO I=1,M
Z = Z+B1(3,I)*XUMAX(I)
ENDDO
IF (ABS(Z).LT.TOL) THEN
C WRITE(*,*) 'Z ',Z,
ISVERTEX = .TRUE.
IU = 0
INFRONT = 0.
NANGS = 0
RETURN
ENDIF
ENDIF

END VERTEX CHECK

IF (ABS(Y).LT.TOL) THEN
SY = 0.
ELSEIF (Y.LT.0.0) THEN
SY = -1
ELSE
SY = 1
ENDIF

IF (DIAGS) THEN
WRITE(*, '(/AS0)') ' First calculations'
WRITE(*, '(/AS0,6F18.6)') ' *PO0* Y = ', Y
WRITE(*, '(/AS0,6F18.6)') ' *PO0* SY = ', SY
ENDIF

END

C Get the angle
IF (DIAGS) THEN
  WRITE(*, '(/A33)') ' Getting angles'
ENDIF

NANGS = 0
DO I=1, M
  ANG = ATAN2(-B1(I,I),B1(2,I))
  IF (ABS(ANG) .GT.PIOVR2) THEN
    IF (ANG.LT.0.) THEN
      ANG = ANG+PI
    ELSE
      ANG = ANG-PI
    ENDIF
  ENDIF
  NANGS = NANGS+1
  THETA(NANGS) = ANG
  ITHETA(NANGS) = I
  IF (DIAGS) THEN
    WRITE(*,' A30, ' ) ' R20* I = ', I, ' ANG = ', ANG
  ENDIF
ENDDO

THETA and ITHETA now sorted by control number
Sort THETA by magnitude. ITHETA gets shuffled the same way

IF (DIAGS) THEN
  WRITE(*, ' A30) ') Before sorting'
  WRITE(*, ' A30,3A0,E18.6) ') ' R20* NANGS = ', NANGS
  WRITE(*, ' A30,6E18.6) ') ' R20* THETA = ', (THETA(I), I=1,NANGS)
ENDIF

CALL SORTC(THETA, ITHETA, NANGS,THETA, ITHETA)
IF (DIAGS) THEN
  WRITE(*, ' A50) ') ' After sorting'
  WRITE(*, ' A30,3A0,E18.6) ') ' R20* NANGS = ', NANGS
  WRITE(*, ' A30,6E18.6) ') ' R20* THETA = ', (THETA(I), I=1,NANGS)
ENDIF

FIND INDEX OF ZERO

DO I=1, NANGS
  IF (THETA(I).GT.0.) THEN
    INDX = I
    DO J=NANGS,I,-1
      THETA(J+1) = THETA(J)
      ITHETA(J+1) = ITHETA(J)
    ENDDO
    THETA(I) = 0.
    ITHETA(I) = 0
    GOTO 193
  ENDIF
ENDDO
193 CONTINUE
IF (DIAGS) THEN
  WRITE(*, '/A50') ' After ma=xxx'
  WRITE(*, '/A30,13') '*ROC* INDEX = ', INDEX
  WRITE(*, '/A30,713') '*ROC* THETA = ', (ITHETA(I), I=1,NANGS+1)
  WRITE(*, '/A30,721.6') '*ROC* THETA = ', (THETA(I), I=1,NANGS+1)
ENDIF

CALL GETEDGE(U1, IU, INFRONT, ISVERTEX,
  & Bl, UMAX, Y, SY, ITHETA, NANGS, U_MIN, U_MAX, INDEX, TOL, M, DIAGS)

IF (DIAGS) THEN
  WRITE(*, '/A50') ' Exiting ROC'
  WRITE(*, '/A30,611') '*ROC* U1 = ', (U1(I), I=1,M)
  WRITE(*, '/A30,13') '*ROC* IU = ', IU
  WRITE(*, '/A30,714.6') '*ROC* INFRONT = ', INFRONT
  WRITE(*, '/A30,13') '*ROC* ISVERTEX = ', ISVERTEX
ENDIF

RETURN
END

----------------------------------------------------------------------------------

SUBROUTINE GETEDGE(U2, JU, INFRONT, ISVERTEX,
  & Bl, UMAX, Y, SY, ITHETA, NANGS, U_MIN, U_MAX, INDEX, TOL, M, DIAGS)

