PROPULSION SYSTEM FLOW STABILITY PROGRAM
(DYNAMIC)

PHASE I FINAL TECHNICAL REPORT
PART XVII—PROPULSION SYSTEM SIMULATION DIGITAL COMPUTER
PROGRAM FORMAT AND ROUTINES

E.H. Kaplan and H.W. Wong
LOS ANGELES DIVISION OF NORTH AMERICAN ROCKWELL CORPORATION

TECHNICAL REPORT AFAPL-TR-68-142, PART XVII

December 1968

*** Export controls have been removed ***

This document is subject to special export controls and each transmittal to
foreign governments or foreign nationals may be made only with prior approval
of the Air Force Aero Propulsion Laboratory (APTA), Air Force Systems Command,
Wright-Patterson Air Force Base, Ohio.

Air Force Aero Propulsion Laboratory
Air Force Systems Command
Wright-Patterson Air Force Base, Ohio

Confirmed Public via DTIC 1/14/2019
PROPULSION SYSTEM FLOW STABILITY PROGRAM (DYNAMIC)

PHASE I FINAL TECHNICAL REPORT

PART XVII. PROPULSION SYSTEM SIMULATION DIGITAL COMPUTER PROGRAM FORMAT AND Routines

E.H. Kaplan and H.W. Wong

*** Export controls have been removed ***

This document is subject to special export controls and each transmittal to foreign governments or foreign nationals may be made only with prior approval of the Air Force Aero Propulsion Laboratory (APTA), Air Force Systems Command, Wright-Patterson Air Force Base, Ohio.

Confirmed Public via DTIC 1/14/2019
FOREWORD

This report describes work accomplished in Phase I of the two-phase program, "Propulsion System Flow Stability Program (Dynamic)" conducted under USAF Contract F33615-67-C-1848. The work was accomplished in the period from 20 June 1967 to 30 September 1968 by the Los Angeles Division of North American Rockwell Corporation, the prime Contractor, and the Subcontractors, the Allison Division of General Motors Corporation (supported by Northern Research and Engineering Corporation), the Autonetics Division of North American Rockwell Corporation (supported by the Aeronautical Division of Honeywell, Incorporated), and the Pratt & Whitney Aircraft Division of United Aircraft Corporation.

The program was sponsored by the Air Force Aero Propulsion Laboratory, Wright-Patterson Air Force Base, Ohio. Mr. H. J. Gratz, APTA, Turbine Engine Division, was the Project Engineer.

This volume is Part XVII of twenty parts and was prepared by the Los Angeles Division of North American Rockwell Corporation.

Publication of this report does not constitute Air Force approval of the report's findings or conclusions. It is published only for the exchange and stimulation of ideas.

Ernest C. Simpson
Chief, Turbine Engine Division
ABSTRACT

The primary objective of Task 7 of the "Propulsion System Flow Stability Program" was to develop a simulation program to be used in Phase II for the evaluation of two control systems capable of sensing and accommodating a transient condition.

Since the work on this task was being performed by three companies, every effort was made to insure compatibility in terminology, units, and program documentation as well as to provide means of communicating the myriad details involved in making computer runs of the system. This documentation format is described in Section II of this volume.

An early element of this task was the selection of a simulation language for use in programming the simulation. The choice of IBM's DSL/90 and the factors involved in making that choice are discussed in Section III.

Simulation programs have a natural tendency to be rather voluminous and, when the system being simulated is as complex as a supersonic inlet, turbofan, and an integrated control system can be, computer storage space is rapidly filled. To alleviate this crowding, numerous logic blocks which were repetitive, such as compressor logic, were removed from the simulation logic deck and made into subroutines or functions. These subprograms are discussed in Section IV.

Once the simulation logic is written, the most difficult task of all begins. The job of initialization is usually not given proper emphasis until many hours of work have convinced all concerned that it is really the most important phase. Section V discusses this task and shows an example of an initialization routine.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>I INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>II DOCUMENTATION FORMAT</td>
<td>3</td>
</tr>
<tr>
<td>General Objective</td>
<td>3</td>
</tr>
<tr>
<td>Variable Names</td>
<td>3</td>
</tr>
<tr>
<td>Forms</td>
<td>6</td>
</tr>
<tr>
<td>Run Summary Sheet</td>
<td>6</td>
</tr>
<tr>
<td>Simulation Logic Base Sheet</td>
<td>10</td>
</tr>
<tr>
<td>Table Base Sheet</td>
<td>10</td>
</tr>
<tr>
<td>Output Base Sheet</td>
<td>10</td>
</tr>
<tr>
<td>Execution Control Base Sheet</td>
<td>10</td>
</tr>
<tr>
<td>Set-Up Base Sheet</td>
<td>10</td>
</tr>
<tr>
<td>Diagrams</td>
<td>16</td>
</tr>
<tr>
<td>Input Data Diagram</td>
<td>16</td>
</tr>
<tr>
<td>Input Tables Diagram</td>
<td>16</td>
</tr>
<tr>
<td>Simulation Logic Diagram</td>
<td>16</td>
</tr>
<tr>
<td>III SELECTION OF SIMULATION LANGUAGE</td>
<td>23</td>
</tr>
<tr>
<td>Comparison</td>
<td>23</td>
</tr>
<tr>
<td>Conclusion</td>
<td>23</td>
</tr>
<tr>
<td>IV SUPPORTING SUBPROGRAMS</td>
<td>29</td>
</tr>
<tr>
<td>General Approach</td>
<td>29</td>
</tr>
<tr>
<td>Descriptions</td>
<td>29</td>
</tr>
<tr>
<td>Subroutine SAACT</td>
<td>30</td>
</tr>
<tr>
<td>Function SADSPA</td>
<td>33</td>
</tr>
<tr>
<td>Subroutine SALIMT</td>
<td>33</td>
</tr>
<tr>
<td>Function SAMOIN</td>
<td>37</td>
</tr>
<tr>
<td>Function SAWCH</td>
<td>37</td>
</tr>
<tr>
<td>Function SAWFAT</td>
<td>37</td>
</tr>
<tr>
<td>Function SLFVFG</td>
<td>42</td>
</tr>
<tr>
<td>Function SLGAM</td>
<td>44</td>
</tr>
<tr>
<td>Function SLMVFG</td>
<td>44</td>
</tr>
<tr>
<td>Function SLTLU</td>
<td>44</td>
</tr>
<tr>
<td>Function SPBLOW</td>
<td>44</td>
</tr>
</tbody>
</table>
## Section

<table>
<thead>
<tr>
<th>Subroutine/Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subroutine SPCOMP</td>
<td>51</td>
</tr>
<tr>
<td>Function SPMEMF</td>
<td>51</td>
</tr>
<tr>
<td>Function SPTACL</td>
<td>51</td>
</tr>
<tr>
<td>Function SPTLU</td>
<td>63</td>
</tr>
<tr>
<td>Subroutine SPTURB</td>
<td>63</td>
</tr>
<tr>
<td>Function SARECT</td>
<td>75</td>
</tr>
<tr>
<td>Function SPTMCV</td>
<td>76</td>
</tr>
</tbody>
</table>

### Appendix

<table>
<thead>
<tr>
<th>Appendix</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>List of Supporting Subroutines</td>
<td>81</td>
</tr>
<tr>
<td>II</td>
<td>Initialization Program</td>
<td>105</td>
</tr>
</tbody>
</table>
LIST OF ILLUSTRATIONS

<table>
<thead>
<tr>
<th>Figure No.</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Propulsion System Numbering Scheme</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td>Sample Dictionary</td>
<td>7</td>
</tr>
<tr>
<td>3</td>
<td>Run Summary Sheet</td>
<td>8</td>
</tr>
<tr>
<td>4</td>
<td>Run Summary Sheet</td>
<td>9</td>
</tr>
<tr>
<td>5</td>
<td>Simulation Logic Base Sheet - Turbofan Inlet</td>
<td>11</td>
</tr>
<tr>
<td>6</td>
<td>Table Base Sheet - Turbofan Inlet</td>
<td>12</td>
</tr>
<tr>
<td>7</td>
<td>Output Base Sheet</td>
<td>13</td>
</tr>
<tr>
<td>8</td>
<td>Execution Control Base Sheet</td>
<td>14</td>
</tr>
<tr>
<td>9</td>
<td>Set-Up Base Sheet</td>
<td>15</td>
</tr>
<tr>
<td>10</td>
<td>Input Data List</td>
<td>17</td>
</tr>
<tr>
<td>11</td>
<td>Input Table List</td>
<td>18</td>
</tr>
<tr>
<td>12</td>
<td>Simulation Logic Diagram</td>
<td>19</td>
</tr>
<tr>
<td>13</td>
<td>Diagram Input Box Format</td>
<td>20</td>
</tr>
<tr>
<td>14</td>
<td>Box Format for Diagrams</td>
<td>21</td>
</tr>
<tr>
<td>15</td>
<td>Sample of Storage Sequence for MILNE Integration</td>
<td>31</td>
</tr>
<tr>
<td>16</td>
<td>Actuator Simulation</td>
<td>32</td>
</tr>
<tr>
<td>17</td>
<td>Revised Actuator Simulation</td>
<td>34</td>
</tr>
<tr>
<td>18</td>
<td>SAACT Subroutine Diagram</td>
<td>35</td>
</tr>
<tr>
<td>19</td>
<td>Function SADSPA - Dead Space</td>
<td>36</td>
</tr>
<tr>
<td>20</td>
<td>Function SALIMT - Special Purpose Limit</td>
<td>38</td>
</tr>
<tr>
<td>21</td>
<td>Function SAMOIN - Mode Controlled Integrator</td>
<td>39</td>
</tr>
<tr>
<td>22</td>
<td>Function SASWCH - Binary Switch Routine With Hysteresis</td>
<td>40</td>
</tr>
<tr>
<td>23</td>
<td>Function SAWFAT</td>
<td>41</td>
</tr>
<tr>
<td>24</td>
<td>Function SLFVPG</td>
<td>43</td>
</tr>
<tr>
<td>25</td>
<td>Function SLOAM</td>
<td>45</td>
</tr>
<tr>
<td>26</td>
<td>Function SLMVPG</td>
<td>46-47</td>
</tr>
<tr>
<td>27</td>
<td>Variable Increment Table Look-Up</td>
<td>48</td>
</tr>
<tr>
<td>28</td>
<td>Variable Increment Table Look-Up</td>
<td>49</td>
</tr>
<tr>
<td>29</td>
<td>Variable Increment Table Look-Up</td>
<td>50</td>
</tr>
<tr>
<td>30</td>
<td>Function SPELOW - Burner Blowout Limit Routine</td>
<td>52</td>
</tr>
<tr>
<td>31</td>
<td>Function SPOMP</td>
<td>53-61</td>
</tr>
<tr>
<td>32</td>
<td>Function SPMEMP</td>
<td>62</td>
</tr>
<tr>
<td>33</td>
<td>Function SPTACL</td>
<td>65</td>
</tr>
<tr>
<td>34</td>
<td>Constant Increment Table Look-Up</td>
<td>66</td>
</tr>
<tr>
<td>35</td>
<td>Constant Increment Table Look-Up</td>
<td>67</td>
</tr>
<tr>
<td>36</td>
<td>Constant Increment Table Look-Up</td>
<td>68</td>
</tr>
<tr>
<td>37</td>
<td>Constant Increment Table Look-Up</td>
<td>69</td>
</tr>
<tr>
<td>38</td>
<td>Function SPTURB</td>
<td>70-74</td>
</tr>
<tr>
<td>39</td>
<td>Function SARECT</td>
<td>75</td>
</tr>
<tr>
<td>40</td>
<td>Function SPTMCV</td>
<td>76</td>
</tr>
<tr>
<td>41</td>
<td>Parameters Required for Initialization</td>
<td>78</td>
</tr>
</tbody>
</table>

(The reverse of this page is blank)
## LIST OF TABLES

<table>
<thead>
<tr>
<th>Table No.</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>Name Prefix List.</td>
<td>4</td>
</tr>
<tr>
<td>II</td>
<td>Comparison of DSL/90 And MIMIC.</td>
<td>24-28</td>
</tr>
</tbody>
</table>
Section I

INTRODUCTION

To accomplish the task of writing a propulsion system simulation, three groups, each knowledgeable in one or more of the three technical areas involved, induction system, engine, and control, were brought together. Initial efforts were on an individual basis with each group programming their portion of the system in their own terminology. At an early stage in the coordination of the effort, it was obvious that to avoid chaos a system of terminology, in commonly understood terms, would be required. Also, that the detailed information as to just what simulation logic was being used at any time must be recorded in a standard fashion so that all groups would know exactly what was being simulated. To accomplish this, the system described in this volume was developed. The system, although primarily developed before much simulation programming was done, continued to evolve as needs for additional capabilities arose.

Exceeding computer storage space is always a danger in a complex problem such as a propulsion system simulation. To forestall, if not prevent, this occurrence any blocks of logic which are general in nature have been removed from the simulation and placed in subprograms. This saves space in two ways. First, the variables calculated internally to the subprogram do not count against the DSL/90 limits on the number of variables. Secondly, the logic is stored only once and is used as many times as is needed. These subprograms are described within this volume with program listings presented in Appendix I.

The last step before a simulation can occur, and normally the step least thought about, is initialization. The steady state operating point must be established before the transient being simulated is introduced. The procedure followed in the program developed under this task is shown by an example discussed in this volume.
Section II

DOCUMENTATION FORMAT

GENERAL OBJECTIVE

The system of documentation described in this section was developed to allow the three participating companies to first, have a common terminology during the development of the propulsion system simulation program and second, to have a method of recording, or documenting, each simulation run. Toward this goal, a naming convention for program parameters, a series of forms to record pertinent system information, and a format for simulation logic diagrams were developed.

VARIABLE NAMES

When naming variables, two opposing methods are open to the programmer. He may use a name similar to the engineering name, severely abbreviated by restrictions on length, six characters, and available symbols, no greek alphabet, no lower case letters, no sub or superscripts, and no non-alphanumeric symbols (i.e. /, , etc). Discouraged by his inability to express more complex engineering terms in a meaningful form, the programmer can then choose the opposite extreme and just number all parameters and have a key list to identify the meaning. This is a most flexible scheme, but causes the loss of all immediate visibility to the program. Parameter 6109 does not mean much until you have memorized several hundred names or looked up its meaning. For these reasons, a compromise system hopefully combining the best features of both was adopted for the propulsion system simulation. This system is described as follows.

Each variable name is composed of six characters. The first three, and in the case of control system variable names the first four, must follow the naming convention. The remaining characters are assignable at the option of the programmer with one exception. If the name describes a table, the letter T shall appear in one of the optional character locations.

