1989

NASA/ASEE SUMMER FACULTY FELLOWSHIP PROGRAM

MARSHALL SPACE FLIGHT CENTER
THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

ITERATIVE PROCEDURES FOR SPACE SHUTTLE MAIN ENGINE PERFORMANCE MODELS

Prepared by:	L. Michael Santi
Academic Rank:	Associate Professor
University and Department:	Memphis State University Mechanical Engineering Department
NASA/MSFC:	
Laboratory:	Propulsion
Division:	Propulsion Systems
Branch:	Performance Analysis
MSFC Colleague:	Charles F. Schafer
Date:	August 8, 1989
Contract Number:	The University of Alabama in Huntsville NGT-01-008-021

ABSTRACT

Performance models of the Space Shuttle Main Engine (SSME) contain iterative strategies for determining approximate solutions to nonlinear equations reflecting fundamental mass, energy, and pressure balances within engine flow systems. Both univariate and multivariate Newton-Raphson algorithms are employed in the current version of the engine Test Information Program (TIP). Computational efficiency and reliability of these procedures is examined. A modified trust region form of the multivariate Newton-Raphson method is implemented and shown to be superior for off nominal engine performance predictions. A heuristic form of Broyden's Rank One method is also tested and favorable results based on this algorithm are presented.

ACKNOWLEDGEMENTS

I would like to express my sincere appreciation to all those individuals who have contributed to this work by their advice, assistance, and encouragement. Gratitude is expressed to Dr. Charles Schafer for his organized approach to defining this effort, suggestions, and periodic review of results. I also wish to acknowledge the efforts Mr. Brian Piekarski of Martin Marietta Corporation for his assistance in preparing and organizing documentation associated with the TIP code. Special thanks are extended to Mr. John Butas for his valuable advice and assistance during the course of this investigation.

INTRODUCTION

Predictions of the steady-state operational characteristics of the Rocketdyne Space Shuttle Main Engine (SSME) are provided by computer programs which model and analyze engine sytem performance. The Test Information Program (TIP) is a FORTRAN based engine analysis software package which performs the following functions.

1. Power Balance - Simulates engine performance by balancing mass and energy flows for assumed nominal operation of SSME components.

2. Data Reduction - Uses actual test data to define operating characteristics of a specific SSME by adjusting component performance parameters.

3. Base Balance - Refines operating predictions of the Data Reduction portion by adjusting nine performance variables.

4. Rated Portion - Extends refined performance model to simulate actual engine operation over a range of conditions.

TIP balances mass and energy flows during engine performance prediction, and balances theoretical results with test information to refine these predictions. Figure 1 displays the SSME flow system network that is balanced by TIP to ensure satisfaction of the conservation of mass and conservation of energy principles as well as adherence to limitations imposed by the Second Law of Thermodynamics.

Since flow processes within the SSME are governed by a set of nonlinear equations, iterative techniques are required to computationally predict a balanced steady-state flow condition. Two subroutines within the TIP code perform iterative nonlinear equation solving functions. These routines are described below.

TIP Iteration Subroutines

1. NLEST - a single nonlinear equation root finding routine
   CALL statement accessed as subroutine NLEST
   ENTRY statement accessed as subroutine NLREST

   NLEST - Find z such that $(x-y) = F(z) = 0$.
   NLREST - Find x such that $(x-y) = F(x) = 0$.

ORIGINAL PAGE IS
OF POOR QUALITY

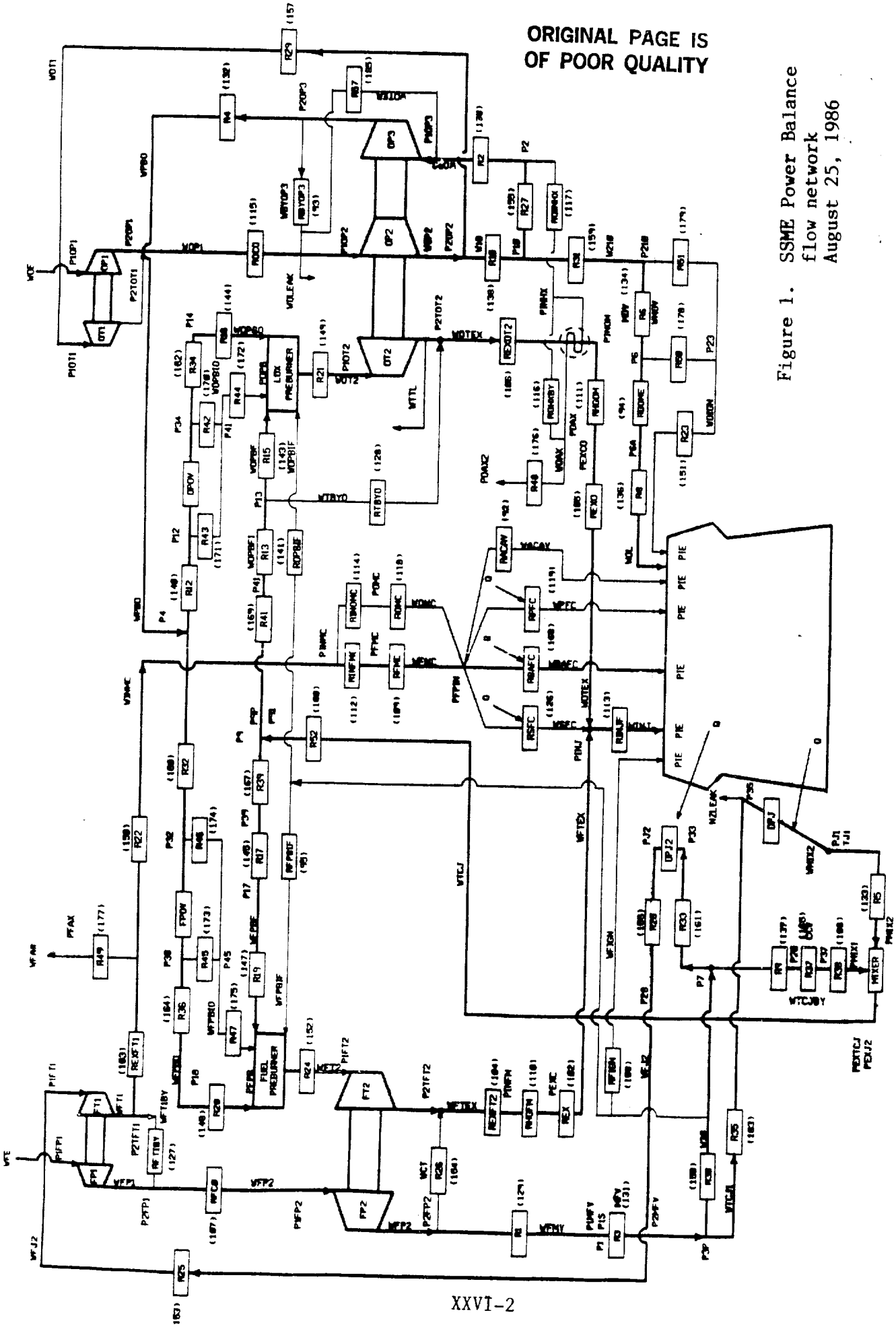


Figure 1. SSME Power Balance flow network August 25, 1986

2. NLESTN - a simultaneous nonlinear equation solver
CALL statement accessed as subroutine NLESTN

NLESTN - Find $\mathbf{x} = (x1, x2 \ldots xN)$ such that

$$y1 = F1(\mathbf{x}) = 0$$
$$y2 = F2(\mathbf{x}) = 0$$
$$\vdots$$
$$yN = FN(\mathbf{x}) = 0$$

The basic function of TIP is to provide steady-state operation and performance predictions. Engine flow conditions are established by solving systems of nonlinear balance equations arising from fundamental flow mechanics. Hence, use of the above described solver routines is fundamental to and pervasive within TIP as evidenced by the ninty-four calls to NLEST (or NLREST) and NLESTN which occur in the current version of this performance analysis package.