IMPLICIT NONE

INTEGER*4 M, SY, UM_X(20)
REAL*4 BI(3,20), U_MAX(20, XU(20), Y
REAL*4 INFRONT, TOL
INTEGER*4 U2(20), JU, SSY
LOGICAL*4 ISVERTEX, DIAGS STUCK
INTEGER*4 I, J, ICOUNT, IX
REAL*4 XK, XU2(20), YPREV, Z, ZI, Z2
SAVE_Y, MOM (3)

IF (DIAGS) THEN
  WRITE(*, '/A50') ' Entering GETEDGE'
ENDIF

DO I=I,M
  U2(I) = I/MAX(I)
ENDDO

SAVE Y = Y
IX = INDX-SY
SSY = SY
STUCK = .FALSE.
ICOUNT = 0
IF (DIAGS) THEN
  WRITE(*, '/A50') ' Beginning GETEDGE DO=I,M'
ENDIF

DO I=1,M
  U2(I) = UMAX(I)
ENDDO

SAVE_Y = Y
IX = INDX-SY
SSY = SY
STUCK = .FALSE.
ICOUNT = 0
IF (DIAGS) THEN
  WRITE(*, '/A50') ' Beginning GETEDGE DO=I,M'
ENDIF

DO I=1,M
  U2(I) = UMAX(I)
ENDDO
WRITE(*, '(A30,EL3.6)') ' GETEDGE* SAVE_Y = ', SAVE_Y
WRITE(*, '(A30,13I3)') ' GETEDGE* IX = ', IX
WRITE(*, '(A30,13I3)') ' GETEDGE* SSY = ', SSY
WRITE(*, '(A30,13I3)') ' GETEDGE* NANGS = ', NANGS
ENDIF
DO WHILE ((SY.EQ.SSY) .AND. (IX.GT.0) .AND. (IX.LE.NANGS))
   ICOUNT = ICOUNT+1
   YPREV = Y
   JU = ITHETA(IX)
   Y = Y-U2(JU)*B1(2, JU)*(U_MAX(JU)-U_MIN(JU))
   SSY = 1
   IF (Y.LT.0.0) SSY = -1
   IF (ABS(Y).LT.TOL) THEN
      Y = 0.
      SSY = 0
   ENDIF
   U2(JU) = -U2(JU)
   IF (SY.NE.0.) THEN
      IX = IX-SY
   ELSE
      IX = IX-1
   ENDIF
   ICOUNT = ICOUNT+1
   IF (ICOUNT GT.20) STUCK = .TRUE.
   IF (DIAGS) THEN
      WRITE(*, '(A30)') ' Bottom of GETEDGE DO WHILE'
   ENDIF
ENDDO
IF (STUCK) THEN
   PAUSE('Stuck in GETEDGE in DO WHILE')
   STUCK = .FALSE.
ENDIF
IF (SY.EQ.SSY) THEN
   STUCK = .FALSE.
   ICOUNT = 0
   IF (DIAGS) THEN
      WRITE(*, '(A30)') ' Beginning Alt. GETEDGE DO WHILE'
   ENDIF
   WRITE(*, '(A30,6I3)') ' GETEDGE* U2 = ', (U2(I), I=1,M)
   WRITE(*, '(A30,EL4.6)') ' GETEDGE* SAVE_Y = ', SAVE_Y
   WRITE(*, '(A30,13I3)') ' GETEDGE* IX = ', IX
   WRITE(*, '(A30,13I3)') ' GETEDGE* SSY = ', SSY
   WRITE(*, '(A30,13I3)') ' GETEDGE* NANGS = ', NANGS
ENDIF
1142 C
1143 Y = SAVE_Y
1144 IX = INDX+SY
1145 DO WHILE ((SY.EQ.SSY).AND.(IX.GT.0).AND.(IX.LE.NANGS+1))
1146 YPREV = Y
1147 JU = ITHETA(IX)
1148 Y = Y-U2(JU)*B1(2,JU)*(U_MAX(JU)-U_MIN(JU))
1149 SSY = 1
1150 IF (Y.LT.0.0) SSY = -1
1151 IF (ABS(Y).LT.TOL) THEN
1152 Y = 0.
1153 SSY = 0
1154 ENDF
1155 U2(JU) = -U2(JU)
1156 IF (SY.NE.0.) THEN
1157 IX = IX+SY
1158 ELSE
1159 IX = IX+1
1160 ENDF
1161 C
1162 ICOUNT = ICOUNT+1
1163 IF (ICOUNT.GT.20) STUCK = .TRUE.
1164 IF (DIAGS) THEN
1165 WRITE(*, '(/A30,L3)') ' Bottom of Alt. GETEDGE DOWHILE'
1166 WRITE(*, '(/A30,L3)') ' GETEDGE* ICOUNT = ', ICOUNT
1167 WRITE(*, '(/A30,L3)') ' GETEDGE* STUCK = ', STUCK
1168 WRITE(*, '(/A30,L3)') ' GETEDGE* YPREV = ', YPREV
1169 WRITE(*, '(/A30,L3)') ' GETEDGE* JU = ', JU
1170 WRITE(*, '(/A30,L3)') ' GETEDGE* SY = ', SSY
1171 WRITE(*, '(/A30,L3)') ' GETEDGE SY.EQ.SSY = ', (SY.EQ.SSY)
1172 WRITE(*, '(/A30,L3)') ' GETEDGE* U2 = ', (U2(I), I=1,M)
1173 WRITE(*, '(/A30,L3)') ' GETEDGE* IX = ', IX
1174 IF (STUCK) THEN
1175 PAUSE('Stuck in GETEDGE in Alt. DOWHILE')
1176 ENDF
1177 IF (SY.EQ.SSY) THEN
1178 JU = 0
1179 INFRONT = 0.
1180 ISVERTEX = .FALSE.
1181 RETURN
1182 C
1183 X = YPREV/(YPREV-Y)
1184 CALL SETU(XU2, U2, U_MIN, U_MAX, M)
1185 Z2 = 0.
1186 DO I=1,M
1187 Z2 = Z2+B1(3,I)*XU2(I)
1188 ENDDO
1189 Z1 = Z2-U2(JU)*B1(3,JU)*(U_MAX(JU)-U_MIN(JU))
Z = XK*Z2+(1.-XK)*Z1