The first two characters in each parameter name must be a prefix from a standard list, shown in table I. The next character is a number which designates the subsystem within which the parameter is generated. In the case of the control system, this is carried one level further by having the third character show the control system designation and the fourth character the subsystem affected. A typical subsystem numbering scheme for a propulsion system with a twin-duct inlet, a turbofan engine, and an integrated propulsion system control is shown in figure 1 with a schematic of the propulsion system and an example of several parameters and their engineering names.
<table>
<thead>
<tr>
<th>Prefix</th>
<th>Description</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aφ</td>
<td>Area</td>
<td>In (^2)</td>
</tr>
<tr>
<td>CN</td>
<td>Input constant</td>
<td></td>
</tr>
<tr>
<td>EC</td>
<td>External command</td>
<td></td>
</tr>
<tr>
<td>ET</td>
<td>Efficiency</td>
<td></td>
</tr>
<tr>
<td>Fφ</td>
<td>Thrust</td>
<td>Lbs</td>
</tr>
<tr>
<td>GM</td>
<td>Ratio of specific heats</td>
<td></td>
</tr>
<tr>
<td>HD</td>
<td>Enthalpy difference</td>
<td>BTU/Lb</td>
</tr>
<tr>
<td>MN</td>
<td>Mach number</td>
<td></td>
</tr>
<tr>
<td>Nφ</td>
<td>Rotor speed</td>
<td>RPM</td>
</tr>
<tr>
<td>NR</td>
<td>Rotor speed ratio</td>
<td></td>
</tr>
<tr>
<td>Pφ</td>
<td>Pressure</td>
<td>PSI</td>
</tr>
<tr>
<td>PR</td>
<td>Pressure ratio</td>
<td></td>
</tr>
<tr>
<td>QA</td>
<td>General variable originated by Autonetics</td>
<td></td>
</tr>
<tr>
<td>QL</td>
<td>General variable originated by LAD</td>
<td></td>
</tr>
<tr>
<td>QP</td>
<td>General variable originated by P&amp;WA</td>
<td></td>
</tr>
<tr>
<td>RE</td>
<td>Reynolds number</td>
<td></td>
</tr>
<tr>
<td>SA</td>
<td>Subroutines originated by Autonetics</td>
<td></td>
</tr>
<tr>
<td>SL</td>
<td>Subroutines originated by LAD</td>
<td></td>
</tr>
<tr>
<td>SP</td>
<td>Subroutines originated by P&amp;WA</td>
<td></td>
</tr>
<tr>
<td>Tφ</td>
<td>Temperature</td>
<td>°R</td>
</tr>
<tr>
<td>TR</td>
<td>Temperature ratio</td>
<td></td>
</tr>
<tr>
<td>Uφ</td>
<td>Velocity</td>
<td>Ft/Sec</td>
</tr>
<tr>
<td>Vφ</td>
<td>Volume</td>
<td>Ft(^3)</td>
</tr>
<tr>
<td>WA</td>
<td>Air flow</td>
<td>Lb/Sec</td>
</tr>
<tr>
<td>WF</td>
<td>Fuel flow</td>
<td>Lb/Sec</td>
</tr>
<tr>
<td>WG</td>
<td>Gas flow</td>
<td>Lb/Sec</td>
</tr>
<tr>
<td>WQ</td>
<td>Weight Quantity</td>
<td>Lbs</td>
</tr>
<tr>
<td>Xφ</td>
<td>Position</td>
<td>In</td>
</tr>
</tbody>
</table>
Figure 1. Propulsion System Numbering Scheme
In order to fully define the parameter, a keying list or "dictionary" is required. Figure 2 shows a sample dictionary based on the same system described above. In this dictionary, the engineering name can be looked up to get the program name and the definition. The greek letter name is handled by spelling out the letter and offsetting the name on the tab card. This causes all greek letter names to sort out separately and in a pseudo-alphabetic order. The dictionary is also sorted by the program name to allow easy cross-reference.

FORMS

To record the information required to describe and, if necessary, duplicate a simulation run, the following series of forms were developed.

RUN SUMMARY SHEET

The basic form for the system is the run summary sheet. This form provides the information on what was run, how it was run and what happened to the run. Copies of this form are distributed to each participant and attached to the computer printout. Figures 3 and 4 illustrate the run summary sheet and its use, also the continuation sheet that may be used as needed. The information in the heading block is self explanatory until the space for set up base is encountered. The set up base states the specific deck set up used which is described on a form identified by the number in this space. The number of the tape containing the DSL/90 system program is entered, if used, in the DSL/90 tape space.

The series of boxes referring to bases are used to identify the component being simulated and the specific simulation logic, associated subprograms, tables, and output being used. The form, as shown, provides for five phases of inlet operation for left and right inlets. The phases, as used, are ST (started), UN (unstarted), EF (empty-fill), SB (subcritical), and HS (hammershock). The form then provides five columns for engine components of which the example, using a single turbofan engine, only uses one. The control system is identified by the final column. If separate inlet and engine controls were used, the engine control logic would either be included in the engine logic or be identified by one of the engine component boxes.

The input data, other than tables, used for a particular run is recorded in the Input Data columns. Space is provided to record the subsystem requiring the data, which is redundant when the name convention described above is used, the program name, the value, and the engineering name (variable). The notes column should give the purpose of the run, and, after the run, the results of the run. The disposition of the output should be stated in this column.

The continuation page for the run summary sheet is shown in figure 4.
### ALPHABETIC LISTING BY ENGINEERING NAME

<table>
<thead>
<tr>
<th>DELTA</th>
<th>RAMP ANGLE</th>
<th>QL1002</th>
</tr>
</thead>
<tbody>
<tr>
<td>MT</td>
<td>THROAT MACH NUMBER</td>
<td>MN1005</td>
</tr>
<tr>
<td>N1</td>
<td>FAN ROTOR SPEED</td>
<td>NO3001</td>
</tr>
<tr>
<td>N1S</td>
<td>SENSED VALUE OF FAN ROTOR SPEED</td>
<td>NO8301</td>
</tr>
<tr>
<td>TABLE</td>
<td>COMPRESSOR MAP TABLE</td>
<td>WA3T01</td>
</tr>
<tr>
<td>TT6</td>
<td>HIGH TURBINE EXIT TEMPERATURE</td>
<td>TO3006</td>
</tr>
</tbody>
</table>

### ALPHABETIC LISTING BY PROGRAM NAME

<table>
<thead>
<tr>
<th>MT</th>
<th>THROAT MACH NUMBER</th>
<th>MN1005</th>
</tr>
</thead>
<tbody>
<tr>
<td>N1</td>
<td>FAN ROTOR SPEED</td>
<td>NO3001</td>
</tr>
<tr>
<td>N1S</td>
<td>SENSED VALUE OF FAN ROTOR SPEED</td>
<td>NO8301</td>
</tr>
<tr>
<td>DELTA</td>
<td>RAMP ANGLE</td>
<td>QL1002</td>
</tr>
<tr>
<td>TT6</td>
<td>HIGH TURBINE EXIT TEMPERATURE</td>
<td>TO3006</td>
</tr>
<tr>
<td>TABLE</td>
<td>COMPRESSOR MAP TABLE</td>
<td>WA3T01</td>
</tr>
</tbody>
</table>

Figure 2. Sample Dictionary
RUN SUMMARY SHEET

RUN NO. 1414-01    DATE 2/27/68    PAGE 1 OF 1

TITLE CHECKOUT OF STARTED PHASE

ORIGINATOR WONG

SETUP BASE

DSL/90 TAPE M461

ESTIMATED ACTUAL

SIMULATION 2" 2"

MACHINE 2' 1'20"

<table>
<thead>
<tr>
<th>SUBSYSTEM</th>
<th>BASE</th>
<th>INLET</th>
<th>ENGINE COMPONENT</th>
<th>CONTROL SYSTEM</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>ST</td>
<td>UN</td>
<td>EF</td>
</tr>
<tr>
<td>SIMULATION LOGIC</td>
<td>1</td>
<td>L</td>
<td>R</td>
<td>L</td>
</tr>
<tr>
<td>USER SUPPLIED ROUTINES</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TABLE</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OUTPUT</td>
<td>*</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

* DEBUG PRINTING USED

INPUT DATA

<table>
<thead>
<tr>
<th>SUBSYSTEM</th>
<th>NAME</th>
<th>VALUE</th>
<th>VARIABLE</th>
<th>NOTES</th>
</tr>
</thead>
<tbody>
<tr>
<td>INLET</td>
<td>CN1001</td>
<td>53174852</td>
<td>λ</td>
<td>Turbofan and control system - dummy logic used.</td>
</tr>
<tr>
<td></td>
<td>CN1100</td>
<td>1.0</td>
<td>BASE</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CN1004</td>
<td>1.0</td>
<td>κAI</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CN1002</td>
<td>0.96</td>
<td>κU</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CN1003</td>
<td>1.0</td>
<td>κBP</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CN1006</td>
<td>100000</td>
<td>κHS</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CN1042</td>
<td>0.4</td>
<td>κDZ</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CN1067</td>
<td>0.2</td>
<td>κYZ</td>
<td></td>
</tr>
<tr>
<td></td>
<td>AΦ1008</td>
<td>840.0</td>
<td>Ae</td>
<td></td>
</tr>
<tr>
<td></td>
<td>XΦ1001</td>
<td>36.0</td>
<td>XL</td>
<td></td>
</tr>
<tr>
<td></td>
<td>XΦ1002</td>
<td>196.0</td>
<td>X2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>XΦ1003</td>
<td>64.0</td>
<td>XT</td>
<td></td>
</tr>
<tr>
<td></td>
<td>XΦ1021</td>
<td>36.0</td>
<td>XI</td>
<td></td>
</tr>
<tr>
<td></td>
<td>XΦ1022</td>
<td>64.0</td>
<td>XII</td>
<td></td>
</tr>
<tr>
<td></td>
<td>XΦ1023</td>
<td>86.0</td>
<td>XIII</td>
<td></td>
</tr>
</tbody>
</table>

Figure 3. Run Summary Sheet
<table>
<thead>
<tr>
<th>SUBSYSTEM</th>
<th>NAME</th>
<th>VALUE</th>
<th>VARIABLE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 4. Run Summary Sheet
SIMULATION LOGIC BASE SHEET

This sheet lists the diagrams which define the simulation logic used for a particular subsystem. The sheet, illustrated by figure 5, contains the name of each logic block, its diagram number, and the dash number or version of the diagram.

TABLE BASE SHEET

The tables, usually tabular representation of curves, used for a simulation run are listed on this sheet, as shown in figure 6. The information is similar to that provided in the previously described form.

OUTPUT BASE SHEET

DSL/90 provides for two methods of output of data. One is a procedure by which any of the program variables can be printed at a specified print time increment by listing the names to be printed on the PRINT control card. The maximum and minimum values of any parameter can also be obtained by listing the name of that parameter on the RANGE control card. Plotted data is also available on IBM 1627 equipment using the original IBM DSL/90 system and on SC-4020 equipment using the North American Rockwell Corporation (NR) modified DSL/90 system. Provision for other equipment must be provided by the user.

The list or lists of variables desired to be printed or plotted are recorded on the output base sheet shown in figure 7.

EXECUTION CONTROL BASE SHEET

Information dealing with the actual run parameters, such as the integration method, time increment, value for run termination, and the tolerance specifications, is recorded on this form. One note of caution concerning the information on this sheet is that the time increments for printing, and plotting, specified above, override the time increment for execution of fixed step integration methods if these times are smaller.

An example of an execution control base is shown in figure 8.

SET-UP BASE SHEET

The physical deck arrangement is pictured by this form. An example of one such arrangement is shown in figure 9.
SIMULATION LOGIC BASE SHEET

<table>
<thead>
<tr>
<th>DESCRIPTION</th>
<th>SUBSYSTEM</th>
<th>AIDL</th>
<th>PHASE</th>
<th>BASE NO.</th>
<th>DATE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Turbofan Inlet</td>
<td>ST</td>
<td>1</td>
<td>1/3/68</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>LOGIC BLOCK</th>
<th>DIAGRAM</th>
<th>DASH NO.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input Data</td>
<td>1100</td>
<td>01</td>
</tr>
<tr>
<td>Input Tables</td>
<td>1101</td>
<td>01</td>
</tr>
<tr>
<td>Upstream Properties</td>
<td>1110</td>
<td>01</td>
</tr>
<tr>
<td>Properties At The Terminal Shock Station</td>
<td>1120</td>
<td>01</td>
</tr>
<tr>
<td>Properties Behind The Normal Shock</td>
<td>1130</td>
<td>01</td>
</tr>
<tr>
<td>Duct Volume and Mach Number</td>
<td>1150</td>
<td>01</td>
</tr>
<tr>
<td>Subsonic Flow Total Pressure Losses</td>
<td>1151</td>
<td>01</td>
</tr>
<tr>
<td>Duct Properties</td>
<td>1152</td>
<td>01</td>
</tr>
<tr>
<td>Duct Continuity &amp; Energy</td>
<td>1153</td>
<td>02</td>
</tr>
<tr>
<td>Properties At Station Z</td>
<td>1169</td>
<td>01</td>
</tr>
<tr>
<td>Helmholtz Volume Position</td>
<td>1170</td>
<td>02</td>
</tr>
<tr>
<td>Helmholtz Volume Properties</td>
<td>1171</td>
<td>01</td>
</tr>
<tr>
<td>Zone III Bleed Upstream Of Shock</td>
<td>1180</td>
<td>01</td>
</tr>
<tr>
<td>Zone III Bleed Downstream Of Shock</td>
<td>1181</td>
<td>01</td>
</tr>
<tr>
<td>Bypass And Engine Systems Airflow</td>
<td>1182</td>
<td>01</td>
</tr>
<tr>
<td>Phase Switches</td>
<td>1189</td>
<td>01</td>
</tr>
<tr>
<td>Throat Mach Number</td>
<td>1190</td>
<td>01</td>
</tr>
</tbody>
</table>

Figure 5. Simulation Logic Base Sheet - Turbofan Inlet
<table>
<thead>
<tr>
<th>NAME</th>
<th>TABLE DASH NO.</th>
<th>VARIABLE</th>
<th>REMARKS</th>
</tr>
</thead>
<tbody>
<tr>
<td>PHT50</td>
<td>01</td>
<td>$\frac{P_{tx}}{P_{tu}}$</td>
<td></td>
</tr>
<tr>
<td>AØ1TO0</td>
<td>01</td>
<td>duct area</td>
<td></td>
</tr>
<tr>
<td>VØ1TO0</td>
<td>01</td>
<td>duct volume</td>
<td></td>
</tr>
<tr>
<td>MN1T31</td>
<td>01</td>
<td>$M_a$</td>
<td></td>
</tr>
<tr>
<td>QLT44</td>
<td>01</td>
<td>$\epsilon$</td>
<td></td>
</tr>
<tr>
<td>QLT45</td>
<td>01</td>
<td>$\phi_x$</td>
<td></td>
</tr>
<tr>
<td>QLT46</td>
<td>01</td>
<td>$\phi_y$</td>
<td></td>
</tr>
<tr>
<td>WALT32</td>
<td>01</td>
<td>$\frac{W_{iz}}{W_0}$</td>
<td></td>
</tr>
</tbody>
</table>

Figure 6. Table Base Sheet - Turbofan Inlet
### OUTPUT BASE SHEET

**DESCRIPTION:**

**PHASE**

**BASE NO.**

**DATE**

**PRINT TIME INCREMENT** _____ SECONDS

**NAMES TO BE PRINTED:**

<table>
<thead>
<tr>
<th>IND. NAME</th>
<th>NAME 1</th>
<th>NAME 2</th>
<th>NAME 3</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**NAMES FOR WHICH MAXIMA AND MINIMA ARE TO BE PRINTED:**

<table>
<thead>
<tr>
<th>IND. NAME</th>
<th>NAME 1</th>
<th>NAME 2</th>
<th>NAME 3</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**PLOT TIME INCREMENT** _____ SECONDS

**NAMES TO BE PLOTTED:**

<table>
<thead>
<tr>
<th>IND. NAME</th>
<th>NAME 1</th>
<th>NAME 2</th>
<th>NAME 3</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

Figure 7. Output Base Sheet
EXECUTION CONTROL BASE SHEET

DESCRIPTION:

CHECKOUT

INTEGRATION METHOD

- MILNE - MILNE VARIABLE TIME INCREMENT*
- RKS - RUNGE-KUTTA VARIABLE TIME INCREMENT*
- RKSFX - RUNGE-KUTTA FIXED TIME INCREMENT**
- SIMP - SIMPSON'S RULE**
- TRAPZ - TRAPEZOIDAL**
- RECT - RECTANGULAR**
- CENTRL - CENTRAL USER SUPPLIED**

* DELMIN __________ MINIMUM TIME INCREMENT (SECONDS)
** DELT __________ FIXED TIME INCREMENT (SECONDS)

NAME VALUE FOR RUN TERMINATION

<table>
<thead>
<tr>
<th>NAME</th>
<th>VALUE</th>
<th>NAME</th>
<th>VALUE</th>
<th>NAME</th>
<th>VALUE</th>
<th>NAME</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

TOLERANCES

<table>
<thead>
<tr>
<th>NAME</th>
<th>REL. ERROR (MILNE OR RKS)</th>
<th>ABS. ERROR RKS</th>
<th>NAME</th>
<th>REL. ERROR (MILNE OR RKS)</th>
<th>ABS. ERROR RKS</th>
<th>NAME</th>
<th>REL. ERROR (MILNE OR RKS)</th>
<th>ABS. ERROR RKS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 8. Execution Control Base Sheet
Figure 9. Set-Up Base Sheet
The logic for simulating each component of the propulsion system must be committed to paper in such a way that it is not only available to be coded into logic statements for the computer but also so that it may be understood by people who are not computer oriented. There are almost as many diagramming conventions as there are people so a standard diagram procedure was established for this project and is described below. Each subsystem will have one or more of each of the diagrams described.

**INPUT DATA DIAGRAM**

The first diagram in each subsystem lists the input data required by that subsystem to perform its calculations. As shown by the example in figure 10, the computer name, the engineering name, the description, and the units are specified.