In addition to being pervasive, the iteration loops initiated by calls to subroutines NLEST and NLESTN are programmed in a complex sequence with considerable nesting and crossover. As an example of this complexity, consider the iteration loop sequencing of subroutine BAL exhibited in Appendix 1. Subroutine BAL is the routing routine for the Power Balance component of TIP. Significant iteration loop nesting, as many as seven loops deep, and loop crossover are apparent upon examination of this listing.

OBJECTIVES

Because of the intrinsic importance of the iterative nonlinear equation solvers in the TIP code, it is important that these procedures be both reliable and efficient. The objectives of this research effort are

1. To evaluate the iterative schemes employed within the TIP performance model

2. To modify and test these schemes as suitable to achieve greater reliability and efficiency

3. To perform a cursory review of fundamental TIP code logic and procedure

NONLINEAR EQUATION SOLVERS

The nonlinear equation root finding scheme currently employed in the TIP subroutine NLEST is a heuristic procedure which combines a secant method search with a false positions method (see e.g. Burden and Faires [1]). Initially the secant method is used to update the root approximation. Whenever successive applications of the secant method establish a bracket about the root, NLEST switches to a false positions technique. It is possible to lose the established root bracket due to loop nesting and the accompanying effects of shifting due to inner loop convergence tolerances. Because of this an ad hoc strategy for switching between secant and false positions searches is employed.

Within the context of the TIP code, the existing root finding scheme in NLEST was tested against several other well known procedures. The procedures implemented and tested by this author are listed below.

1. Finite difference Newton-Raphson
2. Secant (only)
3. False positions with step bracketing
4. Quadratic interpolation polynomial

In few instances were results superior to those obtained using the current NLEST algorithm, and in many cases the overall performance and convergence of the TIP code were adversely affected. Hence, the heuristic root finding strategy implemented within subroutine NLEST appears to be a robust and effective method within the TIP performance analysis model.

The simultaneous nonlinear equation solver currently implemented within subroutine NLESTN is a finite difference multivariate Newton-Raphson method (often referred to as simply Newton's method, see e.g. reference [1], page 486) with a direction skewing trust region boundary. To evaluate the effectiveness of this algorithm it is necessary to refer to the theoretical basis of the Newton-Raphson method. This basis will be outlined below.

The fundamental problem addressed by simultaneous nonlinear equation solvers can be expressed mathematically as follows.

Find $\mathbf{x} = (x1, x2 \ldots xN)^T$ in R^N

such that $\mathbf{F} = (F1(\mathbf{x}), F2(\mathbf{x}) \ldots FN(\mathbf{x}))^T = \mathbf{0}$

or equivalently

$$\begin{aligned} F1(\mathbf{x}) &= 0 \\ F2(\mathbf{x}) &= 0 \\ &\vdots \\ FN(\mathbf{x}) &= 0 \end{aligned} \qquad (1)$$

where $\mathbf{F} : R^N \longrightarrow R^N$

At stage k the fundamental iteration of the Newton-Raphson method for solving problems of this type can be written

$$\mathbf{x}_{k+1} = \mathbf{x}_k - J_k^{-1} * \mathbf{F}_k \qquad (2)$$

where

$$\mathbf{F}_k = \mathbf{F}(\mathbf{x}_k)$$

and

J_k = the Jacobian of $\mathbf{F}$ evaluated at $\mathbf{x}_k$

Rapid convergence of the Newton-Raphson method to the solution of (1) can be established rigorously if, for some k, $\mathbf{x}_k$ is

sufficiently close to the solution vector designated $\mathbf{x}^*$. Formal conditions for convergence are stated in the following theorem (see e.g. Rheinboldt [2]).

Theorem.
If $\mathbf{F}$ has continuous first partial derivatives in some neighborhood of the solution $\mathbf{x}^*$, if the Jacobian of $\mathbf{F}$ is nonsingular at $\mathbf{x}^*$ and its elements satisfy a Lipschitz condition, and if $\mathbf{x}_k$ is sufficiently close to $\mathbf{x}^*$ for some k, then the Newton-Raphson method is well defined for all k and converges at second order, i.e. there exits a positive integer m and a positive real number b such that

$$\|\mathbf{F}_{k+1}\| \; / \; \|\mathbf{F}_k\|^2 \; < \; b \quad \text{whenever} \quad k \; > \; m$$

Despite the second order (or quadratic) convergence indicated in the above theorem, the Newton-Raphson method suffers from two serious disadvantages from the point of view of practical calculation. First, computation of the Jacobian matrix at each stage of the iteration is extremely costly in terms of computer resources. Often analytical partial derivatives are not available and finite difference approximations lack the precision necessary for ultimate convergence. The second disadvantage of the Newton-Raphson method arises from the need to have a

sufficiently accurate initial estimate of the solution in order to guarantee convergence. Satisfaction of this requirement is impossible to measure before initiation of the iteration sequence and often difficult to obtain in practice.

Extension of the basic Newton-Raphson procedure to include a subiteration or line search have been somewhat successful in removing the accurate initial estimate requirement. These methods include a strategy to select a positive real number lambda such that the iteration scheme

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \text{lambda} * J_k^{-1} * \mathbf{F}_k \tag{3}$$

reduces some measure of error, typically

$$\|\mathbf{F}_{k+1}\| < \|\mathbf{F}_k\| \tag{4}$$

at each stage. Although Newton-Raphson methods incorporating line searches extend the domain of convergence, they do so with significant computational overhead.

Trust region or restricted step methods are a compromise between convergence limited self-scaling iteration procedures and computationally intense methods incorporating line searches. These methods simply provide an upper bound on the distance between iteration steps. This bound may be absolute or scaled by position within the domain of **x**.

The current version of subroutine NLESTN provides a multivariate Newton-Raphson method with a trust region approach. Unfortunately, the trust region bound is applied componentwise on x which has the effect of skewing the correct Newton-Raphson method search direction. This skewing process removes any theoretical convergence characteristics and indicates the possibility of convergence difficulties when the iteration procedure is initiated at a point remote from the immediate vicinity of the solution. Difficulties of this type are currently experienced as will be discussed in the next section.

In an effort to correct the skewing problem inherent in the current version of NLESTN, a modified trust region form of the multivariate Newton-Raphson method was implemented within TIP. The basic iteration sequence of this method is given by

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \text{lambda} * J_k^{-1} * \mathbf{F}_k \tag{5}$$

where

$$\text{lambda} = \min \left[1, \frac{\|\text{PCTMAX} * \mathbf{x}_k\|}{\|J_k^{-1} * \mathbf{F}_k\|} \right] \tag{6}$$

PCTMAX is a user defined parameter which serves to scale the size of the trust region. A comparison of search steps obtained using the skewed direction trust region approach and the corrected strategy are displayed in Figure 2. A complete listing of the modified trust region form of NLESTN implemented within TIP is given in Appendix 2. Results using this implementation are presented in the next section.

