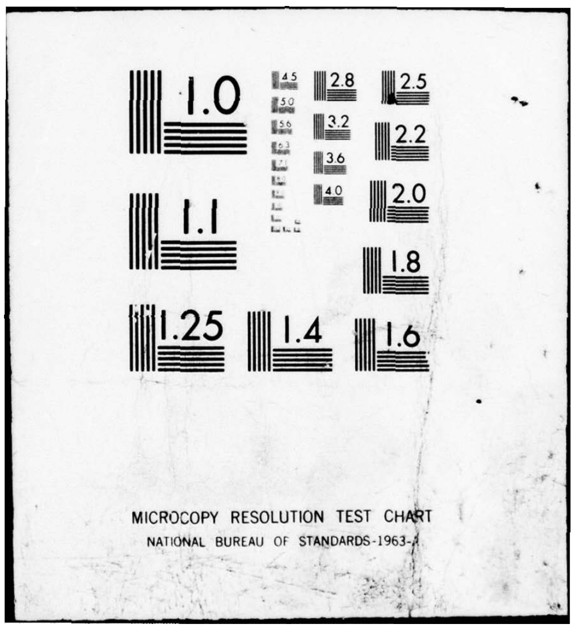
AD-A064 944 GENERAL ELECTRIC CO CINCINNATI OHIO AIRCRAFT ENGINE GROUP F/G 21/5
ANALYTICAL DERIVATIVES.(U)
DEC 78 D F BERG, W C COLLEY, G L CONVERSE F33615-78-C-2203
UNCLASSIFIED R78AEG517 AFAPL-TR-78-101 NL

1 OF 1
AD
A064944

END
DATE
FILMED
4--79
DDC


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
5.6
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-

AFAPL-TR-78-101

ANALYTICAL DERIVATIVES

FINAL TECHNICAL REPORT

General Electric Company
Cincinnati, Ohio 45215

December 1978

(FINAL REPORT FOR PERIOD JUNE 1978 - SEPTEMBER 1978)

Approved for Public Release; Distribution Unlimited

AIR FORCE AERO PROPULSION LABORATORY
AIR FORCE SYSTEMS COMMAND
WRIGHT-PATTERSON AFB, OHIO 45433

NOTICE

When Government drawings, specifications, or other data are used for any purpose other than in connection with a definitely related Government procurement operation, the United States Government thereby incurs no responsibility nor any obligation whatsoever; and the fact that the government may have formulated, furnished, or in any way supplied the said drawings, specifications, or other data, is not to be regarded by implication or otherwise as in any manner licensing the holder or any other person or corporation, or conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.

This report has been reviewed by the Information Office (OI) and is releasable to the National Technical Information Service (NTIS). At NTIS, it will be available to the general public, including foreign nations.

This technical report has been reviewed and is approved for publication.

JAMES R. RUBLE
Technical Manager
Turbine Engine Performance

JACK RICHENS, Chief
Performance Branch

FOR THE COMMANDER

ERNEST C. SIMPSON, Director
Turbine Engine Division

"If your address has changed, if you wish to be removed from our mailing list, or if the addressee is no longer employed by your organization please notify AFAPL/TBA *, W-PAFB, OH 45433 to help us maintain a current mailing list".*

Copies of this report should not be returned unless return is required by security considerations, contractual obligations, or notice on a specific document.

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER	AFAPL-TR-78-101
2. GOVT ACCESSION NO.	
3. RECIPIENT'S CATALOG NUMBER	
4. TITLE (and Subtitle)	ANALYTICAL DERIVATIVES.
5. TYPE OF REPORT & PERIOD COVERED	Final Technical Report. June [redacted]8 - Sep[redacted]978
6. PERFORMING ORG. REPORT NUMBER	R78AEG517
7. AUTHOR(s)	D.F. Berg, W.C. Colley, G.L. Converse
8. CONTRACT OR GRANT NUMBER(s)	F33615-78-C-2203 new
9. PERFORMING ORGANIZATION NAME AND ADDRESS	General Electric Company, Aircraft Engine Group, Cincinnati, Ohio 45215
10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS	3066-11-35
11. CONTROLLING OFFICE NAME AND ADDRESS	Air Force Aero Propulsion Laboratory (TBA), Wright-Patterson AFB, Ohio 45433
12. REPORT DATE	December 1978
13. NUMBER OF PAGES	72p.
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)	
15. SECURITY CLASS. (of this report)	Unclassified
15a. DECLASSIFICATION/DOWNGRADING SCHEDULE	N/A

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release: distribution unlimited

62203F

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Analytical Derivatives
Engine Cycle Balance
Iteration Techniques

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

The variable cycle engines now being studied for preliminary engine-aircraft design have greatly increased the complexity of engine modeling, control element design, and cycle calculation relative to the simpler turbofan and turbojet engines which preceeded them. The production of engine data is the major computations cost element in mission studies and optimization investigations; this problem can become even more severe in the future unless → next page

403 468

improvements in cycle calculations and component modeling techniques are developed and introduced now.

The current method of obtaining balanced cycle data points is to use the simultaneous Newton-Raphson iteration method. The partial derivatives are obtained by numerical differentiation. This report documents the results obtained by calculating the partial derivatives from analytical expressions obtained by differentiating a cycle deck. Examples illustrating both the method of obtaining the analytical derivatives as well as setting up the control logic are given. A cost comparison was carried out between two engine simulations using numerical derivatives and the same pair of engine simulations using analytical derivatives. A data matrix consisting of the same 411 operating points was used for each of the engine simulation comparisons.

A cost saving of about 44 percent was obtained for the deck run internally and 52 percent for the deck run externally (the difference is due to the use of a larger set of output parameters for the internal deck). It is concluded that the greatest return per dollar of cost results when only that portion of the model for which the coding remains relatively constant is differentiated. Applying this philosophy an annual saving of about $75,000 is estimated for the running of internal (AEG-Evendale) cycle decks.

PREFACE

This report describes a design study effort conducted by the General Electric Company and sponsored by the Turbine Engine Division of the Air Force Aero Propulsion Laboratory, Air Force Systems Command, Wright-Patterson AFB, Ohio under Project 3066-11-35, Contract F33615-78-C-2203. Mr. James R. Ruble, AFAPL/TBA, was the Air Force Project Engineer.

The work reported herein was performed during the four-month period beginning June 1978 and ending September 1978. The GE Engineering Manager was Donald F. Berg who was assisted principally by William C. Colley and George L. Converse. The Program Manager was Donald E. Uehling.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

LIST OF SYMBOLS

STATION NUMBERS

1	ambient
2	fan entrance
21	fan exit/compressor and duct entrance
3	compressor exit/burner entrance
4	burner exit/hi pressure turbine entrance
5	hi press. turbine exit/lo press. turbine entrance
55	lo press. turbine exit
6	afterburner entrance
7	afterburner exit
8	main nozzle throat
9	main nozzle exit
23	duct burner entrance
24	duct burner exit
25	duct exit (if mixed-flow engine)
28	duct nozzle throat
29	duct nozzle exit

THERMODYNAMIC PROPERTIES

AM	Mach number
FAR	fuel-air ratio
H	total enthalpy (Btu/lb)
P	total pressure (atmospheres)
PS	static pressure (atmospheres)
S	total entropy
T	total temperature (degrees R)
TS	static temperature (degrees R)
V	velocity (feet/second)

COMPONENT SYMBOLS

A	afterburner
AFT	afterburner
B	burner
C	compressor
COM	burner
D	duct
DUC	duct
F	fan
M	main nozzle
NOZ	nozzle
OB	overboard
T	total
THP	hi pressure turbine
TLP	lo pressure turbine

LIST OF SYMBOLS (Continued)

ENGINE SYMBOLS

BL	bleed (pounds/sec)
CN	corrected speed ratio
DHT	turbine delta enthalpy (Btu/lb)
DHTC	turbine delta enthalpy (temperature corrected) (Btu/lb)
DP	pressure drop (lb/sq.in.)
DT	temperature increase (degrees R)
ETA	efficiency
ETAR	ram recovery
HPEXT	horsepower extracted
PCBL	percent bleed
PCN	percent speed
PR	pressure ratio
TFF	turbine flow function
WA	airflow (pounds/sec)
WF	fuel flow (pounds/sec)
WG	gas flow (pounds/sec)
Z	pressure-ratio ratio

MISCELLANEOUS

A	area (sq.in.)
ALTP	altitude (ft)
AM	Mach number of aircraft
BYPASS	bypass ratio
CF	correction factor
CS	ambient speed of sound (ft/sec)
CV	nozzle velocity coefficient
DEL	delta degradation coefficient
DS	design value
DUM	dummy (not used)
FG	gross thrust (lbs)
FGM	momentum thrust (lbs)
FGP	pressure thrust (lbs)
FN	net thrust (lbs)
FRD	ram drag (lbs)
GU	initial or guess values
ITRYS	number of loops through engine before quitting
SFC	specific fuel consumption
TOLALL	tolerance
VA	velocity of aircraft (ft/sec)
VJ	jet velocity (ft/sec)

NOTE: Some symbols may be truncated when combined with other symbols due to six character limit imposed by FORTRAN Computer Language.

SUMMARY

3 The next generation of military aircraft weapon systems will be required to achieve substantial improvements in design mission performance and effectiveness, multimission versatility, life-cycle costs, and survivability. These requirements will place increased demands on the propulsion system for advanced cycle concepts, advanced material and design technology, variable geometry capabilities, and a more effective engine-airframe installation with particular emphasis on inlet-engine airflow matching and engine-airframe thrust matching across the complete operating regime. The resulting propulsion system concept and evaluation and cycle selection process will require a more complex engine-airframe interaction which will involve a substantially greater number of engine and airframe design parameters, more extensive control and scheduling requirements, and a broader spectrum of mission and operational requirements.

The studies necessary to satisfy these increased design requirements will necessitate calculating many more balanced engine cycle performance points. This study program, Analytical Derivatives, was aimed at reducing the computer cost for calculating engine cycle performance by utilizing analytical derivatives instead of finite difference derivatives in the engine cycle balance iteration procedure. The technique developed is to differentiate the engine cycle and to use the numerical values of these analytical derivatives in the cycle balance iteration procedure. The calculation of the analytical derivatives will require less computer processor time than the calculation of finite difference derivatives which require evaluating the complete cycle for each independent variable. This technique was applied to a Variable Cycle Engine simulation and the resulting computer program was used to calculate 411 flight operating points. The results indicated that the computer costs were reduced about 52% when using the WPAFB CDC6600/CYBER 74 computer.

During this study, an alternate approach of applying the analytical derivative concept was devised. The differentiation of the complete engine cycle requires the constant updating of these analytical derivatives whenever a new component is added or equations are changed or added to an existing component. This alternate approach differentiates only the low level subroutines, such as those subroutines which calculate thermodynamic and aerodynamic properties. These subroutines represent approximately 10% of engine cycle program logic but account for about 75% of the computer processor time. By applying the analytical derivative technique the calculation time for these subroutines will be reduced and since these subroutines remain unchanged changes in the engine cycle will not require additions to this set of analytical derivatives. Utilizing the same Variable Engine Cycle program to calculate the 411 flight operating points, the results showed about a 47% reduction in computer processor time on a Honeywell 6000 computer. Using the ratio of the Honeywell 6000 computer time savings for the alternate method to the analytical derivative technique, it is estimated that the alternate method would result in a 40% costs savings using the WPAFB CDC6600/CYBER74 computer.

It can be concluded that in any engine simulation model using numerical derivatives to obtain cycle points, a considerable savings in both computer time and cost can be obtained by the inclusion of analytic derivatives. The greatest return per dollar of cost results when only that portion of the model for which coding remains relatively constant is differentiated. For the Air Force engine simulation model (SMOTE), the entire deck should be differentiated since the coding changes do not appear to occur frequently.

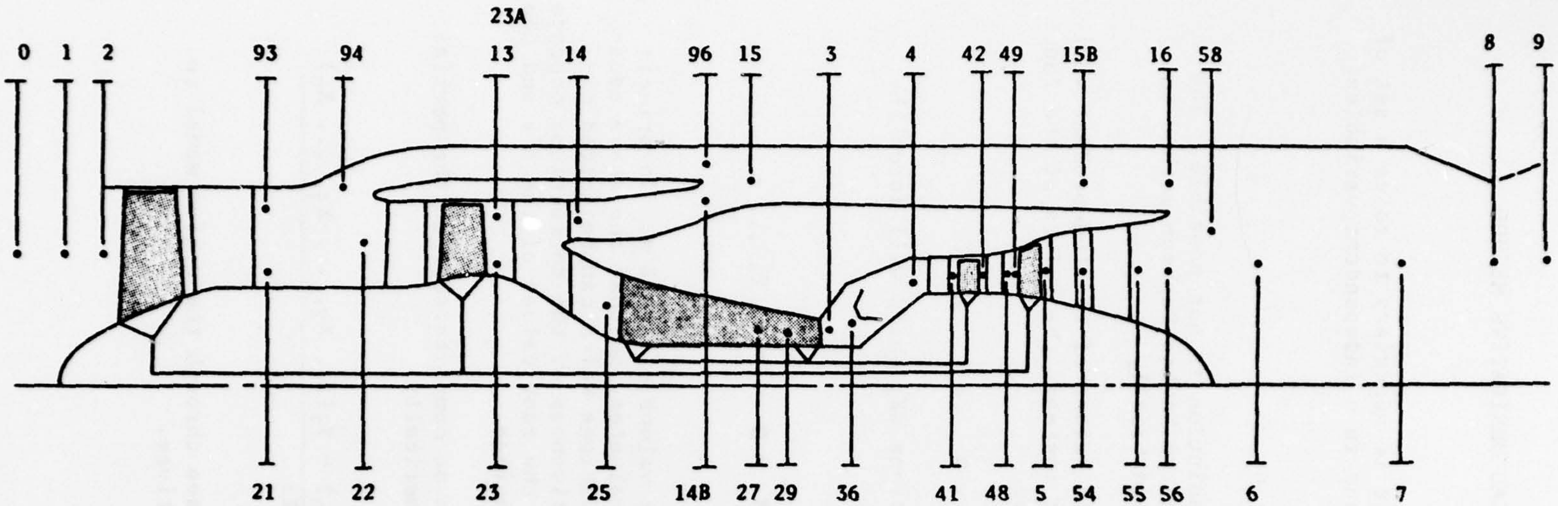
1.0 INTRODUCTION

The Aircraft Engine Group (AEG) of the General Electric Company has completed a contract entitled "Analytical Derivatives" which was sponsored by the Turbine Engine Divison of the Air Force Aero Propulsion Laboratory. The Air Force Project Engineer was James R. Ruble of the AFAPL Performance Branch. The overall objective of the contract effort was to develop and assess the suitability and the cost reduction potential of the analytical derivative procedure as applied to the computer modeling of turbine engine performance. In particular, this program developed the relevant mathematical expressions for the analytic derivatives and applied the analytic derivative technique to the calculation of balanced cycle performance for a selected Variable Cycle Engine (VCE) simulation.

2.0 SELECTION OF ENGINE MODEL

The analytical derivative study utilized two point design Variable Cycle Engine (VCE) simulations which are AEG inhouse computer programs. A schematic of the VCE used during the study is shown in Figure 1. Both the fan blocks and the compressor representation within the VCE simulation exist in parametric form. The remainder of the cycle equations are consistent with those utilized in parametric cycle decks, thereby assuring that all analytical derivatives generated in the program effort would be useful in full parametric performance decks. By utilizing the parametric feature, the engine can be changed easily by selecting different compressor and fan designs from the family of designs available in the model.

