INFORMATION TO USERS

This dissertation was produced from a microfilm copy of the original document. While the most advanced technological means to photograph and reproduce this document have been used, the quality is heavily dependent upon the quality of the original submitted.

The following explanation of techniques is provided to help you understand markings or patterns which may appear on this reproduction.

1. The sign or “target” for pages apparently lacking from the document photographed is “Missing Page(s)”. If it was possible to obtain the missing page(s) or section, they are spliced into the film along with adjacent pages. This may have necessitated cutting thru an image and duplicating adjacent pages to insure you complete continuity.

2. When an image on the film is obliterated with a large round black mark, it is an indication that the photographer suspected that the copy may have moved during exposure and thus cause a blurred image. You will find a good image of the page in the adjacent frame.

3. When a map, drawing or chart, etc., was part of the material being photographed the photographer followed a definite method in “sectioning” the material. It is customary to begin photoing at the upper left hand corner of a large sheet and to continue photoing from left to right in equal sections with a small overlap. If necessary, sectioning is continued again – beginning below the first row and continuing on until complete.

4. The majority of users indicate that the textual content is of greatest value, however, a somewhat higher quality reproduction could be made from “photographs” if essential to the understanding of the dissertation. Silver prints of “photographs” may be ordered at additional charge by writing the Order Department, giving the catalog number, title, author and specific pages you wish reproduced.

University Microfilms
300 North Zeeb Road
Ann Arbor, Michigan 48106
A Xerox Education Company
KOENIG, David Mark, 1939-
OSUSIM: A MODULAR APPROACH TO DYNAMIC SIMULATION.

The Ohio State University, Ph.D., 1972
Engineering, chemical

University Microfilms, A XEROX Company, Ann Arbor, Michigan

THIS DISSERTATION HAS BEEN MICROFILMED EXACTLY AS RECEIVED.
OSUSIM: A MODULAR APPROACH TO DYNAMIC SIMULATION

DISSERTATION

Presented in Partial Fulfillment of the Requirements for the Degree Doctor of Philosophy in the Graduate School of The Ohio State University

By

David M. Koenig, B.S., M.S.

The Ohio State University 1972

Approved by

[Signature]
Adviser
Department of Chemical Engineering
PLEASE NOTE:

Some pages may have
indistinct print.
Filmed as received.

University Microfilms, A Xerox Education Company
ACKNOWLEDGEMENTS

I take great pleasure in acknowledging my indebtedness to the following four people for their friendship, assistance, advice and encouragement:

Edward J. Freeh
Harry C. Hershey
Wilda Mae Rousculp Koenig
Jacob R. Koenig
14 October 1939 . . . Born - Columbus, Ohio
1961 . . . B.S. University of Chicago
1965-1968 . . . Instructor, Department of Chemical Engineering, Catholic University, Washington, D.C.

PUBLICATIONS


FIELDS OF STUDY

Major Field: Chemical Engineering

Studies in Control Theory. Professor Edward J. Freeh

Studies in the Chemical Process Industries. Professor Thomas L. Sweeney

Studies in Applied Mathematics. Professor Stefan Drobot

Studies in Statistics. Professor Harry C. Hershey
TABLE OF CONTENTS

ACKNOWLEDGEMENTS ........................................ ii
VITA ........................................................ iii
LIST OF FIGURES ........................................... viii
LIST OF TABLES ............................................ xii
NOMENCLATURE AND ACRONYMS ............................. xiii

CHAPTER

I. INTRODUCTION ............................................ 1
II. COMPUTER ASSISTED SIMULATION ......................... 5
   A. THE ANALOG COMPUTER APPROACH .................... 5
   B. THE DIGITAL COMPUTER APPROACH ................... 8
      1. THE FORTRAN LANGUAGE ......................... 8
      2. DIGITAL-ANALOG SIMULATORS ..................... 12
      3. EQUATION ORIENTED LANGUAGES .................. 13
      4. MODULAR SIMULATION PROGRAMS ................... 16
         a. Early Work .................................. 21
         b. PACER and CHESS ............................ 24
         c. Comparison of First and Second
            Order Methods .............................. 30
         d. The CHEVRON System and FLOWSIM ........... 33
      5. DYNAMIC MODULAR SIMULATION PROGRAMS .......... 35
TABLE OF CONTENTS
(continued)

5. DYNAMIC MODULAR SIMULATION
PROGRAMS (cont'd)

a. REMUS ............................ 35
b. PRODYC ............................ 38
c. DYNSYS ............................ 40
d. DYFLO .............................. 44
C. SUMMARY ............................ 45

III. AN INTRODUCTION TO OSUSIM ............ 48
A. THE NODE CONCEPT ..................... 48
B. EXAMPLE NODES ......................... 52
C. THE ANALYSIS PACKAGE ................. 59
D. THE COMPUTE PACKAGE ................... 62
E. SOME FEATURES OF THE NODE SUBROUTINE . 66
F. SOME ASPECTS OF THE DATA STRUCTURE  .. 69
G. ARRANGEMENT OF INPUT DATA .......... 71
H. SOME FEATURES OF THE PREDICTOR-
CORRECTOR ALGORITHM ................ 74

IV. THE DATA STRUCTURE OF OSUSIM ......... 77
A. DEFINITION OF THE SECTORS IN IVEC ... 80
B. DEFINITION OF THE SECTORS IN PPVEC .. 86
TABLE OF CONTENTS  
(continued)  

<table>
<thead>
<tr>
<th>V. THE ANALYSIS PACKAGE OF OSUSIM</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>A. OVERALL LOGIC</td>
<td>93</td>
</tr>
<tr>
<td>B. INPUT DATA PROCESSING</td>
<td>99</td>
</tr>
<tr>
<td>C. PRELIMINARY INTERROGATION OF THE NODES IDENTIFICATION OF THE LDVS</td>
<td>110</td>
</tr>
<tr>
<td>D. COMPILATION OF THE GLOBAL IMPLICIT ALGEBRAIC SYSTEM</td>
<td>120</td>
</tr>
<tr>
<td>E. DECOUPLING THE SINGLE EQUATIONS FROM THE GLOBAL ALGEBRAIC SYSTEM</td>
<td>134</td>
</tr>
<tr>
<td>F. DISJOINT SUBSYSTEM DECOUPLING</td>
<td>142</td>
</tr>
<tr>
<td>G. UTILITY ROUTES</td>
<td>158</td>
</tr>
<tr>
<td>H. RECOMMENDATIONS</td>
<td>172</td>
</tr>
<tr>
<td>I. CONCLUDING REMARKS</td>
<td>177</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>VI. THE ANALYSIS OF THREE EXAMPLE PROBLEMS</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>A. A LINEAR MULTIPLE RECYCLE SYSTEM</td>
<td>178</td>
</tr>
<tr>
<td>B. A SOLIDS DRYING SYSTEM WITH FEEDFORWARD AND FEEDBACK CONTROL</td>
<td>197</td>
</tr>
<tr>
<td>C. A DOUBLE EFFECT EVAPORATOR</td>
<td>236</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>VII. THE COMPUTE PACKAGE OF OSUSIM</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>A. OVERALL LOGIC</td>
<td>268</td>
</tr>
<tr>
<td>B. DISPLAY OF SIMULATION RESULTS</td>
<td>276</td>
</tr>
<tr>
<td>C. SOLUTION OF THE GLOBAL ALGEBRAIC SYSTEM</td>
<td>277</td>
</tr>
<tr>
<td>D. SOLUTION OF THE ORDINARY DIFFERENTIAL EQUATIONS</td>
<td>294</td>
</tr>
<tr>
<td>TABLE OF CONTENTS</td>
<td>Page</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>VII.  THE COMPUTE PACKAGE OF OSUSIM (cont'd)</td>
<td></td>
</tr>
<tr>
<td>E.   EVALUATION OF THE LEAVS</td>
<td>306</td>
</tr>
<tr>
<td>F.   PLACING THE LOCAL VALUES IN THE</td>
<td>310</td>
</tr>
<tr>
<td>AUGMENTED JACOBIAN MATRIX</td>
<td></td>
</tr>
<tr>
<td>G.   TRANSFORMATION OF THE AUGMENTED JACOBIAN</td>
<td>315</td>
</tr>
<tr>
<td>MATRIX BY MEANS OF THE GAUSS-JORDAN</td>
<td></td>
</tr>
<tr>
<td>METHOD</td>
<td></td>
</tr>
<tr>
<td>H.   RESETTING THE GAVS AFTER THE SOLUTION OF</td>
<td>320</td>
</tr>
<tr>
<td>AN ALGEBRAIC SYSTEM</td>
<td></td>
</tr>
<tr>
<td>I.   DIAGNOSTIC OUTPUT</td>
<td>322</td>
</tr>
<tr>
<td>VIII. THE SIMULATION OF THE EXAMPLE PROBLEMS</td>
<td>324</td>
</tr>
<tr>
<td>A.   THE LINEAR MULTIPLE RECYCLE SYSTEM</td>
<td>324</td>
</tr>
<tr>
<td>1.   ARRANGEMENT OF THE INPUT DATA</td>
<td>324</td>
</tr>
<tr>
<td>2.   ANALYTICAL SOLUTION</td>
<td>328</td>
</tr>
<tr>
<td>3.   DISCUSSION OF THE SIMULATION RESULTS.</td>
<td>331</td>
</tr>
<tr>
<td>B.   THE SOLIDS DRYING SYSTEM WITH FEED­</td>
<td>334</td>
</tr>
<tr>
<td>FORWARD AND FEEDBACK CONTROL</td>
<td></td>
</tr>
<tr>
<td>C.   THE DOUBLE EFFECT EVAPORATOR SYSTEM</td>
<td>341</td>
</tr>
<tr>
<td>IX.  SUMMARY, CONCLUSIONS AND RECOMMENDATIONS</td>
<td>348</td>
</tr>
<tr>
<td>APPENDIX A: DERIVATION OF THE PREDICTOR­</td>
<td>352</td>
</tr>
<tr>
<td>CORRECTOR ALGORITHM</td>
<td></td>
</tr>
<tr>
<td>APPENDIX B: DERIVATION OF THE NEWTON-RAPHSON</td>
<td>356</td>
</tr>
<tr>
<td>ALGORITHM</td>
<td></td>
</tr>
<tr>
<td>APPENDIX C: MEMORY MAPS</td>
<td>358</td>
</tr>
<tr>
<td>APPENDIX D: PROGRAM LISTINGS</td>
<td>361</td>
</tr>
<tr>
<td>BIBLIOGRAPHY</td>
<td>403</td>
</tr>
</tbody>
</table>
LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Simple Recycle System</td>
<td>31</td>
</tr>
<tr>
<td>2</td>
<td>Simuland to Simulation</td>
<td>46</td>
</tr>
<tr>
<td>3</td>
<td>Tee Junction</td>
<td>49</td>
</tr>
<tr>
<td>4</td>
<td>The Water Tank</td>
<td>54</td>
</tr>
<tr>
<td>5</td>
<td>The Energy Exchanger</td>
<td>56</td>
</tr>
<tr>
<td>6</td>
<td>Sequence of Calculations in a Prediction of Correction phase</td>
<td>65</td>
</tr>
<tr>
<td>7</td>
<td>General NSR Flow Diagram</td>
<td>67</td>
</tr>
<tr>
<td>8</td>
<td>Illustration of the Pointer-Director Sector System</td>
<td>79</td>
</tr>
<tr>
<td>9</td>
<td>OSUSIM Data Structure</td>
<td>81</td>
</tr>
<tr>
<td>10</td>
<td>MAINP (for Analysis Package)</td>
<td>94</td>
</tr>
<tr>
<td>11</td>
<td>INTLZE</td>
<td>100</td>
</tr>
<tr>
<td>12</td>
<td>INP</td>
<td>107</td>
</tr>
<tr>
<td>13</td>
<td>SRN7</td>
<td>112</td>
</tr>
<tr>
<td>14</td>
<td>COMPX</td>
<td>115</td>
</tr>
<tr>
<td>15</td>
<td>Initial Pointings for IVEC(1), LASTV, and LYID</td>
<td>122</td>
</tr>
<tr>
<td>16</td>
<td>SETUPS</td>
<td>123</td>
</tr>
<tr>
<td>17</td>
<td>MKLST</td>
<td>128</td>
</tr>
<tr>
<td>18</td>
<td>DECUP</td>
<td>136</td>
</tr>
<tr>
<td>19</td>
<td>SBSYS</td>
<td>145</td>
</tr>
</tbody>
</table>
## LIST OF FIGURES
(continued)

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>Group List Sectors</td>
<td>152</td>
</tr>
<tr>
<td>21</td>
<td>Node Lists</td>
<td>156</td>
</tr>
<tr>
<td>22</td>
<td>CNLST</td>
<td>159</td>
</tr>
<tr>
<td>23</td>
<td>ADDL</td>
<td>161</td>
</tr>
<tr>
<td>24</td>
<td>GLOB</td>
<td>163</td>
</tr>
<tr>
<td>25</td>
<td>LOAD</td>
<td>165</td>
</tr>
<tr>
<td>26</td>
<td>TRANS</td>
<td>167</td>
</tr>
<tr>
<td>27</td>
<td>PERT</td>
<td>171</td>
</tr>
<tr>
<td>28</td>
<td>Occurrence Matrices for Example Problem</td>
<td>174</td>
</tr>
<tr>
<td>29</td>
<td>DECI</td>
<td>176</td>
</tr>
<tr>
<td>30</td>
<td>Example No. 1 Digraph</td>
<td>179</td>
</tr>
<tr>
<td>31</td>
<td>Input Data to Analysis Package, Example No. 1</td>
<td>183</td>
</tr>
<tr>
<td>32</td>
<td>Initial Occurrence Matrix</td>
<td>190</td>
</tr>
<tr>
<td>33</td>
<td>OM after Single Equation Decoupling</td>
<td>191</td>
</tr>
<tr>
<td>34</td>
<td>OM after Subsystem Decoupling</td>
<td>191</td>
</tr>
<tr>
<td>35</td>
<td>Integer Output from Analysis Package, Example No. 1</td>
<td>193</td>
</tr>
<tr>
<td>36</td>
<td>Real Output from Analysis Package, Example No.</td>
<td>195</td>
</tr>
<tr>
<td>37</td>
<td>Digraph for the Dryer System</td>
<td>198</td>
</tr>
<tr>
<td>38</td>
<td>Transport Lag</td>
<td>204</td>
</tr>
<tr>
<td>39</td>
<td>Source-Lag Example</td>
<td>205</td>
</tr>
<tr>
<td>40</td>
<td>Servo Controller Node</td>
<td>208</td>
</tr>
<tr>
<td>Figure</td>
<td>Description</td>
<td>Page</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
<td>------</td>
</tr>
<tr>
<td>41</td>
<td>Feedback Controller</td>
<td>211</td>
</tr>
<tr>
<td>42</td>
<td>Solids Dryer</td>
<td>213</td>
</tr>
<tr>
<td>43</td>
<td>N-Lump Approximation to the Solids Dryer</td>
<td>215</td>
</tr>
<tr>
<td>44</td>
<td>LAV and LDV Identification of Solids Dryer</td>
<td>222</td>
</tr>
<tr>
<td>45</td>
<td>Input Data to Analysis Package, Example No. 2</td>
<td>229</td>
</tr>
<tr>
<td>46</td>
<td>Punched Card Output from Analysis Package, Example No. 2</td>
<td>233</td>
</tr>
<tr>
<td>47</td>
<td>Digraph for Double Effect Evaporator System</td>
<td>237</td>
</tr>
<tr>
<td>48</td>
<td>Evaporator Module</td>
<td>240</td>
</tr>
<tr>
<td>49</td>
<td>Pump-Valve Node</td>
<td>249</td>
</tr>
<tr>
<td>50</td>
<td>Input Data to Analysis Package, Example No. 3</td>
<td>256</td>
</tr>
<tr>
<td>51</td>
<td>Occurrence Matrices for Example No. 3</td>
<td>262</td>
</tr>
<tr>
<td>52</td>
<td>Punched Output from Analysis Package, Example No. 3</td>
<td>263</td>
</tr>
<tr>
<td>53</td>
<td>MAINP (for Compute Package)</td>
<td>269</td>
</tr>
<tr>
<td>54</td>
<td>SAVE</td>
<td>278</td>
</tr>
<tr>
<td>55</td>
<td>ALGCMP</td>
<td>284</td>
</tr>
<tr>
<td>56</td>
<td>SRN8P1</td>
<td>296</td>
</tr>
<tr>
<td>57</td>
<td>EXPL</td>
<td>307</td>
</tr>
<tr>
<td>58</td>
<td>TRANS</td>
<td>311</td>
</tr>
<tr>
<td>59</td>
<td>GJ</td>
<td>316</td>
</tr>
<tr>
<td>Figure</td>
<td>Description</td>
<td>Page</td>
</tr>
<tr>
<td>--------</td>
<td>-----------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>60</td>
<td>RESET</td>
<td>321</td>
</tr>
<tr>
<td>61</td>
<td>Input Data to Compute Package, Example No. 1</td>
<td>325</td>
</tr>
<tr>
<td>62</td>
<td>Simplified Multiple Recycle System</td>
<td>329</td>
</tr>
<tr>
<td>63</td>
<td>Plotted Output for Example No. 1</td>
<td>332</td>
</tr>
<tr>
<td>64</td>
<td>Final Values of STRMV and FPAR Sectors</td>
<td>333</td>
</tr>
<tr>
<td>65</td>
<td>Input Data to Compute Package, Example No. 2</td>
<td>335</td>
</tr>
<tr>
<td>66</td>
<td>Plotted Output for Example No. 2</td>
<td>339</td>
</tr>
<tr>
<td>67</td>
<td>Input Data to Compute Package, Example No. 3</td>
<td>342</td>
</tr>
<tr>
<td>68</td>
<td>Plotted Output for Example No. 3</td>
<td>345</td>
</tr>
</tbody>
</table>
## LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Dependent Variables for the Solids Dryer</td>
<td>216</td>
</tr>
<tr>
<td>2</td>
<td>Variable Definition for the Solids Dryer</td>
<td>218</td>
</tr>
<tr>
<td>3</td>
<td>Energy Transfer Terms</td>
<td>223</td>
</tr>
<tr>
<td>4</td>
<td>Variable Definition for the Evaporator</td>
<td>241</td>
</tr>
<tr>
<td>5</td>
<td>Identification of LDVs and LAVs for the Evaporator</td>
<td>246</td>
</tr>
<tr>
<td>6</td>
<td>Identification of the Local Variables for the Pump-Valve</td>
<td>253</td>
</tr>
<tr>
<td>7</td>
<td>NAMELIST and LIST5 Elements</td>
<td>272</td>
</tr>
<tr>
<td>8</td>
<td>IPC Values and Meanings</td>
<td>274</td>
</tr>
<tr>
<td>9</td>
<td>Conditions for Subsystem Inversion when ICT3 is a Multiple of LIST5(1) and not less than LIST5(7)</td>
<td>283</td>
</tr>
<tr>
<td>10</td>
<td>SV subsector Structure</td>
<td>302</td>
</tr>
</tbody>
</table>
### Nomenclature and Acronyms

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AJM</td>
<td>augmented Jacobian matrix</td>
</tr>
<tr>
<td>ASFNC</td>
<td>analog single function node circuit</td>
</tr>
<tr>
<td>ASI</td>
<td>algebraic subsystem identification</td>
</tr>
<tr>
<td>DELT</td>
<td>time step length (see h)</td>
</tr>
<tr>
<td>DO</td>
<td>diagnostic output</td>
</tr>
<tr>
<td>$f_i$</td>
<td>a function of LAVs and LDVs that is equal to $dx_i/dt$</td>
</tr>
<tr>
<td>$g_i$</td>
<td>a function of LAVs and LDVs that is equal to zero</td>
</tr>
<tr>
<td>$G_i$</td>
<td>a function that when set equal to zero is the $i$th GAE</td>
</tr>
<tr>
<td>GAE</td>
<td>global algebraic equation</td>
</tr>
<tr>
<td>GAV</td>
<td>global algebraic variable</td>
</tr>
<tr>
<td>GDV</td>
<td>global differential variable</td>
</tr>
<tr>
<td>$h$</td>
<td>time step length (see DELT)</td>
</tr>
<tr>
<td>ICTE</td>
<td>number of GAEs</td>
</tr>
<tr>
<td>ICTV</td>
<td>number of GAVs</td>
</tr>
<tr>
<td>ICT3</td>
<td>number of time steps after a disturbance or initialization</td>
</tr>
<tr>
<td>IVBAC</td>
<td>initial value, bounds and code</td>
</tr>
<tr>
<td>JM</td>
<td>Jacobian matrix</td>
</tr>
<tr>
<td>KCO</td>
<td>number of singly decoupled GAEs</td>
</tr>
<tr>
<td>LAV</td>
<td>local algebraic variable</td>
</tr>
<tr>
<td>Abbreviation</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>-------------</td>
</tr>
<tr>
<td>LDE</td>
<td>local differential equation</td>
</tr>
<tr>
<td>LDV</td>
<td>local differential variable</td>
</tr>
<tr>
<td>LEAE</td>
<td>local explicit algebraic equation</td>
</tr>
<tr>
<td>LEAV</td>
<td>local explicit algebraic variable</td>
</tr>
<tr>
<td>LIAE</td>
<td>local implicit algebraic equation</td>
</tr>
<tr>
<td>MSP</td>
<td>modular simulation program</td>
</tr>
<tr>
<td>NEUB</td>
<td>number of nodes in digraph</td>
</tr>
<tr>
<td>NG</td>
<td>number of disjoint algebraic subsystems</td>
</tr>
<tr>
<td>NLA\textsuperscript{V}</td>
<td>number of LAVs per node</td>
</tr>
<tr>
<td>NLD\textsuperscript{E}</td>
<td>number of LDEs per node</td>
</tr>
<tr>
<td>NLE\textsuperscript{AV}</td>
<td>number of LEAVs per node</td>
</tr>
<tr>
<td>NL\textsuperscript{IA}\textsuperscript{E}</td>
<td>number of LIAEs per node</td>
</tr>
<tr>
<td>NSR</td>
<td>node subroutine</td>
</tr>
<tr>
<td>NSUB</td>
<td>number of streams in digraph</td>
</tr>
<tr>
<td>ODE</td>
<td>ordinary differential equation</td>
</tr>
<tr>
<td>OM</td>
<td>occurrence matrix</td>
</tr>
<tr>
<td>SED</td>
<td>single equation decoupling</td>
</tr>
<tr>
<td>SFN</td>
<td>single function node</td>
</tr>
<tr>
<td>(w_i)</td>
<td>a function of LAVs and LDEs that equals the (i)th LAV if that LAV is a LEAV</td>
</tr>
<tr>
<td>(x_i)</td>
<td>the (i)th LDV</td>
</tr>
<tr>
<td>(Y_i)</td>
<td>the (i)th LAV</td>
</tr>
<tr>
<td>(Y_i)</td>
<td>the (i)th GAV</td>
</tr>
</tbody>
</table>
CHAPTER I
INTRODUCTION

This thesis deals with dynamic modular mathematical simulation and begins by reviewing and introducing some terminology. According to Webster's Collegiate Dictionary, the work simulate is a verb meaning "to assume the appearance of, without the reality; to feign". Associated with every simulation is a model whose behavior simulates the behavior of some object (the simuland). In non-mathematical simulation, the model is usually a physical device. For example, a RLC electrical circuit can be a model for a mechanical system consisting of springs, masses and dashpots. On the other hand an ordinary differential equation can be a mathematical model for a RLC circuit.

To effect a mathematical simulation one must first describe the simuland in mathematical terms, i.e., formulate a mathematical model. In those cases where the simuland has structure, the global model can be modular in that it may consist of a collection of interrelated local models or modules. For example, a simuland such as a chemical processing plant can be said to have structure since it is made up of interconnected
unit operations such as pumps, tanks, distillation towers, etc. A global model for such a plant, if it is modular, would consist of a collection of local models or modules, one for a tank, one for a pump, one for a distillation tower, etc. In some cases where there is more than one of a particular kind of unit, a local model (or module) could be used more than once.

Second, the modeler must prepare the model for solution by determining parameter values and by prescribing initial and boundary conditions. Again, if the simuland has structure and the models are modular, part of the model preparation will consist of prescribing the topology, or how the modules are interconnected.

Finally, the modeler must make the model "go" or he must "solve the model". The exact meaning of this phase will depend on the type of mathematical model and on the purpose of the simulation. In general it will mean to determine values of the dependent variables as functions of the initial and boundary conditions, the parameters and the independent variables (if there are any). If the simulation is dynamic then one of the independent variables will be time. Throughout what follows it will be assumed that the parameter values are available. Unfortunately, the parameter determination problem is often
one of the most difficult and interesting steps in a simulation.

In formulating a model there are three questions that must be asked. First, what mathematical concepts are available for model description and what mathematical tools are available for model solution? The answer to this question would preclude the formulation of a model consisting of partial differential equations when a FORTRAN Gauss-Jordan routine is the only model solution tool available.

Second, how accurate or how close to reality should the model be? Usually as the model becomes more accurate, the larger must be the repertoire of mathematical concepts and tools.

Third, on what dependent and independent variables should the model be based? The steady state model of a continuous stirred-tank reactor might be based only on one dependent variable, say the outlet product concentration, whereas the dynamic model for a packed column reactor could be based on two dependent variables, concentration and temperature, and two independent variables, time and axial position. It should be noted that in all the models considered here, the domain of definition for the independent variable is the set of real numbers, i.e., discrete system simulation is not considered.
The modeler can be assisted in carrying out the two steps of formulating and solving a model in many ways. At the lowest level, the modeler will be assisted by pencils, paper, slide rule, and perhaps hand calculators. At higher levels the modeler can make use of electronic services such as analog and digital computers. In some cases the modeler may wish to develop a specialized program or language that will either simplify the process of using the computer's power or increase the power and flexibility of the computer. In developing such a program or language, there are two questions that should be posed. First, in terms of what mathematical concepts will the models that the program has to deal with be written? Second, if the program is to assist in the solution of the model, what mathematical tools will it employ?

A digital computer program named OSUSIM has been developed that assists the modeler in carrying out dynamic modular mathematical simulation and in the following summary of computer assisted simulation it will be shown where OSUSIM fits in. The remaining parts of this thesis will show the logical foundation on which OSUSIM is based and will illustrate its use with three example problems.
CHAPTER II
COMPUTER ASSISTED SIMULATION

This chapter will show how both the analog and digital computer can be used by the modeler to assist in simulation. The application of the analog computer to systems of ordinary differential equations (ODEs) is discussed first. This is followed by a brief description of the FORTRAN language and its application to the numerical solution of ODEs. Digital-simulators and equation oriented languages and their application to the same problem are discussed next. The modular concept is then introduced and features of several of the steady state modular simulation programs are presented. Finally a survey is given of five currently available programs that apply the modular concept to dynamic or unsteady state simulation.

A. The Analog Computer Approach

The analog computer is essentially a collection of high-gain, negative feedback electronic amplifiers, resistors and capacitors that can be wired together such that voltages at certain points in the electrical network behave just as the dependent variables in the
mathematical model would behave during model solution. In effect the modeler is making an electrical model (or analog) of his mathematical model. This can be accomplished because the amplifiers, resistors and capacitors can be prewired to form subcircuits or single function nodes (SFNs) that, when embedded in another circuit of the modeler's design, can perform voltage integration, voltage addition, voltage multiplication by a constant, and special operations such as voltage-voltage multiplication and function generation.

To be more specific, if y is the output voltage, u is the time, and \( x_1, \ldots, x_n \) are the n input voltages, SFNs can be described as

\[
y(u) = T(u, x_1, \ldots, x_n, p_1, \ldots, p_m)
\]  

(1)

where \( T \) is a linear or nonlinear operator and the \( p_i \) are parameters. For example, if the SFN performs integration we can write

\[
y(u) = y(0) + \int_0^u \sum_{i=1}^n p_i x_i(v) \, dv
\]

where the \( p_i \) are coefficients determined by values of resistors in the SFN subcircuit. Likewise a function
multiplication SFN would be described as

\[ y(u) = x_1(u) \times x_2(u) \]

Hence to solve a model the modeler devises and constructs, on a patchboard, a circuit (the analog single function node circuit or ASFNC) consisting of these SFNs connected by wires that he assumed to be resistance-free. Then at various points in the ASFNC he observes voltages that will correspond to dependent variables in his mathematical model. This ASFNC is then the simulation assisting program.

Because the SFNs can perform integration, the analog computer is best used to solve models consisting of systems of ordinary differential equations. Explicit non-linear algebraic expressions of the type given in Eqn. 1 (with \( m=0, n=1 \) and the \( u \) dependence supressed) can also be handled by means of function generation SFNs wherein the dependence of \( y \) on \( x \) is approximated by a piecewise linear representation. Circuits can be devised for solving models described by partial differential equations but usually at a cost of greatly increased ASFNC complexity. In general as the model becomes more detailed and involved so does the ASFNC. This means more work for the modeler since he must devise, construct and test the ASFNC. Also most analog computers have a fixed limit on the number of SFNs of each type that can
be included in the circuit.

Another constraint derives from the permitted voltage range peculiar to the analog computer being used. Hence the modeler must predict the value ranges of the model's dependent variables and then if necessary scale his model equations before constructing the ASFNC. This is called magnitude scaling. A second type of scaling, time scaling, is necessary if the dependent variables significantly fluctuate during short periods of computing time which will correspond to the independent variable. In this case computing time must be scaled down (or up) relative to model time so that changes in the voltages corresponding to the dependent variables can be followed by the computer circuitry.

The problems associated with scaling and the ASFNC construction and testing are deterrents to the use of the analog computer as a device to assist in carrying out a simulation.

B. The Digital Computer Approach

1. The FORTRAN Language

A FORTRAN program consists of a sequenced list of statements written in terms of symbols resembling those in the English language and in algebra. There are four general types of statements but it is necessary to consider
only two. The arithmetic statement has the form of an explicit algebraic statement where a variable name is "equated" to an algebraic expression. For example, the statement

12    A=B1*A - X**2

would instruct the computer to square the current value of the variable named X, subtract the result from the product of the current values of the variables named B1 and A, and assign the value of the difference to the variable named A. The "12" is the statement number which is necessary only if this statement is referred to by some other statement. Unlike algebra the equals sign says "calculate the value of the expression to my right and assign that value to the variable on my left".

The expression-calculation-assignment operations are carried out in the order of appearance of the arithmetic statements. Transfer statements can be used to alter the order in which the arithmetic statements are executed. Unconditional transfer statements have the form

GO TO n

where n is the number of some other statement in the program where the computer will go to get its next instructions, or to which the flow of control will be transferred. The conditional transfer statement has the form
IF(algebraic expression) i, j, k
where the flow of control will be transferred to statements i, j, or k, if the algebraic expression is negative, zero, or positive, respectively.

There is much more to the FORTRAN language than what has been presented here, however even with these three special types of statements the modeler has an awesomely powerful tool at his disposal. To use the power and flexibility of the FORTRAN statements, one must break the model solving process into a recurring finite sequence of algebraic steps. This is, of course, just what one does when solving problems by numerical methods. Therefore, FORTRAN can be used to implement numerical model solutions.

Many numerical algorithms have been pre-programed in a general setting and are available as subroutine or stand alone programs. For example there are subroutines available in the IBM 360 Scientific Subroutine Package (19) for the numerical solution of systems of differential equations of the form

\[ \frac{dx_i}{du} = f_i(u,x_1,\ldots,x_n) \quad i = 1,\ldots,n \quad 0 \leq u \leq U \]

To use this routine as an aid in model solving, the modeler would need to write statements calculating the \( f_i \) values and a few statements invoking the subroutine.
Another example is the LEANS package (24) which is capable of solving ordinary or partial differential equations if the user supplies three of his own routines which define the initial conditions, the equation format, and the output format.

The approaches to model solving using the analog and digital computer therefore are in rather sharp contrast. Integration via the analog computer's SFN concept is simply an input/output operation and if there are several integrator SFNs in an ASFNC the integrations are performed in parallel. Integration via FORTRAN requires detailed programming according to a chosen numerical algorithm and the operations are executed in a sequential manner. According to Schiesser and Franks (8) prewritten integration subroutines such as the one mentioned above are "generally of little help in simulation studies because of their inflexible structure". However the extremely wide value range of the FORTRAN variables makes scaling problems virtually non-existent and the greater reproducibility and reliability coupled with the logic and memory capabilities make the digital computer with FORTRAN a very attractive aid in simulation.
2. Digital-Analog Simulators

In order to combine the virtues of the analog computer and digital computer with FORTRAN, languages like MIDAS (15) and IBM 1130 CSMP (20) were developed for the digital computer. These programs (sometimes called digital-analog simulators) allow the use of the SFN concept but, instead of constructing an actual ASFNC on a patchboard, the modeler transforms the topology of the ASFNC (the way the various SFNs are interconnected) into a series of coded statements that form part of the input. For each SFN in the ASFNC there is a statement containing the SFN type (summer or integrator, etc.) and number (a number arbitrarily designated by the modeler for identification purposes when he designs the ASFNC) and the numbers of other SFNs connected to it. Also, for those SFNs that require it, there are statements giving parameter values. With this basic information the digital computer will simulate the analog operations in such a manner that the user need not be concerned with the details of the numerical algorithm being used. Although the calculations are being carried out sequentially, the operations appear to the user to be parallel. Because everything is done digitally there are no scaling problems and the memory and logic capabilities can be used to
provide more accurate nonlinear function generation and true time delays.

3. Equation Orientated Languages

Use of either the analog computer or the digital-analog simulators required that an ASFNC be devised after the model has been formulated. Since the behavior of the SFNs in the ASFNC is described mathematically in the same terms that the model is, there is motivation to develop a language based directly on the equations in the model rather than on the ASFNC topology. This is the basis for languages such as MIMIC (31) and IBM 360 CSMP (18), which are best suited for models described in terms of ordinary differential equations and explicit algebraic expressions. To apply IBM 360 CSMP to a model, one writes "structure statements" which with a few exceptions are very similar to FORTRAN arithmetic statements; in fact most of the explicit algebraic expressions in a model can be coded as a FORTRAN-like statement of the form:

variable name = algebraic expression.

The equals sign in the statement has the same meaning as in FORTRAN, however the variable name on the left of the equals sign should not appear in the algebraic expression unless an IMPL function (to be discussed below) is used.
The differential equations in a model can also be coded as a structure statement. For example

\[
dx/du = y + 2x^2 \quad x(0) = 2.5
\]

could be coded as

\[
X = \text{INTGL}(2.5, Y + 2.*X**2)
\]

where INTGL represents but one of about fifty so called mathematical functions available to the programmer.

The general form of this kind of structure statement is

\[
\text{output variable name} = \text{function name (parameter names or values, input variable names or values)}
\]

where the input variable name can sometimes be replaced by an algebraic expression. In effect, the form is the same as Eq. 1 which described the SFN, i.e., the structure statements can be written in any order and the CSMP processor can be requested to sort the statements automatically, insuring a correct execution sequence. Centralized integration means that all integrator outputs are in effect computed simultaneously at the end of each iteration cycle.

Even this brief glance at CSMP should impress the reader that a wide range of models can be solved using this language, however there are many other features of CSMP that greatly extend its flexibility and
applicability. One of these deserves discussion here if only because it deals with a problem that OSUSIM treats in a different way, namely the problem of implicit algebraic functional relationships. In general, a single implicit relation has the form

\[ g(x, y, z) = 0 \]  

(2)

where there is no restriction on how \( g \) depends on \( x, y, \) or \( z \). A special case of the form in Equation 2 occurs when one of the variables can be partially solved for, that is,

\[ z = w(x, y, z) \]

This semi-explicit equation can be handled by CSMP if the IMPL function is used. For example, the equation

\[ z = ae^{-t} + b\sin(z) = w(t, z) \]

would be coded in CSMP statements as

\[
\begin{align*}
  z_1 &= \text{IMPL}(z_0, \text{ERR}, z_2) \\
  z_2 &= a\exp(-t) + b\sin(z_1)
\end{align*}
\]

where \( z_0 \) is the initial guess for the value of \( z \). The CSMP processor will supervise the calculations so that the value of \( z \) is determined by successive substitution, that is,
\[ z_{n+1} = w(t, z_n) \quad n = 0, 1, 2, \ldots \]

(z specified as first parameter in IMPL argument list)

Unfortunately convergence will be obtained only if

\[ \left| \frac{\partial w}{\partial z} \right| < 1 \]

during the iterations (16). Requiring that an implicit relation be partially solvable and that the resulting expression yield convergence can be quite a handicap in some problems, especially those dealing with the simulation of hydraulic phenomena.

4. Modular Simulation Programs

It is often convenient and profitable to look at the simuland as a network of subsystems each of which can be modeled on a stand-alone basis. Chemical plants consisting of reactors, extractors, distillation units and controllers connected by material or information carrying streams have, as was pointed out earlier, structure and are amenable to this concept. In situations like this a modeling approach that tries to treat the simuland as a whole, that is, that is globally equation oriented, can be unattractive because of its unwieldiness. Important details of a subsystem model
can become obscured by the large number of equations representing the overall model. Conversely, requiring the modeler, at the same time, to be aware of the microscopic structure of the model might hinder the study of the model's macroscopic properties. Finally, any change in the topology of the simuland would require an uncomfortable amount of variable redefinition and equation modification.

Instead, a localized approach is needed wherein a self-contained model, defined in terms of local variables, is developed for each subsystem. In this case an overall model defined in terms of global variables for the system as a whole would then be a suitably interconnected and integrated collection of the subsystem models. Hence, rather than constructing an ASFNC based on all of the model equations as required by the analog computer and digital-analog simulators, a digraph (a collection of nodes connected by directed, information carrying streams) is constructed based on the topology of the simuland. Since the nodes are modeled on a local basis, there is no concern, at least initially, with the global mathematical features of the total model. On the contrary, the primary objective is to correctly fit the local self-contained models together.
In the digraph for a chemical plant, nodes might represent processing units like mixers, splitters, reactors, or controllers, and the streams can represent material-carrying pipelines or signal-transmitting electrical lines. Unlike real streams there is, by definition, no change in the quality of the quantities as they pass between nodes. If, for example, there is a significant pipeline fluid flow resistance to be modeled, the effect may be lumped into a special resistance node or it may be taken into account at the node representing the subsystem from which the real stream exits or at the node representing the subsystem to which the stream flows.

There are several programs extant that assist in system simulation by keying on the structure or topology of the system rather than the model describing equations. These programs usually consist of executive routines augmented by a library of subroutines (or modules) written for each kind of node in the digraph. The use of these so-called modular simulation programs (MSPs) is similar to that of the digital simulators. Each node and stream in the digraph is described in the input to the executive portion of the MSP by encoding for each node (1) its digraph number, (2) the number of each stream connected to it, and (3) the name or number of the
node subroutine (NSR) that describes the node's local behavior. This family of information will be referred to as the digraph description. Numerical values of parameters and variables associated with the nodes and streams are also part of the input as are miscellaneous flags and codes. If there are modules available for every type of node in the digraph, the modeler can carry out a simulation without concerning himself with numerical algorithm details or even the local details of the node models. Being able to provide this somewhat detached global viewpoint is the main feature of the MSPs. Also, changes in the simuland topology can be accomplished by simply modifying the input statements—there is no reprogramming necessary.

The MSPs differ in the ease with which the modeler can construct new modules for nodes not already modeled by the available library subroutines. Some MSPs such as PACER (39), FLOWTRAN (5), and FLOWSIM (4) have an extensive library of modules and their authors claim that the user need not understand the mathematics and programming involved in the module's construction but merely be able to construct the digraph and encode the necessary input statements. On the other hand, programs like OSUSIM and DYNSYS (1) do not have extensive module libraries and therefore the modeler must understand the
overall philosophy of the program and the detailed logic involved in the NSR construction. Requiring the modeler to develop his own NSRs may be an advantage for several reasons. First, the results of a simulation can be more easily explained and defended. Second, each simuland will have peculiar features, some significant, some not, that are not included in the library NSRs, especially when the modeler is interested in a dynamic simulation.

If an extensive simulation is to be carried out using the modular approach it is visualized that there will be four phases. In phase one, a first version of the digraph is conceived. In phase two, the preliminary versions of the modules are written and tested on a stand-alone basis. In phase three, the simulation is carried out and the results are studied. In phase four, modifications are made to the digraph structure, modules are modified and perhaps new modules are written. Phase three is then repeated and followed perhaps by phase four, and so on until the modeler obtains the desired results. It is in this fourth phase that the flexibility and generality of programs like OSUSIM and DYNSYS should be appreciated.

The MSPs also differ in the type of problem that they attempt to solve. The first MSPs dealt exclusively
with the steady state simulation of chemical plants. Later, programs appeared that attempted to simulate the dynamic behavior of chemical plants. Although OSUSIM is in the latter category it is not necessarily oriented toward chemical plants but toward any physical system that is conveniently modeled by a digraph.

a. Early Work

One of the first to consider the chemical plant as a network of nodes connected by streams was the Russian M. F. Nagiev (29) who in 1957 published an article in Chemical Engineering Progress on a method of solving material balances. The quantity \( a_{ij} \) was defined as the weight fraction of the component of interest in the stream leaving the \( i \)th node and flowing to the \( j \)th node. Furthermore, if \( g_m(k) \) is the total charge of the component in the \( m \)th node (or "reactor" as the author referred to it) after the \( k \)th iteration, then Nagiev wrote

\[
g_m(k) = \sum_{i=1}^{M} a_{im} g_i^{(k-1)}, \quad m=1,\ldots, M
\]

This equation was his starting point for the solution of several reasonably practical examples. Although Nagiev did not refer to a computer program and made no mention of key words like "topology" or "module," his \( a_{ij} \) matrix
in effect gave a topological description of the system whose steady state material balance behavior he simulated. In 1962 Rosen (36) extended this approach to handle more than one component and also wrote a digital computer program to carry out the calculations.

In the same year, 1962, Rubin (37) published an article in the CEP symposium series proposing a method of determining an optimal node calculation sequence for systems where there are recycle paths. The topology of the system was described by an incidence matrix $M = (m_{ij})$ where $m_{ij}$ is negative if stream $j$ leaves node $i$ and positive if the stream enters the node; otherwise $m_{ij}$ is zero. The nodes were all defined by steady state input/output relations such that if the quantities in the inlet streams to the node were known the outlet stream quantities could be calculated directly. These input/output relations could come from material and/or energy balances.

One method of solving the equations that describe the system would be to proceed node to node starting at the inlet. However, one would soon come to a node where all of the inlet quantities are not known because of a recycle loop. At this point values could be guessed and the calculations continued. Subsequent iterations
should in principle improve these guessed values until convergence is obtained. Based on the information in the incidence matrix and on $q_j$, the number of calculable quantities associated with the $j$th stream, Rubin's algorithm yields a not necessarily unique sequence of nodes that minimize the number of quantities whose values must be guessed in order for calculations to proceed.

Rubin also presented an algorithm to solve the system of equations generated by his analysis. It is of interest to summarize the highlights of this method because it is very similar to the conventional multi-dimensional Newton-Raphson method (see Appendix B) which is used in OSUSIM. Let the $n$-dimensional system of equations be described as

$$f_i(x_1, \ldots, x_n) = 0, \quad i = 1, \ldots, n$$

Let $x_j^{(k)}$, $j=1,\ldots, n$, $k=0,1,\ldots, n$, be the $k$th preliminary estimate of the $j$th unknown variable. If each of the $n$ preliminary estimates, $x_j^{(k)}$, $k=1,\ldots, n$ are close to $x_j^{(0)}$ then a first order expansion can be written

$$f_i^{(k)} = f_i^{(0)} + \sum_{j=1}^{n} \frac{\partial f_i}{\partial x_j} \bigg|_{x(o)} (x_j^{(k)} - x_j^{(o)}) \quad (3)$$

$$i=1,\ldots, n, \quad k=1,\ldots, n$$

where the zeroth estimate is considered to be the best.
In matrix notation Equation 3 becomes

\[
\{ \Delta f^{(k)} \} = \left( \frac{\partial f}{\partial x} \right)^T \{ \Delta x^{(k)} \}
\]

\( k-1, \ldots, n \)

but by partitioning, all of these \( n \) preliminary expansions can be written in one equation as

\[
(\Delta f^{(1)} \ldots \Delta f^{(n)}) = \left( \frac{\partial f}{\partial x} \right) (\Delta x^{(1)} \ldots \Delta x^{(n)})
\]

(4)

Thus having made \( n+1 \) preliminary estimates, Equation 4 can be solved for the partial derivatives which then can be used to calculate the result of the first iteration, \( x_j^{(n+1)}, \ j=1, \ldots, n, \) from

\[
\{ \Delta x^{(n+1)} \} = -\left( \frac{\partial f}{\partial x} \right) \{ f(0) \}
\]

This new vector can be used to update the partial derivatives via Equation 4. Iterations can continue in this manner until convergence is obtained. This method avoids the \( n^2 \) perturbations necessary in the conventional Newton-Raphson method to evaluate \( \partial f/\partial x \) at each iteration at the cost of making \( n+1 \) preliminary estimates.

b. PACER and CHESS

Although Nagiev, Rosen and Rubin used topological concepts either implicitly or explicitly there was still
no use of the modularity concept; instead, each chemical plant model was equation oriented from the global viewpoint and no attempt was made to write self contained local models for each type of node. One of the first to do this was Shannon who reported on his work in 1963 in an article in Chemical Engineering Education (38) and in a thesis of his student, Mosler, in 1964 (27). However, the first versions of the program, called PACER, which implemented his ideas appear to have been written in 1961 (27).

PACER consists essentially of executive routines and a collection of subroutines, one for each type of node. As with the approach of Rubin, the nodes are defined in terms of equations based on mass and energy balances relating outlet stream quantities directly to inlet stream quantities.

The topological information on the system is contained in a process matrix having elements $p_{ij}$ where the $i$th row contains the stream numbers of the $I$ entering streams and the $L$ leaving streams associated with the $i$th node. By convention, the first element in each row is equal to the row index $i$ if there is a node in the digraph with that number; the next $I$ positions in the $i$th row contain the entering stream numbers signed positively; the next $L$ positions contain the leaving
stream numbers signed negatively. The contents of this matrix make up part of the input to PACER.

The calculations proceed as follows. First, a primary pass is made through all the nodes and, for those nodes having completely known input streams, the appropriate node subroutine is called and the outlet stream quantities are computed. If at the end of this pass the values of quantities associated with all the streams have been completely determined the computations cease. In general, because of recycle paths this will not be the case. Consequently, if, during the primary pass, the values of any of the stream quantities were determined, a second primary pass will be made since now some of the nodes previously having unknown input streams may now be calculable. This type of pass is repeated until no more streams change status from unknown to known.

At this point, if there are still unknown streams, the analysis for iterative calculations commences. First, a list of the unknown streams is made. One of those streams is chosen and by scanning the process matrix it is determined whether or not a closed path exists that includes the unknown stream and that will allow recalculation of the stream's quantities should initial values be guessed and node to node calculations
around the path be carried out. If no such path can be found, the same analysis is attempted for the second unknown stream on the list. If there is no single stream that will satisfy the closed path requirement, the whole analysis is repeated for combinations of two unknown streams. Finally, if this fails, the analysis is attempted for combinations of three unknown streams. According to Mosler (27) it is unlikely that there will be systems that will not satisfy these tests. Hence, assume that a closed path is found. In this case an ordered list of the nodes on this path is made and iterative calculations commence. When convergence is obtained, the whole process is repeated. This cycle of calculations will continue until at the end of a primary pass all streams are known.

Although this algorithm is not particularly elegant and convergence can be very slow, PACER has enjoyed wide acceptance and several groups have copied many of its features in the process of developing their own modular steady state simulation programs. The most notable of these is the program called CHESS (28) as developed by Rudi Motard in 1968 at the University of Houston which even follows PACER in its variable naming scheme. The only major algorithmic difference between the two programs is that CHESS requires the user to supply the node
lists for the iterative calculations.

The data structure of CHESS is very similar to PACER and, because it appears to be a point of departure for many of the dynamic simulation programs including OSUSIM, it will be summarized here. Like PACER, the topology of the digraph which represents the chemical plant is described in a process matrix called KPM. The values of the stream variables are stored in two stream matrices SEXTSV and SINTSV. For the ith stream, the ith row of SEXTSV contains values of the extensive variables such as overall flow rate and component molar flow rates in a fixed order. Similarly, the intensive variables such as temperature and pressure are stored in SINTSV.

Values of parameters appearing in the node-describing equations are stored in a matrix named EQPAR in which the ith row is reserved for the parameters associated with the ith node. The node describing equations are not written in terms of elements of SINTSV or SEXTSV. This is because before each node calculation, values of the stream variables are transferred from SINTSV and SEXTSV to several local matrices defined so that the node subroutine writer need not known the actual stream indices (and hence the row number in SINTSV or SEXTSV) of the streams connected to that node. As a result, he can write the equations in terms of the variables associated with, say, the first or second input stream as seen
by that node instead of having to fetch the digraph indices of the first or second input stream.

For example, in a mixer mode where the outlet flow is the sum of the two inlet flows the appropriate equation would be written as

\[ \text{SOMOLE}(1) = \text{SIMOLE}(1) + \text{SIMOLE}(2) \]

and, as mentioned above, before program control is transferred to the mixer subroutine the elements of the SOMOLE, SIMOLE and other local vectors are filled from the appropriate rows of the SEXTSV and SINTSV matrices. This feature supposedly simplifies node subroutine coding at the cost of increased storage and execution time. It should be noted that this value transfer is not selective, i.e., all of the stream values are transferred from the global stream matrices SINTSV and SEXTSV to the local matrices independent of whether or not the associated variable appears in the node describing equations.

In later discussions it will be shown that the dynamic modular simulation program DYNSYS uses a similar data structure but does not make use of this preliminary global to local transfer of values. Further it will be shown that OSUSIM, which uses a slightly different data structure, does make the preliminary transfer
but on a selective basis.

Since the node-describing energy and mass balances, when written in terms of enthalpies, mole fractions and flow rates, are independent of the nature of the chemical species which make up the streams, PACER and CHESS include general subroutines that can supply enthalpies, $K_{eq}$ values and other thermodynamic quantities for any species in a selected indexed list. Consequently the node subroutines can be written on a more fundamental basis and whenever an enthalpy, $K_{eq}$, or a specific volume is needed in a calculation a simple call, giving the component index, to the thermodynamic property routine is all that is necessary. In effect the properties routines, separate and self-contained, is just another extension of the modular concept.

c. Comparison of First and Second Order Methods

Consider the system depicted in Figure 1 where the circled numbers are stream indices and boxed numbers are node indices. Assume that there is only one variable associated with each stream and that each node defines an input/output relation just as has been assumed in the previous discussion. Therefore the system is described by the following equations where $x_i$ is the $i$th stream
FIGURE 1. A SIMPLE RECYCLE SYSTEM
variable and $g_j$ is the function describing the $j$th node:

$$
\begin{align*}
  x_2 &= g_1(x_1, x_6) \\
  x_3 &= g_2(x_2) \\
  x_4 &= g_2(x_2) \\
  x_6 &= g_3(x_4)
\end{align*}
$$

Note that because of the restrictive node definition, stream variable $x_4$ is independent of stream variable $x_3$ because both are output stream variables and therefore dependent only on $x_2$. Solving for $x_6$ yields

$$
x_6 = g_3(g_2(g_1(x_1, x_6))) = G(x_6)
$$

where $x_1$ can be considered a system input and therefore known. In effect, PACER and CHESS would solve for $x_6$ iteratively, either by

$$
\begin{align*}
  x_2^{(k)} &= g_1(x_1, x_6^{(k-1)}) \\
  x_4^{(k)} &= g_2(x_2^{(k)}) \\
  x_6^{(k)} &= g_3(x_4^{(k)})
\end{align*}
$$

or by

$$
x_6^{(k)} = g_3(g_2(g_1(x_1, x_6^{(k-1)}))) = G(x_6^{(k-1)})
$$

$k=1,2,...$

where $x_1^{(0)}$ denotes the initial guess of the $i$th variable. As mentioned before, in the section on CSMP, convergence of this algorithm requires that

$$
\left| \frac{dG(x_6)}{dx_6} \right| = \left| \frac{\partial g_3}{\partial A} \cdot \frac{\partial g_2}{\partial B} \cdot \frac{\partial g_1}{\partial C} \right| < 1
$$
If the system is a chemical plant and the $x_i$ are mass fractions, then convergence is assured since mass cannot be created in any node. In any case if convergence does occur it can be shown (9) that the rate of convergence is first order, i.e., where $e^{(k)}$ is the error after the $k$th step.