A trust region form of the Broyden Rank One [3] nonlinear system solver was also implemented and tested in an effort to

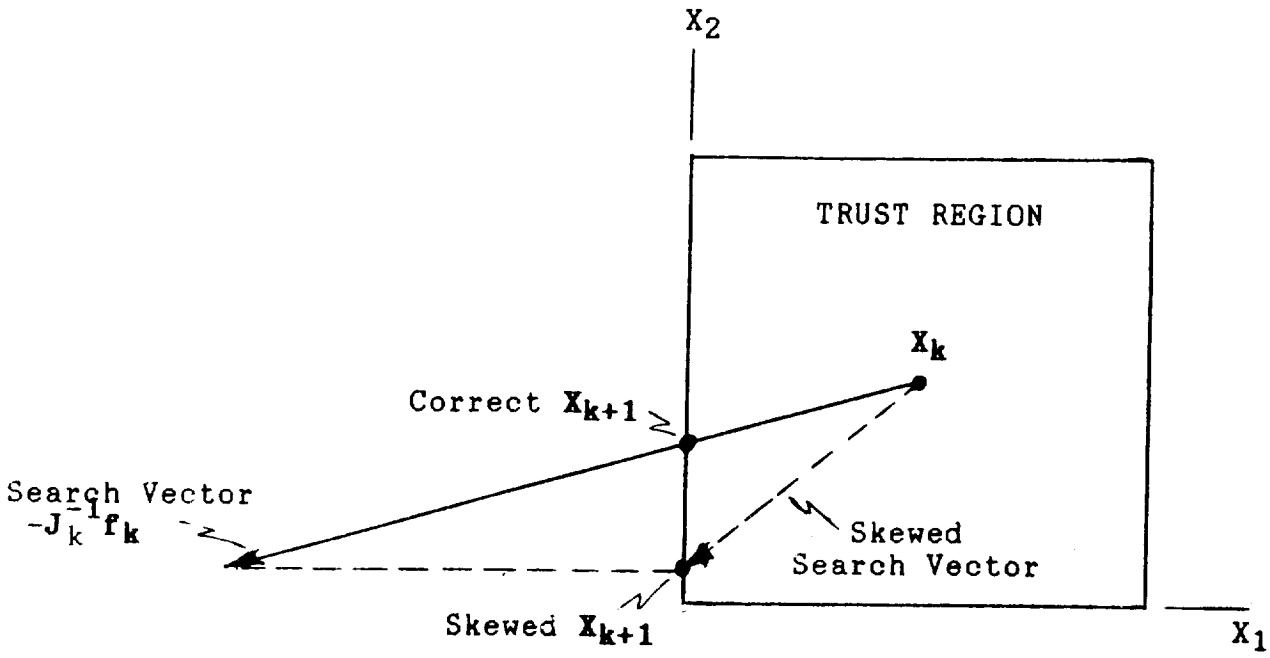


Figure 2. Comparison of skewed trust region search vector and correct trust region search vector.

reduce the computational overhead associated with Jacobian calculations and linear system solvers in the Newton-Raphson algorithm. The Broyden Rank One method is a modified Newton-Raphson iteration in which the Jacobian matrix is replaced by an approximation B which is updated at each iteration by using current information about F. In the Broyden strategy, a rank one modification of B is made at each iteration. The basic method is described below.

Basic Broyden Rank One Method

0. Select x_0 the iteration starting point and
   B_0 an N x N matrix approximation to the inverse Jacobian of F at x_0 .

For k = 0,1,2...

1. Evaluate F_k

2. Set $s_k = - B_k * F_k$

3. Set $x_{k+1} = x_k + s_k$

4. Evaluate F_{k+1}

5. Set $y_k = F_{k+1} - F_k$

6. If $y_k = 0$ Set $B_{k+1} = B_k$

   Otherwise set $B_{k+1} = B_k + (s_k - B_k y_k) * v^T$

   where $v^T = s^T * B_k / (s^T * B_k * y_k)$

Theoretical convergence properties of the basic Broyden Rank One algorithm have been clarified by Gay [4,5]. Although not as rapidly convergent theoretically as the Newton-Raphson method, testing has often shown the Broyden technique to be superior [3].

A complete listing of the Broyden Rank One trust region form of NLESTN implemented within TIP is given in Appendix 3. Results using this implementation are presented in the next section.

Function minimization strategies have often been used to solve nonlinear systems. A class of specialized minimization techniques (see e.g. [6]) been developed which address least squares problems of the type

$$\text{minimize} \quad F(x)^T * F(x) \qquad \text{by selection of } x \tag{7}$$

General nonlinear minimization algorithms can also be employed to solve problems such as (7) which arise naturally from nonlinear equation systems. These methods are typically computer intensive although potentially effective and robust.

Although no testing of TIP performance using sophisticated minimization methods was conducted, such techniques hold promise for future application within engine performance models.

COMPUTATIONAL RESULTS

Computations involving the TIP performance model were conducted on the EADS network at Marshall Space Flight Center. An IBM 3084 was the host processor. Within TIP, only Power Balance model prediction analyses were run. Initializing input data was fixed for all computations with the exception of LOX-fuel ratio and power level expressed as a percentage of engine rated power. Values for these flow characteristics were specified within and outside of the nominal operating ranges defined below.

Nominal SSME Operating Range

LOX-Fuel ratio	6.0 - 6.5
Power Level (%)	65 - 109

Variation of these operating control parameters permitted an evaluation of the function of the nonlinear equation solving algorithms over a range of practical problems.

Output from a typical Power Balance model analysis is extensive and includes many operating and performance characteristics of the SSME subsystems. These physical data will not be presented. Since the object of this investigation was to test and evaluate performance model iteration methods, computational performance data involving subroutines NLEST and NLESTN were collected. These data were organized and presented for each convergent Power Balance analysis as shown in Table 1. The prescribed values of mixture ratio and power level for the specific analysis are displayed at the top of this table. In addition, the univariate and multivariate iteration procedures used in the particular analysis are presented. Performance evaluation data for each iterative procedure are presented as described below.

NLEST Loop Summary Information

Loop	-	iteration loop identification number
Entered	-	number of times specific iteration loop was entered
Closed	-	number of times specific iteration loop successfully converged and closed
Max	-	maximum number of iterations required for convergence of specific iteration loop
Average	-	average number of iterations required for convergence of specific iteration loop

Table 1. Iteration loop performance data for a typical Power Balance model analysis.

MIXTURE RATIO - 6.5 POWER LEVEL - 109 %

NLEST - ORIGINAL
NLESTN - BROYDEN RANK ONE

NLEST LOOP SUMMARY

LOOP	ENTERED	CLOSED	MAX	AVERAGE
1	71	67	3	1.060
2	67	59	3	1.136
3	589	294	3	2.003
5	6	3	2	2.000
6	10	5	2	2.000
7	434	428	3	1.014
8	315	174	3	1.810
9	87	69	2	1.261
11	149	67	7	2.224
12	788	428	2	1.841
14	1002	550	4	1.822
15	550	294	3	1.871
16	1029	510	5	2.018
17	119	69	3	1.725
18	1100	428	4	2.570
19	657	290	5	2.266
20	679	290	4	2.341
21	13	5	3	2.600
23	1113	428	4	2.600
24	2108	720	3	2.928
25	861	428	3	2.012
26	1698	788	3	2.155
27	1737	856	3	2.029
28	977	428	4	2.283
30	117	69	3	1.696
31	5528	2517	3	2.196
32	191	69	8	2.768
35	442	69	8	6.406
36	3	1	3	3.000
37	5	1	5	5.000
38	429	428	2	1.002
39	191	191	1	1.000
40	5	1	5	5.000
42	856	428	2	2.000
43	69	19	6	3.632
45	69	69	1	1.000
46	769	408	4	1.885
48	388	294	2	1.320
49	431	294	2	1.466
50	174	87	2	2.000
60	594	360	2	1.650
61	177	122	3	1.451
62	122	67	3	1.821
67	3	1	3	3.000
70	4	1	4	4.000

NLESTN SUMMARY

LOOP	ENTER	JACOB	CLOSE	MAX	JE/C	LP/C
1	510	157	149	5	1.054	3.423
2	19	8	1	8	8.000	19.000
3	769	361	136	3	2.654	5.654
4	294	65	69	3	0.942	4.261
5	286	79	69	3	1.145	4.145
6	3375	675	675	1	1.000	5.000

NLESTN Loop Summary Information

Loop - same as above
Enter - same as above for entered
Jacob - total number of Jacobian or Jacobian approximation evaluations
Close - same as above for closed
Max - same as above
JE/C - average number of Jacobian evaluations per convergent iteration
LP/C - average number of iteration loop passes per convergent iteration

The amount of effort expended in iteration processes is evident from Table 1, with over 15,000 univariate loop entries and over 5,000 multivariate loop entries documented for this specific Power Balance analysis in order to achieve convergence. This effort level is typical of convergent analyses performed in this study.