The demonstration of the analytical derivative procedure was carried out using two different engines. The initial evaluation used an engine with a fan, core-driven booster, and a compressor, each selected from the parametric system by design pressure ratios of 2.97, 1.35, and 7.00, respectively. This was designated as Engine 1. For the second engine, the design pressure ratios of these components were 3.28, 1.52, and 5.60. The second engine was designated as Engine 2.



0	Free Stream Air Conditions	4	HP Turbine Vane Inlet	7	Exhaust Nozzle Inlet
1	Engine Inlet	41	HP Turbine Rotor Inlet	8	Exhaust Nozzle Throat
2	Front Fan Inlet	42	HP Turbine Rotor Exit	9	Exhaust Nozzle Exit
21	Front Fan Hub Discharge	48	LP Turbine Vane Inlet	93	Front Fan Tip Discharge
22	Second Fan Inlet	49	LP Turbine Rotor Inlet	13	Second Fan Tip Discharge
23	Second Fan Hub Discharge	5	LP Turbine Rotor Exit	14	Inner Duct Inlet
23A	Second Fan Hub Average	54	LP Turbine OGV Exit	14B	Inner Duct Mixer Exit
25	HP Compressor Inlet	55	LP Turbine Frame Exit	15	Front Mixer Mixed
27	HPC 5th Stage	56	Core Mixer Exit	15B	Bypass Mixer Inlet
29	HPC 6th Stage	58	Aft Mixer Mixed	16	Bypass Mixer Exit
3	HPC Discharge	6	Augmentor Inlet	94	Outer Duct Inlet
36	Combustor Inlet			96	Outer Duct Mixer Exit

Figure 1. Schematic of Typical VCE Showing Station Designations.

3.0 DESCRIPTION OF THE ANALYTICAL DERIVATIVE METHOD

In order to balance an engine cycle, it is necessary to solve a set of n simultaneous non-linear algebraic equations in n independent variables, represented symbolically by:

$$Y_i(X_1, X_2, \ldots X_n) = 0 \quad i = 1, 2, \ldots n$$

Since the equations are nonlinear, direct solution is not possible. The solution is found by trial and error, using the Newton-Raphson method to guide convergence. This method is summarized briefly.

In general, for some set of approximate values of the independent variables, the equations will not be satisfied; instead, the values of the functions Y_i will differ from zero by some error

$$Y_i(X_1, X_2, \ldots X_n) = E_i \quad i = 1, 2, \ldots n$$

The object is to determine a set of corrections δX_i which will force the erors to zero:

$$Y_i + \frac{\partial Y_i}{\partial X_1} \delta X_1 + \frac{\partial Y_i}{\partial X_2} \delta X_2 + \ldots + \frac{\partial X_i}{\partial X_n} \delta X_n = 0 \quad i = 1, 2, \ldots n$$

The Y_i's are evaluated using the aproximate values of the X's. In principal, the partial derivatives can also be evaluated, so that the above equations form a set of simultaneous linear equations which can be solved directly for the corrections δX_i. The effectiveness of the corrections depends upon the accuracy of the approximations to the root values of the X's and the linearity of the original simultaneous equations.

In engine cycles, the functions Yi are so complicated that the partial derivatives have usually been evaluated numerically:

$$\frac{\partial Y_i}{\partial X_j} \approx \frac{Y_i(X_1, X_2, \ldots X_j + \Delta X_j, \ldots X_n) - Y_i(X_1, X_2, \ldots X_j, \ldots X_n)}{(X_j + \Delta X_j) - X_j}$$

This procedure requires n + 1 complete passes through the engine model to compute each set of Y's and partial derivatives.

Direct evaluation of the partial derivatives requires much less computation time, as will be shown, although a completely different logic path through the engine cycle is required.

3.1 ANALYTICAL DERIVATIVE APPROACH - COMPLETE DERIVATIVE SET

A description of the basic analytical derivative approach follows. Each component module has a list of needed inputs (X_1, X_2---X_n) and calculates a set of dependent variables (Y_1, Y_2, Y_3---Y_m) utilized in other modules. A set of total differential equations can be mathematically derived:

$$dY_1 = \frac{\partial Y_1}{\partial X_1} dX_1 + \frac{\partial Y_1}{\partial X_2} dX_2 \text{ --- } + \frac{\partial Y_1}{\partial X_n} dX_n$$

$$dY_2 = \frac{\partial Y_2}{\partial X_1} dX_1 + \frac{\partial Y_2}{\partial X_2} dX_2 \text{ --- } + \frac{\partial Y_2}{\partial X_n} dX_n$$

$$dY_m = \frac{\partial Y_m}{\partial X_1} dX_1 + \frac{\partial Y_m}{\partial X_2} dX_2 \text{ --- } + \frac{\partial X_m}{\partial X_n} dX_n$$

The partial derivatives $\partial Y_1/\partial X_1$, $\partial Y_2/\partial X_n$, etc. are in general complex groupings of parameters calculated on the base point. The total differential equations can be derived and programmed in each subroutine. When the subroutines are executed in order, the values of dY_1 from an upstream subroutine frequently become the input values of dX_1 needed in a downstream calculation. In this way, the derivative expressions can be modularized and utilized in a variety of cycles consisting of different ordered sets of subprogram modules.

To iterate a cycle to balance, partial derivatives with respect to a set of independent variables Z_1, Z_2, --- Z_n are required. This set of variables is a sub-set of total set of X_i's utilized in each subroutine as independent variables. Similarly, a subset of the Y_i's represents a set of variables to be driven to zero by the iterative technique. To evaluate the partial derivatives of the Y's needed by the iterative technique with respect to, say, Z_1, the value of dX_i corresponding to X_i is set equal to unity and all other dX values corresponding to Z_2 through Z_n are set to zero. By calculation through the derivative equations the appropriate derivative values can be evaluated. The derivative equations must be evaluated in this manner once for each independent iteration variable. Although this procedure leads to multiple evaluations of the derivative equations, it does make the derivative equations independent of subroutine order and only dependent on a fixed set of X_i's and Y_i's in each module (unique interface parameter sets between routines). More importantly, perhaps, it permits the derivative equations to be programmed in a much more compact manner. Large portions of derivative evaluations are common to different independent variables and need not be recalculated for each variable.

As an example, consider a core compressor. Two of the input variables (X's) could be core speed (PCN25) and inlet air temperature (T25). Corresponding output variables (Y's) would be stall margin (SM25) and outlet air temperature (T3). The core speed is typical of the independent iteration variables (Z's) while T25 is calculated in an upstream component. Similarly, stall margin is typically a dependent iteration variable (driven to a demand value) while T3 is used as an input to the combustor. The equations would take the form:

$$dT3 = \frac{\partial T3}{\partial T25} dT25 + \frac{\partial T3}{\partial PCN25} dPCN25 + \ldots$$

$$dSM25 = \frac{\partial SM25}{\partial T25} dT25 + \frac{\partial SM25}{\partial PCN25} dPCN25 + \ldots$$

With dPCN25 set equal to unity and the differentials of all other independent iteration variables dZ's set equal to zero the set of equations reduces to:

$$dT3 = \partial T3/\partial PCN25$$

$$dSM25 = \partial SM25/\partial PCN25$$

If, on the other hand, we set dPCN25 equal to zero and set the differential of some other independent variable (say fan speed dPCN2) to unity, then the equations reduce to:

$$dT3 = \frac{\partial T3}{\partial T25} dT25$$

$$dSM25 = \frac{\partial SM25}{\partial T25} dT25$$

Although the magnitude of dT3 is different for the two cases, the same combustor and other downstream equations (utilizing dT3) can be used.

Note that, in the second example, the temperature T25 is a function of fan speed, so that:

$$dT25 = \frac{\partial T25}{\partial PCN2} dPCN2 + \ldots$$

through the chaining of logic in preceeding component modules. When dPCN2 = 1, and all other independent variable differentials are zero,

$$dT3 = \frac{\partial T3}{\partial T25} \frac{\partial T25}{\partial PCN2} = \frac{\partial T3}{\partial PCN2}$$

although the terms are not actually collected in this form.

3.2 SUBROUTINE DERIVATIVE STRUCTURE

in general, the total differential equations described previously were programmed into the high-level (component module) subroutines, even though evaluation of some variables occurs in low-level utility routines. The utility rouitines were modified as required to provide values of partial derivatives. Exceptions were the compressor and turbine map subroutines, where the differentiation was performed in the subroutines.

Differentiation of most of the formulae was straightforward, since, usually, a dependent variable was expressed explicitly as a function of independent variables:

$$Y = f(X_1, X_2, \ldots X_n)$$

$$dY = \frac{\partial f}{\partial X_1} dX_1 + \frac{\partial f}{\partial X_2} dX_2 + \cdots + \frac{\partial f}{\partial X_n} dX_n$$

Even in cases where the equations could not be solved explicitly for the dependent variables, and iteration was required for evaluation of the variables themselves, the derivatives of the functions could always be obtained, so that explicit linear relations for the differentials were obtained, which did not require iteration.

For a system of equations:

$$f_i(X_1, X_2, \ldots X_n, Y_1, Y_2, \ldots Y_m) = 0 \qquad i = 1, 2, \ldots m$$

the total differential expressions reduce to:

$$\frac{\partial f_i}{\partial Y_1} dY_1 + \frac{\partial f_i}{\partial Y_2} dY_2 + \ldots + \frac{\partial f_i}{\partial Y_m} dY_m$$

$$= -\left(\frac{\partial f_i}{\partial X_1} dX_1 + \frac{\partial f_i}{\partial X_2} dX_2 + \cdots + \frac{\partial f_i}{\partial X_n} dX_n\right) \qquad i = 1, 2, \ldots m$$

which are linear in the dy's and can be solved directly.

The algebraic terms representing the partial derivatives in the above formulae are evaluated during the first derivative pass, using values of independent and dependent variables obtained during the base pass. Since, in subsequent derivative passes, only the dX's change, the values calculated in the first pass are retained in storage arrays for reuse. Additional computation time was saved by performing derivative calculations in only part of the engine model on each derivative pass. Derivative calculations downstream of the point where the last dependent iteration variable is evaluated are unnecessary, as are calculations upstream of the point where each independent iteration variable first appears, except for evaluation of partial derivative terms on the first derivative pass. First-guess and engine sizing formulae were not differentiated.

In the General Electric cycle decks, three different thermodynamic property subroutines are used, in each of which the variables by lnP (log of pressure), FAR (fuel air ratio) and WAR (water air ratio) are independent. Of the variables T, H and S, one is independent and the other two dependent, according to which subroutine is called. Other dependent variables are R (gas constant), CS (ratio of specific heats at contant entropy), ALPHA and BETA (defined in the following equations). In differentiating these relations, WAR, CS, ALPHA, and BETA were considered to be constants. The differentiation formulae were derived as follows:

$$H = H\ (T,\ \ln P,\ FAR)$$

$$dH = \left.\frac{\partial H}{\partial T}\right|_{P,FAR} dT + \left.\frac{\partial H}{(\ln P)}\right|_{T,FAR} d(\ln P) + \left.\frac{\partial H}{\partial (FAR)}\right|_{T,P} d(FAR)$$

by standard thermodynamic relations, (Reference 1):

$$\left(\frac{\partial H}{\partial T}\right)_{P,\ FAR} = CP = CS\ (1 - ALPHA)$$

$$ALPHA = -\frac{T}{R}\left(\frac{\partial R}{\partial T}\right)_{P,\ FAR}$$

$$\left.\frac{\partial H}{\partial (\ln P)}\right|_{T,\ FAR} = P\left(\frac{\partial H}{\partial P}\right)_{T,\ FAR} = (ALPHA)(R)T$$

The derivative functions CS and ALPHA were already returned from the thermo routines. The coding was modified to compute and return the remaining derivative $\partial H/\partial(FAR)_{T,P}$.

Thus,

$$dH = CS\ (1 - ALPHA)\ dT + (ALPHA)RT\ d(\ln P) + \left.\frac{\partial H}{\partial (FAR)}\right|_{T,P} d\ (FAR)$$

This formula is readily inverted to exchange dependent and independent variables:

$$dT = \frac{dH - (ALPHA)RT\ d(\ln P) - \left.\frac{\partial H}{\partial (FAR)}\right|_{T,P} d(FAR)}{CS\ (1 - ALPHA)}$$

Similarly:

$$S = S\ (T,\ \ln P,\ FAR)$$

$$dS = \left.\frac{\partial S}{\partial T}\right|_{FAR,P} dT + \left.\frac{\partial S}{\partial (\ln P)}\right|_{FAR,T} d(\ln P) + \left.\frac{\partial S}{\partial (FAR)}\right|_{T,P} d(FAR)$$

$$dS = CS\ (1 - ALPHA)\ \frac{dT}{T} - R\ (1 - ALPHA)\ d\ \ln P + \left.\frac{\partial S}{\partial (FAR)}\right|_{T,P} d(FAR)$$

and

$$dR = \frac{\partial R}{\partial T}\bigg|_{P,FAR} dT + \frac{\partial R}{\partial(\ln P)}\bigg|_{T,FAR} d(\ln p) + \frac{\partial R}{\partial(FAR)}\bigg|_{T,P} d\ (FAR)$$

using the definition of ALPHA above and

$$BETA = -\frac{P}{R}\left(\frac{\partial R}{\partial P}\right)_{T,\ FAR} = -\frac{1}{R}\left(\frac{\partial R}{\partial(\ln P)}\right)_{T,\ FAR}$$

then

$$dR = -\ (ALPHA)R\ \frac{dT}{T} - (BETA)R\ d(\ln P) + \frac{\partial R}{\partial(FAR)}\bigg|_{T,P} d\ (FAR)$$

For lean gas mixtures not subject to chemnical dissociation, ALPHA = BETA = 0, and the thermodynamic properties at a given temperature become linear functions of fuel-air ratio when expressed per pound of dry air; for example:

$$H = H_a + (FAR)\ H_f + (WAR)\ H_w$$

$$\frac{\partial H}{\partial(FAR)}\bigg|_{T} = H_f$$

where H_f is the enthalpy of the CO_2 and H_2O formed by burning a pound of fuel, less the oxygen consumed and is a function of temperature only, and H_w is the enthalpy of the water.

Other low-level utility subroutines for which derivative formulae were derived included the following functions:

- Adiabatic Irreversible Compression or Expansion
- Isentropic Acceleration to Sonic Velocity
- Bivariate and Trivariate Table Interpolation
- Mach Number and Pressure Drop as Functions of Flow Function

3.3 LOW LEVEL DERIVATIVE SET

In the course of differentiating the complete engine model, an alternate method of applying the analytical derivative concept was devised. The development of this method was carried forward on a parallel path under separate funding, i.e., not Contract F33615-78-C-2203 funds. As a matter of general information and to report completely on analytical derivative techniques, a description of this alternate method is included.

The analytical derivative technique for calculating the derivative data needed to balance an engine cycle requires the partial differentiation of all of the component subroutines. The analytical derivative method will require the differentiation of any new subroutines or as a minimum, the addition of partial derivatives to account for any equation changes. Therefore, time and effort are required to implement this procedure for any new engine cycle.

As an alternate approach, differentiate the lower level subroutines, such as those subprograms which calculate thermodynamic and aerodynamic properties, and use these derivatives to update the subroutine output data. This approach is implemented by identifying two separate calculation paths in each subroutine. One of the paths is the normal subroutine calculation and this path is taken whenever a base point calculation is made. Added to this path are the calculations of the partial derivatives of the output properties with respect to each of the input parameters, and the storing of both the base point data and the calculated partial derivatives. This added subprogram logic does increase the calculation time for a base point (10%).