To increase the rate of convergence, the one dimensional Newton-Raphson method (see appendix B) can be applied to the function

$$F(x_6) = x_6 - G(x_6)$$

yielding an algorithm of the following form

$$x_6^{(k+1)} = x_6^{(k)} - (F'(x_6^{(k)}))^{-1}F(x_6^{(k)})$$

which, should it converge, will do so quadratically:

$$e^{(k+1)} = b^{(k)} * (e^{(k)})^2, \quad |b^{(k)}| < 1$$

This second order convergence rate is obtained at the cost of having to evaluate the derivative which is not a trivial problem in multidimensional systems. Note that the method proposed by Rubin and mentioned above attempts to partially overcome this derivative-evaluation problem.

d. The Chevron System and FLOWSIM

One of the first to combine the three concepts of modularity, topology, and second order algorithms, were
Ravicz and Norman of California Research Corp., who published their work in 1964 in Chemical Engineering Progress (35). The topology of the system is described in a manner similar to PACER however, based on this information, an association matrix $A$ with elements $a_{ij}$ is constructed where $a_{ij}$ is non-zero if there is a stream flowing from node $i$ to node $j$. Using an algorithm developed by the authors, successively higher powers of $A$ are examined and recycle nests are identified.

The calculations commence as in PACER at the first node and proceed until a node is reached having unknown input streams. If this node is a member of a recycle nest and if all the input streams to this nest are known, the Newton-Raphson method is invoked and the equations describing the nodes in the nest are solved iteratively and simultaneously as well. If this node is not a nest member then it is skipped and calculations are resumed at the next node and continued until all of the nodes in the digraph have been processed. At this point if there are still unknown streams the whole calculation scheme is repeated. These ideas form the basis for a program presently distributed by the Chevron Research Co.

Four months after the appearance of the Ravicz-Norman paper, L. M. Naphtali of Brooklyn Polytechnic
Institute published a paper (30) also in Chemical Engineering Progress which apparently went one step further and relied almost exclusively on the Newton-Raphson method to solve the equations describing the digraph. Unfortunately, in contrast to the article by Ravicz and Norman, this article was not at all clearly written; consequently it is difficult to deduce how the calculations were carried out. However the ideas supposedly set forth in this article have evolved into the commercially available program named FLOWSIM (4).

During the 1960s a number of other programs appeared based in principle either on the PACER/CHESS first order successive substitution approach or the Ravicz-Norman second order Newton-Raphson approach. Most of these programs were reviewed by Evans (3) in 1968.

5. Dynamical Modular Simulation Programs

In this section, four dynamic MSPs that are currently available will be introduced in the following chapter and the remainder of this thesis will deal with its logical structure and with its application to three example problems.

a. REMUS (Routine for Executive Multi-Unit Simulation)

Written by Peter G. Ham, a student of Warren D. Seider at the University of Pennsylvania, REMUS is
documented in Ham's doctoral dissertation (10) and in several manuals (11, 12, 13, 14). The primary goal of REMUS is to determine the steady state operating conditions of a chemical plant by solving the equations describing its transient behavior. The solution of the equations provides a simulation of a start up and the final values give the steady state conditions. Each stream is defined in a fixed format by its temperature, pressure, volumetric flow rate, and component molar flow rates. As with the steady state MSPs, subroutines (or modules) are written for each type of chemical processing unit that represents a node in the digraph. Each node can be described by ODEs and explicit algebraic equations similar in concept to the input-output relations defining the nodes in the steady state MSP format.

The integrations in REMUS are carried out and the explicit algebraic expressions are evaluated locally on a node-to-node basis during a once-per-time-step pass through the nodes in the digraph. The order of the nodes in the calculation sequence must be specified by the modeler in the input data and should follow the flow of material as much as possible. Where there are recycle nests, at least one node during a given pass will have inputs that are not up to date consequently there is a
time lag of one time step implicit in all recycle streams. This may be a tolerable situation if the time step is small enough or if the primary interest is in the final steady state values and not the transient values.

The ODEs can be solved by any one of a variety of fourth order methods each with an adjustable time step feature. In cases where the effective time constants of the ODEs describing a node are significantly larger than the others, the user can stipulate, by changing an input parameter called SKIP, that this node not be updated every time stsp.

The program accepts the input data in a free format therefore it is difficult to deduce the data structure. However, from the nature of the input information and from the discussion of the above mentioned example problem, it appears that the data structure is quite similar to PACER/CHESS. REMUS, being chemical plant-oriented, includes a thermodynamic properties package but details on the scope of this module are not given in the User's Manual.

The results of the simulation can be displayed in an impressive variety of ways: (1) numerical tables where the values of up to 11 system variables can be given for each time step, (2) scaled plots of one or more
variables versus time, and (3) cross plots of selected pairs of process variables.

b. PRODYC

PRODYC (21), written by Don M. Ingels and Rudi L. Motard, can be described as a preprocessor for the IBM CSMP language in the sense that given input data similar in content to that presented to REMUS, it will generate CSMP statements which when executed in a subsequent step will produce the desired simulation. The streams are defined similarly to CHESS; in fact CHESS can be used to compute initial steady state conditions for PRODYC.

The node models are defined, as in REMUS, in terms of ODEs and explicit algebraic equations. However the associated node subroutines, though written in Fortran, must generate the CSMP statements that make up the output of PRODYC instead of simply calculating values of the node describing functions. Therefore while only 48 CSMP statements are generated by PRODYC in preparation for the simulation of a single mixer, the mixer node subroutine is written in no less than 107 Fortran statements and, most importantly, examination of the Fortran listing gives no clue as to what the statements attempt to accomplish. Since there is no guide for the construction of the Fortran node describing subroutines, it must be concluded that the authors do not expect the user to
frequently require new modules.

In the process of compiling the CSMP statements from
the input data, "tearing" (17) is used to simplify the
equations associated with recycle loops. Like REMUS,
PRODYC is chemical plant oriented and, if the user so
desires, thermodynamic properties can be supplied to the
node subroutines by the same module that CHESS uses.
However, as the authors point out, this choice can be
very time consuming especially if values are evaluated
at every time step. To alleviate the time consumption
problem, provisions are made for less frequent and/or
approximate schemes for evaluating the thermodynamic
properties at the cost of decreased accuracy.

The data inputs to the system take slightly dif-
f erent forms depending on whether or not CHESS is used to
calculate initial steady state values. In the former
case the input data is presented to CHESS in a free for-
mat. CHESS produces the steady state values which are
then augmented by values of certain parameters and
presented as input to PRODYC. A CSMP program is compiled
during the second or PRODYC step.

Finally, the simulation can be effected by executing
the output of PRODYC which is a CSMP program. Like any
CSMP program, values of certain parameters must be
described in the input. If CHESS is not used then the
user must present the initial conditions, topological description and parameters to PRODYC in a fixed format. The form of the output of the third or execution step is the same as the output from any IBM CSMP execution, namely a graphical-numerical display of selected quantities versus time.

c. DYNSYS

As can be gathered from the previous discussion of REMUS and PRODYC, the documentation of these programs is not complete enough to justify extensive discussion, let alone criticism. This is not the case with DYNSYS, developed by A. I. Johnson's group at McMaster University, and the following discussion is based on both a detailed manual (1) and a journal article (32).

The data structure is very similar to that of CHESS. The process matrix, defined exactly as the one is CHESS, describes the topology and the order of the appearance of the row indices in the process matrix determines the calculation sequence. Unlike CHESS, the stream variables are stored in a three dimensional matrix S(I,J,K) where I equals one or two depending on whether the value is associated with the present or previous time step. These two values are needed because of the way the predictor-corrector method for solving the ODEs is implemented.
The variable J denotes the stream number and K indexes the stream quantity. The authors suggest the following format for the positions indexed by K: stream number, stream flag, total mass flow, temperature, pressure, and the concentrations of up to three components. This is a "suggested" format because the authors realistically do not expect the user to be able to simulate his peculiar systems without having to write his own node subroutines (NSRs) and hence, if necessary, redefine the stream quantities.

Like CHESS, values of parameters occurring in node describing equations are stored in a matrix named EP(I,J) where I is the node index and J is the storage position index. This matrix is also used for storage of values of dependent variables associated with the node (rather than the streams) such as the volume of a mixing tank. Should a node require more storage than available in the EP matrix, sectors in an auxiliary parameter vector can be reserved.

The nodes are defined in terms of explicit algebraic equations and ordinary differential equations which are solved by a localized application of the Adams-Moulton-Shell predictor-corrector method. In this localized scheme a prediction pass is made through the nodes during which the predicted values of the differential
variables are determined and values of the explicit algebraic variables are calculated. The same types of calculations are carried out on the correction pass. Because new values are calculated at each node during these passes, it is necessary to carefully specify the order of the nodes in the calculation sequence as determined by the process matrix. The authors propose the simple rule that nodes described by ODEs be serviced first and those described by explicit algebraic equations last. In cases where a node is described by both types of equations the sequence should match the flow of material. When the node is described by an ODE and an algebraic expression of the form

\[ \frac{dx}{dt} = f(x, y) \]
\[ y = g(x) \]

where \( x, y, f, \) and \( g \) may be vectors, the NSR must fetch the appropriate values of the variables involved from the stream matrix \( S \) or the parameter matrix \( EP \), calculate the values of \( f(x, y) \) and \( g(x) \), and call a routine that will return the predicted or corrected values depending on what kind of pass is being carried out. During the prediction step the returned predicted values must be stored locally in the node's row of the \( EP \) matrix for use in the subsequent correction pass.
Hence the node subroutine is written to (1) fetch the values from the global stream and parameter matrices using structural information in the process matrix, (2) calculate values of the functions needed by the prediction-correction algorithm, (3) call the subroutine that implements the algorithm, and (4) if necessary, store predicted values needed in the subsequent correction step. Therefore the NSR is algorithm, data structure, and type-of-pass dependent.

Note that values of the explicit algebraic variables are calculated on both the prediction and correction passes but there is no provision for handling recycle nests or local iterative calculations; the latter being necessary if one of the node's algebraic expressions is "semi" explicit, i.e., of the form

\[ y = g(x, y) \]

a form which can be handled by OSUSIM.

Thermodynamic properties are calculated locally in the NSR but the process is generalized by use of the physical properties matrix \( PP(I,J) \) where the \( i \)th row is used to store necessary physical constants for the \( i \)th chemical component. This way the NSR can be specie independent.

The input data from which the process matrix, the
S matrix, the EP matrix, and other system matrices are constructed is presented to the executive routine in fixed format with a structure similar to that used in OSUSIM. The results of the simulation can be displayed numerically at selected time intervals.

d. DYFLO

Written by R.G.E. Franks, DYFLO (7), is the most recent of the dynamic MSPs and differs from the others in several respects. The main program, which must be custom-built for each simulation, consists of a loop in which a node subroutine (NSR) is called for each node once per time step. Before the loop begins, subroutine INIT, also custom-built, is called and through data statements and calls to special physical property routines, the stream variables are initialized. At the beginning of the loop, subroutine RECIPY, written by the user, is called to set up whatever forcing is necessary and calculate preliminary quantities such as heat fluxes. At the end of the loop after the NSRs have been called, subroutine INIT which solves the algebraic and differential equations defined in the NSRs, is called.

The topology of the system is specified by inserting the appropriate stream numbers in the arguments of the NSR call statement in the main program. Similarly the
values of node parameters are communicated to the NSRs through the call statements. In fact all the information that is presented to the other MSPs in the form of input data is programmed in DYFLO either through call statements or DATA statements. Like REMUS, DYFLO is oriented exclusively toward chemical processing systems in that the streams are defined in terms of a flow, temperature, enthalpy, pressure, and the compositions of 20 components.

The NSR routines apparently define input/output relations just as they do in DYNSYS and REMUS, and the NSR library is claimed by the author to be the most extensive of all the dynamic MSPs but, as has been pointed out earlier, an extensive library of NSRs is not necessarily an advantage.

C. Summary

The flow diagram in figure 2 attempts to show the alternate paths from the simuland to the simulation. Boxes with rounded edges indicate a starting state or a result of a procedure or operation while boxes with sharp edges enclose an operation or a procedure. The dotted line connecting box two with box three indicates that, instead of making explicit use of the system topology, the modeler, usually unconsciously, incorporates the topology when he defines the variables that
FIGURE 2. SIMULAND TO SIMULATION
occur in the model describing equations.

On the other hand in the modular approach the subsystem models are constructed on a completely localized basis and the system topology is combined with the collection of subsystem models only when the latter have been completed. After this combination the only means to a simulation must include the MSPs; in fact the operations carried out in box five are usually carried out by the MSP. This is in contrast with the parallel branch which leads to a model consisting of coupled equations (box six) rather than coupled modules (box eight).

At this point the modeler has three choices as to how the model is to be solved. Furthermore if the modeler chooses to construct an analog representation of the model he has two ways of solving the resulting ASFNC so effectively there are four procedures available for solving the model represented by box six. The so-called "equation oriented programs" in box nine include the programs that implement the equation oriented languages such as IBM 360 CSMP and special purpose packages such as LEANS.
CHAPTER III
AN INTRODUCTION TO OSUSIM

Since the remaining sections of this thesis will describe OSUSIM in detail, this section will present in summary some features of the program for the sake of comparison with the other dynamic MSPs.

A. The Node Concept

Unlike any of the other MSPs, the node concept in OSUSIM does not define input/output relations per se but rather it defines functional relationships between local variables associated with: (1) the streams flowing into or out of the node and (2) the node itself. This definition includes the conventional input/output relations but also covers those situations where variables such as stream pressure or velocity are not completely defined by equations describing one node. This problem would occur when a node model is defined in terms of variables that, depending on the nature of other nodes in the digraph, can be either forcing or responsive.

As an example, consider a node model for a tee junction shown in figure 3. Assuming no transient effects, three algebraic equations, a mass balance and two mechanical energy balances, can be written in terms of the six
FIGURE 3. TEE JUNCTION

$p_1, v_1$  

$p_2, v_2$  

$p_3, v_3$
variables. If these equations are to be used in explicit form it is not clear which three variables are to be solved for. If stream three exhausts to the atmosphere then \( p_3 \) would be fixed, so in general it would not be feasible to solve for \( p_3 \); it will be assumed to be either given or calculated by a downstream node or condition. On the other hand, the flow rate of one of the inlet streams, say \( v_1 \), might be fixed or determined by some upstream node, so in general it would not be feasible to solve for \( v_1 \) or \( v_2 \). This analysis leaves \( p_1, p_2, \) and \( v_3 \) to be solved for, and suggests a general rule for nodes that model hydraulic effects: solve for inlet pressures and outlet velocities.

This rule will require node-to-node or global iteration in many cases, especially when there are recycle loops, and therein lies the dilemma: convergence simply cannot be guaranteed. This was discovered in early work on OSUSIM where such a rule was implemented. Part of the problem results from the square root expression occurring in those cases where the mechanical energy balance equation is solved for velocity. This situation also produces an implicit relation since the velocity occurs in both the kinetic energy term and the friction loss term.

These kinds of problems can be handled by OSUSIM because, in addition to ODEs, the node models can be described in terms of completely implicit, completely explicit, and
partly explicit (or semi-explicit) algebraic relations as will be seen in the following discussion.

Each node's behavior can be described in terms of (a) \textit{local differential equations} (LDEs) of the type

\[ \frac{dx_i}{dt} = f_i(x_1, \ldots, y_1, \ldots, t) \quad i = 1, \ldots, NLDE \]

where the \( x_i \) are \textit{local differential variables} (LDVs) of which there are \( NLDE \) per node and the \( y_i \) are \textit{local algebraic variables} (LAVs) of which there are \( NLAV \) per node,

(b) \textit{local implicit algebraic equations} (LIAEs) of the form

\[ g_j(x_1, \ldots, y_1, \ldots, t) = 0 \quad j = 1, \ldots, NLIAE \]

and (c) \textit{local explicit algebraic equations} (LEAEs) each of which defines a \textit{local explicit algebraic variable} (LEAV). Two classes of LEAVs (of which there are \( NLEAV \) per node) are recognized. A class A LEAV can depend functionally on itself and other class A LEAVs and LDVs associated with the node:

\[ Y_k = w_i(x_1, \ldots, x_{NLDE}, y_k, \text{ other class A LEAVs}) \]

Note that since the \( k \)th LAV is defined by the \( i \)th LEAE, \( \text{the subscripts } i \text{ and } k \text{ are not necessarily equal. A class B LEAV can depend functionally on itself as well as any other LAV or LDV:} \]

\[ Y_k = w_i(x_1, \ldots, x_{NLDE}, y_1, \ldots, y_{NLAV}) \]
These two classes are defined as such because of the different kinds of computation passes (to be discussed below) made through the nodes.

In summary, then, a node can be described in terms of NLDE local differential equations (LDEs), NLIAE local implicit algebraic equations (LIAEs), and NLEAV local explicit algebraic equations (LEAEs). There are therefore NLDE + NLIAE + NLEAV equations describing a given node and these equations are defined in terms of NLDE local differential variables (LDVs) and NLAV local algebraic variables (LAVs), where NLEAV of the local algebraic variables may occur explicitly.

The LDVs and LAVs can also be classified either as external or internal depending on whether or not the variable's value must be accessible to other nodes. As an example of an external variable, consider the temperature of a stream leaving node A; this temperature may be node B's inlet stream temperature and therefore must be accessible to both nodes. The external variables correspond to the stream variables mentioned in connection with the steady state MSPs.

B. Example Nodes

To illustrate the various kinds of variables that can appear in a node description, two example problems will
be discussed in this section.

1. A Water Tank

Consider the water tank depicted in figure 4. An overall mass balance gives

\[ a_1 v_1 - a_3 v_3 = A \frac{dh}{dt} \]

A mechanical energy balance between points two and three gives

\[ \frac{p_k}{\rho} + \frac{g}{g_c} h = \frac{v_3^2}{2g_c} + \rho(v_3) \frac{v_3^2}{2g_c} \frac{L}{D_3} + \frac{p_3}{\rho} \]

where \( L/D_3 \) is the equivalent length in pipe diameters associated with the outlet stream. The height of the water, \( h \), is recognized as the LDV, \( x_1 \), and is classified as an internal variable. The pressures and velocities, \( p_1, v_1, p_3, \) and \( v_3 \), are classified as external variables and will be designated as the LAVs \( y_1 \) through \( y_4 \) respectively. The node parameters are \( a_1, a_2, A, \) and \( L/D_3 \). Hence the node describing equations can be rewritten as

\[ \frac{dx_1}{dt} = \frac{a_1 y_2 - a_2 y_4}{A} = f_1 \]

\[ g_1 = \frac{y_1 - y_3}{\rho} + \frac{g x_1}{g_c} - \frac{(y_4)^2}{2g_c} (1 + f(y_4) \frac{L}{D_3}) = 0 \]

and it is seen that \( NLD=1, NLIAE=1, NLEAV=0, \) and \( NLAV=4 \).
\( a_1 = \text{internal cross sectional area of pipe} \)

\( A = \text{cross sectional area of tank} \)

**Figure 4. THE WATER TANK**
2. An Energy Exchanger

Consider the mixing tank shown in figure 5 where air and solids exchange energy. Assuming perfect mixing, a mass and energy balance over the solids gives

\[ M_1 - M_2 = \frac{dM}{du} \]  \hspace{1cm} (5)

\[ C(M_1 T_1 - M_2 T_2) - hM(T_2 - t_4) = C \frac{d(MT_2)}{du} \]  \hspace{1cm} (6)

where \( u \) is the time and where the energy transfer term is assumed to be proportional to the holdup, \( M \). Combining equations (5) and (6) yields equation (7) which replaces equation (6):

\[ CM_1 (T_1 - T_2) - hM(T_2 - t_4) = CM \frac{dT_2}{du} \]  \hspace{1cm} (7)

Assuming that the response of the air is at least an order of magnitude quicker than that of the solids, a mass and energy balance over the air gives

\[ \frac{d}{du} \]

\[ m_4 = m_3 \]  \hspace{1cm} (8)

\[ \dot{m}_3 C(t_3 - t_4) + hM(t_2 - t_4) = 0 \]  \hspace{1cm} (9)

Finally assume that the outlet solids flow rate depends linearly on the holdup as follows:

\[ \dot{M}_2 = \frac{M}{R} \]

The temperature \( T_2 \) is recognized as an external LDV,
FIGURE 5. THE ENERGY EXCHANGER
say \( x_1 \), while the holdup, \( M \), is an internal LDV, say \( x_2 \).

The seven flow rates and temperatures \( \dot{M}_1, \dot{T}_1, \dot{M}_2, \dot{M}_3, \dot{m}_3, \dot{t}_3, \dot{m}_4 \), and \( t_4 \) can be identified as the LAVs \( y_1 \) through \( y_7 \), respectively—all of which are external variables. Equation (9) can be solved for \( t_4 \) hence the outlet air flow rate and temperature, \( y_6 \) and \( y_7 \), and the exiting solids flow rate \( y_4 \) can be treated as LEAVs. The node parameters are the heat capacities \( c, C \), the heat transfer coefficient \( h \), and the proportionality constant, \( R \). Hence the node describing equations can be rewritten as

\[
\begin{align*}
\frac{dx_1}{du} &= C y_1 (y_2 - x_1) - h x_2 (x_1 - y_7) = f_1 \\
\frac{dx_2}{du} &= y_1 - x_2 = f_2 \\
y_3 &= \frac{x_2}{R} = w_1 \\
y_6 &= y_4 = w_2 \\
y_7 &= \frac{y_4 c y_5 + h x_2 x_1}{y_4 c + h x_2} = w_3
\end{align*}
\]

where it is seen that \( \text{NLDE} = 2, \text{NLIAE} = 0, \text{NLEAV} = 3, \) and \( \text{NLAV} = 7 \).

It should be noted that \( y_7 (t_4) \) depends on the external variables \( y_4 (\dot{m}_3) \) and \( y_5 (t_3) \), therefore, if the modeler is certain that these two variables do not occur in implicit
algebraic expressions in other nodes, \( y \) can be identified as a class A LEAV. In questionable cases the modeler might want to leave Equation (9) in implicit form as was done with the mechanical energy balance equation in the previous example.

In an alternate approach it might not have been assumed that the air is at steady state and Equations (8) and (9) could have been replaced by differential equations of the form

\[
\frac{d}{du} (\rho V) = \dot{m}_3 - \dot{m}_4 \quad (8a)
\]

\[
\frac{d}{du} (\rho V t_4) = \dot{m}_3 c t_3 - \dot{m}_4 c t_4 + h M (T_2 - t_4) \quad (9a)
\]

where \( \rho \) is the air density and \( V \) is the volume of the air space in the energy exchanger. Experience with the solids dryer discussed in Chapters V and VII, which is similar in concept to the energy exchanger, showed that the effective time constants associated with Equations (8a) and (9a) would be an order of magnitude smaller than those associated with Equations (5) and (7). Therefore, the systems of differential equations defined by Equations (5), (7), and (8a), and (9a) would be "stiff" (2). Stiff systems are difficult to solve numerically hence to avoid this problem the steady state assumption for the air was made.
The loss in rigor may be tolerable if the modeler is primarily interested in the solids dynamics and not the air dynamics.

C. The Analysis Package

OSUSIM consists of two programs: the analysis package and the compute package. The function of the analysis package is to digest the input data which contains information in a fixed format on (1) the digraph description, (2) the initial values, codes and bounds for each dependent variable, (3) the values of parameters occurring in the node describing equations, and (4) the values of flags and miscellaneous parameters. In the process of digesting this data the analysis package must assemble a global system of ODEs and implicit algebraic equations from the collection of initially unrelated local variables and equations associated with the nodes. A one-to-one relation is developed between the LDVs and the global differential variables (GDVs) while one or more of the "free" (not held constant or not occurring explicitly) LAVs may be matched with one global algebraic variable (GAV). This multiple association might exist when a flow rate of a stream connecting two nodes is represented by one LAV from the first node and another LAV from the second node. If these LAVs are free then they will both be matched to one GAV.
The structure of the global implicit algebraic system of equations is best described by the occurrence matrix (OM) wherein a nonzero value in the \( i \)th row and the \( j \)th column indicates that the \( j \)th GAV occurs in the \( i \)th global algebraic equation (GAE) which is matched with one of the LIAEs. The Jacobian matrix of the global implicit algebraic system, which can be constructed from the results of a pass through those nodes for which NLAIE is greater than zero, acts as the OM. Based on an examination of the OM, the analysis package permutes the GAE and GAV order such that the OM is transformed into block diagonal form.

If there are implicit GAEs that depend on but one GAV ("singly decoupled equations") then the first submatrix will be lower triangular with nonzero diagonal elements. A nonzero element, in, say, the \( i \)th row and \( J \)th column represents the occurrence of the \( J \)th GAV in the \( I \)th GAE, where \( I \) is greater than \( J \). Since the solution of the \( J \)th GAE provides the value of the \( J \)th GAV, this variable can be considered as a known parameter in the \( I \)th GAE. If some of the remaining GAEs and GAVs are not coupled with others, then during the OM transformation these separate sets of GAEs are segregated into disjoint subsystems. For each subsystem there will be a disjoint submatrix in the transformed OM.
The output of the analysis package consists of the values of three one-dimensional vectors. The first vector, IPOIN, whose elements are integers, contains pointers to sectors in the other two vectors plus information on the dimensions of the global implicit algebraic system. The second vector, IVEC, also inter-valued, is made up of sectors of varying lengths which can contain pointers to sub-sectors in itself or the third vector and can act as lists containing information on the nature and structure of the system. The third vector, FPVEC, whose elements are real numbers, (a) stores the value, bounds and code associated with each of the algebraic and differential variables, (b) stores the values of parameters, and (c) supplies working storage to be used during the numerical solution of the global differential and algebraic equations.

Of the working storage locations, which are not part of the analysis package output, part is taken up by the Jacobian matrix. By using pointers, only the submatrices of the Jacobian matrix associated with the disjoint sub-systems are stored and for the singly decoupled equations only the diagonal elements of the Jacobian matrix are stored. The length of IPOIN is fixed while that of IVEC and FPVEC varies depending on the size and complexity of the digraph. Unlike IPOIN, the vectors IVEC and FPVEC are
not commoned but appear as arguments in almost all of the subroutines that make up the analysis package. Therefore a length change can be accomplished simply by changing the Fortran dimension statement in the eight line main program. Thus the user can avoid wasting space on small problems while being bounded only by the size of the computer memory for complex problems.

D. The Compute Package

The compute package supervises the numerical solution of the global differential and algebraic equations based on the information contained in IPOIN, IVEC, and FPVEC, which are accepted as input. At a given point in time, the sequence of computations is as follows.

First, a pass is made through those nodes for which NLDE is greater than zero and values of the $f_i$ are determined leading to a calculation of predicted values of the LDVs.

Second, a pass is made through those nodes for which NLEAV is greater than zero and values of the class A LEAVs are computed. At each node on this pass the values of the class A LEAVs are recomputed locally until values from two consecutive calculations agree within a specified tolerance. These iterations are completed before control is passed to the next node.
Third, each of the implicit singly decoupled GAEs is transferred to the appropriate node and the values of the function and derivative are determined. Application of the Newton-Raphson algorithm then yields an estimate of the solution. This process is repeated for the equation of interest until convergence is obtained.

Fourth, each of the disjoint subsystems is solved in turn by the multidimensional Newton-Raphson method. For each subsystem, control is transferred to the appropriate nodes and then values of the functions and the derivatives are determined and placed in the Jacobian matrix. Application of the Newton-Raphson algorithm then yields an estimate of the solution vector. This process is repeated for the subsystem of interest until convergence is obtained.

As can be seen by the wording of the foregoing sentence, the solution process for a singly decoupled equation is analogous to that for a disjoint subsystem. The inversion of the Jacobian, required in the Newton-Raphson method, is done in-place; hence on subsequent iterations when a new inverse may not be needed, the equations can be solved by matrix multiplication allowing for a savings in time. The frequency of inversion can be set by the user.

Finally, now that the LDVs, class A LEAVs, and implicitly occurring LAVs have been solved for, a pass is
made through those nodes for which NLEAV is non-zero and the class B LEAVs are evaluated in the same iterative manner as were the class A LEAVs. After these five steps of computation, predicted values of the LDVs and GAVs have been determined, and the so-called prediction phase has been completed. Next, these five steps are repeated, with appropriate changes, and corrected values of these variables are determined. This is the so-called correction phase. The independent variable \( t \) is then incremented and the whole two phase sequence is repeated. In case the initial values of the GAV are not accurately known, the user has the option of prescribing that steps two through five be carried out at time zero. Figure 6 shows the sequence of calculations associated with either the prediction or correction phase.

The order of these different kinds of passes defines a hierarchy of dependent variables according to the types of equations in which they may appear. The LDVs are considered the forcing variables and, being evaluated first, can appear in any node describing equation. The class A LEAVs are, by definition, solely dependent on the LDVs and therefore can also appear in any node describing equation. The implicit LAVs are also considered to be dependent on the LDVs but, being the third group solved for, can not
FIGURE 6. SEQUENCE OF CALCULATIONS IN A PREDICTION OR CORRECTION PHASE
appear in any of the class A LEAV defining equations. Finally, the class B LEAVs, being the last group solved for, can not occur in the class A LEAV defining equations or in the implicit equations.

This hierarchy is slightly complicated if there is a class B LEAV associated with one node, say node B, that depends on the value of a class B LEAV associated with another node, say node A. In this case a proper calculation sequence would require that node A should have a lower node index than node B. The flexibility afforded by this hierarchy of variable definitions provides the user with an opportunity to design his NSR with minimal loss of rigor when compared to the restrictions imposed by the other dynamic MSPs discussed in this section.

During the simulation, the values of $t$ and up to six variables can be displayed numerically or graphically at selected time intervals. At the end of the simulation, the final values of a portion of FPVEC can, at the option of the user, be punched on cards. To continue the simulation of a subsequent run, these cards replace the initial values of FPVEC in the input to the compute package.

E. Some Features of the Node Subroutine

As shown in Figure 7, the NSR is written to respond to six different kinds of calls as determined by the values of ICALL and INST. On the first kind of call (when ICALL
ENTRY

ICALL: 2

fetch or calculate preliminary values

set NLIAE, NLA, NLEAV, IDER

call COMFX for LDVs

RETURN

CALL MKLST for LAVs

RETURN

INST

1

calculate the \( f_1 \)

RETURN

2

calculate class A LEAVs

RETURN

3

calculate class B LEAVs

RETURN

4

calculate the \( g_1 \)

calculate the partial derivatives

ICODE: 1

RETURN

FIGURE 7. GENERAL NSR FLOW DIAGRAM
equals one), supervised by the analysis package, values are assigned to NLIAE, NLAV, NLDE, NLEAV, and IDER which is a flag indicating whether or not values of the NLIAE*NLAV partial derivatives will be supplied by the NSR. If they are not, then the calling routine estimates them numerically by perturbation. Also on this call the routine COMPX, which matches the LDVs with the GDVs, is called.

On the second kind of call (when ICALL equals two) the routine MKLST, which matches the LAVs with the GAVs, is called. On the third kind of call (when ICALL equals three) the NSR produces numerical values of different functions depending on the value of INST. If INST equals one, then values of the $f_i$ are calculated. Next, if INST is two, class A LEAVs are calculated. Third, if INST is three, values of the $g_i$ are calculated and, if this is a NSR for which IDER had been set equal to one and if ICODE is equal to one, the partial derivatives are calculated. The variable named ICODE is a flag set by the calling routine which is unity whenever the Jacobian is to be evaluated. Finally if INST is four, the class B LEAVs are calculated.

The NSR writer need not concern himself with how the correct values are assigned to the LAVs and LDVs; this is taken care of by the routine supervising the call. It
should be noted that certain nodes such as the transport lag and the source node (to be discussed later) need to know whether a prediction or a correction phase is being carried out. This information is contained in the flag IPC which is minus one during the very first prediction phase, zero during all subsequent prediction phases, and plus one during a correction phase.

F. Some Aspects of the Data Structure

Associated with each external variable, whether it be a LDV, LAV or LEAV, is a quadruple consisting of the variable's value, lower and upper bound, and code. Associated with each stream is a subsector of FPVEC. These subsectors, located continuously, make up a sector called STRMV and provide storage for the quadruples associated with the external variables. Associated with each internal variable, if it is a LAV or a LEAV, is a quadruple consisting of the variable's value, lower and upper bound, and code. If the internal variable is a LDV, it is associated with a triple consisting of the variable's value, lower and upper bound, i.e., internal LDVs have no code.

Associated with each node are two subsectors: one in IVEC and one in FPVEC. The subsectors located in IVEC provide storage for any integer valued parameters or flags used in the NSR and are located continuously, forming a
sector called IPAR. The subsectors located in FPVED provide storage for any real valued parameters used by the NSR and storage for the triples or quadruples associated with the internal variables. These subsectors are also located continuously and constitute a sector called FPAR.

The initial value of the code is supplied by the user in the input data to the analysis package and is used during the formulation of the implicit global algebraic system. If the initial code is minus one, then the variable is not to be matched with a GAV. Hence external LDVs should be coded minus one as should those external or internal LAVs that are to remain fixed in value during the simulation since in these two cases a matching obviously should not take place. If the initial code is \(-(N + 1)\), then the analysis package treats the variable as a LEAV associated with node N and no matching takes place. Finally, if the initial code is zero then this variable is "free" and is to be matched with a GAV. Each time a free LAV is matched with a GAV the zero code is replaced by the index of the GAV.

Because pointers are used, each subsector is only as large as the number of locations needed. This is to be contrasted with other MSPs such as CHESS and DYNSYS that use the ith row of a matrix of fixed dimensions to store the parameters for the ith node. Should the ith node
happen not to need parameters or should there not be an
ith node in the digraph then a whole row is wasted. FPAR
and STRMV are the first two sectors in FPVEC. The remain­
der of FPVEC is divided into sectors that are needed for
working storage (such as the sector that the submatrices
of the Jacobian occupy) during the simulation. Hence it
is only these first two sectors of FPVEC that are trans­
ferred between the analysis package and the compute package.

G. Arrangement of Input Data

Assuming that the input device is a card reader, the
structure of the input data set can be described as fol­
lows. The first group of cards contains values to be
assigned to LIST5, a general integer valued utility vec­
tor, and DUT, a real valued utility vector. This group is
followed by one card that contains the largest stream num­
er and the largest node number.

Next comes the data that will compose the KPM sector
of IVEC which contains the digraph description. For each
node there are at least two cards. The first card con­
tains the node's number or index, the number of streams
connected to it, and the alphameric name of the NSR that
describes this node's behavior. On the following card(s)
are the numbers or indices of the streams that are con­
nected with the node.
Usually the order of the stream indices is important. For example, if the node is a mixer then the first two stream numbers refer to the entering streams and the third refers to the exiting stream. Unlike CHESS or DYNSYS the sign of the stream number is unimportant. This is because there is no need to identify output streams as such for calculation purposes. As the cards for each node are read in, appropriate space in IVEC is reserved. For this reason the cards for the lowest numbered node should appear first, the next highest numbered node second, etc.

The fourth group of cards contains data that will compose the FPAR sector in FPVEC. For each node there are again at least two cards. The first card contains the node's index and the length of the subsector of FPAR associated with this node. On the following card(s) is the data that will go into the subsector. As mentioned above this data consists of parameter values and initial values, bounds and codes of all internal variables associated with this node.

The arrangement of this information within the subsector is dictated by the NSR writer. Usually the first set of locations has been used for parameter values, the next 3*NLDE locations for the initial value, lower and upper bounds for the LDVs and, if there happen to be N internal LAVs, the last 4*N locations for their initial
value, lower and upper bounds and code. As with the KPM sector the lowest numbered node's subsector data should appear first, with the higher indexed subsector data following in numerical order. This structure is used again for the data composing the IPAR sector which follows the FPAR sector data.

The final group of cards contains the stream data which will compose the STRMV sector of FPVEC. As with the nodes there are at least two cards for every stream in the digraph. On the first card is the stream number and the length of the subsector of STRMV associated with this stream. On the following card(s) are the data that will make up the subsector which consists of the initial value, bounds and code (IVBAC) of the external variables specified by the NSR writer. For example, he may decide that for material carrying streams the first four locations be used for the IVBAC of the mass flow rate, the second four for the IVBAC of the stream concentration, the third four for the IVBAC of the stream temperature, etc. For those control lines that are treated as streams he may decide that the stream subsector be only four locations long and that it contain the IVBAC of the controller pressure.

Hence the NSR writer can choose his own convention regarding the arrangement of stream data depending on the
type of system the digraph is modeling. Of course some of the details of the NSR will depend on this convention. As with all other sectors the cards associated with the stream should appear in numerical order.

H. Some Features of the Predictor-Corrector Algorithm

On the first time step, the predicted values, as indicated by the superscript $p$, are calculated by the first-order Euler method:

$$x^p(h) = x(0) + h*f(x(0),y(0))$$ (10)

where the absence of a subscript on the symbols $x$, $y$, and $f$, signifies that the quantities are vectors. The corrected values, as indicated by the superscript $c$, are calculated, without iteration, from the second-order trapezoid rule:

$$x^c(h) = x(0) + h*(f(x(0),y(0)) + f(x^p(h),y^p(h)))*.5$$ (11)

which has a truncation error of $h^3*x^{(3)}(z)/12$, where $x^{(3)}(z)$ is a vector of the third derivatives of $x$ evaluated "near" $h$. Once $t$ is greater than $h$, predicted values are calculated from a second order algorithm:

$$x^p(t) = x^c(t-h) + (-f(x^c(t-2*h),y^c(t-2*h))$$

$$+ 3*f(x^c(t-h),y^c(t-h)))*h*.5$$ (12)
which has a truncation error of \(5^*h^3^*x^{(3)}(z)/12\). An improved predicted value, as indicated by the sub bar, is obtained from the following relation, which is derived in appendix A:

\[
x^P(t) = x^P(t) + 5*(x^P(t-h) - x^C(t-h))/6
\]  

(13)

The corrected values are calculated from the trapezoid rule

\[
x^C(t) = x^C(t-h) + (f(x^P(t),y^P(t)) + f(x^C(t-h),y^C(t-h)))*.5
\]  

(14)

and an improved corrected value is obtained from

\[
x^C(t) = x^C(t) + (x^P(t) - x^C(t))/6
\]  

(15)

which is also derived in the appendix.

The formation and solution of the system of implicit and explicit algebraic equations was considered the most important goal of this dissertation. On the other hand, the solution of the system of ODEs was given a lower priority primarily because this aspect of simulation is relatively well-researched, and it was believed that no significant advance would be made in this area. In addition, it is worth noting that in those cases where the GDE system is "stiff" (2) the value of the step size, \(h\), must be small whether or not a higher order method such as the
Adams–Moulton method is used. Consequently, the predictor-corrector method chosen for OSUSIM is relatively simple, however, since the NSRs are algorithm independent, OSUSIM could be modified in future versions to incorporate a higher order method. However, as will be seen when the transport lag and source nodes are discussed, this method should be of the predictor-corrector and not the Runge–Kutta type.
CHAPTER IV
THE DATA STRUCTURE OF OSUSIM

The data structure is based on three one dimensional vectors IPOIN, IVEC, and FPVEC. The elements of IPOIN are also the elements of a Fortran labeled common area named POINT which consists of 39 pointers and counters, each with a Fortran variable name. IVEC and FPVEC are Fortran names for an integer valued dimensioned variable and a real valued dimensioned variable, both being divided up into sectors that are located by pointers rather than names although for reference purposes each sector does have a name.

With one exception, to be discussed below, the sector pointers point to the location just before the first element in the sector. Hence, the third element in the NECALL sector of IVEC, for which the pointer is IPNEC, has its value stored in IVEC(IPNEC + 3). There are three kinds of sectors in IVEC: director sectors, vector sectors, of which NECALL is one, and matrix sectors. A vector sector is simply a set of contiguous elements in IVEC or FPVEC and the location of any element in such a sector can be determined by adding that element's relative position in the sector to the sector pointer.
Matrix sectors are broken up into consecutive subsectors of not necessarily equal length and each matrix sector has a director sector whose elements act as subsector pointers, i.e., the ith element in the director sector is the pointer for the ith subsector of the associated matrix sector. For example, IPLE is the pointer for the director sector named IDLE which is associated with the matrix sector LISTE; therefore the Jth element in the Ith subsector of LISTE would have its value stored in IVEC(IVER(IPLE + I) + J). In figure 8 the fourth element in the second subsector of the LISTE sector is located. Note that in conventional Fortran matrix notation, where the row and column length is fixed, IVEC(IVER(IPLE + 2) + 4) would correspond to LISTE(2,4) if the subsectors were considered as rows.

FPVEC consists of vector and matrix sectors; the latter having their director sectors located in IVEC. The reader will notice that the matrix sector is defined similarly to the conventional matrix except that the subsectors which correspond to the rows (or columns) of a conventional matrix can have varying lengths (including a length of zero).

In the remainder of this section each sector will be briefly defined for reference purposes; hence the reader will probably have to return to these definitions several
FIGURE 8. ILLUSTRATION OF THE POINTER-DIRECTOR SECTOR SYSTEM
times while reading the following sections.

A. Definition of the Sectors in IVEC

In later sections when the operation of the analysis and compute packages are described the definitions given here will be expanded. The reader may refer to figure 9 which shows the schematic structure of IVEC and FPVEC. During the execution of the analysis package when the sectors are being dimensioned and built up, IVEC(1) and FPVEC(1) are used to store the index of their last used location.

The first sector in IVEC, IDKP, having the pointer IPKP, is the director sector for the KPM sector in IVEC. For each node there is a subsector of KPM containing the stream numbers of the streams connected to it and the alphameric name of the NSR that describes the node's behavior. The contents of this sector are read in but as soon as the NECALL sector (to be discussed below) is constructed the locations containing the NSR names are zeroed.

The second sector in IVEC, IDP, having the pointer IPPAR, is the director sector for the sector in FPVEC called FPAR which contains for each node a subsector storing parameter values and values, bounds, and codes for internal variables.
FIGURE 9. OSUSIM DATA STRUCTURE
The third sector in IVEC, IDIP, having the pointer IPIP, is the director sector for the sector in IVEC called IPAR which contains a subsector for each node providing storage for integer valued parameters.

The fourth sector in IVEC, IDST, having the pointer IPST, is the director sector for the sector in FPVEC called STRMV which contains a subsector for each stream providing storage for the value, bounds, and code for each external variable associated with that stream. Hence if there are three external variables associated with a given stream then its subsector would contain twelve elements.

The fifth sector in IVEC, IDSV, having the pointer IPSV, is the director sector for the sector in FPVEC called SV which contains a subsector of length $5*NLDE+1$ for each node. The first element of each subsector is a flag and the remaining elements provide working storage used by the routine implementing the predictor-corrector algorithm. The director sector IDSV is unique in that the pointers point to the first element in each subsector of SV rather than the element just before. This is done because that first element is the above mentioned flag which is referred to relatively infrequently.

The sixth sector in IVEC, IDLV, having the pointer IPLV, is the director sector for the sector in IVEC called LISTV which contains a subsector of length NLAV
for each node. The ith position in each subsector contains the index of the GAV matched with the ith LAV.

The seventh sector in IVEC, IDLE, having the pointer IPLE, is the director sector for the sector in IVEC called LISTE which contains a subsector of length NLIAE for each node. The ith position in each subsector contains the index of the GAE matched with the ith LIAE.

The ith position in the eighth sector in IVEC, NECALL, having the pointer IPNEC, contains the NSR call number of the ith node. This call number is used in a FORTRAN Computed GO TO statement in the subroutine ECALL to direct program control to the proper NSR during a pass through the nodes.

The ninth sector in IVEC, LIST2, having the pointer IPL2, is a list of those nodes for which NLDE is not equal to zero. Program control is passed to these nodes when it is desired to evaluate the $f_i$.

The tenth sector in IVEC, LIST4, having the pointer IPL4, is a list of nodes for which NLEAE is not equal to zero. Program control is passed to these nodes when it is desired to evaluate the LEAVs.

The ith position in the next four sectors, IDERQ, NLDEQ, NLAVQ, and NLIAEQ, having the respective pointers IPDER, IPNLD, IPAV, and IPAE, contains the value of IDER, NLDE, NLAV, or NLIAE, respectively, for the ith node.
The fifteenth sector in IVEC, IDXID, having the pointer IPXID, is the director sector for the LDVDR sector which contains a subsector of length NLDE for each node. The ith position in each subsector contains the location in FPVEC where the value of that node's ith LDV is stored.

The sixteenth sector in IVEC, IDYID, having the pointer IPYID, is the director sector for the sector in IVEC called LAVDR which contains a subsector of length NLAIR for each node. The ith position in each subsector contains the location in FPVEC where the value of that node's ith LAV is stored.

The next six sectors are the matrix sectors KPM, IPAR, LDVDR, LISTE, LISTV, and LAVDR, which have been discussed above.

The ith position in the twenty-third sector of IVEC, GAVDR, having the pointer IPG, gives the position in FPVEC where the value of the ith GAV is stored.

The twenty-fourth sector in IVEC, LIST3, having the pointer IPL3, is a list of nodes having one or more decoupled GAEs. Control is passed to the nodes on this list when the values of the decoupled gi and the appropriate partial derivatives are needed.

Different use is made of the remaining locations in IVEC at different phases of the execution of the analysis.
package. If the original dimension of the global algebraic system was ICTE, then if, during the decoupling phase, NG subsystems are identified, this remaining area will be composed of a sector of length ICTE named LTMP, having the pointer IPTMP, followed by NG sectors, called group list sectors, each of length ICTE + 1, the first sector having the pointer IPSSG. After the decoupling phase is completed these sectors are destroyed and NG + 3 new sectors are defined. The ith position in the first of these new sectors, DIM, having the pointer IPTMP and a length of NG, contains the dimension of the ith subsystem.

The second sector, called PAIRS, with a length of 2*NG and having the pointer IPLI, is made up of pairs of quantities, one for each of the NG subsystems. The first element of the ith pair is a pointer for the list of nodes that have equations in the ith subsystem. The second element is a pointer for the area in the DGDY sector of FPVEC which will store the augmented Jacobian matrix for the ith subsystem.

The next NG sectors, each of length ICTE, are the node lists for the NG subsystems and have pointers stored in the PAIRS sector which has been mentioned above. The last sector, IPOS, having the pointer IPPOS and a length of ICTE, is a utility vector used by the routine GJ which
implements the Gauss-Jordan algorithm used in solving the linear system generated during the execution of the Newton-Raphson method. With the exception of the IPOS sector, the contents of all these 26 + NG previously mentioned sectors of IVEC are punched on to cards at the termination of the analysis package execution.

B. Definition of the Sectors in FPVEC

FPVEC consists of fourteen sectors, the first three, FPAR, STRMV, and SV, having already been mentioned. The ith position in the fourth sector, X, having the pointer IPX and a length equal to the maximum value of NLDE over all the nodes, stores the value of the ith LDV for the node, to whose NSR, program control will be passed next during the passes supervised by the compute package. This linear sector is loaded before control is passed to each NSR, and if $x_k$ occurs in one of the NSR expressions for $f_i$, $g_i$, or $\partial g_i/\partial y_j$, its value can be obtained from FPVEC(IPX + K).

The ith position in the fifth sector, F, having the pointer IPF and a length equal to that of the X sector, stores the value of the $f_i$ for the node to whose NSR program control has just been passed during a pass supervised by the compute package. This sector is loaded by each NSR on the pass that calculates values of the $f_i$ and
its contents are used after control has passed from each NSR by the routine that supervises the solution of the ODEs. Hence, if the NSR must calculate the value of $f_k$ it would be stored in $FPVEC(IPF + K)$.

The sixth sector, FP, having the pointer IPFP, is used during the solution of the ODEs to store old values of the $f_i$.

The ith position in the seventh sector, $y$, having the pointer IPY and a length equal to the maximum value of $NLAV$ over all the nodes, stores the values of the ith LAV just as sector X does for the LDVs. This sector, like sector X, is loaded before control is passed to each NSR, hence if $y_k$ occurs in one of the NSR expressions its value can be obtained from $FPVEC(IPY + K)$.

The ith position in the eighth sector, $g$, having the pointer IPTHL and a length equal to the maximum value of $NLIAE$ over all the nodes, stores the value of the ith LIAE just as sector F does for the LDEs.

The ninth sector, dgdy, having the pointer IPPY and a length equal to the product of the maximums of $NLIAE$ and $NLAV$ over all the nodes. This sector performs a function similar to sector g in that it stores values of the partial derivatives, $\partial g_i/\partial y_j$, supplied by the NSR, in the following order:
\[
\frac{\partial g_1}{\partial y_1}, \frac{\partial g_2}{\partial y_2}, \ldots, \frac{\partial g_{\text{NLIAE}}}{\partial y_1}, \frac{\partial g_1}{\partial y_2}, \ldots, \frac{\partial g_{\text{NLIAE}}}{\partial y_2}, \frac{\partial g_1}{\partial y_3}, \ldots
\]

Therefore, given values of i and j, the value of \( \frac{\partial g_i}{\partial y_j} \) would be stored in \( \text{FPVEC}(\text{IPPY} + (J - 1) \times \text{NLIAE} + I) \).

The tenth sector, \( g\)-save, having the pointer IPES and a length equal to that of the sector \( g \), is used to store values of the \( g_i \) during the perturbation operations that yield the numerical estimates of the partial derivatives.

The eleventh sector, \( y\)-save, having the pointer IPYS and a length equal to that of the sector \( y \), performs the same function for the LAVs that sector \( g\)-save does for the LIAEs.

The twelfth sector is the Jacobian matrix sector, \( \text{DGDY} \), and is composed of \( NG + 1 \) areas: one for the decoupled GAEs and \( NG \) for the subsystems. If there are \( KCO \) decoupled GAEs then the first area, for which IPPZ is the pointer, consists of \( KCO \) pairs of quantities. The first member of the ith pair is \( \frac{\partial G_i}{\partial Y_i} \) and the second is \( -G_i \), where the upper case symbols, \( G_i \) and \( Y_i \), represent the GAE and GAV respectively, as compared to the lower case symbols, \( g_i \) and \( y_i \), which represent the LIAE and LAV.
The next NG areas store the values of the partial derivatives, \( \frac{\partial G_i}{\partial Y_j} \), and the function values (with a sign change), \(-G_i\), for each of the NG subsystems. Each of the NG areas is divided into subsectors which store the rows of the augmented Jacobian matrix associated with that subsystem. Thus if the first subsystem has a dimension of \( N_1 \) then there are \( N_1 \) subsectors of length \( N_1 + 1 \) in the second area of the DGDY sector. Similarly the third area would be used to store the \( N_2^*(N_2 + 1) \) elements of the augmented Jacobian associated with the second subsystem.

Consider the following example where there are two \((KCO = 2)\) singly decoupled GAEs and two subsystems of dimension two and three. The DGDY sector would then have the following composition:

\[
\begin{bmatrix}
\frac{\partial G_1}{\partial Y_1} & -G_1 & \frac{\partial G_2}{\partial Y_2} & -G_2 & \frac{\partial G_3}{\partial Y_3} & \frac{\partial G_3}{\partial Y_4} & -G_3 & \frac{\partial G_4}{\partial Y_3} & \frac{\partial G_4}{\partial Y_4} & -G_4 \\
\frac{\partial G_5}{\partial Y_5} & \frac{\partial G_5}{\partial Y_6} & \frac{\partial G_5}{\partial Y_7} & -G_5 & \frac{\partial G_6}{\partial Y_5} & \frac{\partial G_6}{\partial Y_6} & \frac{\partial G_6}{\partial Y_7} & -G_6 & \frac{\partial G_7}{\partial Y_5} & \frac{\partial G_7}{\partial Y_6} \\
\frac{\partial G_7}{\partial Y_7} & -G_7
\end{bmatrix}
\]

The elements in the subsystem areas are referenced by means of the pointers that reside in the PAIRS sector mentioned above.