In order to compare the efficiency and reliability of various NLESTN implemented multivariate nonlinear equation solving methods, the total number of NLESTN loop entries was tabulated for each of several analyses using different multivariate iteration strategies. These data are presented in Table 2. Examination of the information in Table 2 suggests the following

Conclusions Based on Table 2 Data

1. The current (N-R Orig) Newton-Raphson implementation often fails to converge for mixture ratios or power levels outside the nominal region. This result was expected since the direction skewing trust region method forces the iteration sequence to take less appropriate steps for conditions that cause the solution to be further removed from the initiation data. It is notable that within the nominal region for mixture ratio and power level, the current Newton-Raphson implementation performs almost the same as the modified method (N-R Mod). This occurs because within the nominal operating range, the trust region boundary is not reached during the search process since the initiation data is close to the converged solution.

2. The modified Newton-Raphson method (N-R Mod) with corrected trust region is more efficient, requiring fewer total NLESTN loop passes, for conditions outside and on the boundary of the nominal regions for mixture ratio and power level. This is due to the corrected search direction method employed at the trust region boundary. In addition, the modified Newton-Raphson method is more reliable than the original method, converging for several cases with outside nominal mixture ratios or power levels.

3. The Broyden Rank One method implementation is extremely reliable, having converged for all cases considered. The efficiency of this method is somewhat erratic, often requiring substantially fewer loop passes for convergence than the Newton-Raphson methods and yet occasionally requiring substantially more effort to arrive at a converged solution. This erratic behavior was not wholly unexpected due to the approximate nature of the Jacobian estimate employed and updated by the algorithm.

Further comparisons of the modified Newton-Raphson and Broyden Rank One methods with the current NLESTN implementation are presented in Table 3. Results are presented only for analyses in which the current multivariate iteration strategy achieved convergence. The improved efficiency gained by use of the modified Newton-Raphson method outside the nominal operating range is again evident. The erratic efficiency of the Broyden method is clearly displayed.

Table 2. Loop passes through multivariate subroutine NLESTN.

MR	PL(%)	N-R Orig	N-R Mod	Broyden
6.0	65	9,077	9,077	2,527
6.0	100	10,586	10,586	6,099
6.011	104	4,993	4,993	14,711
6.5	104	9,152	9,152	6,515
6.5	109	13,308	7,209	5,253
6.6	109	F	11,491	8,009
6.7	109	F	11,514	13,434*
6.8	109	F	24,935*	10,303*
6.5	112	11,757	11,723	13,309
6.5	115	15,118	12,213	21,829
6.5	120	F	F	7,809

F - Failure to converge to specified tolerance in allowed number of iterations

* - Trust region interval reduced to +- 5% of current independent variable value

Table 3. Change (%) in number of loop passes through subroutine NLESTN using the original Newton-Raphson method results as standard.

		% Change	
MR	PL(%)	N-R Mod	Broyden
6.0	65	0	- 72.2
6.0	100	0	- 42.4
6.011	104	0	+194.6
6.5	104	0	- 28.8
6.5	109	- 45.8	- 60.5
6.5	112	- 0.3	+ 13.2
6.5	115	- 19.2	+ 44.4

RECOMMENDATIONS

The following recommendations for improvement of the iterative procedures within the Test Information Program are motivated by the computational results described in the previous section of this report and by investigation of TIP logic.

Recommendations for Improvement of Iterative Procedures

1. Immediately implement modified multivariate Newton-Raphson method with corrected trust region approach in subroutine NLESTN. The subroutine described in Appendix 2 is one implementation of this method.

2. Continue to test and refine the potentially effective Broyden Rank One method for the iterative solution of simultaneous nonlinear equations. The subroutine described in Appendix 3 is one implementation of this method.

3. Perform computational experimentation using flexible loop tolerances and flexible trust region bounds in the iterative routines. These modifications could substantially improve the efficiency of the TIP iteration sequence.

4. Incorporate and test a formal line search algorithm within the multivariate iteration scheme to enhance convergence and reliability for strongly off nominal engine operation.

5. Perform a detailed sensitivity analysis of all iteration loop independent variables to determine uncertainty limits associated with loop tolerances.

6. Review iteration loop logic sequencing. Modify and test sequencing to achieve improved computational efficiency.

In addition to the recommendations involving computational procedure listed above, a limited review of the TIP code motivates the structuring recommendations provided below.

Code Structuring Recommendations

1. Clearly identify and separate TIP program components.
   Theoretical base (flow physics)
   Computational base (formal numerical algorithms)
   Experimental base (engineering performance parameters and other approximations)

2. Clearly identify the following.
   Independent variables of model analysis (user defined and controlled physical inputs)
   Arbitrarily prescribed and constant parameters of model analysis (code designer defined and restricted input)
   Dependent variables of model analysis (solution variables requiring initial approximation)

3. Review prescribed flow and performance variable dependencies in model for accuracy and completeness.

4. Formalize an organized data input structure descriptive of SSME flow systems, i.e. number nodes, branches, and devices and formally identify connectivity within the data structure.