The second path in these subroutines is activated whenever one of the independent variables is incremented for the purpose of calculating, by the finite difference method, the partial derivatives used in the iteration method to balance the engine cycle. This alternate path calculates the difference between the saved input values and the new input data and then combines these differences and the saved partial derivatives to calculate updated output data. This path then bypasses the first calculation path and returns to the higher level subroutine. This second path has considerably fewer executable statements and therefore requires less computation time to calculate an updated set of output data.

This approach has the advantage that once the partial derivatives have been derived and included in each of the lower level subroutines they will not change and can be used in any computer cycle program. Component subroutine changes or additions or a completely new component will not require changes in the low level subprograms eliminating the need for updating the derivative calculation procedures. The time and effort to incorporate these new low level subroutines in any computer engine cycle program are minimal.

In order to evaluate this aproach, a new set of low level subroutines was developed and checked out for the chosen VCE cycle. The low level subroutines which were altered to include this new procedure are listed below.

These subroutines represent approximately 10% of the total engine cycle computer program logic; but since they are called numerous times in calculating a balanced engine cycle operating point, they account for about 75% of the processor time. Therefore, any technique which will lower the calculation time of these subroutines will have a substantial effect on the total computer processor time.

ADIABX	Calculates an adiabatic compression/expansion process.
AENOZX	Calculates exhaust nozzle throat area.
COFFEX	Finds the Mach numbers and area required by a variable area mixer at the point of confluency.
CURVEX	Two dimensional table look up procedure Z = f(X,Y).
CURVSX	Three dimensional table look up procedure Z = f(X,Y,W).
PRXXXX	Solves for Mach number given flow function.
SWIRLX	Calculates the turbine exit swirl angle.
6 THOFHX	Calculates the temperature and the thermodynamic properties from enthalpy, fuel air ratio, and pressure.
THOFSX	Calculates the temperature and the thermodynamic properties from entropy, fuel air ratio, and pressure.
THOFTX	Calculates the enthalpy and the thermodynamic properties from temperature, fuel air ratio, and pressure.
GETCXX	Establishes the coefficients required to calculate thermodynamic properties.
HSCALX-CSALPX	Calculates the dissociated thermodynamic properties.

The subroutine MODELX, which controls the order in which the components are called, had to be changed in order to supply an indicator which informs the low level subroutine when a base point or derivative point was being calculated. After incorporating these new subroutines into the engine cycle computer program the calculated performance did not change (within engineering significance).

In order to demonstrate this alternate approach, one of the low level subroutines from the Air Force's computer program SMOTE was chosen to be differentiated and reprogrammed. The subroutine chosen was PROCOM because it is called a large number of times; and if these changes were implemented, there could be a measurable improvement in calculation time. PROCOM calculates the speed of sound (CSEX), ratio of specific-heats (AKEX), gas constant (REX), molecular weight (AMW), specific-heat at constant pressure (CPEX), enthalpy (HEX), and entropy (PHI), as a function of temperature (TEX) and fuel-air ratio (FARX). The procedure was to differentiate CPEX, HEX, PHI, and AMW with respect to TEX and FARX; for example, for CPEX.

CPEX = (CPA + FARX * CPF)/(1. + FARX)

Since CPA and CPF are functions of temperature only, then the following is the derivation of the partial differentiation of CPEX with respect to TEX and FARX.

$$CPA = a_1 + b_1T + c_1T^2 + d_1T^3 + e_1T^4 + f_1T^5 + g_1T^6 + h_1T^7$$

$$CPF = a_2 + b_2T + c_2T^2 + d_2T^3 + e_2T^4 + f_2T^5 + g_2T^6 + h_2T^7$$

$$\frac{\partial CPA}{\partial TEX} = CPAUT = b_1 + 2c_1T + 3d_1T^2 + 4e_1T^3 + 5f_1T^4 + 6g_1T^5 + 7h_1T^6$$

$$\frac{\partial CPF}{\partial TEX} = CPFUT = b_2 + 2c_2T + 3d_2T^2 + 4e_2T^3 + 5f_2T^4 + 6g_2T^5 + 7h_2T^6$$

$$\frac{\partial CPA}{\partial FARX} = \frac{\partial CPF}{\partial FARX} = 0$$

$$\frac{\partial CPEX}{\partial TEX} = \left[\frac{\partial CPA}{\partial TEX} + FARX * \frac{\partial CPF}{\partial TEX}\right] \bigg/ (1. + FARX)$$

$$\frac{\partial CPEX}{\partial TEX} = \underline{CPEXUT} = (CPAUT + FARX * CPFUT)/(1. + FARX)$$

$$\frac{\partial CPEX}{\partial FARX} = \frac{CPF}{(1. + FARX)} - \frac{CPA + FARX * CPF}{(1. + FARX)^2}$$

substitute CPA = (1. + FARX) * CPEX - FARX * CPF

$$\frac{\partial CPEX}{\partial FARX} = \underline{CPEXUF} = (CPF - CPEX)/(1. + FARX)$$

Differentiate HEX with respect to TEX and FARX.

HEX = (HEA + FARX * HEF)/(1. + FARX)

By definition

$$\frac{\partial HEX}{\partial TEX} = \underline{HEXUT} = CPEX$$

and by a derivation similar to the above for CPEX

$$\frac{\partial HEX}{\partial FARX} = \underline{HEXUF} = (HEF-HEX)/(1. + FARX)$$

Differentiate PHI with respect to TEX and FARX.

PHI = (SEA + FARX * SEF)/(1. + FARX)

By definition

$$\frac{\partial PHI}{\partial TEX} = \underline{PHIUT} = CPEX/TEX$$

$$\frac{\partial PHI}{\partial FARX} = \underline{PHIUF} = (SEF - PHI)/(1. + FARX)$$

Differentiate AMW with respect to TEX and FARX

AMW = 28.97 - .946186 * FARX

$$\frac{\partial AMW}{\partial TEX} = 0$$

$$\frac{\partial AMW}{\partial FARX} = -0.946186$$

The remaining variables REX, AKEX, and CSEX are functions of variables which have been calculated. Differentiating these variables and combining base values with partial derivatives times the change in the independent variables leads to a more comlex expression. As an example, differentiate REX with respect to TEX and FARX.

REX = 1.986375/AMW = 1.986375/(28.97 - .946185 * FARX)

$$\frac{\partial REX}{\partial TEX} = 0$$

$$\frac{\partial REX}{\partial FARX} = \left(\frac{dREX}{dAMW}\right)\left(\frac{dAMW}{dFARX}\right)$$

$$\frac{\partial REX}{\partial FARX} = \frac{-1.986375}{(AMW)^2} \frac{dAMW}{dFARX} = \frac{1.986375 * 0.946186}{(AMW)^2}$$

Then

$$REX = [REX]_{base} + 1.879480*(FARX - [FARX]_{base})/(AMW)^2$$

which is more complex than

REX = 1.986375/AMW

In order to calculate the updated information it is necessary to save the following data when a base point is being calculated.

HIST(INH) = FARX

HIST(INH+1) = TEX

HIST(INH+2) = CPEX

HIST(INH+3) = PHI

HIST(INH+4) = HEX

HIST(INH+5) = AMW

HIST(INH+6) = CPEXUT

HIST(INH+7) = CPEXUF

HIST(INH+8) = HEXUF

HIST(INH+9) = PHIUF

HIST(INH+10) = LOOPER

The updated data are then calculated by the following equations:

$DELF = FARX - [FARX]_{base}$

$DELT = TEX - [TEX]_{base}$

$CPEX = [CPEX]_{base} + CPEXUT * DELT + CPEXUF * DELF$

$HEX = [HEX]_{base} + [CPEX]_{base} * DELT + HEXUF * DELF$

$PHI = [PHI]_{base} + [CPEX]_{base} * DELT/[TEX]_{base} + PHIUF * DELF$

$AMW = [AMW]_{base} - 0.946186 * DELF$

It should be noted that setting CPEXUT and CPEXUF to zero results in sufficient accuracy to efficiently balance an AEG cycle calculation. There are two listings of PROCOM showing the current version (Table 1) and the version with the analytical derivative logic included (Table 2). In the analytical derivative version, logic is included which will assign the correct storage space to each call to PROCOM and which will choose the proper path depending upon whether the calculation is a base or a derivative point. Logic must be added to the ENGBAL program to set the indicator IBASE to the correct value depending on whether a base or a derivative point is being calculated. The analytical derivative version assumes that the variable LOOPER is increased on every path through the engine cycle calculation and that IBASE will be set equal to LOOPER whenever a base point calculation is made. In addition, the

Table 1. Listing of Subroutine PROCOM (Thermodynamic Property Calculation).

```
      SUBROUTINE PROCOM(FARX,TEX,CSEX,AKEX,CPEX,REX,PHI,HEX)
      IF(FARX.LE.0.067623) GO TO 1
      FARX=0.067623
      WRITE(8,101)
    1 IF(TEX.GE.300.) GO TO 2
      TEX=300.
      WRITE(8,102)
    2 IF(TEX.LE.4000.) GO TO 3
      TEX=4000.
      WRITE(8,103)
    3 IF(FARX.GE.0.0) GO TO 4
      FARX=0.0
      WRITE(8,104)
C     AIR PATH
    4 CPA =((((((1.0115540E-25*TEX-1.4526770E-21)*TEX
     X +7.6215767E-18)*TEX-1.5128259E-14)*TEX-6.7178376E-12)*TEX
     X +6.5519486E-08)*TEX-5.1536879E-05)*TEX+2.5020051E-01
      HEA=(((((((1.2644425E-26*TEX-2.0752522E-22)*TEX
     X +1.2702630E-18)*TEX-3.0256518E-15)*TEX-1.6794594E-12)*TEX
     X +2.1839826E-08)*TEX-2.5768440E-05)*TEX+2.5020051E-01)*TEX
     X -1.7558886E+00
      SEA=+2.5020051E-01*ALOG(TEX)+((((((1.4450767E-26*TEX
     X -2.4211288E-22)*TEX+1.5243153E-18)*TEX-3.7820648E-15)*TEX
     X -2.2392790E-12)*TEX+3.2759743E-08)*TEX-5.1576879E-05)*TEX
     X +4.5432300E-02
      IF(FARX.LE.0.0) GO TO 5
C     FUEL/AIR PATH
      CPF =((((((7.2678710E-25*TEX-1.3335668E-20)*TEX
     X +1.0212913E-16)*TEX-4.2051104E-13)*TEX+9.9686793E-10)*TEX
     X -1.3771901E-06)*TEX+1.2258630E-03)*TEX+7.3816638E-02
      HEF=(((((((9.0848388E-26*TEX-1.9050949E-21)*TEX
     X +1.7021525E-17)*TEX-8.4102208E-14)*TEX+2.4921698E-10)*TEX
     X -4.5906332E-07)*TEX+6.1293150E-04)*TEX+7.3816638E-02)
     X *TEX+3.0581530E+01
      SEF=+7.3816638E-02*ALOG(TEX)+((((((1.0382670E-25*TEX
     X -2.2226118E-21)*TEX+2.0425826E-17)*TEX-1.0512776E-13)*TEX
     X +3.3228928E-10)*TEX-6.8859505E-07)*TEX+1.2258630E-03)*TEX
     X +6.483398E-01
    5 CPEX=(CPA+FARX*CPF)/(1.+FARX)
      HEX=(HEA+FARX*HEF)/(1.+FARX)
      PHI=(SEA+FARX*SEF)/(1.+FARX)
      AMW=28.97-.946186*FARX
      REX=1.986375/AMW
      AKEX=CPEX/(CPEX-REX)
      CSEX=SQRT(AKEX*REX*TEX*25031.37)
      RETURN
  101 FORMAT(1H0,63HINPUT FUEL-AIR RATIO ABOVE LIMITS, IT HAS BEEN RESET
     X TO 0.067623,6HSSSSSS)
  102 FORMAT(1H0,35HPROCOM INPUT TEMPERATURE BELOW 300.,6HSSSSSS)
  103 FORMAT(1H0,36HPROCOM INPUT TEMPERATURE ABOVE 4000.,6HSSSSSS)
  104 FORMAT(1H0,38HPROCOM INPUT FUEL-AIR RATIO BELOW ZERO,6HSSSSSS)
      END
```

Table 2. Sample Listing of Subroutine PROCOM (with Analytic Derivatives) (Subroutine Not Checked Out).

```
      SUBROUTINE PROCOM(FARX,TEX,CSEX,AKEX,CPEX,REX,PHI,HEX,INH)
      COMMON/HISTRG/IBASE,INHS,LOOPER,HIST(891),INHI(81)
      DIMENSION IHIST(891)
      EQUIVALENCE (HIST(1),IHIST(1))
      IF(INH.NE.0) GO TO 10
      INH=INHS
      INHS=INHS+11
   10 IF(IHIST(INH+10).NE.IBASE) GO TO 20
      DELF=FARX-HIST(INH)
      DELT=TEX-HIST(INH+1)
      CPEX=HIST(INH+2)+HIST(INH+6)*DELT+HIST(INH+7)*DELF
      HEX=HIST(INH+4)+HIST(INH+2)*DELT+HIST(INH+8)*DELF
      PHI=HIST(INH+3)+HIST(INH+2)*DELT/HIST(INH+1)
     X +HIST(INH+9)*DELF
      AMW=HIST(INH+5)-0.946186*DELF
      GO TO 6
   20 IF(FARX.LE.0.067623) GO TO 1
      FARX=0.067623
      WRITE(8,101)
    1 IF(TEX.GE.300.) GO TO 2
      TEX=300.
      WRITE(8,102)
    2 IF(TEX.LE.4000.) GO TO 3
      TEX=4000.
      WRITE(8,103)
    3 IF(FARX.GE.0.0) GO TO 4
      FARX=0.0
      WRITE(8,104)
C     AIR PATH
    4 CPA =((((((1.0115540E-25*TEX-1.4526770E-21)*TEX
     X +7.6215767E-18)*TEX-1.5128259E-14)*TEX-6.7178376E-12)*TEX
     X +6.5519486E-08)*TEX-5.1536879E-05)*TEX+2.5020051E-01
      CPAUT =(((((7.*1.0115540E-25*TEX-6.*1.4526770E-21)*TEX
     X +5.*7.6215767E-18)*TEX-4.*1.5128259E-14)*TEX-3.*6.7178376E-12)
     X *TEX+2.*6.5519486E-08)*TEX-5.1536879E-05
      HEA=(((((((1.2644425E-26*TEX-2.0752522E-22)*TEX
     X +1.2702630E-18)*TEX-3.0256518E-15)*TEX-1.6794594E-12)*TEX
     X +2.1839826E-08)*TEX-2.5768440E-05)*TEX+2.5020051E-01)*TEX
     X -1.7558886E+00
      SEA=+2.5020051E-01*ALOG(TEX)+(((((((1.4450767E-26*TEX
     X -2.4211288E-22)*TEX+1.5243153E-18)*TEX-3.7820648E-15)*TEX
     X -2.2392790E-12)*TEX+3.2759743E-08)*TEX-5.1576879E-05)*TEX
     X +4.5432300E-02
      IF(FARX.LE.0.0) GO TO 5
C     FUEL/AIR PATH
      CPF =((((((7.2678710E-25*TEX-1.3335668E-20)*TEX
     X +1.0212913E-16)*TEX-4.2051104E-13)*TEX+9.9686793E-10)*TEX
     X -1.3771901E-06)*TEX+1.2258630E-03)*TEX+7.3816638E-02
      CPFUT =(((((7.*7.2678710E-25*TEX-6.*1.3335668E-20)*TEX
     X +5.*1.0212913E-16)*TEX-4.*4.2051104E-13)*TEX+3.*9.9686793E-10)
     X *TEX-2.*1.3771901E-06)*TEX+1.2258630E-03
      HEF=(((((((9.0848388E-26*TEX-1.9050949E-21)*TEX
     X +1.7021525E-17)*TEX-8.4102208E-14)*TEX+2.4921698E-10)*TEX
     X -4.5906332E-07)*TEX+6.1293150E-04)*TEX+7.3816638E-02)
     X *TEX+3.0581530E+01
      SEF=+7.3816638E-02*ALOG(TEX)+(((((((1.0382670E-25*TEX
     X -2.2226118E-21)*TEX+2.0425826E-17)*TEX-1.0512776E-13)*TEX
```