The thirteenth sector, DELTY, has the pointer IPDZ and a length equal to ICTV which is the total number of
GAVs occurring in the global implicit algebraic system. The first \( KCO \) locations of \( \text{DELTY} \) store the \( \Delta Y_i \) obtained from the one-dimensional Newton-Raphson method applied in turn to each of the decoupled GAES. If the first subsystem has a dimension of \( N_1 \) then the next \( N_1 \) locations of \( \text{DELTY} \) store the values of the first subsystem's \( N_1 \)-dimensional solution vector and so on for each of the remaining subsystems.

The fourteenth and last sector, \( \text{RM MAX} \), having the pointer \( \text{IPTM} \), and a length of \( \text{ICTV} \), stores the maximum absolute value of each row in the Jacobian matrix and is used to normalize the Jacobian matrix before the Gauss-Jordan reduction is carried out.

In summary, then, there are four kinds of sectors in \( \text{FPVEC} \). Sectors \( \text{FPAR} \) and \( \text{STRMV} \) store current information on the nodes and the streams. Sectors \( \text{SV} \), \( \text{DGDY} \), \( \text{DELTY} \), and \( \text{RMAX} \) store information used at each time step for the numerical solution of the algebraic and differential equations. Sectors \( \text{X} \), \( \text{F} \), \( \text{FP} \), \( \text{y} \), \( g \), and \( \text{dgdy} \) are used to transfer information between the nodes and the executive routines that are supervising the numerical solutions.

Of all the sectors, the contents of this last group change most frequently. For example, when the ODEs are being solved, before the call to node A, sector \( \text{X} \) is loaded with the values of the node A LDVs that are fetched
from the appropriate locations in the FPAR and STRMV sectors. Program control is passed to the NSR for node A where values of the \( f_i \) are calculated and loaded into sector F. The necessary use is made of the information in sector F by the supervising routine, SRN8Pl, then sector X is reloaded with LDV values for node B and the values of node B's \( f_i \) are calculated and loaded into sector F as was done for node A. This process is continued for the remaining nodes in the LIST2 sector.

Finally sectors g-save and y-save can be classed together because they provide temporary storage for values of the functions \( y \) and \( g \) during the perturbation operations that yield estimates of the derivatives.
CHAPTER V

THE ANALYSIS PACKAGE OF OSUSIM

This chapter will give a detailed discussion of the analysis package logic. The overall logic is presented by describing the subroutine MAINP which serves as the analysis package executive routine. Subsequent sections of this chapter are devoted to each of the primary functions of the analysis package.

Routines which read in the input data and set up the sector boundaries are discussed first. The process of identifying the local differential variables and the construction of the sectors associated with them is described next. Third, the logic of the routines that supervise the identification of the global algebraic variables is presented.

After the structure of the global algebraic system has been determined, it is analyzed in an attempt to decouple single equations and disjoint subsystems. This decoupling is the subject of the next two sections in the chapter. These sections are followed by a section on the several utility routines that are used in the above mentioned operations. Finally, recommendations for extending the analysis of the global algebraic system are made.
A. Overall Logic

The subroutine MAINP acts as the executive program for the analysis package, hence a discussion of its features will reveal the overall logic. Referring to the flow diagram in figure 10, it is seen that MAINP consists mostly of calls to other subroutines which actually carry out the analysis package functions. The first of these functions, as carried out by the subroutine INTLZE, is to read and echo the cards that contain the input data, sort the data and construct the FPAR and STRMV sectors in FPVEC and the NECALL, KPM, and IPAR sectors in IVEC. Of course the director sectors associated with the FPAR, STRMV, KPM, and IPAR sectors are constructed as well. INTLZE also sets up the pointers to most of the other sectors which are still unfilled.

The next subroutine to be called, SRN7, supervises a pass through all the nodes during which sectors IDERQ, NLDEQ, NLAQV, NLAEQ, LIST4, LIST2, and LDVDR (and its director sector) are constructed, the last two being the result of identifying the LDVs and the LDEs. The identification of the LIAEs and LAVs and their matching with the GAEs and GAVs is carried out by SETUPS, the third routine to be called by MAINP. During this matching phase, the sectors LISTE, LISTV, and LAVDR, along with their director sectors are constructed. Also constructed
ENTRY

CALL INTLZE
read in system structure

CALL SRN7
supervise LDW identification

CALL SETUPS
supervise LAV-GAV matching

ICTE: 0 ≠ 1
no decoupling to be attempted

ICODE = 1, ITURN = 1, ICALL = 3, INST = 3

CALL ZEROX
zero DGDY sector

CALL JACO
evaluate Jacobian matrix

CALL OCCUR
display Jacobian matrix

attempt SED

LIST5(11): 1 ≠ 5

FIGURE 10. MAINP (for analysis package)
call DECUP

attempt SED

NG = 0
IPPO = IPTMP
IPLI = 0

attempt ASI

call SBSYS

NG = 1
IVEC(IPTMP + 1) = ICTE
IPLI = IPTMP+1
IVEC(IPLI + 1) = IPLI + 2
IVEC(IPLI+2) = IPKCO
IPPO = IPLI + 2 + ICTE

call ZERO (to zero list area)

call CNLST(IPLI+3, 0,ICTE, IVEC)

print and punch contents of IPOIN, IVEC, FPVEC

read ICODE

begin a new problem

FIGURE 10. (continued)
is the first version of the GAVDR sector which will remain unchanged if there is no single equation decoupling (SED) or algebraic subsystem identification (ASI). As a result of the matching phase, a coupled algebraic system of ICTE GAES in terms of ICTV GAVs is identified. Obviously ICTE must equal ICTV, otherwise one of the NSRs is incorrectly conceived or there is an error in the input data.

If ICTE is found to be zero then SED or ASI is not possible and control is transferred to box one. However if ICTE is not zero then the following preparation for the SED and ASI operations is made: (1) sector DGDY is zeroed, (2) subroutine JACO, which evaluates the ICTE*ICTE Jacobian, is called and (3) subroutine OCCUR is called to display the Jacobian in rectangular row-column form with blanks for zero elements and "1"s for nonzero elements. If the modeler does not wish SED to take place, the value of LIST5(11) can be set to zero in the input data. This option might be exercised when the modeler knows a priori that none of the nodes are described by algebraic equations. In this case the DECUP routine can be replaced by a dummy routine with the same name allowing for a savings in computer memory.

If DECUP is called and SED is attempted, then the value of the variable KCO represents the number of singly
decoupled equations and ICTE will be decreased by that amount. If ICTE is zero then the system has been entirely decoupled and no ASI is possible so control is transferred to box one. Even if ICTE is greater than zero after SED, the modeler can prevent the ASI operation by assigning a value of zero to LIST5(12) in the input data. The reasons for choosing this option would be similar to those mentioned above in connection with the avoidance of the SED operation. Actually the space savings would be much greater here since the routine SBSYS which implements the ASI is by far the largest routine in the analysis package.

In any case, if ICTE is greater than zero and LIST5(12) is not equal to one, then NG, the number of algebraic subsystems, is assigned the value one and the sector DIM, consisting of one element, is assigned the value of ICTE. The next sector, PAIRS, consists of two elements, a pointer to the list of nodes that will provide the values of the GAE, and a pointer to the area of the DGDY sector that will be used during the solution of the equations. Since there is only one subsystem, there will be at most two areas in the DGDY sector: the first, of size 2*KCO, for the singly decoupled equations, and the second, of size ICTE*(ICTE + 1), for the coupled
subsystem. Finally, a value is assigned to the pointer for the IPOS sector which follows the not yet constructed node list. A call to ZEROX zeroes the node list sector and a call to CNLST constructs it.

In box one, to which control is transferred if ASI is not possible, NG and the pointer to the PAIRS sector are both set to zero since there is no coupled algebraic system. Also the pointer to the IPOS sector is given the value of the pointer IPTMP since sectors DIM, PAIRS and the node lists do not exist. When ASI is desired, control is transferred to box two where the routine SBSYS, which carries out ASI, is called.

This completes the analysis of the problem and control is transferred to box three where the values of IPOIN, IVEC (up to the IPOS sector), and FPVEC (up to the SV sector) are punched on cards. If there are more problems to be analyzed, the card following the input data for the problem that has just been analyzed will be blank and when that card is read, a zero value will be assigned to ICODE which will then cause control to be transferred to box four where the analysis of another problem will commence. Otherwise ICODE will have a nonzero value and control will be returned to the main program which will terminate execution of the analysis package. Hence the
input data for each problem must be separated by a blank card and the input data for the last problem must be followed by a card with a nonzero integer in column one.

B. Input Data Processing

Two routines INTLZE and INP are used to process the input data. INTLZE sets up pointers and prepares for calls to INP which actually reads in the data. Referring to figure 11 it is seen that the first card contains the problem title which is printed as soon as it is read. The next set of cards contain values that are assigned to two utility vectors, LIST5 (integer valued) and DUT (real valued). The last card read by INTLZE contains values that are assigned to NEUB, the number of nodes, and NSUB, the number of streams. These two quantities are then used to set up sector boundaries for the first sixteen sectors in IVEC which will consume NDIM locations. These locations are then zeroed by a call to ZEROX. The next group of cards contains the data that will make up the KPM sector.

The actual card reading is done by INP which needs three pieces of information. First, the value of the pointer for the director sector associated with the matrix sector must be assigned to IDIR. Since the director sector, IDKP, for the KPM sector is the first sector in IVEC,
set up some of the pointers to sectors in IVEC

IPKP = 1, NDIM = NEUB + 1, IPPAR = IPKP + NEUB
IPIP = IPPAR + NEUB, IPST = IPIP + NEUB
IPSV = IPST + NSUB, IPLV = IPSV + NEUB
IPLE = IPLV + NEUB, IPNEC = IPLE + NEUB
IPL2 = IPNEC + NEUB, IPL2 = IPL2 + NDIM
IPDER = IPL2 + NDIM, INLD = IPDER + NEUB
IPAV = IPNLD + NEUB, IPA + = IPAV + NEUB
IPXID = IPAE + NEUB, IPXID = IPXID + NEUB
IPYID = IPXID + NEUB, NDIM = IPYID + NEUB

call ZEROX (zero area in IVEC)

IDIR = 1, IVEC(1) = NDIM

call INP(IV, FP, -1, NEWMAXL)

ISTOP = NEWMAXL + 1

FIGURE 11. INTIZE
DO 100 I = 2, ISTOP

IM1 = I - 1, IARG1 = IVEC(1)

IARG1 = 0 -> 100

IP1 = I + 1

DO 101 J = IP1, ISTOP

IARG1 = IVEC(J)

IARG1 = 0 -> 101

IARG3 = IVEC(IARG1)

1

DO 102 K = 1, 17

IARG3 = NMI.LST(K)

= ≠ 102

output: "name not on list"

100

FIGURE 11. (continued)
print NECALL sector

IDIR = IPPAR
FPVEC(1) = 1.

call INP(IVEC, FPVEC, 1, NEMAX2)
read in FPAR sector

IDIR = IPST

NEMAX = NEMAX1
IDIR = IPST

call INP(IVEC, FPVEC, 1, NSMAX)

IF1 = FPVEC(1)

RETURN

FIGURE 11. (continued)
IDIR is given the value of one. Second, the last used or reserved location in the vector to be filled must be assigned to the first location of that vector. Since the KPM sector is located in IVEC and since the first NDIM locations have been already reserved, IVEC(1) is given the value of NDIM. Third, a flag, which is the third argument of INP, must be set indicating what kind of matrix sector is being read in. For the matrix sectors in FPVEC, namely FPAR and STRMV, the flag will be set to one. For the IPAR sector in IVEC the flag will be set to zero. Although KPM is also in IVEC certain manipulations to be discussed below must be made so it is differentiated from IPAR by a flag value of minus one.

After INP reads in and constructs the KPM sector and its director sector IDKP there is, for each node, a subsector of the KPM sector containing the stream numbers of the connected streams followed by a zero and then by the alphanumeric name of the NSR that describes that node's behavior. Before any more data is read in, INTLZE scans the KPM sector, picks out the NSR name for each node, matches it with one of the 17 names on the vector list named NMLIST and determines the node's call number. These call numbers are then placed in the NECALL sector.

The problem is complicated by the location of the
NSR name at the end of each node's subsector which can have variable lengths. To simplify the following presentation let \( j \) equal \( i-1 \). Now, if the name for the \( j \)th node is sought, the location just before the beginning of the next subsector must be determined. Unfortunately there may be no node index equal to \( i \) so the subsector pointer for the \( j \)th node will not necessarily yield the NSR name for the \( j \)th node. The flow diagram on the second page of figure 9 shows how this problem is solved.

First, \( IARG1 \), the pointer for the \( j \)th node's subsector in KPM, is fetched from the IDKP sector. If \( IARG1 \) is zero this indicates that there is no node in the digraph with an index equal to \( i=1 \) and \( i \) is incremented. If \( IARG1 \) is not zero then IDKP is searched for the next non-zero pointer and its value is assigned to \( IARG4 \) which points directly to the NSR name of the \( j \)th node. Should the index of the highest numbered node be \( j \), then IDKP will not yield such a pointer but the value of \( IARG4 \) can be extracted from \( IVEC(1) \) which points to the last used location in IVEC. This will be the last position in the last subsector of KPM since the KPM sector is the most recently constructed sector.

In any case the NSR name can be obtained from \( IVEC(IARG4) \) and stored in \( IARG3 \) which is then compared
with the elements of NMLIST. If IARG3 matches the kth name on NMLIST then k is assigned to the jth element of the NECALL sector and the location formerly storing the NSR name is zeroed. This is done because the contents of IVEC are punched and printed in integer format which in general can not accept alphameric characters. Future versions of OSUSIM might be modified so that NECALL is constructed as each node's subsector is read in. This would save 2*NEUB locations in IVEC but of course INTLZE and INP would have to be changed.

The next group of cards contains the data that will make up the FPVEC sector. Preparation is made for the call to INP by (a) setting IDIR equal to the value of FPVEC's director sector pointer, IPPAR, (b) assigning the first location of FPVEC a value of one since no locations in FPVEC have as yet been used or reserved, and (c) assigning a value of one to the flag.

The third group of cards contains the data that will make up the IPAR sector in IVEC. This time the value of IVEC(1) is not set before the call to INP because during the KPM sector construction IVEC(1) was continually updated. Likewise before the final call to INP when the STRMV sector data is read in FPVEC(1) is not set because it was updated during the construction of the FPAR sector.
After this last sector has been constructed FPVEC(1) contains the last used location in FPVEC and its value is assigned to IPL which will be used in the Fortran statements that punch and print the contents of FPVEC.

Subroutine INP is in effect a utility routine designed to read in and construct any one of the FPAR, STRMV, IPAR, and KPM sectors. Referring to figure 12 it is seen that for a given sector the data for each subsector is read in and processed separately in five steps. The subsector index IX, the subsector length LNGTH, and the NSR name (meaningful only for the KPM sector) are read in first. Second, the last used or reserved location, KLAST, is fetched from IVEC(1) or FPVEC(1) depending on the value of the flag ITYPE. Third, the number of cards containing the subsector's data, NCARD, is determined based on the value of LNGTH. Fourth, the IX-th position in the appropriate director sector is assigned the value of KLAST. Fifth, the contents of the next NCARD cards are read into the next available locations in IVEC or FPVEC, and IVEC(1) or FPVEC(1) is updated.

Following this fifth step, if a subsector of KPM is being processed, a zero is placed at the end of the subsector data, followed by the NSR name. This zero is vestigial and serves no purpose hence INP should be modified.
FIGURE 12. INP(IVAR,VAR,ITYPE,IMAX,IVEC,FPVEC)
I = 0, IVAR(IDIR + IX) = KLAST

I = I + 1, KSTART = KLAST + 1

KSTOP = KLAST + NPCRD

I: NCARD

NLOV = LNGTH - (NCARD - 1)*NPCRD
KSTOP = KLAST + NLOV

ITYPE: 0

read integer values from card into IVEC sector
IVEC(1) = KSTOP

read real values from card into FPVEC sector
FPVEC(1) = KSTOP

KLAST = KSTOP

ITYPE: 0

KLAST = KLAST + 1, IVAR(KLAST) = 0
KLAST = KLAST + 1, IVAR(1) = KLAST
IVAR(KLAST) = NAME

read next subsector data

FIGURE 12. (continued)
in future versions of OSUSIM. After each subsector is constructed, control is returned to box one where a new subsector index is read in. A negative value of this index indicates that the sector processing is complete and control is returned to the calling routine INTLZE.

It should be noted that except for the KPM sector the order of the subsectors within the sector is unimportant. This is because the algorithm that constructs NECALL sector from the KPM sector data assumes that the subsectors are located in the order of their indices. However it is recommended that the subsector data for every sector appear in the order of its index. Also note that if there is a value of i, less than NEUB, that is not assigned to a node in the digraph, there will be a zero in the ith position of the director sector IDKP and no space at all in the KPM sector. This is to be contrasted with programs that use FORTRAN two dimensional matrices where the ith row would be wasted.

The five steps associated with the subsector and director sector construction can be summarized as follows. Just before the data for the ith node's or stream's subsector is read into IVEC or FPVEC, the last used or reserved location in that vector is fetched from IVEC(l) or FPVEC(l) and placed in the ith location in the appropriate
director sector. This is done because the value stored in the ith location of any director sector must point to the location just before the beginning of the ith subsector of the appropriate sector. The subsector data, consisting of, say, N values, is then read into the next N available locations of IVEC or FPVEC and the location of the last data value is assigned to IVEC(I) or FPVEC(I) because it will be needed when the next subsector data is read in.

C. Preliminary Interrogation of the Nodes and Identification of the LDVs

Subroutine SRN7 supervises the first pass through the nodes. The objectives of this pass are to (a) determine the values of IDER, NLDE, NLAV, and NLIAE, for each node and to place these values in the IDERQ, NLDEQ, NLAVQ, and NLAIEQ sectors respectively, (b) determine the value of NLEAV and construct a list of nodes, LIST4, having positive values of NLEAV, (c) set up the boundaries for the subsectors of the SV sector, (d) determine NLDEM, the maximum value of NLDE, NLAVT, the total number of LAVs, and NLAET, the total number of LIAEs, (e) cause the NSRs to call the COMPRX subroutine which identifies the LDVs and constructs the LDVDR sector, and (f) construct a list of nodes, LIST2, having positive values of NLDE.
Referring to figure 13 it is seen that, for each node index from one to NEMAX (or NEUB), the following steps are carried out. First, the KPM subsector pointer for each node index is fetched. If it is nonzero, then the node index is assigned to the variable NE and the IPAR and FPAR subsector pointers are assigned to IBIP and IBPA. The variables IBIP and IBPA can be used by the NSRs for fetching values from the IPAR and FPAR sectors. Second, a call is made to subroutine ECALL which calls the correct NSR by using NE to select the appropriate call number from the NECALL sector and since ICALL has been set to unity (refer to figure 7), the NSR will call subroutine COMPX if NLDE is nonzero. After program control has been returned to SRN7, the values of NLDE, NLAV, and NLEAV will be available since they are FORTRAN COMMON variables and several program control transfers will be made depending on their values.

If NLDE is non-zero, control is transferred to box one where LIST2 is added to, the boundary of the NE-th subsector of the SV sector is set up, the value of NLDE is entered into the appropriate location of the NLDEQ sector and the value of NLDEM is updated. Next, if NLAV is non-zero, control is transferred to box three where the NLAVQ, NLIAEQ, and IDRERQ sectors are modified and the
ENTRY

\[
\text{NCALL} = 1, \quad \text{NIAVT} = 0, \quad \text{NIAET} = 0, \quad \text{NLDEM} = 0, \quad \text{IL2} = 1, \quad \text{I14} = 1
\]

DO 100 J = 1, NMAX

\[\text{NE} = \text{IVEC}(J+1)\]

\[\text{NE} = 0 \quad \Rightarrow \quad 100\]

\[\text{NE} = J, \quad \text{IBIP} = \text{IVEC}(\text{IPIP}+\text{NE}), \quad \text{IBPA} = \text{IVEC}(\text{IPPAR}+\text{NE})\]

CALL ECALL

\[\text{NLDE} = 0 \quad \Rightarrow \quad \text{1}\]

\[\text{NLAV} = 0 \quad \Rightarrow \quad \text{2}\]

\[\text{NLEAV} = 0 \quad \Rightarrow \quad \text{3}\]

\[\text{IVEC}(\text{IPL}4 + \text{I14}) = \text{NE} \quad \Rightarrow \quad \text{100}\]

\[\text{I14} = \text{I14} + 1\]

RETURN

FIGURE 13. SRN7
construct list of nodes with ODEs

fetch last location used in FPVEC

construct director sector for SV

zero 1st location and reserve space for subsector

construct NLDEQ sector

construct NLAVQ sector

count total no. of LAVs

construct NLAEQ sector and count total no. of LIAEs

construct IDRFRQ sector

**FIGURE 13. (continued)**
running sums NLAVT and NLAET are updated. The quantities NLDEM, NLAVT, and NLAET determined here by SRN7 will be used by other routines to set up the boundaries of several other sectors.

Finally, if NLEAV is not zero, NE is added to LIST4. Since LIST4 is added to in the order that the nodes are numbered and since this is the order that the nodes will be called during the pass that computes the LEAVs, the modeler may want to choose the node indices carefully. For example, a LEAV associated with node A may depend on the value of a LEAV associated with node B. In this case the node number of node A must be greater than that of node B. The node numbering will also be important in cases involving source nodes and delay nodes.

The subroutine COMPX is called by the NSRs once for their internal LDVs and once for each connected stream that has external LDVs associated with it. Referring to figure 14 it is seen that upon entry, the pointer to the node's subsector of LDVDR is fetched. If it is zero then this is the first call to COMPX for node NE so the last used or reserved location in IVEC is fetched from IVEC(1) and assigned to the appropriate location in the director sector for LDVDR.

The value of the flag ICODE determines whether external or internal LDVs are being dealt with. In the latter
ENTRY
\[ IB2 = IVEC(IPXID + NE) \]

for external LDVs

1

\[ IB2 = IVEC(1), \ IVEC(IPXID + NE) = IB2 \]

for internal LDVs

\[ ICODE = 0 \]

\[ INO = 3, \ IB1 = FPVEC(IPPAR + NE) \]

\[ IA = IXIS - 1, \ IB = ISTA - INO \]

DO 10 I = 1, NLDVS

\[ IB = IB + INO, \ IARI = IB1 + IB, \ IA = IA + 1 \]

\[ IARG = IB2 + IA, \ IVEC(IARG) = IARI \]

\[ IVEC(1) = IARG \]

10

RETURN

FIGURE 14. COMFX(ICODE, ISTA, IXIS, NLDVS, KSTR, IVEC, FPVEC)
FIGURE 114. (continued)
case, for each LDV three locations in the node's subsector of FPAR are used to store the value, lower bound, and upper bound (the LDV's triple) so INO is given the value three. Also IBl, the pointer to the subsector where the triple is stored, is fetched from FPAR's director sector. Program control is now transferred to box two where the actual construction of the LDVDR sector will take place.

If external LDVs are being dealt with, control is transferred from box three to box one where INO is set to four since the value, lower bound, upper bound, and matching code (the LDV's quadruple) of each LDV are stored in the appropriate subsector of STRMV. In order to determine IBl which is the pointer to the subsector of STRMV where this quadrangle is stored, the digraph stream index must first be determined and to do this, the value of KSTR which is the local number of the connected stream, must be specified in the argument list of the FORTRAN call statement in the NSR. For example, in the energy exchanger model described in part 5 d of chapter II, the second local stream as seen by the node is the outlet solids stream so if COMPX were to be called for the external LDV \( \chi_1 \) (the solids temperature), KSTR would be assigned the value of two.

The global stream number J can then be fetched from the KPM sector and used in conjunction with the direct
sector and used in conjunction with the direct sector for the STRMV sector to determine IBI. As with the internal LDVs, control is now transferred to box two where three more quantities from the argument list are used. The first, NLDVS, specifies how many LDVs are being dealt with on the call. The second, IXIS, specifies the local LDV index of the first of the NLDVS LDVs being dealt with, assuming that the indices of the LDVs increase in order. The third, ISTA, specifies the relative position in the subsector of FPAR or STRMV where the triple or quadruple of the first LDV to be dealt with is stored.

For example, in the energy exchanger model, one of the two LDVs is external and associated with the second local stream, so NLDVS and IXIS would both be one. Now suppose that the NSR writer decided that each of the solids transporting streams would be defined by its moisture fraction, temperature, and flow rate, in that order; then each subsector in the STRMV sector would consist of twelve locations and ISTA would be five since this model does not use the moisture fraction.

Therefore, knowing the values of NLDVS, IXIS, and ISTA, allows the determination, for each of the NLDVS LDVs, of IARI which is the location in FPVEC where the LDV's value is stored. The value of IARI is then placed
in the appropriate location in node NE's subsector of LDVDR. Finally, IVEC(1) is updated and control is returned to the calling program.

To make optimal use of the COMPX routine, the NSR writer should number his LDVs so that, for example, $x_1$ through $x_m$ are associated with one connected stream, $x_{m+1}$ through $x_n$ are associated with another stream and, say, $x_{n+1}$ through $x_p$ are internal and have their triples stored consecutively in FPAR. A simple rule to follow for FPAR design would be to allot the first $r$ locations for the values of the $r$ node parameters, the next $3s$ locations for the $s$ internal LDVs, and the last $4t$ locations for the $t$ internal LAVs. As an example, consider the water tank model where there are four real valued node parameters, one internal LDV, and no internal LAVs; hence the FPAR subsector for a node described by the water tank model would contain seven elements since $r = 4$, $s = 1$, and $t = 0$. Also note that in the call to COMPX for this internal LDV, ISTA would be five.

Each time a LDV is identified, a message is printed telling whether the LDV is internal or external, the node number it is associated with, where its value is stored in FPVEC and with what stream it is associated if it is an external LDV.
D. Compilation of the Global Implicit Algebraic System

Subroutine SETUPS supervises the second pass through the nodes during which the sectors LISTE, LISTV, LAVDR, and GAVDR are constructed. That these sectors actually represent the integration of all the LIAEs scattered throughout the nodes into a global system of ICTE GAEs of the form

\[ G_j(Y_1, \ldots, Y_{ICTE}) = 0, \quad j=1, \ldots, ICTE \]

can be seen in the following brief discussion. The global system is solved iteratively using the Newton-Raphson method (see appendix B), i.e., the kth iteration yields \( \Delta Y \), a vector of corrections having elements \( \Delta Y_i \), resulting from the negative product of the Jacobian matrix and the vector \( G \), having elements \( G_i \):

\[ \Delta Y_i = - \sum_{j=1}^{ICTE} \frac{\partial G_i}{\partial Y_j} G_j \]

To get values for the global quantities, \( \partial G_i/\partial Y_j \) and \( G_i \), a pass is made through the nodes and, using the LDVDR and LAVDR sectors which tell where the values of the local variables are stored, values of the local quantities \( g_i \) and \( \partial g_i/\partial y_j \) are computed. Since LISTE and LISTV give the GAE index and GAV index associated with each LIAE and LAV,
respectively, these two lists can be used to transfer the values of the local quantities to the global quantities. After each iteration, the global quantities $\Delta Y_i$ must be added to the contents of the appropriate location in the STRMV or FPAR sectors. Use is made of the GAVDR sector during this operation since GAVDR gives the location in FPVEC of each GAV's value. Therefore it is concluded that the description of the global algebraic system effectively resides in the contents of these four sectors.

The mechanics of constructing these sectors will now be outlined. Referring to figure 16 it is seen that before the pass through the nodes begins, various counters are initialized and two pointers are given values. The first, LASTV, points initially to the last location reserved for the LISTE sector which contains NLAET elements and follows the just-constructed LDVDR sector (see figure 15). LASTV will be used by subroutine MKLST during the construction of the LISTV sector. The second pointer, LYID, points initially to the last location reserved for the LISTV sector which contains NLAVT elements and follows the LISTE sector. LYID will be used by MKLST during the construction of the LAVDR sector which follows the LISTV sector. Note that IVEC(1) still contains the last location used during the construction of the LDVDR sector so it will
FIGURE 15. INITIAL POINTINGS FOR IVEC(1), LASTV AND LYID
ENTRY

$\text{ICTV} = 1, \quad \text{NLAED} = 0, \quad \text{NLAVD} = 0, \quad \text{NLAEM} = 0, \quad \text{NLAVM} = 0, \quad \text{ICTE} = 1$

$\text{LASTV} = \text{IVEC}(1) + \text{NLAET}, \quad \text{LYID} = \text{LASTV} + \text{NLAVT}$

$\text{ICALL} = 2$

DO 2 NE = 1, NEMAX

Call FETCH

$\text{NLAV} \neq 0$

Call ECALL

Last $= \text{IVEC}(1)$, $\text{ISTART} = \text{LAST} + 1$, $\text{IVEC}(\text{IPLE} + \text{NE}) = \text{LAST}$

$\text{NLAV} : \text{NLAVM} \rightarrow \text{NLAVM} = \text{NLAV}$

$\text{NLAV} : \text{NLA}D$\n
$\text{IDER} = 0 \rightarrow \text{NLA}D = \text{NLAV}$

$\text{NLAEM} = 0$

FIGURE 16. SETUPS
DO 100 J = 1, NLIAE

IVEC(ISTART) = ICTE, ICTE = ICTE + 1, ISTART = ISTART + 1

100

IVEC(1) = LAST + NLIAE

NLIAE: NLAEM

NLAEM = NLIAE

NLIAE: NLAED

NDER: 0 =

2

1

print LISTV and LISTE sectors

ICTE = ICTE - 1, ICTV = ICTV - 1, IPG = LYID, IPTMP = IPG + ICTV

IPL3 = IPTMP

IVEC(1) = IPTMP

FIGURE 16. (continued)
FIGURE 16. (continued)
be used during the construction of the LISTE sector which follows the LDVDR sector.

Next, for each node index from one to NEUB, the following steps are carried out. First, the subroutine FETCH is called which fetches NLIAE, NLAV, IDER, NLDE, and the IPAR and FPAR subsector pointers IBIP and IBPA for node NE. If NLAV is zero then there is no interest in that node so NE is incremented and step one is repeated. Second, ECALL is called and program control is transferred to the appropriate NSR and since ICALL has been set to two (refer to figure 6) subroutine MKLST will be called by the NSR. During this call to MKLST, parts of sectors LISTV and LAVDR will be constructed.

Third, after program control has been returned to SETUPS, the last used location in the LISTE sector is fetched and assigned to the appropriate location in LISTE's director sector. The fourth step consists of determining NLAVM, the maximum value of NLAV over all the nodes, and NLAVD, the maximum value of NLAV over those nodes that require numerical evaluation of the partial derivatives. In the fifth step, the counter ICTE is entered into the appropriate location of the LISTE sector and incremented. This is done once for each LIAE. Then IVEC(1) is updated. In the sixth step, NLAEM, the maximum value of NLIAE over
all the nodes, and NLAED, the maximum value of NLIAE over all those nodes that require numerical evaluation of the partial derivatives, are determined.

After these six steps are carried out for each node, the sectors LISTE, LISTV, and LAVDR have been completed. Subroutine GLOB, which constructs the GAVDR sector based on the information in the LISTV and LAVDR sectors, is then called. Finally, more pointers are given values based on the values of quantities such as NLAEM and ICTE that have been determined during the two passes through the nodes.

Subroutine MKLST is very similar to COMPX in that it constructs the LAVDR sector for the LAVs just as COMPX set up the LDVDR sector for the LDVs. In addition, MKLST must interrogate the LAV's code and decide whether the LAV is to be matched with a GAV and if so which one. Like COMPX, MKLST must be called by the NSR once for the node's internal LAVs and once for each stream connected to the node that has external LAVs associated with it.

Referring to figure 17 it is seen that upon entry, the node's pointer to its subsector of LAVDR is fetched. If it is zero, then this is the first call to MKLST for this node so the appropriate nonzero value is placed in LAVDR's director sector. Next, the LAV counter, ICOUN, is incremented and if ITYPE is one (or greater than zero)
FIGURE 17. MKLST(LOCNP, LSN, ITYPE, ISTB, NVAR, IVEC, FPVEC)
FIGURE 17. (continued)
the LAV is external and control is transferred to box one. If the LAV is internal, then IARH, the location in the director sector IDP that will yield the pointer to node NE's subsector in FPAR is fetched. Control is then transferred to box seven.

In box one the value of IARH for the external LAV is determined but the logic is more complicated since the global index of the stream with which the LAV is associated must first be found based on the value of the local stream index LSN. A similar type of determination was carried out in COMPX and will not be discussed in any more detail here.

Hence for both types of LAVs, control eventually is directed to box seven where the value of ISTA, the relative location of the LAV's value in the FPAR or STRMV subsector, is used to determine IBl, the absolute location of the LAV's value, which in turn can be used to find JARG, the absolute location of the matching code. The code, ICODE, for each LAV, internal or external, is then fetched from FPBEC(JARG) and control is transferred to box two where IBl is stored in the LOCNO th element of node NE's subsector of the LAVDR sector.

Next, ISTAR, node NE's subsector pointer for the LISTV sector is fetched. If it is zero, then this is again the first call to MKLST for node NE and the value of LASTV is assigned to ISTAR and also placed in the director sector
for LISTV. Knowing ISTAR allows the calculation of IARG1, which is the location in LISTV where the GAV index associated with the NE-th node's LOCNO-th LAV will be stored.

Now, to set the value of that index, control is transferred to different places in MKLST depending on the value of ICODE. First, if ICODE is less than -1, the variable is a LEAV so no matching with a GAV takes place and the value of ICODE is placed in IVEC(IARG1). Second, if ICODE is -1, the LAV's value is to be held constant throughout the simulation so no matching takes place and a value of zero is stored in IVEC(LARG1). Third, if ICODE is zero, then the LAV is to be matched with the ICTV-th GAV so the value of ICTV is assigned to both IVEC(LARG1) and to FPVEC(JARG). Then ICTV is incremented in anticipation of the next matching. Finally, if ICODE is greater than zero, then this LAV will share a GAV with another LAV that has already been matched.

This situation might arise when stream C flows from node A to node B. Then, if node A has a lower node index than node B, MKLST will be called for node A before node B and, on one of the calls to MKLST from the node A NSR, the outlet stream velocity, which is also the stream C velocity, will be matched to, say, the Jth GAV. Consequently, the value of J will be placed in both IVEC(IARG1)
and FPVEC(JARG). Now, one of the subsequent calls from
the node B NSR to MKLST will be for the node B inlet
stream velocity, which is also the stream C velocity, and
on this call, when the code is fetched from FPVEC(JARG)
it will be found to equal J because the stream C velocity
has already been matched during the call to MKLST from
the node A NSR.

In all four cases, control is eventually transferred
to box 40 where LASTV is compared to IARG1. Since IARG1
is the last used location in the LISTV sector, its value
is assigned to LASTV if it is greater than the current
value of LASTV. Next, the LAV counter, ICOUN, is compared
with NVAR, which is the number of LAVs to be processed on
this call. If they are equal, control is returned to the
calling NSR; if not, LOCNO and ISTA are incremented and
control is passed to box five where the processing of
the next LAV will commence.

To illustrate the use of MKLST in the NSR, consider
the water tank example. Here there are two pairs of LAVs,
all external, with one pair, \( y_1 \) and \( y_2 \), associated with
local stream number one and the other pair, \( y_3 \) and \( y_4 \),
with local stream number two. Therefore there would be
two calls to MKLST. Now suppose that the fluid transporting
streams are defined in terms of four quantities: the
concentration of some chemical component, the pressure, the velocity, and the temperature. Hence on the first call the MKLST for the first pair, the argument list variables would be given the following values: LOCNP = 1, LSN = 1, ITYPE = 1, NVAR = 2. On the second call for the second pair, the argument list variables would be: LOCNP = 3, LSN = 2, ITYPE = 1, ISTB = 5, NVAR = 2.

The reader should note that the order of numbering the LAVs should always be the same as the order in which the variables are stored in the STRMV subsector, i.e., to identify $p_1$ and $v_1$ with $y_4$ and $y_3$ instead of $y_1$ and $y_2$ would mean the MKLST must be called once for each LAV. Also note that in both MKLST and COMPX it should not be necessary to supply both a local stream number and a flag indicating the type of variable involved. Instead, a zero value of the local stream number should be sufficient to indicate that the variable is internal and a nonzero value could indicate both the external nature of the variable as well as its local stream number. This modification is suggested for future versions of OSUSIM.

Each time a LAV is identified, a message is printed telling whether the LAV is internal or external, the node number it is associated with, where its value is stored in FPVEC, the initial code, and with what stream it is associated if it is an external LAV. If the LAV
is matched with a GAV, the GAV index will also be printed.

E. Decoupling the Single Equations from the Global Algebraic System

If the Ith GAE happens to depend upon one and only one GAV, say the Jth GAV, then the Ith row in the Jacobian matrix (JM) will have but one nonzero entry and it will occur in the Jth column. Since LISTE gives the GAE index for each LAE, one of the elements of this sector will have a value of unity and one will have a value of I. Exchanging the contents of the locations having these values is equivalent to interchanging the Ith row with the first row in the JM. Furthermore, since LISTV gives the GAV index for each LAV, at least one of the elements of this sector will have a value of unity and at least one will have a value of J. If each location in this sector that has a value of unity is given a value of J and vice versa, then the next evaluation of the JM will exhibit an interchange of the old first and Jth columns.

If both the LISTE and LISTV transformations are carried out and the JM is re-evaluated, there will be a nonzero value in the first position of the first row of the JM and zeroes in the other positions of that row. The elements of the first column however will not necessarily be zero since the first GAV (formerly the Jth) may
occur in other GAEs. Now the first GAE (formerly the Ith) can be solved separately from the others and if this is done, then the first GAV can be considered as a known parameter in the other GAEs, hence the dimension of the coupled algebraic system can be reduced by one.

If another GAE, say the Mth, in the reduced system also depends on one and only one GAV, say the Nth, the twos and Ms can be exchanged in the LISTE sector and the twos and the Ns can be exchanged in the LISTV sector. After the re-evaluation, the first two rows in the JM will have zero elements to the right of the diagonal. If the first element in the second row is nonzero, then this means that both $Y_1$ and $Y_2$ (formerly $Y_J$ and $Y_N$) occur in the expression for $G_2$ (formerly $G_M$) but since the value of $Y_1$ can be determined by solving $G_1(Y_1)=0$, $G_2$ can be considered as dependent only on $Y_2$. In general, after KCO decouplings the diagonal elements of the first KCO rows will be nonzero, the elements to the right of these diagonal elements will be zero and the elements to the left may be zero or nonzero.

The reduction of the JM's dimension by identifying singly decoupleable equations is the object of the subroutine DECUP which will be described in the following. Referring to figure 18 it is seen that before entering the loop that starts at box 17, the values of the three
initialize flags

ENTRY

ICODE = 1, INST = 3, ICALL = 3, KCO = 0, KCO = 0, ITURN = 1

IDIM = ICTE + 1, IPROD = ICTE*IDIM, IPDZ = IPKCO + IPROD
IPTM = IPDZ + ICTV

call ZEROX zero Jacobian matrix area

call JACO evaluate Jacobian matrix

KROW = 0, ISTA = IPKCO - IDIM

have looked at all rows

KROW = KROW + 1 increment row counter

NNZ = 0, ISTA = ISTA + IDIM count number of nonzero elements in KROWth row

DO 10 I = 1, ICTE

DUM = ABS(FPVEG(ISTA + I))

DUM > 10^-15

NNZ = NNZ + 1 MEL = I

FIGURE 18: DECUP
have found a single nonzero element in KROWth row and MELth column
KCO = KCO + 1
ISA = IP0 - 2*NLAVT + 1

call SWTCH(ISA, NLAVT, IVA, KCO, IVEC)
ISA = ISA - NLAVT

call SWTCH(ISA, NLAVT, IEQN, KCO, IVEC)

reconstruct GAVBR sector
KCO: KCO ≤

no new nonzero elements found in last pass

reduce dimensions of Jacobian area

call GLOB
ICTE = ICTV - KCO
IPKCO = IPPZ + 2*KCO
KCO = KCO

FIGURE 18. (continued)
Construct LIST3

FIGURE 18. (continued)
flags, ICALL, INST, and ICODE, that determines the flow of control in the NSR (see figure 7), are set. Also, KCO, the number of singly decoupled equations is initialized to zero as is KCOP which will be used to indicate changes in KCO. Finally, the flag ITURN, which plays a part in the subroutine TRANS logic (to be discussed later), is set.

The boundaries of the DGDY sector that contains the elements of the augmented JM are set up, the area is zeroed by a call to ZEROX, and the JM is evaluated by a call to JACO. Each row of the JM is now scanned for non-zero elements. If, after the KROW-th row has been scanned, the variable NNZ is found to be unity, the scanning stops. At this point the column index of the nonzero element has been assigned to MEL, hence, if KCOP equations had already been decoupled, the GAE index, IEQN, and the GAV index, IVA, for the newly discovered decouplable equation can be determined. KCO is now incremented and, in order to exchange the contents of the appropriate locations in LISTV and LISTE, two calls are made to the subroutine SWITCH. Since KCO is now not equal to KCOP, subroutine GLOB is called and the GAVDR sector is reconstructed.

Next, the dimension of the JM is corrected and IPKCO, the pointer for the portion of the DGDY sector to be used by the new JM is computed. Note that, as mentioned in part B of chapter IV, IPPZ is the pointer for the first
portion of the DGDY sector, consisting of 2*KCO locations, which is used to store the function values and the appropriate derivatives that are necessary in the application of the one dimensional Newton-Raphson method. Now control is returned to box 17 where the JM will be evaluated and scanned again for rows with only one non-zero element. Each time such an element is found, the scan stops, the GAV and GAE indices are switched, KCO is incremented, the JM dimension is lowered by one, a new smaller JM is evaluated, and another scanning commences. This process continues until all the singly decouplable equations are identified. Note that the decoupling of one equation may cause another GAE, which previously was defined in terms of two GAVs, to become decouplable because one of the GAVs now can be treated as a known parameter.

Finally, control is transferred to box four where, if KCO is nonzero LIST3 will be constructed as follows. Starting with the first decoupled GAE index, one, every node's subsector of LISTE is scanned until a value of unity is found. The index of the node, in whose subsector the match is made, is placed in the first position in the LIST3 sector. This scan is then repeated and LIST3 is added to until all of the values, one through KCO, have been matched. It is apparent then that one node index can appear more than once in the LIST3 sector. Note that
at the completion of the SED process the original algebraic system of ICTV and GAEs has been reduced to a system of KCO singly decoupled GAEs and ICTE coupled GAEs; hence ICTV = KCO + ICTE.

As an example, consider the augmented JM displayed below in rectangular form where the "1" symbols refer to nonzero elements and the eighth column contains the values of the functions $G_i$, $i=1,\ldots,ICTV$.

\[
\begin{array}{cccccccc}
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\
1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\
2 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\
3 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \\
4 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\
5 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
6 & 0 & 1 & 0 & 1 & 0 & 1 & 1 \\
7 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\
\end{array}
\]

It is seen that three of the seven GAEs have been decoupled leaving a reduced system of four coupled GAEs; therefore KCO = 3 and ICTE = 4. In the DGDY sector only the elements of the JM that are used by the Newton-Raphson method are stored linearly in the following order: $J_{11}, J_{18}, J_{22}, J_{28}, J_{33}, J_{38}, J_{44}, J_{45}, \ldots, J_{48}, J_{54}, J_{55}, \ldots, J_{58}, J_{64}, J_{65}, \ldots, J_{68}, J_{74}, J_{78}, \ldots, J_{78}, J_{84}, \ldots, J_{88}$. Note that the elements $J_{31}$ and $J_{62}$, although nonzero, are not stored since they are not necessary for the solution of the equations. In the next section where the disjoint subsystem analysis is discussed, only the ICTE\textit{x}ICTE portion
of the JM associated with the ICTE coupled GAEs will be of interest and this portion will be called the reduced JM. Note that the pointer IPKCO would point at the J38 element since that element is in the last location in DGDY before the area which stores the reduced JM.

F. Disjoint Subsystem Decoupling

After the SED operations have been completed, subroutine SBSYS attempts to identify disjoint groups of coupled algebraic equations and reorder the GAEs and GAVs so that the JM is transformed into block diagonal form with each block representing one subgroup. For example, the following augmented JM

\[
\begin{array}{cccccccc}
1 & 2 & 3 & 4 & 5 & 6 \\
1 & 0 & 0 & 1 & 0 & 1 \\
2 & 0 & 0 & 1 & 0 & 1 & 1 \\
3 & 0 & 0 & 1 & 0 & 1 & 1 \\
4 & 0 & 1 & 0 & 1 & 0 & 1 \\
5 & 0 & 0 & 1 & 0 & 1 & 1 \\
\end{array}
\]

would be transformed into

\[
\begin{array}{cccccccc}
1 & 2 & 3 & 4 & 5 & 6 \\
1 & 0 & 1 & 0 & 0 & 1 \\
2 & 0 & 1 & 1 & 0 & 0 & 1 \\
3 & 1 & 0 & 1 & 0 & 0 & 1 \\
4 & 0 & 0 & 0 & 1 & 1 & 1 \\
5 & 0 & 0 & 0 & 1 & 1 & 1 \\
\end{array}
\]

which represents two algebraic subsystems that would be solved separately. The elements of the transformed augmented JM would be stored in the following order:
J11, J12, J13, J16, J21, J22, J23, J26, J13, J23, J33, J36, J44, J45, J46, J54, J55, J56, i.e., only the elements needed in the numerical solution are stored.

The transformation is accomplished by scanning each column and making, for each column, a list of the rows having nonzero elements. Two lists are combined whenever a row number occurs in both, meaning that there is a GAV that occurs in both sets of equations represented by the lists. When the scanning and list combining are completed, if there is more than one list, then the equations represented by each list are defined in terms of a set of GAVs none of which occur in any of the other equation. Hence, at this point, disjoint sets of GAEs have been identified.

Next, the GAEs and GAVs must be re-indexed so that the JM will be transformed into block diagonal form. This is done in two steps. First, the \( N_1 \) GAE indices on the first list are exchanged with the numbers \( KCO+1 \) through \( KCO+N_1 \) in the LISTE sector so that the first \( N_1 \) rows of the reduced JM will be associated with the equations represented by the first list. Next the \( N_2 \) GAE indices on the second list are exchanged with the numbers \( KCO+N_1+1 \) through \( N_1+N_2+KCO \) in the LISTE sector, and so on for each list.

At the end of this process the rows of the JM will be in the correct order but the columns will not. At this point the JM is re-evaluated with the new GAE indices
and each row of the reduced JM is searched for nonzero elements. When the first nonzero element is found, say, in the Jth column, the indices KCO+1 and KCO+J are exchanged in the LISTV sector. The GAVDR sector is then reconstructed, the JM re-evaluated, and the search along the same row is continued. When a row is finished the search begins on the next row at the (K+1)th column position where K is the number of times the GAVDR sector has been reconstructed. When the second nonzero element is found, say, in the Lth column, the indices KCO+2 and KXO+L are exchanged in the LISTV sector, the GAVDR sector is reconstructed, the JM is re-evaluated, and the search is continued until all of the rows have been scanned.

In the following paragraphs a more detailed exposition on the workings of the SBSYS subroutine will be given by going through the logic shown in figure 19. To simplify the presentation it will be assumed that program control has just been transferred to SBSYS. As in DECUP, the three flags ICODE, INST, and ICALL, that determine the flow of control in the NSR, and the flag ITURN, used by TRANS, must be set. Next, the DGDY sector is zeroed and the JM evaluated. Third, pointers to the first elements in the LISTV and LISTE sectors (ISV and ISE) are computed and finally the boundaries of the LTMP sector are set up.
ENTRY

INST = 3, ICODE = 1, ITURN = 1, ICALL = 3

call ZEROX

zero Jacobian matrix area

pointers to LISTV and LISTE sectors

ISV = IPG - 2*NLAVT + 1
ISE = ISV - NLAET

ICTP = ICTE + 1, ITMP1 = IPTMP + 1, NG = 1
IPSSG = IPTMP + ICTE, ISTO = IPSSG + ICTP

JCOL = 0

ITMP1 = IPTMP + 1, IPTMP = IPTMP + 1, NG = 1
IPSSG = IPTMP + ICTE, ISTO = IPSSG + ICTP

JCOL = JCOL + 1

have finished columns

JCOL = ICTE

zero LTMP sector

NNZ = 0, IB = IPTMP
IA = IPKCO + JCOL - IDIM

look down JCOLth column and construct LTMP sector

DO 100 I = 1, ICTE

IA = IA + IDIM, DUM = ABS(FPVEC(IA))

DUM: 10^-15 <

NNZ = NNZ + 1, IVEC(IPTMP + NNZ) = I

100

FIGURE 19. SBSYS
JCOL: 1

IA = IPSSG + 1

IFT = IA + 1, IVEC(IA) = NNZ, IB = IPTMP

DO 101 I = 1, NNZ

IA = IA + 1, IB = IB + 1, IVEC(IA) = IVEC(IB)

ISTO = IPSSG

IA = IPSSG - ICTP, K = 0, KSW = 0

K = K + 1

K: NO

IA = IA + ICTP

LN = IVEC(IA + 1), IST = IA + 2

call TLST(ITMP1, NNZ, IST, LN, IVEC, ICO)

ICO: 1

no match so go to next group

FIGURE 19. (continued)
add elements of LTMP to Kth group list

call ADDL(IPTMP1, NRZ, IST, LN, IVEC)

KSW = K

KSW: 0 ≠ 8

17

elements of LTMP have already been added to KSWth group list

ISA = IST, LA = LN, ISB = IPSSB + (KSW - 1) * ICTP + 2
LB = IVEC(ISB - 1)

call ADDL(ISA, LA, ISB, LB, IVEC)

add Kth group to KSWth group

zero group K locations

call ZERONX

NS = NG, NG = NG - 1

K: NS

zeroed group was last group so go back and make new LTMP

IENDA = IPSSG + (K - 1) * ICTP + 1 + ICTE
ISC = IENDA + 1, LC = (NS - K) * ICTP

call SHOVE(ISC, LC, ICTP, IVEC)

shove remaining groups left by ICTP locations

start on a new column

7 have finnished comparing LTMP with group lists

KSW: 0 ≠ 2

5

FIGURE 19. (continued)
get ready to create new group

\[ \text{NG} = \text{NG} + 1, \quad \text{ISO} = \text{IPSSG} + \text{NG} \times \text{ICTP} \]
\[ \text{ISA} = \text{ISO} + \text{ICTE} \]

call ZEROX

zero area for new group list

go to group list creating algorithm

\[ \text{MEQN} = \text{KCO} + 1, \quad \text{KVAR} = \text{MEQN} \]
\[ \text{ING} = 0 \]

call ZEKOX

have finished constructing group lists, now modify LISTE

\[ \text{IA} = \text{IPSSG} + 1 \]

have finished all groups

go to next group list

\[ \text{IA} = \text{IFT} + 1, \quad \text{L} = \text{IVEC}(\text{IA}), \quad \text{LP} = \text{KCO} + \text{L} \]

call SWITCH(\text{ISE}, \text{NLAET}, \text{LP}, \text{MEQN}, \text{IVEC})

switch LP and MEQN in LISTE

\[ \text{MP} = \text{MEQN} - \text{KCO}, \quad \text{IA} = \text{IPSSG} + 1 \]

now switch L and MP in all of the group lists

\[ \text{DO} 110 \quad \text{JNG} = 1, \text{NG} \]
\[ \text{INJ} = \text{IVEC}(\text{IA}), \quad \text{ISJ} = \text{IA} + 1 \]

call SWITCH(\text{ISJ}, \text{INJ}, \text{L}, \text{MP}, \text{IVEC})

\[ \text{IA} = \text{IA} + \text{ICTP} \]

\[ \text{110} \]
\[ \text{MEQN} = \text{MEQN} + 1 \]
\[ \text{IA} = \text{IPSSG} + 1 \]

FIGURE 19. (continued)
have rearranged rows of Jacobian matrix, now rearrange columns

\[
KSW = KCO, \quad KSWP = KSW + 1 \\
L = 0
\]

\[
L = L + 1
\]

\[
L; ICTE \\
J = KSW
\]

\[
J = J + 1
\]

\[
J; ICTV
\]

\[
IA = IPXCO + (L-1)*IDIM + J - KCO \\
X = \text{ABS}(FPVEC(IA))
\]

\[
X: 10^{-16}
\]

have found GAV now see if it is already in correct position

\[
\text{call SWITCH(ISV,NLAVT,KSWP,J,IVEC)}
\]

\[
IA = ISV + NLAVT - 1
\]

\[
KSW = KSW + 1 \\
KSWP = KSW + 1
\]

switch KSWP and J in LISTV

\[
\text{call GLOB}
\]

\[
\text{call ZEROX}
\]

\[
\text{call JACO}
\]

FIGURE 19. (continued)
call OCCUR

IA = IPSSG + 1, IB = IPTMP + 1
IPLI = IPTMP + NG, IDUM = IPLI + 2*NG

3 = 0

I = I + 1

I: NG

IVEC(IB) = IVEC(IA)
IA = IA + ICTP, IB = IB + 1

call ZEROX

ILOC = IpKoo, LOW = Koo
LN = 0, I = 0

I = I + 1

I: NG

LN = IVEC(IPTMP + 1)

ILOC = IVEC(IB - 1) = IDUM

IPDZ = ILOC

IP = IPDZ + ICTV
IPPO = IDUM
ICTE = LN

RETURN

FIGURE 19. (continued)
After the contents of the LTMP sector are set to zero, a scan is made of the first column of the first column of the JM and the numbers of rows having nonzero elements are placed in LTMP which therefore becomes a list of GAEs in which $Y_{KCO+1}$ occurs.