**INPUT TABLES DIAGRAM**

The second diagram in the subsystem set lists the tables required by the subsystem simulation logic. The table name, description, and units are given, as well as the logic diagram in which the table is used. The example in figure 11 illustrates this diagram.

**SIMULATION LOGIC DIAGRAM**

The simulation logic is diagrammed according to the following procedure. An example is shown in figure 12.

Inputs to a diagram enter on the left using a dashed box with the program name of the parameter inside. The source of the input parameter is denoted by a subsystem name above the arrow to the left of the input box. If the input is from another diagram within the same subsystem the diagram number should be noted under the arrow, otherwise no entry is placed under the arrow. The engineering name appears to the right of the input box, as it does on all boxes, above the arrow showing the path of the logic. When an input box is a table name, no engineering name is used.

The numbering convention for diagrams is similar to that for parameter names in that the numbers follow the convention used for the third character of the program name. A list of these numbers with abbreviations for the subsystems of the previously used example in figure 1 are given in figure 13. Also shown are the general rules for the above described input boxes and several examples. The formats for logic boxes within the diagram are shown on figure 14. In example B, the function described is a routine which, when given the flow parameter, computes Mach number. Since there
<table>
<thead>
<tr>
<th>NAME</th>
<th>VARIABLE</th>
<th>DESCRIPTION</th>
<th>UNITS</th>
</tr>
</thead>
<tbody>
<tr>
<td>CN1001</td>
<td>$\lambda$</td>
<td>Sonic Flow Constant</td>
<td></td>
</tr>
<tr>
<td>CN1002</td>
<td>$K_u$</td>
<td>Inlet Throat Sonic Flow Coefficient</td>
<td></td>
</tr>
<tr>
<td>CN1003</td>
<td>$K_{bp}$</td>
<td>$P_{tbp}/P_{t2}$</td>
<td></td>
</tr>
<tr>
<td>CN1004</td>
<td>$K_A$</td>
<td>$A_d/A_{dgeo}$</td>
<td></td>
</tr>
<tr>
<td>CN1006</td>
<td>$K_{HS}$</td>
<td>Hammershock Indicator Constant</td>
<td></td>
</tr>
<tr>
<td>CN1100</td>
<td>Base</td>
<td>I/C Base No.</td>
<td></td>
</tr>
<tr>
<td>CN1042</td>
<td>$K_{dz}$</td>
<td>Duct Total Pressure Loss Constant Between</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Stations d and z</td>
<td></td>
</tr>
<tr>
<td>CN1067</td>
<td>$K_{yz}$</td>
<td>Helmholtz Volume Total Pressure Loss Constant</td>
<td></td>
</tr>
<tr>
<td>XØ1001</td>
<td>$X_L$</td>
<td>Cowl Lip Station</td>
<td>in.</td>
</tr>
<tr>
<td>XØ1002</td>
<td>$X_2$</td>
<td>Engine Face Station</td>
<td>in.</td>
</tr>
<tr>
<td>XØ1003</td>
<td>$X_T$</td>
<td>Throat Station</td>
<td>in.</td>
</tr>
<tr>
<td>XØ1021</td>
<td>$X_I$</td>
<td>Station I</td>
<td>in.</td>
</tr>
<tr>
<td>XØ1022</td>
<td>$X_{II}$</td>
<td>Station II</td>
<td>in.</td>
</tr>
<tr>
<td>XØ1023</td>
<td>$X_{III}$</td>
<td>Station III</td>
<td>in.</td>
</tr>
<tr>
<td>Q11101</td>
<td>$l$</td>
<td>Helmholtz Volume Length</td>
<td>in.</td>
</tr>
<tr>
<td>AØ1008</td>
<td>$A_c$</td>
<td>Capture Area</td>
<td>in.$^2$</td>
</tr>
</tbody>
</table>

**DIAGRAM 1100-01**

**STARTED PHASE INPUT DATA**

*Figure 10. Input Data List*
<table>
<thead>
<tr>
<th>TABLE NAME</th>
<th>DESCRIPTION</th>
<th>OUTPUT UNITS</th>
<th>DIAGRAM WHERE USED</th>
</tr>
</thead>
<tbody>
<tr>
<td>AØ1T00</td>
<td>Duct area versus station, throat area</td>
<td>in.²</td>
<td>1120</td>
</tr>
<tr>
<td>MN1T31</td>
<td>$M_A$ versus $M_o$, $\alpha_o$, $\psi_o$</td>
<td>--</td>
<td>1169</td>
</tr>
<tr>
<td>PR1T50</td>
<td>$P_{tx}/P_{to}$ versus $M_o$, $\alpha_o$, $\psi_o$</td>
<td>--</td>
<td>1110</td>
</tr>
<tr>
<td>QL1T44</td>
<td>$\epsilon$ versus $M_A$, throat area</td>
<td>--</td>
<td>1110</td>
</tr>
<tr>
<td>QL1T45</td>
<td>$\phi_x$ versus $M_A$, throat area</td>
<td>--</td>
<td>1180</td>
</tr>
<tr>
<td>QL1T46</td>
<td>$\phi_y$ versus $M_A$, throat area</td>
<td>--</td>
<td>1181</td>
</tr>
<tr>
<td>WØ1T00</td>
<td>Duct volume versus station, throat area</td>
<td>ft.³</td>
<td>1150</td>
</tr>
<tr>
<td>WALT32</td>
<td>$W_{II}/W_o$ versus $M_o$, $\alpha_o$, $\psi_o$</td>
<td>--</td>
<td>1110</td>
</tr>
</tbody>
</table>

DIAGRAM 1101-01

STARTED PHASE INPUT TABLES

Figure 11. Input Data List
Figure 12. Simulation Logic Diagram
### DIAGRAM TERMINOLOGY

<table>
<thead>
<tr>
<th>Designation</th>
<th>Subsystem</th>
<th>Diagram Numbers</th>
</tr>
</thead>
<tbody>
<tr>
<td>AISL</td>
<td>Inlet duct, left</td>
<td>1---</td>
</tr>
<tr>
<td>AISR</td>
<td>Inlet duct, right</td>
<td>2---</td>
</tr>
<tr>
<td>TFAN</td>
<td>Turbofan engine</td>
<td>3---</td>
</tr>
<tr>
<td>PCS</td>
<td>Propulsion control system</td>
<td>4---</td>
</tr>
</tbody>
</table>

Example A:

(Input from diagram in same subsystem.)

![Diagram Input Box Format Example A](image)

Example B:

(Input from another subsystem.)

![Diagram Input Box Format Example B](image)

Figure 13. Diagram Input Box Format
Figure 14. Box Format for Diagrams
are two answers possible, the note $M_x \geq 1.0$ is used to show which solution is desired. Similarly, in example C the function INSW is described in mathematical terms so that visibility to the engineer is enhanced. In example D, a box is shown in which a computation is made and no program name is given. This means that the computation is made internally to the next box upstream on the logic path and the result of that unnamed box is not available as an output.
Section III

SELECTION OF SIMULATION LANGUAGE

COMPARISON

A study was made of features of both the MIMIC and DSL/90 simulation languages. The results of that study are tabulated in table II.

An existing simulation program utilizing the General Electric Company Dynasyar language was converted to both DSL/90 and MIMIC simulation languages. Both ran satisfactorily at the USAF Aero Propulsion Laboratory and at North American Rockwell Corporation. Comparisons of engine face total pressure versus time, inlet terminal shock position versus time, and shock velocity versus time showed near identical results.

CONCLUSION

As a result of the study, it was decided to proceed with DSL/90 as the simulation language for the propulsion system simulation.
<table>
<thead>
<tr>
<th><strong>Language</strong></th>
<th><strong>DSL/90</strong></th>
<th><strong>MIMIC</strong></th>
<th><strong>Comments</strong></th>
<th><strong>Advantage</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>TABLE II. COMPARISON OF DSL/90 AND MIMIC</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>1) Configuration Description</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>2) Integrator</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>a) Method</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Group A - Variable interval</td>
<td>Fourth order Runge-Kutta with variable</td>
<td>Same relative or absolute tolerances for all integrators.</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Group B - Fixed interval fourth order</td>
<td>Runge-Kutta</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>b) Tolerance Control</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tolerances for each individual integrator. Relative and absolute tolerances for Runge-Kutta. Only relative tolerances for Milne.</td>
<td>Same relative or absolute tolerance for all integrators.</td>
<td>Individual tolerances permit looser tolerances for variables that do not require high accuracy during integration, but are the controlling variables during parts of the transient. Result is a saving of machine time.</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>3) Function Routine</td>
<td>DSL/90</td>
<td>MIMIC</td>
<td>Comments</td>
<td>Advantage</td>
</tr>
<tr>
<td>---------------------</td>
<td>--------</td>
<td>--------</td>
<td>----------</td>
<td>-----------</td>
</tr>
<tr>
<td>a) Specification</td>
<td>FORTRAN IV and MAP.</td>
<td>FORTRAN IV and MAP.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>b) Changes</td>
<td>Can be easily altered.</td>
<td>Can be easily altered.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>c) Additions</td>
<td>Restricted only by core storage.</td>
<td>Only five functions with specified names can be added.</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>d) Loading</td>
<td>Only functions called for in simulation are loaded.</td>
<td>All functions must be loaded.</td>
<td>Minimize storage.</td>
<td>X</td>
</tr>
<tr>
<td>4) Input Data</td>
<td>Floating point, variable field identified by variable name. Restricted by card size.</td>
<td>Floating point, fixed field and identified by the order of input. Restricted to six values per card.</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>a) Format</td>
<td>Any order; constant not input are assumed to be zero or previous case value.</td>
<td>Input complete list of constants beginning and in order.</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>b) Constant</td>
<td>Any order.</td>
<td>Input complete list of parameters in order. Must reinput for each subsequent case.</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>c) Parameter</td>
<td>Any order.</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
TABLE II. COMPARISON OF DSL/90 AND MIMIC (Continued)

<table>
<thead>
<tr>
<th>Language</th>
<th>DSL/90</th>
<th>MIMIC</th>
<th>Comments</th>
<th>Advantage</th>
</tr>
</thead>
<tbody>
<tr>
<td>d) Tables Book Linear and La Grange Linear interpolation. Function value equal to table limit when independent variable is outside of tabulated range. Warning is printed out.</td>
<td>Linear interpolation. Function value equal to zero when independent variable is outside of tabulated range.</td>
<td>DSL/90 input to tables is more convenient.</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>3 Dimensional</td>
<td>Not available.</td>
<td>Linear interpolation and independent values must fall inside tabulated range.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>e) Array</td>
<td>Yes.</td>
<td>No direct provision, but can use an alternate three dimensional table input method. Only three values per card possible.</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>5) Output a) Specification</td>
<td>Variables by name.</td>
<td>Heading and variables by name.</td>
<td>DSL/90 offers an easier callout procedure. MIMIC offers a greater flexibility in printout.</td>
<td></td>
</tr>
<tr>
<td>Language</td>
<td>DSL/90</td>
<td>MIMIC</td>
<td>Comments</td>
<td>Advantage</td>
</tr>
<tr>
<td>----------</td>
<td>--------</td>
<td>-------</td>
<td>----------</td>
<td>-----------</td>
</tr>
<tr>
<td><strong>b) Format</strong></td>
<td>Time, variable name and values printed. If nine or fewer variables are printed, names are printed only once at column heads.</td>
<td>Variable names are printed only at initial time.</td>
<td>Variable names being printed only once is desirable only if there are few enough variables so that they may be printed on one line allowing columnar tabulation.</td>
<td>X</td>
</tr>
<tr>
<td><strong>c) Special Printout</strong></td>
<td>Minimum and maximum. All variables at each iteration beginning at specified time.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td><strong>6) Special Feature</strong></td>
<td>Part or all of the program can be omitted in sorting. Procedural logic. Repeatable procedural logic. Compiled simulation deck.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Saves sorting time.</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Saves sorting time.</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Additional capability in branching reduces number of statements.</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Saves machine time in sorting and compiling.</td>
<td>X</td>
</tr>
</tbody>
</table>
TABLE II. COMPARISON OF DSL/90 AND MIMIC (Concluded)

<table>
<thead>
<tr>
<th>Language</th>
<th>DSL/90</th>
<th>MIMIC</th>
<th>Comments</th>
<th>Advantage</th>
</tr>
</thead>
<tbody>
<tr>
<td>7) Run Time</td>
<td>Program is divided into two steps, Translate and Simulate. The two steps require more time than the single step of MIMIC. Because there is a compiled simulation deck, overall time should be better than MIMIC in subsequent production runs which would not require the Translate step.</td>
<td>Program sorts and assembles a machine language program for each run.</td>
<td>No comparable time study made. Load time has been shorter for MIMIC. One short run will be better with MIMIC.</td>
<td></td>
</tr>
<tr>
<td>8) Program Capacity</td>
<td>Can be increased with overlay. Tradeoff with tables possible.</td>
<td>Tradeoff with tables possible.</td>
<td>DSL/90 has greater flexibility.</td>
<td>X</td>
</tr>
</tbody>
</table>
Section IV

SUPPORTING SUBPROGRAMS

GENERAL APPROACH

To minimize the creation of parameter names which were not required as printed output and to economize on computer storage space usage, numerous subprograms have been written to support the propulsion system simulation program.

A naming convention for all support subprograms was established so that subprograms could be written by each of the three participants without danger of name duplication and also to allow the routine to be readily identified as to origin. This convention has all Autonetics routines begin with SA, Pratt and Whitney with SP, and NR Los Angeles Division with SL.

All basic algorithms have been removed from the simulation logic and placed in either function or subroutine form. An example of a large block of logic thus removed is the compressor subroutine SPCOMP. This logic is used three times in the simulation of the turbofan and once in a turbojet simulation with only the particular map being used and the names of the inputs and outputs being changed. Removing this section from the simulation logic removes a large number of new variable names from the restricted number DSL/90 allows and saves the locations the duplicated logic instructions would use.

DESCRIPTIONS

The subprograms are described in alphabetical order. A short description of the purpose of the subprogram is given followed by a flow diagram and the computer compilation. In the case of several control routines, the flow diagrams are presented in several forms to provide maximum understanding of their purpose.

When the coding of the subprograms began, the decision had been made, primarily on previous experience with General Electric's DYNAHYAR simulation language, to use a variable time step integration method. The method chosen was the DSL/90 MILNE integration scheme. The first checkouts of separate propulsion system components, several inlets, a turbojet engine, and a turbofan engine were successfully run using MILNE. When an integrated, although simplified, propulsion system control was added to an inlet and a turbojet some strange things began to occur. In the course of tracing these strange occurrences a liberal education in DSL/90 was obtained.
The MILNE integration scheme uses six slices of history plus the current calculation to operate its predictor-corrector. The method used for "cutting back" or reducing the time slice because the current calculations exceed tolerances is an involved one.

The problem is best described by the example shown on figure 15. This example illustrates the changes in past history made by three successive failures of the current calculation to meet tolerances. The effect on the subprograms that use any past values is to require constant testing of time on each execution pass and appropriate changes to the subprogram history. It was also discovered that DSL/90's HSTRSS (hysteresis) routine did not appear to handle this history correctly. Since considerable work appeared to be required to clear up all the problems brought on by use of the variable time step it was decided to dispense with it for the time being. It will be reconsidered after the simulations required in Phase II are full operational. In the meantime, one of the fixed time step integration methods will be used.

As a result of this decision, some of the subprogram listings presented in Appendix I have sections dealing with past history that do not show up on the diagrams and flow charts in the following figures. This added logic is being removed as time is available and the subprogram decks will ultimately agree with the diagrams presented herein. If it appears that use of the variable step iteration has advantages worth the cost of implementation, a supplementary report will be issued on the variable step versions of these programs.

SUBROUTINE SAACT

Simulation of the numerous actuators within the Propulsion Control System has been achieved by use of subroutine described below and in the accompanying figures.