IF (Z.LT.0.) INFRONT = -1.

ISVERTEX = ((Y.EQ.0 ) .AND. (ABS(Z) .LT.TOL))

IF (DIAGS) THEN
  WRITE(*,' 'A9,2 _ )
  WRITE(* ' A35,61 . _)
  WRITE(* A0,6F14.6) )
  WRITE(* A_C,13} _)
  WRITE(* ' A30,E18.6]
  WRITE(* ' A30,EIS.6,
  WRITE(*,' A30,KlS.6)
  WRITE(*, ' A£:2, Sg_.6 '
  WRITE(*, ' 17._: f l::.6' '
  WRITE(*, ' A! i_ L3 ')  
ENDIF

SUBROUTINE ISFACET(ISOK, IUOUT, JUOUT, U123, & IU, JU, U1, U2, B, M, TOL)

IMPLICIT NONE

LOGICAL*4 ISOK
INTEGER*4 IU, JU, M, UI(20), U2(20), IUOUT, JUOUT, U123(20,3)
REAL*4 B(3,20), TOL

INTEGER*4 UXI(20), UX2(20), I, J, K, II, JJ, K_K
REAL*4 THEMAT(2,2), MATINV(2,2), MATDET, T2(2), TI(3), TESTFACET(20)
REAL*4 TEMP
INTEGER*4 DIM, OBJ(20), UDEF(20), ITF(20), THEOBJ(20)