Table 2. Sample Listing of Subroutine PROCOM (with Analytic Derivatives) (Subroutine Not Checked Out)(Concluded).

```
     X +3.3228928E-10)*TEX-6.8859505E-07)*TEX+1.2258630E-03)*TEX
     X +6.483398E-01
   5  DUM=1.0+FARX
      CPEX=(CPA+FARX*CPF)/DUM
      CPEXUT=(CPAUT+FARX*CPFUT)/DUM
      CPEXUF=(CPF-CPEX)/DUM
      HEX=(HEA+FARX*HEF)/DUM
      HEXUF=(HEF-HEX)/DUM
C      HEXUT=CPEX
      PHI=(SEA+FARX*SEF)/DUM
      PHIUF=(SEF-PHI)/DUM
C       PHIUT=CPEX/TEX
      AMW=28.97-.946186*FARX
      HIST(INH)=FARX
      HIST(INH+1)=TEX
      HIST(INH+2)=CPEX
      HIST(INH+3)=PHI
      HIST(INH+4)=HEX
      HIST(INH+5)=AMW
      HIST(INH+6)=CPEXUT
      HIST(INH+7)=CPEXUF
      HIST(INH+8)=HEXUF
      HIST(INH+9)=PHIUF
      IHIST(INH+10)=LOOPER
   6  REX=1.986375/AMW
      AKEX=CPEX/(CPEX-REX)
      CSEX=SQRT(AKEX*REX*TEX*25031.37)
      RETURN
 101  FORMAT(1H0,63HINPUT FUEL-AIR RATIO ABOVE LIMITS, IT HAS BEEN RESET
     X TO 0.067623,6H$$$$$$)
 102  FORMAT(1H0,35HPROCOM INPUT TEMPERATURE BELOW 300.,6H$$$$$$)
 103  FORMAT(1H0,36HPROCOM INPUT TEMPERATURE ABOVE 4000.,6H$$$$$$)
 104  FORMAT(1H0,38HPROCOM INPUT FUEL-AIR RATIO BELOW ZERO,6H$$$$$$)
      END
```

call sequence to PROCOM must be increased to contain a unique history array pointer. Since there are 81 possible calls to PROCOM, the history storage array (HIST) is dimensioned by 891 (81*11) and the history pointer array (INHI) by 81.

There is added logic at the beginning of the subroutine PROCOM which determines if that specific call to PROCOM had been executed previously and if not, sets the storage indicator (INH) and updates the history counter INHS by 11. This logic has been added to ensure that the first time that each call to PROCOM is executed the history storage array for that storage indicator has been filled with correct data.

This alternate approach was incorporated into the VCE cycle and the performance for the 411 approved flight operating points was calculated. The results showed that this approach used 47% less computer processor time than the original computer program but required 5K more computer storage. The alternate method is easily implemented using minimal time and cost.

4.0 APPLICATION OF THE METHOD TO SMOTE

4.1 CONTROL LOGIC

As a vehicle to illustrate the application of the method using the complete analytic derivative set, the Air Force engine cycle simulation program was selected. This program, titled SMOTE (Simulation of Turbofan Engine) was developed in the Components Branch, Turbine Engine Division, Air Force Aero Propulsion Laboratory and is described in References 2 and 3. A simultaneous Newton-Raphson iteration method is used to calculate balanced cycle performance. The partial derivatives are obtained by numerical differentiation. A matrix of differential error equations is then solved to determine the correct values of the independent variables which would produce zero errors. A flow chart of the program is shown in Figure 2.

As an example of the method, the derivation of the analytical derivatives and the associated computer program logic for a compressor and a combustor will be shown. It should be pointed out that none of the Fortran IV subroutine listings referred to in this section have been checked out. The listings are intended as an aid in understanding the examples, not as executable code. Control of the SMOTE program is contained in the main subroutine ENGBAL. This subroutine controls all engine balancing loops, checks tolerances and number of loops, and loads the matrix. The functioning of this subroutine is shown schematically in Figure 2, and a listing of the subroutine is given in Table 3. The incorporation of analytic derivatives would necessitate the rewriting of the ENGBAL subroutine. The new flowpath would be similar to that shown in Figure 3. In Figure 3, the differentials of the independent variables are designated by the symbol VU with an appropriate subscript. These differentials are incremented sequentially (from 1 to 6) by unity. When the indicator IDERV = 1, the deck is on a derivative path. In this case, the subroutine base point logic (the block labeled ENGINE) is jumped and only the derivatives are calculated in the individual subroutines. The errors can be calculated from the equation:

$$\text{ERR(I)} = \text{ERR(I)}\Big)_{\substack{\text{BASE}\\ \text{POINT}}} + d\,(\text{ERR(I)})$$

The error differentials are obtained by differentiating the individual error definitions. The unknown terms will then represent derivatives and/or differentials evaluated in the individual subroutine.

For example, Table 4 shows the independent and dependent variables (errors) for the SMOTE deck. The differentiation of the first error would proceed as follows:

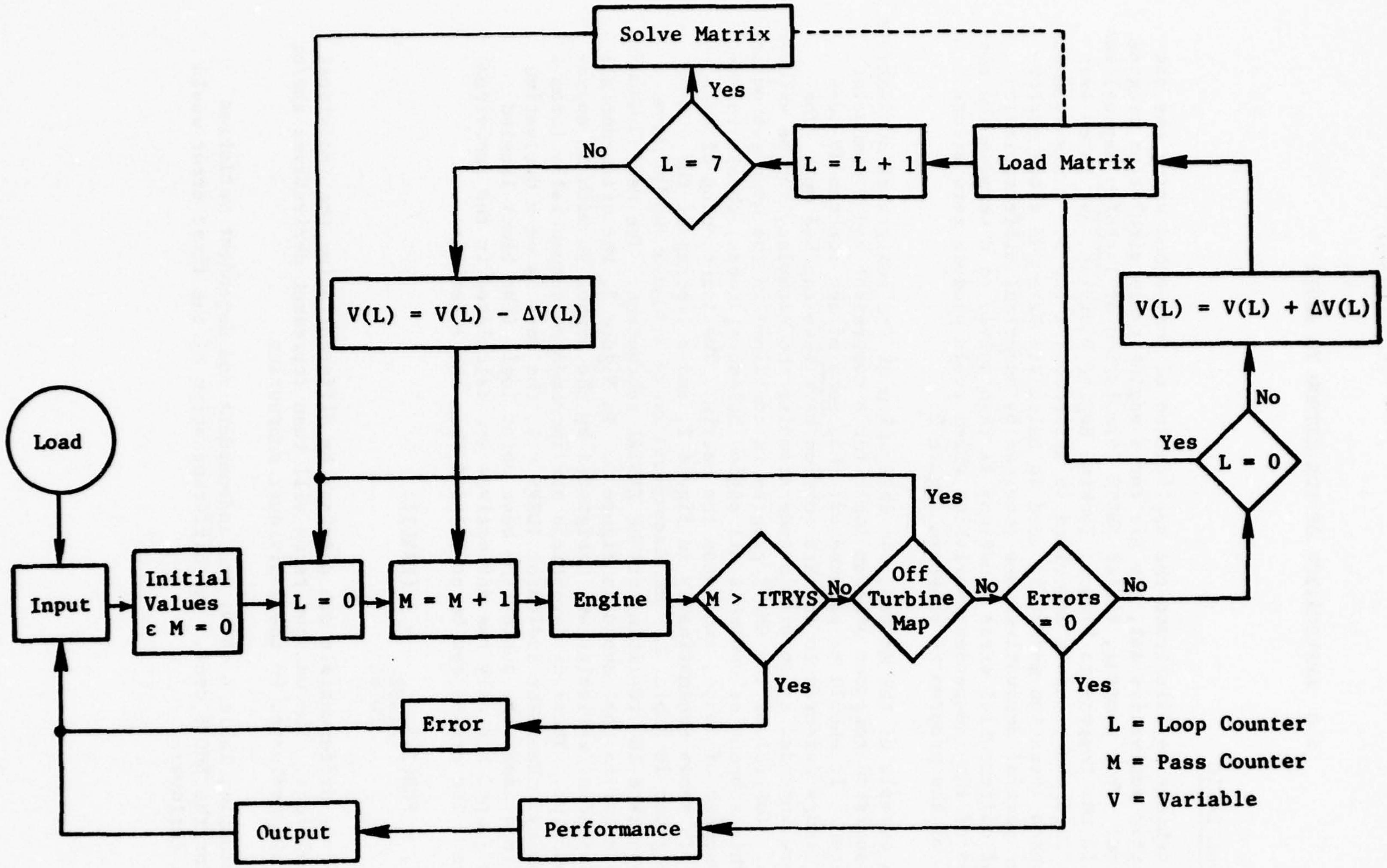


Figure 2. SMOTE Computer Program Flow Chart.

Table 3. Listing of Subroutine ENGBAL.

```
      SUBROUTINE ENGBAL
      COMMON /  ALL/
     1WORD  ,IDFS  ,JDFS  ,KDFS  ,MODE  ,INIT  ,IDUMP ,IAMTP ,
     2IGASMX,IDBURN,IAFTBN,IDCD  ,IMCD  ,IDSHOC,IMSHOC,NOZFLT,
     3ITRYS ,LOOPER,NOMAP ,NUMMAP,MAPEDG,TOLALL,ERR(6)
      COMMON /DESIGN/
     1PCNFGU,PCNCGU,T4GU  ,DUMD1 ,DUMD2 ,DELFG ,DELFN ,DELSFC,
     2ZFDS  ,PCNFDS,PRFDS ,ETAFDS,WAFDS ,PRFCF ,ETAFCF,WAFCF ,
     3ZCDS  ,PCNCDS,PRCDS ,ETACDS,WACDS ,PRCCF ,ETACCF,WACCF ,
     4T4DS  ,WFBDS ,DTCODS,ETABDS,WA3CDS,DPCODS,DTCOCF,ETABCF,
     5TFHPDS,CNHPDS,ETHPDS,TFHPCF,CNHPCF,ETHPCF,DHHPCF,T2DS  ,
     6TFLPDS,CNLPDS,ETLPDS,TFLPCF,CNLPCF,ETLPCF,DHLPCF,T21DS ,
     7T24DS ,WFDDS ,DTDUDS,ETADDS,WA23DS,DPDUDS,DTDUCF,ETADCF,
     8T7DS  ,WFADS ,DTAFDS,ETAADS,WG6CDS,DPAFDS,DTAFCF,ETAACF,
     9A55   ,A25   ,A6    ,A7    ,A8    ,A9    ,A28   ,A29   ,
     APS55  ,AM55  ,CVDNOZ,CVMNOZ,A8SAV ,A9SAV ,A28SAV,A29SAV
      COMMON / FRONT/
     1T1    ,P1    ,H1    ,S1    ,T2    ,P2    ,H2    ,S2    ,
     2T21   ,P21   ,H21   ,S21   ,T3    ,P3    ,H3    ,S3    ,
     3T4    ,P4    ,H4    ,S4    ,T5    ,P5    ,H5    ,S5    ,
     4T55   ,P55   ,H55   ,S55   ,BLF   ,BLC   ,BLDU  ,BLOB  ,
     5CNF   ,PRF   ,ETAF  ,WAFC  ,WAF   ,WA3   ,WG4   ,FAR4  ,
     6CNC   ,PRC   ,ETAC  ,WACC  ,WAC   ,ETAB  ,DPCOM ,DUMF  ,
     7CNHP  ,ETATHP,DHTCHP,DHTC  ,BLHP  ,WG5   ,FAR5  ,CS    ,
     8CNLP  ,ETATLP,DHTCLP,DHTF  ,BLLP  ,WG55  ,FAR55 ,HPEXT ,
     9AM    ,ALTP  ,ETAR  ,ZF    ,PCNF  ,ZC    ,PCNC  ,WFB   ,
     ATFFHP ,TFFLP ,PCBLF ,PCBLC ,PCBLDU,PCBLOB,PCBLHP,PCBLLP
      COMMON /  SIDE/
     XXP1   ,XWAF  ,XWAC  ,XBLF  ,XBLDU ,XH3   ,DUMS1 ,DUMS2 ,
     XXT21  ,XP21  ,XH21  ,XS21  ,T23   ,P23   ,H23   ,S23   ,
     3T24   ,P24   ,H24   ,S24   ,T25   ,P25   ,H25   ,S25   ,
     4T28   ,P28   ,H28   ,S28   ,T29   ,P29   ,H29   ,S29   ,
     5WAD   ,WFD   ,WG24  ,FAR24 ,ETAD  ,DPDUC ,BYPASS,DUMS3 ,
     6TS28  ,PS28  ,V28   ,AM28  ,TS29  ,PS29  ,V29   ,AM29
      COMMON /  BACK/
     XXT55  ,XP55  ,XH55  ,XS55  ,XT25  ,XP25  ,XH25  ,XS25  ,
     XXWFB  ,XWG55 ,XFAR55,XWFD  ,XWG24 ,XFAR24,XXP1  ,DUMB  ,
     3T6    ,P6    ,H6    ,S6    ,T7    ,P7    ,H7    ,S7    ,
     4T8    ,P8    ,H8    ,S8    ,T9    ,P9    ,H9    ,S9    ,
     5WG6   ,WFA   ,WG7   ,FAR7  ,ETAA  ,DPAFT ,V55   ,V25   ,
     6PS6   ,V6    ,AM6   ,TS7   ,PS7   ,V7    ,AM7   ,AM25  ,
     7TS8   ,PS8   ,V8    ,AM8   ,TS9   ,PS9   ,V9    ,AM9   ,
     8VA    ,FRD   ,VJD   ,FGMD  ,VJM   ,FGMM  ,FGPD  ,FGPM  ,
     9FGM   ,FGP   ,WFT   ,WGT   ,FART  ,FG    ,FN    ,SFC
      DIMENSION VAR(6),DEL(6),ERRB(6),DELVAR(6),EMAT(6,6),VMAT(6),
     1AMAT(6)
      DATA AWORD/6HENGBAL/
      DATA VDELTA,VLIM,VCHNGE,NOMISX/
     1 0.001,0.100,0.850,4/
      CALL INPUT
      IF(INIT.EQ.1) GO TO 50
      TFFHP=TFHPDS
      TFFLP=TFLPDS
   50 LOOPER=0
      NUMMAP=0
      NOMISS=0
    1 LOOP=0
```