The actuator simulation depicted in figure 16 is composed of the following components:

1. A limiter acting on the input or command value (Xc)
2. A feedback signal which may be selected from either the output of the integrator (X) or the output of the actuator (XH) which includes hysteresis effects
3. A loop gain term (KA)
4. A limiter acting on the rate (XDOT)
5. An integrator.
<table>
<thead>
<tr>
<th>HISTORY STORAGE LOCATIONS</th>
<th>CURRENT STORAGE</th>
<th>TIME STEP</th>
<th>TOLERANCE TEST</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1/4</td>
<td>2/4</td>
</tr>
<tr>
<td>0</td>
<td>1/4</td>
<td>2/4</td>
<td>3/4</td>
</tr>
<tr>
<td>0</td>
<td>1/4</td>
<td>2/4</td>
<td>3/4</td>
</tr>
<tr>
<td>0</td>
<td>1/4</td>
<td>2/4</td>
<td>3/4</td>
</tr>
<tr>
<td>0</td>
<td>1/4</td>
<td>2/4</td>
<td>3/4</td>
</tr>
<tr>
<td>0</td>
<td>1/4</td>
<td>2/4</td>
<td>3/4</td>
</tr>
<tr>
<td>0</td>
<td>2/4</td>
<td>4/4</td>
<td>6/4</td>
</tr>
<tr>
<td>0</td>
<td>2/4</td>
<td>4/4</td>
<td>6/4</td>
</tr>
<tr>
<td>8/4</td>
<td>9/4</td>
<td>10/4</td>
<td>11/4</td>
</tr>
<tr>
<td>8/4</td>
<td>9/4</td>
<td>10/4</td>
<td>11/4</td>
</tr>
<tr>
<td>11/4</td>
<td>23/8</td>
<td>12/4</td>
<td>25/8</td>
</tr>
<tr>
<td>11/4</td>
<td>23/8</td>
<td>12/4</td>
<td>25/8</td>
</tr>
<tr>
<td>11/4</td>
<td>23/8</td>
<td>12/4</td>
<td>25/8</td>
</tr>
<tr>
<td>11/4</td>
<td>12/4</td>
<td>13/4</td>
<td>14/4</td>
</tr>
</tbody>
</table>

**Figure 15.** Sample of Storage Sequence for MILNE Integration
Figure 16. Actuator Simulation
The integrator requires an initial condition value (XIC) and a derivative (XDOT). A parameter, γ, is used to select the feedback signal desired. This parameter is set at either unity or zero with the former selecting feedback from the integrator output.

In order to implement this simulation in DSL/90, a revision to the simulation diagram of figure 16 was made. This revised diagram, shown in figure 17, is functionally identical to the original diagram except in arrangement. The revised diagram allows the components in the dashed box to be placed in the subroutine under discussion (SAACT).

The FORTRAN flow diagram for SAACT is shown on figure 18.

FUNCTION SADSPA

A routine to perform switching operations with provision for a "dead space" is required in the Propulsion Control System (PCS). The standard DSL/90 dead space routine (DEADSP) is limited to a linear output function with unity slope which is not suitable for the discrete function switching requirements of the PCS.

The inputs to SADSPA are three in number, the independent parameter, X, and the left and right limits of the dead space. When the value of X is below the left limit, the function value is a negative one (-1), above the right limit the function value is a positive one (+1), and between the limits the output is zero (0).

This routine has a secondary usage as a switch similar to function SASWCH without hysteresis when either the left or right limits are set above or below any possible value of the independent parameter.

The operation of this function is shown in diagrammatic form and FORTRAN flow form in figure 19.

SUBROUTINE SALIMT

When the limits of a function are computed values there exists the possibility of the minimum limit exceeding the maximum limit. In such an instance in PCS simulation it is desirable to be able to specify which limit has priority. To accomplish this the SALIMT routine was written.

By the use of the argument TYPE the routine will give priority to maximum (TYPE=1.0), minimum (TYPE=0.0) or ignore both limits (TYPE=-1.0). When the maximum and minimum limits are in their normal positions the routine functions as a normal limit routine.
Figure 17. Revised Actuator Simulation
Figure 18. SAAC T Subroutine Diagram
Figure 19. Function SADSPA - Dead Space
The basic limiter operation is illustrated in figure 20. The special priority feature is not shown on the upper diagram but is shown on the FORTRAN flow representation.

FUNCTION SAMOIN

To perform the integral plus proportional function of the PCS, a specialized integrator is required. This integrator must be capable of three modes of operation; normal integration, holding at a limiting value with immediate change when the derivative changes sign, and resetting to an initial value not necessarily the same as the original initial value.

This capability is obtained by use of the function SAMOIN in conjunction with a DSL/90 integrator. The operation of this function, shown diagrammatically and in FORTRAN flow form, is presented in figure 21.

FUNCTION SASWCH

A binary (0, 1) switching function that provides hysteresis is required for PCS simulation. The two entries, SAOFON and SAONOF, provide this capability for a normally off and a normally on binary switch, respectively.

These entries may be used as normally off and normally on switches without hysteresis by setting the upper and lower limits, XL and XU, equal.

Figure 22 illustrates the operation of these entries to the function and the FORTRAN flow diagram.

FUNCTION SAWFAT

This function supplies a binary signal to a mode controlled integrator indicating the need for attenuating the maximum Wf/P4 limit. When the output of this routine is zero (0) the gain in the Wf/P4 logic is reduced by a amount determined by input data. When the SAWFAT inputs change, allowing the output to return to unity (1), the integrator returns the fuel flow limit gain to its original value.

Figure 23 shows the flow of FORTRAN logic for this function.
<table>
<thead>
<tr>
<th>GENERAL FORM</th>
<th>FUNCTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>$y = \text{SALIMT}(x,x_L,x_U,\text{type})$</td>
<td>WHEN: $x_L \leq x \leq x_U$ THEN: $y = x$</td>
</tr>
</tbody>
</table>

**Figure 20. Function SALIMT - Special Purpose Limit Function**

```
SALIMT (y,x_L,x_U,TYPE)

SALIMT = x

RETURN

TYPE < 0.0

XL > XU

SALIMT = AMAX1(AMIN1(x,x_U),x_L)

RETURN

TRUE

SALIMT = AMAX1(x,x_L)

RETURN

SALIMT = AMIN1(x,x_U)

RETURN

TYPE = 0.0

TRUE

RETURN
```
<table>
<thead>
<tr>
<th>General Form</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_{dt} = \text{SAMOIN}(dx_{dt}, x, x_{l}, x_{u}, \text{PATH}, x_{IC}, dx_{dtm})$</td>
<td>$\int_0^t x_{d\dot{t}} dt + I/C$ \text{PATH} \neq 0</td>
</tr>
<tr>
<td>$x = \text{INTGRN}(I/C, x_{d\dot{t}})$</td>
<td>$x = x_{IC}$ \text{PATH} = 0</td>
</tr>
<tr>
<td>$x_{l} \leq \int_0^t x_{d\dot{t}} dt + I/C \leq x_{u}$</td>
<td>[ x \leq x_{l} ] \text{PATH} = 0</td>
</tr>
</tbody>
</table>

**Diagram**

---

**Figure 21. Function SAMOIN - Mode Controlled Integrator**

39
<table>
<thead>
<tr>
<th>General Form</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>( y = \text{SASFON}(x_1, x, x_1, x) )</td>
<td><strong>OFF-ON Switch With Hysteresis</strong></td>
</tr>
<tr>
<td>WHEN: ( x &lt; x_1 ); ( y = 0.0 ) ( x &gt; x_u ); ( y = 1.0 ) ( x &lt; x_u ) and ( y_{n-1} = 0.0 ); ( y = 0.0 ) ( x &gt; x_l ) and ( y_{n-1} = 1.0 ); ( y = 1.0 )</td>
<td></td>
</tr>
<tr>
<td>( y )</td>
<td></td>
</tr>
<tr>
<td>( 1.0 )</td>
<td></td>
</tr>
<tr>
<td>( 0 )</td>
<td></td>
</tr>
<tr>
<td>( x )</td>
<td></td>
</tr>
<tr>
<td>( x_l )</td>
<td></td>
</tr>
<tr>
<td>( x_u )</td>
<td></td>
</tr>
<tr>
<td><strong>ON-OFF Switch With Hysteresis</strong></td>
<td></td>
</tr>
<tr>
<td>( y = \text{SAFNON}(x_1, x, x_1, x) )</td>
<td></td>
</tr>
</tbody>
</table>

**Figure 22. Function SASWCH - Binary Switch Routine With Hysteresis**
Figure 23. Function SAWFAT
FUNCTION SLFVPG

This utility routine computes the airflow parameter as a function of pressure ratio and gamma (ratio of specific heats).

The derivation of this function is shown below.

W  airflow        lb/sec
T  static temperature  °R
T'T  total temperature  °R
P  static pressure     lb/sq. in.
P't  total pressure     lb/sq. in.
A  area               sq. in.
TR  temperature ratio (T'T/T)
PR  pressure ratio (P't/P)

\[ W = AV_P = \frac{AMP}{\sqrt{T}} \sqrt{\frac{\gamma g}{R}} \]

\[ \frac{W \sqrt{T}}{AP} = M \sqrt{\frac{\gamma g}{R}} \]

\[ \frac{W \sqrt{T'}}{AP_T} = \frac{W \sqrt{T}}{AP} \frac{TR^{\frac{1}{2}}}{PR} \]

\[ M = \sqrt{2(TR - 1.0)/(\gamma - 1.0)} \]

\[ TR = PR \frac{\gamma}{1-\gamma} \]

Therefore:

\[ \frac{W \sqrt{T'}}{P'T} = \sqrt{\frac{2(TR - 1)}{\gamma - 1}} \sqrt{\frac{\gamma g}{R}} \frac{\gamma^{\frac{1}{2}}}{TR^{2(\gamma - 1)}} \]

Figure 24 shows the FORTRAN flow diagram of this function.
Figure 24. Function SLFVG

\[
FP = 0.0
\]

\[
PR = PRATIO
\]

\[
\text{GAM} = \text{GAMMA}
\]

\[
\text{CTR} = 1 + \left( \frac{\text{GAM}-1.0}{2.0} \right)
\]

\[
\text{TR} = \frac{\text{GAM}}{\text{CTR}}
\]

\[
\text{TR} > \text{CTR}
\]

\[
FP = \sqrt{1.206 \times \frac{(\text{TR}-1.0)/(\text{GAM}-1.0)}{(\text{CTR}+1.0)/(\text{TR}+2.0)(\text{GAM}-1.0)}}
\]

\[
\text{SLFVG} = FP
\]

RETURN
FUNCTION SIGAM

This utility routine computes the ratio of specific heats ($\gamma$) as a function of total temperature and fuel-air ratio. The calculations use Grade JP-4 fuel combustion characteristics and are applicable to all of the JP family of fuels.

Two input arguments are used when gamma for a fuel-air mixture is desired. The first argument is temperature, in degrees Rankine, the second is fuel-air ratio, dimensionless. When gamma for air is required, only the temperature argument need be input.

The FORTRAN flow diagram is shown in figure 25.

FUNCTION SLMVFG

This function computes mach number as a function of flow parameter and gamma. The calculation uses a Newton-Raphson iteration using the initial guess for Mach number to determine which solution, subsonic or supersonic, is desired. On successive passes through the routine, the output from the previous pass is used as the initial guess to reduce iteration time.

Inputs to the routine are the initial guess on mach number, XMI, the flow parameter, FLOWP, and the ratio of specific heats, $\gamma$. The flow parameter is derived in the description of function SLFVPG. The FORTRAN flow diagram is shown on figure 26 and 27.

FUNCTION SLTLU

This function provides a general purpose table look-up program for use with DSL/90 simulation programs. Functions of one, two and three independent parameters are handled as well as constants. Interpolation in univariant and bivariant tables can be selected as either linear or LaGrangian for each independent parameter. In the trivariant tables, the interpolation between bivariant families is linear.

Figures 27, 28, and 29 illustrate the usage of this routine for tables of one, two, and three independent parameters. Also shown on figure 27 is the usage when the table value is a constant.

FUNCTION SPBLOW

The burner blowout routine incorporates logic which effectively "blows out" a burner when it tries to operate below certain minimum conditions for combustion. A curve plotting minimum burner inlet total pressure (for combustion) versus fuel-air ratio is compared with actual
FUNCTION SLGAM

\[ f(T_t, \frac{f}{a}) \rightarrow \sigma \]

**CONSTANTS**
- \( C_1 = 2.3996 \)
- \( C_2 = 0.068558 \)
- \( C_3 = 0.12149 \)
- \( C_4 = 9.00974 \times 10^{-4} \)
- \( C_5 = -7.07129 \times 10^{-7} \)
- \( C_6 = 3.41709 \times 10^{-10} \)
- \( C_7 = -8.10801 \times 10^{-14} \)
- \( C_8 = 0.00834 \)
- \( P = 5526.0/T_t \)

ARGQ (N)

**CASE 1:**
- \( N < 2 \)
  - \( f/a = 0.0 \)
  - \( T_t = 0.0 \)
  - \( \gamma = 1.4 \)
  - **RETURN**

**CASE 2:**
- \( C_{\text{pair}} = C_1 + C_2 \left( \frac{P}{e^{t_1}} \right)^{2e^p} \)
- \( C_{\text{fuel}} = C_3 + C_4 T_t + C_5 T_t^2 + C_6 T_t^3 + C_7 T_t^4 + C_8 T_t^5 \)

\[ C_{\text{gas}} = C_{\text{pair}} + f/a (C_{\text{fuel}}) \]

\[ R_{\text{gas}} = C_2 + \left( \frac{C_9 \cdot f/a}{1.0 + f/a} \right) \]

\[ \gamma = \frac{C_{\text{gas}}}{C_{\text{gas}} - R_{\text{gas}}} \]

**RETURN**

*Figure 25. Function SLGAM*
Figure 26. Function SLMVFG
Figure 26. Function SLMVFG (Concluded)
TWO DIMENSIONAL

\[ y = f(X) \]

CODING:

STORAGE YA( )

\[ y = SLTLU(YA,X) \]

TABULAR INPUT ARRAY:

<table>
<thead>
<tr>
<th>YA(1)</th>
<th>2.0</th>
<th>DENOTES TWO DIMENSIONAL TABLE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>N\text{\textsc{o}}. OF X's</td>
</tr>
<tr>
<td></td>
<td></td>
<td>N\text{\textsc{o}}. OF X POINTS FOR INTERPOLATION</td>
</tr>
</tbody>
</table>

LIST OF X's

\[
\begin{align*}
(4) & \quad X_1 \\
(5) & \quad X_2 \\
(6) & \quad X_3 \\
(7) & \quad X_4 \\
(8) & \quad X_5 \\
\vdots & \quad \vdots \\
\end{align*}
\]

LIST OF Y's

\[
\begin{align*}
() & \quad Y_1 \\
() & \quad Y_2 \\
() & \quad Y_3 \\
() & \quad Y_4 \\
() & \quad Y_5 \\
\vdots & \quad \vdots \\
\end{align*}
\]

FOR CONSTANT INPUT

<table>
<thead>
<tr>
<th>YA(1)</th>
<th>1.0</th>
<th>DENOTES CONSTANT FOLLOWING</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>VALUE OF CONSTANT</td>
</tr>
</tbody>
</table>

Figure 27. Variable Increment Table Look-Up
THREE DIMENSIONAL

\[ Z = f(X,Y) \]

STORAGE ZA( )

TABULAR INPUT ARRAY:

<table>
<thead>
<tr>
<th>ZA(1)</th>
<th>3.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>(2)</td>
<td></td>
</tr>
<tr>
<td>(3)</td>
<td></td>
</tr>
<tr>
<td>(4)</td>
<td></td>
</tr>
<tr>
<td>(5)</td>
<td></td>
</tr>
<tr>
<td>(6)</td>
<td></td>
</tr>
</tbody>
</table>

- DENOTES THREE DIMENSIONAL TABLE
- NO. OF X's
- NO. OF X POINTS FOR INTERPOLATION
- NO. OF Y's
- NO. OF Y POINTS FOR INTERPOLATION

LIST OF X's
- \( X_1 \)
- \( X_2 \)
- \( X_3 \)
- \( \vdots \)

LIST OF Y's
- \( Y_1 \)
- \( Y_2 \)
- \( \vdots \)

LIST OF Z's for \( Y_1 \)
- \( Z_{1,1} \)
- \( Z_{2,1} \)
- \( Z_{3,1} \)
- \( \vdots \)

LIST OF Z's for \( Y_2 \)
- \( Z_{1,2} \)
- \( Z_{2,2} \)
- \( Z_{3,2} \)
- \( \vdots \)

Figure 28. Variable Increment Table Look-Up
Z = f(X,Y,W)

STORAGE ZA( )

TABULAR INPUT ARRAY:

<table>
<thead>
<tr>
<th>ZA(1)</th>
<th>ZA(2)</th>
<th>ZA(3)</th>
<th>ZA(4)</th>
<th>ZA(5)</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

CODING:

Z = SLTLU(ZA,X,Y,W)

DENOTES FOUR DIMENSIONAL TABLE
NO. OF W's

LIST OF W's

W_1
W_2
W_3

THREE DIMENSIONAL FOR W,
NO. OF X's
NO. OF X POINTS FOR INTERPOLATION
NO. OF Y's
NO. OF Y POINTS FOR INTERPOLATION

LIST OF X's

X_1
X_2

SAME AS THREE DIMENSIONAL INPUT

REPEAT FOR W_2, W_3, ... ...

