MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A
FAULT DIAGNOSIS OF
NONLINEAR ANALOG CIRCUITS

VOLUME III
FAULT DIAGNOSIS
IN THE TABLEAU CONTEXT
FINAL REPORT

L. Rapisarda
R. DeCarlo

School of Electrical Engineering
Purdue University
West Lafayette, Indiana 47907

April 1983

This work was supported by Office of Naval Research,
FAULT DIAGNOSIS OF NONLINEAR ANALOG CIRCUITS,
VOLUME III, FAULT DIAGNOSIS IN THE TABLEAU
CONTEXT.

Larry Rapisarda, Ray DeCarlo

PURDUE UNIVERSITY
School of Electrical Engineering
West Lafayette, Indiana 47907

OFFICE OF NAVAL RESEARCH
Arlington, Virginia

This report details a multifrequency tableau approach for the fault
diagnosis of analog linear circuits and systems. The approach utilizes
a modified Newton-Raphson solution algorithm to solve a nonlinear (often
quadratic) set of tableau equations which characterize the parameters to
be identified for the purposes of fault diagnosis. The theory and
algorithm are applied to the fault diagnosis of a video amplifier circuit.
Fault Diagnosis in the Tableau Context
Final Report

L. Rapisarda
R. DeCarlo
School of Electrical Engineering
Purdue University
West Lafayette, Indiana 47907

March 1983

This work was supported by Office of Naval Research
Contract No. N00014-81-K-0323
ACKNOWLEDGMENT

The authors would like to thank the Office of Naval Research for the support of this work under ONR Contract No. N00014-81-0323. We would also like to express our gratitude to those responsible for the operation of the Engineering Computer Network at Purdue. The use of this excellent computing resource was vital to our research.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>List of Tables</td>
<td>iv</td>
</tr>
<tr>
<td>List of Figures</td>
<td>v</td>
</tr>
<tr>
<td>Abstract</td>
<td>vi</td>
</tr>
<tr>
<td>Chapter 1 Completion of &quot;Full Diagnosis&quot; Research</td>
<td>1</td>
</tr>
<tr>
<td>1. Introduction</td>
<td>1</td>
</tr>
<tr>
<td>2. The Component Connection Model and the Fault Diagnosis Equations</td>
<td>1</td>
</tr>
<tr>
<td>3. An Example</td>
<td>4</td>
</tr>
<tr>
<td>4. Summary</td>
<td>13</td>
</tr>
<tr>
<td>Chapter 2 Fault Diagnosis in the Tableau Context with the Assumption of</td>
<td>14</td>
</tr>
<tr>
<td>Limited Simultaneous Faults</td>
<td></td>
</tr>
<tr>
<td>1. Introduction</td>
<td>14</td>
</tr>
<tr>
<td>2. Motivational Examples</td>
<td>16</td>
</tr>
<tr>
<td>3. Introduction to Notation</td>
<td>25</td>
</tr>
<tr>
<td>4. Limited Fault Algorithm</td>
<td>26</td>
</tr>
<tr>
<td>5. Limited Fault Algorithm Examples</td>
<td>31</td>
</tr>
<tr>
<td>6. Conclusions</td>
<td>42</td>
</tr>
<tr>
<td>REFERENCES</td>
<td>43</td>
</tr>
<tr>
<td>APPENDICES: Computer Program Listings</td>
<td>44</td>
</tr>
<tr>
<td>A. Full Diagnosis Program</td>
<td>44</td>
</tr>
<tr>
<td>B. Limited Fault Diagnosis Program</td>
<td>57</td>
</tr>
</tbody>
</table>
# LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Nonzero entries of $L_{11}$ ($26 \times 26$)</td>
<td>4</td>
</tr>
<tr>
<td>2. Nonzero entries of $L_{12}$ ($26 \times 4$)</td>
<td>6</td>
</tr>
<tr>
<td>3. Nonzero entries of $L_{21}$ ($6 \times 26$)</td>
<td>6</td>
</tr>
<tr>
<td>4. Nonzero entries of $L_{21}^R$ ($26 \times 6$)</td>
<td>7</td>
</tr>
<tr>
<td>5. Nonzero entries of $V$ ($26 \times 20$)</td>
<td>8</td>
</tr>
<tr>
<td>6. Nominal values for $a_1$ and $a_2$</td>
<td>10</td>
</tr>
<tr>
<td>7. Nominal values for $a_3$ and $a_4$</td>
<td>11</td>
</tr>
<tr>
<td>8. Parameter values and solution results</td>
<td>13</td>
</tr>
<tr>
<td>9. Summary of Consistency Check (Component 1 Faulty)</td>
<td>20</td>
</tr>
<tr>
<td>10. Summary of Consistency Check (Component 2 Faulty)</td>
<td>21</td>
</tr>
<tr>
<td>11. Summary of Parameter Computation at $s_1 = 1$ (Components 1 and 2 Faulty)</td>
<td>23</td>
</tr>
<tr>
<td>12. Summary of Parameter Computation at $s_1 = 2$ (Components 1 and 2 Faulty)</td>
<td>24</td>
</tr>
<tr>
<td>13. Component information for Example 5.1</td>
<td>34</td>
</tr>
<tr>
<td>14. Nonzero entries of the connection matrices for Example 5.1</td>
<td>34</td>
</tr>
<tr>
<td>15. Nonzero entries of the $L_{21}^R$ and $V$ matrices for Example 5.1</td>
<td>35</td>
</tr>
<tr>
<td>16. Fault list and algorithm performance summary for $s_1 = j10$ and $s_2 = j8$</td>
<td>36</td>
</tr>
<tr>
<td>17. Fault list and algorithm performance summary for $s_1 = j4$ and $s_2 = j2$</td>
<td>38</td>
</tr>
<tr>
<td>18. Nonzero entries of matrices for Example 5.7</td>
<td>39</td>
</tr>
<tr>
<td>19. Nominal parameter values and selected 5-faults</td>
<td>40</td>
</tr>
<tr>
<td>20. Nominal values for $g_1$ and $g_2$</td>
<td>41</td>
</tr>
<tr>
<td>21. Results for example 5.7</td>
<td>41</td>
</tr>
</tbody>
</table>
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Figure 1. Fault Diagnosis Example (26 parameters)</td>
<td>5</td>
</tr>
<tr>
<td>2. Figure 2. Circuit for Motivational Examples</td>
<td>17</td>
</tr>
<tr>
<td>3. Figure 3. Basic Algorithm Structure</td>
<td>29</td>
</tr>
<tr>
<td>4. Figure 4. Detailed Algorithm Structure</td>
<td>32</td>
</tr>
<tr>
<td>5. Figure 5. Fault Diagnosis Example (12 parameters)</td>
<td>33</td>
</tr>
</tbody>
</table>
Abstract

This report finishes the work supported by the Office of Naval Research as cited in the acknowledgement. In particular it indicates the numerical and practical feasibility of the tableau approach to multifrequency fault diagnosis previously introduced in reference [1]. The first chapter presents a 26 component fault diagnosis example based on a linearized model of a video amplifier circuit. The diagnosis of the example circuit depends on the solution of the Tableau Fault Diagnosis Equations. The development of the equations as well as a solution technique are the subject of an earlier report (see reference [1]). Chapter 2 investigates the properties of the tableau fault diagnosis equations under the assumption that the number of simultaneous faults in a system under test is limited. The approach employs multifrequency testing to extract a maximum of information from a given set of test points. The analysis specifically addresses the case in which the number of faulty components exceeds the number of measurement points. The development includes a detailed description of the solution algorithm. Several example problems and solutions conclude the chapter. These include the video amplifier circuit which is used to illustrate the feasibility of the approach. Appendices are included which detail the Fortran code of the computer programs which implement the fault diagnosis techniques for the examples of the report.
Chapter 1
Completion of "Full Diagnosis" Research

1. Introduction

The purpose of this report is the completion of the work presented in [1]. This research concentrated on the "full diagnosis" problem, i.e. the problem of determining system/circuit parameters from multifrequency output measurements under the assumption that all components may be faulty simultaneously. Before discussing new material we summarize the major contributions of the previous report [1].

First is the development of a set of Tableau Fault Diagnosis Equations based on the use of the Component Connection Model (CCM). Due to the nature of the CCM the fault diagnosis equations and the associated Jacobian, utilized in the solution algorithm, have an elegant and sparse structure. Another important characteristic of these equations is that in many cases they are quadratic. The formulation of the diagnosis equations appears in summary in Chapter 2 of this report. The second contribution of [1] is the presentation of a theory of diagnosability for the fault diagnosis equations. This includes a test for diagnosability based on the computation of the rank of the sparse Jacobian. Third is the development of a solution algorithm for the nonlinear Tableau Fault Diagnosis Equations. Of particular interest is a modification which exploits the quadratic nature of the equations and substantially improves the convergence properties of the solution procedure. Finally [1] presents several illustrative examples which include algorithm performance data.

The third section of this chapter completes our work in the application of the fault diagnosis equations to the "full diagnosis" problem with an example which illustrates the feasibility of the Tableau approach for large systems. The second chapter then adapts this work to the assumption that the number of simultaneous faults is limited. Before proceeding to new material we present a review of the CCM and the derivation of the fault diagnosis equations.

2. The Component Connection Model and the Fault Diagnosis Equations.

Let a linear system have N components where the i-th component is characterized by the transfer function $Z_i(s,r_i)$ (s is the Laplace Transform variable and $r_i$ is a parameter which characterizes the component). Denote the i-th component input and output as $a_i(s)$ and $b_i(s)$ respectively. Then
\[ b_i(s) = Z_i(s,r) a_i(s) \]  
(2.1)

is the component input/output equation. Next define the composite component input/output vectors:

\[ a(s) = \text{col} \left( a_1(s), a_2(s), \ldots, a_N(s) \right) \]  
(2.2)

\[ b(s) = \text{col} \left( b_1(s), b_2(s), \ldots, b_N(s) \right) \]  
(2.3)

and the composite component transfer function

\[ Z(s,r) = \text{block diag}(\ldots, Z_i(s,r_i), \ldots) \]  
(2.4)

where \( r = \text{col} \left( r_1, r_2, \ldots, r_N \right) \). The composite component input and output vectors are related by

\[ b(s) = Z(s,r) a(s) \]  
(2.5)

The connection laws (e.g. KVL, KCL) are then expressed in terms of the following equations:

\[ a(s) = L_{11} b(s) + L_{12} u(s) \]  
(2.6)

\[ y(s) = L_{21} b(s) + L_{22} u(s) \]  
(2.7)

where \( u(s) \) and \( y(s) \) are the circuit/system input and output vectors respectively and the \( L_{ij} \) are determined by the connections. Equations 2.5, 2.6 and 2.7 form the CCM equations [5] and have a frequency domain tableau formulation:

\[
\begin{bmatrix}
Z(s,r) - I & a(s) \\
-1 & L_{11}
\end{bmatrix}
\begin{bmatrix}
b(s) \\
0
\end{bmatrix} =
\begin{bmatrix}
-L_{12}u(s)
\end{bmatrix}
\]  
(2.8a)

\[ y(s) = L_{21} b(s) + L_{22} u(s) \]  
(2.8b)

Suppose the circuit/system modeled by equation 2.8 is to be diagnosed; that is the value of \( r \) in \( Z(s,r) \) is sought. Test inputs are applied at \( q \) different test input/frequency combinations and the corresponding outputs are measured. Let \( u(s_i) \) be the test input vector and \( y^W(s_i) \) be the test output vector, for \( i=1,2,\ldots,q \). Because the circuit/system is linear all components of \( u(s_i) \) and \( y^W(s_i) \) are phasors. It is possible to construct a set of fault diagnosis equations of the form [1]:

\[
\begin{bmatrix}
Z(s_i,r) - V \left[ I_{11} \mathbf{v}_a + a_o(s_i) \right] \mathbf{a}_i \\
0
\end{bmatrix} = b_o(s_i)
\]  
(2.9)

for \( i=1,2,\ldots,q \), where

(1) \( q \) is the number of test input/frequency combinations,
(2) \( b_0(s_i) = L_{21}^R y^W(s_i) - L_{22} u(s_i) \),
(3) \( L_{21}^R \) is any right inverse of \( L_{21} \),
(4) \( a_0(s_i) = L_{11} b_0(s_i) + L_{12} u(s_i) \),
(5) \( V \) is a matrix whose columns span the null space of \( L_{21} \),
(6) \( r \) is the unknown parameter vector, and
(7) \( a_i \) is a vector of auxiliary unknowns which characterizes the ambiguity in the solution for \( r \) at any single frequency, \( s_i \).

In particular if

\[
\begin{align*}
    f_i(r) & \triangleq [Z(s_i,r)] - V \\
    g_i(a_i) & \triangleq \left[ L_{11} V a_i + a_0(s_i) \right] \\
    \beta_i & \triangleq b_0(s_i) \\
    x & \triangleq x_{col} [a_1, a_2, \ldots, a_q, r]
\end{align*}
\]

then the fault diagnosis equations have the equivalent form

\[
F(x) = \begin{bmatrix}
    f_1(r)g_1(a_1) - \beta_1 \\
    \vdots \\
    f_q(r)g_q(a_q) - \beta_q
\end{bmatrix} = \Theta
\]

where \( \Theta \) is the zero vector. Using a Newton-Raphson scheme, one iteratively solves for the solution, say \( x^* \), via

\[
J_F(x^k)[x^{k+1} - x^k] = -F(x^k)
\]

where \( x^k \) is the k-th estimate of the solution to equation 2.14 and \( J_F(\cdot) \) is the Jacobian of \( F(\cdot) \). Because of the product structure of equation 2.13 and the sparsity inherent in 2.9, the Jacobian is both elegant and sparse. Specifically

\[
J_F(x) = \begin{bmatrix}
    f_1(r) \frac{\partial g_1(a_1)}{\partial a_1} & 0 & \cdots & \frac{\partial f_1(r) g_1(a_1)}{\partial r} \\
    0 & f_2(r) \frac{\partial g_2(a_2)}{\partial a_2} & \cdots & \frac{\partial f_2(r) g_2(a_2)}{\partial r} \\
    \vdots & \vdots & \ddots & \vdots \\
    0 & 0 & \cdots & f_q(r) \frac{\partial g_q(a_q)}{\partial a_q} \frac{\partial f_q(r) g_q(a_q)}{\partial r}
\end{bmatrix}
\]

(2.16)
3. An Example

To illustrate the use of the fault diagnosis equations developed in [1] consider Figure 1 which is the high-frequency AC equivalent circuit of a four transistor video amplifier. This circuit is based on the example circuit given in Fig. 7 of reference [9]. In addition to the ground node we consider the nodes labeled A through F as accessible since these correspond to inputs, outputs, or power supply connections on the original circuit. These accessible points are used to provide the four inputs and six outputs shown in Figure 1. The nonzero entries of the sparse set of connection matrices, $L_{11}$, $L_{12}$, and $L_{21}$, appear in Tables 1 through 3. All entries of $L_{22}$ are zero.

Table 1.
Nonzero entries of $L_{11}$ (26x26).

<table>
<thead>
<tr>
<th>row, column</th>
<th>value</th>
<th>row, column</th>
<th>value</th>
<th>row, column</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1,2</td>
<td>-1</td>
<td>1,4</td>
<td>-1</td>
<td>1,5</td>
<td>-1</td>
</tr>
<tr>
<td>2,1</td>
<td>1</td>
<td>2,3</td>
<td>-1</td>
<td>3,2</td>
<td>1</td>
</tr>
<tr>
<td>3,4</td>
<td>1</td>
<td>3,6</td>
<td>1</td>
<td>4,1</td>
<td>1</td>
</tr>
<tr>
<td>4,3</td>
<td>-1</td>
<td>5,1</td>
<td>1</td>
<td>5,7</td>
<td>-1</td>
</tr>
<tr>
<td>5,26</td>
<td>-1</td>
<td>6,1</td>
<td>1</td>
<td>6,3</td>
<td>-1</td>
</tr>
<tr>
<td>7,5</td>
<td>1</td>
<td>7,6</td>
<td>1</td>
<td>7,8</td>
<td>-1</td>
</tr>
<tr>
<td>7,10</td>
<td>-1</td>
<td>7,11</td>
<td>-1</td>
<td>8,7</td>
<td>1</td>
</tr>
<tr>
<td>8,9</td>
<td>-1</td>
<td>8,26</td>
<td>1</td>
<td>9,8</td>
<td>1</td>
</tr>
<tr>
<td>9,10</td>
<td>1</td>
<td>9,12</td>
<td>1</td>
<td>9,13</td>
<td>-1</td>
</tr>
<tr>
<td>9,20</td>
<td>-1</td>
<td>10,7</td>
<td>1</td>
<td>10,9</td>
<td>-1</td>
</tr>
<tr>
<td>10,26</td>
<td>1</td>
<td>11,7</td>
<td>1</td>
<td>12,7</td>
<td>1</td>
</tr>
<tr>
<td>12,9</td>
<td>-1</td>
<td>12,26</td>
<td>1</td>
<td>13,9</td>
<td>1</td>
</tr>
<tr>
<td>13,14</td>
<td>-1</td>
<td>13,15</td>
<td>-1</td>
<td>14,13</td>
<td>1</td>
</tr>
<tr>
<td>14,16</td>
<td>-1</td>
<td>14,17</td>
<td>-1</td>
<td>15,13</td>
<td>1</td>
</tr>
<tr>
<td>15,17</td>
<td>-1</td>
<td>15,18</td>
<td>1</td>
<td>16,14</td>
<td>1</td>
</tr>
<tr>
<td>17,14</td>
<td>1</td>
<td>17,15</td>
<td>1</td>
<td>17,19</td>
<td>-1</td>
</tr>
<tr>
<td>17,26</td>
<td>-1</td>
<td>16,14</td>
<td>1</td>
<td>19,17</td>
<td>1</td>
</tr>
<tr>
<td>19,18</td>
<td>-1</td>
<td>20,9</td>
<td>1</td>
<td>20,21</td>
<td>-1</td>
</tr>
<tr>
<td>20,22</td>
<td>-1</td>
<td>21,20</td>
<td>1</td>
<td>21,23</td>
<td>-1</td>
</tr>
<tr>
<td>21,24</td>
<td>-1</td>
<td>22,20</td>
<td>1</td>
<td>22,24</td>
<td>-1</td>
</tr>
<tr>
<td>22,25</td>
<td>1</td>
<td>23,21</td>
<td>1</td>
<td>24,21</td>
<td>1</td>
</tr>
<tr>
<td>24,22</td>
<td>1</td>
<td>24,26</td>
<td>-1</td>
<td>25,21</td>
<td>1</td>
</tr>
<tr>
<td>26,5</td>
<td>1</td>
<td>26,6</td>
<td>-1</td>
<td>26,8</td>
<td>-1</td>
</tr>
<tr>
<td>26,10</td>
<td>-1</td>
<td>26,12</td>
<td>-1</td>
<td>26,17</td>
<td>1</td>
</tr>
<tr>
<td>26,18</td>
<td>-1</td>
<td>26,24</td>
<td>1</td>
<td>26,25</td>
<td>-1</td>
</tr>
</tbody>
</table>
Figure: Fault Diagnosis Example (26 parameters).
Table 2.
Nonzero entries of $L_{12}$ (26x4).

<table>
<thead>
<tr>
<th>row,column</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1,1</td>
<td>1</td>
</tr>
<tr>
<td>9,2</td>
<td>1</td>
</tr>
<tr>
<td>19,3</td>
<td>1</td>
</tr>
<tr>
<td>22,4</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 3.
Nonzero entries of $L_{21}$ (6x26).