The reader should remember that a previous call to DECUP has identified the first KCO GAVs as belonging to singly decoupled equations, hence the first GAV that can be considered on the scan is $Y_{KCO+1}$ rather than $Y_1$ unless of course if no GAEs were decoupled in which case KCO would be zero. Since this is the first column that has been scanned there are no other group lists with which LTMP should be compared, so program control is transferred to box five where the first "group list" is constructed from the elements of LTMP and where NG, the number of group lists, is set to unity. This so-called group list is at this point simply a list of GAE indices of those GAEs in which $Y_{KCO+1}$ occurs, i.e., identical in content to the LTMP sector. After the scanning process is completed, each group list will contain the GAE indices of a disjoint set GAEs.

Referring to figure 20, note that the first element of each group list sector is its number of non-zero members and the following locations will contain the row
FIGURE 20. GROUP LIST SECTORS
indices. Each group list sector has a reserved length of 
ICTE+1 where ICTE is the dimension of the reduced JM cor-
responds to the (KCO+I)th GAE. Control is now returned 
to box two where the second column is scanned and a new 
LTMP sector is constructed. The LTMP sector is now a list 
of the GAE indices of those GAEs in which $Y_{KCO+2}$ occurs.
At the end of this scan, control is transferred to box 
four where preparations are made for the contents of the 
LTMP sector to be compared with the elements of each of 
the other (so far only one) group lists.

The variable K is the group list number and KSW will 
be set equal to K if a match is made between the Kth 
group list and LTMP. The actual comparison is done by 
subroutine TLST which assigns a value of unity to the flag 
ICO if a match has been made. A match will be made if any 
of the GAE indices in group list one is also in group list 
two or vice versa; then either $Y_{KCO+1}$ or $Y_{KCO+2}$ or both 
occur in both sets of GAEs and therefore the sets are not 
disjoint and the group lists should be combined. Assume 
however that there is no match between LTMP and the group 
one list, i.e., that the two sets of GAEs are so far dis­
joint. Therefore, since NG is one, control is transferred 
to box seven where, since KSW is zero, preparations such 
as incrementing NG to two are made for the construction of
a second group list.

The group list is then constructed at box five and the scan on the third column begins. As a result of this scan a new LTMP sector (a list of the GAE indices of those GAEs in which $Y_{KCO+3}$ occurs) is constructed and it is compared in turn with group lists one and two. Assume that there is a match with group list one. Since KSW is zero, subroutine ADDL is called and the non-matching elements of LTMP are added to group list one. KSW is now set to unity and LTMP is next compared to group list two. If there were no match, indicating that there are now two disjoint sets of GAEs, as represented by the two group lists, control would eventually be transferred to box two and a fourth scan started.

Assume however that a match is made between LTMP and group list two. Since KSW is not zero, control is transferred to box 17 where the elements of group two are added to group one and then destroyed since now, because of the matchings, there are no longer two disjoint sets of GAEs. NG is decremented to one and K is compared with the former value of NG. Since K is two and NG was two, the zeroed group list was the last group and control is returned to box two where the fourth column scan commences. However, if NG were greater than two, say four, then the zeroed group list would not have been the last and the remaining
two group lists would be shoved left filling in the locations previously used by group list two. This would be accomplished by a call to SHOVE after which control would be transferred to box 18 where LTMP would be compared with the remaining two group lists. This process will be continued until all columns have been scanned and NG disjoint sets of GAEs have been identified at which point control is transferred to box three where preparations are made for re-ordering the GAE indices.

Suppose the first element of the first group list is L. Since this is a row index in the reduced JM, KCO is added to L to get LP, the GAE index. Subroutine SWTCH is then called and LP and KCO+1 are switched in LISTE. Next, the values of L and one are switched in all of the group lists by NG calls to SWTCH. The appropriate counters are incremented and the process is repeated for the second element of the first list. This process is then repeated for all the elements in all the lists and after this is finished, control is transferred to box 10 where the JM is transferred to box 20 where preparations (such as computing a value of IDUM which will point to the as yet unconstructed node list for group one) are made for the construction of the DIM and PAIRS sectors (see figure 9). First, the lengths of each group list are transferred to successive locations in the DIM sector which now occupies
FIGURE 21. NODE LISTS
the area formerly known as the LTMP sector (see figure 21).

The PAIRS sector is constructed next. As the reader will remember from the discussion in part A of chapter three, the PAIRS sector consists of NG pairs, one for each subgroup, and each pair consists of the pointer to that group's node list and the pointer to the area of the DGDY sector used by that group's AJM. The construction of the PAIRS sector proceeds as follows. First, the area formerly occupied by the group lists is zeroed. Next, the pointer ILOC is assigned the value of IPKCO which points to the area of the DGDY sector used by the disjoint subgroups and the counter LOW is given the value of KCO which is the index of the last of the singly decoupled GAEs. The dimension of the first subgroup, LN, is now fetched from DIM. This group contains LN AGES whose indices range from LOW+1 through LHIGH where LHIGH=LOW + LN. The augmented JM for this subsystem will need LN*(LN+1) locations which will start at the location following the one pointed to ILOC. Hence, the value of ILOC is assigned to the second element of the first pair and then incremented for the next group.

The first element of the first pair is given the value of IDUM, which is the pointer for the group one node list which will be constructed in a subsequent call
to CNLST. Like ILOC, IDUM is then incremented for the next group. The call to CNLST makes a list of the nodes that supply the values of the GAEs in the first group. Control is now returned to box 15 where the process is repeated for the next group. After all of the NG groups have been processed there will be, in addition to the DIM and PAIRS sectors, NG sectors containing the node lists that will allow the GAEs to be evaluated.

G. Utility Routines

In the foregoing discussion reference has frequently been made to several programs that carry out specialized functions. In the following, the logic of these routines, as represented by their flow diagrams or listings, will be briefly discussed.

1. CNLST

The function of this routine whose flow diagram is shown in figure 22 is to construct a list, starting at location LSTRT in IVEC, of those nodes that have LIAEs with GAE indices greater than LOW and less than or equal to LHIGH. This function is accomplished by fetching, for each node in turn, the pointer to its subsector of the LISTE sector and comparing the contents of the subsector with LOW and LHIGH. If one of the GAE indices is within
FIGURE 22. CNLST(ISTR, LOW, LHIGH, IVEC)
the specified limits the node index is added to the list.

2. ADDL

This routine (see figure 23) is called several times by SBSYS and its function is to add the elements of list A, starting at location ISA in IVEC and having a length of LA, to list B which starts at ISB and has a length of LB. Each element of list A is first tested to see if it is already on list B by calling SRN4. If it is not then it is added to the end of list B and the length of list B, which is stored just before the first element, is increased by one.

3. TLST and SRN4

These two routines are so simple that only the listings need be consulted. The function of TLST is to fetch each element in turn from list A which starts at ISA and has LA elements, and test it to see if it matches an element on list B, which starts at ISB and has LB elements, by calling SRN4. Subroutine SRN4 in turn consists simply of a FORTRAN DO loop which tests one element against the elements of a specified list.

4. GLOB

The function of GLOB is to construct the GAVDR sector whose ith element gives the location in FPVEC where the
FIGURE 23. ADDL(ISA, IA, ISB, LB, IVEC)
value of the $i$th GAV is stored. Referring to figure 24, it is seen that this task is accomplished by fetching, for each node, the values of NLAV and the pointers to its subsectors of LISTV and LAVDR. Then, for each of that node's NLAV LAVs, the GAV index, IGLOB, is fetched from LISTV. If it is less than or equal to zero then that LAV is not matched with a GAV and therefore is ignored. If it is greater than zero then the location in FPVEC of the LAV's value is fetched from the LAVDR sector and assigned to the IGLOBth position in the GAVDR sector.

5. JACO

The function of this routine is to supervise the evaluation of the elements of the DGDY sector. The reader can refer to part B of section III for a discussion of this sector's structure. Because of the simplicity of the routine, reference will be made only to the FORTRAN listing which shows that JACO consists of a DO loop where for each node index from one to NEMAX (or NEUB) the following steps are performed. First, subroutine FETCH is called which fetches the values of NLIAE, NLAV, IDER, NLDE, IBPA and IBIP for node NE and places them in COMMON. If NLIAE is zero then this node can not provide any values for the DGDY sector so NE is incremented. If NLIAE is not zero, then LOAD is called which places the correct
ENTRY

DO 3 NE = 1, NMAX
NLAV = IVEC(IPAV + NE)

NLAV: 0 → 3

IB1 = IVEC(IPLV + NE), IB2 = IVEC(IPYID + NE)

DO 3 I = 1, NLAV
IGLOB = IVEC(IB1 + I)

IGLOB: 0 → 3

ILOC = IVEC(IB2 + I), IVEC(IPG + IGLOB) = ILOC

RETURN

FIGURE 21. GLOB
values of the LAVs and LDVs in the y and X sectors. Sub-routine ECALL is then called and control is transferred to the NSR which describes the NE-th node and values of the local quantities $g_i$ and $\partial g_i/\partial y_j$ are computed as functions of the LAVs and LDVs and placed in the $g$ and $dg/dy$ sectors. If IDER is equal to zero then a call is made to PERT which perturbs the LAVs and estimates values of the partial derivatives. If IDER is unity then the NSR has already calculated the derivatives from analytical expressions. Finally, TRANS is called to transfer the contents of the $g$ and $dg/dy$ sectors to the DGDY sector and NE is incremented for the next node.

6. LOAD

The function of LOAD is to load the X and y sectors with the correct values of the NLDE LDVs and the NLAV LAVs for node NE. Referring to figure 25 it is seen that to load the y sector the pointer to node NE's subsector of LAVDR is fetched and assigned to IBL. Then for each of the LAVs the location of its value, IARG, is fetched from the LAVDR sector and used to place the correct value in the appropriate location of the y sector. The same operations are repeated for the X sector using the LDVDR sector. Now when control is transferred to the NSR the local quantities $f_i,g_i$, and $\partial g_i/\partial y_j$ can be computed.
ENTRY

NLAV: 0

IBL = IVEC(IPYID + NE)

DO 13 I = 1, NIAV

IARG = IVEC(IBL + I)
FPVEC(IPY + I) = FPVEC(IARG)

13

NLDE: 0

RETURN

IBL = IVEC(IPXID + NE)

DO 18 I = 1, NLDE

IARG = IVEC(IBL + I)
FPVEC(IPX + I) = FPVEC(IARG)

18

RETURN

FIGURE 25. LOAD
as functions of the LAVs and LDVs since their values can be obtained from sectors y and x.

7. TRANS

The function of TRANS is to transfer the values of the local quantities $q_i$ and the partial derivatives (that have been placed in sectors $g$ and dgdy by the NSR that describes node NE) to the correct positions in the DGDY sector.

There are two versions of TRANS: one for the analysis package and one for the compute package which will be discussed later. They are different because the latter must take into account which of the subsystems the value transfer is for, while the former does not need to recognize subsystems. In addition, the analysis package version has several unnecessary features because it attempts to place values for the singly decoupled as well as the coupled equations when it is really only necessary to transfer values of the partial derivatives to that portion of the DGDY sector used by the coupled GAEs during the period of analysis when the subsystems are being identified. Therefore in future editions the analysis package version of TRANS could be considerably simplified.

Referring to figure 26 it is seen that upon entry, the pointers to node NE's subsector of LISTE and LISTV are
ENTRY

ISTRE = IVEC(IPL + NE), ISTRV = IVEC(IPL + NE)
LEN = 0

1

LEN = LEN + 1
LEN ≤ NLA

IL = IVEC(ISTRE + LEN)
IL = 0

IL = KCO and ITURN = 1?
yes

do not evaluate decoupled GAEs

no

IL = KCO and ITURN = 0?
yes
do not evaluate coupled GAEs

no

evaluate coupled GAEs

ITURN = 1

IARH = IPTHL + LEN

IL = KCO

IARG = IPFZ + 2*IL

IARG = IPKCO + (IL-KCO)*IDIM

FPVEC(IARG) = -FPVEC(IARH)

ICODE = 1

no inversion so no more transfer

FIGURE 26. TRANS
LVARN = 0, IARJ = IPPT - NLIAE

IARJ = IARJ + NLIAE, LVARN = LVARN + 1

LVARN ≥ NLA

IARI = ISTRV + LVARN, J1 = IVEC(IARI)

J1: 0

I1: KCO

J1: II

IARI = IARJ + LEN
FPVEC(IARG-1) = FPVEC(IARI)

IARH = IARG - IDIM + J1 - KCO
IARI = IARJ + LEN
FPVEC(IARH) = FPVEC(IARI)

FIGURE 26. (continued)
fetched and the local equation counter, LEN, is initialized. Once into the loop that starts at box one, LEN is incremented and if it is not greater than NLIAE the associated GAE index Il is fetched from LISTE. As will be pointed out in the discussion of the compute package, the flag ITURN determines whether the singly decoupled or the coupled equations are being evaluated by taking on a value of zero or unity, respectively. The reader will remember that both DECUP and SBSYS which require Jacobian matrix evaluations set ITURN to unity since there is no interest in the singly decoupled equations. Therefore if Il is not greater than KCO and if ITURN is unity then the value of $g_{\text{LEN}}$ and the partial derivatives of $g_{\text{LEN}}$ will be ignored and control will be transferred back to box one. The same thing will happen if Il is greater than unity and ITURN is zero. If neither of these conditions obtains then if ITURN is one, control is transferred to box seven where the coupled equations are evaluated.

At box seven the appropriate position, IARG, in the DGDY sector is determined and the sign of $g_i$ is changed before its value is placed in FPVEC(IARG). Note that the placement in the DGDY sector differs depending on whether or not a singly decoupled equation is being evaluated. Next, if ICODE is unity, the NLAV partial derivatives of $g_{\text{LEN}}$ will also be placed in the DGDY sector. To do this a
loop is set up wherein for each LAV the GAV index $J_l$ is fetched. Then because IARG (the location where $g_{\text{LEN}}$ was placed) is known, the appropriate location in DGDY for $\frac{\partial g_{\text{LEN}}}{\partial y_{\text{LARN}}}$ can be computed. After the partial derivative is placed in the proper position the process is repeated for the next LAV and so on until it has been done for all the LAVs. Then control is returned to box one where the whole process is repeated for the next LIAE.

Essentially then, knowledge of the GAE and GAV indices is sufficient for the transfer of the local quantities from the $g$ and dgdy sectors to the proper positions in the DGDY sector.

8. PERT

The function of this routine is to numerically estimate the NLAV*NLIAE partial derivatives associated with node NE. Referring to figure 27 it is seen that this task is accomplished as follows. In a preliminary move the contents of the $g$ and $y$ sectors are transferred to the g-save and y-save sectors. There are the current unperturbed values of the LIAEs and LAVs about which the perturbations will be made.

Next, a loop is set up in which the following operations are carried out for each of the NLAV LAVs. First, the GAV index is fetched and only if it is greater than
ENTRY
EPS = DUT(3)

DO 19 I = 1, NLIAE
FPVEC(IPES + I) = FPVEC(IPTHL + I)

DO 20 I = 1, NLIAV
FPVEC(IPYS + I) = FPVEC(IPY + I)

IARK = IPPY - NLIAE
DO 21 J = 1, NLIAV

IARK = IARK + NLIAE
II = IVEC(IPLV + ME) + J1
II = IVEC(I1)

II: 0 <=
ABS(YS): 10^-5 <

DY = YS*EPS

DY = EPS

FPVEC(IPY + J) = YS + DY

call ECALL

FPVEC(IPY + J) = YS

DO 25 K = 1, NLIAE
FPVEC(IARK + K) = (FPVEC(IPTHL + K) - FPVEC(IPES + K))/DY

DO 24 I = 1, NLIAE
FPVEC(IPTHL + I) = FPVEC(IPES + I)

RETURN
zero will that LAV be perturbed. Second, the size of the perturbation in $y_j$, $DY$, is calculated in one of two ways depending on the magnitude of $y_j$. Third, the perturbed value of $y_j$ is placed in the $y$ sector. Fourth, subroutine ECALL is called and control is transferred to the NSR that describes node NE where a fresh set of values will be computed and placed in the $g$ sector. When control is returned to PERT the old value of $y_j$ is returned to the $y$ sector.

At this point, an inner loop is set up wherein the old values of the NLIAE LIAEs are subtracted from the values resulting from the most recent NSR calculation and the difference is divided by $DY$. This operation yields estimates of $\partial g_i/\partial y_j$ for $i=1,...,NLIAE$, and these values are placed in the appropriate portion of the $dgdy$ sector. Now control is returned to the beginning of the outer loop where the operations are repeated for the next LAV. After all of the LAVs have been perturbed the values of the $g$ sector are restored from the $g$-save sector and control is returned to the calling program.

H. Recommendations

The analysis of the global algebraic system can be improved and extended. Presently, the first step consists of decoupling single algebraic equations. In the following system of equations,
\[ G_1(Y_1) = 0 \]
\[ \vdots \]
\[ G_n(Y_1, Y_n) = 0 \]
\[ \vdots \]

the decoupling of the first equation makes possible the decoupling of the nth equation; therefore the decoupled equations are necessarily ordered. The second step consists of identifying disjoint subsystems of algebraic equations. Since these systems are disjoint they are not ordered. This is the extent of the analysis.

The next step would logically consist of attempting to decompose each disjoint subsystem into an ordered set of coupled subsystems. For example, consider the 9x9 system for which the occurrence matrix (OM) is shown in figure 28. This system can yield neither singly decoupled equations nor disjoint subsystems; however it can be decomposed into an ordered set of coupled subsystems as will be seen in the following discussion. First, change the order of two equations and two variables and obtain the OM shown in figure 28. Here the 2x2 system in the upper left hand corner of the OM can be solved for \( Y_1 \) and \( Y_4 \) which in turn allow equations 1 and 3 to be singly decoupled. This leaves a 5x5 system which can be further decomposed into a 2x2 and a 3x3 system as shown in figure 28c. Finally, the 3x3 system can be decomposed to a singly
<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

(a)

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>4</th>
<th>2</th>
<th>3</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

(b)

<table>
<thead>
<tr>
<th></th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

(c)

**FIGURE 28. OCCURRENCE MATRICES FOR EXAMPLE PROBLEM**
decoupled equation and a 2x2 system as shown in Figure 28d. Therefore, by permuting the GAV and GAE indices, the original 9x9 system can be decomposed into an ordered set of coupled subsystems as shown in figure 28e.

On the other hand there can be too much decoupling. This would occur if, for example, one node was described by n singly decoupled equations. Since each equation is solved separately and iteratively there may be several calls to that node's NSR for each equation. As can be seen from figure 7, there is presently no way to prevent the calculation of all algebraic equations and their partial derivatives on each call, hence much computing time may be lost because of the decoupling.

One way to partially overcome this problem would be to place a call to the subroutine DECI (flow diagram shown in figure 29) in the NSR just before the calculations associated with equation. Subroutine DECI checks to see if the ILAEth equation should be evaluated and sets a flag (ISIG) accordingly. After returning control to the NSR, control could be transferred around the calculations if that particular equation is not to be evaluated. This logic might also be incorporated into the PERT routine so that only the necessary perturbations would be executed when each equation or subsystem is being solved.
FIGURE 29. DECI(ILAE, IVEC, ISIG)
I. Concluding Remarks

This chapter has discussed the structure of the analysis package in great detail. It is seen that the function of this program is to prepare a simulation problem for computation. For a given simulation problem the input to the analysis package consists essentially of information on the topological structure of the digraph which models the simuland and information pertaining to the nature of the individual nodes and streams which make up the digraph. At the conclusion of an analysis package execution, cards are punched containing information on the mathematical structure of the system to be simulated. These cards form the bulk of the input to the compute package which carries out the actual mathematical simulation and which will be discussed in chapter VII. In effect then the analysis package receives information on a system in a form easily compiled by the user and transforms it into a form easily assimilated by the compute package.
CHAPTER VI
THE ANALYSIS OF THREE EXAMPLE PROBLEMS

In this chapter, three example problems that have actually been analyzed by the analysis package will be discussed. In Chapter VIII the results of applying the compute package to these same examples will be discussed.

A. A Linear Multiple Recycle System

This example, consisting of a moderately complex collection of three very simple types of nodes (mixers, splitters, and accumulators), provides the reader with an introduction to NSR construction and input data arrangement, and illustrates the decoupling features of the analysis package.

1. The Digraph

The digraph, consisting of 21 nodes (three accumulators, nine mixers, and nine splitters) and 31 streams, each being defined in terms of a single quantity which can be interpreted as a fluid flow rate, is shown in figure 30. The nodes are depicted as numbered boxes with M signifying a mixer, S a splitter, and A an accumulator. The streams are shown as directed numbered lines connecting two nodes.

178
FIGURE 30. EXAMPLE NO. 1 DIGRAPH
Before discussing the nodes in detail a few comments are in order. First, note that the two main recycle branches, starting with streams 22 and 23, are identical in nodal composition so the flows in streams 20 and 31 should be identical. Second, since the mixers and splitters are described only by algebraic relations, the recycle loops associated with nodes 2, 3, 4, 5 should contribute an algebraic subsystem as should the loops associated with nodes 9, 10, 11, 12, and 16, 17, 18, 19.

2. Node Models

The simplest node model is the mixer wherein $y_3$, the flow rate of the outlet stream (local stream number three) is the sum of the flow rates $y_1$ and $y_2$ associated with the inlet streams (local streams one and two). If this relation were expressed explicitly the modeler would have to number the nodes so that, on the pass that evaluates the LEAVs, the inlet quantities would be computed before the outlet quantities. Because of this restriction and because of the interest in seeing how the analysis package could handle the implicit relations the node expression was written as

$$g_1 = y_1 + y_2 - y_3$$  \hspace{1cm} (16)

The MXR subroutine describes this node and referring to its listing (MXR number one in the appendix) it is seen that,
although consistent with figure 7, the logic is simpler because NLDE, IDER, and NLEAV are zero. Since each of the three LAVs (all external) are associated with different streams, three calls to MKLST must be made. Eq. 16 is coded after statement three by using the pointers IPTHL for \( g_1 \) and IPY for \( y_1, y_2, \) and \( y_3 \). An alternate listing (MXR number two in the appendix) of the MXR subroutine shows how the coding is modified when IDER is not zero, i.e., when the partial derivatives

\[
\frac{\partial g_1}{\partial y_1} = 1, \quad \frac{\partial g_1}{\partial y_2} = 1, \quad \frac{\partial g_1}{\partial y_3} = -1
\]

are supplied by the NSR.

The splitter node model, described by the SPTR NSR, consists of two equations

\[
\begin{align*}
    g_1 &= y_1 - y_2 - y_3 \\
    g_2 &= y_3 - f^*y_1
\end{align*}
\]

where \( y_1 \) is the inlet stream (local stream no. 1) flow rate, \( y_2 \) the first outlet stream (local stream no. 2) flow rate, and \( y_3 \) the second outlet stream (local stream no. 3) flow rate. The node parameter, \( f \), is the fraction of the flow that leaves via the second outlet stream and, as shown in the NSR listing, is stored in the first and only location of the node's subsector of FPAR.

The accumulator node model described by the SFTK NSR,
is defined by the following two equations
\[
\frac{dx_1}{dt} = f_1 = y_1 - x_1/R
\]
\[
g_1 = y_2 - x_1/R
\]
where \( y_1 \) and \( y_2 \) are the inlet and outlet flows while \( x_1 \) is an artificial quantity indicative of accumulated fluid and \( R \) is a parameter. The logic is slightly more complicated here since NLDE is not zero, so COMPX must be called for the internal LDV. The subsector of FPAR for this node will consist of four locations: one for the value of the parameter \( R \), and three for the value, lower and upper bound for \( x_1 \). Note that unlike the MXR and SPTR NSRs the outlet flow rate \( y_2 \) would be best treated as a class A LEAV.

Each of these three models have been coded and tested with the outlet quantities treated as explicit algebraic variables. For example, the explicit version of the splitter model is described by the ND3 NSR and like the other NSRs mentioned so far, its listing is shown in the appendix. Note that since there are only 17 NSR names available, some of them will be used by more than one model.

3. Arrangement of the Input Data

Figure 31 shows a listing of the analysis package input data for the 21 node, 31 stream system whose digraph
21 NODE - 31 STREAM LINEAR SYSTEM ANALYSIS

LIST5=10*0, 1, 1, 18*0, CUT=2*0, 5, 12*0.

<table>
<thead>
<tr>
<th>Node</th>
<th>Node</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>1</td>
<td>9</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>11</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>10</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>11</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>7</td>
<td>3</td>
</tr>
<tr>
<td>7</td>
<td>9</td>
</tr>
<tr>
<td>8</td>
<td>3</td>
</tr>
<tr>
<td>8</td>
<td>12</td>
</tr>
<tr>
<td>9</td>
<td>12</td>
</tr>
<tr>
<td>22</td>
<td>14</td>
</tr>
<tr>
<td>10</td>
<td>3</td>
</tr>
<tr>
<td>14</td>
<td>18</td>
</tr>
<tr>
<td>11</td>
<td>3</td>
</tr>
<tr>
<td>16</td>
<td>15</td>
</tr>
<tr>
<td>12</td>
<td>3</td>
</tr>
<tr>
<td>17</td>
<td>18</td>
</tr>
<tr>
<td>13</td>
<td>2</td>
</tr>
<tr>
<td>19</td>
<td>20</td>
</tr>
<tr>
<td>14</td>
<td>3</td>
</tr>
</tbody>
</table>

**FIGURE 31. INPUT DATA TO ANALYSIS PACKAGE, EXAMPLE NO. 1**
<table>
<thead>
<tr>
<th>15</th>
<th>20</th>
<th>28</th>
<th>17</th>
<th>26</th>
<th>18</th>
<th>24</th>
<th>19</th>
<th>23</th>
<th>20</th>
<th>13</th>
<th>21</th>
<th>29</th>
<th>-9</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>31</td>
<td>27</td>
<td>3</td>
<td>25</td>
<td>3</td>
<td>27</td>
<td>3</td>
<td>24</td>
<td>3</td>
<td>22</td>
<td>2</td>
<td>31</td>
<td>1</td>
</tr>
</tbody>
</table>

**FPAR SECTOR**

---

<table>
<thead>
<tr>
<th>5</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>12</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

*FIGURE 31. (continued)*
FIGURE 31. (continued)
FIGURE 31. (continued)
<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>28</td>
<td>4</td>
<td>50</td>
</tr>
<tr>
<td>29</td>
<td>4</td>
<td>50</td>
</tr>
<tr>
<td>30</td>
<td>4</td>
<td>50</td>
</tr>
<tr>
<td>31</td>
<td>4</td>
<td>-1</td>
</tr>
<tr>
<td>-9</td>
<td></td>
<td>50</td>
</tr>
</tbody>
</table>

FIGURE 31. (continued)
is shown in figure 30. The first card contains the title which can consume all 80 columns if necessary. The next three cards define the values of the vectors LIST5 and DUT via FORTRAN Namelist statements. Since both SED and ASI are desired LIST5(11) and LIST5(12) are both unity and since the analysis package makes no use of the other elements of LIST5 these are the only nonzero elements. Likewise, DUT(3), which is the fractional step size used by the PERT routine, is the only nonzero element in the DUT vector.

The values of NEUB and NSUB are read from the fifth card. The next group of cards contain the data that is read into the KPM sector. The reader can refer to the discussion in part G of chapter III and to part B of chapter V for a detailed explanation of the data order. Note that, on the cards containing the node index, columns 41 through 80 can be used for comments by the modeler.

The second group of cards contains FPAR subsectors for the nodes described by the SFTK and SPTR NSRs. Note also that for all of the splitters the fraction f is chosen as .5 while for each of the three accumulators the resistance R is 9.0 and the initial value of the LDV is zero.

The third group of data contains the STRMV subsectors for the 31 streams. Since there is only one stream
quantity, viz., the flow, each subsector consists of only four positions and except for stream 30 the initial value of the flows is zero as is the matching code. The whole system can be considered to be forced by a constant nonzero flow in stream 30 so the initial value here is unity and the code is minus one.

4. Results of the Analysis

The output of the analysis package consists of about 1000 printed lines and 161 punched cards therefore only a summary will be given here. On the pass supervised by SRN7, three LDVs associated with nodes 6, 13, 21, were identified. On the pass supervised by SETUPS, 30 GAVs and GAEs were identified and the occurrence matrix (OM) had the form shown in figure 32. Eighteen of the 30 GAEs were identified as singly decouplable during the operations supervised by DECUP after which the OM had the form shown in figure 31. Finally, three subsystems, each of dimension four, were identified during the operations supervised by SBSYS, after which the OM had the form shown in figure 33.

The DGDY sector will consist of 36 locations for the 18 singly decoupled GAEs, and 60 locations for the three disjoint subsystems. If there had been no decoupling, 930 locations would have been needed in the DGDY sector to
FIGURE 32. INITIAL OCCURRENCE MATRIX
FIGURE 33. CM AFTER SINGLE EQUATION DECOUPLING

FIGURE 34. CM AFTER SUBSYSTEM DECOUPLING
store the elements of the augmented Jacobian matrix. This problem will need 701 locations in IVEC and 321 in FPVEC when the compute package is executed.

Figure 35 shows a listing of the 39 elements of the labeled common area called POINT followed by the elements of the IVEC vector sans the IPOS sector. To show how the user can analyze the punched output consider the first LAV associated with node three. To determine the GAV index this local variable the following steps must be taken. First, IPLV, the pointer to the director sector for the LISTV sector which contains the GAV indices, is seen to have the value of 116 since it is the fifth element of the POINT common area. The pointer to node three's sub-sector of LISTV will then be the 119th element of IVEC which appears in the 19th position of the 8th row of IVEC in figure 35. The value of this element is seen to be 490 hence the GAV index of the first LAV associated with the third node will be the 491st element of IVEC which appears in the 11th position of the 27th row in figure 35. The value of this element is seen to be 17, consequently the LAV is matched with the 17th GAV.

Figure 36 shows a listing of the first 146 elements of FPVEC which make up the FPAR and STRMV sectors. Consider the single flow variable which defines stream five.
FIGURE 35. INTEGER OUTPUT FROM ANALYSIS PACKAGE, EXAMPLE NO. 1
<table>
<thead>
<tr>
<th>107</th>
<th>111</th>
<th>135</th>
<th>143</th>
<th>47</th>
<th>51</th>
<th>55</th>
<th>71</th>
<th>67</th>
<th>55</th>
<th>111</th>
<th>119</th>
<th>115</th>
<th>107</th>
<th>75</th>
<th>75</th>
<th>143</th>
<th>103</th>
<th>23</th>
<th>27</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>63</td>
<td>87</td>
<td>95</td>
<td>91</td>
<td>63</td>
<td>59</td>
<td>43</td>
<td>35</td>
<td>131</td>
<td>127</td>
<td>135</td>
<td>123</td>
<td>13</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>13</td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>19</td>
<td>19</td>
<td>20</td>
<td>9</td>
<td>5</td>
<td>21</td>
<td>15</td>
<td>14</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>661</td>
<td>219</td>
<td>673</td>
<td>239</td>
<td>685</td>
</tr>
<tr>
<td>259</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>C</td>
<td>0</td>
<td>C</td>
</tr>
<tr>
<td>0</td>
<td>C</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

FIGURE 35. (continued)
FIGURE 36. REAL OUTPUT FROM ANALYSIS PACKAGE, EXAMPLE NO. 1
To determine its initial value, its bounds and its matching code, the following steps must be taken. First, IPST, the pointer to the director sector for the STRMV sector is seen to have the value of 64 since it is the eleventh element of the POINT common area. The pointer to stream five's subsector of STRMV will then be the 69th element of IVEC which appears in the 9th position of the 6th row of IVEC in figure 35. The value of this element is seen to be 38 hence the first member of the stream five subsector which contains only 4 elements will be the 39th element of FPVEC which appears in the 4th position of the 8th row in figure 36. Here it is seen that the initial value of the stream five variable is zero, that its bounds are zero and 50.0 and that it is matched with the 8th GAV.

Note that the listings that compose figures 35 and 36 are simply listings of the cards punched by the analysis package.

It will probably appear to the reader that the process of interpreting the punched card output from the analysis package is at best tedious. To aid the user in this process a supplementary program has been written that reads the analysis package punched card output and prints a detailed interpretation of the contents of the punched cards. However it is expected that with the aid of just
the printed output from the analysis package the user will develop the ability to rapidly interpret the punched card output after a few runs and therefore he will rarely bother to use the supplementary program. For this reason there is no further discussion of the supplementary program in this thesis.

B. A Solids Drying System with Feedforward and Feedback Control

This example illustrates the use of a source or forcing node, a transport lag node, two types of controller nodes and a multi-lump approximation to a distributed system.

1. The Digraph

The digraph shown in figure 37 consists of seven nodes. Node one represents a cylindrical solids-air dryer where the wet solids enter the dryer via stream one and leave via stream two. These two streams are defined in terms of three variables: the moisture fraction, solids temperature, and solids flow rate. The drying air flows counter to the solids and enters via stream three and leaves via stream four. These two streams are defined in terms of the air humidity, air temperature, and air flow
FIGURE 37. DIGRAPH FOR THE DRYER SYSTEM
rate. To aid the drying process, steam enters (via stream six) a jacket that effectively surrounds the cylinder. Stream six is defined only in terms of a steam temperature. The value of the steam temperature is set by node five which is a PID controller acting in response to a set point communicated from node three by means of stream six. Node three merely adds the feedforward component of the set point coming from node six (by means of stream ten) to the feedback component coming from node two via stream seven.

Node two is a feedback PID controller that receives the solids moisture fraction of stream two, compares it to the set point supplied by stream nine and produces a controller signal via stream seven. Node four receives the solids flow rate from stream one and computes a desired steam temperature set point which is transmitted via stream eight to node six which delays the signal a prescribed amount before it is sent to node three where it is added to the feedback component. The system is forced by node seven which is set up in this example to vary the solids flow rate. Hence streams one, two, three and four are each defined in terms of three variables while streams five, six, seven, eight, nine and ten are defined in terms of a single variable.
2. Node Modules

a. Source Node

The source node, described by the ND2 NSR, effects system-forcing by calculating (or setting) values of class B LEAVs associated with the node's output stream during the prediction phase (see part D of chapter III for a discussion of the order of the passes). Let \( S(t) \) be a vector of forcing functions at time \( t \). Now, on the LDV prediction pass through the nodes, values of \( f(x^C(T-h), y^C(T-h)) \) are calculated and used to produce values of \( x^P(T) \). Hence, the output of the source node should be \( S(T-h) \).

On the subsequent correction pass that will yield \( x^C(T) \), values of \( f(x^P(T), y^P(T)) \) are calculated, so in preparation for this pass the outpost of the source node, \( S(T) \), should be evaluated just before this pass commences but after all other calculations associated with the prediction phase are completed. Since class B LEAVs are evaluated last in the five step sequence associated with the prediction phase it is possible to obtain the proper sequence of calculations; however it is critical that the source node number be higher than any other node having class B LEAVs except for lag nodes which will be discussed below. Now, on the class B LEAV evaluation pass following the correction operations, the source
node LAVs are not evaluated since on the subsequent prediction operations at time $T + h$, values of $f(x^c(T),y^c(T))$ will be calculated and the source node output will have to be $S(T)$ which it already is.

Referring to the listing of subroutine ND2, it is seen that the forcing function value is placed in the IPSIG-th position of the subsector of the STRMV sector associated with the node's outlet stream. The value of IPSIG is stored in the first position in the node's subsector of IPAR. Note first, that the names IV and FP are used in place of IVEC and FPVEC for the sake of brevity and second, that II and IR, the pointers to the node's subsector of IPAR and FPAR, are available in the labeled common area named BASE.

The forcing function is calculated only if $T$ is not equal to zero and, if $T$ is zero, the output will be the initial value. Similarly, the flag INST must be four, indicating a class B LEAV evaluation pass, and IPC must be zero or minus one indicating that calculations are being carried out during the prediction phase (the flag IPC is minus one during the first prediction phase, zero during all other prediction phases, and plus one during all correction phases and will be discussed in chapter VI). Finally, since LIST5(4) is used to store the iteration
number for the algebraic system calculations and the explicit calculations, program control is returned to the calling routine if LIST5(4) is greater than one.

The forcing function for this particular source node is

\[ y_1 = \text{ref} + \text{ampl} \times \sin(2\pi \frac{t}{\text{per}} + \text{phase}) \]

where the four parameters \( \text{ref}, \text{ampl}, \text{per}, \) and \( \text{phase}, \) are stored in the node's subsector of the FPAR sector.

b. Transport Lag Node

In the following algorithm, a transport lag of \( M \times h \) time units is effected for \( N \) quantities by left-shifting \( N \) lists, each of length \( M \):

\[
\begin{align*}
Y_{N+1}^{(t)} &= y_1(t-M \times h) \\
Y_{N+2}^{(t)} &= y_2(t-M \times h) \\
&\vdots \\
Y_{N+N}^{(t)} &= y_N(t-M \times h)
\end{align*}
\]

where the first \( N \) LAVs are associated with the inlet stream and the second \( N \) with the outlet stream. As with the source node, the outlet stream LAVs, \( y_{N+1}, \ldots, y_{2N}, \) are classified as class B LEAVs and the left-most values are extracted from the \( N \) lists and assigned to these \( N \) LAVs only during the prediction phase. The \( N \) lists are also pushed left at this time. However, the right-most elements
of the lists are updated only during the correction phase after all the LAVs and LDVs have taken on their final corrected values for that time step.

As an example, consider figure 38 where node C is a lag with a delay of 2*h and stream B is defined by one variable, say an external LDV associated with node A. Hence, for node C, \( y_1 \) is the input from stream B and \( y_2 \) is the stream D variable. Figure 38 shows that the initial values of \( y_1 \) and \( y_2 \) are \( a(0) \) and \( d \). Since \( y_1 \) is the LDV from node A, steps one in both the prediction and correction phase will alter its value. In step five of the prediction phase, the left-most value is extracted from the push-left list and assigned to \( y_2 \) then the list is pushed left. At this time the right-most position in the list is undefined and not until step five of the correction phase is the value of \( y_1 \) placed there.

It should be obvious to the reader that this sequence will also be correct when the stream B variable is a class A or B LEAV or an implicit algebraic variable associated with node A. If, for example, the stream B variable should be a class B LEAV associated with node A, then in general the node number of node C must be higher than that of node A. However, in the source-lag example shown in figure 39, the node numbering is unimportant because the
<table>
<thead>
<tr>
<th>t</th>
<th>phase</th>
<th>step</th>
<th>$y_1$</th>
<th>list</th>
<th>$y_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
<td>$a(0)$</td>
<td>$b, c$</td>
<td>$d$</td>
</tr>
<tr>
<td>$h$ pred</td>
<td>1</td>
<td></td>
<td>$a^P(h)$</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$b$</td>
<td>$c$</td>
</tr>
<tr>
<td>corr</td>
<td>1</td>
<td></td>
<td>$a^c(h)$</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$a^c(h), b$</td>
<td></td>
</tr>
<tr>
<td>$2h$ pred</td>
<td>1</td>
<td></td>
<td>$a^P(2h)$</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$b$</td>
<td></td>
</tr>
<tr>
<td>corr</td>
<td>1</td>
<td></td>
<td>$a^c(2h)$</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$a^c(2h), a^c(h)$</td>
<td></td>
</tr>
</tbody>
</table>

**FIGURE 38. TRANSPORT LAG**
\begin{figure}
\begin{center}
\includegraphics{figure39.png}
\end{center}
\end{figure}

\begin{table}
\begin{tabular}{|c|c|c|c|c|}
\hline
\textbf{t} & \textbf{phase} & \textbf{step} & \textbf{\(y_1\)} & \textbf{\(y_2\)} \\
\hline
0 & & & a(0) & d* \\
\hline
h & pred & 1 & a(h) & b, c \\
\hline
 & & 5 & a(h), b & c* \\
\hline
 & corr & 1 & a(2h) & b \\
\hline
 & & 5 & a(2h), a(h) & b* \\
\hline
2h & pred & 1 & a(3h) & a(h) \\
\hline
 & & 5 & a(3h), a(2h) & a(h)* \\
\hline
 & corr & 1 & a(3h) & a(h) \\
\hline
3h & pred & 1 & a(3h) & a(h) \\
\hline
 & & 5 & a(3h), a(2h) & a(h)* \\
\hline
\end{tabular}
\end{table}

\textbf{FIGURE 39. SOURCE - LAG EXAMPLE}
output of the source node is evaluated on step five of the prediction phase while the right-most element of the list is not modified until step five of the correction phase. The asterisk indicates which value would be plotted or printed.

Referring to the listing of the LAG NSR, it is seen that M and N are stored in the node's IPAR subsector pointed to by II. The N push-left lists, each containing M elements, are stored sequentially in the node's FPAR subsector just after the value of TAU, which is the delay time. Note that this parameter is not used in the calculations. As with the source NSR, calculations are carried out only if T is greater than zero, INST is four (a class B LEAV evaluation pass), and LIST5(4) is one (the first iteration). If IPC is equal to or less than zero (prediction phase) the left-most element of each of the N lists is extracted and assigned to the appropriate LAV. Then the lists are shoved left one element. If IPC is greater than zero (correction phase), the current values of the input stream LAVs are placed in the right-most positions of the lists.

c. The Adder Node

This node which is described by the NSR MXR and which adds the values of the two inlet streams and assigns the
sum to the outlet stream, has been discussed in detail in
the previous example.

d. Servo Controller

The object of this node is to provide a standard propor­
tional-integral-derivative control response to a change
in set point according to the block diagram in figure 40,
which shows one inlet stream (local stream no. 1) that
transmits the set point and one outlet stream (local
stream no. 2) which transmits the controller response.
Let $y_1$ be the set point, $x_1$ the measured output signal,
and $x_3$ the node's output signal, then the error signal is
\[ e = y_1 - x_1 \]
where $x_1$ is determined from
\[ \frac{dx_1}{dt} = \frac{x_3 - x_1}{\tau_m} = f_1 \]
and $x_3$ from
\[ \frac{dx_3}{dt} = \frac{y_2 - x_3}{\tau_L} = f_3 \]
where $y_2$ is the PID output. By definition, the idealized
PID controller action is
\[ y_2(t) = y_2(0) + K_c [e(t) + \tau_D e(t) + \frac{1}{\tau_i} \int_0^t e(v) \, dv] \]
Now define $x_2$ as
FIGURE 140. SERVO CONTROLLER NODE
\[ x_2(t) = \frac{K_C}{\tau_I} \int_0^t dv \ e(v) + y_2(0) \]

or

\[ \frac{dx_2}{dt} = \frac{K_C}{\tau_I} \ e(t) = f_2 \]

\[ x_2(0) = y_2(0) \]

and specify that the initial value of \( x_2 \) is \( p(0) \), which is the initial value of the PID output can be written

\[ y_2(t) = x_2(t) + K_C [e(t) + \tau_D \dot{e}(t)] \]

where the derivative is defined as a simple backward difference

\[ \dot{e}(t) = \frac{e(t) - e(t-h)}{h} \]

Hence this model is described by three LDVs and two LAVs one of which is a LEAV.

Referring to the listing of the LLCN NSR, which implements the model, it is seen that the output of the node is placed in the IPX3-th position of the subsector of the STRMV sector associated with the output stream and the set point value is extracted from the IPY1-th position of the input stream's subsector of STRMV. Positions 16 through 19 of the node's subsector of FPAR comprise a four element push-left list which is used to store a history of the errors for use in the derivative computation although the
version of LLCN shown here uses only the top two locations. The list is pushed left once per time step after the prediction operations. Note that \( y_2 \), the PID output is interpreted as a class A LEAV under the assumption that the value of the set point \( y_1 \) which affects the error calculation will be constant. If this is not the case then \( y_2 \) might be better treated as a class B LEAV.

In other versions of LLCN, the output of the PID has been interpreted as a LDV:

\[
\frac{dx}{dt} = k_c [e + T_D \cdot \dot{e} + \frac{\ddot{e}}{T_I}]
\]

but difficulties were encountered in evaluating the second derivative.

e. Feedback Controller

As figure 41 indicates, this node model is similar to the servo controller; however, the object here is to provide a PID response to an external signal after it has been compared to the set point. Let \( y_1 \) be the set point which is communicated to the node by local stream number one, and let \( y_2 \) be the measured value transmitted to the node via local stream number two. After a first order measurement lag
FIGURE 14. FEEDBACK CONTROLLER
\[
\frac{dx_1}{du} = \frac{y_2 - x_1}{\tau_m} = f_1
\]

the error is computed from \( e = y_1 - x_1 \). As with the servo controller the output of the PID algorithm is written as

\[
y_3(u) = x_2(u) + K_C[e(u) + \tau_D e(u)]
\]

Finally there is a first order lag between the PID output and the node output:

\[
\frac{dx_3}{du} = \frac{y_3 - x_3}{\tau_L} = f_3
\]

Referring to the listing of the HETR NSR, which describes the model, it is seen to be so similar in concept and construction that further comment is unnecessary.

f. Solids Dryer

Although this model is for a hypothetical piece of equipment, the simuland can be considered to be a jacketed cylindrical shell shown in figure 42 where wet solids enter via local stream number one and exit via stream number two. Dry air enters the opposite end via local stream three and leaves via stream four. Finally steam enters the jacket via local stream five. The shell rotates and this action
FIGURE 42. SOLIDS DRYER
in conjunction with the pitch of the shell causes the solids to move from left to right. The dependent variables are shown in table 1.

Since the solids moisture and temperature will vary with axial position as well as with time, this simuland might be modeled by a system of partial differential equations where the independent variables would be time and axial position. However, the node description in OSUSIM does not admit partial differential equations; therefore the solids dryer of length L is divided into N lumps of length \( z = L/N \).

Figure 43 shows the result of the division where the dependent variables now have the subscript of the lump or stream with which they are associated. Note that \( M_k \) refers to the bone dry solids hold-up associated with the kth lump. Also associated with the kth lump is \( T_w^k \), the temperature of the wall separating the solids and air mixture from the steam, and \( T_s \), the temperature of the steam in the jacket which effectively surrounds the lump. For this model the steam jacket temperature is considered constant for every lump so there is no subscript. To develop equations which can be solved for these dependent variables, mass and enthalpy balances are made over each lump. First, a bone dry solids mass balance gives
FIGURE 43. N-LUMP APPROXIMATION TO SOLIDS DRYER
### TABLE 1.

DEPENDENT VARIABLES FOR THE SOLIDS DRYER

<table>
<thead>
<tr>
<th>Variable</th>
<th>Symbol</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>Solids flow rate</td>
<td>M</td>
<td>16 BDS/hr</td>
</tr>
<tr>
<td>Moisture</td>
<td>X</td>
<td>16 H₂O/16 BDS</td>
</tr>
<tr>
<td>Temperature</td>
<td>T</td>
<td>°R</td>
</tr>
<tr>
<td>Air flow rate</td>
<td>G</td>
<td>lb BDA/hr ft²</td>
</tr>
<tr>
<td>Humidity</td>
<td>Y</td>
<td>lb H₂O/lb BDA</td>
</tr>
<tr>
<td>Temperature</td>
<td>t</td>
<td>°R</td>
</tr>
<tr>
<td>Steam temperature</td>
<td>Tₛ</td>
<td>°R</td>
</tr>
<tr>
<td>Wall temperature</td>
<td>Tᵥ</td>
<td>°R</td>
</tr>
</tbody>
</table>

BDS = Bone Dry Solids  
BDA = Bone Dry Air
and a water balance on the solids gives

\[ X_{k-1} \dot{M}_{k-1} - X_k \dot{M}_k - W_k = \frac{d}{du} (M_k X_k) \]  

(18)

where \( W_k \) is the lbs of water per hr transferred to the air. Combining Eq. 17 with Eq. 18 gives

\[ \frac{dX_k}{du} = -\frac{W_k}{M_k} + \frac{\dot{M}_{k-1}}{M_k} (X_{k-1} - X_k) \]  

(19)

which replaces Eq. 18. Next, an enthalpy balance on the solids stream yields

\[ \dot{M}_{k-1} h_{k-1} - \dot{M}_k h_k - W_k H_w(T_k) - Q_{TW} - Q_{TA} = \frac{d}{du} (M_k h_k) \]  

(20)

where \( H_w(T_k) \) is the enthalpy of the water leaving the solids

\[ H_w(T_k) = \lambda + C_{P_{wv}} (T_k - T_o) \]  

(21)

and where \( h_k \) is the enthalpy in BTU/lb BDS

\[ h_k = C_{P_{BDS}} (T_k - T_o) + X_k C_{P_{w1}} (T_k - T_o) \]  

(22)

Note that most of the quantities occurring in these equations are defined in table 2. Combining Eqs. 19, 20, 21, and 22, yields, after some manipulation, the following expression

\[ \dot{M}_{k-1} (C_{P_{BDS}} + X_{k-1} C_{P_{w1}}) (T_{k-1} - T_k) - W_k \lambda \]

\[ - (Q_{TW} + Q_{TA}) + W_k (T_k - T_o) (C_{P_{w1}} - C_{P_{wv}}) \]
\[ = M_k(C_{P_{BDS}} + X_k C_{P_{wl}}) \frac{dT_k}{du} \]  

An energy balance over the wall of the jacket gives

\[ \frac{dT_{wk}}{du} = \frac{Q_{SQ} + Q_{AW} + Q_{TW}}{C_{P_{w}} \Delta \rho_{w} \pi D_w z} \]  

**TABLE 2. VARIABLE DEFINITION FOR SOLIDS DRYER**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q_{TW}</td>
<td>Energy transfer from solids to wall</td>
<td>BTU/hr</td>
</tr>
<tr>
<td>Q_{TA}</td>
<td>Energy transfer from solids to air</td>
<td>BTU/hr</td>
</tr>
<tr>
<td>Q_{SW}</td>
<td>Energy transfer from steam to wall</td>
<td></td>
</tr>
<tr>
<td>Q_{AW}</td>
<td>Energy transfer from air to wall</td>
<td></td>
</tr>
<tr>
<td>C_{P_{BDS}}</td>
<td>Heat capacity of BDS</td>
<td>BTU/hr °F</td>
</tr>
<tr>
<td>C_{P_{wl}}</td>
<td>Heat capacity of liquid water</td>
<td></td>
</tr>
<tr>
<td>C_{P_{wv}}</td>
<td>Heat capacity of water vapor</td>
<td></td>
</tr>
<tr>
<td>\lambda</td>
<td>Heat of vaporization of water</td>
<td>BTU/hr</td>
</tr>
<tr>
<td>C_{P_{w}}</td>
<td>Heat capacity of wall</td>
<td></td>
</tr>
<tr>
<td>\Delta R</td>
<td>Thickness of jacket wall</td>
<td></td>
</tr>
<tr>
<td>P_{w}</td>
<td>Density of wall</td>
<td></td>
</tr>
<tr>
<td>D_{w}</td>
<td>Diameter of jacket wall</td>
<td></td>
</tr>
<tr>
<td>C_{P_{BDA}}</td>
<td>Heat capacity of bone dry air</td>
<td></td>
</tr>
<tr>
<td>T_{o}</td>
<td>Reference temperature</td>
<td></td>
</tr>
<tr>
<td>t_{o}</td>
<td>Reference temperature (H_{2}O B.P.)</td>
<td></td>
</tr>
<tr>
<td>h_{AT}</td>
<td>Transfer coefficient for solids to air</td>
<td></td>
</tr>
<tr>
<td>h_{AW}</td>
<td>Transfer coefficient for air to wall</td>
<td></td>
</tr>
</tbody>
</table>
A water balance on the air gives

\[ G_A F (Y_{k+1} - Y_k) + w_k = 0 \]  \hspace{1cm} (25)

where \( G \) is considered constant for every lump and \( A_F \) is the cross sectional area seen by the air. As can be seen from Eq. 25, the dynamic response of the air is considered to be at least an order of magnitude quicker than that of the solids so, to avoid local stiffness, the air is considered to be at steady state relative to the solids. In effect then, the air responds directly to the solids fluctuations. Solving Eq. 25 for the lump outlet humidity gives

\[ Y_k = Y_{k+1} + \frac{w_k}{G_A F} \]  \hspace{1cm} (26)

An enthalpy balance on the air gives

\[ G_A F [H_{k+1} - H_k] + Q_{WA} + Q_{TA} + w_k H_w = 0 \]  \hspace{1cm} (27)

where \( H_k \) is the enthalpy of the air:

\[ H_k = C_{PBDA} (t_k - t_0) + Y_k (\lambda + C_{P_{wv}} (t_k - t_0)) \]

and where the dynamics of the air have again been ignored relative to those of the solids. As with the water balance
on the air, Eq. 27 is solved for the lump outlet air temperature

\[
\begin{align*}
G_A p (C_p + y_{k+1} C_p) & T_{k+1} + h_A T M K T_k + w K C_p T_k + h_A W T w_k \\
& = G_A F (C_p + y_{k+1} C_p) + h_A T M K + w K C_p + h_A W
\end{align*}
\]

(28)

At this point there are 4*N ODEs and 2*N explicit algebraic expressions describing the solids dryer. However there are 7*N dependent variables:

\((Y_k, t_k, X_k, M_k, T_k, Tw_k, M_k^*)\), \(k=1, \ldots, N\)

while the following seven variables will either have fixed values or will be determined by other nodes:

\(X_0, \dot{M}_0, T_0, T_s, Y_{N+1}, T_{N+1}, G\).