Table 3. Listing of Subroutine ENGBAL (Continued).


```
      MISMAT=0
      NOMAP=0
      IGO=2
      DO 2 I=1,6
      VMAT(I)=0.
      AMAT(I)=0.
      DELVAR(I)=0.
      DO 2 L=1,6
2     EMAT(I,L)=0.
3     LOOPER=LOOPER+1
      CALL COFAN
      WORD=AWORD
      IF(LOOPER.GT.ITRYS) GO TO 20
      IF(NOMAP.GT.0) GO TO 1
      NUMMAP=0
55    VAR(1)=ZF*100.
      IF(MODE.NE.3) VAR(2)=PCNF
      IF(MODE.EQ.3) VAR(2)=T4/10.
      VAR(3)=ZC*100.
      IF(MODE.NE.1) VAR(4)=PCNC
      IF(MODE.EQ.1) VAR(4)=T4/10.
      VAR(5)=TFFHP
      VAR(6)=TFFLP
      DO 4 I=1,6
      IF(ABS(ERR(I)).GT.TOLALL) GO TO 5
    4 CONTINUE
      CALL PERF
      CALL ERROR
5     IF(LOOP.GT.0) GO TO 7
      MAPEDG=0
      MAPSET=0
      DO 6 I=1,6
      ERRB(I)=ERR(I)
6     DEL(I)=VDELTA*VAR(I)
      GO TO 9
7     IF(MISMAT.GT.0) GO TO 30
      IF(MAPEDG.EQ.0) GO TO 70
      MAPEDG=0
      MAPSET=1
      VAR(LOOP)=VAR(LOOP)+2.*DEL(LOOP)
      GO TO 10
70    IF(MAPSET.EQ.0) VAR(LOOP)=VAR(LOOP)+DEL(LOOP)
      IF(MAPSET.EQ.1) VAR(LOOP)=VAR(LOOP)-DEL(LOOP)
      MAPSET=0
      DO 8 I=1,6
8     EMAT(I,LOOP)=(ERRB(I)-ERR(I))/DEL(LOOP)
9     LOOP=LOOP+1
      IF(LOOP.GT.6) GO TO 11
      VAR(LOOP)=VAR(LOOP)-DEL(LOOP)
10    ZF=VAR(1)/100.
      IF(MODE.NE.3) PCNF=VAR(2)
      IF(MODE.EQ.3) T4=VAR(2)*10.
      ZC=VAR(3)/100.
      IF(MODE.NE.1) PCNC=VAR(4)
      IF(MODE.EQ.1) T4=VAR(4)*10.
      TFFHP=VAR(5)
      TFFLP=VAR(6)
```

Table 3. Listing of Subroutine ENGBAL (Continued).

```
      IF(ZF.LT.0.) ZF=0.05
      IF(ZC.LT.0.) ZC=0.05
      GO TO (1,3),IGO
11    DO 12 I=1,6
12    AMAT(I)=-ERRB(I)
      DO 14 I=1,6
      IZERO=0
      DO 13 LOOP=1,6
13    IF(EMAT(I,LOOP).EQ.0.) IZERO=IZERO+1
      IF(IZERO.LT.6) GO TO 14
      WRITE(6,100)I
      LOOPER=ITRYS+100
      GO TO 20
14    CONTINUE
      DO 16 LOOP=1,6
      IZERO=0
      DO 15 I=1,6
15    IF(EMAT(I,LOOP).EQ.0.) IZERO=IZERO+1
      IF(IZERO.LT.6) GO TO 16
      WRITE(6,101)LOOP
      LOOPER=ITRYS+100
      GO TO 20
16    CONTINUE
17    CALL MATRIX(EMAT,VMAT,AMAT)
      LBIG=0
      VARBIG=0.0
      DO 18 L=1,6
      ABSVAR=ABS(VMAT(L))
      IF(ABSVAR.LE.VLIM*VAR(L)) GO TO 18
      IF(ABSVAR.LE.VARBIG) GO TO 18
      LBIG=L
      VARBIG=ABSVAR
18    CONTINUE
      VRATIO=1.0
      IF(LBIG.GT.0) VRATIO=VLIM*VAR(LBIG)/VARBIG
      ERRAVE=0.0
      VMTAVE=0.0
      DELAVE=0.0
      DO 19 L=1,6
      DELVAR(L)=VRATIO*VMAT(L)
      ERRAVE=ERRAVE+ABS(AMAT(L))
      VMTAVE=VMTAVE+ABS(VMAT(L))
      DELAVE=DELAVE+ABS(DELVAR(L))
19    VAR(L)=VAR(L)+DELVAR(L)
      ERRAVE=ERRAVE/6.
      VMTAVE=VMTAVE/6.
      DELAVE=DELAVE/6.
      IF(MISMAT.GT.0) GO TO 32
      IF(NOMISS.EQ.0) MISMAT=1
      IF(MISMAT.EQ.0) IGO=1
20    WRITE(8,102) LOOPER
      DO 21 I=1,6
21    WRITE(8,103) AMAT(I),(EMAT(I,L),L=1,6),VMAT(I),DELVAR(I),VAR(I)
      WRITE(8,104) ERRAVE,VMTAVE,DELAVE
22    IF(LOOPER.LT.ITRYS) GO TO 10
      CALL ERROR
      RETURN
```

Table 3. Listing of Subroutine ENGBAL (Concluded).

```
#30    VMTAVX=VMTAVE
      DO 31 I=1,6
#31    AMAT(I)=-ERR(I)
      GO TO 17
#32    WRITE(8,105) AMAT,ERRAVE,DELVAR,DELAVE,VMAT,VMTAVE,VAR
      MISMAT=MISMAT+1
      IF(VMTAVE.LT.VCHNGE*VMTAVX) GO TO 22
      WRITE(8,106)
      IF(MISMAT.LT.NOMISX) NOMISS=1
      MISMAT=0
      LOOP=0
      IGO=2
      GO TO 55
#100   FORMAT(4H0ROW,I2,16H IS ZERO IN EMAT)
#101   FORMAT(7H0COLUMN,I2,16H IS ZERO IN EMAT)
#102   FORMAT(8HB    ERRB,28X23HERROR MATRIX AFTER LOOP,I4,29X4HVMAT,
     16X6HDELVAR,7X14HVARIABLES$$$$$)
#103   FORMAT(1H0,F8.4,8X6F10.4,10XF10.4,F11.4,4XF11.4,6H$$$$$$)
#104   FORMAT(1H0,F8.4,32X14HAVERAGE VALUES,31X,2F11.4,6H$$$$$$)
#105   FORMAT(12H0-----   AMAT,7F16.6,6H$$$$$$,
     1/,      12H -----DELVAR,7F16.6,6H$$$$$$,
     2/,      12H -----   VMAT,7F16.6,6H$$$$$$,
     3/,      12H -----    VAR,6F16.6,6H$$$$$$)
#106   FORMAT(1H0,50X22HCHANGE TOO SMALL$$$$$$)
      END
```

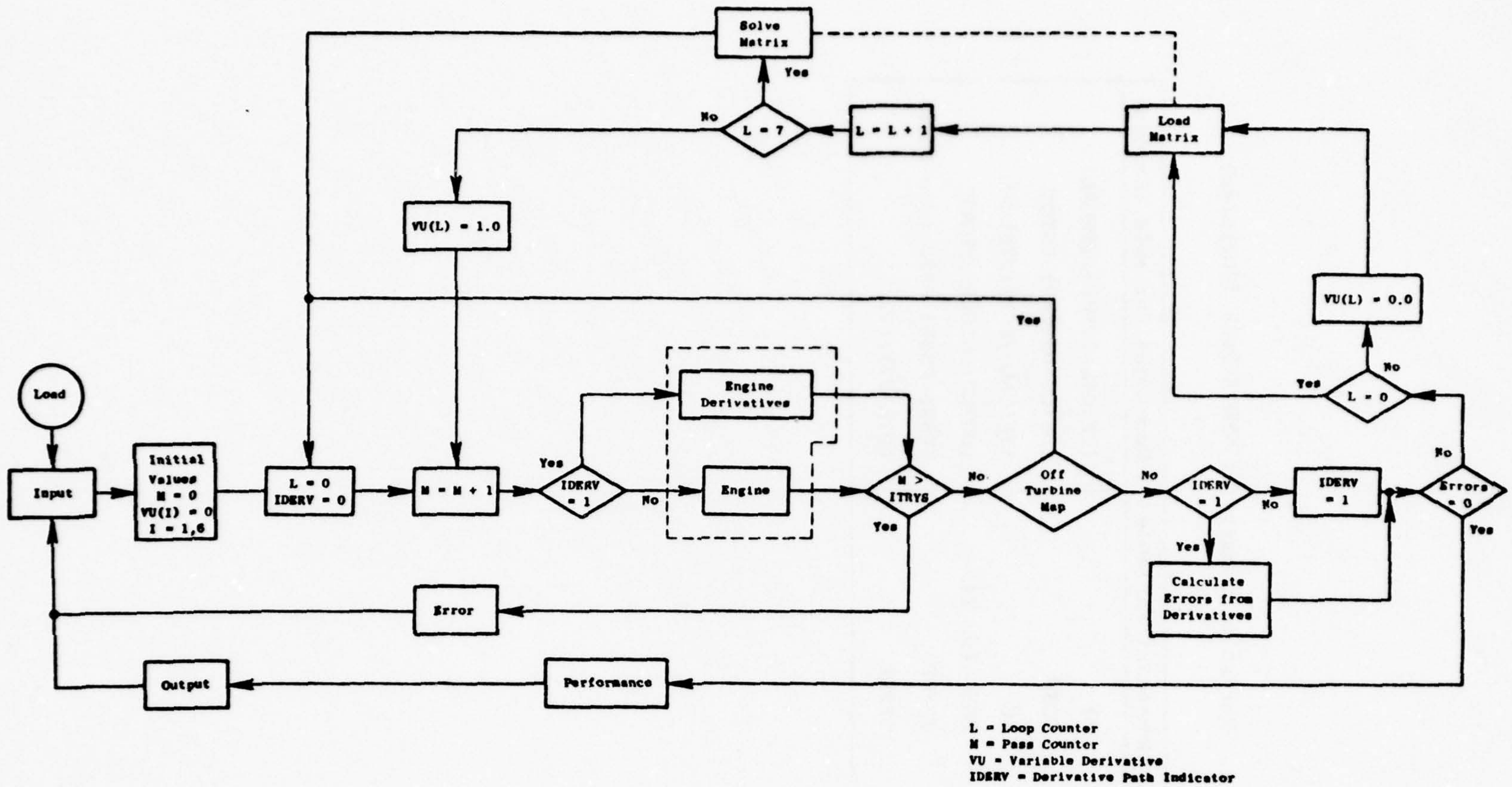


Figure 3. SMOTE Computer Program Flow Chart (with Analytical Derivatives).

Table 4. Variable Summary for SMOTE Deck (Turbofan).

Number	Independent Variable	Dependent Variable (Errors)
1	ZF	(TFHCAL-TFFHP)/TFHCAL
2	PCNF	(DHTCC-DHTCHP)/DHTCC
3	ZC	(TFLCAL-TFFLP)/TFLCAL
4	PCNC (or T4)	(DHTCF-DHTCLP)/DHTCF
5	TFFHP	(PS25-PS55)/PS25 (Sep Flow)
6	TFFLP	(P7R-P7)/P7R

$$ERR(1) = (TFHCAL - TFFHP)/TFHCAL$$

$$dERR(1) = \frac{TFHCAL * (TFHCAL - dTFFHP) - (TFHCAL - TFFHP) * d\ TFHCAL}{(TFHCAL)^2}$$

$$dERR(1) = -\frac{dTFFHP}{TFHCAL} + \frac{TFFHP * dTFHCAL}{(TFHCAL)^2}$$

$$dERR(1) = -\left(\frac{TFFHP}{TFHCAL}\right) * \left(\frac{dTFFHP}{TFFHP}\right) + \left(\frac{TFFHP}{TFHCAL}\right) * \left(\frac{dTFHCAL}{TFHCAL}\right)$$

$$dERR(1) = +\left(\frac{TFFHP}{TFHCAL}\right) * \left[\left(\frac{dTFHCAL}{TFHCAL}\right) - \left(\frac{dTFFHP}{TFFHP}\right)\right]$$

Note that all terms on the right-hand side of the above equation would be evaluated at the base point. The quantities TFFHP and TFHCAL are normally computed when the base point is calculated. The two differential terms are evaluated on the derivative path.

In general,

$$dERR(1) = \frac{\partial ERR(1)}{\partial V_1} dV_1 + \frac{\partial ERR(2)}{\partial V_2} dV_2 + \ldots + \frac{\partial ERR(n)}{\partial V_n} dV_n$$

On any given derivative path all the dV's will be zero except one whose value will be unity.

Therefore,

$$dERR(1) = \frac{\partial ERR(1)}{\partial V_1} \quad \text{after the first pass } (L = 1)$$

$$dERR(1) = \frac{\partial ERR(1)}{\partial V_2} \quad \text{after the second pass } (L = 2)$$

and $dERR(n) = \frac{\partial ERR(n)}{\partial V_n}$ after the nth (last) pass.

The subroutine ENGBAL as it might appear with the analytic derivative control logic in place is shown in Table 5. The subroutine AERR referred to in the listing would return the individual error derivatives, i.e., dERR(i) = ERRU(I). It would contain the logic for all the error term derivatives derived in a manner similar to that shown for the first error term. The base point values and the value of the derivative and/or differential terms could be communicated to the subroutine via labeled common.