<table>
<thead>
<tr>
<th>row,column</th>
<th>value</th>
<th>row,column</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1,1</td>
<td>1</td>
<td>2,2</td>
<td>1</td>
</tr>
<tr>
<td>2,4</td>
<td>1</td>
<td>2,6</td>
<td>1</td>
</tr>
<tr>
<td>3,9</td>
<td>1</td>
<td>4,19</td>
<td>1</td>
</tr>
<tr>
<td>4,26</td>
<td>1</td>
<td>5,22</td>
<td>1</td>
</tr>
<tr>
<td>6,5</td>
<td>1</td>
<td>6,6</td>
<td>-1</td>
</tr>
<tr>
<td>6,8</td>
<td>-1</td>
<td>6,10</td>
<td>-1</td>
</tr>
<tr>
<td>6,12</td>
<td>-1</td>
<td>6,17</td>
<td>1</td>
</tr>
<tr>
<td>6,18</td>
<td>-1</td>
<td>6,24</td>
<td>1</td>
</tr>
</tbody>
</table>

The computation of $L_{21}^R$ and $V$ was performed via the IMSL routines LGINF and LSVDF respectively [8]. These too are spare matrices with their nonzero entries given in Tables 4 and 5.
Table 4.
Nonzero entries of $L^{-R}_{21} (28 \times 6)$.

<table>
<thead>
<tr>
<th>row, column</th>
<th>value</th>
<th>row, column</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1,1</td>
<td>1</td>
<td>2,2</td>
<td>0.346154</td>
</tr>
<tr>
<td>2,6</td>
<td>0.0384615</td>
<td>4,2</td>
<td>0.346154</td>
</tr>
<tr>
<td>4,6</td>
<td>0.0384615</td>
<td>5,2</td>
<td>0.0384615</td>
</tr>
<tr>
<td>5,6</td>
<td>0.115385</td>
<td>6,2</td>
<td>0.307692</td>
</tr>
<tr>
<td>6,6</td>
<td>-0.0769231</td>
<td>8,2</td>
<td>-0.0384615</td>
</tr>
<tr>
<td>8,6</td>
<td>-0.115385</td>
<td>9,3</td>
<td>1</td>
</tr>
<tr>
<td>10,2</td>
<td>-0.0384615</td>
<td>10,6</td>
<td>-0.115385</td>
</tr>
<tr>
<td>12,2</td>
<td>-0.0384615</td>
<td>12,6</td>
<td>-0.115385</td>
</tr>
<tr>
<td>17,2</td>
<td>0.0384615</td>
<td>17,6</td>
<td>0.115385</td>
</tr>
<tr>
<td>18,2</td>
<td>-0.0384615</td>
<td>18,6</td>
<td>-0.115385</td>
</tr>
<tr>
<td>19,4</td>
<td>0.5</td>
<td>22,5</td>
<td>1</td>
</tr>
<tr>
<td>24,2</td>
<td>0.0384615</td>
<td>24,6</td>
<td>0.115385</td>
</tr>
<tr>
<td>25,2</td>
<td>-0.0384615</td>
<td>25,6</td>
<td>-0.115385</td>
</tr>
<tr>
<td>26,4</td>
<td>0.5</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Table 5. Nonzero entries of V (26x20).

<table>
<thead>
<tr>
<th>row, column</th>
<th>value</th>
<th>row, column</th>
<th>value</th>
<th>row, column</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>2,1</td>
<td>0.80808</td>
<td>3,11</td>
<td>1</td>
<td>4,1</td>
<td>-0.42086</td>
</tr>
<tr>
<td>4,2</td>
<td>-0.105202</td>
<td>4,3</td>
<td>-0.105202</td>
<td>4,4</td>
<td>-0.105202</td>
</tr>
<tr>
<td>4,5</td>
<td>0.0743889</td>
<td>4,6</td>
<td>-0.02899</td>
<td>4,7</td>
<td>0.105202</td>
</tr>
<tr>
<td>4,8</td>
<td>-0.105202</td>
<td>4,9</td>
<td>0.109844</td>
<td>5,3</td>
<td>0.109844</td>
</tr>
<tr>
<td>5,1</td>
<td>-0.0475652</td>
<td>5,2</td>
<td>0.109844</td>
<td>5,3</td>
<td>0.109844</td>
</tr>
<tr>
<td>5,4</td>
<td>0.109844</td>
<td>5,5</td>
<td>0.69435</td>
<td>5,6</td>
<td>0.109844</td>
</tr>
<tr>
<td>5,7</td>
<td>-0.109844</td>
<td>5,8</td>
<td>0.109844</td>
<td>5,9</td>
<td>0.132594</td>
</tr>
<tr>
<td>5,10</td>
<td>0.629435</td>
<td>6,1</td>
<td>0.360521</td>
<td>6,2</td>
<td>0.105202</td>
</tr>
<tr>
<td>6,3</td>
<td>0.105202</td>
<td>6,4</td>
<td>0.105202</td>
<td>6,5</td>
<td>-0.0475652</td>
</tr>
<tr>
<td>6,6</td>
<td>0.105202</td>
<td>6,7</td>
<td>-0.105202</td>
<td>6,8</td>
<td>0.105202</td>
</tr>
<tr>
<td>6,9</td>
<td>0.629435</td>
<td>6,10</td>
<td>-0.0475652</td>
<td>7,12</td>
<td>1</td>
</tr>
<tr>
<td>8,1</td>
<td>0.0475652</td>
<td>8,2</td>
<td>0.890156</td>
<td>8,3</td>
<td>-0.109844</td>
</tr>
<tr>
<td>8,4</td>
<td>-0.109844</td>
<td>8,5</td>
<td>0.0776715</td>
<td>8,6</td>
<td>-0.109844</td>
</tr>
<tr>
<td>8,7</td>
<td>0.109844</td>
<td>8,8</td>
<td>-0.109844</td>
<td>8,9</td>
<td>-0.132594</td>
</tr>
<tr>
<td>8,10</td>
<td>0.0776715</td>
<td>9,1</td>
<td>0.109844</td>
<td>9,2</td>
<td>-0.109844</td>
</tr>
<tr>
<td>10,3</td>
<td>-0.109844</td>
<td>10,4</td>
<td>0.0776715</td>
<td>10,5</td>
<td>0.0776715</td>
</tr>
<tr>
<td>10,8</td>
<td>-0.109844</td>
<td>10,7</td>
<td>0.109844</td>
<td>10,8</td>
<td>-0.109844</td>
</tr>
<tr>
<td>10,9</td>
<td>-0.132594</td>
<td>10,10</td>
<td>0.0776715</td>
<td>11,13</td>
<td>1</td>
</tr>
<tr>
<td>12,1</td>
<td>0.0475652</td>
<td>12,2</td>
<td>-0.109844</td>
<td>12,3</td>
<td>-0.109844</td>
</tr>
<tr>
<td>12,4</td>
<td>-0.109844</td>
<td>12,5</td>
<td>0.0776715</td>
<td>12,6</td>
<td>0.890156</td>
</tr>
<tr>
<td>12,7</td>
<td>0.109844</td>
<td>12,8</td>
<td>-0.109844</td>
<td>12,9</td>
<td>-0.132594</td>
</tr>
<tr>
<td>12,10</td>
<td>0.0776715</td>
<td>13,14</td>
<td>1</td>
<td>14,15</td>
<td>1</td>
</tr>
<tr>
<td>15,16</td>
<td>1</td>
<td>16,17</td>
<td>1</td>
<td>17,1</td>
<td>-0.0475652</td>
</tr>
<tr>
<td>17,2</td>
<td>0.336293</td>
<td>17,3</td>
<td>0.336293</td>
<td>17,4</td>
<td>0.336293</td>
</tr>
<tr>
<td>17,5</td>
<td>0.336293</td>
<td>17,6</td>
<td>0.336293</td>
<td>17,7</td>
<td>0.336293</td>
</tr>
<tr>
<td>17,8</td>
<td>0.336293</td>
<td>17,9</td>
<td>-0.301185</td>
<td>17,10</td>
<td>-0.237795</td>
</tr>
<tr>
<td>18,1</td>
<td>0.0475652</td>
<td>18,2</td>
<td>-0.109844</td>
<td>18,3</td>
<td>0.890156</td>
</tr>
<tr>
<td>18,4</td>
<td>-0.109844</td>
<td>18,5</td>
<td>0.0776715</td>
<td>18,6</td>
<td>-0.109844</td>
</tr>
<tr>
<td>18,7</td>
<td>0.109844</td>
<td>18,8</td>
<td>-0.109844</td>
<td>18,9</td>
<td>-0.132594</td>
</tr>
<tr>
<td>18,10</td>
<td>0.0776715</td>
<td>19,5</td>
<td>0.5</td>
<td>19,10</td>
<td>-0.5</td>
</tr>
<tr>
<td>20,18</td>
<td>1</td>
<td>21,19</td>
<td>1</td>
<td>23,20</td>
<td>1</td>
</tr>
<tr>
<td>24,1</td>
<td>-0.0475652</td>
<td>24,2</td>
<td>0.109844</td>
<td>24,3</td>
<td>0.109844</td>
</tr>
<tr>
<td>24,4</td>
<td>0.109844</td>
<td>24,5</td>
<td>-0.0776715</td>
<td>24,6</td>
<td>0.109844</td>
</tr>
<tr>
<td>24,7</td>
<td>0.890156</td>
<td>24,8</td>
<td>0.109844</td>
<td>24,9</td>
<td>0.132594</td>
</tr>
<tr>
<td>24,10</td>
<td>-0.0776715</td>
<td>25,1</td>
<td>0.0475652</td>
<td>25,2</td>
<td>-0.109844</td>
</tr>
<tr>
<td>25,3</td>
<td>-0.109844</td>
<td>25,4</td>
<td>-0.109844</td>
<td>25,5</td>
<td>0.0776715</td>
</tr>
<tr>
<td>25,6</td>
<td>-0.109844</td>
<td>25,7</td>
<td>0.109844</td>
<td>25,8</td>
<td>0.890156</td>
</tr>
<tr>
<td>25,9</td>
<td>-0.132594</td>
<td>25,10</td>
<td>0.0776715</td>
<td>26,5</td>
<td>-0.5</td>
</tr>
</tbody>
</table>

The component transfer functions are: \( Z_i(s,r_i) = r_i s \) for \( i = 4, 5, 10, 11, 16, 17, 23, 24 \) and \( Z_i(s,r_i) = r_i \) for the remaining \( i \).

The nominal component values appear in Table 8 which includes the solution results. Note that the component values are scaled to improve the numerical condition of the problem. The impedance scale factor is \( 10^2 \) and the frequency scale factor is \( 10^7 \).
For this example $M=N=26$ and $p=20$. Using theorem 6.1 the minimum number of input/frequency combinations is $q=3$. For this value of $q$ we found no set of inputs and frequencies for which the circuit would meet the diagnosability condition of Theorem 4.2. Increasing $q$ to 4 we find that the following inputs and frequencies render the circuit diagnosable:

\[
\begin{align*}
    u(s_1) &= u(j.05) = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \\
    u(s_2) &= u(j.2) = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \\
    u(s_3) &= u(j.1) = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \\
    u(s_4) &= u(j.0) = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}
\end{align*}
\]

The nominal values for the $a_i$ to be used along with the nominal parameters as the first solution estimate are in Tables 6 and 7.
Table 6.
Nominal values for $a_1$ and $a_2$.

<table>
<thead>
<tr>
<th></th>
<th>real part</th>
<th>imag. part</th>
<th></th>
<th>real part</th>
<th>imag. part</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a_1$</td>
<td>$-36813E-01$</td>
<td>$-10676E-01$</td>
<td>$a_2$</td>
<td>$-35685E-01$</td>
<td>$-22477E-02$</td>
</tr>
<tr>
<td></td>
<td>$10673E+00$</td>
<td>$11199E-01$</td>
<td></td>
<td>$79972E-01$</td>
<td>$18134E-01$</td>
</tr>
<tr>
<td></td>
<td>$23506E+00$</td>
<td>$-23526E+00$</td>
<td></td>
<td>$15327E+00$</td>
<td>$-11376E+00$</td>
</tr>
<tr>
<td></td>
<td>$25592E+00$</td>
<td>$20359E+00$</td>
<td></td>
<td>$-16606E+00$</td>
<td>$36613E+00$</td>
</tr>
<tr>
<td></td>
<td>$90589E+00$</td>
<td>$-17871E+00$</td>
<td></td>
<td>$-37858E+00$</td>
<td>$-28573E-01$</td>
</tr>
<tr>
<td></td>
<td>$64606E+00$</td>
<td>$-60119E+00$</td>
<td></td>
<td>$-26366E+00$</td>
<td>$25066E+00$</td>
</tr>
<tr>
<td></td>
<td>$31807E+00$</td>
<td>$-88404E-01$</td>
<td></td>
<td>$22878E-01$</td>
<td>$-28580E-01$</td>
</tr>
<tr>
<td></td>
<td>$78337E-01$</td>
<td>$-74524E-01$</td>
<td></td>
<td>$10206E+00$</td>
<td>$-43404E-01$</td>
</tr>
<tr>
<td></td>
<td>$-51649E-01$</td>
<td>$18461E+00$</td>
<td></td>
<td>$54176E-01$</td>
<td>$-39788E-01$</td>
</tr>
<tr>
<td></td>
<td>$-10085E-01$</td>
<td>$64373E+00$</td>
<td></td>
<td>$15076E+00$</td>
<td>$52261E-01$</td>
</tr>
<tr>
<td></td>
<td>$29313E+01$</td>
<td>$-42463E+01$</td>
<td></td>
<td>$83646E+00$</td>
<td>$-55977E-01$</td>
</tr>
<tr>
<td></td>
<td>$12641E+01$</td>
<td>$-19962E+01$</td>
<td></td>
<td>$35621E+00$</td>
<td>$-17659E+00$</td>
</tr>
<tr>
<td></td>
<td>$49954E+00$</td>
<td>$31602E+00$</td>
<td></td>
<td>$17659E+00$</td>
<td>$35621E+00$</td>
</tr>
<tr>
<td></td>
<td>$31785E+00$</td>
<td>$-23551E+00$</td>
<td></td>
<td>$16082E+00$</td>
<td>$-29919E-01$</td>
</tr>
<tr>
<td></td>
<td>$13877E-01$</td>
<td>$-25264E-01$</td>
<td></td>
<td>$60824E-02$</td>
<td>$-94932E-02$</td>
</tr>
<tr>
<td></td>
<td>$39654E+00$</td>
<td>$-44345E+00$</td>
<td></td>
<td>$33091E+00$</td>
<td>$-52113E-01$</td>
</tr>
<tr>
<td></td>
<td>$83160E-01$</td>
<td>$33443E-01$</td>
<td></td>
<td>$94932E-01$</td>
<td>$69834E-01$</td>
</tr>
<tr>
<td></td>
<td>$44210E+00$</td>
<td>$-77673E-01$</td>
<td></td>
<td>$13150E+00$</td>
<td>$-29041E-01$</td>
</tr>
<tr>
<td></td>
<td>$-22943E-02$</td>
<td>$-91810E-02$</td>
<td></td>
<td>$18614E-02$</td>
<td>$-24575E-02$</td>
</tr>
<tr>
<td></td>
<td>$22977E-01$</td>
<td>$-57358E-02$</td>
<td></td>
<td>$24575E-01$</td>
<td>$18614E-01$</td>
</tr>
</tbody>
</table>
Table 7.
Nominal values for $a_3$ and $a_4$.

<table>
<thead>
<tr>
<th>real part</th>
<th>imag. part</th>
<th>real part</th>
<th>imag. part</th>
</tr>
</thead>
<tbody>
<tr>
<td>.21705E-01</td>
<td>.83072E-03</td>
<td>-.37897E-01</td>
<td>-.54919E-03</td>
</tr>
<tr>
<td>-.14510E+00</td>
<td>-.54401E-01</td>
<td>.71877E-01</td>
<td>-.54767E-02</td>
</tr>
<tr>
<td>.25918E+00</td>
<td>-.21989E+00</td>
<td>.77680E-01</td>
<td>-.27986E-01</td>
</tr>
<tr>
<td>-.18065E+00</td>
<td>-.16936E+00</td>
<td>.17237E+00</td>
<td>.68360E-01</td>
</tr>
<tr>
<td>.12023E+01</td>
<td>-.33955E+00</td>
<td>-.52193E+00</td>
<td>-.52442E-03</td>
</tr>
<tr>
<td>-.37126E+00</td>
<td>.20523E-01</td>
<td>.90419E-01</td>
<td>-.25381E-01</td>
</tr>
<tr>
<td>.20697E+00</td>
<td>-.82907E-01</td>
<td>.80488E+00</td>
<td>.26400E+00</td>
</tr>
<tr>
<td>-.17104E+00</td>
<td>-.54506E-01</td>
<td>.31594E-01</td>
<td>.17439E+00</td>
</tr>
<tr>
<td>.11877E+00</td>
<td>.10451E+00</td>
<td>.10015E+00</td>
<td>.72396E-02</td>
</tr>
<tr>
<td>-.99168E+00</td>
<td>.43984E+00</td>
<td>.34497E+00</td>
<td>-.90537E-03</td>
</tr>
<tr>
<td>-.77921E-01</td>
<td>-.93654E-01</td>
<td>.53431E+00</td>
<td>.44778E-01</td>
</tr>
<tr>
<td>.25116E+00</td>
<td>-.79205E-01</td>
<td>-.19705E-01</td>
<td>.30812E-01</td>
</tr>
<tr>
<td>.39836E-01</td>
<td>.12568E+00</td>
<td>-.15406E+00</td>
<td>-.90794E-01</td>
</tr>
<tr>
<td>-.32688E+00</td>
<td>-.42537E-01</td>
<td>.88285E-01</td>
<td>.72016E-02</td>
</tr>
<tr>
<td>.40200E-01</td>
<td>-.16473E-01</td>
<td>.59702E-03</td>
<td>-.22861E-02</td>
</tr>
<tr>
<td>.97777E+00</td>
<td>.69241E-01</td>
<td>.23879E+00</td>
<td>.13878E-01</td>
</tr>
<tr>
<td>.82365E-01</td>
<td>.20100E+00</td>
<td>.11340E+00</td>
<td>.29851E-01</td>
</tr>
<tr>
<td>.65528E-01</td>
<td>.13644E-01</td>
<td>-.22168E-01</td>
<td>.32441E-01</td>
</tr>
<tr>
<td>-.28236E-02</td>
<td>.64917E-04</td>
<td>-.40046E-02</td>
<td>.17978E-01</td>
</tr>
<tr>
<td>-.32495E-03</td>
<td>-.14112E-01</td>
<td>-.89834E+00</td>
<td>-.20048E+00</td>
</tr>
</tbody>
</table>

We next establish a set of "actual" parameter values which are to be determined from the measurement data. These values likewise appear in Table 8. The simulated measurements corresponding to the actual parameters and the inputs of equation 7.1 are:

\[
y^M(j.05) = \begin{pmatrix} .30131E+01 \\ .54763E-01 \\ .13175E+01 \\ .11089E+01 \\ .23035E-01 \\ .36800E+00 \end{pmatrix} + j \begin{pmatrix} -.41295E+01 \\ -.74921E-01 \\ -.10620E+01 \\ .35500E+00 \\ -.77400E+00 \\ .76442E+00 \end{pmatrix} \quad (7.2a)
\]

\[
y^M(j.2) = \begin{pmatrix} .76267E+00 \\ .13862E-01 \\ .79305E+00 \\ .37143E+00 \\ .42976E+00 \\ .66459E+00 \end{pmatrix} + j \begin{pmatrix} -.30754E-01 \\ -.54854E-03 \\ -.14179E+00 \\ -.64531E-02 \\ .40539E-01 \\ .41443E-01 \end{pmatrix} \quad (7.2b)
\]

\[
y^M(j.1) = \begin{pmatrix} -.10196E+00 \\ -.16504E-02 \\ -.10686E+00 \\ .17685E+01 \\ -.28683E+00 \\ -.45287E+00 \end{pmatrix} + j \begin{pmatrix} -.94506E-01 \\ -.17186E-02 \\ -.78825E-01 \\ -.75712E+00 \\ -.11197E+00 \\ -.61275E-02 \end{pmatrix} \quad (7.2c)
\]
Table 8 summarizes the results of the solution algorithms. The first solution used the usual Newton-Raphson iteration step while the second solution used the modified algorithm discussed in Chapter 5 of [1]. Both solutions employed a FORTRAN program compiled and executed on a VAX-11/780 with single precision arithmetic. Use of the usual or modified iteration is a program option. This program is included as Appendix A of this report. Although both solution methods converged for this example the modified algorithm required 7 iterations (17 min.) to converge compared to 17 iterations (36.8 min.) for the Newton-Raphson algorithm. The execute times for the programs are somewhat long due to the fact that sparse matrix techniques were not used. The Jacobian for this example is a 208x186 matrix with 5.4% of its elements nonzero.
### Table 8.
Parameter values and solution results.