5. Fully document TIP program physical logic sequence.

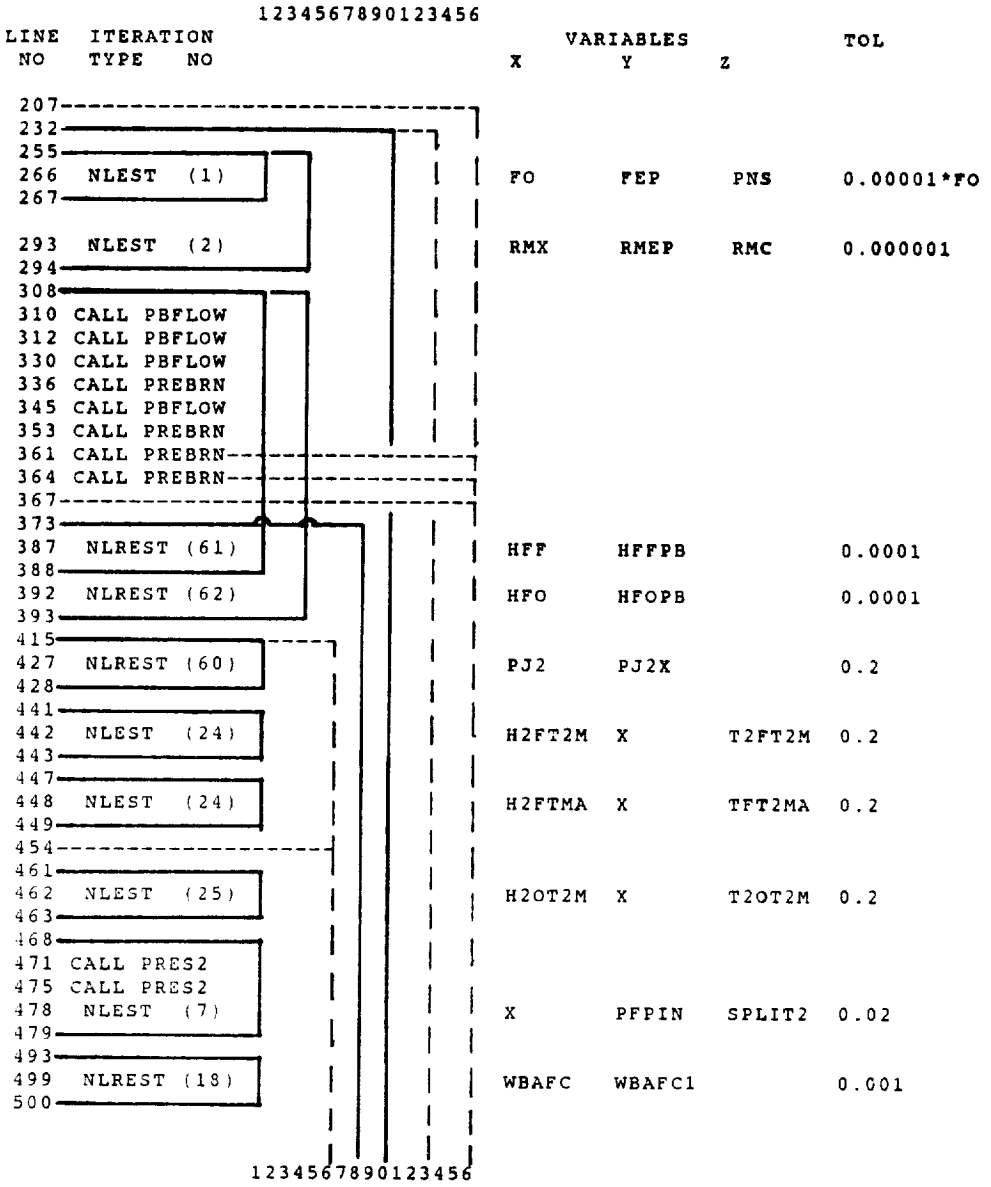
6. Construct postprocessors that clearly exhibit physical balances for appropriate engine subsystems as a means of verification.

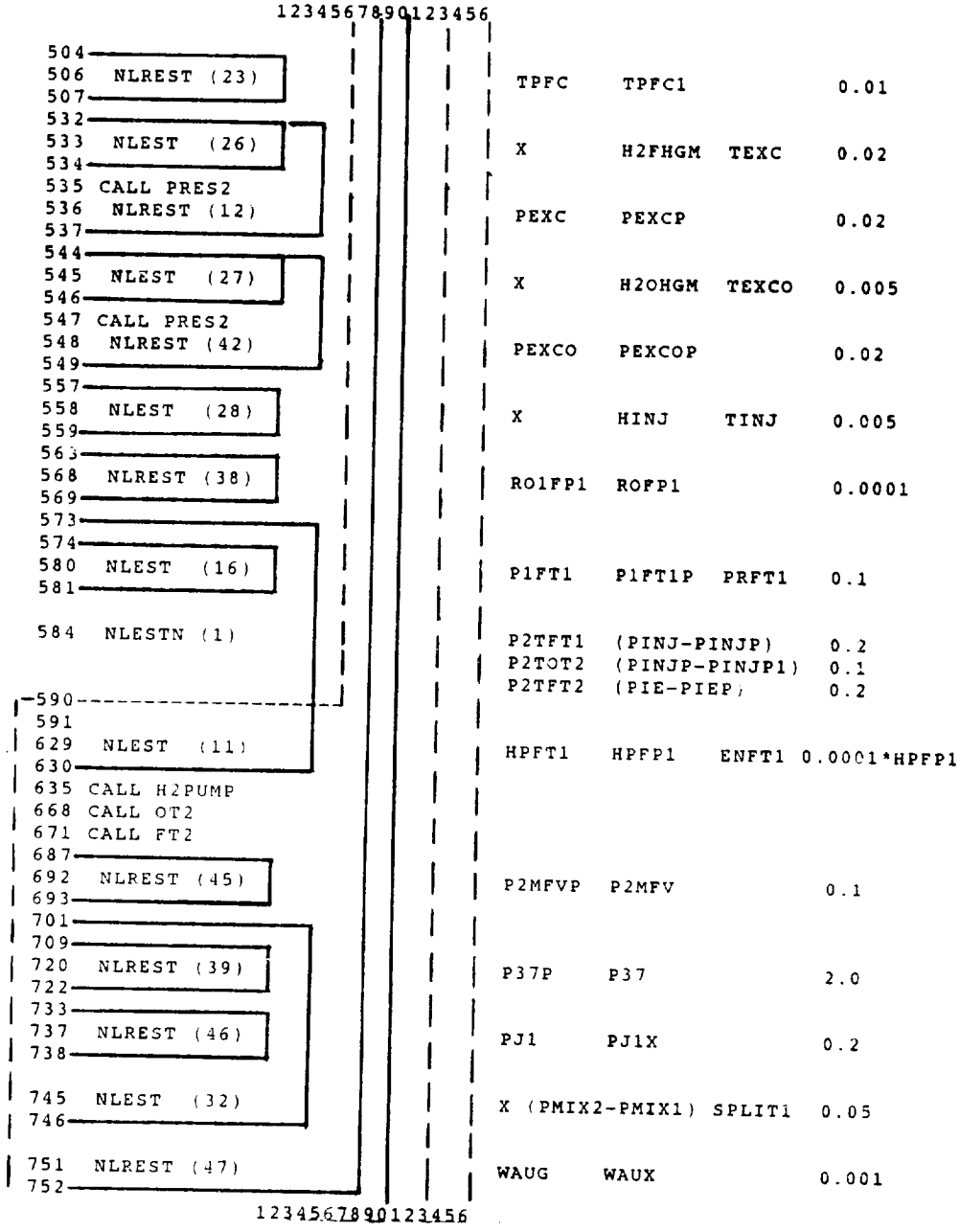
These recommendations are very basic and if implemented will improve confidence in and reliability of TIP analysis results.