Figure 29. Variable Increment Table Look-Up
operating conditions. If the actual inlet total pressure is below the minimum value, the burner efficiency is reduced to zero on a time constant to simulate the blowout. This reduces the burner temperature rise to zero. When inlet total pressure returns to a value above the minimum the efficiency is restored on a time constant.

Figure 30 shows the FORTRAN flow diagram for this routine. The coding has not been accomplished for this routine so the listing will not be found in Appendix I.

SUBROUTINE SPCOMP

The purpose of this routine is to provide a simulation model for fans and compressors. The characteristics of the particular component being used are entered as table names in the argument list of the routine. Other required input data such as the incoming pressure ($P_{\text{tin}}$), the incoming temperature, ($T_{\text{tin}}$), the exit pressure ($P_{\text{out}}$), and the rotor speed ($N$) are also entered as arguments. The outputs of the routine are temperature ($T_{\text{tout}}$), change in enthalpy ($\Delta h$), airflow ($W$), and the average ratio of specific heats ($\gamma$).

Several versions of the subroutine are documented. The earliest (-01) has the outputs described above, the later version (-02) has added outputs for surge margin (SRGM) and efficiency ($\eta$). These added outputs were always computed within SPCOMP but not available externally. They were added to make them available for the SPTACL calculations.

Addition features are shown in the flow diagram of figure 31, such as interstage bleed and variable geometry provisions. These sections have not been incorporated in the coding as of this report.

FUNCTION SPMEMF

This function is to provide a history for a variable in order to break an implicit mathematical loop. The initial value, XIC, must be provided from initialization logic. After the first pass the output value is equal to the computed value, $X$, from the previous pass.

Figure 32 shows the flow diagram for the FORTRAN logic.

FUNCTION SPTACL

The purpose of the acceleration schedule calculator, SPTACL, is to calculate the maximum fuel flow ($W_{\text{fe}}/P_{\text{t}4}$) that the engine can hit on an acceleration without exceeding either a set high compressor surge margin or a maximum turbine inlet temperature. The schedule is usually used in
Figure 30. Function SPBLow - Burner Blowout Limit Routine
FUNCTION SPCOMP

\[ f(M, P_{\text{in}}, P_{\text{out}}, T_{\text{in}}, N, \text{CCT}, \text{WAT, ETAT, PRST, WAST, REYT}) \]

SPCOMP 1-01

Figure 31. Function SPCOMP
Figure 31. Function SPCOMP (Continued)
Figure 31. Function SPCOMP (Continued)
Figure 3.1. Function SPCOMP (Continued)
\[ \Delta n_{\text{surf}} = \text{CCT}(b) \]

\[ \Delta \text{SRGM} = \text{CCT}(a) \]

\[ \Delta \left( \frac{W \bar{e}}{\bar{e} t} \right) = f(l, \tau, \text{CCT}(s), \text{WCN}) \]

\[ \text{SRGM} = C(l+36) \]

\[ \text{SRGM} - 1.0 - \text{DSRGn} \]

\[ \Delta n_{\text{surf}} = 1.0 \]

\[ \Delta \text{SRGM} = 0.0 \]

\[ \Delta \left( \frac{W \bar{e}}{\bar{e} t} \right) = 1.0 \]

\[ C(l+2) = 0.0 \]

\[ C(l+1) = 0.0 \]

\[ C(l) = \text{TIME} \]

\[ \text{SPCOMP 3-01} \]
Fig. 31. Function SPCOMP (Continued)
Figure 31. Function SPCOMP (Continued)
\[ \Delta h = \Delta T_e \frac{\theta}{(\theta - 1) \cdot 0.06854} \]

\[ T_{\text{calc}} = T_{\text{in}} + \Delta T_e \]

\[ T_{\text{out}} = f(L + 18, \tau, T_{\text{calc}}) \]

\[ \phi = \left( \frac{T_{\text{in}} + T_{\text{out}}}{2.0} \right) \]

\[ \text{RETURN} \]

SPCOMP 5-01

Figure 31. Function SPCOMP (Concluded)
Figure 32. Function SPMEMF
the control as a function of high rotor speed, $N_2$ (for a turbofan engine).

The routine basically calculates the amount of fuel, $W_{fs}$, which would cause the engine to operate at a point corresponding to a set surge margin ($CAC_1$). It does this from a calculation of the turbine inlet temperature at this point of set surge margin. If this temperature exceeds the maximum allowable turbine inlet temperature ($Tt5\ MAX$), then the fuel flow calculation is limited to the value which gives $Tt5\ MAX$.

The values of maximum fuel flow calculated in this manner are divided by burner pressure $Pt4$ to obtain the schedule necessary for the control.

Figure 33 shows the FORTRAN flow diagram for this routine.

FUNCTION $SPrLU$

This routine provides a specialized multi-use table look-up routine for use in DSL/90 simulation. The routine is designed to handle data tabulated at constant increments of each independent parameter. It functions as a general univariant and bivariant table look-up for curves tabulated at constant increments and as a special purpose routine reading multiple tables to obtain a single answer.

The special features read several tables and compute a single value as an output. This provision was made to allow the compressor map presentations to be entered with a single statement although three of a set of four tables are actually required to read the map. Provision was also made to make a similar reading of a thrust table possible. These special features are used by entering key numbers in the first location of the table array to be used. For the thrust table the key number is zero (0.0), for the compressor map a one (1.0).

Figures 34, 35, 36, and 37 show the use of the various forms of the function $SPrLU$ with information on the diagram representation, the coding usage and the method for entering data.

SUBROUTINE $SPtURB$

This routine provides a simulation model for low and high pressure turbines. The characteristics of the turbine being simulated are entered as table names in the argument list of the routine. Also required are data such as incoming airflow ($W_{ein}$), cooling airflow ($W_{Tc}$), temperature of cooling flow ($T_{t4}$), uncooled turbine inlet temperature ($T_{tin}$), inlet total pressure ($P_{tin}$), discharge total pressure ($P_{tout}$) and rotor speed ($N$). Outputs of the routine are total gas flow into turbine ($WEBI$), discharge temperature ($T_{tOut}$), change in enthalpy ($\Delta h$), total gas flow.
out of turbine ($W_{EB}$) and the ratio of specific heats ($\gamma$).

The second (-02) version of this routine has one additional output, the cooled inlet temperature ($T_{lb}$).

Figure 38 shows the FORTRAN flow diagram of this routine.

FUNCTION SARECT

This function uses the rectangular method for integration and has various options; normal integration, holding at limit values with immediate change when the derivative changes sign, and resetting to a preset value.

Figure 39 shows the various tests that are made.

FUNCTION SPIMCV

This function provides a simplified and accurate representation of a time constant function and does not require the use of a pure integration such as the real pole function in the DSL/90 program.

Figure 40 shows the derived equation in block diagram form.
Figure 33. Function SPTACL
**Example:**

\[ Y = f(X) \]

\[
\begin{align*}
X_1 &= 1.1 & Y_1 &= 4.0 \\
X_2 &= 1.2 & Y_2 &= 4.2 \\
X_3 &= 1.3 & Y_3 &= 4.8 \\
X_4 &= 1.4 & Y_4 &= 4.8 \\
X_5 &= 1.5 & Y_5 &= 4.6
\end{align*}
\]

**Table Input Array:**

<table>
<thead>
<tr>
<th>PR3T02(1)</th>
<th>2.0</th>
<th>denote two dimensional table</th>
</tr>
</thead>
<tbody>
<tr>
<td>(2)</td>
<td>1.1</td>
<td>minimum value of independent variable</td>
</tr>
<tr>
<td>(3)</td>
<td>1.5</td>
<td>maximum value of independent variable</td>
</tr>
<tr>
<td>(4)</td>
<td>.1</td>
<td>constant increment of independent variable</td>
</tr>
<tr>
<td>(5)</td>
<td>4.0</td>
<td>( Y_1 )</td>
</tr>
<tr>
<td>(6)</td>
<td>4.2</td>
<td>( Y_2 )</td>
</tr>
<tr>
<td>(7)</td>
<td>4.8</td>
<td>( Y_3 )</td>
</tr>
<tr>
<td>(8)</td>
<td>4.8</td>
<td>( Y_4 )</td>
</tr>
<tr>
<td>(9)</td>
<td>4.6</td>
<td>( Y_5 )</td>
</tr>
</tbody>
</table>

\( \text{Figure 34. Constant Increment Table Look-Up} \)
GENERAL

THREE DIMENSIONAL

\[
f(GM3T01, Tt4, f/a)
\]

CODING: \(SPTLU(GM3T01, T03004, QP3004)\)

example:

\[Z = f(X, Y)\]

<table>
<thead>
<tr>
<th>(X)</th>
<th>(1.2)</th>
<th>(1.4)</th>
<th>(1.6)</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.0</td>
<td>•1</td>
<td>•2</td>
<td>•4</td>
</tr>
<tr>
<td>3.5</td>
<td>•2</td>
<td>•3</td>
<td>•6</td>
</tr>
<tr>
<td>4.0</td>
<td>•3</td>
<td>•4</td>
<td>•8</td>
</tr>
<tr>
<td>4.5</td>
<td>•4</td>
<td>•5</td>
<td>1.0</td>
</tr>
</tbody>
</table>

TABULAR INPUT ARRAY:

- **GM3T01(1)**: 3.0, DENOTE THREE DIMENSIONAL TABLE
- **GM3T01(2)**: 3.0, MINIMUM VALUE OF INDEPENDENT VARIABLE X
- **GM3T01(3)**: 4.5, MAXIMUM VALUE OF INDEPENDENT VARIABLE X
- **GM3T01(4)**: 0.5, INCREMENT VALUE OF INDEPENDENT VARIABLE X
- **GM3T01(5)**: 1.2, MINIMUM VALUE OF INDEPENDENT VARIABLE Y
- **GM3T01(6)**: 1.6, MAXIMUM VALUE OF INDEPENDENT VARIABLE Y
- **GM3T01(7)**: 0.2, INCREMENT VALUE OF INDEPENDENT VARIABLE Y

LIST OF Z FOR \(Y_1\):

- **GM3T01(8)**: •1, \(Z_{1.1}\) at \(X_1\)
- **GM3T01(9)**: •2, \(Z_{2.1}\) at \(X_2\)
- **GM3T01(10)**: •3, \(Z_{3.1}\) at \(X_3\)
- **GM3T01(11)**: •4, \(Z_{4.1}\) at \(X_3\)

LIST OF Z FOR \(Y_2\):

- **GM3T01(12)**: •2, \(Z_{1.2}\) at \(X_1\)
- **GM3T01(13)**: •3, \(Z_{2.2}\) at \(X_2\)
- **GM3T01(14)**: •4, \(Z_{3.2}\) at \(X_3\)
- **GM3T01(15)**: •5, \(Z_{4.2}\) at \(X_4\)

LIST OF Z FOR \(Y_3\):

- •
- •
- •
- •

Figure 35. Constant Increment Table Look-Up

67
SPF.BIAL
THRUST TABLE LOOK UP

PT9E
PT9E
PAMB

f(FØ3TO1, PT9E, PAMB, 9E) THPE

FØ3001 = SPTLU(FØ3TO1, PR3009, GM3009)

METHOD:

CONTROL INFORMATION AND THREE INDEPENDENT TABLES ARE ENTERED IN ONE ARRAY

TABLE

P1
P2

f(TABLE, P1, P2)
X

IF P1 < TABLE(4) X = f(TABLE1, loge P1, P2) * f(TABLE2, P1)
P1 ≥ TABLE(4) X = f(TABLE2, loge P1, P2) * f(TABLE3, P1)

TABULAR INPUT ARRAY

TABLE(1) 0.0 DENOTE SPECIAL TABLE LOOK-UP
(2) 101 1st LOCATION OF TABLE 2 IN ARRAY (INTEGER)
(3) 141 1st LOCATION OF TABLE 3 IN ARRAY (INTEGER)
(4) VALUE SEPARATING TABLE 2 AND 3

TABLE(5) 1st LOCATION OF THREE DIMENSIONAL TABLE 1
... (INPUT SAME AS DESCRIBED UNDER GENERAL TABLE LOOK-UP)
... ...

TABLE(101) 1st LOCATION OF TWO DIMENSIONAL TABLE 2
... ...

TABLE(141) 1st LOCATION OF TWO DIMENSIONAL TABLE 3
... ...

Figure 36. Constant Increment Table Look-Up
SPECIAL SINGLE VALUE FROM FOUR TABLES

CODYNG: WA3001 = SPTLU(WA3T01, N3001, PR3001)

METHOD:

CONTROL INFORMATION AND FOUR INDEPENDENT TABLES ARE ENTERED IN ONE ARRAY

TABLE

\[
S_1 = f(\text{TABLE}^1, P)
\]

IF \( P_2 < S_1 \):

\[
X = f(\text{TABLE}^2, P_1) + f(\text{TABLE}^3, S_1 - P_2, P_1)
\]

IF \( P_2 \geq S_1 \):

\[
X = f(\text{TABLE}^2, P_1) + f(\text{TABLE}^4, P_2 - S_1, P_1)
\]

TABULAR INPUT ARRAY

TABLE (1) 1.0 DENOTE SPECIAL TABLE LOOK UP

(2) 51 1st LOCATION OF TABLE 2 IN ARRAY (INTEGER)

(3) 1st LOCATION OF TABLE 3 IN ARRAY (INTEGER)

(4) 1st LOCATION OF TABLE 4 IN ARRAY (INTEGER)

TABLE (5) 1st LOCATION OF TWO DIMENSIONAL TABLE 1

INPUT SAME AS DESCRIBED UNDER GENERAL TABLE LOOK-UP

TABLE (51) 1st LOCATION OF THREE DIMENSIONAL TABLE 2

Figure 37. Constant Increment Table Look-Up
FUNCTION SPTURB

Figure 38. Function SPTURB
FUNCTION SPTURB

\[ f(M, \text{Wein}, Wt, f/a, Tt, Ttin, Ptin, Pout, N, TCT, WGT, ETAT) \]

SPTURB

SPTURB 1-02

Figure 38. Function SPTURB (Continued)
Figure 38. Function SPTURB (Continued)
Figure 38. Function SPTURB (Continued)
Figure 38. Function SPTURB (Concluded)
<table>
<thead>
<tr>
<th>GENERAL FORM</th>
<th>FUNCTION</th>
</tr>
</thead>
</table>
| $X = \text{SARECT}(x_{IC}, \dot{x}, x_L, x_u, p, x_{RESET})$ | $p = 1.0$  
$X_L \leq \int_0^t \dot{x} \, dt + x_{IC} \leq x_u$  
$p = 0.0$  
$X = x_{RESET}$ |

**Figure 39. Function SARECT-Mode-Controlled Rectangular Integrator**
<table>
<thead>
<tr>
<th>GENERAL FORM</th>
<th>FUNCTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>$y = SPTMCV(\tau, x)$</td>
<td>$\tau \dot{y} + y = x$</td>
</tr>
<tr>
<td>GENERAL TIME CONSTANT FUNCTION</td>
<td>EQUIVALENT LAPLACE TRANSFORM $\frac{1}{\tau s + 1}$</td>
</tr>
</tbody>
</table>

**Figure 40. Function SPTMCV-General Time Constant Function**
Section V
INITIALIZATION

GENERAL REQUIREMENTS

The simulation of a system experiencing a transient, of necessity begins at a steady-state or quasi-steady-state point. The driving forces, that must be zero to maintain this point, are, in the case of the propulsion system, usually the difference between two large numbers. This effectively means that the initial values which enter into the calculation of the driving forces must be accurate and identical in method of calculation to those within the simulation. The simulation should be stable enough when initialized so that if no transient is introduced, it will maintain its steady-state point.

One method used to initialize a simulation is to provide initial values from steady-state calculations and then run the dynamic program for some time in order to stabilize at the steady-state condition prior to the introduction of the transient. This method is less than satisfactory for several reasons. First, it requires too much preparatory work in that either steady-state programs or hand calculations would have to be executed to provide the initial data. As stated above, accuracy is important which means that the method used for these calculations would have to be compatible to that used in the simulation. Since the simulation logic itself is already available for these calculations its use insures compatibility. Secondly, it does not always work. When initial values are not exact, the start-up transient could drive the simulation into instability.