<table>
<thead>
<tr>
<th>component</th>
<th>units</th>
<th>nominal</th>
<th>actual</th>
<th>solution 1</th>
<th>solution 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>ohm</td>
<td>12</td>
<td>11</td>
<td>11</td>
<td>11</td>
</tr>
<tr>
<td>2</td>
<td>mho</td>
<td>0.1</td>
<td>0.11</td>
<td>0.109</td>
<td>0.109</td>
</tr>
<tr>
<td>3</td>
<td>ohm</td>
<td>56.7</td>
<td>55</td>
<td>55</td>
<td>55</td>
</tr>
<tr>
<td>4</td>
<td>farad</td>
<td>60</td>
<td>100</td>
<td>100.2</td>
<td>100.2</td>
</tr>
<tr>
<td>5</td>
<td>farad</td>
<td>5</td>
<td>4.5</td>
<td>4.5</td>
<td>4.5</td>
</tr>
<tr>
<td>6</td>
<td>mho</td>
<td>10</td>
<td>9</td>
<td>9.016</td>
<td>9.017</td>
</tr>
<tr>
<td>7</td>
<td>ohm</td>
<td>30</td>
<td>37</td>
<td>37</td>
<td>37</td>
</tr>
<tr>
<td>8</td>
<td>mho</td>
<td>0.1</td>
<td>0.15</td>
<td>0.1498</td>
<td>0.1498</td>
</tr>
<tr>
<td>9</td>
<td>ohm</td>
<td>10</td>
<td>8</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>10</td>
<td>farad</td>
<td>50</td>
<td>65</td>
<td>65</td>
<td>65</td>
</tr>
<tr>
<td>11</td>
<td>farad</td>
<td>5</td>
<td>6.2</td>
<td>6.2</td>
<td>6.2</td>
</tr>
<tr>
<td>12</td>
<td>mho</td>
<td>10</td>
<td>14</td>
<td>14</td>
<td>14</td>
</tr>
<tr>
<td>13</td>
<td>mho</td>
<td>0.3</td>
<td>0.22</td>
<td>0.22</td>
<td>0.22</td>
</tr>
<tr>
<td>14</td>
<td>ohm</td>
<td>10</td>
<td>11</td>
<td>11.03</td>
<td>11.03</td>
</tr>
<tr>
<td>15</td>
<td>ohm</td>
<td>2</td>
<td>2.5</td>
<td>2.5</td>
<td>2.5</td>
</tr>
<tr>
<td>16</td>
<td>farad</td>
<td>50</td>
<td>45</td>
<td>45.04</td>
<td>45.04</td>
</tr>
<tr>
<td>17</td>
<td>farad</td>
<td>5</td>
<td>4.8</td>
<td>4.8</td>
<td>4.8</td>
</tr>
<tr>
<td>18</td>
<td>mho</td>
<td>10</td>
<td>9</td>
<td>9.007</td>
<td>9.007</td>
</tr>
<tr>
<td>19</td>
<td>ohm</td>
<td>10</td>
<td>11</td>
<td>11</td>
<td>11</td>
</tr>
<tr>
<td>20</td>
<td>mho</td>
<td>0.3</td>
<td>0.33</td>
<td>0.33</td>
<td>0.33</td>
</tr>
<tr>
<td>21</td>
<td>ohm</td>
<td>10</td>
<td>11</td>
<td>11.23</td>
<td>11.23</td>
</tr>
<tr>
<td>22</td>
<td>ohm</td>
<td>10</td>
<td>12</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>23</td>
<td>farad</td>
<td>50</td>
<td>49</td>
<td>49</td>
<td>49</td>
</tr>
<tr>
<td>24</td>
<td>farad</td>
<td>5</td>
<td>6</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>25</td>
<td>mho</td>
<td>10</td>
<td>3.6</td>
<td>3.798</td>
<td>3.798</td>
</tr>
<tr>
<td>26</td>
<td>ohm</td>
<td>0.78</td>
<td>0.7</td>
<td>0.7</td>
<td>0.7</td>
</tr>
</tbody>
</table>

### 4. Summary

The preceding example completes the Tableau Fault Diagnosis Equations documented in [1] by demonstrating the feasibility of their use for a large system. These equations are easily computed, requiring no matrix inversions in their construction or in the construction of the associated Jacobian. The polynomial order of the Tableau Fault Diagnosis Equations is dependent on the component characteristics and not the size of the system. For many systems this means that the equations are quadratic, a fact which has been exploited to improve the convergence properties of the solution algorithm. Finally a substantial improvement in the efficiency of the solution algorithm is possible upon the application of sparse matrix techniques.
Chapter 2  
Fault Diagnosis in the Tableau Context  
with the Assumption of Limited Simultaneous Faults

1. Introduction

The discussion of the previous report [1] presumed that ALL parameters of a circuit/system could differ significantly from their nominal values. This assumption is appropriate when there is some form of interdependence among the parameters. An example of this is the parameter set of a transistor model. A transistor functioning abnormally could cause all the parameters of a linearized model to change significantly. However parameter failures in many circuits/systems are often statistically independent. In this case the likelihood of more than a few simultaneous failures is extremely small. The fact that most parameters are at or near their nominal values represents information useful to the solution process and leads to the following objective for this chapter:

Utilize the Tableau fault diagnosis equations developed in [1] to solve for the circuit/system parameter vector, \( \mathbf{r} \), given i) the measurement data and ii) the assumption that at most \( n_f \) of the \( N \) parameters differ from nominal where the integer, \( n_f \), denotes the "number of assumed faults".

Several compelling motivations underly this recasting of the fault diagnosis problem. First it simplifies the calculations since the solution algorithm deals with fewer equations and unknowns. In fact there are recent well documented circumstances [2-4] in which the computations proceed via linear methods. More important than the simplification of the diagnosis equations is the fact that the fault diagnosis process requires fewer test points when the assumption of a limited number of faults is valid. Call the number of test points (outputs) \( n_o \). Each of the \( n_o \) test points is a source of information which in composite should suffice to determine the parameters. Intuitively, the amount of information necessary is proportional to the dimension of the parameter space. Since the limited fault assumption forces the solution to lie in a lower dimension subspace of the parameter space, it follows that its use in the diagnosis process should require fewer test points than a complete diagnosis (i.e. no limited parameter failure assumption). Keep in mind however that a tradeoff arises between the reduction of test points and the ease of solving the resulting equations. For a given value of \( n_f \) as the number of test points, \( n_o \), decreases...
the difficulty of solving the fault diagnosis equations should increase.

Current research in the literature has concentrated on simplifying the fault diagnosis calculations. For example Huang, Lin and Liu [2] have developed a node-fault approach which solves for the change in the branch admittances of a linear network using voltage measurements at a set of "accessible" nodes in the network. If the number of accessible nodes is \( m + 1 \) (including the reference node) and the number of faults is \( k \), this approach requires that \( k \leq m \) for the fault to be uniquely determined using linear techniques. (Note: For this approach the number \( m \) is analogous to the number \( n_e \) in the Tableau approach and \( k \) is analogous to \( n_f \).) For any set of \( k \) faulty branches where \( k < m \), the set of possible values for the network response lies in a \( k \)-dimensional linear subspace of \( R^m \). If a measurement does not lie in any such subspace then \( k \geq m \) and it is not possible to determine the fault uniquely. If on the other hand the measurement does lie in a subspace corresponding to a particular \( k \)-fault then the branch admittances corresponding to that fault are solvable.

Likewise, Saeks et al. [3] have introduced a fault diagnosis algorithm in the limited failure context based on the CCM. The authors recognized the excessive computational cost associated with solving a set of fault diagnosis equations for every possible fault combination and their approach circumvents this problem. Their algorithm, applicable to nonlinear as well as linear networks, has the following structure:

i) Partition the components into two groups. Assume that the group 1 components are "good". Using the measurement data and the characteristics of the group 1 components determine the inputs and outputs of the group 2 components which would give rise to the measurement data.

ii) Test each component in group 2 by determining if the component outputs and inputs are consistent with the nominal component characteristic. If the inputs and outputs of component \( i \) are consistent with its nominal characteristic then component \( i \) is assumed good, if not then no decision is possible.

iii) Repartition the components including all components found to be good in group 1. Go to step ii and continue the process until group 1 consists entirely of good components.

Several remarks concerning this procedure are relevant to the development of the algorithm in this paper. First the decision rule used in [3] to establish the identity of the good components is exact for single component failures but for multiple faults the decision rule must be based on the assumption that the effects of multiple faults cannot cancel. This assumption is reasonable when the "good" components are represented by parameters which are exactly nominal but may break down when actual values of the good parameters are randomly spread around nominal due to production variations. Second, the algorithm in [3] requires that the number of test points for a linear system be sufficient to permit the computation of the inputs and outputs of the components under test with single frequency test data. Third, use of
the multiple fault decision rule requires that the number of faults be strictly less than the number of components under test. (Note: These two requirements are equivalent to the restriction in the Huang, Lin and Liu approach [2] that \( k < m \).) Finally although this method employs the CCM it differs fundamentally from the Tableau Approach presented in this paper.

A final example of the use of the limited fault assumption is that of Biernacki and Bandler who developed an approach to multiple fault location for linear networks. Here the faults are modeled as loads applied to an invariant network which represents the system in its nominal state [4]. Their fault diagnosis equations build on voltage measurements taken at a single test frequency. The identification of the faults depends upon checking the consistency of a set of linear equations. Like the other approaches the linearity of the fault diagnosis equations requires that the number of voltages measured be greater than the number of simultaneous faults to be located. (Note: As for the previous approach this assumption is equivalent to the restriction in the Huang, Lin and Liu approach [2] that \( k < m \) or equivalently in the Tableau context that \( n_f < n_o \).)

All the above approaches share the constraint that \( n_f < n_o \). The algorithm for limited fault analysis developed in this chapter does not require that the number of faults, \( n_f \), be less than the number of measurement points, \( n_o \). Philosophically, the idea is to use the multifrequency techniques developed thus far to "squeeze" as much information as possible out of a given set of test points. The next section presents several simple examples in the context of the Tableau Fault Diagnosis Equations applied to the limited fault assumption. These examples will clearly illustrate how the relationship between \( n_o \) and \( n_f \) affects the approach to the solution.

2. Motivational Examples

The purpose of this section is to illustrate the properties of the Tableau Fault Diagnosis Equations under the assumption that the number of parameters which deviate significantly from nominal is limited. The examples are designed to highlight the differences which occur as the relation between \( n_f \) and \( n_o \) changes and to serve as background to the development of the solution algorithm.

Consider the example shown in Figure 2. The CCM equations for this example are:

\[
\begin{bmatrix}
    b_1(s) \\
    b_2(s) \\
    b_3(s) \\
    b_4(s)
\end{bmatrix} =
\begin{bmatrix}
    r_1 & r_2/s & 0 \\
    0 & r_3 & 0 \\
    0 & r_4 & 0
\end{bmatrix}
\begin{bmatrix}
    a_1(s) \\
    a_2(s) \\
    a_3(s) \\
    a_4(s)
\end{bmatrix}
\]

\[(2.1a)\]

\[
\begin{bmatrix}
    a_1(s) \\
    a_2(s) \\
    a_3(s) \\
    a_4(s)
\end{bmatrix} =
\begin{bmatrix}
    0 & -1 & 0 & 0 \\
    1 & 0 & -1 & 0 \\
    0 & 1 & 0 & 0 \\
    0 & 0 & 0 & 0
\end{bmatrix}
\begin{bmatrix}
    b_1(s) \\
    b_2(s) \\
    b_3(s) \\
    b_4(s)
\end{bmatrix}
+ \begin{bmatrix}
    i_1 \\
    i_2 \\
    i_3 \\
    i_4
\end{bmatrix}
\]

\[(2.1b)\]
Figure 2. Circuit for Motivational Examples.
\[
\begin{bmatrix}
y_1(s) \\
y_2(s)
\end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 
\end{bmatrix} \begin{bmatrix} b_1(s) \\ b_2(s) \\ b_3(s) \\ b_4(s) \end{bmatrix} + [0] u(s)
\] (2.1c)

where

\[
\begin{bmatrix}
r_1 \\
r_2 \\
r_3 \\
r_4
\end{bmatrix} = \begin{bmatrix} G_1 \\
1/C_2 \\
G_3 \\
G_4
\end{bmatrix}
\quad \text{and} \quad \begin{bmatrix} y_1(s) \\ y_2(s) \end{bmatrix} = \begin{bmatrix} I \\ V_0
\end{bmatrix}
\]

The right inverse and null space basis for \( L_{21} \) are:

\[
L_{21}^R = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0
\end{bmatrix}
\quad \text{and} \quad V = \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \\ -1 & 0
\end{bmatrix}
\]

The nominal value for the parameter vector, \( r_o \), is \( r_o = \text{col}[1 1 1 1] \) and \( n_o = 2 \).

**Case 1: \( n_f = 1 \) (Component 1 faulty)**

Assume that only one parameter value differs from nominal and that measurements occur for a single real test frequency, \( s = s_1 \). Recall that the fault diagnosis equation is:

\[
\begin{bmatrix} Z(s_1, r) | -V \end{bmatrix} \begin{bmatrix} L_{11} V_{\alpha_1} + a_0(s_1) \\ \alpha_1 \end{bmatrix} - b_o(s_1) = 0
\] (2.2)

where:

i. \( b_o(s_1) = \begin{bmatrix} L_{11}^R \{ y^M(s_1) \} - L_{22} u(s_1) \end{bmatrix} \);
ii. \( a_0(s_1) = L_{11} b_o(s_1) + L_{12} u(s_1) \);
iii. The columns of \( V \) span \( \text{null} [L_{21}] \);
iv. \( L_{21}^R \) is any right inverse of \( L_{21} \);
v. \( y^M(s_1) \) is the measurement (an \( n_o \) dimensional vector);
vi. The dimension of \( \text{null} [L_{21}] \) is \( p \);
vii. \( \alpha_1 = \text{col}(\alpha_1(s_1), \alpha_2(s_1), \ldots, \alpha_p(s_1)) \).

Next, rearrange the fault diagnosis equation 2.2 to produce:

\[
[y - Z(s_1, r) L_{11} V] \alpha_1 = Z(s_1, r) a_o(s_1) - b_o(s_1)
\] (2.3)

Substituting the information for this example into equation (2.3) yields:

\[
\begin{bmatrix} 1 \\ -r_2/s_1 \\ 0 \\ -1
\end{bmatrix} \begin{bmatrix} \alpha_1(s_1) \\ \alpha_2(s_1) \\ \alpha_3(s_1) \\ \alpha_4(s_1) 
\end{bmatrix} = \begin{bmatrix} r_1 u(s_1) \\ 0 \\ r_4 u(s_1) 
\end{bmatrix} + \begin{bmatrix} -1 \\ r_2/s_1 \\ 0 \\ 0
\end{bmatrix} \begin{bmatrix} y_1^M(s_1) \\ y_2^M(s_1)
\end{bmatrix}
\] (2.4)
Notice the following characteristics of this set of equations:

i. There are 4 equations and 6 unknowns (4 parameters and 2 ambiguity variables);

ii. Each \( r_i \) appears only in the \( i \)'th equation;

iii. If the nominal numerical values for \( r \) replace the variables, the equation set is linear with the \( \alpha_1 \) and \( \alpha_2 \) as unknowns;

iv. If one parameter is faulty then a solution for \( \alpha_1 \) and \( \alpha_2 \) must exist upon deleting the equation containing the nominal value of the faulty component.

The above characteristics suggest a solution method. Namely eliminate the \( i \)'th equation. If the remaining equations are not consistent then \( r_i \) is not faulty (or the single fault assumption is not valid).

To illustrate such a solution method let component 1 be faulty with \( r_1 = 2 \). Given the test frequency \( s_1 = 1 \) and input \( u(s_1) = 1 \) the corresponding test outputs are:

\[
\begin{bmatrix}
  y_1^H(s_1) \\
  y_2^H(s_1)
\end{bmatrix} = \begin{bmatrix} 2 \\ 5 \end{bmatrix} \quad (2.5)
\]

Equation 2.3 becomes:

\[
\begin{bmatrix}
  1 & 0 \\
  -1 & 1 \\
  0 & 1 \\
  -1 & 0
\end{bmatrix} \begin{bmatrix}
  \alpha_1 \\
  \alpha_2
\end{bmatrix} = \begin{bmatrix} -1.5 \\ 1.5 \\ 0.5 \\ 1. \end{bmatrix} \quad (2.6)
\]

where \( \alpha_1 = \alpha_1(1) \) and \( \alpha_2 = \alpha_2(1) \). The results of the successive elimination of each of the rows of equation 2.6 appear in summary form in Table 9.
Table 9.
Summary of Consistency Check (Component 1 Faulty).

<table>
<thead>
<tr>
<th>Row Elim</th>
<th>Resulting Equation</th>
<th>Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$\begin{bmatrix} -1 &amp; 1 \ 0 &amp; 1 \ 0 &amp; 1 \end{bmatrix} \begin{bmatrix} \alpha_1 \ \alpha_2 \end{bmatrix} = \begin{bmatrix} 1.5 \ 0.5 \ 1 \end{bmatrix}$</td>
<td>$\begin{bmatrix} \alpha_1 \ \alpha_2 \end{bmatrix} = \begin{bmatrix} -1 \ 0.5 \end{bmatrix}$</td>
</tr>
<tr>
<td>2</td>
<td>$\begin{bmatrix} 1 &amp; 0 \ 0 &amp; 1 \ -1 &amp; 0 \end{bmatrix} \begin{bmatrix} \alpha_1 \ \alpha_2 \end{bmatrix} = \begin{bmatrix} -1.5 \ 0.5 \ 1 \end{bmatrix}$</td>
<td>Inconsistent</td>
</tr>
<tr>
<td>3</td>
<td>$\begin{bmatrix} 1 &amp; 0 \ -1 &amp; 1 \ -1 &amp; 0 \end{bmatrix} \begin{bmatrix} \alpha_1 \ \alpha_2 \end{bmatrix} = \begin{bmatrix} -1.5 \ 1.5 \ 0.5 \end{bmatrix}$</td>
<td>Inconsistent</td>
</tr>
<tr>
<td>4</td>
<td>$\begin{bmatrix} 1 &amp; 0 \ -1 &amp; 1 \ 0 &amp; 1 \end{bmatrix} \begin{bmatrix} \alpha_1 \ \alpha_2 \end{bmatrix} = \begin{bmatrix} -1.5 \ 1.5 \ 0.5 \end{bmatrix}$</td>
<td>Inconsistent</td>
</tr>
</tbody>
</table>

The data in Table 9 clearly indicates that the only possible single fault for the given measurement data is component 1. To determine the parameter values simply substitute the $\alpha_i$ values into 2.7a and 2.7b to compute actual component inputs and outputs:

$$b(1) = b_0(1) + V \alpha_1$$  \hspace{1cm} (2.7a)

$$a(1) = a_0(1) + L_1 V \alpha_1$$  \hspace{1cm} (2.7b)

Then use the composite component transfer function matrix equation $b(1) = Z(1, r)a(1)$ to compute $r$. For this example equation 2.7 yields:

$$b(1) = \begin{bmatrix} 1 \\ 0.5 \\ 1 \end{bmatrix}$$ and $$a(1) = \begin{bmatrix} 0.5 \\ 0.5 \\ 1 \end{bmatrix}$$

and the parameter values are $r_1 = 2$ and $r_2 = r_3 = r_4 = 1$ as expected.