ISOK = .FALSE.
DO I=1,M
  UX1(I) = U1(I)
  UX2(I) = U2(I)
ENDDO
U1(IU) = 0
U2(JU) = 0
DIM = 0
DO I = 1,M
  OBJ(I) = U1(I)
  IF (((I.EQ.IU) .OR. (I.EQ.JU) .OR. (U2(I).NE.U1(I))) ) THEN
    OBJ(I) = 0
    DIM = DIM + 1
    UDEF(DIM) = I
  ENDIF
ENDDO
ISOK = .TRUE.
DO II=1,3
  JJ = MOD(II,3)+1
  KK = MOD(JJ,3)+1
  THEMAT(1,1) = B(II,UDEF(1))
  THEMAT(1,2) = B(JJ,UDEF(1))
  THEMAT(2,1) = B(II,UDEF(2))
  THEMAT(2,2) = B(JJ,UDEF(2))
  MATDET = THEMAT(1,1) * THEMAT(2,2) - THEMAT(1,2) * THEMAT(2,1)
  IF (MATDET.NE.0.) THEN
    MATINV(1,1) = THEMAT(2,2)/MATDET
    MATINV(1,2) = -THEMAT(1,2)/MATDET
    MATINV(2,1) = THEMAT(1,1)/MATDET
    MATINV(2,2) = THEMAT(1,1) / MATDET
    T2(1) = -MATINV(1,1)*B(KK,UDEF(1)) - MATINV(2,1)*B(KK,UDEF(2))
    T2(2) = -MATINV(2,1)*B(KK,UDEF(1)) - MATINV(2,2)*B(KK,UDEF(2))
    T1(KK) = 1.
    T1(II) = T2(1)
    T1(JJ) = T2(2)
  DO I=I,M
    TESTFACET(I) = 0.
    ITF(I) = 0
    UDEF(I) = 0
    DO J=1,3
      TESTFACET(I) = TESTFACET(I) + T1(J)*B(J,I)
    ENDDO ! DO J=1,3
  IF (ABS(TESTFACET(I)).LT.TOL) THEN
    ITF(I) = 0
  ELSEIF (TESTFACET(I).GT.0.) THEN
    ITF(I) = 1
  ELSEIF (TESTFACET(I).LT.0.) THEN
    ITF(I) = -1
  ENDIF
  ENDDO ! DO I=I,M
DIM = 0
DO I=1,M
  THEOBJ(I) = OBJ(I)
  IF ((OBJ(I).EQ.0) .OR. (ITF(I).EQ.0) .OR. (ITF(I).NE.OBJ(I))) THEN
    THEOBJ(I) = 0
    DIM = DIM + 1
    UDEF(DIM) = I
  ENDIF
ENDDO ! DO I=1,M
IF (DIM.NE.2) THEN
  DIM = 0
  J = 0
  DO I=1,M
    THEOBJ(I) = OBJ(I)
    IF ((OBJ(I).EQ.0) .OR. (ITF(I).EQ.0) .OR. (-ITF(I).NE.OBJ(I))) THEN
      THEOBJ(I) = 0
      DIM = DIM + 1
      UDEF(DIM) = I
    ENDIF
  ENDDO ! DO I=1,M
ENDIF : IF (DIM.NE.2) THEN
IF (DIM.NE.2) ISOK = .FALSE.
GOTO 194
ENDIF ! IF (MATDET.NE.0.) THEN
ENDDO ! DO II=1,3
ENDIF ! IF (DIM.EQ.2)
C
IF (ISOK) THEN
IUOUT = UDEF(1)
JUOUT = UDEF(2)
DO I=1,M
DO J=1,3
U123(I,J) = OBJ(I)
ENDDO
ENDIF
U123(IUOUT,1) = 1
U123(JUOUT,1) = 1
U123(IUOUT,2) = -1
U123(JUOUT,2) = 1
U123(IUOUT,3) = 1
U123(JUOUT,3) = -1
ELSE
IUOUT = 0
JUOUT = 0
DO I=1,M
DO J=1,3
U123(I,J) = 0
ENDDO
ENDIF
RETURN
END

SUBROUTINE SORTC(X, IY, N, XS, IYC)

IMPLICIT NONE
INTEGER*4 I, IL_SORTC(36), IMED, IP1, IPR, ITY, IU_SORTC(36)
INTEGER*4 IY(20), IYC(20), J, JMI, JMK, K, L, LMI, M, MID, N
REAL*4 HOLD, AMED, TX, X(20), XS(20)

CHECK THE INPUT ARGUMENTS FOR ERRORS
IPR=11
IF(N.LT.1)GOTO50
IF(N.EQ.1)GOTO55
HOLD=X(1)
DO60I=2,N
IF(X(I) .NE.HOLD)GOTO90
60 CONTINUE
COPY THE VECTOR X INTO THE VECTOR XS
DO100I=1,N
XS(I)=X(I)
100 CONTINUE