At the start of this program the ground rule was established that the only initial values to be required were air vehicle Mach number, ambient pressure and temperature, air vehicle angle of attack and yaw, and the position of the power lever. This ground rule is illustrated by the diagram on figure 41.

IMPLEMENTATION

With the ground rule for initialization established several things had to be considered. The most obvious way to insure accuracy and compatibility in computing initial values is to simply use the simulation logic calculations. This would be most efficiently done if sections of the simulation could be executed under the control of an initialization routine. DSL/90 provides a method to do this but unfortunately requires that an invaluable feature of the language be compromised to use it. To explain this anomaly the nature of the DSL/90 system must be lightly touched on.
Figure 41. Parameters Required for Initialization
The power of a language such as DSL/90 is its ability to accept the simulation logic equations in any order whatsoever and sort them in such a manner that variables are always available when needed. To accomplish this, the language must prevent the use of logic which transfers the program control from one area in the program to another since after sorting the effect would not likely be the desired one. The language does however, provide a procedure called NOSORT, which allows the insertion of transfer instructions into the simulation. At first this seems to provide the answer to executing sections of the program under control of an initialization program except that a second look reveals that insertion of a NOSORT block separates the sorting procedure such that the statements (equations) ahead of the NOSORT block do not get sorted into the statements after the NOSORT block. This effectively means that the program must be run through the translation phase which sorts the simulation logic then the original simulation logic rearranged by hand according to the sorted simulation and then the transfer information inserted in NOSORT blocks in this sorted deck. Since this seems to, at least partially, negate one of the very desirable features of DSL/90 another path seemed advisable.

The basic reason, other than execution time which is negligible, to execute the simulation logic in sections while initializing is the potential for errors which the computer finds unforgiveable such as taking square roots of negative numbers, raising negative numbers to fractional powers, etc. These problems can be overcome in the checkout phase of the initialization by careful choice of initial values of key parameters. In accordance with the ground rule on required inputs stated above, initial values are calculated, not entered as input.

SAMPLE PROGRAM

A sample initialization program used for the initialization of a turbojet engine with a simple integrated propulsion system control and a started inlet phase simulation is presented in Appendix II. This initialization has been checked out only at one power setting at one altitude - Mach condition.
Appendix I

LIST OF SUPPORTING SUBROUTINES
TABLE I. SAACT

$1BFTC SAACT
SUBROUTINE SAACT (K,XC,X,CCT,XDOT,XIC,XH)
COMMON/KEYS/NALARM,SKIP(17),KSIM/HMAX/H,KEEP/MEMRY/S(3)
DIMENSION CCT(1)
L=K
XLIM=AMIN1(CCT(1),AMAX1(XC+0.0))
IF(S(L+2)-KSIM) 100*200*100
100 IF(KEEP) 110+120+110
110 S(L+2)=KSIM
XIC=XLIM
120 XH=HSTRSS(L,X,CCT(6),CCT(7),X)
DXDT=CCT(2)*(XLIM-(X*(1.-CCT(5))+CCT(5)*XH))
XDOT=AMIN1(CCT(4),AMAX1(DXDT*CCT(3)))
RETURN
END
TABLE II. SADSPA

SIBFTC SADSPA

FUNCTION SADSPA(X, XL, XR)
SADSPA = 0.0
IF (X < XL) SADSPA = -1.0
IF (X > XR) SADSPA = 1.0
RETURN
END

SADSPA00
SADSPA02
SADSPA03
SADSPA04
SADSPA05
SADSPA06
SADSPA07
TABLE III. SALIMT

$IBFTC SALIMT
     FUNCTION SALIMT(X, XL, XU, TYPE)
     C TYPE INDICATES LIMIT PROCEDURE DESIRED
     C 0 - MIN PRIORITY  1 - MAX PRIORITY  -1 - IGNORE LIMITS
     C
     IF (TYPE.LT.0.0) GO TO 400
     IF (XL.GT.XU) GO TO 100
     SALIMT = AMAX1(AMIN1(X, XU), XL)
     RETURN
     100 IF (TYPE.EQ.0.0) GO TO 300
     SALIMT = AMIN1(X, XU)
     RETURN
     300 SALIMT = AMAX1(X, XL)
     RETURN
     400 SALIMT = X
     RETURN
     END
TABLE IV. SAMOIN

FUNCTION SAMOIN (K,DXDT,XIN,XL,XU,PATH,XIC,DXDTM)
COMMON/CURVAL/TIME
COMMON/KEYS/NALARM,SKIP(17),KSIM/HMAX/H,KEEP
COMMON/MEMRY/S(15)
EQUIVALENCE (KN(1),S(1))
DIMENSION KN(1)
I=K
L=1
XDOT=DXDT
X=XIN
TYPE=PATH
IF(K(NL+14)=KSIM) 100,200,100
100 IF(KFFP) 110,120,110
110 KN(L+14)=KSIM
S(L)=TIME
S(L+1)=TYPE
S(L+12)=X
120 IF(S(L+1)) 960,950,960
200 DO 300 JJ=1,5
210 IF(TIME>S(L)) 300,210,310
300 L=L+2
310 IF(K(KEEP)) 400,700,400
400 LL=L+12
DO 500 JJ=1,5
500 S(LL+2)=S(LL-2)
600 S(L)=TIME
S(L+1)=TYPE
700 IF(TYPE) 900,710,900
710 IF(S(L+3)) 720,800,720
720 S(L+12)=XIC
730 XDOT=DXDTM
GO TO 750
740 XDOT=-DXDTM
750 S(L+13)=XDOT
GO TO 960
800 IF(X=S(I+12)) 810,950,820
810 IF(S(I+13)) 840,950,830
820 IF(S(I+13)) 830,950,840
830 XDOT=S(I+13)
GO TO 960
840 X=S(I+12)
GO TO 950
900 IF(XDOT) 910,960,930
910 IF(X-XL) 920,950,960
920 X=XL
GO TO 950
930 IF(X-XU) 960,950,940
940 X=XU
950 XDOT=0.0
XIN=X
960 SAMOIN=XDOT
1000 RETURN
END
TABLE V. SASWCH

S1BFTC SASWCH

FUNCTION SAOFON(I,X, XL, XU)
COMMON/MEMRY/C(I)
SAOFON = 1.0
IF(X<XL OR X<XL AND C(I)<LT+0)SAOFON = 0.0
GO TO 100
ENTRY SAONOF(I,X,XL,XU)
SAOFON = 0.0
IF(X<XL OR X<XL AND C(I) NE 0)SAOFON = 1.0
100   C(I) = SAOFON
RETURN
END
TABLE VI. SAWFAT

SUBROUTINE AUTONETICS WF ATTENUATION
INTEGER FUNCTION SAWFAT(TSENS,PRSENS,WSENS,NCORR,PRT,WAT,CN,TACC)
C
C SIMULATION OF WF/P4 ATTENUATION CONTROL LOOP
C
REAL NCORR
DIMENSION CN(2)
SAWFAT = 1
IF(TSENS.GE.CN(2))GO TO 200
IF(NCORR.GE.CN(1))GO TO 100
IF(PRSENS.GE.SLTLUI(PRT,NCORR,TACC))GO TO 200
RETURN
100 IF(SLTLUI(WAT,NCORR,TACC).GE.WSENS)GO TO 200
RETURN
200 SAWFAT = 0
RETURN
END
### TABLE VII. SLFVPG

**SIBFTC SLFVPG**

**FLOW PARAMETER AS A FUNCTION OF PRESSURE RATIO AND GAMMA**

**FUNCTION SLFVPG(PRATIO,GAMMA)**

**C**

G = 32.174049  
R = 53.34991

**GM = GAMMA**

**PR = PRATIO**

IF(PR.LE.1.0)GO TO 10

**C**

**COMPUTE CRITICAL CONDITIONS FOR MAXIMUM VALUE**

CTR = 1.0 + (GM - 1.0)/2.0

TR = PR ** (1(GM-1.0)/GM)

IF(TR.GT.CTR)TR=CTR

**C**

**COMPUTE FLOW PARAMETER**

FP = SQRT(1.20615195*GM*(TR-1.)/GM/1.)/(GM-1.)/TR**((GM+1.)/2.)/(GM-1.))

SLFVPG = FP

RETURN

10 FP = 0.0

GO TO 20

END
TABLE VIII. SLGAMF

$IBFTC$ SLGAMF
C GAMMA AS A FUNCTION OF TOTAL TEMP AND FUEL-AIR RATIO
FUNCTION SLGAM(TT,FARI)
DIMENSION C(5)
T = TT
FAR = FARI
CALL ARGO(N)
IF(N.LT.2)FAR = 0.0
IF(T.EQ.0.0)GO TO 1000
POWFR = 5526.0 / T
CPAIR = 23996 + 068558*((POWER/(EXP(POWER)-1.0))**2*EXP(POWER))
CPFUEL = 12149
DO 10 1=1,5
10 CPFUEL = CPFUEL + C(1) * T**1
CPGAS = (CPAIR + FAR * CPFUEL) / (1.0 + FAR)
RGAS = 068558 + 000835 * FAR / (1.0 + FAR)
SLGAM = CPGAS / (CPGAS - RGAS)
RETURN
1000 SLGAM = 1.4
RETURN
END
TABLE IX. SLMVFG

$SFRTC SLMVFG

FUNCTION SLMVFG (LM*MXI*FLOWP*GAMMA)
COMMON/MEMRY/C(1)
C MACH NO. AS FUNCTION OF FLOW PARAMETER AND GAMMA
C G=32.174049
C R=5.34991
C F=FLOWP
IF(FP) 100*100*110
100 XM=0.0
GO TO 510
110 GAM=GAMMA
C1=1.01/1.01
C2=(GAM+1.0)/2.0/(GAM-1.0)
Z=FP/SORT(GAM*603075975)
C COMPUTE MAXIMUM VALUE
ZMAX=1.0/(1.0+C1)**C2
IF(Z*LT.ZMAX) GO TO 200
XM=1.0
GO TO 510
200 L=LM
IF(C(L)) 220*220*210
210 XM=C(L)
GO TO 320
220 XM=MXI
GO TO 320
300 XM=XM*C3*(Z*C4-XM)/(XMS-1.0)
IF(XM) 310*310*320
310 XM=2
320 XMS=XM**2
C3=1.0+C1*XMS
C4=C3**C2
ZM=XM/C4
IF(ABS(Z-ZM)*GT. *0000001) GO TO 300
500 C(L)=XM
510 SLMVFG=XM
RETURN
END
TABLE X. SLTLU

SUBFC SLTLU

TABLE LOOK-UP CONTROL PROGRAM

FUNCTION SLTLU (LM*XY+YW)

COMMON/CURVAL/A(I)
COMMON/LCURVE/LOC(A(1))
DATA N/0/
IPASS=0

L=LM
IF(L) 60+60+90
60 IF(N) 80+70+80
70 N=LOC(A(I))=LOC(A(I))
JL=LOC(A(I))
LOC(A(I))=LOC(A(I))+1+N
DO 77 J=3+JL
72 LOC(A(J-1))=LOC(A(J-1)+LOC(A(J-2))
80 L=-L
L=LOC(A(L))
90 N=A(L+1)
IX=A(L+2)
IF(A(L)-3+0) 100+270+200
100 IF(A(L)-1+0) 110+110+120
110 Z=A(L+1)
GO TO 400
120 NY=0
IV=0

NY=N
LX=L+3
LY=LX+NY
GO TO 390
200 NW=NX
L1=L+3
L2=L+1+NW
L=L2
NX=A(L+1)
NY=A(L+3)
IF(W-A(L)-1) 280+280+210
210 DO 230 LW=L+L2
IF(W-A(LW)) 240+250+220
220 L=L+NX+NY+4+NX+NY
NX=A(L+1)
NY=A(L+3)
230 CONTINUE
GO TO 280
240 IFASS=1
IPASS=(W-A(LW-1)/(A(LW)-A(LW-1)))
GO TO 280
250 L=L+NX+NY+4+NX+NY
260 NY=A(L+1)
270 NY=A(L+3)
280 LX=L+5
LY=LX+NX

91
TABLE X. SLTLU (CONTINUED)

LZ=LY+NY
IY=A(L+4)
IX=A(L+7)
290 IF(IY-2) 300,300,320
300 IF(IY-2) 310,310,320
310 CALL SLTLU2 (A(LX),A(LY),A(LZ),NX,NY,X,Y,Z)
     GO TO 330
320 CALL SLTLU3 (A(LX),A(LY),A(LZ),NX,NY,X,Y,X,Y,Z)
330 IF(IPASS) 350,400,340
340 IPASS=-1
     W1=Z
     L=LZ-1+NX*NX
     GO TO 260
350 Z=W1+RATW*(Z-W1)
400 SLTLU=Z
500 RETURN
END
TABLE XI. SLTLU2

SLBFTC SLTLU2
C LINFOR INTERPOLATION FOR THREE DIMENSIONAL TABLE
SUBROUTINE SLTLU2 (AX,AY,AZ,NX,NY,X,Y,Z)
DIMENSION
  1 AX(1),AY(1),AZ(1)
  IF (X-AX(1)) IFIX 10,10,20
10 JX=1
  GO TO 40
20 DO 30 I=2,NX
  JX=I
  IFIX=AX(I)
50,40,30
30 CONTINUE
40 RATX=0.0
  GO TO 60
50 RATX=(AX(JX)-X)/(AX(JX)-AZ(JX-1))
60 IF (NY) 70,70,80
70 Z=AZ(JX)-RATX*(AZ(JX)-AZ(JX-1))
  GO TO 200
80 IF (Y-AY(JY)) 90,90,100
90 JY=1
  GO TO 120
100 DO 110 J=2,NY
  JY=J
  IF (Y-AY(JY)) 130,120,110
110 CONTINUE
120 RATY=0.0
  GO TO 140
130 RATY=(AY(JY)-Y)/(AY(JY)-AY(JY-1))
140 JZ=NX*(JY-1)
  ZZ=AZ(JZ)-RATX*(AZ(JZ)-AZ(JZ-1))
  JZ=JZ-NX
  Z1= AZ(JZ)-RATX*(AZ(JZ)-AZ(JZ-1))
  Z=ZZ-RATY*(ZZ-Z1)
200 RETURN
END
**TABLE XII. SLTLU3**

```plaintext
**LAGRANGE INTERPOLATION FORMULA FOR THREE DIMENSIONAL TABLE**

**SUBROUTINE SLTLU3** (AX,AY,AZ,NX,NY,IX,IY,X,Y,Z)

**DIMENSION**

1 AX(I) *AY(I) *AZ(I) *YY(10) *C(10)

CALL SLTLU4(AX,NX,IX,NX,N2)
CALL SLTLU4(AY,NY,IY,NY,N2)
IF(N2) 50 *10 *50
10 IF(M2) 30 *20 *30
20 JZ=NI+NX*(M1-1)
   Z=AZ(JZ)
   GO TO 200
30 JY=NI+NX*(M1-2)
   L=0
   DO 40 J=M1,M2
      L=L+1
      JY=JY+NX
      YY(L)=AZ(JY)
   40 CONTINUE
   GO TO 130
50 P=1.0
   K=0
   DO 80 J=NI,N2
      K=K+1
      C(K)=1.0
      P=P*(X-AX(J))
   80 CONTINUE
   DO 110 I=NI,N2
      IF(I-J) 70 *80 *70
70 C(K)=C(K)/(AX(I)-AX(J))
   80 CONTINUE
   IF(M2) 100 *90 *100
90 M2=M1
100 L=0
   DO 110 I=M1,M2
      L=L+1
      YY(L)=0.0
      JZ=NI+1+NX*(I-1)
      K=0
   110 CONTINUE
      YY(L)=YY(L)*P*AZ(JZ)/(X-AX(J))*C(K)
110 CONTINUE
   IF(M1-M2) 130 *120 *130
120 Z=YY(1)
   GO TO 200
130 P=1.0
   L=0
   DO 150 I=M1,M2
      L=L+1
      C(L)=1.0
      P=P*(Y-AY(I))
   150 CONTINUE
   IF(I-J) 140 *150 *140
140 C(L)=C(L)/(AY(J)-AY(I))
   150 CONTINUE
```