**Case 2: $n_1=1$ (Component 2 faulty)**

The solution to a limited fault problem is not necessarily unique. To demonstrate this repeat the example with component 2 faulty. Let $r_2 = 2$ with the remainder nominal. The test outputs for this case given $s_1 = 1$ and $u(s_1) = 1$ are:

$$\begin{bmatrix} y^1(s_1) \\ y^2(s_1) \end{bmatrix} = \begin{bmatrix} 1.6 \\ 0.4 \end{bmatrix}$$  \hspace{1cm} (2.8)

Substitute into equation 2.4 to get:
Table 10 contains the results.

Table 10.
Summary of Consistency Check (Component 2 Faulty).

<table>
<thead>
<tr>
<th>Row Elim</th>
<th>Resulting Equation</th>
<th>Solution</th>
</tr>
</thead>
</table>
| 1        | \[
\begin{bmatrix}
-1 & 1 \\
0 & 1 \\
-1 & 0
\end{bmatrix}
\begin{bmatrix}
\alpha_1 \\
\alpha_2
\end{bmatrix} = \begin{bmatrix}
-1 \\
1.2 \\
4 \\
1
\end{bmatrix}
\] | Inconsistent |
| 2        | \[
\begin{bmatrix}
1 & 0 \\
0 & 1 \\
-1 & 0
\end{bmatrix}
\begin{bmatrix}
\alpha_1 \\
\alpha_2
\end{bmatrix} = \begin{bmatrix}
-1 \\
1.2 \\
4 \\
1
\end{bmatrix}
\] | \[
\begin{bmatrix}
\alpha_1 \\
\alpha_2
\end{bmatrix} = \begin{bmatrix}
-1 \\
1
\end{bmatrix}
\] | \[
\begin{bmatrix}
\alpha_1 \\
\alpha_2
\end{bmatrix} = \begin{bmatrix}
-1 \\
1
\end{bmatrix}
\] |
| 3        | \[
\begin{bmatrix}
1 & 0 \\
-1 & 1 \\
-1 & 0
\end{bmatrix}
\begin{bmatrix}
\alpha_1 \\
\alpha_2
\end{bmatrix} = \begin{bmatrix}
-1 \\
1.2 \\
4 \\
1
\end{bmatrix}
\] | \[
\begin{bmatrix}
\alpha_1 \\
\alpha_2
\end{bmatrix} = \begin{bmatrix}
-1 \\
1
\end{bmatrix}
\] |
| 4        | \[
\begin{bmatrix}
1 & 0 \\
-1 & 1 \\
0 & 1
\end{bmatrix}
\begin{bmatrix}
\alpha_1 \\
\alpha_2
\end{bmatrix} = \begin{bmatrix}
-1.5 \\
1.5 \\
.5
\end{bmatrix}
\] | Inconsistent |

The data in Table 10 indicates that there are two possible single faults which satisfy the fault diagnosis equations. The values of α from the second row in the table correspond to parameter values: r₂ = 2 with the remainder nominal. Those of the third row correspond to r₃ = .5 with the remainder nominal. This illustrates the potential for ambiguity in the solutions. It is easy to see for this simple circuit that the ambiguity is generic; that is, it will always occur for this test arrangement whenever parameters two or three are faulty. Although ambiguities are always possible, it would be convenient to be able to avoid such generic ambiguities. A later section will develop a test to determine if such a problem exists for a given diagnosis situation.

To summarize cases 1 and 2 of the example consider the following characteristics: First, the number of faults, nₓ, is strictly less than the number of test points, nₑ. Next, each fault generated a new set of equations from the original fault diagnosis equations. When a solution failed to exists for the set corresponding to some fault then that fault was not possible. The existence of solutions for more than one fault possibility may be generic to the circuit/system test point combination. Finally, notice that the solution process proceeded via linear methods. This is possible because nₓ < nₑ. Here the use of measurements at a single test frequency provides sufficient data for the solution process in that the set of measurements for each fault combination must lie in a linear subspace of the entire measurement space. Although the solution to these
cases is computationally different from the approaches discussed earlier [2-4] the use of linear methods and its underlying justification is an essential commonality which makes them all philosophically equivalent.

Case 3: n_f=2 (Components 1 and 2 faulty)

Consider the possibility that the circuit modeled by equation 2.1 has two simultaneous faults. There are six possible fault combinations, i.e.: \( r_1, r_2 \) faulty; \( r_1, r_3 \) faulty; etc.; all combinations of four parameters taken two at a time. Consider equation 2.4 under the circumstance of two simultaneous faults. As before there are 4 equations and 6 unknowns. Since a fault combination includes two parameters eliminating the equations which contain an assumed fault while setting the remaining parameters to nominal leaves two equations in two unknowns. This means that a solution for \( \alpha_1 \) and \( \alpha_2 \) is likely to exist for each possible fault combination.

Suppose that the actual fault combination for this example is components 1 and 2 (\( r_1 = 2 \) and \( r_2 = 2 \)) and that \( s_1 = 1 \). The test outputs for this case are:

\[
\begin{align*}
\left[ y^N_1(s_1) \right] &= 11.86 \\
\left[ y^N_2(s_1) \right] &= .57
\end{align*}
\]  

As before use the nominal parameter values in equation 2.4 with the understanding that two of the four equations must be incorrect since two parameters are faulty. Equation 2.4 becomes:

\[
\begin{align*}
\begin{bmatrix}
1 & 0 \\
-1 & 1 \\
0 & 1 \\
-1 & 0 \\
\end{bmatrix}
\begin{bmatrix}
\alpha_1 \\
\alpha_2 \\
\end{bmatrix}
&= 
\begin{bmatrix}
1.86 \\
.57
\end{bmatrix}
\]  

Table 11 summarizes the computations of the \( \alpha_i \) and the corresponding \( r \) for each possible fault.
Table 11.
Summary of Parameter Computation at $s_1 = 1$ (Components 1 and 2 Faulty).

<table>
<thead>
<tr>
<th>Rows Elim.</th>
<th>Resulting Equation</th>
<th>Solution</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>1&amp;2</td>
<td>$\begin{bmatrix} 0 &amp; 1 \ -1 &amp; 0 \end{bmatrix} \begin{bmatrix} a_1 \ a_2 \end{bmatrix} = \begin{bmatrix} .57 \ 1 \end{bmatrix}$</td>
<td>$\begin{bmatrix} a_1 \ a_2 \end{bmatrix} = \begin{bmatrix} -1 \ .57 \end{bmatrix}$</td>
<td>$r_1 = 2$ $r_2 = 0$ $r_3 = 1$ $r_4 = 1$</td>
</tr>
<tr>
<td>1&amp;3</td>
<td>$\begin{bmatrix} -1 &amp; 1 \ -1 &amp; 0 \end{bmatrix} \begin{bmatrix} a_1 \ a_2 \end{bmatrix} = \begin{bmatrix} 1.29 \ 1 \end{bmatrix}$</td>
<td>$\begin{bmatrix} a_1 \ a_2 \end{bmatrix} = \begin{bmatrix} -1 \ .29 \end{bmatrix}$</td>
<td>$r_1 = 2$ $r_2 = .5$ $r_3 = 1$ $r_4 = 1$</td>
</tr>
<tr>
<td>1&amp;4</td>
<td>$\begin{bmatrix} -1 &amp; 1 \ 0 &amp; 1 \end{bmatrix} \begin{bmatrix} a_1 \ a_2 \end{bmatrix} = \begin{bmatrix} 1.29 \ 1 \end{bmatrix}$</td>
<td>$\begin{bmatrix} a_1 \ a_2 \end{bmatrix} = \begin{bmatrix} -1.72 \ .57 \end{bmatrix}$</td>
<td>$r_1 = 2.7$ $r_2 = 1$ $r_3 = 1$ $r_4 = .72$</td>
</tr>
<tr>
<td>2&amp;3</td>
<td>$\begin{bmatrix} 1 &amp; 0 \ -1 &amp; 0 \end{bmatrix} \begin{bmatrix} a_1 \ a_2 \end{bmatrix} = \begin{bmatrix} -1.43 \ 1 \end{bmatrix}$</td>
<td>Inconsistent</td>
<td>Inconsistent</td>
</tr>
<tr>
<td>2&amp;4</td>
<td>$\begin{bmatrix} 1 &amp; 0 \ 0 &amp; 1 \end{bmatrix} \begin{bmatrix} a_1 \ a_2 \end{bmatrix} = \begin{bmatrix} -1.43 \ .57 \end{bmatrix}$</td>
<td>$\begin{bmatrix} a_1 \ a_2 \end{bmatrix} = \begin{bmatrix} -1.43 \ .57 \end{bmatrix}$</td>
<td>$r_1 = 1$ $r_2 = -4$ $r_3 = 1$ $r_4 = 1.43$</td>
</tr>
<tr>
<td>3&amp;4</td>
<td>$\begin{bmatrix} 1 &amp; 0 \ -1 &amp; 1 \end{bmatrix} \begin{bmatrix} a_1 \ a_2 \end{bmatrix} = \begin{bmatrix} -1.43 \ 1.29 \end{bmatrix}$</td>
<td>$\begin{bmatrix} a_1 \ a_2 \end{bmatrix} = \begin{bmatrix} -1.43 \ -.14 \end{bmatrix}$</td>
<td>$r_1 = 1$ $r_2 = 1$ $r_3 = -.25$ $r_4 = 1.43$</td>
</tr>
</tbody>
</table>

As expected the computation produces inconclusive results. Clearly the resolution of the ambiguity in the data of Table 11 requires additional information. To acquire the needed information repeat the same procedure using a second test frequency (the verification step). Let $s_2 = 2$ for which measurement at the output would yield:

$$\begin{bmatrix} y^N(s_2) \\ y^M(s_2) \end{bmatrix} = \begin{bmatrix} 2 \\ 5 \end{bmatrix}$$ (2.12)

As before substitute this information into equation 2.4 to obtain:

$$\begin{bmatrix} 1 & 0 \\ .5 & .5 \\ 0 & 1 \\ -1 & 0 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} -1 & -1 \\ .5 & -1 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 2 \\ 5 \end{bmatrix} = \begin{bmatrix} -1.5 \\ .5 \\ .5 \\ 1 \end{bmatrix}$$ (2.13)

Table 12 contains the summary of results for the computation at this test frequency.
Table 12.
Summary of Parameter Computation at $s_1 = 2$ (Components 1 and 2 Faulty).

<table>
<thead>
<tr>
<th>Rows Elim.</th>
<th>Resulting Equation</th>
<th>Solution</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>1&amp;2</td>
<td>$\begin{bmatrix} 0 &amp; 1 \ -1 &amp; 0 \end{bmatrix} \begin{bmatrix} a_1 \ a_2 \end{bmatrix} = \begin{bmatrix} .5 \ 1 \end{bmatrix}$</td>
<td>$\begin{bmatrix} a_1 \ a_2 \end{bmatrix} = \begin{bmatrix} -1 \ .5 \end{bmatrix}$</td>
<td>$r_1 = 2$, $r_3 = 1$; $r_2 = 2$, $r_4 = 1$</td>
</tr>
<tr>
<td>1&amp;3</td>
<td>$\begin{bmatrix} -.5 &amp; .5 \ -1 &amp; 0 \end{bmatrix} \begin{bmatrix} a_1 \ a_2 \end{bmatrix} = \begin{bmatrix} .5 \ 1 \end{bmatrix}$</td>
<td>$\begin{bmatrix} a_1 \ a_2 \end{bmatrix} = \begin{bmatrix} -1 \ 0 \end{bmatrix}$</td>
<td>$r_1 = 2$, $r_3 = 0$; $r_2 = 1$, $r_4 = 1$</td>
</tr>
<tr>
<td>1&amp;4</td>
<td>$\begin{bmatrix} -.5 &amp; .5 \ 0 &amp; 1 \end{bmatrix} \begin{bmatrix} a_1 \ a_2 \end{bmatrix} = \begin{bmatrix} .5 \ 1 \end{bmatrix}$</td>
<td>$\begin{bmatrix} a_1 \ a_2 \end{bmatrix} = \begin{bmatrix} .5 \ 0 \end{bmatrix}$</td>
<td>$r_1 = 3$, $r_3 = 1$; $r_2 = 1$, $r_4 = .5$</td>
</tr>
<tr>
<td>2&amp;3</td>
<td>$\begin{bmatrix} 1 &amp; 0 \ -1 &amp; 0 \end{bmatrix} \begin{bmatrix} a_1 \ a_2 \end{bmatrix} = \begin{bmatrix} -1.5 \ 1 \end{bmatrix}$</td>
<td>Inconsistent</td>
<td>Inconsistent</td>
</tr>
<tr>
<td>2&amp;4</td>
<td>$\begin{bmatrix} 1 &amp; 0 \ 0 &amp; 1 \end{bmatrix} \begin{bmatrix} a_1 \ a_2 \end{bmatrix} = \begin{bmatrix} -1.5 \ .5 \end{bmatrix}$</td>
<td>$\begin{bmatrix} a_1 \ a_2 \end{bmatrix} = \begin{bmatrix} .5 \ -1.5 \end{bmatrix}$</td>
<td>$r_1 = 1$, $r_3 = 1$; $r_2 = \infty$, $r_4 = 1.5$</td>
</tr>
<tr>
<td>3&amp;4</td>
<td>$\begin{bmatrix} 1 &amp; 0 \ -.5 &amp; .5 \end{bmatrix} \begin{bmatrix} a_1 \ a_2 \end{bmatrix} = \begin{bmatrix} -1.5 \ 1.5 \end{bmatrix}$</td>
<td>$\begin{bmatrix} a_1 \ a_2 \end{bmatrix} = \begin{bmatrix} .5 \ -1.5 \end{bmatrix}$</td>
<td>$r_1 = 1$, $r_3 = -1$; $r_2 = 1$, $r_4 = 1.5$</td>
</tr>
</tbody>
</table>

The final step in this two-fault case is the comparison of the Tables 11 and 12. The only common solution between the two tables is that for the component 1 and 2 fault combination. This fault set therefore is the only one which simultaneously satisfies the equations derived from measurements at both test frequencies.

This case for which the number of faults, $n_f$, equals the number of test points, $n_o$, prompts the following observations: In general a single real test frequency is insufficient to resolve the ambiguity. Data from the measurement at a second test frequency is necessary. Since the equations resulting from the different test frequencies are solved separately, linear techniques are still possible. The only significant difference between this case and the previous two lies in the necessity for the verification step (the second test frequency). This step was not necessary in cases 1 and 2 since the equations used to solve for the $\alpha$'s were overspecified due to the fact that $n_f < n_o$.

Case 4: $n_f = 2$, $n_o = 1$

Finally consider the same example (Figure 2) with one test point instead of two. The CCM equations are the same as equation 2.1 except that the output equation is:

$$y_1(s) = [1 \ 0 \ 0 \ 1] \begin{bmatrix} b_1(s) \\ b_2(s) \\ b_3(s) \\ b_4(s) \end{bmatrix} + [0 \ 0 \ 0 \ 0] u(s)$$

(2.14)

Accordingly the right inverse and null space basis of $L_{21}$ change to:
The fault diagnosis equations at a single test frequency $s_i$ for $u(s_i) = 1$ are:

$$
\begin{bmatrix}
  r_1 & 0 & 0 & 0 & -1 & 0 & 0 \\
  0 & r_2/s_i & 0 & 0 & 0 & -1 & 0 \\
  0 & 0 & r_3 & 0 & 0 & 0 & -1 \\
  0 & 0 & 0 & r_4 & 1 & 0 & 0
\end{bmatrix}
\begin{bmatrix}
  1 - a_2(s_i) \\
  a_2(s_i) \\
  a_1(s_i) \\
  a_2(s_i) \\
  a_3(s_i)
\end{bmatrix}
\begin{bmatrix}
y_1(s_i) + a_1(s_i) - a_2(s_i) \\
a_2(s_i) \\
a_1(s_i) \\
a_2(s_i) \\
a_3(s_i)
\end{bmatrix}
- \begin{bmatrix}
y_1(s_i) \\
0 \\
0 \\
0 \\
0
\end{bmatrix}
= \Theta
$$

For the previous cases $n_o \geq n_f$. To illustrate the properties of the fault diagnosis equations for $n_f > n_o$ let the number of allowable simultaneous faults be two ($n_f = 2$) and notice that $n_o = 1$. Under these circumstances observe that for equation 2.15, the elimination of two equations corresponding to an assumed fault combination with the remaining two parameters set to nominal leaves two equations with three unknowns. This means that the ambiguity must be resolved by the use of several test frequencies. The resulting equations must be solved simultaneously with the unfortunate consequence that the solution method cannot employ linear techniques.

This case represents a more general limited fault problem than the previous cases since the number of test outputs, $n_o$, may be smaller than the number of assumed faults, $n_f$. Since this more general approach is the subject of the remainder of this chapter the actual solution process will appear in a later section.

Current research [2-4] has concentrated on the fault diagnosis problem with the restriction that $n_o \leq n_f$. A motivation for this is that the solution method may use linear techniques as illustrated by all but the last case. Unfortunately the restriction to linear methods fails to exploit all of the information available at the test points. Thus problems such as case 4 are not solvable via such methods. The remainder of this chapter considers the development of a more general algorithm for limited fault analysis in the CCM context. Here the restriction $n_o \leq n_f$ will not apply. The method will exploit all the information available at a given set of test points by utilizing multifrequency testing.

3. Introduction to Notation

The purpose of this brief section is to develop the notation to assist in describing the $n_f$-fault solution algorithm presented in the next section. To this purpose we define the following:

$$<i_1, i_2, \ldots, i_{n_f}>$$

The fault index

It is an ordered $n_f$-tuple of positive integers subject to the following:

$$<i_1, i_2, \ldots, i_{n_f} \in N>$$

(3.1)
\[ \Omega(N,n_f) = \{ \text{The set of all possible } n_f \text{-tuples satisfying 3.1} \} \]

There are \( C_{N,n_f} \) elements in \( \Omega(N,n_f) \). (Note: \( C_{N,n_f} \) denotes the number of combinations of \( N \) things taken \( n_f \) at a time.)

If \( \gamma = <i_1, i_2, \ldots, i_{n_f}> \) then let \( M_{\gamma} \) be the following matrix with \( n_f \) columns:
\[
M_{\gamma} = \left[ e_{i_1} | e_{i_2} | \ldots | e_{i_{n_f}} \right]
\]  

where \( e_i \) is the \( N \)-dimensional unit vector with a 1 in the \( i \)-th position and 0's elsewhere. For the fault \( \gamma \in \Omega(N,n_f) \) define the fault space \( P_{\gamma} \) in the following way:
\[
P_{\gamma} = \{ r \mid r = r_0 + M_{\gamma} \rho, \rho \in \mathbb{R}^{n_f} \}
\]  

We say that the fault index \( \gamma \) "represents" the parameter \( r \) if \( r \in P_{\gamma} \). In other words \( \gamma \) is a specific \( n_f \)-fault combination.