Table 5. Sample Listing of Subroutine ENGBAL (with Analytical Derivative Logic) (Subroutine Not Checked Out).

```
      SUBROUTINE ENGBAL
      COMMON /   ALL/
     1WORD   ,IDFS   ,JDFS   ,KDFS   ,MODF   ,INIT   ,IDUMP  ,IAMTP  ,
     2IGASMX,IDBURN,IAFTBN,IDCD   ,IMCD   ,IDSHOC,IMSHOC,NOZFLT,
     3ITRYS  ,LOOPER,NOMAP  ,NUMMAP,MAPFDG,TOLALL,ERR(6)
      COMMON /DESIGN/
     1PCNFGU,PCNCGU,T4GU   ,DUMD1  ,DUMD2  ,DELFG  ,DELFN  ,DELSFC,
     2ZFDS   ,PCNFDS,PRFDS  ,ETAFDS,WAFDS  ,PRFCF  ,ETAFCF,WAFCF  ,
     3ZCDS   ,PCNCDS,PRCDS  ,ETACDS,WACDS  ,PRCCF  ,ETACCF,WACCF  ,
     4T4DS   ,WFBDS  ,DTCODS,ETABDS,WA3CDS,DPCODS,DTCOCF,ETABCF,
     5TFHPDS,CNHPDS,ETHPDS,TFHPCF,CNHPCF,ETHPCF,DHHPCF,T2DS   ,
     6TFLPDS,CNLPDS,ETLPDS,TFLPCF,CNLPCF,ETLPCF,DHLPCF,T21DS  ,
     7T24DS  ,WFDDS  ,DTDUDS,ETADDS,WA23DS,DPDUDS,DTDUCF,ETADCF,
     8T7DS   ,WFADS  ,DTAFDS,ETAADS,WG6CDS,DPAFDS,DTAFCF,ETAACF,
     9A55    ,A25    ,A6     ,A7     ,A8     ,A9     ,A28    ,A29    ,
     APS55   ,AM55   ,CVDNOZ,CVMNOZ,A8SAV  ,A9SAV  ,A28SAV,A29SAV
      COMMON / FRONT/
     1T1     ,P1     ,H1     ,S1     ,T2     ,P2     ,H2     ,S2     ,
     2T21    ,P21    ,H21    ,S21    ,T3     ,P3     ,H3     ,S3     ,
     3T4     ,P4     ,H4     ,S4     ,T5     ,P5     ,H5     ,S5     ,
     4T55    ,P55    ,H55    ,S55    ,BLF    ,BLC    ,BLDU   ,BLOB   ,
     5CNF    ,PRF    ,ETAF   ,WAFC   ,WAF    ,WA3    ,WG4    ,FAR4   ,
     6CNC    ,PRC    ,ETAC   ,WACC   ,WAC    ,ETAB   ,DPCOM  ,DUMF   ,
     7CNHP   ,ETATHP,DHTCHP,DHTC   ,BLHP   ,WG5    ,FAR5   ,CS     ,
     8CNLP   ,ETATLP,DHTCLP,DHTF   ,BLLP   ,WG55   ,FAR55  ,HPEXT  ,
     9AM     ,ALTP   ,ETAR   ,ZF     ,PCNF   ,ZC     ,PCNC   ,WFB    ,
     ATFFHP  ,TFFLP  ,PCBLF  ,PCBLC  ,PCBLDU,PCBLOB,PCBLHP,PCBLLP
      COMMON /  SIDE/
     XXP1    ,XWAF   ,XWAC   ,XBLF   ,XBLDU  ,XH3    ,DUMS1  ,DUMS2  ,
     XXT21   ,XP21   ,XH21   ,XS21   ,T23    ,P23    ,H23    ,S23    ,
     3T24    ,P24    ,H24    ,S24    ,T25    ,P25    ,H25    ,S25    ,
     4T28    ,P28    ,H28    ,S28    ,T29    ,P29    ,H29    ,S29    ,
     5WAD    ,WFD    ,WG24   ,FAR24  ,ETAD   ,DPDUC  ,BYPASS,DUMS3  ,
     6TS28   ,PS28   ,V28    ,AM28   ,TS29   ,PS29   ,V29    ,AM29
      COMMON /  BACK/
     XXT55   ,XP55   ,XH55   ,XS55   ,XT25   ,XP25   ,XH25   ,XS25   ,
     XXWFB   ,XWG55  ,XFAR55,XWFD   ,XWG24  ,XFAR24,XXP1   ,DUMB   ,
     3T6     ,P6     ,H6     ,S6     ,T7     ,P7     ,H7     ,S7     ,
     4T8     ,P8     ,H8     ,S8     ,T9     ,P9     ,H9     ,S9     ,
     5WG6    ,WFA    ,WG7    ,FAR7   ,ETAA   ,DPAFT  ,V55    ,V25    ,
     6PS6    ,V6     ,AM6    ,TS7    ,PS7    ,V7     ,AM7    ,AM25   ,
     7TS8    ,PS8    ,V8     ,AM8    ,TS9    ,PS9    ,V9     ,AM9    ,
     8VA     ,FRD    ,VJD    ,FGMD   ,VJM    ,FGMM   ,FGPD   ,FGPM   ,
     9FGM    ,FGP    ,WFT    ,WGT    ,FART   ,FG     ,FN     ,SFC
      COMMON /DFRIVX/Z1UX,Z2UX,Z1UY,Z2UY,Z2UZ1,CPA,CPF,CPFX,AKEX,
     &REX,HUFAR,RUFAR,PUFAR,ZFU,PCNFU,XT4,ZCU,PANCU,TFFHPU,
     &TFFLPU
      DIMENSION VAR(6),DEL(6),ERRB(6),DELVAR(6),EMAT(6,6),VMAT(6),
     1AMAT(6),VU(6)
      DATA AWORD/6HENGBAL/
      DATA VDELTA,VLIM,VCHNGE,NOMISX/
     1 0.001,0.100,0.850,4/
      CALL INPUT
      IF(INIT.EQ.1) GO TO 50
      TFFHP=TFHPDS
      TFFLP=TFLPDS
   50 LOOPER=0
```

Table 5. Sample Listing of Subroutine ENGBAL (with Analytical Derivative Logic) (Subroutine Not Checked Out) (Continued).

```
      NUMMAP=0
      NOMISS=0
#1     LOOP=0
      MISMAT=0
      NOMAP=0
      IGO=2
      DO 2 I=1,6
      VMAT(I)=0.
      AMAT(I)=0.
      VU(I)=0.0
      DELVAR(I)=0.
      DO 2 L=1,6
#2     EMAT(I,L)=0.
#3     LOOPER=LOOPER+1
      CALL COFAN
      WORD=AWORD
      IF(LOOPER.GT.ITRYS) GO TO 20
      IF(NOMAP.GT.0) GO TO 1
      NUMMAP=0
55    IDERV=0
      VU(1)=100.0*ZFU
      IF(MODE.NE.3)VU(2)=PCNFU
      IF(MODE.EQ.3)VU(2)=XT4*T4/10.0
      VU(3)=100.0*ZCU
      IF(MODE.NE.1)VU(4)=PCNCU
      IF(MODE.EQ.1)VU(4)=XT4*T4/10.0
      VU(5)=TFFHPU
      VU(6)=TFFLPU
      IF(IDERV.NE.1)GO TO 200
      CALL AERR
      DO 200 I=1,6
      ERR(I)=ERRB(I)+ERRU(I)
      GO TO 210
200   IDERV=1
210   DO 4 I=1,6
      IF(ABS(ERR(I)).GT.TOLALL) GO TO 5
4     CONTINUE
      CALL PERF
      CALL ERROR
5     IF(LOOP.GT.0) GO TO 7
      DO 6 I=1,6
      ERRB(I)=ERR(I)
      DEL(I)=1.0
      GO TO 9
7     DO 8 I=1,6
8     EMAT(I,LOOP)=(ERRB(I)-ERR(I))/DEL(LOOP)
      VU(LOOP)=0.0
9     LOOP=LOOP+1
      IF(LOOP.GT.6)GO TO 11
      VU(LOOP)=1.0
10    ZFU=VU(1)/100.0
      IF(MODE.NE.3)PCNFU=VU(2)
      IF(MODE.EQ.3)XT4=VU(2)*10.0/T4
      ZCU=VU(3)/100.0
      IF(MODE.NE.1)PCNCU=VU(4)
      IF(MODE.EQ.1)XT4=VU(4)*10.0/T4
      TFFHPU=VU(5)
```

Table 5. Sample Listing of Subroutine ENGBAL (with Analytical Derivative Logic) (Subroutine Not Checked Out) (Continued).

```
      TEFLPU=VU(6)
      IF(ZF.LT.0.) ZF=0.05
      IF(ZC.LT.0.) ZC=0.05
      GO TO (1,3),IGO
11     DO 12 I=1,6
12     AMAT(I)=-ERRB(I)
      DO 14 I=1,6
      IZERO=0
      DO 13 LOOP=1,6
13     IF(EMAT(I,LOOP).EQ.0.) IZERO=IZERO+1
      IF(IZERO.LT.6) GO TO 14
      WRITE(6,100)I
      LOOPER=ITRYS+100
      GO TO 20
14     CONTINUE
      DO 16 LOOP=1,6
      IZERO=0
      DO 15 I=1,6
15     IF(EMAT(I,LOOP).EQ.0.) IZERO=IZERO+1
      IF(IZERO.LT.6) GO TO 16
      WRITE(6,101)LOOP
      LOOPER=ITRYS+100
      GO TO 20
16     CONTINUE
17     CALL MATRIX(EMAT,VMAT,AMAT)
      LBIG=0
      VARBIG=0.0
      DO 18 L=1,6
      ABSVAR=ABS(VMAT(L))
      IF(ABSVAR.LE.VLIM*VAR(L)) GO TO 18
      IF(ABSVAR.LE.VARBIG) GO TO 18
      LBIG=L
      VARBIG=ABSVAR
18     CONTINUE
      VRATIO=1.0
      IF(LBIG.GT.0) VRATIO=VLIM*VAR(LBIG)/VARBIG
      ERRAVE=0.0
      VMTAVE=0.0
      DELAVE=0.0
      DO 19 L=1,6
      DELVAR(L)=VRATIO*VMAT(L)
      ERRAVE=ERRAVE+ABS(AMAT(L))
      VMTAVE=VMTAVE+ABS(VMAT(L))
      DELAVE=DELAVE+ABS(DELVAR(L))
19     VAR(L)=VAR(L)+DELVAR(L)
      ERRAVE=ERRAVE/6.
      VMTAVE=VMTAVE/6.
      DELAVE=DELAVE/6.
      IF(MISMAT.GT.0) GO TO 32
      IF(NOMISS.EQ.0) MISMAT=1
      IF(MISMAT.EQ.0) IGO=1
20     WRITE(8,102) LOOPER
      DO 21 I=1,6
21     WRITE(8,103) AMAT(I),(EMAT(I,L),L=1,6),VMAT(I),DELVAR(I),VAR(I)
      WRITE(8,104) ERRAVE,VMTAVE,DELAVE
22     IF(LOOPER.LT.ITRYS) GO TO 10
      CALL ERROR
```

Table 5. Sample Listing of Subroutine ENGBAL (with Analytical Derivative Logic) (Subroutine Not Checked Out) (Concluded).

```
      RETURN
30     VMTAVX=VMTAVE
      DO 31 I=1,6
31     AMAT(I)=-ERR(I)
      GO TO 17
32     WRITE(8,105) AMAT,ERRAVE,DELVAR,DELAVE,VMAT,VMTAVE,VAR
      MISMAT=MISMAT+1
      IF(VMTAVE.LT.VCHNGE*VMTAVX) GO TO 22
      WRITE(8,106)
       IF(MISMAT.LT.NOMISX) NOMISS=1
      MISMAT=0
      LOOP=0
      IGO=2
      GO TO 55
100    FORMAT(4H0ROW,I2,16H IS ZERO IN EMAT)
101    FORMAT(7H0COLUMN,I2,16H IS ZERO IN EMAT)
102    FORMAT(8HB    ERRB,28X23HERROR MATRIX AFTER LOOP,I4,29X4HVMAT,
     16X6HDELVAR,7X14HVARIABLES$$$$$)
103    FORMAT(1H0,F8.4,8X6F10.4,10XF10.4,F11.4,4XF11.4,6H$$$$$$)
104    FORMAT(1H0,F8.4,32X14HAVERAGE VALUES,31X,2F11.4,6H$$$$$$)
105    FORMAT(12H0-----   AMAT,7F16.6,6H$$$$$$,
     1/,      12H -----DELVAR,7F16.6,6H$$$$$$,
     2/,      12H -----   VMAT,7F16.6,6H$$$$$$,
     3/,      12H -----    VAR,6F16.6,6H$$$$$$)
106    FORMAT(1H0,50X22HCHANGE TOO SMALL$$$$$$)
      END
```

4.2 MAIN COMPRESSOR SUBROUTINE DERIVATIVES

A listing of the main compressor subroutine (COCOMP) is given in Table 6. A flowpath showing the inputs and outputs required on the derivative path is shown in Figure 4. As can be seen from the figure, the inputs to the subroutine (COCOMP) are obtained from four different sources. These sources are:

1) Calculations from the upstream components. These calculations determine the derivatives of the thermodynamic properties, i.e., dP21, dT21, dH21, and dS21.

2) Inputs from the control logic (subroutine ENGBAL). This logic will increment the independent deck variables.

3) Derivatives calculated from the compressor map. The derivatives of all the dependent map variables (PRC, WACC, ETAC) with respect to the independent variables (ZC, CNC).

4) Values of thermodynamic properties and derivative values returned from the thermo-subroutines.

All of these input values could be returned either through expanded argument lists or through labeled common. The outputs from the subroutine include the derivatives of the outlet thermodynamic properties as well as the flow derivatives (including parasitic flow derivatives).

The procedures for differentiating the subroutine will now be given. The source equation to be differentiated will be given first; the differentiaton will then follow together with a FORTRAN statement of the derivative formula. A listing of the subroutine (COCOMP) with analytic derivatives is given in Table 7.

Note that in the equivalent FORTRAN statements an X prefix has been used to indicate a log derivative (i.e., XT = dT/T), a trailing U to indicate a differential (i.e., HU=dH), and an imbeded U to indicate a derivative (i.e., PRCUZC = ∂PR/∂ZC).

1) THETA = SQRT (T21/518.668) — Equation

$$\frac{dTHETA}{THETA} = \frac{1}{2}\frac{dT21}{T21}$$ — Derivative

THETAU = 0.5 * THETA * XT21 — FORTRAN Statement

THIS PAGE IS BEST QUALITY PRACTICABLE
FROM COPY FURNISHED TO DDC

Table 6. Listing of Subroutine COCOMP (Main Compressor).

```
      SUBROUTINE COCOMP
      COMMON /   AIL/
     1WORD  ,IDFS   ,JDFS   ,KDFS   ,MODE   ,INIT   ,IDUMP  ,IAMTP  ,
     2IGASMX,IDBURN,IAFTBN,IDCD   ,IMCD   ,IDSHOC,IMSHOC,NOZFLT,
     3ITRYS ,LOOPER,NOMAP  ,NUMMAP,MAPEDG,TOLALL,ERR(6)
      COMMON /DESIGN/
     1PCNFGU,PCNCGU,T4GU   ,DUMD1  ,DUMD2  ,DELFG  ,DELFN  ,DELSFC,
     2ZFDS   ,PCNFDS,PRFDS  ,ETAFDS,WAFDS  ,PRFCF  ,ETAFCF,WAFCF  ,
     3ZCDS   ,PCNCDS,PRCDS  ,ETACDS,WACDS  ,PRCCF  ,ETACCF,WACCF  ,
     4T4DS   ,WFBDS  ,DTCODS,ETABDS,WA3CDS,DPCODS,DTCOCF,ETABCF,
     5TFHPDS,CNHPDS,ETHPDS,TFHPCF,CNHPCF,ETHPCF,DHHPCF,T2DS   ,
     6TFLPDS,CNLPDS,ETLPDS,TFLPCF,CNLPCF,ETLPCF,DHLPCF,T21DS  ,
     7T24DS  ,WFDDS  ,DTDUDS,ETADDS,WA23DS,DPDUDS,DTDUCF,ETADCF,
     8T7DS   ,WFADS  ,DTAFDS,ETAADS,WG6CDS,DPAFDS,DTAFCF,ETAACF,
     9A55    ,A25    ,A6     ,A7     ,A8     ,A9     ,A28    ,A29    ,
     APS55  ,AM55   ,CVDNOZ,CVMNOZ,A8SAV  ,A9SAV  ,A28SAV,A29SAV
      COMMON / FRONT/
     1T1     ,P1     ,H1     ,S1     ,T2     ,P2     ,H2     ,S2     ,
     2T21    ,P21    ,H21    ,S21    ,T3     ,P3     ,H3     ,S3     ,
     3T4     ,P4     ,H4     ,S4     ,T5     ,P5     ,H5     ,S5     ,
     4T55    ,P55    ,H55    ,S55    ,BLF    ,BLC    ,BLDU   ,BLOB   ,
     5CNF    ,PRF    ,ETAF   ,WAFC   ,WAF    ,WA3    ,WG4    ,FAR4   ,
     6CNC    ,PRC    ,ETAC   ,WACC   ,WAC    ,ETAB   ,DPCOM  ,DUMF   ,
     7CNHP   ,ETATHP,DHTCHP,DHTC   ,BLHP   ,WG5    ,FAR5   ,CS     ,
     8CNLP   ,ETATLP,DHTCLP,DHTF   ,BLLP   ,WG55   ,FAR55  ,HPEXT  ,
     9AM     ,ALTP   ,ETAR   ,ZF     ,PCNF   ,ZC     ,PCNC   ,WFB    ,
     ATFFHP ,TFFLP  ,PCBLF  ,PCBLC  ,PCBLDU,PCBLOB,PCBLHP,PCBLLP
      COMMON / COMP/CNX(15),PRX(15,15),WACX(15,15),ETAX(15,15),
     1NCN,NPT(15)
      DIMENSION WLH(2)
      DATA AWORD,WLH/6HCOCOMP,6H (LO) ,6H (HI) /
      WORD=AWORD
      THETA=SQRT(T21/518.668)
      CNC=PCNC/(100.*THETA)
      IF(ZC.LT.0.) ZC=0.
      IF(ZC.GT.1.) ZC=1.
      CNCS=CNC
      CALL SEARCH(ZC,CNC,PRC,WACC,ETAC,
     1CNX(1),NCN,PRX(1,1),WACX(1,1),ETAX(1,1),NPT(1),15,15,IGO)
      IF(MODE.EQ.1) GO TO 1
      IF((CNC-CNCS).GT.0.0005*CNC) MAPEDG=1
1      IF(IGO.EQ.1.OR.IGO.EQ.2) WRITE(8,1000) CNCS,WLH(IGO)
1000   FORMAT(19H0* * *  CNC OFF MAP,F10.4,2XA6,11H* * *$$$$$$)
      WAC=WACC*P21/THETA
      IF(IDES.NE.1) GO TO 2
      PRCCF=(PRCDS-1.)/(PRC-1.)
      ETACCF=ETACDS/ETAC
      WACCF=WACDS/WAC
      WRITE(6,100)PRCCF,ETACCF,WACCF,T21DS
100    FORMAT(18H0COMPRESSOR DESIGN,6X8H  PRCCF=,E15.8,8H ETACCF=,E15.8,
     18H  WACCF=,E15.8,8H  T21DS=,E15.8)
2      PRC=PRCCF*(PRC-1.)+1.
      ETAC=ETACCF*ETAC
      WAC=WACCF*WAC
      CALL THCOMP(PRC,ETAC,T21,H21,S21,P21,T3,H3,S3,P3)
      IF(PCBLC.GT.0.) BLC=PCBLC*WAC
      WA3=WAC-BLC
```