To complete the description of the node the following relation for the solids flow rate (as it leaves the lump) is assumed:

\[
\dot{M}_k = \frac{M_k}{d}
\]

(29)

where \(d\) is the average lump residence time. Therefore the node is now locally solvable.

Next, the interphase transfer terms must be discussed. First, the moisture transfer term is assumed to have the following form:

\[
W_k = K * M_k^* (X_k - X_e)
\]

(30)

where \(X_e\) is a hypothetical interface moisture in equilibrium with the air flowing past at that point:
\[ X_e = R H_k (a_1 + a_2 T_k) \]  

Eq. 31 defines a linear relationship between the equilibrium moisture and the relative humidity, \( R H_k \), which is given by

\[ R H_k = \frac{Y_k/(R+Y_k)}{Y_s/(R+Y_s)} \]

where \( R \) is the ratio of the molecular weight of water to that of air, and \( Y_s \) is the saturation humidity of the air given by

\[ Y_s = b_1 + b_2 T_k \]

The proportionality constant \( K \) in Eq. 30 is assumed to be a function of the solids temperature as follows:

\[ K = c_1 + c_2 T_k \]

Second, the four energy transfer terms are given in table 3 where it is seen that three of the transfer terms are dependent on \( z \), the length of the jacket wall associated with the lump. Involved in the expressions for the energy transfer terms are six constants, \( \bar{c}_{16}, \bar{c}_{30}, \bar{c}_{13}, \bar{c}_{11}, \bar{c}_{14}, \) and \( \bar{c}_{17} \). The barred constants occur in transfer terms that depend on \( z \) while the unbarred constants occur in transfer terms that are independent of the jacket wall length.

Now it is necessary to identify the dependent variables as either LAVs or LDVs (see figure 44). Associated with the inlet solids stream are \( X_0, T_0, \) and \( M_0 \) which
FIGURE 4.1. LAV AND LDV IDENTIFICATION FOR SOLIDS DRYER
TABLE 3. ENERGY TRANSFER TERMS

\[ Q_{TW} = h_{TW}A \cdot (T_k - T_{wk}) \]
\[ h_{TW}A = [\overline{c}_{16} \times 10^{2} + \overline{c}_{30}(T_k - 560)]2\pi Rz \]

\[ Q_{TA} = h_{AT}M_k \cdot (T_k - t_k) \]
\[ h_{AT} = c_{13} + c_{11} \times 10^{-3}(t_k - 630) \]

\[ Q_{WA} = h_{AW}A \cdot (T_s - T_{wk}) \]
\[ h_{AW}A = \overline{c}_{14} \cdot 2\pi Rz \]

\[ Q_{SW} = h_{SW}A \cdot (T_s - T_{wk}) \]
\[ h_{SW}A = \overline{c}_{17} \cdot 2\pi Rz \]

\[ R = \text{Shell radius} \]

are identified as \(y_1\), \(y_2\), and \(y_3\). Referring to the listing of the NSR TANK, which describes the solids dryer node, it is seen that one call to MKLST is made for these three variables. Associated with the outlet solids stream are \(x_N\), \(T_N\), and \(M_N\). The first two are the external LDVs \(x(3 + 4*(N-1))\) and \(x(4 + 4*(N-1))\) for which one call to COMPX is made. The third variable, \(M_N\), is identified as \(y_4\) for which one call to MKLST is made.

Associated with the outlet air stream are \(Y_1\), \(t_1\), and \(G\) which are identified as \(y_5\), \(y_6\), and \(y_7\) and for which one call to MKLST is made. The inlet air stream variables \(Y_{N+1}\), \(t_{N+1}\), and \(G\) are identified with \(y(6+2*N)\) through \(y(8+2*N)\) for which one call to MKLST is made. The last
of the external variables is the local stream five steam temperature $T_s$ which is identifies as $y(9+2*N)$ and for which one call to MKLST is made.

Now the $N-1$ humidity-air temperature pairs, $(y_k, t_k)$, $k=2,\ldots,N$, are identified as $(y(4+2*k), y(5+2*k))$, $k=2,\ldots,N$, for which one call to MKLST is made. The value, lower bound, upper bound, and code for each of these $2*(N-1)$ internal LAVs are stored sequentially in the last $8*(N-1)$ locations of the node's subsector of the FPAR sector.

Next, the $N-1$ wall temperature, lump hold up, solids moisture, solids temperature quadruples, $(T_{w_k}, M_k, T_k, X_k)$, $k=1,\ldots,N-1)$, plus the wall temperature and hold up of the Nth lump, $T_{w_N}$ and $M_N$, are identified as the $4*N-2$ LDVs, $x(1)$ through $x(2+4*(N-1))$, for which one call to COMPX is made. The value, lower bound and upper bound of each internal LDV is stored in the $4*N-4$ locations just before the area used by the internal LAVs.

Note that the $N-1$ solids flow rates, $M_k$, $k=1,\ldots,N-1$ are not identified as either a LAV or a LDV since whenever needed they can be calculated by means of Eq. 29. Consequently the node is described in terms of $2*N+9$ LAVs, of which $2*N+2$ are LEAVs, and $4*N$ LDVs. The first three LAVs will either be fixed in value or determined by other nodes. The fourth LAV will be a class A LEAV since it can be
calculated from Eq. 29. LAVs five, six, and eight through 5+2*N are also class A LEAVs and are calculated from Eqs. 26 and 28. The last four LAVs will either be fixed in value or determined by other nodes. For the four LDVs per lump, $T_w, M_k, X_k, T_k$, Eqs. 24, 17, 19, and 23, suitably normalized, yield the values of the $f_i$.

The listing of TANK shows that $N$, the number of lumps, is stored in the node's subsector of IPAR while the first six positions in the node's subsector of FPAR contain the four coefficients in the energy transfer terms, $c_{16}, c_{30}, c_{14}, c_{17}$, that are lump size independent, plus the value of $z$ and $d$. The next two areas of the node's subsector of FPAR contain information on the internal LAVs and LDVs as mentioned above.

On the pass where the calls to MKLST are made, the four energy transfer coefficients are fetched and modified, depending on the lump size, in preparation for the simulation calculations. On the computation pass, a loop is set up where for each lump, starting with lump one, values of the lump's inlet and outlet quantities are fetched, then $h_{AT}, h_{TW}$ and $w_k$ are calculated and a transfer of control is made depending on the value of INST. If INST is one, then the four $f_i$ values are determined and, if INST is two, the two $y_i$ values are calculated. In
evaluating the two class A LEAVs per lump, local iteration will be required since the right hand sides of Eqs. 26 and 28 depend weakly on the LEAVs themselves, hence these equations can be called semi-explicit.

e. Feed Forward Computation Node

A desired steam temperature for the dryer steam jacket, \( y_2 \), is computed as a function of the solids flow rate, \( y_1 \), which is transmitted to the stream via local stream number one. The relation between \( y_1 \) and \( y_2 \) is written in implicit form:

\[
g_1 = -y_2 + 0.140054(y_1 - 7800.) + 700.
\]

Referring to the listing of the TMCN NSR, which describes this node, it is seen that the value of \( y_1 \) is fetched from the ninth location in the subsector of the STRMV sector associated with local stream number one. This is because the solids stream is defined in terms of the moisture fraction, the temperature, and the flow rate. The value of \( y_2 \) is stored in the first position in the subsector of the STRMV sector associated with local stream number two since this stream is defined by only one variable.

3. Arrangement of the Input Data

Figure 45 shows a listing of the cards that make up the input data. There are to be sight lumps in the dryer
<table>
<thead>
<tr>
<th>STRM</th>
<th>FEED</th>
<th>SCLICS</th>
<th>STRM</th>
<th>FEED</th>
<th>SCLICS</th>
<th>STRM</th>
<th>FEED</th>
<th>SCLICS</th>
<th>STRM</th>
<th>FEED</th>
<th>SCLICS</th>
<th>STRM</th>
<th>FEED</th>
<th>SCLICS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>23456</td>
<td>0.</td>
<td>1</td>
<td>-1</td>
<td>56C</td>
<td>0</td>
<td>12000</td>
<td>-8</td>
<td>-1</td>
<td>56C</td>
<td>0</td>
<td>-1</td>
<td>56C</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0.205</td>
<td>0.</td>
<td>1</td>
<td>-1</td>
<td>56C</td>
<td>0</td>
<td>12000</td>
<td>-2</td>
<td>630</td>
<td>0</td>
<td>1000</td>
<td>-2</td>
<td>630</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0.011</td>
<td>0.</td>
<td>0</td>
<td>-1</td>
<td>630</td>
<td>0</td>
<td>10000</td>
<td>0</td>
<td>630</td>
<td>0</td>
<td>1000</td>
<td>-2</td>
<td>630</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0.011</td>
<td>0.</td>
<td>1</td>
<td>-2</td>
<td>630</td>
<td>0</td>
<td>10000</td>
<td>0</td>
<td>630</td>
<td>0</td>
<td>1000</td>
<td>-2</td>
<td>630</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>595</td>
<td>0.</td>
<td>0</td>
<td>-2</td>
<td>630</td>
<td>0</td>
<td>10000</td>
<td>0</td>
<td>630</td>
<td>0</td>
<td>1000</td>
<td>-2</td>
<td>630</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>7CC</td>
<td>672</td>
<td>730</td>
<td>-1</td>
<td>630</td>
<td>0</td>
<td>10000</td>
<td>0</td>
<td>630</td>
<td>0</td>
<td>1000</td>
<td>-2</td>
<td>630</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>700</td>
<td>672</td>
<td>730</td>
<td>0</td>
<td>630</td>
<td>0</td>
<td>10000</td>
<td>0</td>
<td>630</td>
<td>0</td>
<td>1000</td>
<td>-2</td>
<td>630</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>700</td>
<td>-100</td>
<td>100</td>
<td>-1</td>
<td>630</td>
<td>0</td>
<td>10000</td>
<td>0</td>
<td>630</td>
<td>0</td>
<td>1000</td>
<td>-2</td>
<td>630</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>700</td>
<td>672</td>
<td>730</td>
<td>0</td>
<td>630</td>
<td>0</td>
<td>10000</td>
<td>0</td>
<td>630</td>
<td>0</td>
<td>1000</td>
<td>-2</td>
<td>630</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>700</td>
<td>672</td>
<td>730</td>
<td>0</td>
<td>630</td>
<td>0</td>
<td>10000</td>
<td>0</td>
<td>630</td>
<td>0</td>
<td>1000</td>
<td>-2</td>
<td>630</td>
<td>0</td>
</tr>
</tbody>
</table>

FIGURE 45. (continued)
<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0.</td>
<td>1000.</td>
<td>7CC.</td>
<td>C.</td>
<td>1CC0.</td>
</tr>
<tr>
<td>0.011</td>
<td>0.</td>
<td>1.</td>
<td>-2.</td>
<td>63C.</td>
</tr>
<tr>
<td>0.</td>
<td>0.</td>
<td>C.</td>
<td>-100.</td>
<td>10C.</td>
</tr>
<tr>
<td>0.</td>
<td>0.</td>
<td>0.</td>
<td>C.</td>
<td>0.</td>
</tr>
<tr>
<td>0.</td>
<td>6.</td>
<td>31</td>
<td>0.</td>
<td>0.</td>
</tr>
<tr>
<td>0.</td>
<td>0.</td>
<td>0.</td>
<td>C.</td>
<td>0.</td>
</tr>
<tr>
<td>0.</td>
<td>0.</td>
<td>C.</td>
<td>0.</td>
<td>0.</td>
</tr>
<tr>
<td>0.</td>
<td>0.</td>
<td>0.</td>
<td>C.</td>
<td>0.</td>
</tr>
<tr>
<td>7800.</td>
<td>0.</td>
<td>1.</td>
<td>C.</td>
<td>0.</td>
</tr>
</tbody>
</table>

**IPAR**

**FCR CRYER (AC. CF LUMPS)**

**FOR FBC**

**FOR SERVC**

**FCR LAG (N NANC M)**

FIGURE 45. (continued)
**KPM**

<table>
<thead>
<tr>
<th>Tank</th>
<th>SCLICS Dryer</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>FBC</td>
</tr>
<tr>
<td>7</td>
<td>MCR</td>
</tr>
<tr>
<td>10</td>
<td>LNCR</td>
</tr>
<tr>
<td>5</td>
<td>SERVO TO EFFECT STEAM TEMPERATURE</td>
</tr>
<tr>
<td>6</td>
<td>LAG Transport Delay</td>
</tr>
<tr>
<td>8</td>
<td>SRCRE ACCE</td>
</tr>
</tbody>
</table>

**FPRAR**

<table>
<thead>
<tr>
<th>SCLICS Dryer (8 LUMP)</th>
</tr>
</thead>
<tbody>
<tr>
<td>8. 0719 206 134 4</td>
</tr>
<tr>
<td>1000 97.5 0 1000 .205 0 1000 560</td>
</tr>
<tr>
<td>0 1000 710 C 1000 97.5 0 1000</td>
</tr>
<tr>
<td>.205 0 10 560 0 1000 710 0</td>
</tr>
<tr>
<td>1000 57.5 0 1000 .205 0 1000 560</td>
</tr>
<tr>
<td>0 1000 710 C 1000 97.5 0 1000</td>
</tr>
<tr>
<td>.205 0 10 560 0 1000 710 0</td>
</tr>
<tr>
<td>1000 57.5 0 1000 .205 0 1000 560</td>
</tr>
<tr>
<td>0 1000 710 C 1000 97.5 0 1000</td>
</tr>
<tr>
<td>.205 0 10 560 0 1000 710 0</td>
</tr>
<tr>
<td>1000 97.5 0 1000 .205 0 1000 560</td>
</tr>
</tbody>
</table>

FIGURE 4.5. INPUT DATA TO ANALYSIS PACKAGE, EXAMPLE NO. 2
node as is indicated by the value in node one's subsector of IPAR. Consequently the dryer node's subsector of FPAR has 152 locations. The first six contain the values of selected parameters. Note that the lump residence time is .0125 hrs, so the total dryer residence time is six minutes. Note also that the lump length is four feet, so the dryer is 32 feet in length. The next 90 locations contain the value, lower and upper bound for the 30 internal LDVs (four per lump for the first seven lumps and two for the eighth) while the last 56 locations contain the value, lower and upper bound, and code for the 14 internal LAVs (two per lump for lumps two through eight).

The subsectors of IPAR associated with the feedback controller and the servo controller indicate that the values to be extracted from the subsectors of the STRMV sector associated with the input and output streams should all come from the first position. The subsectors of FPAR for these two nodes show that there is to be no derivative or integral action and that the feedback controller overall gain factor $K_c$ is to be -10.0 while that of the servo is to be 10.0. Note that the codes for the internal class A LEAVs are set equal to $-(n+1)$ where $n$ is the number of the node with which they are associated and that the last four positions of the subsectors, serving as the error push down list, are set to zero.
The IPAR subsector for the lag node indicates that one variable is to be delayed one time step and the FPAR subsector shows thirty positions in the push down list even though only one of them will be used on this run. This done so that if, on subsequent computation runs, it is desired to increase the delay, only one value in the cards punched by the analysis package need be changed thus allowing the modeler to continue his simulation study without making another run with the analysis package.

Finally, the IPAR subsector for the source node shows that the forcing value is to be placed in position nine of the subsector of the STRMV sector associated with the output stream which means that the dryer inlet solids flow rate will be varied according to the source node algorithm. The subsector of the STRMV sector associated with stream one, which is the dryer inlet solids stream, shows that the first two variables are coded as constants while the third is coded as a LEAV associated with node number seven, the source node. The single variable associated with stream five is coded minus one since it is a LDV associated with node five. On the other hand, the single variables associated with streams six and eight are coded as zero since they occur in implicit expressions in nodes three and four respectively and will be matched with GAVs
during the ensuing analysis.

4. Results of the Analysis

The output consists of about 750 printed lines and 131 punched cards. On the pass supervised by SRN7, a total of 38 LDVs are identified: 32 with node one, the dryer, three with node two, the feedback controller, and three with node five, the servo controller. On the pass supervised by SETUPS, only two GAVs and GAEs are identified, these coming from nodes three and four, the feedforward computation node and the adder node. These two GAEs obviously can be and are decoupled; hence the algebraic system is defined in terms of the 16 LEAVs from node one, the four LEAVs from nodes two, five, six and seven, and the two singly decoupled GAEs from nodes four and three.

Figure 46 shows the images of the punched cards that will make up part of the input to the compute package. If for example it is desired to change the delay associated with node six, the following operations would be carried out. First, reference to the POINT common area shows that IPIP, the third element, is 15. IPIP can be used to locate the pointer (stored in position 21 of IVEC and having a value of 156) to node six's subsector of IPAR. Since M, the number of time steps, is the second element of the subsector, the contents of position 158 of IVEC should be
FIGURE 14. PUNCHED CARD OUTPUT FROM ANALYSIS PACKAGE, EXAMPLE NO. 2
FIGURE 46. (continued)
FIGURE 16. (continued)
changed. From the listing it is seen that this quantity is the 18th number on the tenth punched card. Now, say that it is desired to change the overall gain in the feedback controller (node two). Reference to POINT shows that IPPAR is eight. Reference to position ten in IVEC shows that the pointer for node two's subsector of FPAR is 153. Hence, since $K_c$ resides in the first position of the subsector, the 154th location of FPVEC should be changed. This can be accomplished by changing the fourth number on the 47th card.

C. A Double Effect Evaporator System

This example illustrates the simulation of the hydraulic phenomena as well as the mass and energy dynamics associated with evaporators. As mentioned earlier in this thesis it is the hydraulic phenomena that pose the greatest challenge since an effective simulation demands that the nonlinear momentum balances be formulated implicitly. Hence, unlike the previous examples, OSUSIM is the only MSP that can handle modules of this nature since it is the only MSP that admits node models described in terms of implicit algebraic equations. This example also illustrates how one NSR can be used for more than one node.

1. The Digraph

Figure 47 shows that the digraph consists of six nodes.
FIGURE 4.7. DIGRAPH FOR DOUBLE EFFECT EVAPORATOR SYSTEM
connected by 13 streams. Node one represents evaporator no. 1 where weak liquor enters via stream one and steam enters the steam chest via stream two. Stream one is defined by its velocity, temperature, pressure and concentration, while stream two is defined by its mass flow rate, temperature, and pressure. The evaporated component, water in this case, leaves via stream four and passes on to the steam chest of evaporator no. 2. The concentrated liquid outlet from evaporator no. 1 passes via stream three to node two which represents a pump-valve combination. In this case only the valve is active since the pressure in the evaporator chamber of evaporator no. 2 is lower than that in evaporator no. 1.

The liquid level in evaporator no. 1 is transmitted via stream five to a controller represented by node three. Here the level is compared to a set point transmitted to the controller via stream eight and a PID control response is transmitted to node two, the control valve, via stream six. Streams five, eight, and six are defined in terms of only one variable. The remaining part of the digraph is identical in structure to the part just discussed. The pressure of the vapors leaving evaporator no. 2 via stream 11 is considered fixed in value as a result of perhaps a steam ejector. Likewise, the pressure of the liquid leaving node five via stream nine is considered
fixed at one atmosphere.

2. Node Modules

a. Evaporator

Figure 48 shows a schematic of an evaporator module where with the exception of L, the liquid flow rate, upper case letters refer to vapor stream quantities and lower case letters to the liquid stream. Note that the circled numbers give the local stream number and that Table 4 gives the definition of the variables.

The model building begins with a mass balance over the liquid phase:

\[ L_o - V' - L_1 = \rho_L A \frac{dz}{du} \]  \hfill (32)

and a component mass balance over the liquid phase:

\[ L_o c_o - L_1 c_1 = \rho_L A \frac{dz}{du} (z c_1) \]  \hfill (33)

An enthalpy balance over the liquid phase results in

\[ L_o h_o + Q - V'H_1 - L_1 h_1 = \rho_L A \frac{dz}{du} (z h_1) \]  \hfill (34)

Finally, a mass balance over the vapor phase gives

\[ V' - V_1 = \frac{d}{du} (\rho_v A (Z - z)) \]  \hfill (35)

There are several manipulations that must be made in order to get Eqs. 33, 34, and 35 into more tractable form. First, Eqs. 32 and 33 are combined to give

\[ L_o (c_o c_1) + c_1 V' = \rho_L A \frac{d c_1}{du} \]  \hfill (36)

which replaces Eq. 33. If the liquid phase enthalpy, which
FIGURE 48. EVAPORATOR MODULE
is given by

\[ h(t) = C_P \cdot (t - t_R) + h(t_R) \]

where \( t_R \) is a reference temperature, is combined with Eqs. 32 and 34 the result is

\[ L_0 C_p (t_0 - t_1) + Q - \upsilon' \lambda = \rho_L A z C_p \frac{dt}{du} \]  \hspace{1cm} (37)

which replaces Eq. 34.

<table>
<thead>
<tr>
<th>TABLE 4. VARIABLE DEFINITION FOR EVAPORATOR</th>
</tr>
</thead>
<tbody>
<tr>
<td>( V )</td>
</tr>
<tr>
<td>( T )</td>
</tr>
<tr>
<td>( P )</td>
</tr>
<tr>
<td>( v )</td>
</tr>
<tr>
<td>( L )</td>
</tr>
<tr>
<td>( t )</td>
</tr>
<tr>
<td>( p )</td>
</tr>
<tr>
<td>( c )</td>
</tr>
<tr>
<td>( z )</td>
</tr>
<tr>
<td>( \upsilon' )</td>
</tr>
<tr>
<td>( \rho_L )</td>
</tr>
<tr>
<td>( \rho_V )</td>
</tr>
<tr>
<td>( Z )</td>
</tr>
<tr>
<td>( A )</td>
</tr>
<tr>
<td>( M )</td>
</tr>
<tr>
<td>( R )</td>
</tr>
</tbody>
</table>
Finally, if Eq. 35 is combined with the ideal gas law
\[ \rho_v = \frac{PM}{RT} \]
and Eqs. 32 and 37, the following equation is obtained
\[
\frac{R}{M}(v' - v_1) + \frac{P_1}{t_1 \rho_L}(L_o - v' - L_1)
\]
\[
= \frac{(Z-z)}{t_1 L_z C_p} [L_0 C_p (t_o - t_1) + Q - v' \lambda] = \frac{\Lambda (Z-z)}{t_1} \frac{dP_1}{dU}
\]
(38)
which replaces Eq. 35.

Now several algebraic relations must be developed. First, a temperature-pressure relationship for saturated water vapor is obtained in the following form
\[
T_1 = E(P_1) = \frac{10^3}{a_1 + a_2 \log_{10} P_1}, \quad a_1 = 1.7, \quad a_2 = 0.247
\]
(39)
based on data from the Combustion Engineering Steam Tables (41). Next, the Dühring relation (25) is put in the following algebraic form:
\[
t_1 = D(c_1, T_e) = a_3 + a_4 c_1^2 + (a_5 + a_6 c_1) \cdot T_e
\]
(40)
\{a_3 = 1.5502826, a_4 = 223.35509, a_5 = 0.97596295, a_6 = 0.25473122\}
where \( t_1 \) is the temperature of the outlet liquid stream which is considered to be at its boiling point and \( T \) is the temperature of saturated steam at the pressure of the liquid. The values of the coefficients were determined by using the routine REGRES which is part of the OSU Chemical
Engineering Department Computer Program Library. Third, a mechanical energy balance between the inlet and the surface of the liquid inside the evaporator yields

\[ \frac{D_0 - P_1}{144} + \rho_L z = 0 \]  (41)

where the kinetic energy term at the inlet \((v_0)^2(2g_c)\) is nullified by the entrance loss term \(K_{ent}(v_0)^2/(2g_c)\) since \(K_{ent}\) is unity (6a). Fourth, a mechanical energy balance between the surface of the liquid and the outlet of the evaporator yields

\[ (P_1 - P_L)\frac{144}{P_L} + z - \frac{6fL_ev_1^2}{D_0g_c} = 0 \]  (42)

where \(D_0\) is the diameter of the outlet pipe and \(L_e\) is the equivalent length. The factors 6.0 and 144.0 result from a conversion to a common set of units. The Weissbach friction factor \(f\) which is four times the fanning friction factor is calculated from the following algorithm based on data in the text by Foust (6b):

\[
\begin{align*}
    f &= \frac{64}{Re} & \text{Re} < 800 \\
    f &= \exp\{.20501 - .448462\log_{10} Re\} & 800 \leq \text{Re} < 3000 \\
    f &= .0056 + .5/Re^{.32} & \text{Re} \geq 3000
\end{align*}
\]

where \(Re\) is the Reynolds Number and where the expression for \(f\) values in the \(800 \leq \text{Re} < 3000\) interval is an artificial straight line on the log-log Weissbach plot connecting
the laminar and turbulent regions. Without this artificial correlation the (f,Re) relation is discontinuous and convergence problems result during the iterative solution of the algebraic equations.

The outlet liquid temperature $t_1$ plays two different roles in the two independent and physically valid equations as can be seen when Eqs. 36, 37, 38, 40, and 39 are written in the following format:

\[ c_1 = A (L_0, c_0, \ldots) \]  
\[ t_1 = B (L_0, t_0, \ldots) \]  
\[ P_1 = C (V'_1, V_1, \ldots) \]  
\[ t_1 = D (c_1, T_e) \]  
\[ T_e = E(P_1) \]

To alleviate this problem, Eqs. d and e are combined to give

\[ t_1 = D(c_1, E(P_1)) \]  
and then Eq. f is differentiated:

\[ t_1 = \frac{\partial D}{\partial c_1} c_1 + \frac{\partial D}{\partial T_e} T_e + \frac{\partial E}{\partial P_1} P_1 = \alpha c_1 + \beta P_1 \]

where

\[ \alpha = 2a_4 c_1 + a_6 T_e \]

\[ \beta = (a_5 + a_6 c_1) \frac{a_2 \times 10^3}{(a_1 + a_2 \log_{10} P_1)^2 P_1} \]

Next, Eqs. a, b, c, and g are combined giving
\[ B - \alpha A - \beta C = 0 \]  
which replaces Eq. 37.

The energy transferred from condensing vapors to boiling liquid can be written as

\[ Q = U_A C (T_O - t_1) = V_O \lambda \]  
but, since \( T_O \) is considered to be saturated steam, Eq. 44 can be written as

\[ V_O = \frac{U_A C (E(P_O) - t_1)}{\lambda} \]  

Hence if the outlet vapor pressure is not fixed, as will be the case in the first effect, the evaporator model is described in terms of three LDEs, Eq. 32, 36, and 38, three LIAEs, Eqs. 41, 42, and 43, and two LEAEs, Eqs. 40 and 45. Table 5 shows which variables are defined as LDVs and LAVs. Note that the one asterisk indicates that the LAV is a LEAV. Two asterisks indicate that the saturated steam temperature is not a LEAV in the usual sense but is a parasitic variable in that there is a one-to-one relation, by virtue of Eq. 39, between it and the vapor pressure. This relation is similar to the relation that existed between \( M_k \) and \( M_k \) in the solids dryer model discussed above. If the outlet vapor pressure is fixed, as will be the case in the second effect, then since the derivative will be zero, Eq. 38 is treated as a LIAE instead of a LDE.
TABLE 5. IDENTIFICATION OF LDVs AND LAVS

<table>
<thead>
<tr>
<th>Name</th>
<th>Association</th>
</tr>
</thead>
</table>
| $v_o$ | Stream 1    | $y_1$  
| $t_o$ | "           | $y_2$  
| $p_o$ | "           | $y_3$  
| $c_o$ | "           | $y_4$  
| $v_o$ | Stream 2    | $y_5^*$  
| $T_o$ | "           | $y_6^{**}$  
| $p_o$ | "           | $y_7$  
| $v_1$ | Stream 3    | $y_8$  
| $t_1$ | "           | $y_9^*$  
| $p_1$ | "           | $y_{10}$  
| $c_1$ | "           | $x_1$  
| $v_1$ | Stream 4    | $y_{11}$  
| $T_1$ | "           | $y_{12}^{**}$  
| $p_1$ | "           | $y_{14}$ or $x_3$  
| $z$ | Stream 5    | $x_2$  
| $v'$ | Node        | $y_{13}$  


Referring to the listing of the PRES NSR which describes the evaporator model it is seen that a code, ICP, stored in the node's subsector of the IPAR sector, indicates whether or not the outlet vapor pressure is to be held constant. On the pass supervised by SRN7 (ICALL equals one), not only do the values of NLAV, NLIAE, and NLDE but also the numbers of calls to COMPX depend on the value of ICP. Similarly on the call supervised by SETUPS (ICALL equals two), an extra call to MKLST for \( y_{14} \) is made if ICP indicates that \( P_1 \) is to be held constant.

On the computation pass (ICALL equals three), \( P_1 \), represented by the FORTRAN variable VAR, is assigned the value of \( x_3 \) if \( P_1 \) is allowed to vary or the value of \( y_{14} \) if it is to be held constant. Note that the FORTRAN function subroutine VAPR is used to calculate the temperature of the inlet and outlet vapor streams. If INST is one then, depending on ICP, two or three \( f_i \) values are computed and if INST is three then, again depending on ICP, three or four \( g_i \) values are calculated. By referring to Eqs. 38 and c, it is seen that if \( P_1 \) is to be held constant, the quantity \( C \) in Eq. 43 must be zero; however \( C \) is a component of Eq. 43 which also must equal zero. Hence in the PRES listing, Eq. 43 is associated with \( g_3 \), and the FORTRAN variable S3, which occurs in the expression for \( g_3 \),
is assigned to $g_4$ if ICP indicates that $P_1$ is to be held constant.

Finally if INST is two or four, the two LEAVs are calculated. Note that it is not obvious whether they should be considered class A or B LEAVs so they are treated as both, which simply means that, at worst, they are recomputed more often than necessary. Note that subroutine FRIC, and function subroutine DUHR and VAPR supply, respectively, the Weissbach friction factor, the outlet liquid temperature (using the Dühring correlation), and the saturated steam temperature for a given pressure.

The node's subsector of FPAR consists of ten elements, the first six of which contain values of parameters occurring in the equations and the last four contain the value, lower and upper bound, and code for the internal LAV $y_{13}$.

b. Liquid Level Controller

Nodes three and six are liquid level controllers and are described by the HETR NSR which was discussed in detail in the previous section.

c. Pump-Valve

Figure 49 shows a schematic of the pump-valve node that is described by the PUVA NSR. Local stream one transmits
the velocity, temperature, pressure and concentration of a liquid stream to the node and local stream three serves as the exit stream for the liquid. Local stream two transmits a control pressure which actuates the valve. The model building proceeds as follows. A mechanical energy balance between points two and three gives

$$\frac{p_2}{\rho} = \frac{p_3}{\rho} + E_L \quad \text{or} \quad \Delta p = p_2 - p_3 = \rho E_L \quad (45)$$

where $E_L$ is the energy loss due to friction at the valve. The Taylor Handbook (42) gives the following definition for the control valve constant:

$$C_v = \frac{Q}{\sqrt{G \Delta p}} \quad (46)$$

where $Q$ is the water flow rate in U.S. gallons/minute, $G$ is the specific gravity of water, assumed to be unity henceforth, and $\Delta p$ is the pressure drop in psi. If Eq. 46 is solved for $\Delta p$ the result is

$$\Delta p = \frac{Q^2}{C_v^2} = p_2 - p_3 \quad (47)$$

therefore $\rho E_L = \frac{Q^2}{(C_v)^2}$. The flow rate can be related to the velocity (ft/sec) as follows

$$Q = \gamma A v$$

where $A$ is the pipe cross sectional area (ft$^2$) and $\gamma$ is a proportionality constant having a value of 435.73 USgal*sec/(cu ft*min); hence Eq. 45 becomes
\[ p_2 = p_3 + \frac{\gamma^2 A^2 v^2}{C_v^2} \]  

(48)

The control valve characteristic, which gives \( C_v \) as a function of \( x \), the percentage opening of the valve, can usually be represented by a quadratic:

\[ C_v = C_{v1}x + C_{v2}x^2 \]  

(49)

Now, a mechanical energy balance between points one and two gives:

\[ 144 \frac{p_1}{\rho} + h(v) = 144 \frac{p_2}{\rho} \]  

(50)

where \( h(v) \), the head (ft) supplied by the pump, is allowed to be a linear function of velocity:

\[ h(v) = h_m(1. - \frac{v}{v_m}) \]  

(51)

Combining Eqs. 48 and 50 results in:

\[ C_v^2[p_1 - p_3 + \frac{\rho h(v)}{144}] - \gamma^2 A^2 v^2 = 0 \]  

(52)

If \( p_c \) is the control pressure transmitted to the node by stream two then the pressure that is used to calculate the per cent opening is obtained after a first order lag:

\[ \frac{dp_v}{dU} = \frac{p_c - P_v}{\tau_L} \]  

(53)

If, for example, the valve is of the air-to-open type, the percentage opening is directly dependent on the lagged control valve pressure \( p_v \) as follows:

\[ x = 100. \ast \frac{(p_v - p_{max})/(p_{max} - p_{min})}{\frac{p_c - P_v}{\tau_L}} \]
where \( p_{\text{min}} \) and \( p_{\text{max}} \) are the minimum and maximum control pressures (usually 3.0 and 15.0 psi).

Table 6 shows which variables are identified as LDVs and which as LAVs. In view of these identifications the node describing equations can be rewritten as follows:

\[
g_1 = C_v^2 [y_3 - y_7 + \frac{\rho h_m}{144} (1 - \frac{y_1}{v_m}) - (\gamma A y_1)^2]
\]

\[
g_2 = y_1 - y_5
\]

\[
y_6 = w_1 = y_2
\]

\[
y_8 = w_2 = y_4
\]

\[
\frac{dx_1}{dU} = f_1 = \frac{y_9 - x_1}{\tau_L}
\]

Note that since the concentration and temperature of the liquid are assumed to be unchanged by the action of the pump or valve, the outlet stream temperature and concentration are treated as LEAVs. Even though the stream velocity is constant as it passes through the node, the outlet stream velocity is not treated as a LEAV. To show that this implicit formulation is necessary, consider the following three node example:

where node A represents a storage tank providing water at a constant head and node B is the pump-valve node while
node C is a splitter or tee. If only the hydraulics are considered, then node A will be described by a mechanical energy balance (MEB)

\[ g_A(p_1, v_1) = 0 \]  \hspace{1cm} (a)

node B by a MEB

\[ g_B(p_1, v_1, p_2) = 0 \]  \hspace{1cm} (b)
and a mass balance
\[ g_{B_2}(v_1, v_2) = v_2 - v_1 = 0 \]  \hspace{1cm} (c)

node C by two MEBs
\[ g_{C_1}(p_2, v_2, p_3, v_3) = 0 \]  \hspace{1cm} (d)
\[ g_{C_1}(p_2, v_2, p_4, v_4) = 0 \]  \hspace{1cm} (e)

and a mass balance
\[ g_{C_3}(v_2, v_3, v_4) = 0 \]  \hspace{1cm} (f)

If streams three and four exhaust to the atmosphere, \( p_3 \) and \( p_4 \) will be fixed hence there are six coupled equations in six unknowns that must be solved simultaneously.

Now if Eq. (c) were reformulated in explicit form as follows
\[ v_2 = w_B(v_1) = v_1 \]  \hspace{1cm} (c')

and treated as a class B LEAV then \( v_2 \) would not be calculated until after the other five equations were solved for \( p_1, v_1, p_2, v_3, \) and \( v_4 \). Therefore the values of these last five variables will depend on the value of \( v_2 \) obtained from Eq. c' and since there is no iteration between the five simultaneous equations and the single explicit equation the results of the calculations will not be strictly correct. In some cases this sacrifice of rigor might be tolerable since the dimension of the coupled
implicit algebraic system would be decreased allowing for a savings in memory space and computation time. Furthermore in those cases where the outlet streams of the node in question either exhaust to the atmosphere or lead to other nodes not described by implicit algebraic equations there is no loss of rigor if equations like Eq. c are reformulated explicitly.

Referring to the listing of PUVA it is seen that the value of the flag IC (stored in the node's subsector of the IPAR sector) determines whether the value is an air-to-open or an air-to-close type and consequently how the percentage opening is calculated. The first eight positions in the node's subsector of FPAR contain the coefficients (CV1 and CV2) that appear in Eq. 49 which gives the valve characteristic, the maximum and minimum control pressure (XMAX and XMIN), the maximum head (HMAX), the maximum velocity (VMAX), the cross sectional area of the pipe (AREA) and the time constant (TAUL) in Eq. 53. The next three positions contain the value, lower and upper bound of $x_1$. If no pumping action is desired, the value of HMAX is set to zero and the node acts strictly as a valve.

3. Arrangement of the Input Data

Figure 50 shows a listing of the input data. The FPAR subsectors associated with the two pump-valve nodes are
**KPM**

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>14</td>
<td>PRES EVAP NC. 1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>FUVA PLM-VALVE NO. 1</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>6</td>
<td>HETR CONTROLLER NC. 1</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>PRES EVAP NC. 2</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>FUVA PLM-VALVE NO. 2</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>14</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>3</td>
<td>HETR CONTROLLER NC. 2</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>12</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**PPAR**

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10</td>
<td>EVAP NO. 1</td>
<td></td>
</tr>
<tr>
<td>1C</td>
<td>5</td>
<td>.111 600.</td>
<td>1. 5C. .5  C.</td>
</tr>
<tr>
<td>5C</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>11</td>
<td></td>
<td></td>
</tr>
<tr>
<td>.C65</td>
<td>15. 3. 0. 100. .004 10.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>3. 15.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>19</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2C</td>
<td>10. C. C. 1. 10. 3. 15.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>3. 15. 1C. 2. 15. -4. 0.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0. 0.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>5. .111 6CC. 1. 5C. .5 0.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5C</td>
<td>0.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>11</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**FIGURE 50. INPUT DATA TO ANALYSIS PACKAGE, EXAMPLE NO. 3**
<table>
<thead>
<tr>
<th></th>
<th>.065</th>
<th>0.</th>
<th>15.</th>
<th>3.</th>
<th>25.</th>
<th>100.</th>
<th>.004</th>
<th>10.</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>6</td>
<td>15</td>
<td>3</td>
<td>15</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>10</td>
<td>C</td>
<td>C</td>
<td>1</td>
<td>10</td>
<td>3</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>3</td>
<td>15</td>
<td>10</td>
<td>15</td>
<td>15</td>
<td>-7</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**IPAR**

1 1
2 3 1
4 1
5 3
6 1

**EVAP NO. 1 (PRESS CODE)**

- PLVA (AIR TC CLCSE CCCE)
- CONTR. AC. 1
- EVAP NO. 2
- FLVA AC. 2
- CONTR NO. 2

**INLET LIQ (VTPC)**

<table>
<thead>
<tr>
<th></th>
<th>8.19</th>
<th>0.</th>
<th>0.</th>
<th>-1.</th>
<th>560</th>
<th>0.</th>
<th>0.</th>
<th>-1.</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.24</td>
<td>0.</td>
<td>100.</td>
<td>0.</td>
<td>-2</td>
<td>C</td>
<td>0.</td>
<td>-1.</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>12</td>
<td>5</td>
<td>-2.</td>
<td>685</td>
<td>0.</td>
<td>999.</td>
<td>-1.</td>
<td></td>
</tr>
<tr>
<td>19.7</td>
<td>0.</td>
<td>0.</td>
<td>-1.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**CUTLET LIQ (EVAP NO. 1)**

<table>
<thead>
<tr>
<th></th>
<th>6.82</th>
<th>0.</th>
<th>5C.</th>
<th>C.</th>
<th>570</th>
<th>0.</th>
<th>999.</th>
<th>-2.</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.87</td>
<td>0.</td>
<td>50.</td>
<td>0.</td>
<td>-2</td>
<td>C</td>
<td>0.</td>
<td>-1.</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>12</td>
<td>1C.</td>
<td>-5.</td>
<td>585</td>
<td>C</td>
<td>999.</td>
<td>-1.</td>
<td></td>
</tr>
</tbody>
</table>

**VAPCR FRCM EVAP NO. 1**

<table>
<thead>
<tr>
<th></th>
<th>.366</th>
<th>0.</th>
<th>1C.</th>
<th>-5.</th>
<th>585</th>
<th>C</th>
<th>999.</th>
<th>-1.</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.14</td>
<td>0.</td>
<td>99.</td>
<td>-1.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
LEVEL SIGNAL FROM EVAP NO. 1

CCNTRL PRESS TO PUVA NC. 1

LIQ FROM PUVA NO. 1

LIQ EXITING TO ATMOSPHERE

LIQ FROM EVAP NO. 2

VAPCR TC VACUUM PUMP

LEVEL SIGNAL FROM EVAP NO. 2

SET PT. TC CONTR NO. 2

CCNTRL PRESS TO PUVA

FIGURE 50. (continued)
the same except that node two has zero head. The two valve characteristic coefficients, CV1 and CV2, were obtained by placing a straight line (hence CV2 is zero) through \( C_v \) vs. percentage opening data that appears in the Taylor Handbook (42). The IPAR subsectors associated with the two evaporator nodes are different because ICP for the first is unity indicating that the outlet vapor pressure is to be varied while ICP for the second is minus one since the outlet vapor pressure is to be held constant. The IPAR subsectors for the two pump-valve nodes both indicate that the valve is to act as an air-to-close type.

For stream one, which transmits the weak liquid to the first evaporator, the subsector of STRMV indicates that, while the velocity is fixed (since the code is minus one) in value at 8.19 ft/sec, the pressure is treated as a variable. For stream two, which transmits the steam to the evaporator, the flow rate is classified as a LEAV associated with node one, while the pressure and temperature are fixed in value. Actually since the steam is saturated, the temperature value given in the stream two subsector of STRMV is ignored and instead it is calculated in the evaporator NSR using Eq. 39. The pressure of stream four, which transmits vapor from node one to node four, is coded minus one even though it will vary because it is to
be treated as a LDV associated with node one. The flow rate of stream four is classified as a LEAV associated with node four.

The pressure of stream nine, which transmits the liquid from the second pump-valve node to the atmosphere, is held constant at 15.0 psi, while the concentration and temperature are classified as LEAVs associated with node five. Stream 14 shows that the control pressure that actuates the valve in node five has an initial value of 10.0 psi and lower and upper bounds of 3.0 and 15.0 psi. Finally, the pressure of stream 11 is held constant at 1.96 psi since this stream is assumed to be connected to a steam ejector.

4. Results of the Analysis

The output consists of about 1000 printed lines and 173 punched cards. On the pass supervised by SRN7 a total of 13 LDVs are identified: three with node one, evaporator no. 1, one with node two, pump-valve no. 1, three with node three, level controller no. 1, two with node four, evaporator no. 2, one with node five, pump-valve no. 2, and three with node six, level controller no. 2. The second evaporator has one less LDV than the first since the outlet vapor pressure is held constant.

On the pass supervised by SETUPS, 11 GAES and GAVs
are identified and the occurrence matrix (OM) has the form shown in figure 51a. The operations supervised by DECUP result in two GAEs being decoupled after which the OM has the form shown in figure 51b. The operations supervised by SBSYS result in no subsystem identification but the row and column order is permuted slightly resulting in an OM given in figure 51c.

Figure 52 shows the images of the cards punched by the analysis package. To determine which nodes the two decoupled GAEs are associated with, the following analysis of the data in figure 52 can be made. The LISTE sector contains the GAE indices so the value of IPLE, the pointer to the director sector for the LISTE sector, must be determined. By definition, IPLE is the sixth element of the POINT common area, therefore by referring to the sixth element of the first row in figure 52 it is seen that the value of IPLE is 45. This means that the first element of the director sector for LISTE is the 46th element of IVEC which is located in the sixth position of the 5th row in figure 52. The value of this element is 164 which means that node one's subsector of LISTE begins with the 165th element of IVEC which is located in the 5th position of the eleventh row in figure 52.

The value of this element is one indicating that the first LIAE of node one is matched with the first GAE which
FIGURE 51a.

FIGURE 51b.

FIGURE 51c.
<table>
<thead>
<tr>
<th>1</th>
<th>7</th>
<th>13</th>
<th>33</th>
<th>39</th>
<th>45</th>
<th>51</th>
<th>291</th>
<th>57</th>
<th>19</th>
<th>3</th>
<th>2</th>
<th>3</th>
<th>3</th>
<th>45</th>
<th>8</th>
<th>487</th>
<th>292</th>
<th>295</th>
<th>298</th>
<th>277</th>
</tr>
</thead>
<tbody>
<tr>
<td>397</td>
<td>301</td>
<td>315</td>
<td>319</td>
<td>221</td>
<td>290</td>
<td>89</td>
<td>83</td>
<td>95</td>
<td>101</td>
<td>375</td>
<td>379</td>
<td>71</td>
<td>288</td>
<td>64</td>
<td>9</td>
<td>11</td>
<td>10</td>
<td>5C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>288</td>
<td>107</td>
<td>114</td>
<td>115</td>
<td>124</td>
<td>131</td>
<td>136</td>
<td>1</td>
<td>11</td>
<td>22</td>
<td>41</td>
<td>51</td>
<td>62</td>
<td>141</td>
<td>142</td>
<td>143</td>
<td>146</td>
<td>147</td>
<td>148</td>
<td>81</td>
<td></td>
</tr>
<tr>
<td>97</td>
<td>109</td>
<td>125</td>
<td>137</td>
<td>141</td>
<td>145</td>
<td>161</td>
<td>165</td>
<td>161</td>
<td>157</td>
<td>2CS</td>
<td>213</td>
<td>217</td>
<td>222</td>
<td>23E</td>
<td>244</td>
<td>260</td>
<td>271</td>
<td>277</td>
<td>175</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>13</td>
<td>9</td>
<td>3</td>
<td>14</td>
<td>9</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>0</td>
<td>4</td>
<td>2</td>
<td>0</td>
<td>151</td>
<td>154</td>
<td>155</td>
<td>158</td>
<td>160</td>
<td></td>
<td></td>
</tr>
<tr>
<td>161</td>
<td>226</td>
<td>239</td>
<td>248</td>
<td>251</td>
<td>265</td>
<td>274</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>0</td>
<td>3</td>
<td>6</td>
<td>7</td>
<td>0</td>
<td>0</td>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>7</td>
<td>4</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>C</td>
<td>C</td>
<td>10</td>
<td>14</td>
<td>5</td>
<td>0</td>
<td>13</td>
<td>12</td>
<td>14</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>-1</td>
<td>1</td>
<td>1</td>
<td>-1</td>
<td>-1</td>
<td>1</td>
<td>1</td>
<td>122</td>
<td>128</td>
<td>134</td>
<td>20</td>
<td>28</td>
<td>31</td>
<td>142</td>
<td>154</td>
<td>21C</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6C</td>
<td>68</td>
<td>71</td>
<td>218</td>
<td>1</td>
<td>4</td>
<td>5</td>
<td>3</td>
<td>6</td>
<td>2</td>
<td>5</td>
<td>7</td>
<td>8</td>
<td>10</td>
<td>11</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>4</td>
<td>-2</td>
<td>3</td>
<td>-5</td>
<td>0</td>
<td>5</td>
<td>4</td>
<td>-2</td>
<td>3</td>
<td>C</td>
<td>6</td>
<td>-3</td>
<td>2</td>
<td>-3</td>
<td>0</td>
<td>0</td>
<td>C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>-3</td>
<td>2</td>
<td>-3</td>
<td>-5</td>
<td>0</td>
<td>7</td>
<td>-5</td>
<td>10</td>
<td>8</td>
<td>0</td>
<td>9</td>
<td>0</td>
<td>7</td>
<td>-5</td>
<td>10</td>
<td>0</td>
<td>1</td>
<td>11</td>
<td>-6</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>-6</td>
<td>0</td>
<td>C</td>
<td>-7</td>
<td>E2</td>
<td>E6</td>
<td>SC</td>
<td>54</td>
<td>58</td>
<td>102</td>
<td>106</td>
<td>110</td>
<td>114</td>
<td>118</td>
<td>126</td>
<td>130</td>
<td>8</td>
<td>110</td>
<td></td>
<td></td>
</tr>
<tr>
<td>114</td>
<td>118</td>
<td>122</td>
<td>146</td>
<td>150</td>
<td>154</td>
<td>158</td>
<td>142</td>
<td>162</td>
<td>138</td>
<td>34</td>
<td>146</td>
<td>150</td>
<td>154</td>
<td>158</td>
<td>126</td>
<td>130</td>
<td>134</td>
<td>182</td>
<td>186</td>
<td></td>
</tr>
<tr>
<td>15C</td>
<td>158</td>
<td>2C2</td>
<td>48</td>
<td>206</td>
<td>182</td>
<td>186</td>
<td>190</td>
<td>194</td>
<td>166</td>
<td>170</td>
<td>174</td>
<td>178</td>
<td>218</td>
<td>214</td>
<td>21C</td>
<td>74</td>
<td>90</td>
<td>154</td>
<td>118</td>
<td></td>
</tr>
<tr>
<td>110</td>
<td>8</td>
<td>146</td>
<td>182</td>
<td>158</td>
<td>48</td>
<td>150</td>
<td>166</td>
<td>1</td>
<td>4</td>
<td>6</td>
<td>253</td>
<td>357</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C.252CCC00E C3</td>
<td>C.1CCCCC00E 02</td>
<td>C.50000000E 01</td>
<td>0.11099941E 00</td>
<td>0.60000000E 03</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.10000000E 01</td>
<td>0.20000000E 02</td>
<td>C.3608438E CC</td>
<td>C.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C.40000000E 01</td>
<td>0.11999995E 00</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C.0</td>
<td>C.1CCCCC00E C3</td>
<td>C.3S55S55S55E-C2</td>
<td>0.15000000E C2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.30000000E 01</td>
<td>0.15000000E 02</td>
<td>C.2CCCC000E C2</td>
<td>C.10000000E 02</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C.25000000E 03</td>
<td>C.5CCCCC00E 02</td>
<td>0.24979439E 01</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.8564C554E 01</td>
<td>0.30000000E 02</td>
<td>C.15000000E 02</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.15000000E 02</td>
<td>-0.40000000E 01</td>
<td>0.20790100E-C2</td>
<td>C.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.2C61214E-02</td>
<td>C.1CCCCC00E 02</td>
<td>0.50000000E 01</td>
<td>0.11099941E 00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.10000000E 01</td>
<td>0.20000000E 02</td>
<td>C.461644E5E CC</td>
<td>C.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C.10000000E 02</td>
<td>0.11999995E 00</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C.35000000E 02</td>
<td>0.10000000E 02</td>
<td>C.3S55S55S55E-C2</td>
<td>0.15000000E C2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.30000000E 01</td>
<td>0.15000000E 02</td>
<td>C.2CCCC000E C2</td>
<td>C.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C.25000000E 03</td>
<td>C.5CCCCC00E 02</td>
<td>0.24877672E 01</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**FIGURE 52. PUNCHED OUTPUT FROM ANALYSIS PACKAGE, EXAMPLE NO. 3**
FIGURE 52. (continued)
has been decoupled. Reference to the listing of PRES, which describes nodes one and four, shows that the first LIAE is the mechanical energy balance between the inlet and the surface of the boiling liquid. Since the only free variable appearing in that equation is the inlet pressure the decoupling is logical.