Note: Limiting the number of allowable faults to \( n_f \) means that there are at most \( n_f \) faults since any fault space with less than \( n_f \) faults is contained in some \( P_{\gamma} \) as defined in 3.3.

Finally let
\[
F_{\gamma}(x_\gamma) \triangleq F(x) \bigg|_{r_i=r_{oi} \quad \forall \gamma}
\]

where:

(i) \( F(x) \) is as defined in equation 2.14 of Chapter 1.
(ii) \( x = \text{col}(a_1,a_2,\ldots,a_q,r) \)
(iii) \( x_\gamma = \text{col}(a_1,a_2,\ldots,a_q,r_{\gamma 1},\ldots,r_{\gamma q}) \)
(iv) \( \gamma_i \) denotes the \( i \)-th element of the fault index \( \gamma \)
(v) \( r_{oi} \) is the nominal value of the \( i \)-th parameter.

If a system fault is known to lie in \( P_{\gamma} \) then the actual values of the faulty parameters must satisfy \( F_{\gamma}(x_\gamma) = 0 \). Suppose a system is \( n_f \)-fault diagnosable, then there will generally be one fault index, \( \gamma \), for which \( F_{\gamma}(x_\gamma) = 0 \).

4. Limited Fault Algorithm

This section has the following objective:

Given a circuit/system which is \( n_f \)-fault diagnosable and which has a fault characterized by a parameter vector, \( r \in P_{\gamma}, \gamma \in \Omega(N,n_f) \), develop a solution algorithm which will:

i) determine which circuit/system parameters are faulty (find \( \gamma \)); and
ii) solve for the values of the faulty parameters, i.e. find \( r \in P_{\gamma}, \gamma \in \Omega(N,n_f) \).

The information available to accomplish this objective is the fact that the parameter vector, \( r \), and the associated ambiguity variables, \( a_i, i = 1,2,\ldots,q \), which combine to form the composite unknown vector, \( x \), must satisfy \( F(x) = 0 \) AND the constraint that \( r \in P_{\gamma} \) where \( P_{\gamma} = \gamma \in \Omega(N,n_f) P_{\gamma} \). The latter constraint makes it impossible to utilize the Newton-Raphson iteration directly. To illustrate the nature of the problem imposed by this constraint consider the following "brute force" approach as a solution method:
Try to solve for \( x_\gamma \) in \( F_\gamma(x_\gamma) = 0 \) for each \( \gamma \in \Omega(N,n_f) \). If for a particular \( \gamma \), a solution fails to exist, \( \gamma \) denotes an incorrect fault combination. If a solution does exist then \( \gamma \) is a potential fault combination. If only one \( \gamma \) yields a solution, the fault is uniquely identified.

The modified Newton-Raphson algorithm developed in [1] is a stable method for testing each \( F_\gamma \). The number of such equations which must be tested is \( C_{N,n_f} \), where

\[
C_{N,n_f} = \frac{n!}{m!(n-m)!} \tag{4.1}
\]

The major shortcoming of such an approach is that as \( N \) gets large with \( n_f > 2 \), the number of possible faults becomes extremely large resulting in excessively long computation time.

The excessive number of fault possibilities for large systems is the main reason for considering the alternative approach described in this section. For clarity we begin with a brief overview of the idea. Recall that the available information for a solution algorithm to determine the fault index, \( \gamma \), and solve for the parameter vector, \( r \), is:

\[
F(x) = 0 \tag{4.2}
\]

and

\[
 r \in P = \bigcup_{\gamma \in \Omega(N,n_f)} P_\gamma \tag{4.3}
\]

Suppose we view equation 4.2 as a set of constraints and reformulate the restriction in 4.3 in the following way:

Define:

\[
\varphi(r,\gamma) \triangleq \| r - r_\gamma \|_2 \tag{4.4}
\]

where (i) \( r \in \mathbb{R}^n \), (ii) \( r_\gamma \in \mathbb{P}_\gamma \), (iii) \( \gamma \in \Omega(N,n_f) \), and (iv) \( \| \cdot \|_2 \) denotes the Euclidean norm. Notice that since \( \varphi \geq 0 \), equation 4.3 is satisfied if we require that \( \varphi(r,\gamma) \) be a minimum. In other words \( \varphi \), the objective function to be minimized, is the distance from the solution, \( r \), to a point in some \( P_\gamma \). This distance is zero for any solution that can be described by a fault index, \( \gamma \in \Omega(N,n_f) \). Thus the problem statement becomes:

\[
\text{minimize } \varphi(r,\gamma) \tag{4.5}
\]

subject to:

\[
F(x) = 0 \tag{4.6}
\]

To develop some insight into the solution process to be developed consider first a solution method for the general nonlinear constrained minimization problem given by Luenberger[6] and known as the "Gradient Projection Method". This method proceeds in the following manner:
(i) Given a starting point use some solution procedure to find a "feasible point" that is a point which satisfies the equality constraints.

(ii) Next update the feasible point by adding to it a component which lies in the space tangent (at the feasible point) to the curve defined by the equality constraints. The direction of the tangent component is the projection of the gradient of the objective function into the tangent space (hence the Gradient Projection Method). The updated point is the value along the tangent projection for which the objective function is minimum.

(iii) Using the updated point as the starting point repeat these two steps until the procedure converges to a solution point.

To apply this type of procedure to the solution of 4.5 and 4.6, modify the second step of the Gradient Projection Method in the following manner:

(i) Let \( \Gamma \subseteq \Omega(N, n_f) \). For each \( \gamma \in \Gamma \) compute the tangent space component for which \( \varphi(r, \gamma) \) is a minimum.

Note: Although this requires a search as in the direct approach the time required can be kept relatively small if the number of elements in \( \Gamma \) is small.

(ii) To choose the elements of \( \Gamma \) consider the parameter values at the feasible point. Any parameters which are close to nominal (say within 10%) consider good. This will eliminate some fault combinations from consideration thus decreasing the number of elements of \( \Gamma \) relative to the number of elements of \( \Omega(N, n_f) \).

(iii) Another criterion which will limit the size of \( \Gamma \) is to consider the component whose parameter value at the feasible point is farthest from nominal as faulty.

We now present the details of the algorithm to solve the limited fault problem. This algorithm has the structure shown in Figure 3. The first step of the algorithm is to choose an initial guess. The best available information about the solution is the nominal values for the circuit/system under test. Therefore let \( x^0 = x_0 \).

The next step is to find a feasible solution, that with \( x_0 \) as an initial guess find a point, \( x \in \mathbb{R}^{n_f + N} \), which satisfies \( F(x) = 0 \). Recall that the modified Newton-Raphson iteration step is[1]:

\[
\begin{align*}
J_F(x^k)d^k &= -F(x^k) \\
x^{k+1} &= x^k + \lambda d^k
\end{align*}
\]  

where \( \lambda \) is chosen to satisfy

\[
|F(x^k + \lambda d^k)|_2 < |F(x^k)|_2
\]

\( J_F(x^k) \) can be expected to have a nontrivial null space. If it did not then there is
Select an initial estimate of the solution

Use an estimate of the solution to find a "feasible solution"

Update the feasible solution with a Tangent space correction

CONVERGED?

No

Yes

STOP

Step 1

Step 2

Step 3

Figure 3. Basic Algorithm Structure
sufficient information to find the solution without the limited fault assumption. This means that 4.7 does not have a unique solution. To make the solution to 4.7 unique requires that \( d^k \) be the "least squares" solution. This "least squares" solution can theoretically be found via Moore-Penrose pseudo-inverse\[7\] but in practice the solution is determined using software such as IMSL routine LLCQF\[8\]. Implemented in this manner the modified Newton-Raphson iteration provides a means of finding a feasible point.

The third step depicted in Figure 3 is the tangent space update of the feasible point to minimize the objective function. To assist in the explanation of the implementation of this step define the following:

- \( x_{fe} \) the feasible solution resulting from Step 2.
- \( r_{fe} \) the parameter part of \( x_{fe} \) (i.e. the last \( N \) entries of \( x_{fe} \)).
- \( V_F(x_c) \) the matrix whose columns span \( \text{Null}[J_F(x_{fe})] \).
- \( V_r(x_{fe}) \) the matrix consisting of the last \( N \) rows of \( V_F(x_{fe}) \).
- \( T_F(x_{fe}) \) the tangent space of the surface \( F(x) = \theta \) at the point \( x_{fe} \).
- \( x_i \) a point in \( T_F(x_{fe}) \).
- \( r_i \) the last \( N \) entries of \( r_i \).
- \( D_y \) for \( \gamma = <i_1, i_2, \ldots, i_{\gamma}> \) this is the identity matrix with rows \( i_1, i_2 \) etc. deleted.

Note: \( D_y \) is a matrix operator for the parameter vector which produces a vector consisting of the entries of \( r \) whose indices are not contained in \( \gamma \).

The objective of this step is to find a point in the tangent space which minimizes the function \( \phi \). If \( x_i \) is a point in \( T_F(x_{fe}) \) it has the following characterization:

\[ x_i = x_{fe} + V_F(x_{fe}) \beta \]  \hspace{1cm} (4.8)

where \( \beta \in \mathbb{R}^M \) and \( M = \text{dim} \{ \text{Null}[J_F(x_{fe})] \} \). Thus the goal for this step becomes: Find \( \beta \) which minimizes \( \phi \). Since the objective function, \( \phi \), involves the parameter alone only the last \( N \) equations of 4.8 require consideration. Thus

\[ r_i = r_{fe} + V_r(x_{fe}) \beta \]  \hspace{1cm} (4.9)

characterizes the parameter part of \( x_i \) used in the objective function. Let \( \gamma \) be fixed. Note that:

\[ \min_{x_i \in T_F(x_{fe})} \phi(r_i, \gamma) = \min_{x_i \in T_F(x_{fe})} \| r_i - r_{\gamma} \|_2 \]  \hspace{1cm} (4.10)

\[ = \min_{\beta \in \mathbb{R}^M} \| D_y(r_{fe} + V_r(x_{fe}) \beta - r_{\gamma}) \|_2 \]

But since \( D_y r_{\gamma} = D_y r_{\gamma} \) this become:

\[ \min_{x_i \in T_F(x_{fe})} \phi(r_i, \gamma) = \min_{\beta \in \mathbb{R}^M} \| A_{\gamma} \beta - B_{\gamma} \|_2 \]  \hspace{1cm} (4.11)

where

\[ A_{\gamma} = D_y V_r(x_{fe}) \quad \text{and} \quad B_{\gamma} = D_y (r_{\gamma} - r_{fe}) \]  \hspace{1cm} (4.12)

For a given \( x_{fe} \) and \( \gamma \) the matrix \( A_{\gamma} \) and the vector \( B_{\gamma} \) are known. Thus equation 4.11 is:
a linear least squares problem readily solved via techniques discussed previously[7,8].

Let \( \beta_\gamma \) denote the vector which satisfies 4.11. To find the minimum of the objective function for all \( r_i \) and \( \gamma \), compute \( \beta_\gamma \) for each \( \gamma \) and select the one which satisfies:

\[
\min_{\gamma \in \Omega(N,n_0)} |A_\gamma \beta_\gamma - B_\gamma|_2
\]  

(4.13)

As discussed earlier performing this computation for every possible \( \gamma \) is unnecessary. To eliminate some fault indices from consideration construct a set \( \Gamma \subset \Omega(N,n_0) \) of candidate faults based on the following criterion:

(i) If the \( i \)'th entry of \( r_f \) is sufficiently close to the nominal value for the \( i \)'th component (e.g. within 10%) then eliminate from consideration all faults containing component \( i \).

(ii) If the \( j \)'th entry of \( r_f \) has the greatest deviation from nominal then eliminate from consideration all faults which do not contain component \( j \).

Instead of the minimization in 4.13 perform:

\[
\min_{\gamma \in \Gamma} |A_\gamma \beta_\gamma - B_\gamma|_2
\]  

(4.14)

The complete algorithm has the structure shown in Figure 4.

5. Limited Fault Algorithm Examples

The purpose of this section is to present two example problems which will illustrate the theory and limited fault algorithm developed in this chapter. The first example is based on the 12 parameter circuit of Figure 5. The solution algorithm is employed to analyze 16 randomly selected faults within this circuit. The second example is the 26 parameter circuit of Figure 1. Practical implementation of the fault diagnosis procedure for this circuit requires the use of sparse matrix techniques. Although the development of sparse matrix algorithms is beyond the scope of this report the example is included to demonstrate the feasibility of the algorithm for larger systems.

Example 5.1: Consider the circuit of Figure 5. The circuit input and outputs are:

\[
u_1 = V_1 \quad y_1 = Ic_1 \quad y_2 = V_o
\]  

(5.2)

The parameter definitions, nominal values and component transfer functions appear in Table 13.
Figure 4. Detailed Algorithm Structure
Figure 5. Fault Diagnosis Example (12 parameters).
Table 13.
Component information for Example 5.1

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Nominal Value</th>
<th>Definition</th>
<th>$Z(s,r_i)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>r_1</td>
<td>.1</td>
<td>$1/C_1$</td>
<td>$r_1/s$</td>
</tr>
<tr>
<td>r_2</td>
<td>.5</td>
<td>$R_x$</td>
<td>$r_2$</td>
</tr>
<tr>
<td>r_3</td>
<td>1.</td>
<td>$R_m$</td>
<td>$r_3$</td>
</tr>
<tr>
<td>r_4</td>
<td>1.</td>
<td>$1/C_m$</td>
<td>$r_4/s$</td>
</tr>
<tr>
<td>r_5</td>
<td>.1</td>
<td>$1/C_2$</td>
<td>$r_5/s$</td>
</tr>
<tr>
<td>r_6</td>
<td>1.</td>
<td>$1/R_g$</td>
<td>$r_6$</td>
</tr>
<tr>
<td>r_7</td>
<td>1.</td>
<td>$1/R_e$</td>
<td>$r_7$</td>
</tr>
<tr>
<td>r_8</td>
<td>1.</td>
<td>$C_e$</td>
<td>$r_8$</td>
</tr>
<tr>
<td>r_9</td>
<td>.1</td>
<td>$C_e$</td>
<td>$r_9$</td>
</tr>
<tr>
<td>r_10</td>
<td>1.</td>
<td>$g_m$</td>
<td>$r_{10}$</td>
</tr>
<tr>
<td>r_11</td>
<td>1.</td>
<td>$1/R_c$</td>
<td>$r_{11}$</td>
</tr>
<tr>
<td>r_12</td>
<td>.5</td>
<td>$1/R_L$</td>
<td>$r_{12}$</td>
</tr>
</tbody>
</table>

The nonzero entries of the sparse set of connection matrices, $L_{11}$, $L_{12}$, $L_{21}$ and $L_{22}$ appear in Table 14.

Table 14.
Nonzero entries of the connection matrices for Example 5.1.

<table>
<thead>
<tr>
<th>$L_{ij}$</th>
<th>row, column value</th>
<th>row, column value</th>
<th>row, column value</th>
<th>row, column value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$L_{11}$</td>
<td>1.6 1</td>
<td>1.7 1</td>
<td>1.9 1</td>
<td>1.1 -1</td>
</tr>
<tr>
<td></td>
<td>1.11 1</td>
<td>1.12 1</td>
<td>2.7 1</td>
<td>2.9 -1</td>
</tr>
<tr>
<td></td>
<td>2.9 1</td>
<td>2.11 1</td>
<td>2.12 1</td>
<td>3.7 -1</td>
</tr>
<tr>
<td></td>
<td>3.7 1</td>
<td>3.8 -1</td>
<td>3.9 1</td>
<td>3.10 1</td>
</tr>
<tr>
<td></td>
<td>4.12 1</td>
<td>4.10 1</td>
<td>4.11 1</td>
<td>4.12 1</td>
</tr>
<tr>
<td></td>
<td>7.1 -1</td>
<td>7.2 -1</td>
<td>7.3 -1</td>
<td>8.3 1</td>
</tr>
<tr>
<td></td>
<td>8.3 -1</td>
<td>9.1 -1</td>
<td>9.2 -1</td>
<td>9.3 -1</td>
</tr>
<tr>
<td></td>
<td>9.3 -1</td>
<td>10.3 1</td>
<td>11.1 -1</td>
<td>11.2 -1</td>
</tr>
<tr>
<td></td>
<td>12.2 -1</td>
<td>12.4 -1</td>
<td>12.5 -1</td>
<td>12.2 -1</td>
</tr>
<tr>
<td>$L_{12}$</td>
<td>6.1 1</td>
<td>7.1 1</td>
<td>9.1 1</td>
<td>11.1 1</td>
</tr>
<tr>
<td></td>
<td>11.1 1</td>
<td>12.1 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$L_{21}$</td>
<td>1.1 -1</td>
<td>1.2 -1</td>
<td>1.4 -1</td>
<td>1.5 -1</td>
</tr>
<tr>
<td></td>
<td>1.5 -1</td>
<td>2.6 1</td>
<td>2.7 1</td>
<td>2.9 1</td>
</tr>
<tr>
<td></td>
<td>2.9 1</td>
<td>2.11 1</td>
<td>2.12 1</td>
<td></td>
</tr>
<tr>
<td>$L_{22}$</td>
<td>1.1 1</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The nonzero entries of $L_{21}^R$ and $V$ computed via IMSL/B routines LGINF and LLSQF.
respectively, appear in Table 15.

Table 15.
Nonzero entries of the $L_2^R$ and $V$ matrices for Example 5.1.