COPY THE VECTOR IY INTO THE VECTOR IYC
DO150I=1,N
IYC(I)=IY(I)
150 CONTINUE

CHECK TO SEE IF THE INPUT VECTOR IS ALREADY SORTED

NM1=N-1
DO200I=1,NM1
IP1=I+1
IF(XS(I).LE.XS(IP1))GOTO200
GOTO250
200 CONTINUE

RETURN

250 M=1
I=I
J=N
305 IF(I.LE.J)GOTO370
310 K=I
MID=(I+J)/2
AMED=XS(MID)
IMED=IYC(MID)
IF(XS(I).LE.AMED)GOTO320
XS(MID)=XS(I)
IYC(MID)=IYC(I)
GOTO320
320 XS(I)=AMED
325 RETURN
IYC(I) = IMED
AMED = XS(MID)
IMED = IYC(MID)

320 L = J
IF (XS(J) .GE. AMED) GOTO 340
XS(MID) = XS(J)
IYC(MID) = IYC(J)
XS(J) = AMED
IYC(J) = IMED
AMED = XS(MID)
IMED = IYC(MID)
IF (XS(I) .GE. AMED) GOTO 340
XS(MID) = XS(I)
IYC(MID) = IYC(I)
XS(I) = AMED
IYC(I) = IMED
AMED = XS(MID)
IMED = IYC(MID)
GOTO 340

330 XS(L) = XS(K)
IYC(L) = IYC(K)
XS(K) = TX
IYC(K) = ITY

340 L = L - I
IF (XS(L) .GT. AMED) GOTO 340
TX = XS(L)
ITY = IYC(L)

350 K = K + 1
IF (XS(K) .LT. AMED) GOTO 350
IF (K .LE. L) GOTO 330
LMI = L - I
JM = J - K
IF (LMI .LE. JMK) GOTO 360
IL_SORTC(M) = I
IU_SORTC(M) = L
I = K
M = M + 1
GOTO 380

360 IL_SORTC(M) = K
IU_SORTC(M) = J
J = L
M = M + 1
GOTO 380

370 M = M - 1
IF (M .EQ. 0) RETURN
I = IL_SORTC(M)
J = IU_SORTC(M)

380 JMI = J - I
IF (JMI .GE. 11) GOTO 310
IF (I .EQ. I) GOTO 305
I = I - I
M = M + 1
GOTO 380

390 I = I + 1
IF (I .EQ. J) GOTO 370
AMED = XS(I + 1)
IMED = IYC(I + 1)
IF (XS(I) .LE. AMED) GOTO 390
K = I
SUBROUTINE MINNORM(U_MINNORM, SCALE, P, U, UTRIM, MTRIM, UDA, B, U_MIN, U_MAX, M, TIME, XSCALE)

IMPLICIT NONE

** Parameters

INPUTS:
REAL*4 P(20,3), U(20), UTRIM(20), UDA(20)
REAL*4 B(3,20), MTRIM(3), U_MAX(20), U_MIN(20)
REAL*4 TIME, XSCALE
INTEGER*4 M, IMODE

OUTPUTS:
REAL*4 U_MINNORM(20), SCALE

LOCALS:
REAL*4 UKDA(20), UP(20), UDELTA(20), M_ATT(3)
INTEGER*4 I, J, K

SCALE = I.

DO I=1,M
  UKDA(I) = U(I) + UDA(I)
ENDDO

DO I=1,3
  M_ATT(I)=0.
  DO J=1,M
    M_ATT(I)=M_ATT(I)+B(I,J)*UKDA(J)
  ENDDO
ENDDO

DO I=1,M
  IF ((U_MIN(I).EQ.0.).OR.(U_MAX(I).EQ.0.)) SCALE = 0.
  UKDA(I) = U(I) + UDA(I)
  UP(I) = UTRIM(I)
  DO J=1,3
    UP(I) = UP(I) + P(I,J)*(M_ATT(J)-MTRIM(J))
  ENDDO
ENDDO
UDELTA(I) = SCALE*(UP(I)-UKDA(I))

IF (UDELTA(I).NE.0.) THEN
  IF ((UDELTA(I).GT.U_MAX(I)).AND.(U_MAX(I).GT.0.)) THEN
    SCALE = U_MAX(I)/UDELTA(I)
  ELSEIF ((UDELTA(I).LT.U_MIN(I)).AND.(U_MIN(I).LT.0.)) THEN
    SCALE = U_MIN(I)/UDELTA(I)
  ENDIF
ENDIF
ENDDO

SCALE = XSCALE*SCALE

DO I=1,M
  UMINNORM(I) = UKDA(I) + SCALE*(UP(I)-UKDA(I))
ENDDO

RETURN
END

C

C ..............................................................................