The director sector for LISTE also shows that the pointer to node four's subsector of LISTE is 169. Thus the first element in this subsector is located in the tenth position of the fifth row in figure 52. The value of this element is two, indicating that the first LIAE of node four is matched with the second GAE which has also been decoupled. Since node four is also described by the PRES NSR it is clear now which nodes and which equations have been decoupled.

The basic routines that make up the analysis package (not including the NSRs) require 7560 four-byte words on the IBM System/360 model 75 digital computer. For the evaporator system problem described in this section, 800 locations were reserved for IVEC and 1100 for FPVEC, while the common area, which is problem independent, requires 116 words. Three NSRs (PUVA, HETR, PRES) plus three supporting routines (FRIC, DUHR, VAPR) require 1956 words. Normally, the user would be concerned only with the coding
of these last routines. A summary of the memory space required is given below:

<table>
<thead>
<tr>
<th>Routine Type</th>
<th>Words</th>
</tr>
</thead>
<tbody>
<tr>
<td>analysis package routines</td>
<td>7560</td>
</tr>
<tr>
<td>common area</td>
<td>116</td>
</tr>
<tr>
<td>IVEC &amp; FPVEC</td>
<td>7676</td>
</tr>
<tr>
<td>NSRs and supporting routines</td>
<td>1900</td>
</tr>
<tr>
<td>dummy NSRs</td>
<td>9576</td>
</tr>
<tr>
<td>System/360 utility routines</td>
<td>1956</td>
</tr>
<tr>
<td></td>
<td>11532</td>
</tr>
<tr>
<td></td>
<td>11826</td>
</tr>
<tr>
<td></td>
<td>19076</td>
</tr>
</tbody>
</table>

The dummy NSRs consist of a FORTRAN subroutine title card and an END statement. The link-edit step required almost 10 seconds, while the execution of the object module required almost two seconds.
The object of this chapter is to present the features of the compute package in a manner similar to that of chapter IV which dealt with the analysis package. The overall logic of the compute package is presented by discussing the subroutine that acts as the executive routine for the compute package. The routine that supervises the display of the simulation results is described second. The next two sections are devoted to the routines that supervise the numerical solution of the system of global implicit algebraic equations and the system of ordinary differential equations. The fifth section describes the process of evaluating the explicit algebraic equations. The sixth section shows how the values of the local variables are placed in the augmented Jacobian matrix prior to the implementation of the Gauss-Jordan method which is the subject of the seventh section. After the global algebraic system is solved, the results must be placed in the correct location in the FPVEC vector and this is the topic of the eighth section. In the final section the nature of the diagnostic output is discussed.
A. Overall Logic

As with the analysis package the main program consists of an eight line FORTRAN program which assigns values to IN, the card input unit number, IOUT, the print output unit number, and IPUN, the punch output unit number, before program control is transferred to subroutine MAINP which acts as the executive program for the compute package. Since the vectors IVEC and FPVEC are not commoned, their dimensions can be altered by making a change in the FORTRAN Dimension statement in the main program.

To describe the overall logic of the compute package, it will be sufficient to discuss the flow diagram for subroutine MAINP which is given in figure 53. The first operation is to read in and print the information contained on the punched input cards. These cards contain the problem title, the elements of (a) the POINT common area, (b) the IVEC vector, (c) the first two sectors of the FPVEC vector, and (d) Namelists IO and I01 whose elements and their functions are listed in table 7. The cards containing the contents of POINT, IVEC, and FPVEC, are taken directly from the output of the analysis package so the user need only add the elements of the two Namelists and the title card.

In preparation for the solution of the ODEs, a process supervised by subroutine SRN8PL, the SV sector (see figure 9) is zeroed via a call to ZEROX. In another
ENTRY

read/print problem title
read/print elements of POINT
read/print IVEC
read/print FPVEC

call ZEROX to zero SV sector

read/print NAMELIST 10

T = 0, ICT = 1, ICT1 = 0, ICT2 = 1

ICT3 = 1

read/print NAMELIST IOI

T: 0

TTEST = TNXT - DELT*.5

ITYPE: 0

IPC = -1
call ALCMP

ICT3: LIST5(8)

ISK = 1

T = T + DELT, ICT = ICT + 1, ICT1 = ICT1 + 1, ICT3 = ICT3 + 1

FIGURE 53. MAINP(for compute package)
FIGURE 53. (continued)
no more problems

STOP

read ICODE

ICODE: 0

another problem is to solved

13

read/print NAMELIST IO

8

read IAR

read IVEC(IAR)

IAR: 0

respond to interrupts

7

read IAR

read FPVEC(IAR)

IAR: 0

2

FIGURE 53. (continued)
### TABLE 7. NAMELISTS AND LIST5 ELEMENTS

<table>
<thead>
<tr>
<th>IO</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JS(6)</td>
<td>Subsector No. of quantity to be displayed.</td>
</tr>
<tr>
<td>IPOS(6)</td>
<td>Position in subsector</td>
</tr>
<tr>
<td>ISIN(6)</td>
<td>Type of sector</td>
</tr>
<tr>
<td>ISAMP</td>
<td>Frequency of display</td>
</tr>
<tr>
<td>DELT</td>
<td>Time step</td>
</tr>
<tr>
<td>ISK</td>
<td>Diagnostic output code</td>
</tr>
<tr>
<td>NT</td>
<td>No. of quantities to be displayed</td>
</tr>
<tr>
<td>LIST5(30)</td>
<td>Utility vector (integers)</td>
</tr>
<tr>
<td>DUT(15)</td>
<td>&quot;  &quot;  (reals)</td>
</tr>
<tr>
<td>XLB(6)</td>
<td>Lower bounds for plotted quantities</td>
</tr>
<tr>
<td>XUB(6)</td>
<td>Upper &quot;  &quot;</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>IO1</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ITYPE</td>
<td>Code for preliminary alg. pass</td>
</tr>
<tr>
<td>J</td>
<td>Not used</td>
</tr>
<tr>
<td>TNXT</td>
<td>Time of next interrupt</td>
</tr>
<tr>
<td>ISTOP</td>
<td>Code for type of interrupt</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>LIST5</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Frequency of inversion</td>
</tr>
<tr>
<td>2</td>
<td>Punch out code for final values of FPAR &amp; STRMV</td>
</tr>
<tr>
<td>3</td>
<td>No. of iterations allowed before re-inversion</td>
</tr>
<tr>
<td>4</td>
<td>Iteration count</td>
</tr>
<tr>
<td>5</td>
<td>Post correction inversion code</td>
</tr>
<tr>
<td>6</td>
<td>Frequency of normalization</td>
</tr>
<tr>
<td>7</td>
<td>Invert on every iteration if ICT3&lt;List5(7)</td>
</tr>
<tr>
<td>8</td>
<td>Diagnostic print out override</td>
</tr>
<tr>
<td>9</td>
<td>Display code (Ø for plot, 1 for numerical)</td>
</tr>
<tr>
<td>10</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td></td>
</tr>
</tbody>
</table>
preliminary move, the quantities \( T, ICT, ICT_1, ICT_2, \) and \( ICT_3 \), are initialized where \( T \) is the time, \( ICT \) is the overall time counter, i.e., \( T = (ICT-1) \cdot DELT \), \( ICT_1 \) counts the number of time steps between displays, \( ICT_2 \) counts the number of displays, and \( ICT_3 \) is the time counter since the last interruption. Program control is then passed to box 24.

As will be pointed out, an interruption allows changes to be made at time \( TNXT \) in any quantity affecting the simulation. The algorithm which responds to an interruption is described in the logic starting with box 13. There, values can be read into selected positions of IVEC and/or FPVEC.

After the interruption is serviced, control is returned to box 24 where \( ICT_3 \) is initialized and values of Namelist IO1 are read in. If \( T \) is zero, then a call is made to SAVE which supervises the display of the values of selected quantities. In this case it would be a display of the initial values. The quantity \( TTEST \) is calculated for use later on when the program tests to see if it is time for an interruption. If \( ITYPE \) is zero then a preliminary solution of the implicit algebraic equations and a calculation of the class B LEAVs is carried out by a call to the routine ALGCMP. Note that the flag IPC is set
to minus one before this call (see table 8 for a definition of this flag). This option can be exercised when the user is not sure of the initial values of some of the quantities.

TABLE 8. IPC VALUES AND MEANING

<table>
<thead>
<tr>
<th>VALUE</th>
<th>MEANING</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
<td>Preliminary algebraic pass or Eulerian prediction pass</td>
</tr>
<tr>
<td>0</td>
<td>Regular two step prediction pass</td>
</tr>
<tr>
<td>1</td>
<td>Regular two step correction pass</td>
</tr>
</tbody>
</table>

For diagnostic purposes it is possible to have the values of the first two sectors of FPVEC printed out at every time step. When this is desired the flag ISK is set to zero. Box 201 shows that, if ICT3 is greater than LIST5(8), ISK will be set so that no diagnostic output can occur. Box 201 also serves as the beginning of the calculation loop in which the following steps are carried out. First, T and the three counters are incremented. Second, the flag ISK is checked to see if diagnostic output is desired, in which case IGO is set to one and control is transferred to box 15 where sectors FPAR and STRMV of FPVEC are printed out. IGO is used to direct control back to box 21 where ICT3 is compared with the
quantity two to see whether or not the upcoming prediction pass is the first one.

A call is made to subroutine SRN8P1 which supervises the prediction pass calculation which will yield values of $x^p(T), \ldots, x^p_{NLDE}(T)$ and the class A LEAVs for every node. A call to ALGCMP follows which results in the calculation of the values of $y^p(T), \ldots, y^p_{ICTV}(T)$, and of the class B LEAVs associated with each node. The superscript $p$ indicates that the variables have been calculated during the prediction phase. Now IPC is compared with one and, since no match is made, IPC is set to one and a call to SRN8P1 yields values of $x^c(T), \ldots, x^c_{NLDE}(T)$ and the class A LEAVs, for every node. After this, a call to ALGCMP yields the values of $y^c(T), \ldots, y^c_{ICTV}(T)$, and of the class B LEAVs. This completes the calculations for time $T$ so ICT1 is compared to ISAMP to see if these corrected values should be displayed. Next, $T$ is compared to TTEST to see if an interrupt should be serviced. If not, then control is returned to box 201 where calculations for the next time stop are carried out.

If an interruption is to be made, then ISTOP is compared with unity to see what kind of interruption it is. If ISTOP is zero then the simulation is to be interrupted while new values are read in and control is transferred to
box 13 where this is carried out. If ISTOP is one, then this simulation problem is finished, hence the final values in the FPAR and STRMV sectors are printed and if LIST5(2) is nonzero these values are also punched out. Finally the card following the cards associated with the problem just simulated is read in. If it is blank there is another problem to be simulated and control is returned to box six, otherwise execution is terminated.

By placing a sense switch test at box 10 the logic could be easily modified to allow on-line interaction. Hence, after each time step the sense switch would be interrogated. If it is found to be active then control could be transferred to box 13 where new values could be read in via a teletype. Since the system is described by the contents of only two vectors, IVEC and FPVE, the location of any variable or parameter value is accessible through the teletype.

B. Display of Simulation Results

The modeler can choose a graphical or numerical display of the simulation results by setting LIST5(10) to zero or one respectively. He can also choose the frequency of display by assigning values of one, two, etc., to ISAMP. As many as six different variables whose values are located in either the FPAR or STRMV sectors can be
displayed. If ISIN(I) is zero, the value of the Ith displayed variable is in FPAR, while if it is unity the value is in STRMV. JS(I) gives the subsector index while IPOS(I) gives the position in the subsector of the Ith displayed variable.

The flow diagram (figure 54) of subroutine SAVE, which implements the display, shows that on the first call the vector IDIM(I), which gives the location in FPVEC of the Ith displayed variable's value, is constructed. If the values are to be displayed graphically the user must assign values to XLB(I), the lower bound on the plot, and XUB(I), the upper bound. During the first call to SAVE, the XUBs are converted to the spans which are used directly by the plotting algorithm which simply converts each value to be plotted into an integer, say J, greater or equal to one and less than or equal to 101, and assigns a non-blank character to IAR(I) which is printed alphabetically.

C. Solution of the Global Algebraic System

As the reader will remember from the discussion of the analysis package, the global algebraic system consists of KCO ordered decoupled equations of the form

$$G_i(x_i) = 0 \quad i = 1, \ldots, KCO$$

which are solved iteratively by the Newton-Raphson method.
The first $2 \times KCO$ positions of the DGDY sector are used to store the values of the KCO pairs, $(\partial G_i / \partial x_i, G_i)$, $i=1, \ldots, KCO$. The node indices of nodes defined by one or more of these decoupled equations make up the LIST3 sector.

The system also consists of $N_G$ disjoint subsystems, the first, for example, having the form

$$G_i(Y_{KCO+1}, \ldots, Y_{KCO+N_i}) = 0 \quad i=KCO+1, \ldots, KCO+N_1$$

where $N_1$ is the dimension of the first subsystem. The dimension of the $i$th system can be fetched from the $i$th position in the DIM sector of IVEC for which the pointer is IPTMP (see figure 9). There is a list of node indices for the nodes that are described by one or more of the equations in the $i$th subsystem, and there is an area of the DGDY sector (refer to Chapter IV for a discussion of the DGDY sector structure) reserved for the values of that subsystem's augmented Jacobian matrix (AJM).

The pointers for the $i$th subsystem's node list and the $i$th subsystem's area of DGDY are stored as pairs in the PAIRS sector for which the pointer is IPLI; hence the pointer to the $i$th subsystem's area of DGDY can be found in IVEC(IPLI+2*I) and the pointer to its node list in IVEC(IPLI+2*I-1).

In general, no matter whether one of the singly decoupled equations or one of the disjoint subsystems is
being solved, the Newton-Raphson algorithm (see appendix B for derivation) can be written as

$$\Delta Y = -\left( \frac{\partial G}{\partial Y} \right)^{-1} G$$

(54)

where in the former case the quantities are scalars and in the latter they are matrices. Since the solution is obtained iteratively, the algorithm represented by Eqn. 54 will be re-used, for a given singly decoupled equation or a disjoint subsystem, until convergence is obtained, i.e., until

$$|\Delta Y| < \varepsilon$$

where the double bars indicate a norm of some kind. When $\Delta Y$ is a scalar, the norm is simply the absolute value. For each disjoint subsystem, $\Delta Y$ will be a vector and the norm is defined as

$$||\Delta Y|| = \max_i |\Delta Y_i|$$

where $i$ ranges over the indices of the GAVs associated with the subsystem of interest.

On each iteration, the function $G$ will be evaluated in terms of the most recent values of the appropriate GAVs; however, at the user's option, the subsystem inverses need not be re-evaluated at every iteration. In fact, they need not be re-evaluated at every time step. The frequency of subsystem inversion can be specified by
assigning a value to LIST5(1): if its value is one, there will be at least one inversion per subsystem every time step, if its value is two, there will be an inversion every other time step, etc. Whenever there is to be an inversion, the flag IDOCE is set to unity, however even if ICT3 is a multiple of LIST5(1), there are further conditions that must be satisfied before ICODE is set to one. First, before any iterations begin, ICODE will be set to zero if IPC is greater than zero and LIST5(5) is not one, even if ICT3 is a multiple of LIST5(1). Hence, LIST5(5) is a post-correction flag and if it is not set to unity there will be no inversion during the post-correction flag and if it is not set to unity there will be no inversion during the post-correction algebraic calculations. This is done because there is likely to be less change on the correction step than on the prediction step. If LIST5(5) is set to unity, there will still be inversion only on the first iteration.

The user can override these conditions and obtain inversion on every iteration up to the LIST5(7)-th time step by assigning the appropriate positive integer to LIST5(7). This feature can be useful for particularly difficult algebraic systems. These conditions are summarized in table 9 where it is seen that on the first
post-prediction calculations, when IPC is minus one, inversion takes place on every iteration regardless of \( \text{LIST5(5)} \), \( \text{ITER} \), or \( \text{LIST5(7)} \). Note that IPC is also set to minus one on the optional preliminary algebraic pass. This is done since the system variables are likely to be changing most rapidly at the outset of the simulation. For each subsystem, if convergence has not been obtained by the \( \text{LIST5(3)} \)-th iteration, ICODE is set to one and subsequent iterations are carried out with inversion. If no convergence has been obtained by seven iterations, no further attempts are made and the simulation continues as though convergence had been obtained, although warning messages are printed.

<table>
<thead>
<tr>
<th>IPC</th>
<th>( \text{LIST5(5)} )</th>
<th>( \text{ITER} )</th>
<th>INVERSION</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
<td>any value</td>
<td>any value</td>
<td>yes</td>
</tr>
<tr>
<td>0</td>
<td>any value</td>
<td>1</td>
<td>yes</td>
</tr>
<tr>
<td>0</td>
<td>any value</td>
<td>&gt;1</td>
<td>no</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td></td>
<td>no</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>yes</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>&gt;1</td>
<td>no</td>
</tr>
</tbody>
</table>

To further discuss the solution of the algebraic equations reference must be made to figure 55, the flow diagram of ALGCMP, which supervises the calculations.
ENTRY

is KOO = 0 and NG = 0?

no

EPS = DUT(2), ICALL = 3, INST = 3, N = LIST5(1)
ISTAR = IPPZ + 1, ICODE = 0, IBEG = 1 + KCO
IBEGM = KCO, IBEGP = 0, NORM = 0

ICT3: 2

K = ICT3/N, KP = K*N

KP: ICT3

is IPC = 0 and LIST5(5) = 1?

no

ICODE = 1

ICDS = ICODE

KCO = 0

ISA = IPPZ + 2*KCO

ICODE = 1

ITURN = 0, IA = IPPZ + 1

FIGURE 55. ALGOMP
FIGURE 55. (continued)
FIGURE 55. (continued)
FIGURE 55. (continued)
DO 201 I = 1, ICTE
    IARH = IARH + IDIM, IARG = IPTM + IBEGP + I
    FPVEC(IARG) = ABS(FPVEC(IARH + I))

    DO 201 J = 2, ICTE
    TEST = ABS(FPVEC(IARH + J))
    TEST: FPVEC(IARG) > FPVEC(IARG) = TEST

    DO 200
    IARG = IPKCO - IDIM

    DO 203 I = 1, ICTE
    IARG = IARG + IDIM, IARH = IPTM + IBEGP + I

    DO 203 J = 1, IDIM
    FPVEC(IARG + J) = FPVEC(IARG + J)/FPVEC(IARH)

    CALL GJ
    IARG = IPKCO

    DO 105 I = 1, ICTE
    IARG = IARG + IDIM
    FPVEC(IPDZ + IBEGM + I) = FPVEC(IARG)

FIGURE 55. (continued)
ITER 6

IBEG = IBEG, ISO = IBEGM + ICTE

CALL RESET(ISA, ISO, IVEC, FPVEC)

I = ISA - 1

I = I + 1

I: ISO

DUM1 = ABS(FPVEC(IPDZ + I))

DUM1: EPS

ITER: LIST5(3)

ITER: 6

N = 1, LIST5(1) = N
ICODE = 1, ITER = ITER + 1
NORM = 1

IBEG = IBEG + ICTE
IBEGM = IBEG - 1
IBEGP = IBEGM - KCO

FIGURE 55. (continued)
Upon entry, if both KCO and NG are zero, control is transferred to box nine where a call is made to subroutine EXPL which supervises the calculation of the class B LEAVs. If one of these two quantities is not zero, then values are assigned to several flags and the value of ICODE may be changed later on based on the iteration count, its value is stored in ICDS so it can be reset.

Next, if KCO is not zero, control is transferred to box a, where preparations are made for the solution of the singly decoupled equations. First, the appropriate area of the DGDY sector is zeroed if ICODE is unity. Second, a loop is set up wherein, for each of the KCO equations, the following takes place. The node index is fetched from the LIST3 sector and a call is made to subroutine FETCH which, as the reader will recall from the discussion of the analysis package, places, for node NE, the values of IDER, NLDE, NLAV, and NLIAE, in the NODE common area and the values of the FPAR and IPAR subsector pointers in the BASE common area.

At box 39, a subloop is set up in which the iterations for node NE will be executed. The iteration counter, ITER, is incremented, stored in LIST5(4), where it is accessible to other routines in the package, and used to determine ICODE. A call is then made to LOAD, which assigns the appropriate values to the LAVs and LDVs for
node NE, and a call is made to ECALL which transfers control to the appropriate NSR where the values of the $g_i$'s and perhaps the partial derivatives are determined. If IDER and ICODE are both unity a call is made to PERT which supervises the perturbation operations which yield numerical estimates of the partial derivatives. A call to TRANS then transfers the values of $g_i$ and $\partial g_i/\partial y_i$ for node NE to the appropriate locations in the DGDY sector. It is for this routine that ITURN was set to zero before control got into the loop. If ICODE is unity then the scalar inversion is executed and the results are used in the next step to calculate the value of $\Delta Y_i$ which is stored in FPVEC(IB).

A call to RESET adds $\Delta Y_i$ to the appropriate LAV and places the value in the appropriate location in FPVEC. The absolute value of $\Delta Y_i$ is then compared to EPS, whose value is stored in DUT(3), and if convergence is obtained, control is returned to the beginning of the loop where the solution of the next equation commences. If convergence is not obtained, control is returned to box 39 where another iteration is carried out.

The solution procedure for the NG sybsystems is in principle similar to that for the KCO singly decoupled equations except that the dimensionality of the problem
adds a few complications. Box seven signifies the start of a loop in which the IGRP-th subsystem will be solved. First, the pointers for the node list and the DGDY area, IPL1 and IPKCO, and the subsystem dimension, ICTE, are fetched from IVEC. Box two signifies the start of an inner loop in which the ITER-th iteration will be executed. The value of ICODE is modified, if necessary, based on the value of ITER in accordance with the rules presented in table 9. Box three signifies the start of yet another inner loop in which control is passed to each node on the IGRP-th subsystem's node list and the elements in the AJM are determined. Actually, if ICODE is zero, only the elements of the vector function G are computed. After this operation is completed, control is transferred out of this inner loop on to box four where a decision is made as to how the elements of ΔY are to be computed.

If ICODE is zero, then ΔY is calculated by matrix multiplication since the elements of \((\partial G/\partial Y)^{-1}\) have been computed on a previous iteration. Upon completion of the matrix multiplication, control is transferred to box six where RESET is called and the values of \(ΔY_i\), i=IBEG, ...,IBEG+ICTE, are added to the appropriate LAVs. Note that IBEG is the index of the first GAV associated with the IGRP-th subsystem. Then the elements of ΔY are
tested for convergence and, if it is obtained, control is passed to box 2000 where IBEG and other counters derived from it are modified in preparation for the solution of the next subsystem.

If convergence is not obtained, control is transferred to box five where, if ITER is less than LIST5(3), control is transferred to box two where another iteration without inversion will take place. If ITER is not less than LIST5(3) but is less than 7, then the inversion frequency (stored in LIST5(1)) will be set to one, ICODE will be set to one indicating that inversion will take place on the next iteration, and NORM is set to one indicating that a fresh set of normalization elements will be determined. Finally, control is returned to box seven where the solution of the next subsystem commences. Once all the NG subsystems have been solved, control is transferred to box nine where a call is made to EXPL and the class B LEAVs are evaluated.

Now, if inversion is to occur, a decision must be made whether or not a new set of normalization elements for the AJM associated with the IGRP-th subsystem are to be determined. If IPC is less than zero or if ICT3 is a multiple of LIST5(6) or if NORM is one, then control is transferred to box 204 where the maximum element (in
absolute value) of each row of the Jacobian matrix is determined and stored in the RMAX sector. After this, control is passed to box 200 where the AJM is normalized. A call to subroutine GJ follows where the Gauss-Jordan algorithm is applied to the AJM stored in the DGDY sector and the following transformation takes place:

\[
\left( \frac{\partial G}{\partial Y} \right) \begin{bmatrix} G \end{bmatrix} \rightarrow \left( \frac{\partial G}{\partial Y} \right)^{-1} \begin{bmatrix} \Delta Y \end{bmatrix}
\]

(55)

In other words, the last column of the augmented matrix, initially containing the function values, is transformed to the solution vector \( \Delta Y \) and, at the same time, an in-place inversion is executed, which transforms the Jacobian matrix into the inverse matrix. Since the area of the DGDY sector for the IGRP-th subsystem is unmodified during the solution of the other subsystems, the inverse can be stored intact for use, if needed, on subsequent iterations. After control is returned from GJ, the elements in the last column of the AJM are transferred to the portion of the DELTY sector reserved for the IGRP-th subsystem. Control is now transferred to box six which has already been discussed.

D. Solution of the Ordinary Differential Equations

The paragraphs in section H of Chapter III can serve
as an introduction to this section which will consist essentially of a discussion of SRN8P1, the routine that supervises the solution of the ODEs. The sector LIST2, having a length equal to the total number of nodes plus one, contains the node indices of the nodes described by one or more ODEs. For each of these nodes there is a subsector of the SV sector which serves as working storage during the solution process. Table 10 shows that each subsector consists of five so-called slots, each of length NLDE, plus one location used as a code. Note that since NLDE is not necessarily the same for every node, the slot length of one subsector may be different than that of another.

For simplicity of presentation, the reader is asked to assume that the first prediction step (where t=h and IPC= -1) is being carried out. The main loop starts at box 300 (see figure 56), where each node index is fetched in turn from the LIST2 sector. At box 200 it is seen that ISTART is fetched from the director sector for the SV sector. The reader will remember that, unlike the elements of all the other director sectors, ISTART points to the first element of node NE's subsector of SV instead of the element just before that. Program control is next transferred to box 201 where the X and Y sectors are loaded with the values of the LDVs and LAVs for node
ENTRY
INST = 1, ICALL = 3
100
II = 0
100
II = II + 1, NE = IVEC(IPL2 + II)
NE = 0 ≠
ICALL = 4
ICALL = 3 ≠
400
INST 2 call EXPL
(RETURN)
200
ISTART = IVEC(IPSV + NE)
NLDE = IVEC(IPIND + NE)
ICALL = 4
IB1 = IVEC(IPXID + NE)
IDUM = ISTART + 1 ≠ NLDE
DO 202 I = 1, NLDE
VAL = FPVEC(IDUM + I), IARG = IVEC(IB1 + I)
XEX = FPVEC(IARG + I)
VAL = XEX
XEX = FPVEC(IARG + 2)
VAL = XEX
FPVEC(IARG) = VAL
202
100

FIGURE 56. SRN8PL
FIGURE 56. (continued)
FIGURE 56. (continued)
FIGURE 6. (continued)
NE and the appropriate NSR is called. During this call, values of $f_i(x^c(t-h), y^c(t-h)), i=1, \ldots, NLDE,$ are calculated. Note that henceforth whenever the subscript $i$ appears, it will run from 1 to $NLDE$. Also, whenever the symbols $x$ and $y$ appear without subscripts they will be interpreted as vector quantities.

Next, because $ICALL$ is three and $IPC$ is less than zero, the values of $x^c_i(t-h)$ and the just-calculated $f_i$ values are stored in slots one and two, respectively, of SV. Program control is transferred to box $b$ where the simple Euler one step prediction algorithm (Eq. 10) yields values of $x^p_i(t)$ which are then stored in slot five of the SV subsector and control is returned to box 100 where the same procedure is repeated for the next node on LIST2. After this has been done for all of the nodes on LIST2, as signified by a zero value of $NE$ (this is why the LIST2 sector length was chosen to be one greater than the maximum number of nodes), $ICALL$ is changed to four and the loop is restarted at box 300 and this time, for each node, control is passed to box $d$ where the values of $x^p_i(t)$ are fetched from slot five of the SV subsector and compared to the bounds and placed in the appropriate locations in the FPAR or STRMV sectors. At the completion of this pass, control is transferred to
box 400 where the class A LEAVs are evaluated. Hence, the LDVs are not reset until the prediction pass has been completed for all nodes.

If each node's LDVs were reset immediately after prediction then, in the case where node A has an external LDV associated with stream B, which flows from node A to node C, the prediction step calculations at node C might yield \( f_i(x^p(t), y^c(t-h)) \) instead of the desired \( f_i(x^c(t-h), y^c(t-h)) \), if one of node C's external LDVs is associated with stream B.

Now, on the first correction pass (where \( IPC = 1 \) and \( t \) still equals \( h \)) control is again passed to box 201 where, for node NE, values of \( f_i(x^p(t), y^p(t)) \) are calculated. Next, since IPC is greater than zero, control is transferred to box 103 where the X sector is loaded with \( x^c(t-h) \) from the first slot in the SV subsector. Then at box 104, sector FP' is loaded with \( f_i(x^c(t-h), y^c(t-h)) \) from slot two of the SV subsector. Having loaded the X, F, and FP sectors, control is passed to box e where the corrected values of the LDVs, \( x_i^C(t) \), are computed using Eq. 11.

Now, the improvement code is interrogated. Initially, it will be zero because of the preliminary action on the SV sector by the MAINP routine. Hence control will be transferred to box 107 where the \( x_i^C(t) \) are
<table>
<thead>
<tr>
<th>CONTENTS</th>
<th>NO. OF LOCATIONS</th>
<th>SLOT NO.</th>
</tr>
</thead>
<tbody>
<tr>
<td>IMPROVEMENT CODE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(x^c(t-h))</td>
<td>NLDE</td>
<td>1</td>
</tr>
<tr>
<td>(f(x^c(t-h))) or (f(x^c(t-2h)))</td>
<td>NLDE</td>
<td>2</td>
</tr>
<tr>
<td>(x^p(t))</td>
<td>NLDE</td>
<td>3</td>
</tr>
<tr>
<td>(x^c(t)-x^p(t)) or (x^c(t-h)-x^p(t-h))</td>
<td>NLDE</td>
<td>4</td>
</tr>
<tr>
<td>(x^c(t)) or (x^p(t))</td>
<td>NLDE</td>
<td>5</td>
</tr>
</tbody>
</table>

placed in the fifth slot of the SV subsector. If the code is unity, as it will be when \(t\) is greater than \(h\) (because it will have been changed during a previous prediction step), then the difference \(x^c(t)-x^p(t)\) is computed (using \(x^p(t)\) which is stored in slot three) and used to improve the corrected value according to Eq. 15. The difference is stored in slot four of the SV subsector from where it will be fetched on the next prediction step when it will be \(x^c(t-h)-x^p(t-h)\). As with the prediction step, this sequence of operations will be carried out for each node on LIST2, after which control will be transferred to box d where the corrected values will be fetched from slot five of each LIST2 node's subsector of SV and assigned to the appropriate locations in the STRMV or FPAR sectors.
On the second and all subsequent prediction steps (where IPC=0 and t is greater than h) a slightly different path is taken than on the first prediction step. After the call to the NSR which describes node NE, where \( f_i(x^C(t-h), y^C(t-h)) \) is computed, and after the values of \( x^C_i(t-h) \) are stored in slot one of the SV subsector, control is transferred to box 104 where the values of \( f_i(x^C(t-2*h), y^C(t-2*h)) \), stored in slot two of the SV sector, are loaded into the FP sector. Then control is transferred to box 105 where slot two is loaded with the values of the just-calculated values of \( f_i(x^C(t-h), y^C(t-h)) \). Hence on the subsequent correction step, slot two will be able to produce values of \( f_i(x^C(t-h), y^C(t-h)) \) to be used in Eq. 14, and then on the next prediction step, slot two will produce values of \( f_i(x^C(t-2*h), y^C(t-2*h)) \) to be used in Eq. 12.

After slot two has been reloaded, control is transferred to box 106 where the two point predictor formula, Eq. 12, is used to produce values of \( x^P_i(t) \) which are then placed in slot three of SV. Now, the improvement code is interrogated. If \( t=2*h \), the code will still be zero, in which case it will be set to unity and control will be transferred to box 107 where the values of \( x^P_i(t) \) will be loaded into slot five in preparation for
the reset step. If $t$ is greater than $2^*h$, the code will be unity and control will be transferred to box 108 where the predicted values will be improved according to Eq. 14. Note that the improvement process uses the contents of slot four of SV which will contain $x^P(t-h) - x^C(t-h)$ and will have been loaded during a previous correction pass.

The above discussion shows that for every LIST2 node, slot one of the SV subsector is loaded with the current LDV values, or $x^C(t-h)$, at the outset of every prediction step. The contents of slot one are then saved until the outset of the correction step when they are loaded into the X sector to be used in the calculation of the values of $x^C(t)$. Slot two is loaded at the outset of every prediction step with the values of $f(x^C(t-h), y^C(t-h))$. However, except for the first prediction step, just before this loading takes place, the current contents of slot two, which are $f_1(x^C(t-2^*h), y^C(t-h))$, are loaded into the FP sector for use during the two point prediction calculations.

On every prediction step except the first, slot three is loaded with the values of $x^P_i(t)$ just after they are calculated. These values are saved for use on the subsequent correction step when values of the
difference $x^c(t) - x^p(t)$ are calculated. During this correction step the values of this difference are placed in slot four from where they will be retrieved on the subsequent prediction step when the improved predicted values are calculated. Of course, by this time the contents of slot four will be $x^c(t-h) - x^p(t-h)$. Slot five serves as temporary storage for both $x^p(t)$ and $x^c(t)$ until the reset pass is executed.

There are several modifications that could be made to the ODE solution process in future versions of OSUSIM. First, it is suggested by Ralston (34) that use of the corrected value improvement algorithm (Eq. 15) is equivalent to increasing the order of the corrector and may lead to instability problems. Hence this improvement calculation might be suppressed, however, it is important to note that no instability problems were ever observed during all the development and testing of OSUSIM.

Second, no iterations are carried out on the corrected values so future versions might be modified so that the correction step can be repeated until there is an acceptably small change in the corrected value. It is interesting to note that Lapidus suggests (23) that the optimal scheme consists of using a predictor-corrector scheme with no iteration but with both predicted- and corrected-value
improvement calculations (Eq. 13 and 15). This is of course just the scheme that is implemented in OSUSIM.

Third, since the predictor-corrector algorithm has a truncation error which is proportional to $h^3$, higher order methods might be implemented. This would of course require extensive modification of routine SRN8P1 plus slight modification of routine SRN7 which reserves space for the SV sector. If a new algorithm is implemented it probably should be of the predictor-corrector type rather than the Runge-Kutta type because the latter algorithm would require an extensive reformulation of the LDV-LAV relationship as well as the node concept.

E. Evaluation of the LEAVs

If a node is described in terms of LEAVs, then its node index is in the LIST4 sector. If a LEAV is associated with node NE then its code, stored along with its bounds and value, is equal to $-(NE+1)$. This is also the value of the LEAV's GAV index stored in node NE's sub-sector of LISTV. To evaluate the class A LEAVs, EXPL is called by SRN8P1 which sets INST to two before the call. Similarly, the class B LEAVs are evaluated by a call to EXPL from ALGCMP which sets INST to four just before the call.

Referring to figure 57, which gives the flow diagram
FIGURE 57. (continued)
of EXPL, it is seen that the main loop starts at box one
where each node index is fetched in turn from LIST4. The
calls to FETCH, LOAD, and ECALL cause the LEAVs to be
evaluated. Now it remains to discover which of the NLAV
LAVs associated with node NE are explicit so that their
values can be checked for convergence and placed in the
appropriate locations in the FPAR or STRMV sectors. This
is accomplished by setting up an inner loop which inter­
rogates node NE's subsector of the LISTV sector to see if
any of the GAV indices have a value of -(NE+1). If such
a match is found, control is transferred to box a where
the just-calculated value, YNEW, is fetched from the y
sector.

The previous value of the LAV, YOLD, is fetched
from the appropriate location in the FPAR or STRMV sector
using information contained in the LAVDR sector. YNEW
is compared with the LAV's upper and lower bounds and if
it is out of bounds it takes on the boundary value. Then
YNEW and YOLD are compared to see if convergence has been
obtained. If it has, then YNEW is placed in its FPAR or
STRMV sector location and the LAV interrogation proceeds.

If convergence has not been obtained, then IREP is
incremented. If, after all of the LEAVs for node NE have
been checked, IREP is not zero then the whole procedure
is repeated for the same node. If IREP is zero then control is returned to box one where calculations commence for the next node on LIST4. When ICT3 is greater than two, four iterations are allowed; otherwise twenty iterations are allowed.

F. Placing the Local Values in the Augmented Jacobian Matrix

As was mentioned in the chapter describing the analysis package, there are two versions of the subroutine TRANS but both have essentially the same overall objective, viz., to place the values of $g_i$ and $\frac{\partial g_i}{\partial y_j}$, $i=1, \ldots, NLIAE$, $j=1, \ldots, NLAV$, after they have been computed for a selected node, in the proper positions of the augmented Jacobian matrix (AJM). This is one of the most important operations in the OSUSIM compute package because it means that each NSR can be conceived and written on an entirely localized basis.

The basic idea of TRANS (whose flow diagram is shown in figure 58) is to interrogate the GAE index (located in the node's subsector of the LISTE sector) of each LIAE associated with the NE-th node. If the index associated with, say, the LEN-th LIAE falls within prescribed bounds (to be discussed below) then the sign of $g_{LEN}$ is reversed and the resulting value is placed in the AJM. The location in the AJM is determined solely by the type of pass
ENTRY
ISTRE = IVEC(IVEC(IPL + NE)), ISTRV = IVEC(IVEC(IPL + NE))
LEN = 0

LEN = LEN + 1
LEN: NLIAE ➔ RETURN

II = IVEC(ISTRE + LEN)

yes
II = IBEGM & ITURN = 1?

II = KCO and ITURN = 0?

no
no

ITURN = 1 ➔
II: IVEC

ITURN = 1 ➔
II: IDEC

IARG = IPKCO + (II - IBEGM) * IDIM

FPVEC(IARG) = - FPVEC(IARH)

ICODE: 1 ➔

IARG = IPPZ + 2 * II

FIGURE 58. TRANS
Filmed as received
without page(s) 312.

UNIVERSITY MICROFILMS.
FIGURE 58. (continued)
(whether a singly decoupled equation or a subsystem is being solved) and by the GAE index.

Associated with the LEN-th LIAE, whose value has just been placed, are at most NLAV partial derivatives, \( \partial g_{\text{LEN}} / \partial y_{\text{LVARN}} \), \( \text{LVARN}=1,\ldots,\text{NLAV} \), and if ICODE is unity, that is, if an inversion is to take place on this pass, then the value of those partial derivatives for which the \( y_1 \) has a positive GAV index (see box a) must be placed in the AJM. Actually, the conditions are more restrictive than this because first, if a singly decoupled GAE is being solved then only one partial derivative, the one for which the GAE and GAV indices are the same (see box c), need be placed in the DGDY sector.

Second, if a subsystem is being solved, then the GAV index must be greater than KCO and also IBEGM where IBEGM is one less than the GAV index of the first GAV associated with the subsystem of interest (see boxes b and d). The GAV index should also be less than or equal to IBEGM+ICTE where ICTE is the dimension of the subsystem; hence in future versions of OSUSIM, TRANS might be so modified. As a result of not satisfying this last condition the value of an unnecessary partial derivative is placed in a currently unused portion of the DGDY sector, so at worst, a small amount of computation time is wasted.
The conditions that must be satisfied before a value of $q_{\text{LEN}}$ can be placed in the AJM can be described in the following. First, if the $I_l$-th singly decoupled equation is being solved, then ALGCMP will have set ITURN equal to zero and IDEC equal to $I_l$; hence ITURN must be zero and $I_l$, the GAE index associated with $q_{\text{LEN}}$ must equal IDEC. If a subsystem is being solved, then the GAE indices of the equations in this subsystem will range from $\text{IBEGM}+1$ through $\text{IBEGM}+\text{ICTE}$ and ALGCMP will have set ITURN to unity. Hence, boxes e, f, g, and h, require that these conditions be satisfied before any placement of values in the AJM is allowed to take place.

G. The Transformation of the Augmented Jacobian Matrix By Means of the Gauss-Jordan Method

The function of the subroutine GJ, whose flow diagram is shown in figure 59, is to effect the transformation represented by Eq. 55. As has been mentioned before several times, each subsystem has an area of the DGDY sector reserved for it where the rows of its AJM are stored linearly. Subroutine GJ is designed so that to effect the transformation, by applying the Gauss-Jordan reduction, only the pointer (IPKCO) to the area of DGDY and the dimension (ICTE) of the subsystem need be known.
ENTRY
IPPZ = IPKCO

DO 100 I = 1, ICTE
IVEC(IPFO + I) = I
100

DET = 1., IAI = IPPZ - IDIM

DO 171 NR = 1, ICTE
IAI = IAI + IDIM
NR: ICTE = 1

IARG = NR + 1, IARH = IAI + NR
TERM = ABS(FPVEC(IARH)), IMAX = NR

DO 2 I = IARG, ICTE
IARH = IARH + IDIM
DEN = ABS(FPVEC(IARH))

TERM: DEN < IMAX = I, THRM = DEN

NR: IMAX = e

IT = IVEC(IPFO + NR)
IVEC(IPFO + NR) = IVEC(IPFO + IMAX)
IVEC(IPFO + IMAX) = IT

IA2 = IPPZ + (IMAX - 1) * IDIM

DO 105 J = 1, IDIM
TERM = FPVEC(IA1 + J)
FPVEC(IA1 + J) = FPVEC(IA2 + J)
FPVEC(IA2 + J) = THRM
105

(ENTRY)
IPPZ = IPKCO

DO 100 I = 1, ICTE
IVEC(IPFO + I) = I
100

DET = 1., IAI = IPPZ - IDIM

DO 171 NR = 1, ICTE
IAI = IAI + IDIM
NR: ICTE = 1

IARG = NR + 1, IARH = IAI + NR
TERM = ABS(FPVEC(IARH)), IMAX = NR

DO 2 I = IARG, ICTE
IARH = IARH + IDIM
DEN = ABS(FPVEC(IARH))

TERM: DEN < IMAX = I, THRM = DEN

NR: IMAX = e

IT = IVEC(IPFO + NR)
IVEC(IPFO + NR) = IVEC(IPFO + IMAX)
IVEC(IPFO + IMAX) = IT

IA2 = IPPZ + (IMAX - 1) * IDIM

DO 105 J = 1, IDIM
TERM = FPVEC(IA1 + J)
FPVEC(IA1 + J) = FPVEC(IA2 + J)
FPVEC(IA2 + J) = THRM
105

(ENTRY)
IPPZ = IPKCO

DO 100 I = 1, ICTE
IVEC(IPFO + I) = I
100

DET = 1., IAI = IPPZ - IDIM

DO 171 NR = 1, ICTE
IAI = IAI + IDIM
NR: ICTE = 1

IARG = NR + 1, IARH = IAI + NR
TERM = ABS(FPVEC(IARH)), IMAX = NR

DO 2 I = IARG, ICTE
IARH = IARH + IDIM
DEN = ABS(FPVEC(IARH))

TERM: DEN < IMAX = I, THRM = DEN

NR: IMAX = e

IT = IVEC(IPFO + NR)
IVEC(IPFO + NR) = IVEC(IPFO + IMAX)
IVEC(IPFO + IMAX) = IT

IA2 = IPPZ + (IMAX - 1) * IDIM

DO 105 J = 1, IDIM
TERM = FPVEC(IA1 + J)
FPVEC(IA1 + J) = FPVEC(IA2 + J)
FPVEC(IA2 + J) = THRM
105

FIGURE 59. GJ
DEW - FFVEC(lAl+ NR)

FPVEC(IA1+NR) = 1*, DET = DET*DEN

\[ y = \frac{DO 172 J = 1, IDIM}{\text{FPVEC(IA1+J) = FPVEC(IA1+J)/DEN}} \]

IA2 = IPPZ - IDIM

DO 173 I = 1, ICTE

IA2 = IA2 + IDIM

I: NR

TERM = FPVEC(IA2 + NR)

FPVEC(IA2 + NR) = 0.

DO 174 J = 1, IDIM

FPVEC(IA2 + J) = FPVEC(IA2 + J) - TERM*FPVEC(IA1 + J)

I = 0

I = I + 1

J = IVEC(IPPO + I)

I: J

IARG = IPPZ + I

DO 104 K = 1, ICTE

TERM = FPVEC(IARG), FPVEC(IARG) = FPVEC(IARG + J - I)

FPVEC(IARG + J - I) = TERM, IARG = IARG + IDIM

IVEC(IPPO + I) = IVEC(IPPO + J), IVEC(IPPC + J) = J

FIGURE 59. (continued)
During the implementation of the Gauss-Jordan method (40), in order to reduce round-off error and avoid division by zero, a search, down the column from which the divisor will be taken, is made for the element largest in absolute value after which the row containing this element is exchanged with the row on which the division operations were to be carried out. To see the effect of the row interchange, consider the following nth order linear system:

\[ \sum_{j=1}^{n} A_{ij} x_j = b_i, \quad i=1, \ldots, n \]  \hspace{1cm} (56)

If \( A_{ij}^{-1} \) represents the elements of the inverse, the solution can be obtained formally as follows:

\[ \sum_{i=1}^{n} A_{ki}^{-1} \sum_{j=1}^{n} A_{ij} x_j = \sum_{j=1}^{n} x_j \sum_{i=1}^{n} A_{ki}^{-1} A_{ij} = \sum_{i=1}^{n} A_{ki}^{-1} b_i \]  \hspace{1cm} (57)

Now by virtue of the definition of the inverse, Eq. 57 becomes

\[ \sum_{j=1}^{n} x_j \delta_{kj} = x_k = \sum_{i=1}^{n} A_{ki}^{-1} b_i \]  \hspace{1cm} (58)

If rows in the A matrix are exchanged then the index \( i \) in Eqs. 56 and 58 is changed. This means first, that the order of the unknowns, signified by the index \( k \), is unaffected and second, that the columns of the inverse are
exchanged. Therefore, since an in-place inversion (22) is carried out, a record of the row switchings must be kept. This is done by defining the IPOS sector such that the \textit{Ith} position of this sector (signified by IPOS(I) in what follows is the original row number of the present \textit{Ith} row. Initially IPOS(I) will be set equal to I (see box b). Now, if during the transformation, row K must be exchanged with row J, then the contents of IPOS(K) and IPOS(J) are exchanged (see box c).

After the transformation is complete, then for I-1, ..., ICTE, if the value of IPOS(I), represented by J, is not equal to I then the Jth and Ith columns in the inverse are exchanged. After the column exchange, the contents of \textit{IPOS(I)} and \textit{IPOS(J)} are exchanged (see box d) so that columns will not be re-exchanged when the remaining positions in IPOS are tested.

The actual Gauss-Jordan reduction can be described by following the flow diagram. For the purpose of simple presentation, let A (I,J) represent the element of the \textit{Ith} row and \textit{Jth} column of the AJM. The basic idea is to divide the NR-\textit{th} row by the diagonal element A(NR,NR) and to multiply the elements of row NR by A(I,NR) and subtract the result from row I for I=1,...,NR. This operation ordinarily would leave a one in the (NR,NR) position and
zeroes in the other positions of the NRth column. And after it has been applied to every row, the coefficient matrix would be transformed into the identity matrix while the solution vector will reside in the last column of the augmented matrix. However, since an in-place inversion is being carried out, just before the division process on row NR, the (NR,NR) element is set to one. Similarly, just before the multiplications are carried out on the rows, the (I,NR) element is set to zero.

In earlier versions of OSUSIM the inverse was obtained by the following transformation

\[ ((\frac{\partial G}{\partial Y}); \{G\};(I)) \rightarrow ((I); \{\Delta Y\}; (\frac{\partial G}{\partial Y})^{-1}) \]

Hence, instead of carrying out an in-place inversion, the identity matrix was appended to the Jacobian matrix and the G vector and a conventional Gauss-Jordan reduction was applied to the augmented matrix. It is therefore seen that the in-place inversion saves considerable space.

H. Resetting the GAVs after the Solution of an Algebraic System

Subroutine RESET, whose flow diagram is shown in figure 60, is designed to add the values of \( Y_i, i=ISA,\ldots,ISO, \) obtained from the solution of a singly decoupled
ENTRY

DO 7 I = ISA, ISO

IGLOB = IVEC(IPG + 1)
OLD = FPVEC(IGLOB)
VAL = OLD + FPVEC(IPDZ + 1)
YEX = FPVEC(IGLOB + 1)

VAL: YEX

YEX = FPVEC(IGLOB + 2)

VAL: YEX

FPVEC(IGLOB) = VAL

RETURN

FIGURE 60. RESET(ISA, ISC, IVEC, FPVEC)
equation (in which case ISA will equal ISO) or from the solution of a disjoint subsystem, to the old values of $Y_i$ stored in either the FPAR or STRMV sector, as specified by the GAVDR sector. Note that if the new value is out of bounds the boundary value is taken as the GAV's value and $\Delta Y_i$ is modified (see box nine). This is done because after control is returned to the calling routine, ALGCMNP, the $\Delta Y_i$ value(s) are tested against epsilon, stored in DUT(3). Hence, if $\Delta Y_i$ were not modified by RESET, out-of-bounds values could be consistently calculated and convergence would never be obtained. However, by modifying $\Delta Y_i$, two successive out-of-bounds values would cause $\Delta Y_i$ to be set to zero and $Y_i$ would be treated as though it had converged to a boundary value.

I. Diagnostic Output

As was mentioned in part A of this section, the flag ISK is set to zero whenever diagnostic output (DO) is desired. If ISK is assigned the value of one in the input data then there will be no DO at all. If ISK is set to zero initially, then there will be DO for every time step until ICT3 is greater than LIST5(8). Almost every routine in the compute package is coded to print diagnostic information when ISK is zero. First, MAINP prints the contents of the FPAR and STRMV sectors as well as titles announcing that a prediction or correction step is about to take
place. It would be helpful to have the value of $T$ also printed out and future versions OSUSIM should be so modified.

During the solution of the algebraic equations when ALGCMP is in supervisory control, the node index, the function value, and the partial derivatives can be printed for each of the singly decoupled equations and the appropriate area of the DGDY sector before reduction can be dumped for each subsystem. After the reduction, the contents of the appropriate area of the DELTY sector can also be printed. During the Gauss-Jordan reduction, the value of the Jacobian determinant can be printed. The EXPL routine which supervises the evaluation of the LEAVs is coded so that a zero value of ISK causes the node index, the LEAV index, and the new and old LEAV values to be printed.

During the solution of the ODEs, supervised by SRN8Pl, the node index, and the values of ICALL and IPC, can be printed as well as the contents of the X sector just after a prediction or correction is made at each node. Regarding the NSRs there is no set rule for printing out diagnostic information; however, since it is expected that the user will be designing, coding and debugging his own NSRs, it is sufficient to merely make him aware of the potential use of ISK.
CHAPTER VIII
THE SIMULATION OF THE EXAMPLE PROBLEMS

For each example problem discussed in section five, the punched card output from the analysis package is augmented with a title card and the contents of two Fortran Namelists and presented as input to the compute package. The results of the computations are discussed in this section.