Table 6. Listing of Subroutine COCOMP (Main Compressor) (Concluded).

```
      BIDU=PCBIDU*BLC
      BLOB=PCBLOB*BLC
      BIHP=PCBIHP*BLC
      BILP=PCBILP*BLC
      IF(MODE.NE.1) GO TO 3
      IF(ABS(CNC-CNCS).LE.0.001*CNCS) GO TO 4
      WRITE(8,2000)CNCS,CNC
2000   FORMAT(10H0CNC WAS= ,F15.8,11H  AND NOW= ,F15.8,
     124H  CHECK PCNC INPUT$$$$$)
      CALL ERROR
3      PCNC=100.*THETA*CNC
4      CALL COCOMB
      RETURN
      END
```

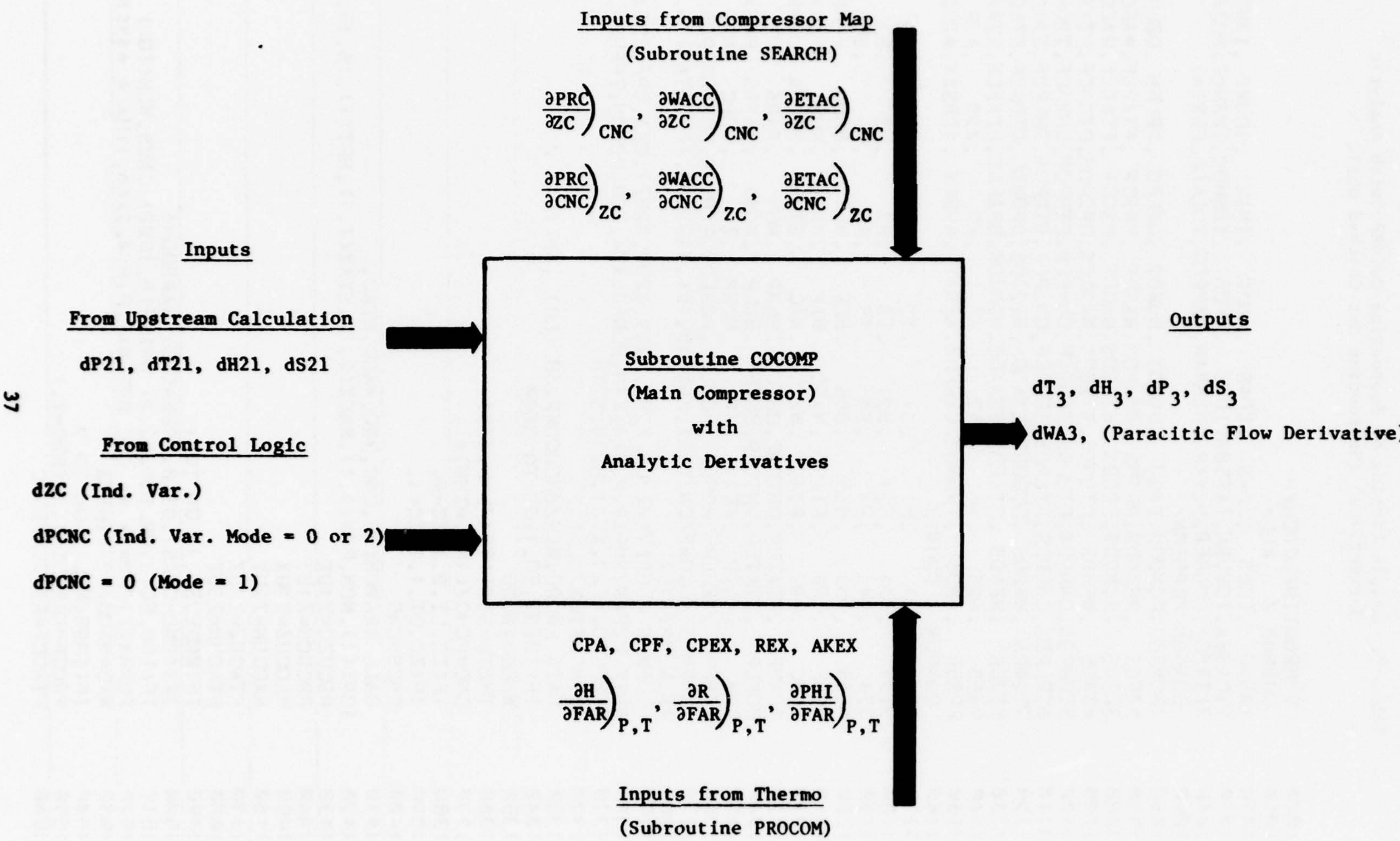


Figure 4. Flowpath for Main Compressor with Analytic Derivatives (SMOTE).

Table 7. Sample Listing of Subroutine COCOMP (with Analytic Derivatives) (Subroutine Not Checked Out).

```
      SUBROUTINE COCOMP
      COMMON /   ALL/
     1WORD  ,IDES  ,JDES  ,KDES  ,MODE  ,INIT  ,IDUMP ,IAMTP ,
     2IGASMX,IDBURN,IAFTBN,IDCD  ,IMCD  ,IDSHOC,IMSHOC,NOZFLT,
     3ITRYS ,LOOPER,NOMAP ,NUMMAP,MAPEDG,TOLALL,ERR(6)
      COMMON /DESIGN/
     1PCNFGU,PCNCGU,T4GU  ,DUMD1 ,DUMD2 ,DELFG ,DELFN ,DELSFC,
     2ZFDS  ,PCNFDS,PRFDS ,ETAFDS,WAFDS ,PRFCF ,ETAFCF,WAFCF ,
     3ZCDS  ,PCNCDS,PRCDS ,ETACDS,WACDS ,PRCCF ,ETACCF,WACCF ,
     4T4DS  ,WFBDS ,DTCODS,ETABDS,WA3CDS,DPCODS,DTCOCF,ETABCF,
     5TFHPDS,CNHPDS,ETHPDS,TFHPCF,CNHPCF,ETHPCF,DHHPCF,T2DS  ,
     6TFLPDS,CNLPDS,ETLPDS,TFLPCF,CNLPCF,ETLPCF,DHLPCF,T21DS ,
     7T24DS ,WFDDS ,DTDUDS,ETADDS,WA23DS,DPDUDS,DTDUCF,ETADCF,
     8T7DS  ,WFADS ,DTAFDS,ETAADS,WG6CDS,DPAFDS,DTAFCF,ETAACF,
     9A55   ,A25   ,A6    ,A7    ,A8    ,A9    ,A28   ,A29   ,
     APS55 ,AM55  ,CVDNOZ,CVMNOZ,A8SAV ,A9SAV ,A28SAV,A29SAV
      COMMON / FRONT/
     1T1    ,P1    ,H1    ,S1    ,T2    ,P2    ,H2    ,S2    ,
     2T21   ,P21   ,H21   ,S21   ,T3    ,P3    ,H3    ,S3    ,
     3T4    ,P4    ,H4    ,S4    ,T5    ,P5    ,H5    ,S5    ,
     4T55   ,P55   ,H55   ,S55   ,BLF   ,BLC   ,BLDU  ,BLOB  ,
     5CNF   ,PRF   ,ETAF  ,WAFC  ,WAF   ,WA3   ,WG4   ,FAR4  ,
     6CNC   ,PRC   ,ETAC  ,WACC  ,WAC   ,ETAB  ,DPCOM ,DUMF  ,
     7CNHP  ,ETATHP,DHTCHP,DHTC  ,BLHP  ,WG5   ,FAR5  ,CS    ,
     8CNLP  ,ETATLP,DHTCLP,DHTF  ,BLLP  ,WG55  ,FAR55 ,HPEXT ,
     9AM    ,ALTP  ,ETAR  ,ZF    ,PCNF  ,ZC    ,PCNC  ,WFB   ,
     ATFFHP ,TFFLP ,PCBLF ,PCBLC ,PCBLDU,PCBLOB,PCBLHP,PCBLLP
      COMMON / COMP/CNX(15),PRX(15,15),WACX(15,15),ETAX(15,15),
     1NCN,NPT(15)
      COMMON /DERIVX/Z1UX,Z2UX,Z1UY,Z2UY,Z2UZ1,CPA,CPF,CPEX,AKEX,
     &REX,HUFAR,RUFAR,PUFAR,ZFU,PCNFU,XT4,ZCU,PCNCU,TFFHPU,
     &TFFLPU,XP21,XT21,H21U,S21U
      DIMENSION WLH(2)
      DATA AWORD,WLH/6HCOCOMP,6H (LO) ,6H (HI) /
      IF(IDERV.EQ.1)GO TO 3000
      WORD=AWORD
      THETA=SQRT(T21/518.668)
      CNC=PCNC/(100.*THETA)
      IF(ZC.LT.0.) ZC=0.
      IF(ZC.GT.1.) ZC=1.
      CNCS=CNC
      CALL SEARCH(ZC,CNC,PRC,WACC,ETAC,
     1CNX(1),NCN,PRX(1,1),WACX(1,1),ETAX(1,1),NPT(1),15,15,IGO)
      PRCUZC=Z1UX
      PRCUCN=Z1UY
      WACCUZ=Z2UX
      WACCUN=Z2UY
      ETACUZ=Z3UX
      ETACUN=Z3UY
      IF(MODE.EQ.1) GO TO 1
      IF((CNC-CNCS).GT.0.0005*CNC) MAPEDG=1
      IF(IGO.EQ.1.OR.IGO.EQ.2) WRITE(8,1000) CNCS,WLH(IGO)
      FORMAT(19H0* * *  CNC OFF MAP,F10.4,2XA6,11H* * *$$$$$$)
      WAC=WACC*P21/THETA
      IF(IDES.NE.1) GO TO 2
      PRCCF=(PRCDS-1.)/(PRC-1.)
      ETACCF=ETACDS/ETAC
```

Table 7. Sample Listing of Subroutine COCOMP (with Analytic Derivatives) (Subroutine Not Checked Out) (Concluded).

```
      WACCF=WACDS/WAC
      WRITE(6,100)PRCCF,ETACCF,WACCF,T21DS
      FORMAT(18H0COMPRESSOR DESIGN,6X8H  PRCCF=,E15.8,8H ETACCF=,
     1E15.8,8H  WACCF=,E15.8,8H  T21DS=,E15.8)
      PRC=PRCCF*(PRC-1.)+1
      ETAC=ETACCF*ETAC
      WAC=WACCF*WAC
      CALL THCOMP(PRC,ETAC,T21,H21,S21,P21,T3,H3,S3,P3)
      CPA3=CPA
      AKEX3=AKEX
      REX3=REX
      IF(PCBLC.GT.0.) BLC=PCBLC*WAC
      WA3=WAC-BLC
      BLDU=PCBLDU*BLC
      BLOB=PCBLOB*BLC
      BLHP=PCBLHP*BLC
      BLLP=PCBLLP*BLC
      IF(MODE.NE.1) GO TO 3
      IF(ABS(CNC-CNCS).LE.0.001*CNCS) GO TO 4
      WRITE(8,2000)CNCS,CNC
      FORMAT(10H0CNC WAS= ,E15.8,11H  AND NOW= ,F15.8,
     124H  CHECK PCNC INPUT$$$$$$)
      CALL ERROR
      PCNC=100.*THETA*CNC
      GO TO 4
C     DERIVATIVE PATH
 3000 CONTINUE
      THETAU=0.5*THETA*XT21
      CNCU=CNSC*(PCNCU/PCNC-0.5*XT21)
      PRCU=PRCCF*(PRCUZC*ZCU+PRCUCN*CNCU)
      ETACU=ETACCF*(WTACUZ*ZCU+ETACUN*CNCU)
      WACCU=WACCF*(WACCUZ*ZCU+WACCUN*CNCU)
      WACU=WACCU/WACC+XP21-THETAU/THETA
      BLCU=0.0
      IF(PCBLC.GT.0)BLCU=PCBLC*WACU
C     FLOW DERIVATIVE
      WA3U=WACU+BLCU
C     PARASITIC FLOW DERIVATIVES
      BLDUU=PCBLDU*BLCU
      BL0BU=PCBL0B*BLCU
      BLHPU=PCBLHP*BLCU
      BLLPU=PCBLLP*BLCU
C     DERIVATIVES OF EXIT THERMODYNAMIC PROPERTIES
      XP3=XP21+PRCU/PRC
      XT3=XT21+(AKEX3-1.0)/AKEX3*PRCU/(PRC-EXP(1.0/AKEX3*
     &ALOG(PRC)))-ETACU/ETAC
      H3U=CPA3*T3*XT3
      S3U=CPA3*XT3-REX3*XP3
    4 CALL COCOMB
      RETURN
      END
```

2) CNC = PCNC/(1100 * THETA) Equation

$$\frac{dCNC}{CNC} = \frac{dPCNC}{PCNC} - \frac{dTHETA}{THETA}$$

$$\frac{dCNC}{CNC} = \frac{dPCNC}{PCNC} - \frac{1}{2}\frac{dT21}{T21}$$ Derivative

CNCU = CNC * (PCNCU/PCNC - 0.5 * XT21) FORTRAN Statement

3) The following compressor map derivatives are returned from SEARCH

PRCUZC = $\partial PRC/\partial ZC)_{CNC}$ = Z1UX

PRCUCN = $\partial PRC/\partial CNC)_{ZC}$ = Z1UY

WACCUZ = $\partial WACC/\partial ZC)_{CNC}$ = Z2UX

WACCUN = $\partial WACC/\partial CNC)_{ZC}$ = Z2UY

ETACUZ = $\partial ETAC/\partial ZC)_{CNC}$ = Z3UX

ETACUN = $\partial ETAC/\partial CNC)_{ZC}$ = Z3UY

4) The total differentials are then obtained as follows:

$$dPRC = \left.\frac{\partial PRC}{\partial ZC}\right)_{CNC} dZC + \left.\frac{\partial PRC}{\partial CNC}\right)_{ZC} dCNC$$

PRCU =PRCUZC * ZCU + PRCUCN * CNCU

The efficiency and flow differentials are obtained in a similar manner.