LAG30000
LAG30000
LAG30010
LAG30020
LAG30030
LAG30040
LAG30050
LAG30060
LAG30070
LAG30080
LAG30090
LAG31000
LAG31010
LAG31020
LAG31022
LAG31030
LAG31040
LAG31050
LAG31060
LAG31070
LAG31072
LAG31082
LAG31090
LAG32000
LAG32010
LAG32020
LAG32023
LAG32030
LAG32040
LAG32050
LAG32060
LAG32070
LAG32072
LAG32074
LAG32080
LAG32090
LAG32092
LAG33000
LAG33002
LAG33010
LAG33020
LAG33030
LAG33040
LAG33049
LAG33050
LAG33050
LAG33060
LAG33062
LAG33070
LAG33072
LAG33080
LAG33090
LAG34000
LAG34010
LAG34020
LAG34030
Z = 0.0
L = 0
DO 160 J=M1+M2
   L = L + 1
   Z = Z + P*YY(L)/(Y-AY(J))*C(L)
160 CONTINUE
200 RETURN
END
TABLE XIII. SLTLU4

$SIBFTC SLTLU4$

C LOCATE RANGE OF POINTS FOR LAGRANGE INTERPOLATION

SUBROUTINE SLTLU4 (AX, N, X, N1, N2)

DIMENSION

1 AX(1)
N2=0

IF(N-L) 220*220*10
10 IF(X-AX(1)) 30, 30, 40
30 N1=1
GO TO 300

40 DO 710 J=2, N
22 IF(X-AX(J)) 60, 30, 210
50 N1=J
GO TO 300

60 JJ=L-1
K1=J-JJ

70 K1=1
GO TO 100

80 K3=J+L-2
IF(K3-N) 100, 100, 90

90 K2=N-JJ
GO TO 110

100 K2=J-1
110 RA=10000.
DO 190 K=K1, K2
KK=K+JJ
C1=X-AX(K)
C2=AX(KK)-X
IF(C1-C2) 140, 120, 130

120 N2=KK
GO TO 200

130 RA=1.0-C2/C1
GO TO 150

140 RA=1.0-C1/C2
150 IF(RB-RA) 190, 160, 170
160 IF(J-L/2-K) 190, 180, 180

170 RB=RA
180 N2=KK
190 CONTINUE
200 N1=N2-JJ
GO TO 300

210 CONTINUE
N1=N
GO TO 300

220 N2=N
N1=1
300 RETURN
END
TABLE XIV. SPCOMP

SUBROUTINE SPCOMP(M,PTI,PTO,TTI,N,CCT,WAT,ETAT,PRST,WAST,REYT,SRGM)

1. ETA, TTO, DH, W, GMA

COMMON/CURVAL/TIME
COMMON/KEYS/NALARM, SKIP(1), KSIM, HMAX, H, KEEP, MEMORY/S

REAL N, NCN
DIMENSION CCT(1), KN(1)

L = M

DE = PTI/14.696
RTH = SQRT(TTI/518.691)
NCN = N/(RTH*CCT(1))
PRATIO = PTO/PTI
PR = CCT(2)*((PRATIO+1.0)+1.0)
WCN = SPTLU(WAT, NCN, PR)

IF(KN(L+39) = KSIM) 120, 130, 130

100 SRGM = S(L+38)

IF(SRGM = 1.0) DSRGM = 110, 130, 130

110 DET = CCT(8)

DSRGM = CCT(9)
DWCN = SPTMCVIL + 0.05*CCT(5)*WCN
GO TO 200

120 KN(L+39) = KSIM

130 DET = 1.0

DSRGM = 0.0
DWCN = 0.0

IF(KN(L+18) = KSIM)

S(L) = TIME
S(L+1) = 0.0
S(L+2) = 0.0

200 S(L+38) = WCN*((SPTLU(PRST, NCN) - 1.0)/CCT(2) + 1.0)/PRATIO/(SPTLU(WAT, WC))/CCT(1))

WC = CCT(6)*(WCN + DWCN)

DET = SPTLU(REYT, CCT(7)*WC) - SPTLU(REYT, (CCT(7)*WC*DE/(RTH**2))

W = WC*(1.0 + DET)/RTH

ETA = DET*(CCT(3)*SPTLU(ETAT, WCN, PR) + CCT(4))*DET

GM = GMA

IF(GM) 300, 300, 310

300 GM = 1.4

310 DT = (PRATIO**((GM-1.0)/GM) - 1.0)*TTI/ETA

DH = DT*0.6854*GM/(GM-1.0)
TTOCAL = TTI + DT
TT = SPTMCV(L+19)*0.05, TTOCAL

GMA = SLGAM(TTI + TTOCAL)
RETURN
END
TABLE XV. SPMEMF

$1BFCT SPMEMF (K+XIC+X)
COMMON/CURLVAL/TIME
COMMON/KEYS/NALARM,SKIP(17),KSIM/HMAX/H,KEEP/MEMRY/S(13)
EQUIVALENCE (KN(1),S(1))
DIMENSION KN(1)
I=K
L=I
IF(KN(L+12)-KSIM) 100*200*100
100 IF(KEEP) 110*120*110
100 KN(L+12)=KSIM
120 OUTPUT*XIC
    IF(KEEP) 800*900*800
200 DO 220 J=1,5
    IF(TIME=S(L)) 220,210,300
210 OUTPUT=S(L+3)
    IF(KEEP) 800*900*800
220 L=L+2
300 OUTPUT=S(L+1)
700 IF(KEEP) 800*900*720
720 LL=I+12
    DO 730 JJ=1,5
    LL=LL-2
    S(LL+1)=S(LL-2)
    S(LL+2)=S(LL-1)
730 S(LL+3)=S(LL)
800 S(L)=TIME
    S(L+1)=X
900 SPMEMF*OUTPUT
RETURN
END
TABLE XVI. SPTACL

```
$IBFTC SPTACL

FUNCTION SPTACL(L*SRGM,TT4,PT4,W4,TT5,TT5B,TT5S,TT5MAX,WFE,WFS,
  FTAR,CAC1,DTVFAT)

  WFS = WFE
  IF(WFS.EQ.0.0)RETURN
  N=0
  TT5S = AMIN1 ((SRGM/CAC1)**2-1.0)*TT5B + TT5 + TT5MAX )

100 TT5T = SPTLU(DTVFAT,WFS/W4,TT4)*ETAB + TT4
  DT5S = TT5S - TT5T
  IF(ABS(DT5S).LE.0.10)GO TO 102
  N=N+1
  IF(N.GT.20)WRITE(6,101)N,TT5S,TT5T,DT5S,WFS,WFF

101 FORMAT(12H SPTACL, N=,13, 2X,5HTT5S=F10.3, 2X,5HTT5T=F10.3
  $ 2X,5HD5S=F10.5, 2X,4HWFS=F10.5, 2X,4HWFE=F10.5)
  IF(N.GT.25)GO TO 102
  WFS = WFS*(1.0+10.0/DT5S/1500.0))
  GO TO 100

102 SPTACL = SPTMCV(L,0.005,WFS/PT4)

105 RETURN

END
```
TABLE XVII. SPTLU

SUBFC SPTLU

FUNCTION SPTLU(LIN+XIN+YIN)

C TABLE LOOK-UP FOR PRATT AND WHITNEY TABLE (CONTROL)

COMMON/CURVAL/C(1)
COMMON/PCURVE/LOCA(1)
DIMENSION KC(1)
EQUIVALENCE (C,KC)
DATA N/0/ L=LIN X=XIN Y=YIN

IF(L) 100 100 140
100 IF(N) 130 110 130
110 N=LOC(LOCA(1))=LOC(C(1))
JL=LOC(A(1))
LOCA(1)=LOC(A(1))+1+N
DO 120 J=3,JL
120 LOCA(J)=LOCA(J-1)+LOCA(J-2)
130 L=-L
L=LOCA(L)
140 IF(ABS(C(L))=1.0) 200 300 400
200 IF(X=C(L+3)) 210 220 220
210 L=L+1
GO TO 230
220 L=L+2
230 L2=KC(L1)+L-1
ANS=SPTLU1(L+4,XLOG(X)-Y)*SPTLU1(L2,X)
GO TO 500
300 DPR=Y-SPTLU1(L+4,X)
IF(DPR) 310 320 320
310 DPR=-DPR
L=L+2
GO TO 330
320 L=L+3
330 L2=KC(L1)+L-1
L3=KC(L1)+L-1
ANS=SPTLU1(L3,X)+C(L)*SPTLU1(L2,DPR,X)
GO TO 500
400 ANS=SPTLU1(L,X,Y)
500 SPTLU=ANS
RETURN
END
**TABLE XVIII. SPTLUI**

```
SIBFTC SPTLUI

FUNCTION SPTLUI(LOC,XIN,YIN)

COMMON/CURVAL/C(1)

DIMENSION XY(2)

L=LOC

X=XIN

IF(X=C(L+1)) 110,110,120

C

110 X=C(L+1)

GO TO 140

120 IF(X=C(L+2)) 140,130,130

C

130 X=C(L+2)

140 N1=(X-C(L+1))/C(L+3)

C1=N1

RATX=(X-C(L+1))/C(L+3)-C1

200 IF(C(L)-2*0) 210,210,220

210 LN=L+N1+4

ANS=C(LN)+RATX*(C(LN+1)-C(LN))

GO TO 300

C

THREE DIMENSIONAL TABLE SECTION

220 Y=YIN

M=(C(L+5)-C(L+4))/C(L+6)+1.01

IF(Y=C(L+4)) 230,230,240

C

230 Y=C(L+4)

GO TO 260

240 IF(C(L+5)-Y) 250,250,260

250 Y=C(L+5)

260 M1=(Y-C(L+4))/C(L+6)

D1=M1

RATY=(Y-C(L+4))/C(L+6)-D1

N=(C(L+2)-C(L+1))/C(L+3)+1.01

LN=L+7+(N*M1)+N1

DO 280 K=1,2

XY(K)=C(LN)+RATX*(C(LN+1)-C(LN))

IF(RATY) 280,270,280

270 ANS=XY(1)

GO TO 300

280 LN=LN+N

ANS=XY(1)+RATY*(XY(2)-XY(1))

300 SPTLUI=ANS

RETURN

END
```
TABLE XIX. SPTURB

SUBROUTINE SPTURB(M,WE,WTC,FA,TT4,TT1,PT1,PTO,N,TCT,WGT,ETAT,TTB)

REAL N
DIMENSION TCT(1)
L=M
WEBI=WE+WTC
TTB=TT4*WTC+TT1*WE/WEBI
PRATIO=PTI/PTO
FP=SPTLU(WGT*PRATIO*(N/SORT(TTB)/TCT(1))/TCT(6))
WEB=FP*TPT1/((TCT(7)/SORT(TTB))
TTB=SPTMCV(L+0.05*0.833*TTB)+SPTMCV(L+19*TCT(8)/WEB+0.167*TTB)
GAM=WG
IF (GAM) 100,100,110

100 GMT=1.4
110 DELTT=PRATIO**((1.0-GMT)/GMT)-1.0
DHT=-DELTT*TTP*0.06854*GMT/(GMT-1.0)
ETA=SPTLU(ETAT,(TCT(2)*N/SORT(DHT))/PRATIO*TCT(5))
ETA=1.0-(1.0-ETA)/TCT(3)*FP*TPTI/(TTP**1.2)**0.08+TCT(4)
DHT=DHTI*ETA
TTI=TTI+ETAT
RETURN
END
TABLE XX. SARECT

$SIBFTC SARECT

FUNCTION SARECT (IXIC, DXDT, XL, XU, PATH, XRESET)
DIMENSION KN(1)
COMMON/CURVAL/TIME
COMMON/KEYS/NALARM, SKIP(17), KSIM/HMAX/H, KEEP /MEMRY/S(14)
EQUIVALENCE (KN(1), S(1))
L=1
IF(KN(1+13) .NE. KSM) GO TO 300
DO 220 J=1,3
IF (TIME .LE. S(J)) 220, 230, 600
220 L=L+4
230 SARECT=SIL+31
RETURN
300 IF (KEEP .EQ. 0) GO TO 320
KN(1+13)=KSM
320 CALL ARG0(KN(I+12))
IF (KN(I+12) .EQ. 7 .AND. PATH .LE. 0.0) GO TO 330
SARECT=IXIC
IF (SARECT .LT. 5) GO TO 340
IF (SARECT .GT. XU) SARECT=XU
IF (SARECT .LT. LX) SARECT=LX
GO TO 340
330 SARECT=XRESET
340 IF (KEEP .NE. 0) GO TO 890
RETURN
600 IF (KN(I+12) .LT. 7 .OR. PATH .NE. 0.0) GO TO 800
DXDT=0.0
SARECT=XRESET
GO TO 850
800 SARECT=(S(L+3)+S(L+2)*(TIME .LE. S(J))
IF (S(L+1) .EQ. 3 .OR. S(L+2) .EQ. 0.0) GO TO 850
IF (S(L+2) .GT. 0.0 .AND. SARECT .GT. XU) SARECT=XU
IF (S(L+2) .LT. 0.0 .AND. SARECT .LT. LX) SARECT=LX
850 IF (KEEP .EQ. 0) RETURN
IF (KEEP .LT. 0 .AND. J .GE. 3) GO TO 890
LL=I+12
DO 880 JJ=J,3
LL=LL-4
S(LL)=S(LL-4)
S(LL+1)=S(LL-3)
S(LL+2)=S(LL-2)
880 S(LL+3)=S(LL-1)
890 S(L)=TIME
S(L+1)=PATH
S(L+2)=DXDT
S(L+3)=SARECT
RETURN
END
TABLE XXI. SPTMCV

FUNCTION SPTMCV(K, TAU, TT)  
COMMON/CURVAL/TIME  
COMMON/KEYS/NALARM, SKIP(17), KSIM,HMAX,H,KEEP,MEMRY,S(10)  
EQUIVALENCE (KN(1),S(1))  
DIMENSION KN(1)  
I=K  
L=1  
IF(KN(L+9)-KSIM) 100,200,100  
100 IF(KEEP) 110,120,110  
110 KN(L+9)=KSIM  
120 OUTPUT=TT  
200 DO 220 J=1,3  
220 L=L+3  
GO TO 900  
300 DT=TIME-S(L)  
P=EXP(-DT/TAU)  
C1=TAU/DT*(1-P)  
OUTPUT=S(L+2)*P+TT*(1-C1)+S(L+1)*(C1-P)  
400 IF(KEEP) 410,900,420  
410 IF(J-3) 420,800,800  
420 LL=L+9  
DO 430 JJ=J,2  
LL=LL-3  
S(LL)=S(LL-3)  
S(LL+1)=S(LL-2)  
430 S(LL+2)=S(LL-1)  
800 S(L)=TIME  
S(L+1)=TT  
S(L+2)=OUTPUT  
900 SPTMCV=OUTPUT  
RETURN  
END
Appendix II

INITIALIZATION PROGRAM
Appendix II

INITIALIZATION PROGRAM

In keeping with the ground rules established for the propulsion system simulation an initialization or initial conditions (I/C) program was developed. Certain elements of the I/C program are general and the example shown in this appendix will suffice to illustrate the general form of such programs. Specific I/C programs must be tailored for the specific system being initialized.

Figure 1 shows the FORTRAN flow diagram for the STEADY program which acts as the controlling element in the I/C phase. The MOVE subroutine referred to in this diagram is a routine which locates the simulation program variables by name in the CURVAL storage areas and places an identifying subscript in the STEADY routine so reference to and from UPDATE, the DSL/90 created routine which actually contains the simulation logic, can be made. The MOVE routine is diagrammed on figure 2. The STEADY program computes initial values for engine and inlet parameters that must have values in order to allow UPDATE to be executed. After execution of the simulation logic (UPDATE) the calculated values are used to compute initial values for several inlet variables which are required for use in SLMASS which computes inlet initial conditions. Figure 3 shows the flow diagram for SLMASS.

After another pass through the UPDATE routine to establish engine face conditions, the SPJENG routine, diagrammed in figure 4, is called to compute the initial conditions which will bring the engine to a balanced condition. After balancing the engine control is returned to STEADY where final calculations are made and UPDATE is called several times to set all initial values. The repeated calling of the UPDATE routine is necessary due to the nature of certain DSL/90 functions such as HSTRSS which must be entered twice before the value appears as an output.