A. The Linear Multiple Recycle System
1. Arrangement of the Input Data

Figure 61 shows a listing of the punched card input where it is seen that, after the title card, which is supplied by the user, the next 56 cards are taken directly from the output of the analysis package. Namelist IO shows that the values of the quantities associated with streams 21, 13, and 2 are to be displayed in plot form since LIST 5(10) is zero. The significance of the other values assigned to the elements of Namelists IO and IO1 can be deduced by referring to part A of Chapter VII. The initial values of all the system variables whether internal or external are zero except for stream 30 which has a "flow rate" of 1.0. Hence, the simulation will show the
| FIGURE 61. INPUT DATA TO COMPUTE PACKAGE, EXAMPLE NO. 1 |
FIGURE 61. (continued)
response of the system to a step change in the inlet stream.

2. Analytical Solution

Figure 30, which shows the digraph for this system, can be simplified to the digraph shown in figure 62 because nodes 2, 3, 4, 5, and nodes 9, 10, 11, 12, and nodes 19, 18, 17, 16, all have the following structure:

Since there are no dynamic elements in this group of nodes, the value of the stream A variable is always equal to that of stream B.

To check the output of the compute package, an analytical solution will be developed for the value associated with stream 20 and since the nodes are all defined by linear relations the Laplace Transform technique will be used. Let \( G_S \) and \( G_A \) be the splitter node and accumulator node transfer functions, then by starting at stream 20 and proceeding against the flow, the following can be obtained:

\[
y_{20} = G_A G_S y_8
\]  
\( (59) \)
FIGURE 62. SIMPLIFIED MULTIPLE RECYCLE SYSTEM
Since all the splitters divide the inlet flow equally between the two outlets, \( Y_9 \) equals \( Y_8 \) and further analysis yields

\[
Y_8 = Y_9 = G_S G_A Y_1 / (1 - G_S G_A) \tag{60}
\]

Now, since stream one is effectively the sum of streams 30, 20, and 31, and since the flow of stream 31 is identical to that of stream 20, the following is obtained:

\[
Y_1 = 2Y_{20} + Y_{30} \tag{61}
\]

Combining Eqs. 59, 60, and 61, and solving for \( Y_{20} \) yields

\[
Y_{20} = \frac{G_A^2 G_S^3}{1 - G_S G_A - 2G_A^2 G_S^2} Y_{30} \tag{62}
\]

The transfer function for the splitter is simply \( 1/2 \) while that for the accumulator is easily shown to be \( 1/(Rs+1) \), where \( R \) is the resistance parameter associated with the outlet flow rate. Setting \( Y_{30} \) equal to \( 1/s \) since the behavior of stream 30 is described by a unit step function, and replacing \( G_A \) and \( G_S \) in Eq. 62 gives

\[
Y_{20} = \frac{1}{8R^2} \frac{1}{(S^2 + \frac{3}{2R} S + \frac{1}{4R^2})} \frac{1}{S}
\]

which can be inverted to give
3. Discussion of the Simulation Results

Figure 63 shows the plotted output for the case where ISAMP=2, TNXT=50., DELT=.5, and the bounds for the three stream variables are 0.0 and 3.0. The value of the time for each plotted point is given in the far left hand column. The plot shows that, qualitatively, the three quantities behave as expected.

Figure 64 shows the contents of the FPAR and STRMV sectors at the conclusion of the simulation when T=TNXT. Consider the STRMV sector whose first element is the third number in row five. Since the stream data was originally presented to the analysis package in the order of increasing stream number (see figure 31) it is relatively simple to pick out the value in figure 64 associated with each stream.

First, the values of streams 20 and 31 (fourth number in row 20 and third number in next-to-last row, respectively) have, as expected, the same value. Second, the value of stream 21 (third number in 21st row) is the sum of the values of streams 20 and 31. Third, the value of stream 5 (fourth number in row eight) is, as expected, twice that of stream two (second number in row six). Fourth, if Eq. 63

\[
y_{20} = \frac{2}{\sqrt{20}} \left( \frac{e^{at}}{-3+\sqrt{5}} - \frac{e^{bt}}{-3-\sqrt{5}} \right) + \frac{1}{2}
\]

\[
a = \frac{-3+\sqrt{5}}{4R} \quad b = \frac{-3-\sqrt{5}}{4R}
\]
FIGURE 63. PLOTTED OUTPUT FOR EXAMPLE NO. 1
FIGURE 64. FINAL VALUES OF STRMV AND FPAR SECTORS
is evaluated for $T=50.0$ with $R=9.0$, the value obtained is $0.297446$ which is to be compared with the stream 20 value shown in figure 64 of $0.297438$. These results indicate that the computations supervised by the compute package are correct.

B. The Solids Drying System with Feedforward and Feedback Control

Figure 65 shows a listing of the punched card input. The contents of the IVEC vector are unchanged from the time when they were produced as output from the analysis package. The punched cards containing the contents of the FPVEC vector are different, having been produced as output from a previous compute package run wherein LIST5(2) was set equal to two. The object of that previous run, which used as input the FPVEC vector from the analysis package output, was to let the system "line out" thereby producing steady state conditions. These steady state conditions describe the initial state of the system which will be subjected to the sinusoidal forcing generated in node seven.

The contents of Namelist 10 show that all six channels will be used to display in plot form (since LIST5(10) is zero) the following quantities:
FIGURE 65. INPUT DATA TO COMPUTE PACKAGE, EXAMPLE NO. 2
| Q.1C00000E C2 | C.599549C7E C3 | C. C | C.10000000E C4 | 0.69240137E 03 |
| 0.0 | 0.10000000E 04 | 0.97500000C E2 | 0.0 | 0.10000000E C4 |
| 0.0 | 0.10000000E 02 | 0.60152368E 03 | 0.0 | 
| C.1725439E C2 | 0. C | 0.10000000E C4 | 0.97500000E 02 |
| 0.0 | 0.10000000E 04 | 0.54680672E-01 | 0.0 | C.10000000C C1 |
| C.0 | 0.10000000E 04 | C.0 | 0.10000000E 04 | -0.20000000E 01 |
| -C.2CCCCCE C0 | 0.05262573E 03 | C.10000000C C1 | -C.2CCCCCE C0 | 0.65185815E 03 |
| 0.49399808E-01 | 0.0 | C.0 | 0.10000000E 04 | 
| C.0 | 0.10000000E 04 | -0.20000000E 01 | 0.43462276E-01 | 0.0 |
| 0.20000000E 04 | 0.37149128E-01 | C.0 | 0.10000000E C4 | 
| C.64836621E C3 | C. C | C.10000000E C1 | -C.20000000E C1 |
| 0.0 | 0.10000000E 04 | 0.99999942E-02 | 0.0 | 
| 0.1C000000E C0 | -C.2CCCCCCE C0 | 0.6416C449E 03 | 0.0 | 
| 0.17506182E-01 | 0.0 | C.1C000000E C4 | -0.20000000E 01 |
| C.0 | 0.10000000E 04 | 0.10000000E 04 | 0.69222E52E C3 | 
| 1.0 | 0.10C00000E 10 | -0.20000000E 01 | 
| 0.0 | 0.10000000E 01 | -0.30000000E 01 | 0.15 |
| 0.19073486E-C5 | 0.1847744CE-05 | 0.95555542E-02 | 0.0 |
| 0.05 | 0.10000000E 02 | 0.95599976E C3 | 0.67200000E C3 | 
| C.7CCCCCE C0 | 0.67200000E 03 | 0.73000000E 03 | 0.73000000E 03 |
| 0.73000000E 03 | -0.60000000E 01 | 0.73000000E 03 | 0.73000000E 03 |
| C.24414063E-03 | 0.0 | 0.73000000E 03 | 0.73000000E 03 |
| C.7CCCCCE C0 | 0.7CCCCCE C3 | C.7CCCCCE C3 | 0.7CCCCCE C3 |
| 0.70000000E 03 | 0.70000000E 03 | 0.70000000E 03 | 0.70000000E 03 |
| C.7CC0000E C3 | 0.7CC0000E C3 | 0.7CC0000E C3 | 0.7CC0000E C3 |
| 0.7CC0000E C3 | 0.7CC0000E C3 | 0.7CC0000E C3 | 0.7CC0000E C3 |
| 0.7CC0000E C3 | 0.7CC0000E C3 | 0.7CC0000E C3 | 0.7CC0000E C3 |
| 0.7CC0000E C3 | 0.7CC0000E C3 | 0.7CC0000E C3 | 0.7CC0000E C3 |
| 0.7CC0000E C3 | 0.7CC0000E C3 | 0.7CC0000E C3 | 0.7CC0000E C3 |
| 1.5707563 | 0.23455989E 06 | 0.0 | 0.10000000E 04 |
| C.56000000E C3 | 0.0 | 0.10000000E 04 | 0.78000000E 03 |

FIGURE 65. (continued)
<table>
<thead>
<tr>
<th>JS</th>
<th>UNIT</th>
<th>POS</th>
</tr>
</thead>
<tbody>
<tr>
<td>2, 1, 1, 3, 5, 7, 5</td>
<td>1, 8, 1, 1, 1</td>
<td></td>
</tr>
<tr>
<td>ISIN = 6 * 1</td>
<td>DELET = .00125, ISK = 1</td>
<td></td>
</tr>
<tr>
<td>NT = 6, ISAMP = 4</td>
<td>LIST5 = 1, 0, 1, 2 * C, 10 C, 0, 3, 0, 0, 20 * 0</td>
<td></td>
</tr>
<tr>
<td>DUT = 60, .05, .005, 11 * 0, 162, 5000, 5000, 670, -10, 630</td>
<td></td>
<td></td>
</tr>
<tr>
<td>XUB = 164, 8000, 6000, 77 C, 1 C, 73 C</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

FIGURE 65 (continued)
<table>
<thead>
<tr>
<th>channel</th>
<th>quantity</th>
<th>bounds</th>
<th>units</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>inlet solids moisture</td>
<td>0.1620, 0.1640 mass fraction</td>
<td>units</td>
</tr>
<tr>
<td>2</td>
<td>inlet solids flow rate</td>
<td>5000, 8000. lbs/hr</td>
<td>lbs/hr</td>
</tr>
<tr>
<td>3</td>
<td>outlet solids flow rate</td>
<td>5000, 8000. lbs/hr</td>
<td>lbs/hr</td>
</tr>
<tr>
<td>4</td>
<td>set point to the servo</td>
<td>670, 770. deg. Fahr.</td>
<td>deg. Fahr.</td>
</tr>
<tr>
<td>5</td>
<td>feedback control signal</td>
<td>-10, 10. deg. Fahr.</td>
<td>deg. Fahr.</td>
</tr>
<tr>
<td>6</td>
<td>steam temperature</td>
<td>630, 730. deg. Fahr.</td>
<td>deg. Fahr.</td>
</tr>
</tbody>
</table>

Figure 66 shows the resulting plot where channel two represents the forcing and channel one represents the response. Since DELT is 60.0 (see figure 65) and since the units of DELT are hours, the printed value of the time at the base of the plot is in units of minutes.

The location in IVEC of the feedforward delay was discussed in part B.-4. of chapter VI and found to be position 158. Therefore from figure 65 it is seen that the delay is 30*DELT and since DELT is 0.00125 hrs or 0.075 minutes, then the delay is 2.25 minutes. As can be seen from the plot, the outlet moisture variation is within ±0.001 mass fraction unit and appears to be increasing with time while the solids inlet flow rate which is the forcing signal varies sinusoidally from 6000.0 to 7800.0 lbs/hr with a period of six minutes. Other runs with smaller feedforward delays showed more variation in the outlet moisture.
FIGURE 66. PLOTTED OUTPUT FOR EXAMPLE NO. 2
Since only 30 locations were reserved for the push-left list associated with node six, the maximum delay that could be prescribed was 30*DELT. For longer delays the analysis package would have to be rerun with the data in node six's IPAR and FPAR subsectors slightly modified. The variation of the feedback control signal (channel 5) is less than ±4 deg. Fahr., while that of the total control signal is on the order of ±12 deg. Fahr., indicating that the feedback component contributes roughly one third of the total signal.

Other runs were made with (a) different forcing frequencies (requiring a change in one element of FPVEC), (b) no feedforward control (requiring a few changes in the TMCN NSR), (c) no feedback control (requiring a change in one element of FPVEC), (d) step change forcing in the flow rate (requiring changes in two elements of FPVEC), and (e) different feedback and servo controller constants (requiring, at most, changes in four elements of FPVEC). Since there was no change in the system topology during these runs and since the changes in the one NSR were minor, there was no need to make another run with the analysis package, therefore the changes were simply made in the input to the compute package.
C. The Double Effect Evaporator System

Figure 67 shows a listing of the punched card input. Like the previous example this system was allowed to reach a steady state by setting LIST5(2) to unity and replacing the cards containing the FPVEC values after each run with the punched output. In this way the simulation was continued several times until steady state was reached. The system was forced by changing the velocity in stream one which is the inlet to the first effect (see figure 47), from 8.19 ft/sec to 7.1 ft/sec. The following quantities are displayed:

<table>
<thead>
<tr>
<th>channel</th>
<th>quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>liquid level, #1 effect</td>
</tr>
<tr>
<td>2</td>
<td>velocity, stream 3</td>
</tr>
<tr>
<td>3</td>
<td>control pressure, stream 6</td>
</tr>
<tr>
<td>4</td>
<td>liquid level, #2 effect</td>
</tr>
<tr>
<td>5</td>
<td>velocity, stream 9</td>
</tr>
<tr>
<td>6</td>
<td>control pressure, stream 14</td>
</tr>
</tbody>
</table>

Figure 68 shows the plot display of these quantities for the first three minutes after the step change. Channel one, the level in the first effect, is seen to drop and then rise in response to the valve-closing control action represented by channel three. This control action causes the outlet velocity of the stream from the first effect (channel two) to drop. The level in the second effect
2 EFFECT EVAP UNDER CONTROL

1  7  13  33  39  45  51  291  57  77  19  302  2  393  498  487  292  295  298  277
397  301  315  319  221  250  89  63  55  121  375  379  71  288  64  9  11  10  90
288  107  114  119  124  131  136  1  11  22  41  51  62  141  142  143  146  147  148  81
97  105  125  137  141  145  161  165  181  197  209  213  217  222  238  244  260  271  277  175
188  197  200  214  223  164  167  169  172  175  5  8  11  5  8  11  1  2  3
4  5  6  0  1  2  3  4  5  6  0  0  0  0  0  3  1  3
2  1  3  13  9  3  14  9  3  3  2  0  4  2  0  151  154  155  158  16C
161  226  239  248  251  265  274  1  2  3  4  5  0  0  3  6  7  0  0  8
5  6  0  0  7  4  10  11  12  C  C  10  14  5  C  G  13  12  14  0
0  1 -1  1 -1  1 -1 -1  1  1  1  122  138  134  20  28  31  142  154  21C
60  68  71  218  1  4  5  3  6  2  9  7  8  10  11  0  0  1  0 -2
0  0  4  2  -3  -3  -5  0  5  4  -2  3  C  6  -3  2  -3  C  C  C  -4
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
C  0  0

C.292CCC0CE  C3  C.1CCCCCCCCE  02  0.500CCCCCE  C1  0.11099941E  00  0.60000000E  03
0.10000000E  01  0.20000000E  02  0.36080436E  0C  C.C
C.4CCCC0000E  01  0.11999995E  00  0.0  0.15000000E  C2  C.200000CCE  C1
0.0  0.10000000E  01  0.15000000E  C2  C.20000000E  C2  C.1CCCC000CE  C2
0.30000000E  01  0.50000000E  01  0.24979439E  01  0.0  0.50000000E  C1
C.25000000E  03  C.5CCCC000E  02  0.24979439E  01  0.0  0.50000000E  C1
0.85640554E  01  0.30000000E  01  0.15000000E  C1  0.15000000E  C2  0.86668615E  01  0.30000000E  01
0.15000000E  02  0.40000000E  01  0.20790100E  C2  0.207133CEC  C2  0.20637512E  c2
C.2C561218E-02  0.10000000E  02  0.50000000E  01  0.11099941E  00  0.60000000E  03
0.10000000E  01  0.20000000E  02  0.46164485E  CC  C.C
C.1CCCC000E  02  0.11999995E  00  0.0  0.15000000E  C2  C.30000000E  C1
C.35000000E  02  0.10000000E  02  0.35555585E-02  C1  0.10000000E  02  0.65310564E  01
0.30000000E  01  0.15000000E  02  0.20000000E  02  C.1CCCC000CE  C2  C.1CCCC000CE  C2  0. C

FIGURE 67. INPUT DATA TO COMPUTE PACKAGE, EXAMPLE NO. 3
<table>
<thead>
<tr>
<th>Number</th>
<th>X Value</th>
<th>Y Value</th>
<th>X Value</th>
<th>Y Value</th>
<th>X Value</th>
<th>Y Value</th>
<th>X Value</th>
<th>Y Value</th>
<th>X Value</th>
<th>Y Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.25000000E03</td>
<td>0.50000000E02</td>
<td>C.24877672E C1</td>
<td>C. C</td>
<td>0.50000000E0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.58558795E01</td>
<td>0.30000000E01</td>
<td>0.15000000E02</td>
<td>0.64715099E 01</td>
<td>C.30000000E C1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.15000000E02</td>
<td>C.7CCCCCCE C1</td>
<td>C.12557C3CE-C1</td>
<td>0.12448311E-01</td>
<td>0.12340546E-01</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.12232789E-01</td>
<td>0.41000000E01</td>
<td>0.0</td>
<td>0.0</td>
<td>C. C</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.56CCCCCCE C2</td>
<td>C. C</td>
<td>0.0</td>
<td>0.0</td>
<td>C.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C.0</td>
<td>C.10CCCCCE 03</td>
<td>C.10000000C0E C1</td>
<td>C.15599999E C0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C.0</td>
<td>-0.10000000E01</td>
<td>0.69043887E C0</td>
<td>0.0</td>
<td>C.50000000E C1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-C.2CCCCCCE 01</td>
<td>C.685CCC0CE 03</td>
<td>C. C</td>
<td>0.99900000E 03</td>
<td>-0.10000000E C1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.19699997E02</td>
<td>0.0</td>
<td>0.0</td>
<td>C.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C.0</td>
<td>0.50000000E02</td>
<td>0.20000000E01</td>
<td>0.67468799E C3</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.99900000E03</td>
<td>-C.20000000C0E 01</td>
<td>C.68757C29E C1</td>
<td>0.0</td>
<td>C.50000000E C1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.30000000E01</td>
<td>0.231651C7E 00</td>
<td>0.0</td>
<td>C.90000000E C1</td>
<td>-C.10000000E C1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C.36C75872E 00</td>
<td>0.0</td>
<td>0.10000000E 02</td>
<td>-0.50000000E 01</td>
<td>0.58500000E 03</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.0</td>
<td>C.595CCC0CE C3</td>
<td>-C.1CCCC0CE C1</td>
<td>C.73E40C17E 01</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C.99000000E02</td>
<td>-0.10000000E 01</td>
<td>0.2498105C 01</td>
<td>C.0</td>
<td>C.50000000E C1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-C.1CCCCCCE C1</td>
<td>0.8666551E 01</td>
<td>C.30000000C0E C1</td>
<td>0.15000000E 02</td>
<td>-0.10000000E 01</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.71180630E 01</td>
<td>0.0</td>
<td>0.0</td>
<td>C.50000000C0E C1</td>
<td>0.67468799E 03</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.0</td>
<td>0.99900000E 03</td>
<td>-0.30000000E 01</td>
<td>0.30389633E 01</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.50000000E02</td>
<td>0.60000000C0E 01</td>
<td>C.231651C7E C0</td>
<td>0.0</td>
<td>0.90000000E 01</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.30000000E01</td>
<td>0.25000000E01</td>
<td>0.0</td>
<td>0.0</td>
<td>-C.10000000C0E C1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C.55868998E C1</td>
<td>C.0</td>
<td>0.99000000E02</td>
<td>0.11000000E02</td>
<td>0.63350269E 03</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.0</td>
<td>C.1CCCCC0CE C4</td>
<td>-0.60000000C0E 01</td>
<td>C.15000000C0E C2</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.0</td>
<td>-0.10000000C0E 01</td>
<td>0.28416038E 00</td>
<td>0.0</td>
<td>C.10000000C0E C2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-C.6CCCCC0CE 01</td>
<td>0.55689989E 01</td>
<td>C. C</td>
<td>0.99000000E 02</td>
<td>0.70000000E 01</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.63350269E 03</td>
<td>0.0</td>
<td>0.0</td>
<td>C.1CCCC0CE C4</td>
<td>-0.50000000C0E 01</td>
<td>0.20027084E 01</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.0</td>
<td>0.15000000E02</td>
<td>0.80000000E01</td>
<td>0.28416038E 00</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.10000000C0E 02</td>
<td>-0.10000000C0E 01</td>
<td>C.46165C1E C0</td>
<td>0.0</td>
<td>0.90000000E 01</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.90000000E01</td>
<td>0.58500000E03</td>
<td>0.0</td>
<td>C.99500000C0E C3</td>
<td>-C.10000000C0E C1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.15599991E C1</td>
<td>0.0</td>
<td>0.0</td>
<td>-0.10000000C0E 01</td>
<td>0.24895178E 01</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.0</td>
<td>0.50000000C0E 01</td>
<td>-C.10000000C0E 01</td>
<td>C.25000000C0E C1</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.0</td>
<td>-0.10000000E01</td>
<td>0.65010986E C1</td>
<td>0.30000000C0E C1</td>
<td>0.15000000E C2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

FIGURE 67. (continued)
FIGURE 67. (continued)
(channel four) is relatively unaffected for approximately one minute, then it also begins to drop in response to the decreased inlet velocity. This decrease in level causes a valve-closing control action represented by channel six which is the second controller's output pressure to the valve, and this valve closing causes the system outlet velocity, channel five, to drop. The upper and lower bounds for the plot were obtained by observing the results of a preliminary run with numerical output.

The basic routines that make up the compute package require 5952 words of IBM 360/75 digital computer memory. For the evaporator system described in this section, 750 locations were reserved for IVEC and 1200 for FPVEC, while the common area, which is problem independent, required 141 words. Three NSRs and three supporting routines required 1956 words. A summary of the memory requirements is given below:

<table>
<thead>
<tr>
<th>Category</th>
<th>Words</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compute package routines</td>
<td>5952</td>
</tr>
<tr>
<td>Common area</td>
<td>142</td>
</tr>
<tr>
<td>IVEC &amp; FPVEC</td>
<td>1950</td>
</tr>
<tr>
<td>NSRs and supporting routines</td>
<td>1956</td>
</tr>
<tr>
<td>Dummy NSRs</td>
<td>294</td>
</tr>
<tr>
<td>System /360 utility routines</td>
<td>7312</td>
</tr>
<tr>
<td>Total</td>
<td>107606</td>
</tr>
</tbody>
</table>
The link-edit step required approximately 10 seconds while the execution step that produced figure 66 required 15 seconds.
CHAPTER IX

SUMMARY, CONCLUSIONS AND RECOMMENDATIONS

In the foregoing, the logical structure of a dynamic modular simulation program, OSUSIM, has been presented and applied to three example problems. When compared to other dynamic MSPs, OSUSIM has the following distinctive features.

First, the node models can be described in terms of not only ODEs and explicit algebraic equations, but also semi-explicit algebraic equations requiring localized iteration and completely implicit algebraic equations which, depending on the nature of the other nodes in the digraph, can be members of a global implicit algebraic system of equations requiring global iteration. In the process of compiling this global algebraic system, two lists are constructed giving the index of the global algebraic variable associated with each local algebraic variable for each node (the GAV list) and the index of the global algebraic equation associated with each local algebraic equation (the GAE list). These lists in effect define the structure of the global system and supply the information needed for the decoupling operations. The inclusion of implicit algebraic equations in the node
description means that OSUSIM can be used to solve problems that the other dynamic MSPs can not. This feature even makes OSUSIM an attractive tool for solving steady state flow distributions and pressure losses in pipeline networks since such a network can easily be represented by a digraph and since the global algebraic variable identification process is made to order for stream pressures and velocities.

Second, the node describing subroutines (NSRs) are designed so that the user need not be concerned with how the local variables are assigned their values on each computation pass. This is possible because, during the analysis phase, lists are constructed giving the value-location for each local algebraic and differential variable.

Third, the data structure is completely flexible; there are no fixed boundaries. Each stream in the digraph can be defined, by the modeler, in terms of any number of variables. OSUSIM then reserves only the necessary number of memory locations for each stream. Similarly, each node can be defined in terms of any number of parameters and differential and algebraic equations. This is possible because all the data is stored in two variable length
vectors, one real-valued and one integer-valued. Therefore, the user can determine and assess the location of every variable and parameter if necessary. To assist the user in this process, there is available a supplementary program which can read the punched cards produced by the analysis package and print a detailed summary of the analysis step. Although it is not necessary to use this supplementary program, it can be helpful if later on the user plans to change the initial values of parameters and variables. This data structure flexibility, of course, places a burden on the user-modeler since he must be familiar with the logical structure of OSUSIM before he can make optimal use of it. However, because of this flexibility, the spectrum of simulation problems that can be solved is much broader.

It is not envisioned that there will be any great advantage in developing an extensive library of NSRs except that a wide range of NSRs will serve as a helpful guide to the modeler when he constructs his own special purpose NSRs. Some NSRs, such as those describing PID controllers, might be used in several different simulation problems, but in general the user of OSUSIM should expect to construct the majority of the NSRs for each problem. Consequently it is not expected or intended that OSUSIM will provide the
rank and file engineer with a design aid. Instead OSUSIM is designed to provide the mathematical modeler with a means of applying the modular concept to simulation. This means different things to different workers in the field, and to this writer it simply means that the modeler has the freedom to concentrate on localized modeling without being burdened with global structure or numerical algorithm details.

Scattered throughout the thesis are suggestions for program improvement, but they mostly pertain to coding changes. Regarding the computing aspect, there are two main areas where improvement can be made. First, the predictor-corrector method used for solving the ODEs could be made more sophisticated: a higher order method would probably allow for larger time steps and decreased computation time. The reader is referred to the conclusion of part D of chapter VII for further comments.

Second, the implementation of the Newton-Raphson method used to solve the global algebraic system of equations might be streamlined. This is mentioned not because there is any obvious modification to be made, but because for typical systems described by both the ODEs and GAEs at least 75% of the computing time is spent solving the GAEs.
APPENDIX A

DERIVATION OF THE PREDICTOR-CORRECTOR ALGORITHM

Both algorithms are derived from fixed grid quadrature formulas of the type

$$\int_{x_i}^{x_k} f(x)dx = \sum_{i=1}^{n} A_i f(x_i) + R[f(x)]$$  \hspace{5cm} (64)

where \(x_i = x_1 + (i-1)h\). After choosing \(j, k, n,\) and \(x_1\), the values of the \(n\) coefficients can be determined by setting \(R(f(x))\) equal to zero for \(f(x) = x^{m-1}\), with \(m=1, 2, \ldots, n\). This process generates an \(n\)th order system of linear equations solvable for the \(n A_i\)'s. The expression for \(R(f(x))\) can be shown (26) to be

$$R[f(x)] = \frac{f^{(n)}(3)}{n!} R(x^n), \quad x_j < z < x_k$$

Therefore, having determined the \(n\) coefficients, the expression for the error term can be determined from Eq. 64 by letting \(f(x) = x^n\). Now if \(f(x) = dx/dt\), then the predictor and corrector formulas can be derived. For example, if \(j=2, k=3,\) and \(n=2\), one can obtain

$$x_3 = x_2 + \frac{h}{2} \left[ -f(x_1) + 3f(x_2) \right] + \frac{5}{12} h^3 f(2)(z)$$  \hspace{5cm} (65)
which is the predictor formula used in OSUSIM. Next, if
j=1, k=2, and n=2, one can obtain

\[ x_2 = x_1 + \frac{h}{2} [f(x_1) + f(x_2)] - \frac{h^3}{12} f''(z) \]  

which is the corrector formula for OSUSIM.

In order to derive the algorithm for improving the
predicted and corrected values, Eqs. 65 and 66 will be re­
written as

\[ x_{i+1}^p = x_i + \frac{h}{2} [-f(x_{i-1}) + 3f(x_i)] \]  

\[ x_{i+1}^c = x_i + \frac{h}{2} [f(x_i) + f(x_{i+1})] \]

where the upper case symbols represent the actual calcula­
ted values and the lower case symbols represent the true
values. Now, if it can be assumed that

\[ f(x_i) = f(x_i), \quad f(x_{i-1}) = f(x_{i-1}), \quad \text{and} \quad f(x_{i+1}) = \]

\[ f(x_{i+1}) \]

then the following equations can be written

\[ x_{i+1} - x_{i+1}^p = \frac{a}{c} q \]  

\[ x_{i+1} - x_{i+1}^c = \frac{b}{c} q \]
where $q$ is $h^3 f^{(2)}(z)$, $a$ is 5, $b$ is -1, and $c$ is 12.

Eliminating $q$ gives

$$\frac{c}{a} (x_{i+1} - x^p_{i+1}) = \frac{c}{b} (x_{i+1} - x^c_{i+1})$$

which can be rearranged to give

$$x_{i+1} = x^c_{i+1} + \frac{b}{a-b} (x^c_{i+1} - x^p_{i+1})$$

which is the improved corrected value algorithm. In a similar fashion, Eqs. 69 and 70 can be combined to give

$$x^c_{i+1} - x^p_{i+1} = \frac{b-a}{c} q$$

Now, the assumption is made that $x^c_{i+1} - x^p_{i+1} = x^c_i - x^p_i$, therefore the following equation can be written

$$x^c_i - x^p_i = \frac{b-a}{c} q$$

and when $q$ is eliminated between Eqs. 70 and 72 the improved predictor algorithm results:

$$x_{i+1} = x^p_{i+1} + \frac{a}{b-a} (x^c_i - x^p_i)$$

Hence both Eqs. 71 and 73 simply estimate the true value, $x_{i+1}$, based on the assumption that $q$ is relatively constant. The only difference is that Eq. 71 is applied after the prediction step which supplies $x^p_{i+1}$ while Eq. 73 is applied after both the prediction and correction.
step, the latter supplying $x^c_{i+1}$. 
APPENDIX B
DERIVATION OF THE NEWTON-RAPHSON ALGORITHM

In what follows, all symbols, with or without subscripts, will be treated as vector quantities unless otherwise specified. The problem is defined as follows: find \( Y \) such that \( G(Y) = 0 \). A truncated Taylor's series can be written as

\[
\]  

where \( G'(Y) \) is the Jacobian matrix, having elements \( \frac{\partial G_i}{\partial Y_j} \), evaluated at \( Y \). By definition, \( G(Y) = 0 \), therefore Eq. 74 can be written as

\[
\Delta Y = Y - Y = (G'(Y))^{-1}(G(Y) - E)
\]  

If \( E \) were known, Eq. 75 could be solved for the true solution \( Y \). However, \( E \) is not known but because \( E \) behaves as follows (33)

\[
\lim_{\Delta Y \to 0} \left\{ \frac{E}{\Delta Y} \right\} = 0
\]

it is set to zero and the following recursive relation replaces Eq. 75:

\[
\Delta Y_{i+1} = Y_{i+1} - Y_i = -(G'(Y_i))^{-1}(G(Y_i)), \ i=1,2,...
\]
where the subscript $i$ indicates the $i$th iteration. The sequence $Y_1, Y_2, \ldots$ generated by Eq. 76 is continued until the following inequality is satisfied:

$$|\Delta Y_i| < \varepsilon$$

In some cases the inverse is not re-evaluated at every iteration, so the recursive relation becomes

$$\Delta Y_{i+1} = -(G'(Y_i))^{-1}(G(Y_i))$$
APPENDIX C
MEMORY MAPS
<table>
<thead>
<tr>
<th>NAME</th>
<th>ORIGIN</th>
<th>LENGTH</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAIN</td>
<td>00</td>
<td>1EF4</td>
</tr>
<tr>
<td>MXR</td>
<td>1EF8</td>
<td>242</td>
</tr>
<tr>
<td>STFK</td>
<td>2140</td>
<td>2FE</td>
</tr>
<tr>
<td>SPTR</td>
<td>2440</td>
<td>27C</td>
</tr>
<tr>
<td>LOAD</td>
<td>26C0</td>
<td>28A</td>
</tr>
<tr>
<td>SBSTK</td>
<td>2940</td>
<td>1266</td>
</tr>
<tr>
<td>MKST</td>
<td>3C0A</td>
<td>6A0</td>
</tr>
<tr>
<td>ZER0X</td>
<td>42A8</td>
<td>1B4</td>
</tr>
<tr>
<td>MAINP</td>
<td>4430</td>
<td>708</td>
</tr>
<tr>
<td>ECALL</td>
<td>4839</td>
<td>576</td>
</tr>
<tr>
<td>DECU</td>
<td>5080</td>
<td>660</td>
</tr>
<tr>
<td>JACO</td>
<td>5710</td>
<td>24C</td>
</tr>
<tr>
<td>CCCUR</td>
<td>5960</td>
<td>20C</td>
</tr>
<tr>
<td>SETUP0S</td>
<td>5C30</td>
<td>6A0</td>
</tr>
<tr>
<td>JOU7</td>
<td>62D0</td>
<td>29C</td>
</tr>
<tr>
<td>INTLZE</td>
<td>6570</td>
<td>91A</td>
</tr>
<tr>
<td>SRN7</td>
<td>61090</td>
<td>4E2</td>
</tr>
<tr>
<td>SHOVE</td>
<td>7278</td>
<td>21C</td>
</tr>
<tr>
<td>INP</td>
<td>7498</td>
<td>69C</td>
</tr>
<tr>
<td>SWTCH</td>
<td>7828</td>
<td>22B</td>
</tr>
<tr>
<td>CNLST</td>
<td>7050</td>
<td>23C</td>
</tr>
<tr>
<td>TLST</td>
<td>8010</td>
<td>23B</td>
</tr>
<tr>
<td>ADDL</td>
<td>9249</td>
<td>272</td>
</tr>
<tr>
<td>GLOB</td>
<td>6400</td>
<td>27A</td>
</tr>
<tr>
<td>PERT</td>
<td>8750</td>
<td>4B4</td>
</tr>
<tr>
<td>COMPX</td>
<td>8808</td>
<td>4B6</td>
</tr>
<tr>
<td>FETCH</td>
<td>9090</td>
<td>1FF</td>
</tr>
<tr>
<td>RESET</td>
<td>9290</td>
<td>276</td>
</tr>
<tr>
<td>SRN4</td>
<td>9508</td>
<td>1EC</td>
</tr>
<tr>
<td>TRAN5</td>
<td>96F8</td>
<td>49A</td>
</tr>
<tr>
<td>ND1</td>
<td>9894</td>
<td>FF8</td>
</tr>
<tr>
<td>ND2</td>
<td>9C90</td>
<td>FF6</td>
</tr>
<tr>
<td>LAG</td>
<td>9D8H</td>
<td>FF6</td>
</tr>
<tr>
<td>LLCN</td>
<td>9E80</td>
<td>FF6</td>
</tr>
<tr>
<td>TMCN</td>
<td>9F78</td>
<td>FF6</td>
</tr>
<tr>
<td>CHWS</td>
<td>A070</td>
<td>FF8</td>
</tr>
<tr>
<td>IHONAMEL*</td>
<td>A168</td>
<td>C67</td>
</tr>
<tr>
<td>IHOCOMH*</td>
<td>AE68</td>
<td>DC4</td>
</tr>
<tr>
<td>IHOCOMH2*</td>
<td>BC90</td>
<td>975</td>
</tr>
<tr>
<td>ND3</td>
<td>C603</td>
<td>FF6</td>
</tr>
</tbody>
</table>

**NAME ORIGIN LENGTH**

<table>
<thead>
<tr>
<th>NAME</th>
<th>ORIGIN</th>
<th>LENGTH</th>
</tr>
</thead>
<tbody>
<tr>
<td>ND4</td>
<td>C700</td>
<td>F6</td>
</tr>
<tr>
<td>PRFS</td>
<td>C7E8</td>
<td>F6</td>
</tr>
<tr>
<td>PUVA</td>
<td>C8F0</td>
<td>F6</td>
</tr>
<tr>
<td>TANK</td>
<td>C9E3</td>
<td>F6</td>
</tr>
<tr>
<td>VALV</td>
<td>CAE0</td>
<td>F6</td>
</tr>
<tr>
<td>IHUFCVTH*</td>
<td>C808</td>
<td>A07</td>
</tr>
<tr>
<td>IHUENFT*</td>
<td>D5E0</td>
<td>7C8</td>
</tr>
<tr>
<td>IHGFIO*</td>
<td>DDA9</td>
<td>1100</td>
</tr>
<tr>
<td>IHUFCOS2*</td>
<td>EEA8</td>
<td>5AC</td>
</tr>
<tr>
<td>IHUJERRM*</td>
<td>F458</td>
<td>5F4</td>
</tr>
<tr>
<td>IHGFCON*</td>
<td>10178</td>
<td>4C2</td>
</tr>
<tr>
<td>IHGCTRCH*</td>
<td>10640</td>
<td>2A6</td>
</tr>
<tr>
<td>IHUQATBL*</td>
<td>108E8</td>
<td>18E</td>
</tr>
<tr>
<td>IHUQSEN*</td>
<td>10A00</td>
<td>110</td>
</tr>
<tr>
<td>IHUQFEN*</td>
<td>10B30</td>
<td>4C2</td>
</tr>
<tr>
<td>IHUQCONO*</td>
<td>10E1B</td>
<td>9E</td>
</tr>
<tr>
<td>IOUT</td>
<td>10EB3</td>
<td>C</td>
</tr>
<tr>
<td>BASE</td>
<td>10FC8</td>
<td>8</td>
</tr>
<tr>
<td>POIINT</td>
<td>10F00</td>
<td>9C</td>
</tr>
<tr>
<td>NODE</td>
<td>10F70</td>
<td>78</td>
</tr>
<tr>
<td>LISTS</td>
<td>10FC8</td>
<td>78</td>
</tr>
<tr>
<td>EOB</td>
<td>11060</td>
<td>C</td>
</tr>
<tr>
<td>SETP</td>
<td>11070</td>
<td>14</td>
</tr>
<tr>
<td>TEMP</td>
<td>11088</td>
<td>10</td>
</tr>
</tbody>
</table>

**ENTRY ADDRESS** 00

**TOTAL LENGTH** 11098
<table>
<thead>
<tr>
<th>NAME</th>
<th>ORIGIN</th>
<th>LENGTH</th>
<th>NAME</th>
<th>ORIGIN</th>
<th>LENGTH</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAIN</td>
<td>00</td>
<td>1FBC</td>
<td>ND5</td>
<td>*</td>
<td>DAF8</td>
</tr>
<tr>
<td>ALGCMP</td>
<td>1FC0</td>
<td>1C8</td>
<td>VALV</td>
<td>*</td>
<td>DBF0</td>
</tr>
<tr>
<td>EXPL</td>
<td>3188</td>
<td>5F0</td>
<td>VAPR</td>
<td>*</td>
<td>DCE8</td>
</tr>
<tr>
<td>SAVE</td>
<td>3778</td>
<td>9F8</td>
<td>IHOFCVTH*</td>
<td>DE58</td>
<td>A07</td>
</tr>
<tr>
<td>ECALL</td>
<td>40F0</td>
<td>576</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RESET</td>
<td>4668</td>
<td>286</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GJ</td>
<td>48F0</td>
<td>720</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>COMPX</td>
<td>5010</td>
<td>F6</td>
<td>IHOEFNTH*</td>
<td>E860</td>
<td>7C8</td>
</tr>
<tr>
<td>SRN4</td>
<td>5108</td>
<td>1EC</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ZEROX</td>
<td>52F8</td>
<td>184</td>
<td>DUHR</td>
<td>*</td>
<td>F028</td>
</tr>
<tr>
<td>PERT</td>
<td>5480</td>
<td>484</td>
<td>IHOEFIOS*</td>
<td>F108</td>
<td>1100</td>
</tr>
<tr>
<td>LOAD</td>
<td>5908</td>
<td>28A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FETCH</td>
<td>5BC8</td>
<td>1FE</td>
<td>IHOFIO2*</td>
<td>102D8</td>
<td>5AC</td>
</tr>
<tr>
<td>TRANS</td>
<td>50C8</td>
<td>4A6</td>
<td>IHOFEIOS*</td>
<td>10888</td>
<td>5G</td>
</tr>
<tr>
<td>MAINP</td>
<td>6270</td>
<td>CA4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SRN8P1</td>
<td>6F18</td>
<td>BEE</td>
<td>IHOERM*</td>
<td>114F0</td>
<td>5F4</td>
</tr>
<tr>
<td>MKLST</td>
<td>7808</td>
<td>F6</td>
<td>CLOCKI</td>
<td>*</td>
<td>11AE8</td>
</tr>
<tr>
<td>PUVA</td>
<td>7C00</td>
<td>628</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PRES</td>
<td>8228</td>
<td>83E</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FRIC</td>
<td>8D89</td>
<td>350</td>
<td>IHOSSC*</td>
<td>118E0</td>
<td>208</td>
</tr>
<tr>
<td>HETR</td>
<td>9108</td>
<td>66C</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TANK</td>
<td>9778</td>
<td>85A</td>
<td>IHOFCONI*</td>
<td>11DE8</td>
<td>315</td>
</tr>
<tr>
<td>LLCN</td>
<td>A2D8</td>
<td>53E</td>
<td>IHOFCON*</td>
<td>12100</td>
<td>4C2</td>
</tr>
<tr>
<td>MXR</td>
<td>A818</td>
<td>29C</td>
<td>IHOETRCH*</td>
<td>125C8</td>
<td>2A6</td>
</tr>
<tr>
<td>ND2</td>
<td>AAB8</td>
<td>32C</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LAG</td>
<td>ADE8</td>
<td>4A2</td>
<td>IHOETYRCH*</td>
<td>125C8</td>
<td>2A6</td>
</tr>
<tr>
<td>TMCN</td>
<td>B290</td>
<td>2B6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SFTK</td>
<td>B548</td>
<td>2FE</td>
<td>IHOCUATBL*</td>
<td>12870</td>
<td>1E8</td>
</tr>
<tr>
<td>SPTR</td>
<td>B848</td>
<td>27C</td>
<td>IHOOUOPT</td>
<td>12A58</td>
<td>318</td>
</tr>
<tr>
<td>ND1</td>
<td>BAC8</td>
<td>F6</td>
<td>IHOQFTEN*</td>
<td>12070</td>
<td>110</td>
</tr>
<tr>
<td>IHUSLOG*</td>
<td>BAC8</td>
<td>1D4</td>
<td>IHOFTEN*</td>
<td>12E80</td>
<td>198</td>
</tr>
<tr>
<td>CHWS</td>
<td>8D98</td>
<td>F6</td>
<td>IHOQCONI*</td>
<td>13018</td>
<td>A0</td>
</tr>
<tr>
<td>IHUSEXP*</td>
<td>8E90</td>
<td>180</td>
<td>IHOQCONO*</td>
<td>13088</td>
<td>9E</td>
</tr>
<tr>
<td>IHOFRXPR*</td>
<td>CO40</td>
<td>183</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IHOECOMH*</td>
<td>C1C8</td>
<td>DC4</td>
<td>INOUT</td>
<td>13158</td>
<td>C</td>
</tr>
<tr>
<td>IHOECOMH2*</td>
<td>CF90</td>
<td>975</td>
<td>POINT</td>
<td>13168</td>
<td>9C</td>
</tr>
<tr>
<td>ND3</td>
<td>0F08</td>
<td>F6</td>
<td>LISTS</td>
<td>13208</td>
<td>7B</td>
</tr>
<tr>
<td>ND4</td>
<td>DA00</td>
<td>F6</td>
<td>NODE</td>
<td>13280</td>
<td>7B</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>CSAVE</td>
<td>132F8</td>
<td>84</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>TRALG</td>
<td>13380</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>BASE</td>
<td>13388</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>CSRN4</td>
<td>13390</td>
<td>10</td>
</tr>
</tbody>
</table>

**ENTRY ADDRESS** 00  
**TOTAL LENGTH** 133AO
APPENDIX D

PROGRAM LISTINGS
SUBROUTINE HETRIV, FP
C
C POSITION CONTROLLER WITH MEAS LAG AND RESPONSE LAG
C STRM 1 - SET PT - IVBAC OF Y(1) STARTING AT IPY1
C STRM 2 - MEAS VALUE - IVBAC OF Y(2) (BEFORE MEAS LAG) AT IPY2
C STRM 3 - - CONTROLLER RESPONSE - IVBAC OF X(3) (AFTER LAG TAU1)
C AT IPX3
C
C DX(1)/DT = (Y(2)-X(1))/TAUM
C ER= Y(1)-X(1)
C Y(3) = X(2)+XKC*(ER+TAUD*EDOT),
C X(2)/(T=0) = INITIAL
C DX(2)/DT = XKC*ER/TAUI
C DX(3)/DT = (Y(3)-X(3))/TAUL
C
C IPAR SUBSECTOR - IPY1,IPY2,IPX3 (VALUE POSITIONS OF Y(1),Y(2),X(3)
C IN RESPECTIVE STREAMS)
C
C FPAR SUBSECTOR - TAUM,TAUL,TAUD,TAUI,XKC (POSITIONS 1-5)
C - IVAB OF X(1),X(2), (POSITIONS 6-11)
C - IVBAC OF Y(3) (POSITIONS 12-15)
C - ERROR STORAGE (POSITIONS 16-19)
C
COMMON/POINT/IPKP,IPPAR,IP1P,IPSV,IPLV,IPLE,IPN0C,IP1L1,IP1L2,
*IPNL,IPST,IPPO,KCO ,IPPZ,IPTM,IPDZ,IPX,IPF,IPFP,IPG ,IPKCO,
*IPY,IPTHL,IPPY,IP1,IP2,IPAE,IPAV,IPXID,IPYID,IPES,IPYS,
*IPDER,IP1L3,IP1L4,ICTE,ICTV,IDIM,IPROD
COMMON/NODE/OUT(15),DELT,ICALL,INST,IPC,NE,NLDE,NLAE,NLAV,IDER,
*NLXAV,ICODE,ITURN,ICT,ISK,IDE
COMMON/BASE/IR,II
COMMON/LISTS/LIST5(30)
DIMENSION IV(1),FP(1)

C

GO TO (1,2,3),ICALL
C
1 CONTINUE
C
IDER=0
NLAE=0
NLAV=3
NLDE=3
NLXAV=1
C
CALL COMPX(1,6,1,2,0,IV,FP)
IPX3=IV(I+3)
CALL COMPX(0,IPX3,3,1,3,IV,FP)
RETURN
C
2 CONTINUE
IPY1=IV(I+1)
IPY2=IV(I+2)
CALL MKLST(1,1,1,IPY1,1,IV,FP)
CALL MKLST(2,2,1,IPY2,1,IV,FP)
CALL MKLST(3,0,0,12,1,IV,FP)
RETURN
C
3 CONTINUE
TAUM=FP(IR+1)
TAUL=FP(IR+2)
TAUD=FP(IR+3)
TAUI=FP(IR+4)
XKC=FP(IR+5)
C
IF(TAUM.EQ.0.)FP(IPX+1)=FP(IPY+2)
ER=FP(IPY+1)-FP(IPX+1)
IF(INST.NE.1)GO TO 4
C
C CALC RHS OF ODEs
C
IF(TAUM.EQ.0.)GO TO 5
FP(IPF+1)=(FP(IPY+2)-FP(IPX+1))/TAUM
GO TO 6
5 CONTINUE
FP(IPF+1)=0.
6 CONTINUE
IF(TAU.I.EQ.0.)GO TO 7
FP(IPF+2)=XKC*ER/TAUI
GO TO 8
7 CONTINUE
FP(IPF+2)=0.
8 CONTINUE
IF(TAUL.EQ.0.)GO TO 9
FP(IPF+3)=(FP(IPY+3)-FP(IPX+3))/TAUL
RETURN
9 CONTINUE
FP(IPF+3)=0.
FP(IPX+3)=FP(IPY+3)
RETURN
C
4 CONTINUE
IF(INST.NE.2)RETURN
C
Y(3) IS A CLASS A EXPLICIT LAV
C
IF(IPC.GT.0.OR.LIST5(4).NE.1)GO TO 10
C
PUSH ERROR LIST DOWN
C
  IP=IR+16
  DO 100 I=1,3
  FP(IP)=FP(IP+1)
  IP=IP+1
  100 CONTINUE
C
  FP(IR+19)=ER
  EDOT=(ER-FP(IR+18))/DELT
  FP(IPY+3)=FP(IPX+2)+XKC*(ER+TAUD*EDOT)
C
  RETURN
END

**********************************************************************************
SUBROUTINE LAG(IV,FP)
C
  PURE TRANSPORT LAG OF DURATION TAU FOR N VARIABLES - TAU=DELT*M
C
  STREAM 1 - IVBAC OF N INPUT VARIABLES - Y(1) THRU Y(N)
C
  STREAM 2 - IVBAC OF N OUTPUT VARIABLES - Y(N+1) THRU Y(N+N)
       WHICH ARE CLASS B LEAVS
C
  FPAR SUBSECTOR - TAU PLUS M*N CELLS FOR THE PUSH DOWN LISTS FOR
       THE N VARIABLES (N*M+1 POSITIONS)
C
  IPAR SUBSECTOR - N, M (2 POSITIONS)
C
  COMMON/LISTS/LIST5(30)
  COMMON/POINT/NG ,IPPAR,IPIP,IPSV,IPLV,IPLE,IPNEC,IPLI,IPL2,
  *IPNL,D,IPST,IPPO,KCO ,IPPDZ,IPX,IPF,IPFP,IPG ,IPKCO,
*IPY, IPTHL, IPPY, IP1, IPTMP, IPAE, IPAV, IPXID, IPYID, IPES, IPYS, IPl, IPTMP, IPAE, IPAV, IPXID, IPYID, IPES, IPYS, LAG 18
*IPDK1, IPL3, IPL4, ICTE, ICTV, IDIM, IPROD LAG 19
COMMON/CSAVE/XLB(6), XUB(6), T, JS(6), IPOS(6), ISIN(6), ICT2, NT LAG 20
COMMON/NODE/OUT(15), DELT, ICALL, INST, IPC, NE, NLDE, NLA, NLAV, IDER LAG 21
*NLXAV, ICODE, ITURN, ICT3, ISK, IDEC LAG 22
DIMENSION IV(1), FP(1) LAG 23
COMMON/BASE/IR, II LAG 24

C GO TO (1, 2, 3), ICALL LAG 25
C
1 CONTINUE
IDER = 0
N = IV(II + 1)
NLAV = 2 * N
NLXAV = N
NLA = 0
NLDE = 0
RETURN
C
2 CONTINUE
N = IV(II + 1)
CALL MKLST(1, 1, 1, 1, N, IV, FP)
NN = N + 1
CALL MKLST(NN, 2, 1, 1, N, IV, FP)
RETURN
C
3 CONTINUE
IF (INST .NE. 4) RETURN
IF (T .EQ. 0.) RETURN
IF (LIST5(4) .GT. 1) RETURN

C
N = IV(II + 1)
M=IV(I+2)

C    IF(IPC.GT.0)GO TO 4

C ON POST PREDN PASS

C PULL OFF BOTTOM VALUE

      IP=IR+2
      DO 100 I=1,N
      IA=I+N+IPY
      FP(IA)=FP(IP)
      IP=IP+M
   100 CONTINUE

C PUSH LIST LEFT

      MN=M+N-1
      IF(MN.EQ.0)RETURN
      IP=IR+2
      DO 101 I=1,MN
      FP(IP)=FP(IP+1)
      IP=IP+1
   101 CONTINUE

C RETURN

4 CONTINUE

C ADD TO RIGHTMOST POSITION OF LIST

      IP=IR+1
DO 102 I=1,N  
IP=IP+M  
IA=IPY+I  
FP(IP)=FP(IA)  
102 CONTINUE  
C  
RETURN  
CEND  
  
*********************  
SUBROUTINE LLCN(IV,FP)  
C  
SERVO WITH EXPLICIT (CLASS A EXPLICIT LAVI CONTROL ALGORITHM  
C  
STRM 1 INPUT SET PT TO BE FOLLOWED - IVBAC OF Y(1)  
C  
STRM 2 OUTPUT OF SERVO - IVBAC OF X(3) (EXT LDV)  
C  
DX(1)/DT = (X(3)-X(1))/TAUM  
E =Y(1)-X(1)  
DX(2)/DT = XKC*E/TAUI  
X(2)(T=0) = IV OF CONTROL PRESS Y(2)  
Y(2) = X(2)+XKC*(E+TAUD*EDOT)  
Y(2) IS CONTROLLER OUTPUT  
DX(3)=(Y(2)-X(3))/TAUL  
C  
IPAR SUBSECTOR - IPY1,IPX3 (POS IN STRMV SUBSECTOR WHERE Y(1)  
AND X(3) ARE STORED  
C  
FPAR SUBSECTOR - TAUM,TAUL,TAUD,TAUI,XKC (POS 1-5)  
- IVBAC OF X(1) AND X(2) (POS 6-11)  
- IVBAC OF Y(2) (POS 12-15)  
- ERROR STORAGE (POS 16-19)  
C  
DIMENSION IV(1),FP(1)  
COMMON/BASE/IR,II
COMMON/LISTS/LIST5(30)
COMMON/POINT/IPKP,IPPAR,IPIP,IPSV,IPLV,IPLE,IPNEC,IPPL1,IPPL2,
*IPNLD,IPST,IPPO,KCO,IPPZ,IPTM,IPDZ,IPX,IPF,IPFP,IPG,IPKCO,
*IPY,IPTHL,IPPY,IP1,IP2,IPAE,IPAV,IPXID,IPYID,IPES,IPYS,
*IPDER,IPPL3,IPL4,ICTE,ICTV,IDIM,IPROD
COMMON/NODE/OUT(15),DELT,ICALT,INST,IPC,NE,NLDE,NLAE,NLAV,IDER,
*NLXAV,ICODE,ITURN,ICT,ISK,IDE

C GO TO (1,2,3),ICALL

1 CONTINUE
IDER=0
NLAV=2
NLAE=0
NLXAV=1
NLDE=3

CALL COMPX(1,6,1,2,0,IV,FP)
IPX3=IV(I+2)
CALL COMPX(0,IPX3,3,1,2,IV,FP)
RETURN

2 CONTINUE
IPYI=IV(I+1)
CALL MKLST(1,1,1,IPYI,1,IV,FP)
CALL MKLST(2,0,0,12,1,IV,FP)
RETURN

3 CONTINUE
ER=FP(IPY+1)-FP(IPX+1)
IF(INST.EQ.1)GO TO 4
IF(INST.NE.2)RETURN

IF(IPC.GT.0.OR.LIST5(4).NE.1)GO TO 5

PUSH ERROR LIST TO LEFT IF ON POST PRED PASS

IP=IR+16
DO 100 I=1,3
FP(IP)=FP(IP+1)
IP=IP+1
100 CONTINUE

5 CONTINUE
FP(IR+19)=ER

EDOT=(ER-FP(IR+18))/DELT
FP(IPY+2)=FP(IPX+2)+FP(IR+5)*(ER+FP(IR+3)*EDOT)
RETURN

4 CONTINUE
FP(IPF+1)=(FP(IPX+3)-FP(IPX+1))/FP(IR+1)

TAU=FP(IR+4)
IF(TAU.EQ.0.)GO TO 6
FP(IPF+2)=FP(IR+5)*ER/TAU
GO TO 7

6 CONTINUE
FP(IPF+2)=0.
7 CONTINUE
FP(IPF+3)=(FP(IPY+2)-FP(IPX+3))/FP(IR+2)
RETURN
SUBROUTINE MXR(IVEC,FPVEC)

C NSR NO. 1 REQUIRING NUMERICAL ESTIMATION OF PARTIAL DERIVATIVES

C Y(1) + Y(2) - Y(3) = 0.
C Y(1) = 1ST STREAM FLOW RATE (INLET)
C Y(2) = 2ND STREAM FLOW RATE (INLET)
C Y(3) = 3RD STREAM FLOW RATE (OUTLET)

COMMON/3ASE/IBPA, IBIP
COMMON/POINT/IPKP, IPPAR, IPP, IPSV, IPLV, IPLE, IPNEC, IPL1, IPL2,
*IPNL, IPST, IPPO, KCO, IPZ, IPTM, IPDZ, IPX, IPF, IPFP, IPG, IPKCO,
*IPY, IPY1, IPY2, IP1, IP2, IPAE, IPAV, IPXID, IPYID, IPES, IPYS,
*IPDEK, IPL3, IPL4, ICTE, ICTV, IDIM, IPROD
COMMON/NUDE/OUT(15), DELT, ICALL, INST, IPC, NE, NLDE, NLAE, NLAV, IBER,
*NLXAV, ICODE, ITURN, ICT, ISK, IDEC
DIMENSION IVEC(I), FPVEC(I)

GO TO (1,2,3), ICALL
1 CONTINUE

C INTEROGATION PASS (SUPERVISED BY SRNT)

C

NLXAV=0
IDER=0
NLAV=3
NLAE=1
NLDE=0
RETURN

3 CONTINUE
COMPUTATION PASS

FPVEC(IPTHL+1)=FPVEC(IPY+1)+FPVEC(IPY+2)-FPVEC(IPY+3)

RETURN

CONTINUE

COUNTING - MATCHING PASS (SUPERVISED BY SETUPS)
FOR Y(1), Y(2), Y(3)

DO 7 I=1,3
CALL MKLST(I, I, 1, 1, 1, IVEC, FPVEC)
CONTINUE
RETURN
END

SUBROUTINE MXR(IVEC, FPVEC)

Y(1)+Y(2)-Y(3)=0
Y(1) = 1ST STREAM FLOW RATE (INLET)
Y(2) = 2ND STREAM FLOW RATE (INLET)
Y(3) = 3RD STREAM FLOW RATE (OUTLET)

COMMON/BASE/IBPA, IBIP
COMMON/POINT/IPKP, IPPA, IP, IPSV, IPLV, IPE, IPNEC, IPL1, IPL2,
*IPNL, IPST, IPPO, KCO, IPPZ, IPMT, IPOS, IPX, IPF, IPFP, IPG, IPKCO,
*IPY, IPTHL, IPPY, IP1, IP2, IPAE, IPAV, IPXID, IPYID, IPES, IPYS,
*IPDER, IPL3, IPL4, ICTE, ICTV, IDIM, IPROD
COMMON/NODE/OUT(15), DELT, ICALL, INST, IPC, NE, NLDE, NLAE, NLAV, IDER,
*NLXAV, ICODE, ITURN, ICT, ISK, IDEC
DIMENSION IVEC(1), FPVEC(1)
GO TO (1,2,3), ICALL

C 1 CONTINUE

C INTERROGATION PASS

C IDER=1
NLAV=3
NLAQ=1
NLDE=0
NLXAV=0
RETURN

C 3 CONTINUE

C COMPUTATION PASS FETCH PARAMETERS, IF ANY, AND CALC THETL

C FPVEC(IPTHL+1)=FPVEC(IPY+1)+FPVEC(IPY+2)-FPVEC(IPY+3)

C IF (ICODE.NE.1) GG TO 6

C FPVEC(IPPY+1)=1.
FPVEC(IPPY+2)=1.
FPVEC(IPPY+3)=-1.