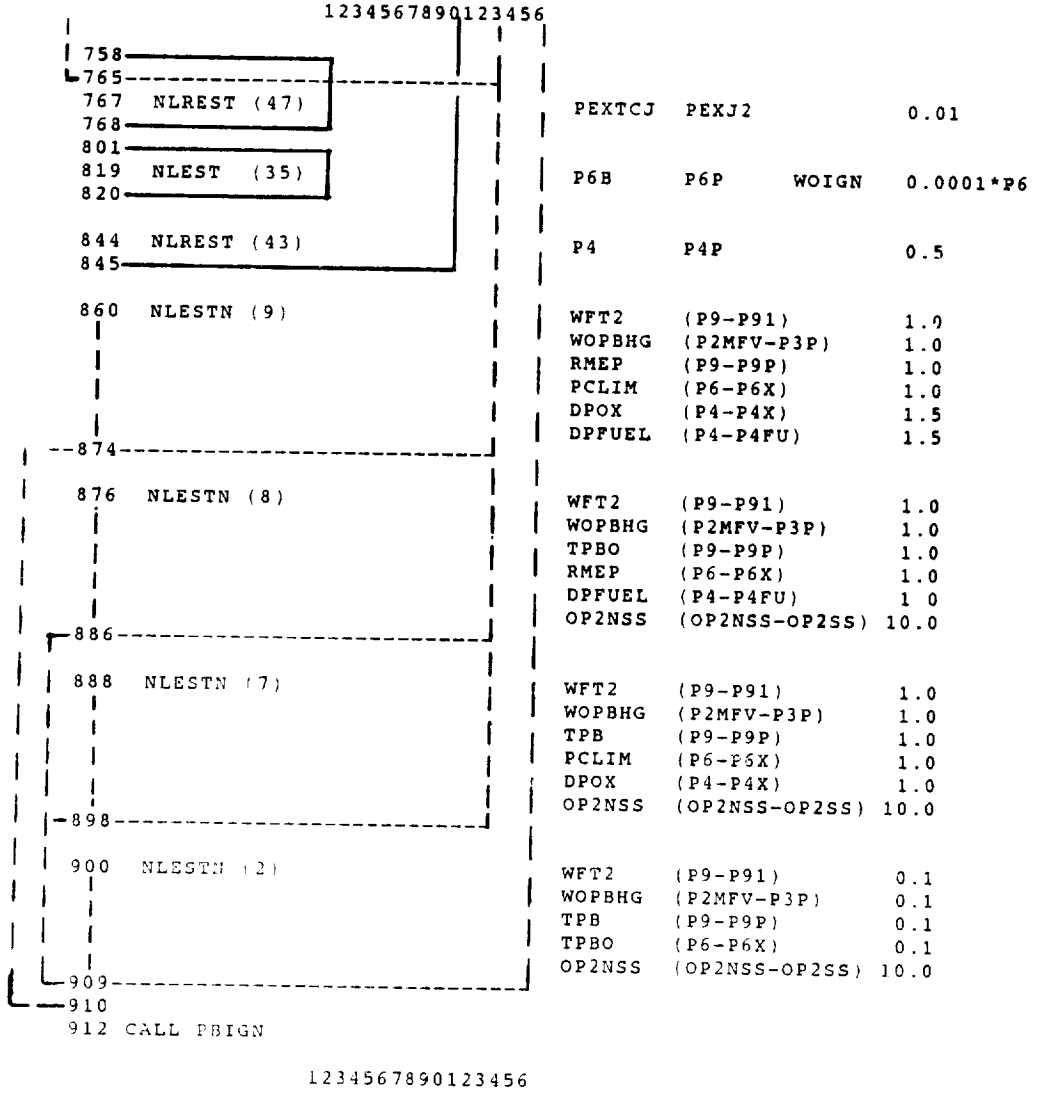
Appendix 1. Iteration loop sequencing in subroutine BAL

TIP88MMC 22 JUN 89

SUBROUTINE BAL - ITERATION LOOP SEQUENCING







Appendix 2. Subroutine NLESTN implementation with corrected trust region Newton-Raphson method.

```
C ********************** CODE DESIGNATION TRUST **********************
C
      SUBROUTINE NLESTN (ID,N,A,KK,X1,Y1,T1,F1,X2,Y2,T2,F2,X3,Y3,T3,F3,
     + X4,Y4,T4,F4,X5,Y5,T5,F5,X6,Y6,T6,F6)
C
      IMPLICIT REAL*8(A-H,O-Z)
C
C  -NLESTN-  2- TO 6-DIMENSIONAL MULTIVARIATE NEWTON-RAPHSON METHOD FOR
C     THE ITERATIVE SOLUTION OF SIMULTANEOUS NONLINEAR ALGEBRAIC
C     EQUATIONS.
C
C     CALL STATEMENT...
C     CALL NLESTN (ID,N,A,KK,X1,Y1,T1,F1,X2,Y2,T2,F2....XN,YN,TN,FN)
C     ...WHERE N IS )=6.
C
C        ID  IS THE ITERATION LOOP NUMBER (MINIMUM VALUE = 1, MAXIMUM
C     VALUE = 10).  EACH NUMBER OF A NEST OF LOOPS MUST HAVE A DIFFERENT
C     LOOP NUMBER, BUT OTHERWISE LOOP NUMBERS ARE ARBITRARY WITHIN THE
C     ABOVE LIMITS.
C        N  IS THE NUMBER OF INDEPENDENT VARIABLES TO BE ITERATED UPON
C     (X1,X2...XN) IN ORDER TO DIMINISH THE DEPENDENT VARAIBLE ERROR
C     VALUES (Y1,Y2...YN) TO WITHIN SPECIFIED TOLERANCES (T1,T2...TN).
C     NOTE- 2 <= N <= 6.
C        (F1,F2...FN) ARE INDEPENDENT VARIABLE INCREMENT MULTIPLIERS
C     USED IN THE FINITE DIFFERENCE APPROXIMATIONS FOR THE JACOBIAN
C     MATRIX PARTIAL DERIVATIVES.
C        A  IS AN N-BY-N MATRIX DIMENSIONED IN THE CALLING PROGRAM.
C        KK  IS A FLAG AS FOLLOWS...
C         KK=-3  SINGULAR MATRIX.
C         KK=-1  ALLOWABLE NUMBER OF ITERATIONS HAS BEEN EXCEEDED.
C         KK= 0  ALL Y'S ARE LESS THAN TOLERANCE, ITERATION COMPLETE.
C         KK= 1  ONE OR MORE Y'S ARE GREATER THAN TOLERANCE, REITERATE.
C
C
      INCLUDE (INSAVE)
      INCLUDE (DPSAVE)
C
      DIMENSION B(36),SX(6),SY(6),ST(6),SF(6),SDF(6)
      DIMENSION D(6,6),A(N,N),C(36)
      DATA LIMIT,PCTMAX/20,0.2/
C
C ************************* ARRAY INITIALIZE SUBROUTINE INPUT ARGUMENTS
      SX(1)=X1
      SX(2)=X2
      SX(3)=X3
      SX(4)=X4
      SX(5)=X5
      SX(6)=X6
C
      SY(1)=Y1
      SY(2)=Y2
      SY(3)=Y3
      SY(4)=Y4
      SY(5)=Y5
      SY(6)=Y6
```

```
C
      ST(1)=T1
      ST(2)=T2
      ST(3)=T3
      ST(4)=T4
      ST(5)=T5
      ST(6)=T6
C
      SF(1)=F1
      SF(2)=F2
      SF(3)=F3
      SF(4)=F4
      SF(5)=F5
      SF(6)=F6
C
C ************************************** STEP 1. CHECK FOR CONVERGENCE
      L=LOOPN(ID)
      NUMN(ID,1)=NUMN(ID,1)+1
      ICONV=0
      DO 10 I=1,N
      IF (ICONV.GT.0) GO TO 10
      IF (DABS(SY(I)).GT.DABS(ST(I))) ICONV=1
   10 CONTINUE
      IF (ICONV.GT.0) GO TO 20
      NUMN(ID,3)=NUMN(ID,3)+1