The program listings for the STEADY, MOVE, SLMASS and SPJENG are given in tables I, II, III, and IV. The print statements sprinkled through all three routines are meant for checkout only and in the final version will be removed.
Figure 1. STEADY Routine
Figure 1. STEADY Routine (Continued)
Figure 1. STEADY Routine (Concluded)
Figure 2. MOVE Routine
Figure 3. SLMASS Routine
Figure 3. SLMASS Routine (Continued)
Figure 3. SLMASS Routine (Concluded)
Figure 4. SPJENG Routine
Figure 4. SPJENG Routine (Continued)
Figure 4. SPJENG Routine (Concluded)
TABLE I. STEADY

SUBROUTINE STEADY
COMMON/CURVAL/C(2)/SYMBOLS/NOSTEP,GOSYMBS,SYMBOL(2)
COMMON/KEYS/NALARM,KPOINT,KPRINT,DELTG,KPLOT,KLAB,DELTG,KFINIS
1KRANGE,KLOCK,KINTYPE,KABS,KPT1,T,RLAST,ALAST,KTITLE,KSIM
DIMENSION
1 HOL(37) *LOC(37)
DIMENSION K(1)
DATA HOL(1) / 2220A08108X08110PO0000C6N004C6N007NO6222T1O105WA6003
$P01015MN811LMN8105A08120A08113WF6000NO8650A8611TO8608AO8602WA1O27
$CN1001CN1003PO1012A06009A08105WQ1967WQ1904CN1004MN0000TO0000CN86TO
3CN81TOA1022A01803NO8650N0602NO8603TO8602/
EQUIVALENCE
1(LOC ( 1), MAT)(LOC ( 2), LXX),(LOC ( 3), LPO)
2*(LOC ( 4), LP4),(LOC ( 5), LPM),(LOC ( 6), LPM2)
3*(LOC ( 7), LTX),(LOC ( 8), LW4),(LOC ( 9), LPX)
4*(LOC ( 10), LMT),(LOC ( 11), LTH),(LOC ( 12), L19)
5*(LOC ( 13), LBP),(LOC ( 14), LWF),(LOC ( 15), L00)
6*(LOC ( 16), LO1),(LOC ( 17), LAJ),(LOC ( 18), IA9)
7*(LOC ( 19), LW7),(LOC ( 20), LCI),(LOC ( 21), LC3)
8*(LOC ( 22), LP2),(LOC ( 23), LA9),(LOC ( 24), LAT)
9*(LOC ( 25), LWH),(LOC ( 26), LWD),(LOC ( 27), LC4)
A*(LOC ( 28), LMO),(LOC ( 29), LTO),(LOC ( 30), LNN)
B*(LOC ( 31), LN1),(LOC ( 32), LW1),(LOC ( 33), IAT)
D*(K(1)+C(1)) *(LOC ( 34), IN2),(LOC ( 35), JN2)
E*(LOC ( 36), IQ0),(LOC ( 37), IAJ)
DATA NAME/6HSTEADY/
IF (KLOCK) 30*20*30
20 C(1)=0.0
30 CONTINUE
CALL MOVE (-1,37*LOC(1),DUM,HOL(1))
DO 10 I=1,1
10 LOC(I)=LOC(I)
TR = (1.0+2*C(LMO)**2)
C(LP4) = 5.0 * C(LPO) * TR**3.5
C(LP7) = C(LP4)/3.5
C(LXX) = 75.0
C(LC4) = 1.0
C(LTX) = C(LTO) * TR
C(LWD) = C(LP4)/C(LTX) * 1948961
C(LWH) = C(LWD) / 10.0
C(LN2) = 10000.
CALL MOVE(0)
C(LAT) = 435.
CALL UPDATE
C(LAT) = SQRT(C(LTX))*C(LW1)/C(LP4)/918861447/C(LMT)*
1 (1.0+2*C(LMT)**2)**3
C(IAT) = C(LAT)
LNT = K(LN1)
C(LTH) = (C(LAT)-C(L19))/C(LNT+34)
CALL PRINT (NAME)
CALL SLMASS

117
TABLE I. STEADY (CONCLUDED)

IMWFX=0
C(LN2)=C(IN2)
C(JN2)=C(IN2)
LNT=K(LNN)
200 CALL SJENG(IMWFX)
   IF(IMWFX) 210, 210, 220
210 C(LQ0)=((C(LWF)/C(LP4)-C(LQ1))/C(LNT+14)
   C(IQ0)=C(LQ0)
   220 WFUEL=C(LWF)
   C(IA9)=C(LA9)
   C(LAJ)=((C(LA9)-210+1)/C(LNT+53)
   C(IAJ)=C(LAJ)
   C(LBP)=((C(LW7)-C(LW2))*SQRT(C(LTX))/((C(LC1)*C(LC3)*C(LP2)
   C(LNW)=C(JN2)
   CALL PRINT (NAME)
   CALL UPDATE
   CALL PRINT (NAME)
   CALL MOVE(0)
   CALL UPDATE
   CALL PRINT (NAME)
   IF(ABS(WFUEL-C(LWF))<.000001) 400, 400, 300
   300 IF(IMWFX) 310, 310, 400
   310 IF(C(LWF)-WFUEL) 320, 320, 330
   320 C(LQ0)=C(LNT+12)
   GO TO 340
   330 C(LQ0)=C(LQ0)
   IMWFX=1
   C(IQ0)=C(LQ0)
   GO TO 200
   400 RETURN
END

118
### TABLE II. MOVE

**SUBROUTINE MOVE** *(NV,LOC,HOL,VAR)*  

C *** WHEN M = -1 DSL/90 PARAMETER NAMES IN HOL ARE LOCATED AND THEIR LOCATION PLACED IN LOC ARRAY  

C *** WHEN M = 0 DSL/90 INTEGRATOR I/C VALUES ARE MOVED TO OUTPUTS  

C *** WHEN M = +1 M = -1 IS EXECUTED THEN VALUES OF C(LOC) ARE MOVED TO VAR ARRAY  

**COMMON CURVAL(C(1)),SYMBOLS/NOINTG**  

**CALL ARGQ(NA)**  

**IF(NA.EQ.0)GO TO 500**  

**NA = NA - 3**  

**DO 100 I = 1,NV**  

**LOC(I) = LOOK(HOL(I))**  

**IF(NA.EQ.0)GO TO 100**  

**LOCA = LOC(1)**  

**VAR(I) = C(LOCA)**  

**100 CONTINUE**  

**RETURN**  

**END MOVE**
TABLE III. SLMASS

$SUBFIC SLMASS$
SUBROUTINE SLMASS
COMMON/CURVAL/C(1)
COMMON/SYMBLS/NOINTG,NOSYM,SYM(1)
COMMON/MEMRY/HIST(1)
DIMENSION
1 HOL(22),VAR(22),LL(22)
DATA HOL(1)/132HX01005X01021X01023QL1101A01TOOCN1004A0160
1A01004V01004P01004L01042WQ1904QL1046QA1022QA1005
2X1002WQ1967T01015WA1027 /
EQUIVALENCE
1 (VAR ( 1)* XI),(VAR ( 2)* XI),(VAR ( 3)* XIII)
2*(VAR ( 4)* HVL),(VAR ( 5)* AREA),(VAR ( 6)* XKAI)
3*(VAR ( 7)* AT),(VAR. ( 8)* ADGEO),(VAR ( 9)* VD)
4*(VAR (10)* PTY),(VAR (11)* PTDH),(VAR (12)* DLPTYZ)
5*(VAR (13)* DLPTDZ),(VAR (14)* DLPTD2),(VAR (15)* WDB)
6*(VAR (16)* PHIY),(VAR (17)* WII),(VAR (18)* WX)
7*(VAR (19)* X2),(VAR (20)* WHB),(VAR (21)* TTD)
8*(LL ( 6)* LAR),(VAR (22)* WZ)
CALL UPDATE
CALL MOVE (1*22*LL(1)*VAR(1)*HOL(1))
SAVE=HIST(1)
HIST(1)=O*4
XTEMP=X
SQRTTD=SQRT(TTD)
IF(XTEMP<XIII) 210*220*220
210 WXBB=WX-(X-XIII)*WII*PHIY
GO TO 230
220 WXBB=WX
230 N=11
DELX=HVL/10*
DP=DLPTYZ/10*
PTN=PTY+DP
11=1
L=LL(20)
240 J=0
WTH=O*O
XTEMP=XTEMP-DLX
DO 380 I=1*N
PTN=PTN+DP
XTEMP=XTEMP+DELX
AR=SLTLM(AREA,XTEMP,AT)
IF(XTEMP<XIII) 250*260*260
250 WN=WX-(X-XTEMP)*WII*PHIY
GO TO 270
260 WN=WXBB
270 WPAR=WN/PTN*SQRTTD/AR
XMX=SLMVG(1*0*4*WPAR,1*4)
330 TN=TDD/(49*021176*SQRT(TN)*XMX)
WN=WN/(49*021176*SQRT(TN)*XMX)
IF(J) 340*360*350

120
TABLE III. SLMASS (CONCLUDED)

340 WN=4.0*WN
   J=1
   GO TO 370
350 WN=WN+WN
360 J=-1
370 WTH=WTH+WN
380 CONTINUE
   C(L)=DELX/36.0*(WTH-WN/2.0)
   IF(L) 400,400,390
390 IJ=1
   L=LL(15)
   N=101
   DELX=(X2-(X+HVL))/100.
   WDB=WDB
   DP=(DLPTZD+DLPTD2)/100.
   PTN=PTY-DLPTYZ+DP
   GO TO 240
400 HIST(L)=SAVE
   PTDPD=VD/C(L)*PTDH/TTD=2.699161
   XMDHS=5*(PTDPD**4-1)
   XMF=(1+2*XMDHS)**3
   CLAR=WZ/PTDH**SQRT(TTD/XMDHS)/.91886145*XMF/ADGEO*XXAI
   CALL MOVEIO
   CALL UPDATE
   RETURN
END
TABLE IV. SPJENG

**C**
PROGRAM TO INITIALIZE SIMPLIFIED INTEGRATED PROPULSION SYSTEM

**C**
5-5-68 TURBOJET ENGINE INITIALIZATION

**SUBROUTINE SPJENG (IMWFX)**

**COMMON/CURVAL/C(11)/SYMBOLS/NOINTG/NOSYMBSYMB (1)**

**DIMENSION**

1 HOL(24) •• LOC(24) • • VAR(8)

**DATA NAME/6HSPJENG/**

**DATA HOL(11) / 144HCN6222N06002T08606T04003PO4003CN367TPR3T07WA3T31**

**1PO6005PO60057PO6005106005WF6001WA6003MD6034MD6056WG6055WG6056A06009JENG0080**

**2CN6005CN6007WG6095WG6096QA8611 /**

**EQUIVALENCE**

1 (LOC ( 9) • LP4) (LOC ( 10) • LP7) (LOC ( 11) • LP5) JENG0100

2 (LOC ( 12) • LT5) (LOC ( 13) • LWF) (LOC ( 14) • LW3) JENG0120

3 (LOC ( 15) • LH4) (LOC ( 16) • LH6) (LOC ( 17) • LW5) JENG0130

4 (LOC ( 18) • LW6) (LOC ( 19) • LA9) (LOC ( 20) • LP4) JENG0140

5 (LOC ( 21) • IP7) (LOC ( 22) • L95) (LOC ( 23) • L96) JENG0150

6 (LOC ( 24) • LWP) (LOC ( 2) • LN2) JENG0160

**REAL**

NO6002 • N2C3NX

IF (IMWFX.EQ.1) GO TO 15

CALL MOVE(0)

CALL MOVE (1.8 * LOC(1), VAR(1), HOL(1))

CALL MOVE (-1.6 * LOC(9), VAR(1), HOL(9))

DO 10 I = 1,1

10 LOC(I) = LOC(I)

N2C3NX = NO6002/SQRT(T04003/S18.69)/C(LN367T+4)

PRHX = SPT(LU/PRT6T7,N2C3NX)

C(LP4) = (PRHX-1.0)/(C(LN367T+4)+1.0)*P04003 * 0.9

C(LP7) = (CLP4)/3.5

C(LWFI) = (CLWP)*C(LP4)

C(LP7) = (CLP7)*C(LWFI)

ICT5 = 100

160 DELT6 = (CLT5) - T08606

WRITE (6,20) N2C3NX, PRHX, C(LP4), C(LP7), C(LWFI)

20 FORMAT (1HO/1H 5F20.8)

30 FORMAT (1H 15.5F19.5/C1H)

100 CALL UPDATE

150 ICT5 = 100

160 DELT6 = (CLT5) - T08606

WRITE (6,30) ICT5, C(LWFI), C(LT5), C(LN2), DELT6

IF (ABS(DEL6) > 0.01) 180, 180, 170

170 ICT5 = ICT5 - 1

200 CALL UPDATE

180 ICT5 = 100

190 DELHP = (CH4) + (CLW5) - (CH6) + (CLW5) - CN6222

WRITE (6,30) ICT5, C(LP7), C(LH4), C(LW3), C(LH6), C(LW5), DELHP

IF (ABS(DLHP) > 0.1) 210, 210, 200

200 ICT5 = ICT5 - 1

C(LP7) = C(LP7) - DELHP/200

CALL UPDATE

IF (ICT5) 224, 190, 190

JENG0310

JENG0390

JENG0400

JENG0420

JENG0430

JENG0440

JENG0450

JENG0460

JENG0470

JENG0490

JENG0500

JENG0510

JENG0520

122
TABLE IV. SPJENG (CONCLUDED)

210 DELWT = C(LW5) - C(LW6)
WRITE (6, 30) ICWT, WA603G, C(LW5), C(LW6), DELWT
IF (ABS(DELWT) > 0.001) 230 230 220
220 ICWT = ICWT - 1
C(LP4) = C(LP4) * (1 + 5 * DELWT / C(LW5))
WRITE (6, 222) ICOMP, ICT5, ICHP, ICWT
222 FORMAT (1H, 4I10)
IF (ICWT) 224 100 100
CALL INTRAN
CALL PRINT (NAME)
C(LA9) = C(LA9) * C(L95) / C(L96)
C(IP4) = C(LP4)
C(IP7) = C(LP7)
RETURN
END
The primary objective of Task 7 of the "Propulsion System Flow Stability Program" was to develop a simulation program to be used in Phase II for the evaluation of two control systems capable of sensing and accommodating a transient condition.

Since the work on this task was being performed by three companies, every effort was made to insure compatibility in terminology, units, and program documentation as well as to provide means of communicating the myriad details involved in making computer runs of the system. This documentation format is described in Section II of this volume.

An early element of this task was the selection of a simulation language for use in programming the simulation. The choice of IBM's DSL/90 and the factors involved in making that choice are discussed in Section III.

Simulation programs have a natural tendency to be rather voluminous and, when the system being simulated is as complex as a supersonic inlet, turbofan, and an integrated control system can be, computer storage space is rapidly filled. To alleviate this crowding, numerous logic blocks which were repetitive, such as compressor logic, were removed from the simulation logic deck and made into subroutines or functions. These subprograms are discussed in Section IV.
Propulsion System Computer Program

Propulsion System Simulation

ITEM 13. ABSTRACT (continued)

Once the simulation logic is written, the most difficult task of all begins. The job of initialization is usually not given proper emphasis until many hours of work have convinced all concerned that it is really the most important phase. Section V discusses this task and shows an example of an initialization routine.
<table>
<thead>
<tr>
<th>Title/Authors</th>
<th>Accession Number</th>
<th>Publication Year</th>
<th>Page Count</th>
<th>Download</th>
</tr>
</thead>
</table>
Accession Number: AD0859282


Descriptive Note: Final technical rept. Jun 67-Sep 68 on Phase 1.

Corporate Author: NORTH AMERICAN ROCKWELL CORP LOS ANGELES CA LOS ANGELES DIV

Personal Author(s): Kaplan, Earl H; Wong, Heeman W

Report Date: Dec 1968

Pagination or Media Count: 133

Abstract: The object of the study was to develop a simulation program to be used for the evaluation of two control systems capable of sensing and accommodating a transient condition. (Author)

Descriptors: *TURBOJET ENGINES, CONTROL SYSTEMS, MATHEMATICAL MODELS, COMPUTER PROGRAMS, TRANSIENTS, DUCT INLETS, SUPersonic CHARACTERISTICS, TURBOFAN ENGINES, FLOW CHARTING, SUBROUTINES, DIGITAL COMPUTERS, INPUT OUTPUT DEVICES

Subject Categories: Jet and Gas Turbine Engines

Distribution Statement: APPROVED FOR PUBLIC RELEASE