6 CONTINUE
RETURN

C 2 CONTINUE

C COUNTING - MATCHING PASS
FOR Y(1),Y(2),Y(3)
DO 7 I=1,3
CALL MKLST(I, I, 1, IVEC, FPVEC)
7 CONTINUE
RETURN
END

************************************************************************************
SUBROUTINE NDI(IVEC, FP)
C
C EXPLICIT VERSION OF OLD LINEAR SFTK
C DX(I)/DT=Y(1)-X(I)/R
C Y(1) - TANK HEIGHT
C Y(1) - INLET FLOW RATE (STRM NO. 1)
C Y(2) - OUTLET FLOW RATE (STRM NO. 2)
C
C R STORED IN POS 1 OF FPAR SUBSECTOR
C IVAB OF X(1) STORED IN POS 2 THRU 4 OF FPAR
C
C COMMON/BASE/IBPA, IDIP
C COMMON/POINT/IPKP, IPPAR, IP1P, IPSV, IPLV, IPLE, IPNEC, IPL1, IPL2,
C *IPNDL, IPST, IPP0, KCU, IPP2, IPTM, IPDZ, IPX, IPF, IPFP, IPC, IPKCO,
C *IPY, IPYH, IPPY, IPL, IP2, IPAE, IPAV, IPXID, IPYID, IPES, IPYS,
C *IPDER, IPL3, IPL4, ICTE, ICTV, IDIM, IPROD
C COMMON/NOVE/DT(15), DELT, ICALL, INST, IPC, NE, NLDN, NLAE, NLAV, IDER,
C *NLXAV, ICODE, ITURN, ICT, ISK, IDSEC
C DIMENSION IVEC(I), FP(I)
C
C GO TO (1, 2, 3), ICALL
C
C 1 CONTINUE
C
C INTEROGATION PASS
C
C IDER=0
CALL COMPLEX(1,2,1,1,0,IVEC,FP)
RETURN

C CONTINUE

C COUNTING - MATCHING PASS

DO 7 I=1,2
CALL MKLS(I,I,1,1,1,IVEC,FP)
7 CONTINUE
RETURN

C CONTINUE

C COMPUTATION PASS

R=FP(IBPA+1)
IF(INST.EQ.2)GO TO 5
FP(IPF+1)=FP(IPY+1)-FP(IPX+1)/R
RETURN

C CONTINUE

FP(IPY+2)=FP(IPX+1)/R
RETURN
SUBROUTINE ND2(IV,FP)

C EXPLICIT VERSION OF OLD LINEAR MXR

C Y(1)=Y(2)+Y(3)

C

COMMON/POINT/IPKP,IPPAR,IPIP,IPSV,IPLV,IPLE,IPNEC,IPL1,IPL2,
*IPNL0,IPST,IPPU,KCO,IPPZ,IPTM,IPDZ,IPX,IPF,IPFP,IPG,IPKCO,
*IPY,IPTHL,IPPY,IP1,IP2,IPAE,IPAV,IPXID,IPYID,IPES,IPYS,
*IPDER,IPL3,IPL4,ICTE,ICTV,IOIM,IPROD

COMMON/NODE/OUT(15),DELT,ICALL,INST,IPC,NE,NLDE,NLAE,NLAV,IDER,
*NLXAV,ICODE,ITURN,ICT,ISK,IDECE

DIMENSION IV(1),FP(1)

C

GO TO (1,2,3),ICALL

C 1 CONTINUE

C

NLXAV=1
IDER=0
NLAV=3
NLAE=0
NLDE=0
RETURN

C 3 CONTINUE

C

FP(IPY+3)=FP(IPY+1)+FP(IPY+2)
RETURN

C 2 CONTINUE

END
C
DO 7 I=1,3
CALL MKLST(I,I,1,1,1,IV,FP)
7 CONTINUE
RETURN
END

******************************************************************************
SUBROUTINE ND2(IV,FP)
C SOURCE NODE - GENERATES VALUE DURING EXPLICIT PASS AFTER POST PRED AND
PASS, HENCE LIST4 MAY HAVE TO MODIFIED SO THAT THIS
NODE IS SERVICED LAST
C NO INPUT STREAM
C OUTPUT STREAM (NO. 1) CARRIES SIGNAL IN POSITION IPSIG OF
APPROPRIATE STRMV SUBSECTOR - Y(1) - CLASS B EXPLICIT LAV
C FPAR SUBSECTOR - REF, AMPL, PER, PHASE
C IPAR SUBSECTOR - IPSIG
C
DIMENSION IV(1),FP(1)
COMMON/POINT/NG,IPPAR,IPIP,IPSV,IPLV,IPLE,IPNEC,IPLI,IPL2,
*IPNL0,IPST,IPPU,KCO,IPPP,IPTM,IPDZ,IPX,IPF,IPFP,IPG,IPKCO,
*IPY,IPTHL,IPPY,IP1,IPTMP,IPAE,IPAV,IPXID,IPYID,IPES,IPYS,
*IPDER,IPL3,IPL4,ICTE,ICTV,IDIM,IPROD
COMMON/CSAVE/XLB(6),XUB(6),T,JS(6),IPOS(6),ISIN(6),ICT2,NT
COMMON/NOUE/OUT(15),DELT,ICALL,INST,IPC,NE,NLDE,NLAE,NLAV,IDER,
*NLXAV,ICODE,ITURN,ICT3,ISK,IDEC
COMMON/JASE/IR,II
COMMON/LISTS/LISTS(30)
ND2 1
ND2 2
ND2 3
ND2 4
ND2 5
ND2 6
ND2 7
ND2 8
ND2 9
ND2 10
ND2 11
ND2 12
ND2 13
ND2 14
ND2 15
ND2 16
ND2 17
ND2 18
ND2 19
ND2 20
ND2 21
ND2 22
ND2 23
ND2 24
ND2 25
CGO TO (1,2,3),ICALL
C
1 CONTINUE
NLAE=0
NLXAV=1
NLDE=0
NLAV=1
IDER=0
RETURN
C
2 CONTINUE
IPSIG=IV(I+1)
CALL MKLST(I,1,1,IPSIG,1,IV,FP)
RETURN
C
3 CONTINUE
IF(T.EQ.0.)RETURN
IF(INST.NE.4.OR.IPC.EQ.1)RETURN
IF(LIST5(4).NE.1)RETURN
C
WE ARE ON POST-PRED EXPLICIT PASS
C
REF=FP(IR+1)
AMPL=FP(IR+2)
PER=FP(IR+3)
PHASE=FP(IR+4)
C
ARG = 6.28318531*T/PER+PHASE
FP(IPY+1)=REF*AMPL*SIN(ARG)
RETURN
END

*****************************************************************************************************************************************

SUBROUTINE ND3(IV,FP)
C
C EXPLICIT VERSION OF OLD LINEAR SPTR
C
C Y(2)=Y(1)-Y(3)
C Y(3)=FRAC*Y(1)
C
STREAM NO. 1 SPLITS INTO STREAMS 2 AND 3
C
COMMON/BASE/IR,II
COMMON/POINT/IPKP,IPPAR,IPIP,IPSV,IPLV,IPLE,IPNEC,IPL1,IPL2,
*IPNLD,IPST,IPPU,KCO,IPZ,IPTM,IPDZ,IPX,IPF,IPFP,IPG,IPKCO,
*IPY,IPTHL,IPPY,IP1,IP2,IPAE,IPAV,IPXID,IPYID,IPES,IPYS,
*IPDEK,IPL3,IPL4,ICTE,ICTV,IDIM,IPROD
COMMON/NUDE/DUT(15),DELT,ICALL,INST,IPC,NE,NLDE,NLAE,NLAV,IDER,
*NLXAV,ICODE,ITURN,ICT,ISK,IDEC
DIMENSION IV(I),FP(I)

GO TO (1,2,3),ICALL
1 CONTINUE
C
NLXAV=2
IDER=0
NLAV=3
NLAE=0
NLDE=0
RETURN
C
2 CONTINUE
DO 7 I=1,3
7 CONTINUE

END
CALL MKLST(I,I,1,1,1,IV,FP)

CONTINUE

RETURN

CONTINUE

FRAC=FP(IR+1)

FP(IPY+2)=FP(IPY)+1-FP(IPY+3)

FP(IPY+3)=FRAC*FP(IPY+1)

RETURN

END

******************************************************************************
SUBROUTINE PRES(IVEC,FP)

EVAPORATOR

STREAM NO. 1 - INLET LIQUID - IVBACS OF VEL,TEMP,PRESS,CONC

ALL ARE LAV - Y(1) TO Y(4)

STREAM NO. 2 - INLET TO STEAM CHEST - IVBACS OF MASS FL RT,TEMP,PRESS

ALL ARE LAV - Y(5) TO Y(7)

Y(6) OK TO IS CALCO DIRECTLY (STRMV CODE=-1)

STREAM NO. 3 - OUTLET LIQUID - IVBACS OF VEL,TEMP,PRESS,CONC

1ST THREE ARE LAV - Y(8) TO Y(10)

4TH IS EXTERNAL LV X(1)

STREAM NO. 4 - OUTLET VAPOR - IVBACS OF MASS FL RT,TEMP,PRESS

1ST TWO ARE LAV - Y(11) AND Y(12)

Y(12) OK TO IS CALCO DIRECTLY (STRMV CODE=-1)

3RD IS Y(14) IF HELD CONSTANT (ICP=-1, STRMV CODE=-1)

IS LV X(3) IF LEFT TO VARY (ICP=1, STRMV CODE=-1)

STREAM NO. 5 - CONTROLSENDING TANK LEVEL - IVBAC OF X(2)

IPAR SUBSECTOR CONTAINS STREAM NO. 4 PRESSURE'S CODE (ICP).
FPAR SUBSECTOR - POS 1 THRU 6 CONTAIN VALUES OF PARAMETERS PRE 22
  TANK AREA, TANK HEIGHT, U, AHTR, DO, CL PRE 23

POS 7 THRU 10 CONTAIN IVBAC OF BOIL-UP RATE, Y(13) PRE 25

DIMENSION IVEC(1), FP(1) PRE 27
COMMON/BASE/IHPA, IBIP PRE 28
COMMON/POINT/IPKP, IPPAR, IPIP, IPSV, IPLV, IPLE, IPNEC, IPL1, IPL2, PRE 29
  *IPNL0, IPST, IPPD, KCU, IPPZ, ITPX, IPX, IPF, IPFP, IPG, IPKCO, PRE 30
  *IPY, IPTHL, IPPY, IP1, IP2, IPAE, IPAV, IPXID, IPYID, IPES, IPYS, PRE 31
  *IPDRI, IPL3, IPL4, ICTT, ICTV, IDIM, IPROD PRE 32
COMMON/NODE/OUT(15), DELT, ICALL, INST, IPC, NE, NLD, NLA, NLAV, IDER, PRE 33
  *NLXAV, ICODE, ITURN, ICT, ISK, IDEC PRE 34
DATA XLAMMDA, RHOL, HTCAP, GSUBC, RGAS, XMOLWT, C1, C2, CW1, CW2, A1, A2/ PRE 35
  *980, .62, 4, 1, .32, 2, 10, 731, 18, .1, 5502826, 223.35509, 97596295, PRE 36
  *25473122, 1, 7797979, -.24691172/ PRE 37

GO TO (1, 2, 3), ICALL PRE 39
CONTINUE PRE 40
INTEROGATION PASS PRE 42
IDER=0 PRE 44
NLXAV=2 PRE 46
FETCH STREAM NO. 4 PRESSURE'S CODE (NOT SAME AS CODE IN STRMV SECTOR) PRE 48
IF ICP.EQ.1 PRESS IS TO BE LDV, IF ICP.EQ.-1 PRESS IS TO BE LAV PRE 49
ICP=IVEC(IBIP+1) PRE 51
NLAV=14 PRE 53
NLOE = 2
NLAE = 4
IF (IICP.EQ.-1) GO TO 4
NLAV = 13
NLOE = 3
NLAE = 3
CONTINUE

C
C MATCH LDV (ALL EXTERNAL) WITH GDV
C
CALL COMPX(0, 13, 1, 1, 3, IVEC, FP)
CALL COMPX(0, 1, 2, 1, 5, IVEC, FP)
IF (IICP.EQ.-1) RETURN
CALL COMPX(0, 9, 3, 1, 4, IVEC, FP)
RETURN

2 CONTINUE

C COUNTING-MATCHING PASS

C FOR STREAM NO. 1 (4 EXTERNAL LAVS - 1 THRU 4)
C
CALL MKLSTU(1, 1, 1, 1, 4, IVEC, FP)

C FOR STREAM NO. 2 (3 EXTERNAL LAVS - 5 THRU 7)
C
CALL MKLSTU(5, 2, 1, 1, 3, IVEC, FP)

C FOR STREAM NO. 3 (3 EXTERNAL LAVS - 8 THRU 10)
CALL MKLST(8,3,1,1,3,IVEC,FP)

FOR STREAM NO. 4 (2 EXT LAVS AND PERHAPS A THIRD)
CALL MKLST(11,4,1,1,2,IVEC,FP)
FOR INTERNAL LAV
CALL MKLST(13,0,0,7,1,IVEC,FP)
IF(IVEC(IBIP+1).EQ.1) GO TO 5
OUTLET VAPOR STREAM PRESSURE HELD CONSTANT
CALL MKLST(14,4,1,9,1,IVEC,FP)
CONTINUE
RETURN

CONTINUE
IF(INST.EQ.4.AND.IPC.GE.0) RETURN
COMPUTATION PASS
FETCH PARAMETERS FROM FPAR SUBSECTOR
AREA=FP(IBPA+1)
Z=FP(IBPA+2)
UCOEF=FP(IBPA+3)
AHTR=FP(IBPA+4)
DO=FP(IBPA+5)
CL = FP(IBPA + 6)

ICP = IVEC(IBIP + 1)
DJM = 3.145926535/576,
AR0 = DUM*D0*2*RHOL
Y1 = FP(IPY + 1)
Y2 = FP(IPY + 2)
Y3 = FP(IPY + 3)
Y4 = FP(IPY + 4)
Y7 = FP(IPY + 7)
TO = VAPR(Y7)
Y8 = FP(IPY + 8)
Y9 = FP(IPY + 9)
Y11 = FP(IPY + 11)
Y13 = FP(IPY + 13)
X1 = FP(IPX + 1)
X2 = FP(IPX + 2)
X3 = FP(IPX + 3)
Y2MF = AR0*Y1
Y5MF = AR0*Y8
VAR = FP(IPY + 14)
IF (ICP.EQ.1) VAR = X3

GO TO (6, 8, 7, 8), INST

CONTINUE

CALC RHS OF LDES

F(2) - OVERALL MASS BALANCE

F1 = (Y2MF - Y13 - Y5MF)/RHOL
DUM2=RHOL*AREA*X2

FP(IPTHL+2)=F1-6.0*FP*CL*Y8**2/DENOM
DENOM=00*5UBEC
CALL FRIC(FP,FP,F1,FO)
FP=F1-IP+101*DHMI+X2
DHMI=144.0/RHOL

Calc Lags
CONTINUE
RETURN

FP(IPF+3)=(F2+F3+FP)/(AREA*(Z-X2))
F2=FP*V2*HTCAP*(V2-V9)+(O-V13*XLAMDA)
F4=(Z-X2)*X3
F3=XP+F1
F2=RAP5YSIS*(V13-V111)*XMDLT
AO=COEF4*ATR*(10-V9)
IF(IPC.EQ.11)RETURN

FP(IPF+1)=F2/(RHOL*AREA*X2)
F2=V2*FP*(Y4-X1+X13)

FL1+FL2=FL1/AREA

FP=IPF+2=FL1/AREA
<table>
<thead>
<tr>
<th>Line</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>198</td>
<td><code>IF (ICP_EQ -1.0P (1P)THL+4) = 3</code></td>
</tr>
<tr>
<td>199</td>
<td><code>IF (1P)THL+3) = T2 + DUM8*ST/0.5</code></td>
</tr>
<tr>
<td>200</td>
<td><code>T2 = 1.0P (1P)THL+3</code></td>
</tr>
<tr>
<td>201</td>
<td><code>DUM8 = 1.0P (1P)THL+3</code></td>
</tr>
<tr>
<td>202</td>
<td><code>T2 = 1.0P (1P)THL+3</code></td>
</tr>
<tr>
<td>203</td>
<td><code>DUM8 = 1.0P (1P)THL+3</code></td>
</tr>
<tr>
<td>204</td>
<td><code>T2 = 1.0P (1P)THL+3</code></td>
</tr>
<tr>
<td>205</td>
<td><code>DUM8 = 1.0P (1P)THL+3</code></td>
</tr>
<tr>
<td>206</td>
<td><code>T2 = 1.0P (1P)THL+3</code></td>
</tr>
<tr>
<td>207</td>
<td><code>DUM8 = 1.0P (1P)THL+3</code></td>
</tr>
<tr>
<td>208</td>
<td><code>T2 = 1.0P (1P)THL+3</code></td>
</tr>
<tr>
<td>209</td>
<td><code>DUM8 = 1.0P (1P)THL+3</code></td>
</tr>
<tr>
<td>210</td>
<td><code>T2 = 1.0P (1P)THL+3</code></td>
</tr>
<tr>
<td>211</td>
<td><code>DUM8 = 1.0P (1P)THL+3</code></td>
</tr>
<tr>
<td>212</td>
<td><code>T2 = 1.0P (1P)THL+3</code></td>
</tr>
<tr>
<td>213</td>
<td><code>DUM8 = 1.0P (1P)THL+3</code></td>
</tr>
<tr>
<td>214</td>
<td><code>T2 = 1.0P (1P)THL+3</code></td>
</tr>
<tr>
<td>215</td>
<td><code>DUM8 = 1.0P (1P)THL+3</code></td>
</tr>
<tr>
<td>216</td>
<td><code>T2 = 1.0P (1P)THL+3</code></td>
</tr>
<tr>
<td>217</td>
<td><code>DUM8 = 1.0P (1P)THL+3</code></td>
</tr>
<tr>
<td>218</td>
<td><code>T2 = 1.0P (1P)THL+3</code></td>
</tr>
<tr>
<td>219</td>
<td><code>DUM8 = 1.0P (1P)THL+3</code></td>
</tr>
<tr>
<td>220</td>
<td><code>T2 = 1.0P (1P)THL+3</code></td>
</tr>
<tr>
<td>221</td>
<td><code>DUM8 = 1.0P (1P)THL+3</code></td>
</tr>
<tr>
<td>222</td>
<td><code>T2 = 1.0P (1P)THL+3</code></td>
</tr>
<tr>
<td>223</td>
<td><code>DUM8 = 1.0P (1P)THL+3</code></td>
</tr>
<tr>
<td>224</td>
<td><code>T2 = 1.0P (1P)THL+3</code></td>
</tr>
<tr>
<td>225</td>
<td><code>DUM8 = 1.0P (1P)THL+3</code></td>
</tr>
<tr>
<td>226</td>
<td><code>T2 = 1.0P (1P)THL+3</code></td>
</tr>
<tr>
<td>227</td>
<td><code>DUM8 = 1.0P (1P)THL+3</code></td>
</tr>
<tr>
<td>228</td>
<td><code>T2 = 1.0P (1P)THL+3</code></td>
</tr>
<tr>
<td>229</td>
<td><code>DUM8 = 1.0P (1P)THL+3</code></td>
</tr>
<tr>
<td>230</td>
<td><code>T2 = 1.0P (1P)THL+3</code></td>
</tr>
<tr>
<td>231</td>
<td><code>DUM8 = 1.0P (1P)THL+3</code></td>
</tr>
<tr>
<td>232</td>
<td><code>T2 = 1.0P (1P)THL+3</code></td>
</tr>
<tr>
<td>233</td>
<td><code>DUM8 = 1.0P (1P)THL+3</code></td>
</tr>
<tr>
<td>234</td>
<td><code>T2 = 1.0P (1P)THL+3</code></td>
</tr>
<tr>
<td>235</td>
<td><code>DUM8 = 1.0P (1P)THL+3</code></td>
</tr>
<tr>
<td>236</td>
<td><code>T2 = 1.0P (1P)THL+3</code></td>
</tr>
<tr>
<td>237</td>
<td><code>DUM8 = 1.0P (1P)THL+3</code></td>
</tr>
<tr>
<td>238</td>
<td><code>T2 = 1.0P (1P)THL+3</code></td>
</tr>
<tr>
<td>239</td>
<td><code>DUM8 = 1.0P (1P)THL+3</code></td>
</tr>
<tr>
<td>240</td>
<td><code>T2 = 1.0P (1P)THL+3</code></td>
</tr>
<tr>
<td>241</td>
<td><code>DUM8 = 1.0P (1P)THL+3</code></td>
</tr>
<tr>
<td>242</td>
<td><code>T2 = 1.0P (1P)THL+3</code></td>
</tr>
<tr>
<td>243</td>
<td><code>DUM8 = 1.0P (1P)THL+3</code></td>
</tr>
<tr>
<td>244</td>
<td><code>T2 = 1.0P (1P)THL+3</code></td>
</tr>
<tr>
<td>245</td>
<td><code>DUM8 = 1.0P (1P)THL+3</code></td>
</tr>
<tr>
<td>246</td>
<td><code>T2 = 1.0P (1P)THL+3</code></td>
</tr>
<tr>
<td>247</td>
<td><code>DUM8 = 1.0P (1P)THL+3</code></td>
</tr>
<tr>
<td>248</td>
<td><code>T2 = 1.0P (1P)THL+3</code></td>
</tr>
<tr>
<td>249</td>
<td><code>DUM8 = 1.0P (1P)THL+3</code></td>
</tr>
<tr>
<td>250</td>
<td><code>T2 = 1.0P (1P)THL+3</code></td>
</tr>
<tr>
<td>251</td>
<td><code>DUM8 = 1.0P (1P)THL+3</code></td>
</tr>
<tr>
<td>252</td>
<td><code>T2 = 1.0P (1P)THL+3</code></td>
</tr>
<tr>
<td>253</td>
<td><code>DUM8 = 1.0P (1P)THL+3</code></td>
</tr>
<tr>
<td>254</td>
<td><code>T2 = 1.0P (1P)THL+3</code></td>
</tr>
<tr>
<td>255</td>
<td><code>DUM8 = 1.0P (1P)THL+3</code></td>
</tr>
<tr>
<td>256</td>
<td><code>T2 = 1.0P (1P)THL+3</code></td>
</tr>
<tr>
<td>257</td>
<td><code>DUM8 = 1.0P (1P)THL+3</code></td>
</tr>
<tr>
<td>258</td>
<td><code>T2 = 1.0P (1P)THL+3</code></td>
</tr>
<tr>
<td>259</td>
<td><code>DUM8 = 1.0P (1P)THL+3</code></td>
</tr>
<tr>
<td>260</td>
<td><code>T2 = 1.0P (1P)THL+3</code></td>
</tr>
<tr>
<td>261</td>
<td><code>DUM8 = 1.0P (1P)THL+3</code></td>
</tr>
<tr>
<td>262</td>
<td><code>T2 = 1.0P (1P)THL+3</code></td>
</tr>
<tr>
<td>263</td>
<td><code>DUM8 = 1.0P (1P)THL+3</code></td>
</tr>
<tr>
<td>264</td>
<td><code>T2 = 1.0P (1P)THL+3</code></td>
</tr>
<tr>
<td>265</td>
<td><code>DUM8 = 1.0P (1P)THL+3</code></td>
</tr>
<tr>
<td>266</td>
<td><code>T2 = 1.0P (1P)THL+3</code></td>
</tr>
<tr>
<td>267</td>
<td><code>DUM8 = 1.0P (1P)THL+3</code></td>
</tr>
<tr>
<td>268</td>
<td><code>T2 = 1.0P (1P)THL+3</code></td>
</tr>
<tr>
<td>269</td>
<td><code>DUM8 = 1.0P (1P)THL+3</code></td>
</tr>
<tr>
<td>270</td>
<td><code>T2 = 1.0P (1P)THL+3</code></td>
</tr>
<tr>
<td>271</td>
<td><code>DUM8 = 1.0P (1P)THL+3</code></td>
</tr>
<tr>
<td>272</td>
<td><code>T2 = 1.0P (1P)THL+3</code></td>
</tr>
<tr>
<td>273</td>
<td><code>DUM8 = 1.0P (1P)THL+3</code></td>
</tr>
<tr>
<td>274</td>
<td><code>T2 = 1.0P (1P)THL+3</code></td>
</tr>
<tr>
<td>275</td>
<td><code>DUM8 = 1.0P (1P)THL+3</code></td>
</tr>
<tr>
<td>276</td>
<td><code>T2 = 1.0P (1P)THL+3</code></td>
</tr>
<tr>
<td>277</td>
<td><code>DUM8 = 1.0P (1P)THL+3</code></td>
</tr>
<tr>
<td>278</td>
<td><code>T2 = 1.0P (1P)THL+3</code></td>
</tr>
<tr>
<td>279</td>
<td><code>DUM8 = 1.0P (1P)THL+3</code></td>
</tr>
<tr>
<td>280</td>
<td><code>T2 = 1.0P (1P)THL+3</code></td>
</tr>
<tr>
<td>281</td>
<td><code>DUM8 = 1.0P (1P)THL+3</code></td>
</tr>
<tr>
<td>282</td>
<td><code>T2 = 1.0P (1P)THL+3</code></td>
</tr>
</tbody>
</table>
SUBROUTINE PUVAIV, FP

C PUMP - VALVE

C STREAM NO. 1 - INLET LIQUID - STRMV CONTAINS IVBAC OF VEL, TEMP, PRESS, AND CONC. - Y(1) THRU Y(4)

C STREAM NO. 2 - CONTROL SIGNAL - STRMV CONTAINS IVBAC OF CONTROL PRESS Y(9) WITH LAG (TAUL) TO X(1)

C STREAM NO. 3 - OUTLET LIQUID - STRMV CONTAINS IVBAC OF VEL, TEMP, PRESS, AND CONC. - Y(5) THRU Y(8)

C FPAR PARAMETERS - CV1, CV2, XMAX, XMIN, HMAX, VMAX, AREA, TAUL (POS 1 THRU 8)

C POS 9 THRU 11 CONTAIN IVAB OF X(1) - LAGGED CONTROL PRESSURE

C TEMP AND PRESS ARE TREATED AS EXPLICIT LAV - CODE = (NODE NO. + 1)

C IPAR PARAMETER - IC (=0 FOR AIR-TO-OPEN)

C (=1 FOR AIR-TO-CLOSE)

C COMMON/POINT/IPKP, IPPAR, IPIP, IPSV, IPLV, IPLE, IPNEC, IPL1, IPL2, PUV 1
C *IPNLDD, IPST, IPPU, KCU, IPPZ, IPTM, IPD2, IPX, IPF, IPFP, IPG, IPKCO, PUV 2
C *IPY, IPTHL, IPPY, IPO, IPZ, IPAE, IPAV, IPXID, IPYID, IPS, IPYS, PUV 3
C *IPDEK, IPL3, IPL4, ICTE, ICTV, IDMIM, IPROD, PUV 4
C COMMON/NODE/OUT(15), DELL, ICALL, INST, IPC, NE, NLDE, NLAE, NLAV, IDER, PUV 5
C *NLXAV, ICODE, ITURN, ICT, ISK, IDEC PUV 6
C COMMON/ASE/IR, II PUV 7
C DIMENSION IV(1), FP(1) PUV 8
C DATA RHGL, GAM/62.4, 435.73/ PUV 9
C
GO TO (1,2,3), ICALL
C
1 CONTINUE
C
INTERROGATION PASS
C
IDER=0
NLAE=2
NLDE=1
NLAV=9
NLXAV=2
C
FOR LAGGED CONTROL PRESSURE (INT LDV X(1) )
C
CALL COMPC(1,9,1,1,0,IV,FP)
C
RETURN
C
2 CONTINUE
C
COUNTING - MATCHING PASS
C
FOR EXT LAV IN STREAM NO. 1
C
CALL MKLST(1,1,1,4,IV,FP)
C
FOR EXT LAV IN STREAM NO. 3
C
CALL MKLST(5,3,1,1,4,IV,FP)
C
FOR EXT LAV IN STREAM NO. 2
CALL MKLST(9,2,1,1,IV,FP)

RETURN

3 CONTINUE

COMPUTATION PASS
FETCH PARAMETERS

CV1=FP(IR+1)
CV2=FP(IR+2)
XMAX=FP(IR+3)
XMIN=FP(IR+4)
HM=FP(IR+5)
VM=FP(IR+6)
AREA=FP(IR+7)
TAUL=FP(IR+8)
IC=IV(I+1)
GO TO (5,7,6,7), INST

5 CONTINUE

IF(TAUL.EQ.0.)GO TO 8

FP(IPF+1)=(FP(IPY+9)-FP(IPX+1))/TAUL
RETURN

8 CONTINUE
FP(IPF+1)=0.
FP(IPX+1)=FP(IPY+9)
RETURN
CONTINUE
X1 = FP(IPX + 1)
IF (IC.EQ.0) GO TO 9
AIR-TO-CLOSE
X = 100.0 * (XMAX - X1) / (XMAX - XMIN)
GO TO 10

CONTINUE
AIR-TO-OPEN
X = 100.0 * (X1 - XMIN) / (XMAX - XMIN)

CONTINUE

T1 = PRESSURE HEAD THAT PUMP MUST OVERCOME
T3 = HEAD ADDED BY PUMP (HM)
FOR NONZERO FLOW, T3 MUST BE .GT. T1
Y1 = FP(IPY + 1)
CV = CV1 * X + CV2 * X**2
CVS = CV**2
T1 = CVS * (FP(IPY + 3) - FP(IPY + 7))
T2 = (GAM * AREA * Y1)**2
T3 = CVS * RHCL * HM * (1 - Y1/VM) / 144.
FP(IPTHL + 1) = T1 - T2 + T3
FP(IPTHL + 2) = Y1 - FP(IPY + 5)
RETURN

CONTINUE

FP(IPY+6)=FP(IPY+2)
FP(IPY+8)=FP(IPY+4)

RETURN

END

******************************************************************************
SUBROUTINE SFTK(IVEC,FPVEC)

DX(1)/DT = Y(1)-X(1)/R , X(1) = TANK VOLUME
Y(1) = INLET FLOW RATE
Y(2) = OUTLET FLOW RATE

DIMENSION IVEC(1) ,FPVEC(1)
COMMON/HASE/IBPA,IBIP
COMMON/POINT/IPKP,IPPAR,IPIP,IPSV,IPLV,IPLE,IPNEC,IPL1,IPL2,
*IPNL0,IPST,IPPD,KCO,IPPZ,IPTM,IPU2,IPX,IPF,IPFP,IPG,IPKCO,
*IPY,IPTHL,IPPY,IP1,IP2,IPAE,IPAV,IPXID,IPYID,IPES,IPYS,
*IPDER,IPL3,IPL4,ICTE,ICTV,IDIM,IPROD
COMMON/NODE/DUT(15),DELT,ICALL,INST,IPC,NE,NLDE,NLAE,NLAV,IDER,
*NLXAV,ICODE,ITURN,ICT,ISK,IDEQ

GO TO (1,2,3),ICALL

CONTINUE

INTEROGATION PASS

IDER=0
NLAV=2
NLAE=1
NLDE=1
NLXAV=0
CALL COMPX(1,2,1,1,0,IVEC,FPVEC)
RETURN
C 3 CONTINUE
C COMPUTATION PASS
C FETCH PARAMETERS
C R=FPVEC(IBPA+1)
C IF(INST.EQ.3)GO TO 5
C CALCULATE RHS OF ODE
C FPVEC(IPF+1)=FPVEC(IPY+1)-FPVEC(IPX+1)/R
RETURN
C 5 CONTINUE
C CALCULATE TETL AND PARTIALS
C FPVEC(IPTHL+1)=FPVEC(IPY+2)-FPVEC(IPX+1)/R
RETURN
C 2 CONTINUE
C COUNTING - MATCHING PASS
DO 7 I=1,2
CALL MKLST(I,I,1,1,1,IVEC,FPVEC)
7 CONTINUE
RETURN
END

******************************************************************************
SUBROUTINE SPTR(IVEC,FPVEC)
COMM/NBASE/IBPA,IaIP
COMM/POINT/IPKP,IPPAR,IPIP,IPSV,IPLV,IPLE,IPNEC,IPL1,IPL2,
*IPNLD,IPST,IPPO,KCO ,IPPZ,IPMT,IPDZ,IPX,IPF,IPFP,IPG ,IPKCO,
*IPY,IPTHL,IPPY,IP1,IP2,IPAE,IPAV,IPXID,IPYID,IPES,IPYS,
*IPDER,IPL3,IPL4,ICTE,ICTV,IDIM,IPROD
COMMON/NUDE/DUT(15),DELT,ICALL,INST,IPC,NE,NLDE,NLAE,NLAV,IDER,
*NLXAV,ICODE,ITURN,ICT,ISK,IDEC
DIMENSION IVECIU,FPVEC(1)
GO TO (1,2,3),ICALL
1 CONTINUE
NLXAV=0
IDER=0
NLAV=3
NLAE=2
NLDE=0
RETURN
3 CONTINUE
FRAC=FPVEC(IBPA+1)
FPVEC(IPTHL+1)=FPVEC(IPY+1)-FPVEC(IPY+2)-FPVEC(IPY+3)
FPVEC(IPTHL+2)=FPVEC(IPY+3)-FRAC*FPVEC(IPY+1)
RETURN
2 CONTINUE
DO 7 I=1,3
CALL MKLST(I,I,1,1,1,IVEC,FPVEC)
7 CONTINUE
SUBROUTINE TANKU(VEC, FPVEC)

LUMP COUNTER CURRENT SOLIDS DRYER

FOR THE K TH LUMP -

DTW(K) = F(1)  WALL ENERGY BAL

DM(K)/DT = F(2)  OVERALL MASS BAL

DX(K)/DT = F(3)  COMPONENT MASS BAL

DT(K)/DT = F(4)  SOLIDS ENTHALPY BALANCE

Y(K) = W(1)  AIR COMPONENT MASS BAL

ST(K) = W(2)  AIR ENTHALPY BAL

STRM 1 - INLET SOLIDS - IVBAC OF MOIST FRAC(M.F.), TEMP(T), FLOW RATE(FL, RT), Y(1) THRU Y(3)

STRM 2 - OUTLET SOLIDS - IVBAC OF M.F., T, FL, RT, X(3+4*(N-1)), X(4+4*(N-1)), Y(4)

STRM 3 - INLET AIR - IVBAC OF HUMIDITY, TEMP, FLOW RATE Y(6+2*N) THRU Y(8+2*N)

STRM 4 - OUTLET AIR - IVBAC OF HUMIDITY, TEMP, FLOW RATE Y(5) THRU Y(7)

STRM 5 - INLET STEAM - IVBAC OF STEAM TEMP - Y(9+2*N)

IPAR SUBSECTOR - N (NO. OF LUMPS)

FPAR SUBSECTOR - C14, C16, C17, C30, DL, DELTA (POS 1-6)

NEXT 3*(N-1)*4 POSITIONS - IVBAC OF WALL TEMP, LUMP MASS, SOLIDS MOIS, SOLIDS TEMP FOR EACH OF 1ST N-1 LUMPS

NEXT 6 POS - IVBAC OF JUST WALL TEMP AND LUMP MASS
(LAST LUMP'S SOLIDS MOIS AND SOLIDS TEMP ARE EXT LDVS, STRM 2)
NEXT 4*(N-1)*2 POS CONTAIN INVAC OF AIR HUMID, AIR TEMP
(INT LAV) OF LAST N-1 LUMPS
(AIR HUMID AND AIR TEMP LEAVING LUMP 1 ARE EXT LAV (STRM 4)
FOR NSR CALCNS

TIN,TOUT SOLIDS TEMPS
XIN, XOUT SOLIDS MOISTURE RATIOS
FRIN, FROUT SOLIDS FLOW RATES
TAIN, TOUT AIR TEMPS
YIN, YOUT AIR HUMIDITYYS
G AIR FLOW RATE

DIMENSION IVEC(1), FPVEC(1)
COMMON/INOUT/IN, IOUT, IPUN
COMMON/PCINT/IPKP, IPPAR, IP1P, IPSV, IPLV, IPE, IPN, IPL1, IPL2,
*IPNL, IPI, IPP0, KCO1, IPPZ, IPTM, IPDZ, IPX, IP1F, IPFP, IPG,
*IPKCO,
*IPP, IPIH, IPPY, IP1, IP2, IP1E, IPAV, IPXI, IPYID, IPYD, IPES, IPYS,
*IPDER, IPL3, IPL4, ICTE, ICTV, IOiM, IPROD
COMMON/NODE/OUT(15), DELL, ICALL, INST, IPC, NE, NLDE, NLA, NLAV, IDEL,
*NLXAV, ICODE, ITURN, ICT, ISK, IDEC
COMMON/BASE/IB, IIP

DATA C1, C2, C3, C4, C5 / 25, 672, 970, 1, 1.94122 /
DATA C10, C11, C12, C13 / 595, 5.0606, 19.635, 2.22 /
DATA C15 / 1 /
DATA C18, C19, C20, C21 / 0.02, 49.5, 2.5, 700. /
DATA C25, C28 / 45.7, 67847 /
DATA BETA2 / 0.00234 /
DATA RATIO / .6206896 /
GO TO (1,2,3),ICALL

1 CONTINUE

C INTERROGATION PASS

C
IDER=0
N=IVEC(IBIP+1)
NLAE=0
NLDE=4*N
NLAV=2*N+9
NLDVS=4*N-2
NLXAV=NLAV

C FOR INTERNAL LDV IN PAR

C
CALL COMPX(1,7,1,NLDVS,0,IVEC,FPVEC)

C FOR EXTERNAL LDV IN STRMV

C
IXIS=NLDVS+1
CALL COMPX(0,1,IXIS,2,2,IVEC,FPVEC)
RETURN

2 CONTINUE

C COUNTING MATCHING PASS

C FOR FEED SOLIDS STREAM (NO. 1)

C
CALL MKLST(1,1,1,1,3,IVEC,FPVEC)
FOR OUTLET SOLIDS STREAM (NO. 2)

CALL MKLST(4,2,1,9,1,IVEC,FPVEC)

FOR OUTLET AIR STREAM (NO. 4)

CALL MKLST(5,4,1,1,3,IVEC,FPVEC)

N=IVEC(IHIP+1)

FOR INTERNAL LAV

NVAR=2*N-2
ISTA=7+3*(4*N-2)
CALL MKLST(8,0,0,ISTA,NVAR,IVEC,FPVEC)

FOR INLET AIR STREAM (NO. 3)

LOCNO=2*N+6
CALL MKLST(LOCNO,3,1,1,3,IVEC,FPVEC)

FOR STEAM LINE (STREAM NO. 5)

STEAM TEMP IS Y(9+2*N)

LOCNO=LOCNO+3
CALL MKLST(LOCNO,5,1,1,1,IVEC,FPVEC)

MODIFY TRANSFER COEFFICIENTS

DL=FPVEC(18+5)
DUM=6.283185*2.5*DL
DO 103 I=1,4
IARG = IB + 1
FPVEC(IARG) = FPVEC(IARG) * DUM
103 CONTINUE
C
RETURN
C
3 CONTINUE
IF (INST.EQ.4. AND. IPC.GE.0) RETURN
C
COMPUTATION PASS
C
C14 = FPVEC(IB + 1)
C16 = FPVEC(IB + 2)
C17 = FPVEC(IB + 3)
C30 = FPVEC(IB + 4)
DL = FPVEC(IB + 5)
DELTA = FPVEC(IB + 6)
C
N = IVEC(IB + IP + 1)
IBF = IPF - 4
IBY = IPY + 5
IARG = IB + 2 * N + IPY
G = FPVEC(IARG)
TS = FPVEC(IARG + 1)
GAF = G * C12
K = 0
600 CONTINUE
KM1 = K
K = K + 1
C
IARG = IPX + 4 * KM1
Tw = FPVEC(IARG + 1)
WT=FPVEC(IARG+2)
XOUT=FPVEC(IARG+3)
TOUT=FPVEC(IARG+4)
IF(K.NE.1)GO TO 602

C
XIN=FPVEC(IPY+1)
TIN=FPVEC(IPY+2)
FRIN=FPVEC(IPY+3)
YOUT=FPVEC(IPY+5)
TAGOUT=FPVEC(IPY+6)

C
602 CONTINUE
FRQOUT=WT/DELTA
IBY=IBY+2
YIN=FPVEC(IBY+1)
TAIN=FPVEC(IBY+2)

C
HTCTA=C13+C11*(TAGOUT-630.1)*1.E-3
YS=-1.2755+.00234*TAOUT
RH=YOUT/(RATIO+YOUT)
RH=RH/YSS*(RATIO+YSS)
S1=1.34-.002*TAOUT
XINT=RH*S1
WK=C5+C28*(TOUT-560.1)*1.E-2
WK=WK*(XOUT-XINT)
WK=WK*WT
HTCTW=C16*1.E2+C30*(TOUT-560.1)

C
IF(INST.EQ.2 OR INST.EQ.4)GO TO 5
C
IBF=IBF+4

C
C  CALC F(3)  COMPONENT MASS BALANCE
C  DUM=FRIN*(XIN-XOUT)
FPVEC(IBF+3)=(DUM-WK)/WT
C  CALC F(4)  ENERGY BALANCE ON SOLIDS
C  DUM1=FRIN*(C15+XIN*C41)
DUM1=DUM1*(TIN-TOUT)
DUM1=DUM1-WK*C3
DUM1=DUM1-HTCTW*(TOUT-TW)
DUM1=DUM1-HTC(TAOUT-TOUT)*WT
DUM3=TOUT-C2
DUM2=C4-C25
DUM1=DUM1+WK*DUM3*DUM2
FPVEC(IBF+4)=DUM1/(WT*(C15+XOUT*C41))
C  CALC F(1)  WALL ENERGY BALANCE
C  DUM=C17*(TS-TW)
DUM=DUM*C14*(TAOUT-TW)
DUM=DUM*HTC(TW)*(TOUT-TW)
FPVEC(IBF+1)=DUM/(0.283*C20*DL*C18*C19)
C  CALC F(2)  OVERALL MASS BALANCE
FPVEC(IBF+2)=FRIN-FROUT
601 CONTINUE
IF(K.EQ.N)GO TO 603
C  XIN=XOUT
TIN = TOUT  
FRIN = FROUT  
YUUT = YIN  
TAOUT = TAIN  
GOTO 600  

C  
603 CONTINUE  
FPVEC(IPY+4) = FROUT  
FPVEC(IPY+7) = G  
RETURN  
C  
5 CONTINUE  
C  
T1P = C1 + YIN * C25  
T1 = T1P * GAF  
T2 = HCTA * WT  
T3 = WK * C25  
UP = T1 * TAIN + (T2 + T3) * TOUT  
UP = UP + C14 * Tw  
DWN = T1 + T2 + T3 + C14  
C  
C  
CALC NEW YOUT AND TAOUT VIA EXPLICIT FUNCTION  
C  
IA = IBY - 2  
IF (K. NE. 1) IA = IA + 1  
FPVEC(IA) = YIN + WK / GAF  
FPVEC(IA+1) = UP / DWN  
GOTO 601  
C  
CEND  

*******************************  
SUBROUTINE TMCN(IVEC, FPVEC)  
*******************************
FEED FORWARD COMPUTATION FOR SOLIDS DRYER SYSTEM

DIMENSION IVEC(1), FPVEC(1)
COMMON/NODE/DUT(15), DELT, ICALL, INST, IPC, NE, NLDE, NLAE, NLAV, IDER,
*NLXAV, ICODE, ITURN, ICT, ISK, IVEC
COMMON/POINT/IPKP, IPPAR, IPIP, IPSV, IPLV, IPLE, IPNEC, IPL1, IPL2,
*IPNLD, IPST, IPPU, KCU, IPPZ, IPTM, IPDZ, IPX, IPF, IPPF, IPG, IPKCO,
*IPY, IPTHL, IPPY, IP1, IP2, IPAE, IPAV, IPX1D, IPY1D, IPES, IPYS,
*IPDER, IPL3, IPL4, ICTE, ICTV, IDIM, IPROD
GO TO (1, 2, 3), ICALL

1 CONTINUE
IDER=1
NLAV=2
NLAE=1
NLDE=0
NLXAV=0
RETURN

2 CONTINUE
CALL MKLST(1, 1, 1, 9, 1, IVEC, FPVEC)
CALL MKLST(2, 2, 1, 1, IVEC, FPVEC)
RETURN

3 CONTINUE
FPVEC(IPTHL+1) = FPVEC(IPY+2) + 0.140054*(FPVEC(IPY+1)-7800.) + 700.
IF (ICODE .NE. 1) RETURN
FPVEC(IPPY+1) = 140054
FPVEC(IPPY+2) = -1.
RETURN
END

************************************************************************************************************
BIBLIOGRAPHY


18. IBM System/360 Continuous System Modeling Program, Form No. H 20-0367, IBM Corp., Data Processing Division, White Plains, N.Y.


20. IBM 1130 Continuous System Modeling Program, Form No. H 20-0282, IBM Corp., Data Processing Division, White Plains, N.Y.


24. LEANS, Chemical Engineering Department. Lehigh University, Bethlehem, Penn., 1971.