      NUMN(ID,4)=MAX0(NUMN(ID,4),KOUNL(ID))
      KOUNL(ID)=0
      KK=0
      GO TO 200
C
C ******************************** STEP 2. CHECK FOR EXCESSIVE ITERATIONS
   20 IF (KOUNL(ID).LT.LIMIT) GO TO 30
      WRITE (4,901) LIMIT,ID,(SY(I),ST(I),SX(I),I=1,N)
  901 FORMAT (
     +  1H ,'ERROR IN NLESTN, NO SOLUTION WITHIN',I3,'ITERATIONS',/,
     +  1H ,'FIRST ARGUMENT IN THE CALL STATEMENT IS',I5,/,
     +  1H ,'ERROR VALUES    TOLERANCES INDEPENDENT VARIABLES',/,
     +       3(G15.6))
      K=1
      GO TO 130
C
C ********************************* STEP 3. INITIALIZE ITERATION SEQUENCE
   30 DO 40 I=1,N
   40 SDF(I)=SF(I)-1.0
      IF (L.GT.0) GO TO 60
      L=0
      DO 50 I=1,N
      A(I,N)=SY(I)
   50 A(I,N-1)=SX(I)
      KOUNL(ID)=KOUNL(ID)+1
      NUMN(ID,2)=NUMN(ID,2)+1
      GO TO 180
C
C ********************************** STEP 4. ESTIMATE JACOBIAN PARTIALS
   60 IF (L.LT.N) GO TO 80
```

```
      DO 70 I=1,N
   70 B(I)=A(I,N)
   80 SX(L)=SX(L)/SF(L)
      DP=SX(L)*SDF(L)
      DO 100 I=1,N
  100 A(I,L)=(SY(I)-A(I,N))/DP
      IF (L.LT.N) GO TO 180
C
C ********************* STEP 5. DETERMINE NEWTON-RAPHSON STEP INCREMENT
C                                COLUMN 1 OF ARRAY D AS RETURNED FROM
C                                FUNCTION ROUTINE ISIMDD IS THE DOT
C                                PRODUCT OF THE INVERSE JACOBIAN WITH
C                                THE DEPENDENT VARIABLE VECTOR
C                                (Y1,Y2...YN)
      DO 110 I=1,N
      DO 110 J=1,N
  110 D(I,J)=A(I,J)
      SCALE=0.
      K=ISIMDD (6,N,1,D,B,SCALE,C)
C
C ********************************** STEP 6. CHECK FOR SINGULAR JACOBIAN
      IF (K.EQ.1) GO TO 150
      WRITE (4,902) K,ID,KOUNL(ID)
  902 FORMAT(1H0,'ERROR IN NLESTN, MATRIX FAILURE USING ','ISIMDD,',
     +           ' ERROR INDICATOR IS',I3,/,
     +       1H ,'FIRST ARGUMENT IN THE CALL STATEMENT IS',I5,
     +           ' LOOP COUNTER IS',I3,
     +           ' THE COLUMN AND SQUARE MATRICES FOLLOW')
      DO 120 I=1,N
  120 WRITE (4,903) B(I),(A(I,J),J=1,N)
  903 FORMAT(1H ,G14.6,5X,6(G14.6))
  130 DO 140 I=1,10
      KOUNL(I)=0
  140 LOOPN(I)=0
      KK=-K
      GO TO 210
C
C ********* STEP 7. INCREMENT INDEPENDENT VARIABLES WITHIN TRUST REGION
  150 FACT=1.0
      DO 160 I=1,N
      TFACT=DABS(D(I,1))/(PCTMAX*DABS(SX(I)))
  160 IF (TFACT.GT.FACT) FACT=TFACT
      DO 170 I=1,N
  170 SX(I)=SX(I)-D(I,1)/FACT
      L=0
      GO TO 190
C
C ****** STEP 8. RESET ARGUMENT LIST INDEPENDENT VARIABLES AND COUNTERS
  180 L=L+1
      SX(L)=SX(L)*SF(L)
  190 KK=L+1
C
      X1=SX(1)
      X2=SX(2)
      X3=SX(3)
      X4=SX(4)
      X5=SX(5)
      X6=SX(6)
C
  200 LOOPN(ID)=L
C
  210 RETURN
      END
```