<table>
<thead>
<tr>
<th>matrix</th>
<th>row, column</th>
<th>value</th>
<th>row, column</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$L_2^R$</td>
<td>1,1</td>
<td>-0.25</td>
<td>2,1</td>
<td>-0.25</td>
</tr>
<tr>
<td></td>
<td>4,1</td>
<td>-0.25</td>
<td>5,1</td>
<td>-0.25</td>
</tr>
<tr>
<td></td>
<td>6,2</td>
<td>0.2</td>
<td>7,2</td>
<td>0.2</td>
</tr>
<tr>
<td></td>
<td>8,2</td>
<td>0.2</td>
<td>11,2</td>
<td>0.2</td>
</tr>
<tr>
<td></td>
<td>12,2</td>
<td>0.2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$V$</td>
<td>1,1</td>
<td>0.866025</td>
<td>2,1</td>
<td>-0.288675</td>
</tr>
<tr>
<td></td>
<td>2,2</td>
<td>-0.57735</td>
<td>2,3</td>
<td>-0.57735</td>
</tr>
<tr>
<td></td>
<td>3,8</td>
<td>1</td>
<td>4,1</td>
<td>-0.288675</td>
</tr>
<tr>
<td></td>
<td>4,2</td>
<td>0.786875</td>
<td>4,3</td>
<td>-0.211325</td>
</tr>
<tr>
<td></td>
<td>5,1</td>
<td>-0.286875</td>
<td>5,2</td>
<td>-0.211325</td>
</tr>
<tr>
<td></td>
<td>5,3</td>
<td>0.786875</td>
<td>6,4</td>
<td>0.861803</td>
</tr>
<tr>
<td></td>
<td>6,5</td>
<td>-0.138197</td>
<td>6,6</td>
<td>-0.138197</td>
</tr>
<tr>
<td></td>
<td>6,7</td>
<td>-0.138197</td>
<td>7,4</td>
<td>-0.138197</td>
</tr>
<tr>
<td></td>
<td>7,5</td>
<td>0.861803</td>
<td>7,6</td>
<td>-0.138197</td>
</tr>
<tr>
<td></td>
<td>7,7</td>
<td>-0.138197</td>
<td>8,9</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>9,4</td>
<td>-0.138197</td>
<td>9,5</td>
<td>-0.138197</td>
</tr>
<tr>
<td></td>
<td>9,6</td>
<td>-0.138197</td>
<td>9,7</td>
<td>0.861803</td>
</tr>
<tr>
<td></td>
<td>10,10</td>
<td>1</td>
<td>11,4</td>
<td>-0.138197</td>
</tr>
<tr>
<td></td>
<td>11,5</td>
<td>-0.138197</td>
<td>11,6</td>
<td>0.861803</td>
</tr>
<tr>
<td></td>
<td>11,7</td>
<td>-0.138197</td>
<td>12,4</td>
<td>-0.447214</td>
</tr>
<tr>
<td></td>
<td>12,5</td>
<td>-0.447214</td>
<td>12,6</td>
<td>-0.447214</td>
</tr>
<tr>
<td></td>
<td>12,7</td>
<td>-0.447214</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

For $q=2$, $s_1 = j10$ and $s_2 = j.6$ and $u(s_1) = u(s_2) = 1$ we compute the following nominal $\alpha$:

\[
\alpha_1 = \begin{bmatrix}
-0.157584e+00 \\
-0.333416e+00 \\
-0.323898e+00 \\
0.634977e+00 \\
0.300373e-01 \\
-0.105105e+00 \\
-0.290531e+00 \\
-0.384966e+00 \\
-0.398398e-01 \\
\end{bmatrix} + j \begin{bmatrix}
0.353877e-01 \\
-0.130905e+00 \\
-0.627419e-01 \\
-0.121971e-01 \\
0.105105e+00 \\
-0.732278e-01 \\
-0.358568e+00 \\
-0.384966e-01 \\
-0.384966e-01 \\
\end{bmatrix} = \alpha_1 \tag{5.3}
\]
The next step in the example is to simulate several 3-faults (3 parameters differ from nominal) and see if the solution algorithm works. Table 16 displays a set of 16 randomly selected fault indices and corresponding faulty parameter values. Each of these faults was simulated and the resulting measurement data applied to the solution algorithm. The final column in Table 16 indicates the performance of the algorithm for each fault. In 11 of the 16 cases the algorithm correctly identified all faulty parameters and determined their actual values. In four cases two of the three faults were correctly identified and in one case the algorithm selected as faulty a component which was actually good.

Table 16.
Fault list and algorithm performance summary for \( s_1=j10 \) and \( s_2=j.6 \).

<table>
<thead>
<tr>
<th>Actual Fault Index</th>
<th>Faulty Parameter Values</th>
<th>Algorithm Fault Index</th>
</tr>
</thead>
<tbody>
<tr>
<td>(&lt;6,7,10&gt;)</td>
<td>( r_6=.5 ) ( r_7=2. ) ( r_{10}=2. )</td>
<td>(&lt;6,7,10&gt;)</td>
</tr>
<tr>
<td>(&lt;2,5,9&gt;)</td>
<td>( r_2=1. ) ( r_5=.05 ) ( r_9=.2 )</td>
<td>(&lt;2,9&gt;)</td>
</tr>
<tr>
<td>(&lt;1,4,9&gt;)</td>
<td>( r_1=.2 ) ( r_4=.5 ) ( r_9=.2 )</td>
<td>(&lt;1,4,9&gt;)</td>
</tr>
<tr>
<td>(&lt;5,5,8&gt;)</td>
<td>( r_5=.3 ) ( r_5=.07 ) ( r_8=1.4 )</td>
<td>(&lt;5,8&gt;)</td>
</tr>
<tr>
<td>(&lt;4,5,9&gt;)</td>
<td>( r_4=.2 ) ( r_5=.05 ) ( r_9=.2 )</td>
<td>(&lt;4,5,9&gt;)</td>
</tr>
<tr>
<td>(&lt;2,8,11&gt;)</td>
<td>( r_2=.1 ) ( r_8=.5 ) ( r_{11}=.5 )</td>
<td>(&lt;2,8,11&gt;)</td>
</tr>
<tr>
<td>(&lt;2,8,12&gt;)</td>
<td>( r_2=.1 ) ( r_8=.5 ) ( r_{12}=.25 )</td>
<td>(&lt;2,8,12&gt;)</td>
</tr>
<tr>
<td>(&lt;7,8,9&gt;)</td>
<td>( r_7=.5 ) ( r_8=.2 ) ( r_9=.2 )</td>
<td>(&lt;7,8,9&gt;)</td>
</tr>
<tr>
<td>(&lt;2,4,12&gt;)</td>
<td>( r_2=.25 ) ( r_4=.5 ) ( r_{12}=.25 )</td>
<td>(&lt;2,4,12&gt;)</td>
</tr>
<tr>
<td>(&lt;5,6,8&gt;)</td>
<td>( r_5=.05 ) ( r_6=.7 ) ( r_8=1.6 )</td>
<td>(&lt;5,6,8&gt;)</td>
</tr>
<tr>
<td>(&lt;2,8,10&gt;)</td>
<td>( r_2=.25 ) ( r_8=.2 ) ( r_{10}=.2 )</td>
<td>(&lt;2,8,10&gt;)</td>
</tr>
<tr>
<td>(&lt;3,4,7&gt;)</td>
<td>( r_3=.5 ) ( r_4=.2 ) ( r_7=.2 )</td>
<td>(&lt;3,4,7&gt;)</td>
</tr>
<tr>
<td>(&lt;3,6,9&gt;)</td>
<td>( r_3=.2 ) ( r_6=.5 ) ( r_9=.2 )</td>
<td>(&lt;3,6,9&gt;)</td>
</tr>
<tr>
<td>(&lt;3,7,12&gt;)</td>
<td>( r_3=.2 ) ( r_7=.2 ) ( r_{12}=.25 )</td>
<td>(&lt;3,7,12&gt;)</td>
</tr>
<tr>
<td>(&lt;2,4,5&gt;)</td>
<td>( r_2=.3 ) ( r_4=1.5 ) ( r_5=.07 )</td>
<td>(&lt;2,4&gt;)</td>
</tr>
<tr>
<td>(&lt;5,7,12&gt;)</td>
<td>( r_5=.05 ) ( r_7=2. ) ( r_{12}=1. )</td>
<td>(&lt;5,7,12&gt;)</td>
</tr>
</tbody>
</table>

Although the circuit is theoretically 3-fault diagnosable the results in Table 16 indicate that there are some practical problems associated with the determination of certain faults. For any given fault the most reliable indication of the theoretical capability to determine the fault is a test of the Jacobian evaluated at the fault but this is impractical since it is impossible to anticipate all possible faults. Instead the
diagnosability test is based on the rank of selected columns of the Jacobian, \( J \), evaluated at the nominal point. Theoretically the results at the nominal point hold for almost all faults. Unfortunately it is quite possible for a matrix which is theoretically full rank to be less than full rank for a solution algorithm due to the finite word length of the computer.

A technique for circumventing this problem is to perform the diagnosis at several sets of test frequencies. Only those faults which are poorly conditioned at all test frequencies used would not be detectable. Suppose that the present example is repeated with the following test frequencies: \( s_1 = j4 \) and \( s_2 = j2 \). The resulting nominal \( a_i \) are:

\[
\begin{align*}
\mathbf{a}_1 & = \begin{bmatrix}
-0.152138e+00 & -0.308290e-01 \\
-0.300670e+00 & -0.464630e-01 \\
-0.639630e+00 & -0.967588e-02 \\
0.411483e-01 & -0.825390e-01 \\
-0.892647e-01 & -0.328109e-01 \\
-0.343296e+00 & -0.961794e-01 \\
0.673079e-01 & -0.867572e-01 \\
-0.267029e+00 & -0.269232e-01 \\
0.673079e-01 & -0.667572e-01 \\
-0.196403e+00 & -0.890725e-01 \\
0.577404e+00 & -0.771998e-01 \\
-0.304301e+00 & -0.467058e-01 \\
-0.246162e+00 & -0.374141e-01 \\
-0.698594e-01 & -0.883199e-01 \\
-0.201163e+00 & -0.788478e-01 \\
\end{bmatrix} + j \begin{bmatrix}
0.152138e+00 \\
0.300670e+00 \\
0.639630e+00 \\
0.411483e-01 \\
0.892647e-01 \\
0.343296e+00 \\
0.673079e-01 \\
0.267029e+00 \\
0.673079e-01 \\
0.196403e+00 \\
0.577404e+00 \\
0.304301e+00 \\
0.246162e+00 \\
0.698594e-01 \\
0.201163e+00 \\
\end{bmatrix}
\end{align*}
\]

\[
\begin{align*}
\mathbf{a}_2 & = \begin{bmatrix}
0.522551e-01 & -0.460309e+00 \\
0.500190e+00 & -0.771991e-01 \\
-0.196403e+00 & -0.690725e-01 \\
0.577404e+00 & -0.307545e+00 \\
-0.304301e+00 & -0.939940e-01 \\
-0.246162e+00 & -0.347141e-01 \\
-0.698594e-01 & -0.883199e-01 \\
-0.201163e+00 & -0.788478e-01 \\
\end{bmatrix} + j \begin{bmatrix}
-0.522551e-01 \\
-0.500190e+00 \\
0.196403e+00 \\
-0.577404e+00 \\
0.304301e+00 \\
0.246162e+00 \\
0.698594e-01 \\
0.201163e+00 \\
0.522551e-01 \\
\end{bmatrix}
\end{align*}
\]

The results of the diagnosis of the same 16 faults appears in Table 17.
Table 17.
Fault list and algorithm performance summary for $s_1=j_4$ and $s_2=j_2$.

<table>
<thead>
<tr>
<th>Actual Fault Index</th>
<th>Faulty Parameter Values</th>
<th>Algorithm Fault Index</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;6,7,10&gt;</td>
<td>$r_8=.5$</td>
<td>$r_7=2.$</td>
</tr>
<tr>
<td></td>
<td>$r_9=.05$</td>
<td>$r_{10}=2.$</td>
</tr>
<tr>
<td>&lt;2,5,9&gt;</td>
<td>$r_2=1.$</td>
<td>$r_5=.2$</td>
</tr>
<tr>
<td>:1,4,9</td>
<td>$r_1=.2$</td>
<td>$r_4=.5$</td>
</tr>
<tr>
<td>&lt;2,5,6&gt;</td>
<td>$r_2=.3$</td>
<td>$r_5=.07$</td>
</tr>
<tr>
<td>&lt;4,5,9&gt;</td>
<td>$r_4=.2$</td>
<td>$r_5=.05$</td>
</tr>
<tr>
<td>&lt;2,6,11&gt;</td>
<td>$r_2=1.$</td>
<td>$r_5=.5$</td>
</tr>
<tr>
<td>&lt;2,6,12&gt;</td>
<td>$r_2=.5$</td>
<td>$r_{12}=25.$</td>
</tr>
<tr>
<td>&lt;7,8,9&gt;</td>
<td>$r_7=.25$</td>
<td>$r_8=.2$</td>
</tr>
<tr>
<td>&lt;2,4,12&gt;</td>
<td>$r_2=.25$</td>
<td>$r_{12}=25.$</td>
</tr>
<tr>
<td>&lt;5,6,8&gt;</td>
<td>$r_5=.05$</td>
<td>$r_8=1.6$</td>
</tr>
<tr>
<td>&lt;2,8,10&gt;</td>
<td>$r_2=.25$</td>
<td>$r_{10}=2.$</td>
</tr>
<tr>
<td>&lt;3,4,7&gt;</td>
<td>$r_3=.5$</td>
<td>$r_4=2.$</td>
</tr>
<tr>
<td>&lt;3,6,9&gt;</td>
<td>$r_3=2.$</td>
<td>$r_5=.2$</td>
</tr>
<tr>
<td>&lt;3,7,12&gt;</td>
<td>$r_3=2.$</td>
<td>$r_{12}=25.$</td>
</tr>
<tr>
<td>&lt;2,4,5&gt;</td>
<td>$r_2=.3$</td>
<td>$r_4=1.5$</td>
</tr>
<tr>
<td>&lt;5,7,12&gt;</td>
<td>$r_5=.05$</td>
<td>$r_7=2.$</td>
</tr>
</tbody>
</table>

A diagnosis based on the combination of the results of Tables 15 and 16 identifies ALL faulty components in the sixteen randomly selected fault combinations. The only diagnosis errors are three of the sixteen cases in which a good component was identified as faulty.

**Example 5.7:** Next reconsider the circuit of Figure 1. Due to the large size of this example a detailed analysis is not feasible without the utilization of sparse matrix techniques. Although the adaptation of the solution program to sparse matrices is beyond the scope of the current research a limited analysis of the example is included to demonstrate the feasibility of this approach for large circuits and to illustrate some of the properties of the algorithm.

The choice of inputs and outputs are denoted in the figure as $a_i$ and $y_i$ respectively. For this example $N=26$, $n_o=4$ and $p=22$. Notice that five of the seven accessible nodes are utilized to provide the single input and four outputs shown in Figure 1. Normally it would be desirable to utilize all accessible points to acquire the maximum of available information for the diagnosis procedure however we have elected to let $n_f=5$ and wish to illustrate the case where $n_o<n_f$.

The nonzero entries of the sparse set of connection matrices, $L_{12}$ and $L_{21}$, appear in Table 18 and those of $L_{11}$ appear in Table 1 (Chapter 1). All entries of $L_{22}$ are zero. The computation of $L_{21}^R$ and $V$ was performed via the IMSL routines LGINF and LSVDF respectively. These too are sparse matrices with their nonzero entries given in Table 18.
Table 18.
Nonzero entries of matrices for Example 5.7.

<table>
<thead>
<tr>
<th>L_{ij}</th>
<th>row, column</th>
<th>value</th>
<th>row, column</th>
<th>value</th>
<th>row, column</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>L_{12}</td>
<td>1,1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L_{21}</td>
<td>1,1</td>
<td>1</td>
<td>2,9</td>
<td>1</td>
<td>3,19</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>3,26</td>
<td>1</td>
<td>4,22</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>L_{21}^-</td>
<td>1,1</td>
<td>1</td>
<td>9,2</td>
<td>1</td>
<td>19,3</td>
<td>0.5</td>
</tr>
<tr>
<td></td>
<td>22,4</td>
<td>1</td>
<td>22,3</td>
<td>0.5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>V</td>
<td>2,2</td>
<td>1</td>
<td>3,3</td>
<td>1</td>
<td>4,4</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>5,5</td>
<td>1</td>
<td>6,6</td>
<td>1</td>
<td>7,7</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>8,8</td>
<td>1</td>
<td>10,9</td>
<td>1</td>
<td>11,10</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>12,11</td>
<td>1</td>
<td>13,12</td>
<td>1</td>
<td>14,13</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>15,14</td>
<td>1</td>
<td>16,15</td>
<td>1</td>
<td>17,16</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>18,17</td>
<td>1</td>
<td>19,1</td>
<td>1</td>
<td>20,18</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>21,19</td>
<td>1</td>
<td>23,20</td>
<td>1</td>
<td>24,21</td>
<td>1</td>
</tr>
</tbody>
</table>

The component transfer functions are: $Z_i(s, r_i) = r_i s$ for $i=4,5,10,11,16,17,23,24$ and $Z_i(s, r_i) = r_i$ for the remaining $i$. The nominal component values as well as parameter values for three 5-faults appear in Table 19. Note that the component values are scaled to improve the numerical condition of the problem. The impedance scale factor is $10^2$ and the frequency scale factor is $10^7$. 
Table 19.
Nominal parameter values and selected 5-faults.

<table>
<thead>
<tr>
<th>component</th>
<th>units</th>
<th>nominal</th>
<th>fault 1 (&lt;4,13,18,20,22&gt;)</th>
<th>fault 2 (&lt;1,9,13,20,24&gt;)</th>
<th>fault 3 (&lt;3,6,14,17,25&gt;)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>ohm</td>
<td>12</td>
<td>12</td>
<td>16</td>
<td>12</td>
</tr>
<tr>
<td>2</td>
<td>mho</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
</tr>
<tr>
<td>3</td>
<td>ohm</td>
<td>56.7</td>
<td>56.7</td>
<td>56.7</td>
<td>35</td>
</tr>
<tr>
<td>4</td>
<td>farad</td>
<td>50</td>
<td>50</td>
<td>50</td>
<td>50</td>
</tr>
<tr>
<td>5</td>
<td>farad</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>mho</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>5</td>
</tr>
<tr>
<td>7</td>
<td>ohm</td>
<td>30</td>
<td>30</td>
<td>30</td>
<td>30</td>
</tr>
<tr>
<td>8</td>
<td>mho</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
</tr>
<tr>
<td>9</td>
<td>ohm</td>
<td>10</td>
<td>10</td>
<td>3</td>
<td>10</td>
</tr>
<tr>
<td>10</td>
<td>farad</td>
<td>50</td>
<td>50</td>
<td>50</td>
<td>50</td>
</tr>
<tr>
<td>11</td>
<td>farad</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>12</td>
<td>mho</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>13</td>
<td>mho</td>
<td>0.3</td>
<td>0.2</td>
<td>0.5</td>
<td>0.3</td>
</tr>
<tr>
<td>14</td>
<td>ohm</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>15</td>
</tr>
<tr>
<td>15</td>
<td>ohm</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>16</td>
<td>farad</td>
<td>50</td>
<td>50</td>
<td>50</td>
<td>50</td>
</tr>
<tr>
<td>17</td>
<td>farad</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>18</td>
<td>mho</td>
<td>10</td>
<td>18</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>19</td>
<td>ohm</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>20</td>
<td>mho</td>
<td>0.3</td>
<td>0.5</td>
<td>0.5</td>
<td>0.3</td>
</tr>
<tr>
<td>21</td>
<td>ohm</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>22</td>
<td>ohm</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>23</td>
<td>farad</td>
<td>50</td>
<td>50</td>
<td>50</td>
<td>50</td>
</tr>
<tr>
<td>24</td>
<td>farad</td>
<td>5</td>
<td>5</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>25</td>
<td>mho</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>15</td>
</tr>
<tr>
<td>26</td>
<td>ohm</td>
<td>0.78</td>
<td>0.78</td>
<td>0.78</td>
<td>0.78</td>
</tr>
</tbody>
</table>

The number of test frequencies is \( q = 2 \) and the inputs and test frequencies are:

\[
\begin{align*}
    u(s_1) &= u(j.2) = 2, \\
    u(s_2) &= u(j.01) = 1 \\
\end{align*}
\]  

The nominal values for the \( a_i \), \( i=1,2 \) appear in Table 20.
The results of algorithm for this example comprise Table 21. The program correctly identified faults 1 and 2 and four of five faulty components in fault 3. The difficulty in identifying parameter 14 is another example of a poorly conditioned problem however in this case the difficulty is not frequency dependent. The sensitivity of the four outputs to changes in parameter 14 is extremely small at all test frequencies. Under such a circumstance it is reasonable to expect difficulty in determining the parameter since the outputs contain little information about it.

Table 21.
Results for example 5.7.

<table>
<thead>
<tr>
<th>fault #</th>
<th>actual γ</th>
<th>solution γ</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>&lt;4,13,18,20,22&gt;</td>
<td>&lt;4,13,18,20,22&gt;</td>
</tr>
<tr>
<td>2</td>
<td>&lt;1,9,13,20,24&gt;</td>
<td>&lt;1,9,13,20,24&gt;</td>
</tr>
<tr>
<td>3</td>
<td>&lt;3,6,14,17,25&gt;</td>
<td>&lt;3,6,17,25&gt;</td>
</tr>
</tbody>
</table>

This solution is a good example of the reduction in the number of fault combinations which results from the determination of a feasible point. For this example \( n_f = 5 \) and \( N = 26 \). Consequently there are \( C_{26}^5 \) or 65780 possible fault combinations. Thus a "brute force" approach to this example requires the solution of 65780 different sets of fault diagnosis equations. One the other hand the algorithm developed here finds a feasible point first and then searches for the fault among the parameters which deviate significantly from nominal at the feasible point.
For example in the solution for fault 1 there were 10 parameters which deviated greater than ±10% from nominal. If the remaining components are considered good and the parameter with the greatest deviation from nominal is considered definitely bad then four faulty components remain to be found from nine possibilities. This means that the algorithm must test $C_9^4$ or 126 different faults. This is significantly fewer than 65780.

These examples employed a FORTRAN program based on the solution algorithm described in the previous section and compiled and executed on a VAX 11/780 computer. It is also noteworthy that both examples identify a reasonable number of simultaneous faults while achieving the goal, $n_c \leq \sqrt{N}$, set in [10].

6. Conclusions

Clearly the Tableau Approach is readily adaptable to the assumption of a limited number of simultaneous faults. The major problem associated with this assumption is the need to avoid testing the enormous number of fault combinations which occurs for large systems with more than a couple of faults possible. The algorithm developed here avoids this problem by utilizing the information available at the surface described by the equality constraints. The most significant aspect of this approach is that it does not require the number of test points to be greater than the number of assumed faults. This allows a reduction in test point requirements over other methods.
REFERENCES


APPENDIX A  Full Diagnosis Program

LATEST VERSION ------6 DEC 82--------------------------
program driver(input, output, tape5=input, tape6=output)

This program solves the fault diagnosis equations under
the assumption that the equations are quadratic. Sparse
matrix techniques are not used. Since all matrices are
sparse, a sparse matrix implementation is advisable.

Development stage began 21 April 1982.
Finished with completion of Chap5 on July 19, 1982.
Modified for CAS Paper October 8, 1982.
Modified to solve for normalized parameters December 6, 1982
Modified to use 11saf instead of 11saf December 6, 1982

This version uses the Newton-Raphson search direction but
uses the fact that the components of $F(x)$ are quadratic to
interpolate $\|F(x)\|_2^2$ along the search direction. Since
$\|F(x)\|_2^2$ is fourth order a five point interpolation is
exact. The interpolant is used to find the point along the
search direction for which $\|F(x)\|_2^2$ is minimized.

variable list:
integers

n number of parameters, also row dimension of Z
q number of test frequencies
p dimension of the ambiguity
in dimension of the input vector
ou dimension of the output vector
zpow power of s in Z(s), usually $-1,0,1$
i in integer work vector used by imsl routine 11saf
kb used by imsl routine 11saf
real
111  com connection matrix
112  com connection matrix
121  com connection matrix
122  com connection matrix
121r  right inverse of 121
u  matrix of basis vectors for the null space of 121
rnom  nominal parameter values
ract  actual parameter values
rkth  parameter values during the iteration process
rkthnm  rkth/rnom (normalized unknown)
deu  deviation from nominal for actual
a  real matrix equivalent to if
ff  equivalent to f (see f)
dd  equivalent to d (see d)
x  equivalent to x (see x)
tol  used by imsl routine llsaf
normsa  function name
lambda  function name
g  array of function values along the search direction
time1  beginning cpu time
time2  ending cpu time
complex
s  array of test frequencies (equiv. to ss)
u  array of input vectors (equiv. to uu)
f  right hand vector of algqr. (equiv. to ff)
d  search direction (equiv. to dd)
x  point along search direction (equiv. to xx)
tmp  temporary storage
aa  work storage
array of nominal test outputs (equiv. to \(Yymnom\))

\(\alpha\) array of nominal ambiguity vectors (equiv. to \(aalpha\))

\(\text{ymact}\) array of actual test outputs (equiv. to \(\text{ymact}\))

Since the free-format read(5,*) won't read complex numbers
the variables \(s, u, Yymnom, \alpha, \text{ymact}\) are made equivalent to
real arrays of double size. The reals are read but the
complex are used.

logical convrs

integer n, q, p, in, ou, zpow(30)
integer lda, ldjf, ld111, lda0, ldb0, ld1a, ldv, ip(200), kb
real 111(30,30), 112(30,5), 121(10,30), 122(10,5), 121r(30,10)
real u(30,25), rnom(30), ract(30), a(250,250), ff(250), dd(250)
real ss(10), uu(10,5), Yymnom(20,5), aalpha(50,5), \(\text{ymact}(20,5)\)
real norms, lambda, tol, xx(250), s(5), dev(30)
real time1, time2, rkth(30), rkthnm(30), c(4)
complex s(5), u(5,5), Yymnom(10,5), aalpha(25,5), \(\text{ymact}(10,5)\)
complex if(150,250), d(125), f(125), b0(30,5), a0(30,5)
complex temp(200), aa(30,5), x(125)
equivalence (ss, s), (uu, u), (aalpha, alpha)
equivalence (\(\text{ymact}, \text{ymact}\)), (\(Yymnom, Yymnom\))
equivalence (ff, f), (dd, d), (xx, x)

call second(time1)

set the row dimensions

lda=250
ldjf=150
ld111=30
ldv=30
ld1a=25
lda0=30
ldb0=30

c
set precision of solution ie .01 = 1% 
prec = .001

c
Input secti,
read in from standard input

read(5,*) n, a, p, in, ou
read(5,*) (rnom(i), i=1, n)
read(5,*) (ract(i), i=1, n)
do 55 i=1, n
55 read(5,*) (111(i, j), j=1, n)
do 60 j=1, in
60 read(5,*) (112(i, j), j=1, n)
do 65 i=1, ou
read(5,*)(121(i, j), j=1, n)
do 70 j=1, in
read(5,*)(122(i, j), i=1, ou)
do 75 j=1, ou
read(5,*)(121r(i, j), j=1, n)
do 80 i=1, p
read(5,*)(u(i, j), i=1, n)
read(5,*)(zrow(i), i=1, n)
read(5,*)(ss(i), i=1, 2*q)
do 85 i=1, in
read(5,*)(uu(2*i-1, j), uu(2*i, j), j=1, q)
do 90 i=1, ou
read(5,*)(xymnom(2*i-1, j), xymnom(2*i, j), j=1, q)
do 95 i=1, p
read(5,*)(alpha(2*i-1, j), alpha(2*i, j), j=1, q)
do 100 i=1, ou
read(5,*)(xymact(2*i-1, j), xymact(2*i, j), j=1, q)
c ngraph=0
itmax=200
do 98 i=1, p
read(5,*, end=101)
read(5,*, end=101) ngraph, itmax
c c
print a program headings
write(6,102)
format(/5x,38hFault Diagnosis Program-Tableau Method,/, &
10x, 28h(Interpolate to find lambda))
write(6,103)
format(/10x,18hNominal Parameters)
write(6,107) (r(r(i, i)'=rnom(i), i=1, n)
format(4(3x,a2,i2,a2,e10.4))
write(6,110)
format(/18x,16hTest Frequencies)
do 111 i=1, q
write(6,112) i, s(i)
write(6,113)(uu(j, i), j=1, 2*in)
c continue
112 format(12x,2hs(i), i2, 2h)=, e11.4, 3h +, e11.4)
113 format(12x,2hu=, 10(f4, 1, 1x, f4, 1, 5x))
c c
compute the vector b0 = 121r(xymact-122*u)
c do 130 i=1, q
do 125 j=1, ou
tmp(j)=cmp1x(0, 0, )
do 120 k=1, in
120 tmp(j)=tmp(j)+122(j, k)*u(k, i)
125 tmp(j)=xymact(j, i)-tmp(j)
do 130 j=1, n
b0(i, i)=cmp1x(0, 0, )
do 130 k=1, ou
130 b0(i, i)=b0(i, i)+121r(j, k)*tmp(k)
c c
c Compute the vector a0

do 204 i=1,q
  do 204 j=1,n
    a0(j,i)=cmplx(0.,0.)
  do 201 k=1,n
201   a0(j,i)=a0(j,i)+111(j,k)*b0(k,i)
  do 204 k=1,n
204   a0(j,i)=a0(j,i)+112(j,k)*u(k,i)

do 280 i=1,n
  rkthnm(i)=1.
280   dev(i)=(ract(i)-rnom(i))*100./rnom(i)
  iter=0
  xlam=-1.
  iter=iter+1
  if(iter.ge.itmax) go to 900

  do 295 i=1,n
295   rkth(i)=rkthnm(i)*rnom(i)
  evaluate f
  do 300 i=1,q
    call quadf(f,i*n-n+1),rkth, alpha(1,i), s(i), zpow, 111, 1d111,
    v, ldv, a0(1,i), b0(1,i), n, p, tmf)
    & continue
    if(normsq(ff,2*n*q), le, 1.e-10) go to 900
  stop because the algorithm appears stuck in a relative min.
  if(xlam.eq.0.) write(6,305)
305 format(10x,37hecaught in a point of relative minimum)
  if(xlam.eq.0.) stop

  call jacob(jf,1d1f,rkth,alpha,1da,s,n,p,q,zpow,111,1d111,
    & v, ldv, a0, lda0, b0, 1db0, tmf, aa)

  call cxtorl(a,lda,jf,ld1f,n,p,q)
  since jacob computes [partial f / partial rkth] 
  the last n columns of the matrix a must be adjusted to get
  [partial f / partial rkthnm]
  do 315 i=1,2*n*q
  do 315 j=1,n
315   a(1,i,2*p+j)=a(1,i,2*p+j)+rnom(j)
  see imsldoc llsqaf for meaning of tol
  tol=0
  value for kb indicates a is assumed to have independent cols.
  kb=2**p+q+n
  kb=0
  use llsqaf to find the search direction
  llsqaf is from the imsl library
  call llsqaf(a,lda,2*n*q,2**p+q+n,f,tol, kb, d, tmf, if, ier)
  cdd
  nf=1
ind=1
0(1)=0.
0(2)=0.
0(3)=1.
call lbbf(a, lda, 2*n*q, 2*p*q+n, f, lda, nf, ind, c, d, lda, ip, jf, ier)
c
NOTE: the r part of d is normalized

c
option to skip the lambda determination and set it to -1

c
if(iter <= nraph) go to 400

c
do 319 j=1, q
319 x((j-1)*p+i)=alpha(i, j)-0.0*d((j-1)*p+i)
do 321 i=1, n
321 xx((2*p**q+i))=(rkthnm(i)-0.0*dd(2*p**q+i))*rnom(i)
c
the -0.0 is the lower end of the interpolation data

c
compute function |fl**2 at lambda=0. -25 -5 -75 -1.
do 339 ilam=1, 5
do 333 i=1, q
call quadf(f(i*n-n+1), xx((2*p**q+i)), x[((i-1)*p+1), s(i), zpow, 111, ldl11, v, lda(1, i), b0(1, i), n, p, tmp)
333 continue
331 continue
329 continue
329 x((j-1)*p+i)=x((j-1)*p+i)-.25*d((j-1)*p+i)
do 331 i=1, n
331 xx((2*p**q+i))=xx((2*p**q+i)-.25*dd(2*p**q+i))*rnom(i)
c
the -.25 is the increment between the interpolation data
339 continue
339 xlam=lambda(s)
c
update the alpha's and r

340 continue

340 continue
310 j=1, q
310 alpha(i, j)=alpha(i, j)+xlam*dd((j-1)*p+i)
cnovr=.true.
do 420 i=1, n
420 if(abs(xlam*dd(2*p**q+i)).eq.1, prec) convr=.false.
420 rkthnm(i)=rkthnm(i)+xlam*dd(2*p**q+i)
420 if(convr) go to 900

400 write the parameters as a progress report
write(6, *) '
rank of a : ', int(c(4))
if(iter <= nraph) then
write(6, 455) iter, xlam
else
write(6, 456) iter, xlam
endif
455 format('/5x, 26hnormalized r at iteration ', i3, 4x, 8h(lambda=, e12.5, 1h))
456 format('/5x, 26hnormalized r at iteration ', i3, 4x, 8h(lambda=, e12.5, 8h: forced))
write(6, 457) (rkthnm(i), i=1, n)
457   format(5x,4e15.5)
go to 290  
   write out final values
900   continue
   call second(time2)
   write(6,923) iter-1, time2-time1
923   format(5x, i5, 2x, 16h iterations and , e12.6, 14h sec. required)
   write(6,925)
925   format(5x, 36h Parameter Values at Solution Point , & 19h% dev. from nominal, 17h% error in sol.)
do 950 i=1, n
927   format(10x,2hr(, i2,4h) = , e11.4, 18x, f10.2, 5x, f10.2)
950   write(6,927) i, rkthnm(i)*rnom(i), dev(i), & 100. *(ract(i)-rkthnm(i)*rnom(i))/ract(i)
   stop
end

subroutine quadf(f,r, alph, s, zpow, l11, ld111, v, ldv, a0, b0, n, p, tmp)

This routine computes the nonlinear function which corresponds to the fault diagnosis equation at one test frequency.

argument list:
f          output: f = Z(s,r)[L Valph +a0(s )] - Valph -b0(s )
i i i i
r          input: parameter vector
alph       input: ambiguity vector
s          input: test frequency
zpow       input: exponent of s in Z
l11        input: com connection matrix
ld111      input: row dimension of l11 in calling program
v          input: null space basis of 121
ldv        input: row dimension of v in calling program
a0         input: particular solution for a
b0         input: particular solution for b
n  input: number of parameters & row dimension of Z
p  input: number of columns in v
tmp input: temporary storage (at least 2*n)

integer zpow(1), n, p, ld111, ldv, i, j
real r(1), l11(1d111, 1), v(ldv, 1)
complex alpha(1), s, f(1), a0(1), b0(1), tmp(1)

do 20 i=1, n
tmp(i)=0.
do 10 j=1, p
 tmp(i)=tmp(i)+v(i, j)*alpha(j)
 20 f(i)=-tmp(i)-b0(i)
do 40 i=1, n
tmp(i+n)=0.
do 30 j=1, n
 tmp(i+n)=tmp(i+n)+l11(i, j)*tmp(j)
tmp(i+n)=tmp(i+n)+a0(i)
40 f(i)=f(i)+r(i)*s**zpow(i)*tmp(i+n)
return
end

This routine computes the fault diagnosis Jacobian to be
used in the Newton-Raphson or other iteration schemes.

argument list

jf  output: the Jacobian matrix
ldj  input: the row dimension of jf in the calling program
r  input: parameter vector
alpha input: matrix of ambiguity vectors
ldal input: row dimension of alpha in the calling program
s  input: array of test frequencies
n input: number of parameters & row dimension of Z
p input: number of columns in v
q input: number of test frequencies
zpow input: exponent of s in Z
111 input: connection matrix
1d111 input: row dimension of 111 in calling program
v input: null space basis of 121
1dv input: row dimension of v in calling program
a0 input: array of particular solutions for a
1da0 input: row dimension of a0 in the calling program
b0 input: array of particular solutions for b
1db0 input: row dimension of b0 in the calling program
tmp input: temporary storage (at least 2*n)
aa input: work array (at least n*q)

Note: For arrays alpha(i,j),a0(i,j),b0(i,j) & aa(i,j) the subscript j means the data of this column corresponds to test frequency s(j).

integer zpow(lda0), n, p, q, ldj, ldj, lda1, ld111, ldv, lda0, ldb0
integer i, j, k, j
real r(lda0), 111(ld111,1d111), v(ldv, lda1)
complex jf(ldj,ldj), alpha(lda1,q), s(q), a0(lda0,q), b0(ldb0,q)
complex tmp(ldj), aa(n,q), sz

Compute the vector aa=L11*V*alpha+a0

\[ \begin{align*}
\text{do 304 } i=1, n \\
\text{do 301 } j=1, n \\
\text{tmp}(j)=\text{compl}(0,0) \\
\text{do 301 } k=1, p \\
\text{301 } \text{tmp}(j)=\text{tmp}(j)+v(j,k)\times\text{alpha}(k,i) \\
\text{do 304 } j=1, n \\
\text{aa}(j,i)=a0(j,i) \\
\text{do 304 } k=1, n \\
\text{304 } \text{aa}(j,i)=\text{aa}(j,i)+111(j,k)\times\text{tmp}(k) \\
\end{align*} \]

Compute the Jacobian JF(x)

\[ \begin{align*}
\text{do 601 } i=1, n+q \\
\end{align*} \]
do 601 j=1, q*p+n
jf(i,j)=cmplx(0.,0.)
do 610 k=1, q
do 605 i=1, n

Compute dfi/dr * si(alpha)

csz=s(k)**2 pow(i)
jf(n*(k-1)+i, q*p+i)=aa(i,k)*sz
do 605 j=1, p
do 605 j.j=1, n

Compute Z(si,r)*L11*V-V

c605 .if(n*(k-1)+i, p*(k-1)+j)=
& .f(n*(k-1)+i, p*(k-1)+j)+111(i, jj)*v(jj, j)
do 610 i=1, n
do 610 j=1, p
sz=s(k)**2 pow(i)
610 .if(n*(k-1)+i, p*(k-1)+j)=jf(n*(k-1)+i, p*(k-1)+j)*r(i)*sz-v(i, i)

return
end

subroutine oxtor1(a, lda, jf, ldjf, n, p, q)

This routine converts the complex Jacobian into an equivalent real matrix. Note: The conversion is somewhat of a hybrid since the original unknown vector is part complex (the alpha's) and part real (the r's).

argument list

a       output: the real equivalent of jf
lda      input: the row dimension of a in the calling program
jf       input: the complex matrix to be converted
ldjf     input: the row dimension of jf in the calling program
n        input: the dimension of r, the real sub-vector
p        input: the dimension of each complex sub-vector
q        input: the number of complex sub-vectors
integer lda, ldj, n, p, q, i, j
real a(lda, 1)
complex jf(ldj, 1)

do 820 i = 1, n
    do 810 j = 1, p
        a(2*i-1, 2*j-1) = real(jf(i, j))
        a(2*i, 2*j) = real(jf(i, j))
        a(2*i-1, 2*j) = -imag(jf(i, j))
        a(2*i, 2*j-1) = imag(jf(i, j))
    enddo
enddo

real function normsq(f, n)
integer n, i
real f(n)
normsq = 0.
    do 10 i = 1, n
        normsq = normsq + f(i)*f(i)
    enddo
    return
end

real function lambda(s)

This program uses 5 points along a function s(x) to find the point, lambda, at which the function is a minimum. The ordinate points are assumed to be 0, 0, -25, -5, -75 and -1

If \( s(x) = a1*x^4 + a2*x^3 + a3*x^2 + a4*x + a5 \) then
variable list

real

g input: contains \( s(-0.0), \ldots, s(-1.0) \)
vand data: contains the vandermonde matrix

vand2 work: scratch copy of vand to compute \( a \)

note: It would be more efficient to use the inverse of vand here.

a work: coefficients of the interpolant

becomes the coefficients of 1st derivative

complex

z work: the zeroes of the 1st derivative

subroutines

leqtlf linear equation solver from imsl

zpolr polynomial root finder

integer i, j
real vand(5, 5), vand2(5, 5), a(5), s(5), wk(10)
complex z(3)
data vand/0., 3.90625e-3, .0625, 31640625, 1.0, -1.5625e-2, -125.,
& 421875, -1.0, .0625, .25, .5, 75.0, 1.
& 1.0, 1.0, 1.0/
set fmin to the value of the function at the current point so that
only points representing a decrease from this value can be selected
fmin=s(1)
do 80 i=1, 5
do 80 j=1, 5
80 vand2(i, j)=vand(i, j)
call leqtlf(vand2, 1, 5, 5, 0, wk, ier)
c
print the coefficients
write(6, 100) (5-i, s(i), i=1, 5)
100 format((//5(1ha, i1, 1h=, e10.4, 2x))
iroot=0
c find coef. of 1st derivative
do 10 i=1, 5
10 a(i)=float(5-i)*s(i)
c find zeroes of 1st derivative
call zpolr(a, 3, z, ier)
do 50 i=1,3
if(abs(z(i)) .gt. 1. e15) go to 30
if(abs(aias(z(i))) .gt. 1. e-5) go to 30
this prevents a search in the opposite direction
if(real(z(i)) .lt. 0.) go to 30
flam=real(z(i))
fmag=0.
do 25 k=1,5
 fmag=fmag+s(k)*flam**(5-k)
25 if(fmin .le. fmag) go to 30
iroot=i
fmin=fmag
continue
50 continue
if(iroot .ne. 0) go to 90
if(s(1) .lt. 0.) then
if the interpolant is upside down try lambda=1.
 lambda=-1.
 return
else
 lambda=0.
 write(6,99)
 endif
99 format(5x,20hERROR: No Minimum !!)
90 lambda=real(z(iroot))
if(abs(lambda) .gt. 1. 15.) then
 lambda=-1.
 write(6,*)'error: lambda too large'
 endif
return
end subroutine second(time)
real time
time=putim()
return
end
APPENDIX B: Limited Fault Diagnosis Program

EXPERIMENTAL VERSION -------11 DEC 82-----------------------

program nfmain

This program solves the fault diagnosis equations under the assumption that the equations are quadratic and that the number of faults is limited.