Appendix 3. Subroutine NLESTN implementation with trust region Broyden Rank One method.

```
C ********************* CODE DESIGNATION BROYDEN **********************
C
      SUBROUTINE NLESTN (ID,N,A,KK,X1,Y1,T1,F1,X2,Y2,T2,F2,X3,Y3,T3,F3,
     + X4,Y4,T4,F4,X5,Y5,T5,F5,X6,Y6,T6,F6)
C
      IMPLICIT REAL*8(A-H,O-Z)
C
C  -NLESTN-  2- TO 6-DIMENSIONAL BROYDEN'S (GOOD) RANK ONE METHOD FOR
C     THE ITERATIVE SOLUTION OF SIMULTANEOUS NONLINEAR ALGEBRAIC
C     EQUATIONS.
C
C     CALL STATEMENT...
C     CALL NLESTN (ID,N,A,KK,X1,Y1,T1,F1,X2,Y2,T2,F2....XN,YN,TN,FN)
C     ...WHERE N IS >=6.
C
C        ID  IS THE ITERATION LOOP NUMBER (MINIMUM VALUE = 1, MAXIMUM
C     VALUE = 10).  EACH NUMBER OF A NEST OF LOOPS MUST HAVE A DIFFERENT
C     LOOP NUMBER, BUT OTHERWISE LOOP NUMBERS ARE ARBITRARY WITHIN THE
C     ABOVE LIMITS.
C        N  IS THE NUMBER OF INDEPENDENT VARIABLES TO BE ITERATED UPON
C     (X1,X2...XN) IN ORDER TO DIMINISH THE DEPENDENT VARAIBLE ERROR
C     VALUES (Y1,Y2...YN) TO WITHIN SPECIFIED TOLERANCES (T1,T2...TN).
C     NOTE- 2 <= N <= 6
C        (F1,F2...FN) ARE INDEPENDENT VARIABLE INCREMENT MULTIPLIERS
C     USED IN THE FINITE DIFFERENCE APPROXIMATION TO THE JACOBIAN
C     MATRIX AT SELECTED STAGES.
C        A  IS AN N-BY-N MATRIX DIMENSIONED IN THE CALLING PROGRAM.
C        KK  IS A FLAG AS FOLLOWS...
C         KK=-3  SINGULAR MATRIX.
C         KK=-1  ALLOWABLE NUMBER OF ITERATIONS HAS BEEN EXCEEDED.
C         KK= 0  ALL Y'S ARE LESS THAN TOLERANCE, ITERATION COMPLETE.
C         KK= 1  ONE OR MORE Y'S ARE GREATER THAN TOLERANCE, REITERATE.
C
C
      INCLUDE (INSAVE)
      INCLUDE (DPSAVE)
C
      DIMENSION NBROY(10),SX(6),SY0(10,6),SY(6),ST(6),SF(6),SDF(6)
      DIMENSION A(N,N),C(36),SK(10,6),SID(6,6)
      DATA LIMIT,PCTMAX,SMNUM/50,0.2,1.0D-8/
C
C ************************ ARRAY INITIALIZE SUBROUTINE INPUT ARGUMENTS
      SX(1)=X1
      SX(2)=X2
      SX(3)=X3
      SX(4)=X4
      SX(5)=X5
      SX(6)=X6
C
      SY(1)=Y1
      SY(2)=Y2
      SY(3)=Y3
      SY(4)=Y4
      SY(5)=Y5
      SY(6)=Y6
```

```
C
      ST(1)=T1
      ST(2)=T2
      ST(3)=T3
      ST(4)=T4
      ST(5)=T5
      ST(6)=T6
C
      SF(1)=F1
      SF(2)=F2
      SF(3)=F3
      SF(4)=F4
      SF(5)=F5
      SF(6)=F6
C
C ***************************************** STEP 1. CHECK FOR CONVERGENCE
      L=LOOPN(ID)
      NUMN(ID,1)=NUMN(ID,1)+1
      ICONV=0
      DO 10 I=1,N
      IF (ICONV.GT.0) GO TO 10
      IF (DABS(SY(I)).GT.DABS(ST(I))) ICONV=1
 10   CONTINUE
      IF (ICONV.GT.0) GO TO 20
      NUMN(ID,3)=NUMN(ID,3)+1
      NUMN(ID,4)=MAX0(NUMN(ID,4),KOUNL(ID))
      KOUNL(ID)=0
      KK=0
      GO TO 200
C
C ******************************** STEP 2. CHECK FOR EXCESSIVE ITERATIONS
 20   IF (KOUNL(ID).LT.LIMIT) GO TO 30
      WRITE (4,901) LIMIT,ID,(SY(I),ST(I),SX(I),I=1,N)
901   FORMAT (
     +  1H ,'ERROR IN NLESTN, NO SOLUTION WITHIN',I3,'ITERATIONS',/,
     +  1H ,'FIRST ARGUMENT IN THE CALL STATEMENT IS',I5,/,
     +  1H ,'ERROR VALUES    TOLERANCES INDEPENDENT VARIABLES',/,
     +      3(G15.6))
      K=1
      GO TO 130
C
C ***************************** STEP 3. DECIDE ON ITERATION PROCEDURE FOR
C                                     CURRENT PHASE
C                                     NBROY(ID)=0    NEWTON-RAPHSON STEP
C                                     NBROY(ID)>0    BROYDEN RANK 1 STEP
 30   IF (KOUNL(ID).LE.0) NBROY(ID)=0
      IF (NBROY(ID).GT.0) GO TO 300
C
C ***************************** STEP 4. INITIALIZE ITERATION SEQUENCE FOR
C                                     MULTIVARIATE NEWTON-RAPHSON STAGE
      DO 40 I=1,N
 40   SDF(I)=SF(I)-1.0
      IF (L.GT.0) GO TO 60
      L=0
      DO 50 I=1,N
```

```
   50  SY0(ID,I)=SY(I)
       KOUNL(ID)=KOUNL(ID)+1
       NUMN(ID,2)=NUMN(ID,2)+1
       GO TO 180
C
C ********************************** STEP 5. ESTIMATE JACOBIAN PARTIALS
   60  SX(L)=SX(L)/SF(L)
       DP=SX(L)*SDF(L)
       DO 100 I=1,N
  100  A(I,L)=(SY(I)-SY0(ID,I))/DP
       IF (L.LT.N) GO TO 180
C
C ***************************** STEP 6. DETERMINE INVERSE JACOBIAN MATRIX
C                                       MATRIX [A] AS RETURNED FROM
C                                       FUNCTION ROUTINE ISIMDD IS THE
C                                       INVERSE JACOBIAN APPROXIMATION
       SCALE=0.
       K=ISIMDD (N,N,-N,A,SID,SCALE,C)
C
C ********************************** STEP 7. CHECK FOR SINGULAR JACOBIAN
       IF (K.EQ.1) GO TO 150
       WRITE (4,902) K,ID,KOUNL(ID)
  902  FORMAT(1H0,'ERROR IN NLESTN, MATRIX FAILURE USING ','ISIMDD,',
      +            ' ERROR INDICATOR IS',I3,/,
      +        1H ,'FIRST ARGUMENT IN THE CALL STATEMENT IS',I5,
      +            ' LOOP COUNTER IS',I3,
      +            ' THE COLUMN AND SQUARE MATRICES FOLLOW')
       DO 120 I=1,N
  120  WRITE (4,903) SY0(ID,I),(A(I,J),J=1,N)
  903  FORMAT(1H ,G14.6,5X,6(G14.6))
  130  DO 140 I=1,10
       KOUNL(I)=0
  140  LOOPN(I)=0
       KK=-K
       GO TO 210
C
C ********* STEP 8. INCREMENT INDEPENDENT VARIABLES WITHIN TRUST REGION
  150  DO 155 I=1,N
       SK(ID,I)=0.0
       DO 155 J=1,N
  155  SK(ID,I)=SK(ID,I)-A(I,J)*SY0(ID,J)
       FACT=1.0
       DO 160 I=1,N
       TFACT=DABS(SK(ID,I)/(PCTMAX*SX(I)))
  160  IF (TFACT.GT.FACT) FACT=TFACT
       DO 170 I=1,N
       SK(ID,I)=SK(ID,I)/FACT
  170  SX(I)=SX(I)+SK(ID,I)
       L=0
       NBROY(ID)=1
       GO TO 190
C
C ************************** STEP 9. PROVIDE BROYDEN RANK ONE UPDATE FOR
C                                    INVERSE JACOBIAN APPROXIMATION
  300  KOUNL(ID)=KOUNL(ID)+1
```

```
      NUMN(ID,2)=NUMN(ID,2)+1
      SBY=0.
      DO 320 I=1,N
      SB=0.
      DO 310 J=1,N
  310 SB=SB+SK(ID,J)*A(J,I)
  320 SBY=SBY+SB*(SY(I)-SY0(ID,I))
      IF (DABS(SBY).LT.SMNUM) GO TO 355
      DO 350 I=1,N
      BY=0.
      DO 330 II=1,N
  330 BY=BY+A(I,II)*(SY(II)-SY0(ID,II))
      DO 350 J=1,N
      SB=0.
      DO 340 JJ=1,N
  340 SB=SB+SK(ID,JJ)*A(JJ,J)
  350 A(I,J)=A(I,J)+(SK(ID,I)-BY)*SB/SBY
  355 DO 360 I=1,N
      SK(ID,I)=0.
      DO 360 J=1,N
  360 SK(ID,I)=SK(ID,I)-A(I,J)*SY(J)
      FACT=1.0
      DO 370 I=1,N
      TFACT=DABS(SK(ID,I)/(PCTMAX*SX(I)))
  370 IF (TFACT.GT.FACT) FACT=TFACT
      DO 380 I=1,N
      SK(ID,I)=SK(ID,I)/FACT
      SX(I)=SX(I)+SK(ID,I)
  380 SY0(ID,I)=SY(I)
      NBROY(ID)=NBROY(ID)+1
      IF (NBROY(ID).GT.N) NBROY(ID)=0
      L=0
      GO TO 190
C
C ***** STEP 10. RESET ARGUMENT LIST INDEPENDENT VARIABLES AND COUNTERS
  180 L=L+1
      SX(L)=SX(L)*SF(L)
  190 KK=L+1
C
      X1=SX(1)
      X2=SX(2)
      X3=SX(3)
      X4=SX(4)
      X5=SX(5)
      X6=SX(6)
C
  200 LOOPN(ID)=L
C
  210 RETURN
      END
```

REFERENCES

1. Burden, R. L. and Faires, J. D., Numerical Analysis. 3rd Edn., PWS, Boston, 1985.

2. Rheinboldt, W. C., Methods for Solving Systems of Nonlinear Equations. SIAM, Philadelphia, 1974.

3. Broyden, C. G., "A Class of Methods for Solving Nonlinear Simultaneous Equations," Math. Comp., Vol. 19, 1965, pp 577-593.

4. Gay, D. M. and Schnabel, R. B., "Solving Systems of Nonlinear Equations by Broyden's Method with Projected Updates," Nonlinear Programming 3, O.L. Mangasarian, R.R. Meyer, and S.M. Robinson, eds., Academic Press, New York, 1978, pp 245-281.

5. Gay, D. M., "Some Convergence Properties of Broyden's Method," SIAM J. Num. Anal., Vol. 16, No. 4, 1979, pp 623-630.

6. Fletcher, R., Practical Methods of Optimization. 2nd Edn., Wiley, New York, 1987.