Development stage begun December 6, 1982.

Based on the fault diagnosis program driver. f.

logical converse
integer n, q, p, in, ou, zpow(30)
integer lda, ldif, ldll1, lda0, ldb0, ldal, ldv, ifp(200)
real 111(30,30), 112(30,5), 121(10,30), 122(10,5), 121r(30,10)
real v(30,25), rnom(30), ract(30), a(250,250), ff(250), dd(250)
real ss(10), uu(10,5), ymom(20,5), aalpha(50,5), ymac(20,5)
real normsa, lambda, xx(250), g(5), dev(30), beta(30)
real time1, time2, rkth(30), rkthnm(30), c(4), anull(250,360)
complex s(5), u(5,5), ymom(10,5), aalpha(25,5), ymac(10,5)
complex ifp(150,300), d(125), f(125), b0(30,30), a0(30,5)
complex tmp(200), aa(30,5), x(125)
equivalence (ss,s), (uu,u), (aalpha,aalpha)
equivalence (ymac,ymac), (ymom,ymom)
equivalence (ff,f), (dd,d), (xx,x), (anull,if)

call second(timel)

set the row dimensions

lda=250
ldif=150
ldll1=30
ldv=30
ldal=25
lda0=30
ldb0=30

set precision of solution ie .01 = 1%
Prec = .0000001

Input section
read in from standard input
read(5, *) n, q, p, in, ou
read(5, *) (rnom(i), i=1, n)
read(5, *) (ract(i), i=1, n)
do 55 i=1, n
read(5, *) (111(i, j), j=1, n)
do 60 j=1, in
read(5, *) (112(i, j), i=1, n)
do 65 i=1, ou
read(5, *) (111(i, j), j=1, n)
do 60 j=1, in
read(5, *) (112(i, j), i=1, ou)
do 65 i=1, in
read(5, *) (121r(i, j), i=1, n)
do 70 j=1, ou
read(5, *) (112j(i, j), i=1, n)
do 70 j=1, in
read(5, *) (121r(i, j), i=1, ou)
do 75 j=1, in
read(5, *) (121r(i, j), i=1, n)
do 80 j=1, p
read(5, *) (v(i, j), i=1, n)
do 85 j=1, in
read(5, *) (uu(2*i-1, j), uu(2*i, j), j=1, q)
do 90 i=1, ou
read(5, *) (yyYnom(2*i-1, j), yyYnom(2*i, j), j=1, q)
do 95 i=1, in
read(5, *) (aaalpha(2*i-1, j), aaalpha(2*i, j), j=1, q)
do 100 i=1, ou
read(5, *) (yyYmaot(2*i-1, j), yyYmaot(2*i, j), j=1, q)

itmax=200
do 98 i=1, p
read(5, *, end=101)
naf=1
error=. 15
read(5, *, end=101) naf, error

c
print a program heading
write(6, 102)
format(//20x, 'Fault Diagnosis Program-Tableau Method', /, & 25x, '(Interpolate to find lambda')
write(6, 103)
format(//25x, 'Nominal Parameters')
write(6, 107) ('r(', i, ')='', rnom(i), i=1, n)
format(4(3x, a2, i2, a2, e10. 4))
write(6, 110)
format(//25x, 'Test Frequencies')
do 111 i=1, q
write(6, 112) i, s(i), (uu(j, i), j=1, 2*iin)
112 format(10x, 's(', i2, ')='', e11. 4, ' +i', e11. 4, & 2x, 'u='', 10(f4. 1, ' +i', f4. 1, 5x))
c
write(6, 106) # of allowable faults, naf
write(6, 107) allowable error, error

c compute the vector b0 = 121r(yyYmaot-122u)
do 130 i=1, q
do 125 j=1, ou
tmp(j)=cmplx(0., 0.)
do 120 k=1, in
120 tmp(j)=tmp(j)+122(j, k)*u(k, i)
125 tmp(j)=tmpact(j, j)-tmp(j)
do 130 j=1, n
b0(j, i)=cmp1x(0., 0.)
do 130 k=1, ou
130 b0(j, i)=b0(j, i)+121r(j, k)*tmp(k)

**Compute the vector a0**
do 204 i=1, q
do 204 j=1, n
a0(j, i)=cmp1x(0., 0.)
do 201 k=1, n
201 a0(j, i)=a0(j, i)+111(j, k)*b0(k, i)
do 204 k=1, in
204 a0(j, i)=a0(j, i)+112(j, k)*u(k, i)
do 280 i=1, n
rkthnm(i)=1.
280 dev(i)=(ract(i)-rnom(i))*100 /rnom(i)
iter=0
itnext=0
xlam=-1.
write(6, '(////))')
290 iter=iter+1
if(iter > itmax) then
      write(6, 'iteration limit exceeded')
      go to 900
endif
295 rkth(i)=rkthnm(i)*rnom(i)
do 300 i=1, q
call quadf(f(i*n-n+1), rkth, alpha(1, i), s(i), zpomw, 111, 1d111.
&       v, 1dv, a0(1, i), b0(1, i), n, p, tmp)
300 continue
fnorm=normsq(ff, 2*n*q)
write(6, ' ITERATION ** iter, ** F1, fnorm
write(6, ' if(fnorm 1e. 1. e-10) go to 500

stop because the algorithm appears stuck in a relative min.
if(xlam.eq.0.) write(6, 305)
305 format(10x, 37hecaught in a point of relative minimum)
if(xlam.eq.0.) stop
call jacob(ff, 1df, rkth, alpha, 1dal, s, n, f, q, zpomw, 111, 1d111.
&       v, 1dv, a0, 1da0, b0, 1db0, tmp, aa)
call extor1(a, 1da, if, 1dif, n, f, q)
since jacob computes \( \frac{\partial f}{\partial \partial} \) the last \( n \) column of the matrix \( \mathbf{A} \) must be adjusted to get \( \frac{\partial f}{\partial \partial} \nabla \mathbf{A} \)

\[
\text{do 315 } i=1, 2*n*q
\]
\[
\text{do 315 } j=1, n
\]
\[
a(i, 2*n*q+j) = a(i, 2*n*q+j)*\text{rnom}(j)
\]

use \text{lbbqf} to find the search direction
\text{lbbqf} is from the \text{imsl} library

\[
\text{nf}=1
\]
\[
\text{ind}=1
\]
\[
\text{c}(1)=0
\]
\[
\text{c}(2)=1, \text{e}-5
\]
\[
\text{c}(3)=1
\]

\[
\text{call lbbqf}(a, \text{lda}, 2*n*q, 2*n*n+q, f, \text{lda}, \text{nf}, \text{ind}, c, d, \text{lda}, \text{ip}, \text{if}, \text{ier})
\]

\text{NOTE: the r part of \( \mathbf{d} \) is normalized}

\[
\text{do 319 } j=1, q
\]
\[
\text{do 319 } i=1, p
\]
\[
\times((j-1)*p+i) = \alpha(i, j)-0.0*d((j-1)*p+i)
\]
\[
\text{do 321 } i=1, n
\]
\[
\times(2*p*q+i) = (\text{rkthnm}(i)-0.0*\text{dd}(2*p*q+i)) * \text{rnom}(i)
\]

the -0.0 is the lower end of the interpolation data

\[
\text{compute function } \mid f \mid **2 \text{ at } \lambda = 0. -25 - .5 - 75 - 1
\]
\[
\text{do 339 } \text{ilam}=1, 5
\]
\[
\text{do 333 } i=1, q
\]

\[
\text{call quadr}(f(i*n+i+4), \times(2*p*q+i), ((i-1)*p+i), s(i), z, \text{fom}, 111, 1d111, v, 1d0, a0(1, i), b0(1, i), n, p, \text{tmp})
\]

\[
\text{continue}
\]

\[
\alpha(i, j) = \text{normsq}(f, 2*n*q)
\]
\[
\text{do 329 } j=1, q
\]
\[
\text{do 329 } i=1, p
\]
\[
\times((j-1)*p+i) = \times((j-1)*p+i) - 25*d((j-1)*p+i)
\]
\[
\text{do 331 } i=1, n
\]
\[
\times(2*p*q+i) = \times(2*p*q+i) - 25*\text{dd}(2*p*q+i)*\text{rnom}(i)
\]

the - 25 is the increment between the interpolation data

\[
\text{continue}
\]

\[
\times(2*p*q+i) = \times(2*p*q+i) - 25*\text{dd}(2*p*q+i)*\text{rnom}(i)
\]

\[
\text{continue}
\]

\[
\times(2*p*q+i) = \times(2*p*q+i) - 25*\text{dd}(2*p*q+i)*\text{rnom}(i)
\]

update the alpha's and \( r \)

\[
\text{do 410 } j=1, q
\]
\[
\text{do 410 } i=1, p
\]
\[
\alpha(i, j) = \alpha(i, j) + \times(2*p*q+i)
\]

\[
\text{convrs} = \text{true}
\]
\[
\text{do 420 } i=1, n
\]

\[
\text{if(} \text{abs}(\times(2*p*q+i)), \text{gt}, \text{prec}) \text{ convrs} = \text{false}
\]

\[
\text{rkthnm}(i) = \text{rkthnm}(i) + \times(2*p*q+i)
\]
\[
\text{if(} \text{convrs} \text{go to 500}
\]

write the parameters as a progress report

\[
\text{write(6,*) 'rank of a : ' } \text{int(c(4))}
\]

\[
\text{write(6,455) iter, } \times(2*p*q+i)
\]

\[
\text{format(10x, 26hnormalized r at iteration, i3, 2x, &}
\]
\[
8h(\text{lambda}=e12.5, 1h))
\]

\[
\text{write(6,456) (rkthnm(i), i=1, n)}
\]

\[
\text{format(5x, 4e15 5)}
\]
at this point the iteration has found a "feasible solution"
if(itnext.eq.1) go to 900

rkth(i)=rkthm(i)*rnom(i)
call jacob(if,ldif,rkth,alpha,lda,s,n,p,q,zpow,111,ld111,nu,ldv,a0,lda,b0,ldb0,tmp,aa)
call cxtor1(a,lda,if,ldif,n,p,q)
do 515 i=1,2*n*q
do 515 j=1,n

alpha(i,j)=alpha(i,j)+d((j-1)*p+i)
convs=. true.
do 620 i=1,n
if(abs(dd(2*p*q+i)).gt.frec) convs=. false.

rkthnm(i)=rkthnm(i)+dd(2*p*q+i)
itnext=iter+1
write(6,630) iter,beta(nldim+1)
beta(nldim+1) contains the best norm for r
format(10x,'normalized r at iteration ',i3.2x,&
  'null space step: rnorm= ',e10.4)'
if(.not.convs) go to 456

write(6,630) iter-1,time2-time1
900 format(5x,15,2x,16hiterations and ,e12.6,14h sec required)
write(6,925)
925 format(5x,36hParameter Values at Solution Point , &
  19% dev. from nominal. 17h % error in sol.)
do 950 i=1,n
927 format(10x,2hr,('i2,4h) = ',e11.4,18x,f10.2,5x,f10.2)
950 write(6,927) i,rkthnm(i)*rnom(i),dev(i),&
  100.*(ract(i)-rkthnm(i)*rnom(i))/ract(i)
stop
end

write(6,'(///)')
go to 290

rkthm(i)=rkthm(i)*rnom(i)
call jacob(if,ldif,rkth,alpha,lda,s,n,p,q,zpow,111,ld111,nu,ldv,a0,lda,b0,ldb0,tmp,aa)
call cxtor1(a,lda,if,ldif,n,p,q)
do 515 i=1,2*n*q
do 515 j=1,n

rkth(i)=rkth(i)*rnom(i)
call jacob(if,ldif,rkthm(alpha,lda,s,n,p,q,zpow,111,ld111,nu,ldv,a0,lda,b0,ldb0,tmp,aa)
call cxtor1(a,lda,if,ldif,n,p,q)
do 515 i=1,2*n*q
do 515 j=1,n

alpha(i,2*p*q+i)=alpha(i,2*p*q+i)*rnom(i)
call null(a,lda,2*n*q,2*p*q+n,anull,lda,nldim,dd,tmp)
call finder(anull(2*p*q+1,1),lda,n,nldim,rkthnm,naf,error,beta)
compute anull*beta and put it in d

do 600 j=1,2*p*q+n
dd(i)=0.
do 600 j=1,nldim
dd(i)=dd(i)+anull(i,j)*beta(j)
c

do 610 i=1,q
do 610 i=1,p

alpha(i,j)=alpha(i,j)+d((j-1)*p+i)
convs=. true.
do 620 i=1,n
if(abs(dd(2*p*q+i)).gt.frec) convs=. false.

rkthnm(i)=rkthnm(i)+dd(2*p*q+i)

format(10x,'normalized r at iteration ',i3.2x,&
  'null space step: rnorm= ',e10.4)'
if(.not.convs) go to 456

write(6,630) iter,beta(nldim+1)
beta(nldim+1) contains the best norm for r

write(6,630) iter, time2-time1
900 format(5x,15,2x,16hiterations and ,e12.6,14h sec required)
write(6,925)
925 format(5x,36hParameter Values at Solution Point , &
  19% dev. from nominal. 17h % error in sol.)
do 950 i=1,n
927 format(10x,2hr,('i2,4h) = ',e11.4,18x,f10.2,5x,f10.2)
950 write(6,927) i,rkthnm(i)*rnom(i),dev(i),&
  100.*(ract(i)-rkthnm(i)*rnom(i))/ract(i)
stop
end

subroutine finder(anull,lda,n,nldim,rkthnm,naf,error,beta)
integer neql, nldim, n, ldr, ineal(30), ineal(30), iwk(30),
isave(30), index(30)
&
real rkthnm(1), diff(30), rdiff(30), rnull(30, 30), r(30),
&
bsave(30), anull(1da, 1), beta(1), wk(2500), c(4)
&
ldr=30
error=err
do 30 i=1, n
30
diff(i)=1. -rkthnm(i)

c
determine which parameters are already at nominal
ii=0
do 100 i=1, n
if(abs(diff(i)) le error) then
  ii=ii+1
  ineal(ii)=i
endif
100 continue

neql=ii
noneql=n-neql

c
set ineal
130
ii=0
do 150 i=1, n
do 140 j=1, neql
140
if(i.eq.ineal(j)) go to 150
ii=ii+1
ineal(ii)=i
150 continue
write(6,*)' possible faults', (ineal(i), i=1, ii)

sumold=1. e+10
icnold=n

c
ijob=0

c
add a group of the possibly bad to the good (nadd=noneql-naf)
nadd=noneql-naf
if(nadd lt 0) nadd=0
160 call incr(index, nadd, noneql, ijob)
170 write(6,*)'index', (index(i), i=1, nadd)
if(ijob.eq.0) go to 240

c
do 170 i=1, nadd
170 ineal(neql+i)=ineal(index(i))
neags=neql+nadd
do 180 i=1, neags
rdiff(i)=diff(ineal(i))
do 180 j=1, nldim
180 rnull(i, j)=anull(ineal(i), j)

c
ind=1
ncols=1
o(1)=0
o(2)=1. e-5
o(3)=1

call llibaf(rnull, ldr, neags, nldim, rdiff, ldr, ncols, ind, o.
& beta, idr, iwk, wk, k)

c compute a trial value of r
  do 190 i=1, n
  r(i)=rkthnm(i)
  do 190 j=1, nldim
  r(i)=r(i)+anull(i, j)*beta(j)

c compute a performance index to evaluate the fault combination
  sum=0.
  do 210 i=1, neqns
  errxx=abs(r(i-1)-1.)
  sum=sum+errxx**2
  210 continue

write(6,*) ' sum: ', sum

write(6,*) ' ieql.(ieql(i)), i=1, neqns), ' sum: ', sum

c reject a solution with negative parameters
  do 215 i=1, n
  if(r(i).lt.0.) go to 160

continue

write(6,*) ' sum: ', sum

write(6,*) ' ieql.(ieql(i)), i=1, neqns), ' sum: ', sum

c compare to all previous
  if(sum.lt.sumold) then
    sumold=sum
    do 220 i=1, neqns
    isave(i)=ieql(i)
    do 230 i=1, nldim
    bsave(i)=beta(i)
    230 continue
  endif
  go to 160

continue
  do 250 i=1, nldim
  beta(i)=bsave(i)
  beta(nldim+1)=sumold
  return
end

subroutine incr(index, ni, n, ijob)

c This subroutine increments the index vector, index.
The vector index should list the indices of n items taken ni at
a time.

variable list

index input/output: The index vector
ni input: Order of index
n input: Order of full set
i job input/output: Control on input 0=initialize
                        1=increment
Flag on output 0=end

integer index(ni, ni, n, ijob)

if((ni. eq. 0) and (i job. eq. 0)) then
  ijob=1
  return
else if((ni. eq. 0) and (i job. eq. 1)) then
  ijob=0
  return
else if(i job. eq. 0) then
  do 110 i=1, ni
       index(i)=i
       return
  enddo
 110
else if(ni. eq. n) then
  ijob=0
  return
else
  index(ni)=index(ni)+1
  do 130 i=ni, 2, -1
     if(index(i). gt. n-ni+i) then
       index(i)=index(i)+1
       index(i-1)=index(i-1)+1
     endif
  130
  continue
  do 150 i=ni+1, 2
     if(index(i). gt. n-ni+i) index(i)=index(i-1)+1
     endif
  150
  endif
end

subroutine nulla(a, lda, nrows, ncols, anull, ldnul l, ndim, s, wk)

This subroutine computes the nullspace of the matrix a.
Initially the nullspace dimension is set to ncols-nrows
if this is positive or zero otherwise. The routine then
checks the singular values. The nullspace dimension is
incremented for every singular value which is too small
relative to the largest (first).

variable list

a input: matrix for which nullspace is desired
  a is destroyed
lda input: row dimension of a
nrows input: number of rows in a
The following subroutine listed in Appendix A are also required:

- quadf
- jacob
- cotnor1
- normsq
- lambda
- second