ETACU = ETACUZ * ZCU + ETACUN * CNCU

WACCU = WACCUZ * ZCU + WACCUN * CNCU

5) WAC = WACC*P21/THETA

$$\frac{dWAC}{WAC} = \frac{dWACC}{WACC} + \frac{dP21}{P21} - \frac{dTHETA}{THETA} = \frac{dWACC}{WACC} + \frac{dP21}{P21} - 0.5\frac{dT21}{T21}$$

WACU = WAC*(WACU/WACC + XP21 - 0.5*XT21)

6) The correction (or scale) factors must now be applied to pressure ratio, efficiency, and flow.

PRCU = PRCCF * PRCU

ETACU = ETACCF * ETACU

WACU = WACCF * WACU

7) The parasitic flow derivatives are obtained as follows:

dBLC = 0

(BLCU = 0)

IF(PCBLC .GT. 0) BLCU = PCBLC * WACU

WACU = WACU - BLCU

BLDUU = PCBLDU * BLCU

BLOBU = PCBLOB * BLCU

BLHPU = PCBLHP * BLCU

BLLPU = PCBLLP * BLCU

8) The derivatives of the compressor exit thermodynamic properties can be obtained in the following manner.

PRC = P31/P21

$$\frac{dP3}{P3} = \frac{dP21}{P21} + \frac{dPRC}{PRC}$$

XP3 = XP21 + PRCU/PRC

$$T3 = T21 \left(\frac{(PR)^{\frac{\gamma-1}{\gamma}} - 1}{\eta_c} \right)$$

$$\frac{dT3}{T3} = \frac{dT21}{T21} + \frac{\frac{\gamma-1}{\gamma}\, d(PR)}{PR-PR^{1/\gamma}} - \frac{dh_c}{\eta_c}$$

$$XT3 = XT21 + \frac{(AKE3-1)/AKEX3 * PRCU}{PRC-EXP(1.0/AKEX3 * ALOG(PRC))} - \frac{ETACU}{ETAC}$$

$$dH3 = Cp\ dT3$$

H3U = CPEX3 * T3 * XT3

$$dS3 = Cp \frac{dT3}{T3} - \frac{RdP3}{P3}$$

S3U = CPEX3 * XT3 - REX3 * XP3

4.3 MAIN COMBUSTOR SUBROUTINE

A listing of the main combustor subroutine (COCOMP) is given in Table 8. A flowpath showing the inputs and outputs required on the derivative path is shown in Figure 5. As was the case with the main compressor inputs are obtained from four different sources. Note that the combustor inputs from upstream are the outputs from the main compressor subroutine. A listing of the subroutine (COCOMP) with analytic derivatives is given in Table 9.

The procedure for differentiating the subroutine will now be summarized. As was the case with the compressor the source equation will be given first followed by the derivative formula and its FORTRAN equivalent:

1. P3PSI = 14.696 * P3 (Source Equation)

   dP3PSI = 14.696 * dP3 (Derivative)

   P3PSIU = 14.696 * XP3 * P3 (Equivalent FORTRAN Equation)

2. WA3C = WAC * SQRT (T3)/P3PSI

$$\frac{dWA3C}{WA3CC} = \frac{dWAC}{WAC} + \frac{1}{2}\frac{dT3}{T3} - \frac{dP3PSI}{P3PSI}$$

   WA3CU = WA3CC * (WA3U/WA3 + 0.5* XT3 - XP3)

3. DPCOM = DPCODS * (WA3C/WA3CD)

   dDPCOM = DPCODS/WA3CDS * dWA3C

   DPCOMU = DPCODS/WA3CDS * WA3CU

4. IF(DPCOM .GT. 1) DPCOM = 1

   IF(DPCOM .EQ. 1) DPCOMU = 0.0

Table 8. Listing of Subroutine COCOMB (Main Combustor).

```
      SUBROUTINE COCOMB
      COMMON /   ALL/
     1WORD  ,IDES  ,JDES  ,KDES  ,MODE  ,INIT  ,IDUMP ,IAMTP ,
     2IGASMX,IDBURN,IAFTBN,IDCD  ,IMCD  ,IDSHOC,IMSHOC,NOZFLT,
     3ITRYS ,LOOPER,NOMAP ,NUMMAP,MAPEDG,TOLALL,ERR(6)
      COMMON /DESIGN/
     1PCNFGU,PCNCGU,T4GU  ,DUMD1 ,DUMD2 ,DELFG ,DELFN ,DELSFC,
     2ZFDS  ,PCNFDS,PRFDS ,ETAFDS,WAFDS ,PRFCF ,ETAFCF,WAFCF ,
     3ZCDS  ,PCNCDS,PRCDS ,ETACDS,WACDS ,PRCCF ,ETACCF,WACCF ,
     4T4DS  ,WFBDS ,DTCODS,ETABDS,WA3CDS,DPCODS,DTCOCF,ETABCF,
     5TFHPDS,CNHPDS,ETHPDS,TFHPCF,CNHPCF,ETHPCF,DHHPCF,T2DS  ,
     6TFLPDS,CNLPDS,ETLPDS,TFLPCF,CNLPCF,ETLPCF,DHLPCF,T21DS ,
     7T24DS ,WFDDS ,DTDUDS,ETADDS,WA23DS,DPDUDS,DTDUCF,ETADCF,
     8T7DS  ,WFADS ,DTAFDS,ETAADS,WG6CDS,DPAFDS,DTAFCF,ETAACF,
     9A55   ,A25   ,A6    ,A7    ,A8    ,A9    ,A28   ,A29   ,
     APS55  ,AM55  ,CVDNOZ,CVMNOZ,A8SAV ,A9SAV ,A28SAV,A29SAV
      COMMON / FRONT/
     1T1    ,P1    ,H1    ,S1    ,T2    ,P2    ,H2    ,S2    ,
     2T21   ,P21   ,H21   ,S21   ,T3    ,P3    ,H3    ,S3    ,
     3T4    ,P4    ,H4    ,S4    ,T5    ,P5    ,H5    ,S5    ,
     4T55   ,P55   ,H55   ,S55   ,BLF   ,BLC   ,BLDU  ,BLOB  ,
     5CNF   ,PRF   ,ETAF  ,WAFC  ,WAF   ,WA3   ,WG4   ,FAR4  ,
     6CNC   ,PRC   ,ETAC  ,WACC  ,WAC   ,ETAB  ,DPCOM ,DUMF  ,
     7CNHP  ,ETATHP,DHTCHP,DHTC  ,BLHP  ,WG5   ,FAR5  ,CS    ,
     8CNLP  ,ETATLP,DHTCLP,DHTF  ,BLLP  ,WG55  ,FAR55 ,HPEXT ,
     9AM    ,ALTP  ,ETAR  ,ZF    ,PCNF  ,ZC    ,PCNC  ,WFB   ,
     ATFFHP ,TFFLP ,PCBLF ,PCBLC ,PCBLDU,PCBLOB,PCBLHP,PCBLLP
      COMMON / COMB/PSI(15),DELT(15,15),ETA(15,15),NPS,NPT(15)
      DIMENSION Q(9),DUMBO(15,15)
      DATA AWORD/6HCOCOMB/
      WORD=AWORD
      Q(2)=0.
      Q(3)=0.
      P3PSI=14.696*P3
      WA3C=WA3*SQRT(T3)/P3PSI
      IF(IDES.EQ.1) WA3CDS=WA3C
      DPCOM=DPCODS*(WA3C/WA3CDS)
      IF(DPCOM.GT.1.) DPCOM=1.
      P4=P3*(1.-DPCOM)
#1     IF(T4.GT.3999.) T4=3999.
      IF(T4.GE.1000.) GO TO 2
      T4=1000.
      IF(MODE.EQ.1) MAPEDG=1
#2     DTCO=T4-T3
      IF(IDES.NE.1) GO TO 3
      DTCOCF=DTCODS/DTCO
#3     DTCO=DTCOCF*DTCO
      P3PSIN=P3PSI
      CALL SEARCH(-1.,P3PSIN,DTCO,ETAB,DUMMY,
     1PSI(1),NPS,DELT(1,1),ETA(1,1),DUMBO(1,1),NPT(1),15,15,IGO)
      IF(IGO.EQ.7) CALL ERROR
#4     IF(IDES.NE.1) GO TO 5
      ETABCF=ETABDS/ETAB
#5     ETAB=ETABCF*ETAB
      HV=((((((-.4594317E-19*T4)-.2034116E-15)*T4+.2783643E-11)*T4
     1+.2051501E-07)*T4-.2453116E-03)*T4-.9433296E-01)*T4+.1845537E+05
      CALL THERMO(P4,HA,T4,XX1,XX2,0,0,0,0)
```

Table 8. Listing of Subroutine COCOMB (Main Combustor) (Concluded).

```
      FAR4=(HA-H3)/(HV*ETAB)
      IF(FAR4.LT.0.) FAR4=0.
      WFBX=FAR4*WA3
      IF(MODE.NE.2) GO TO 8
      ERRW=(WFB-WFBX)/WFB
      DIR=SQRT(WFB/WFBX)
      CALL AFQUIR(Q(1),T4,ERRW,0.,20.,0.0001,DIR,T4T,IGO)
      GO TO (6,9,7),IGO
6       T4=T4T
      GO TO 1
7       CALL ERROR
8       WFB=WFBX
9       CALL THERMO(P4,H4,T4,S4,XX2,1,FAR4,0)
      WG4=WFB+WA3
      IF(IDES.EQ.1) WRITE(6,100) WA3CDS,ETABCF,DTCOCF
100     FORMAT(17H0COMBUSTOR DESIGN,7X8H WA3CDS=,E15.8,8H ETABCF=,E15.8,
     18H DTCOCF=,E15.8)
      CALL COHPTB
      RETURN
      END
```

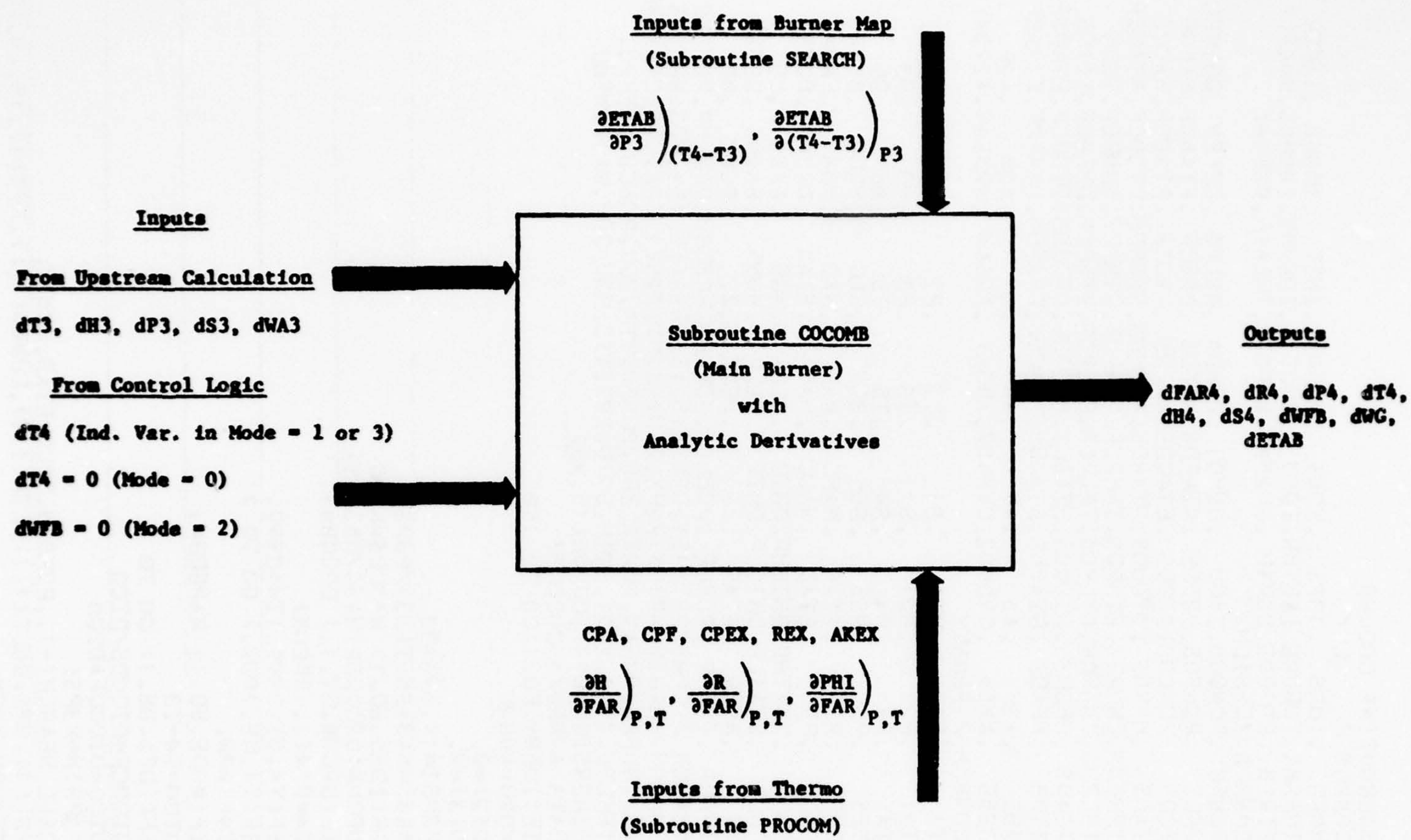


Figure 5. Flowpath for Main Combustor with Analytical Derivatives.

Table 9. **Sample Listing of Subroutine COCOMB (with Analytic Derivatives) (Subroutine Not Checked Out).**

```
      SUBROUTINE COCOMB
      COMMON /   ALL/
     1WORD  ,IDES  ,JDES  ,KDES  ,MODE  ,INIT  ,IDUMP ,IANTP ,
     2IGASMX,IDBURN,IAFTBN,IDCD  ,IMCD  ,IDSHOC,IMSHOC,NOZFLT,
     3ITRYS ,LOOPER,NOMAP ,NUMMAP,MAPEDG,TOLALL,ERR(6)
      COMMON /DESIGN/
     1PCNFGU,PCNCGU,T4GU  ,DUMD1 ,DUMD2 ,DELFG ,DELFN ,DELSFC,
     2ZFDS  ,PCNFDS,PRFDS ,ETAFDS,WAFDS ,PRFCF ,ETAFCF,WAFCF ,
     3ZCDS  ,PCNCDS,PRCDS ,ETACDS,WACDS ,PRCCF ,ETACCF,WACCF ,
     4T4DS  ,WFBDS ,DTCODS,ETABDS,WA3CDS,DPCODS,DTCOCF,ETABCF,
     5TFHPDS,CNHPDS,ETHPDS,TFHPCF,CNHPCF,ETHPCF,DHHPCF,T2DS  ,
     6TFLPDS,CNLPDS,ETLPDS,TFLPCF,CNLPCF,ETLPCF,DHLPCF,T21DS ,
     7T24DS ,WFDDS ,DTDUDS,ETADDS,WA23DS,DPDUDS,DTDUCF,ETADCF,
     8T7DS  ,WFADS ,DTAFDS,ETAADS,WG6CDS,DPAFDS,DTAFCF,ETAACF,
     9A55   ,A25   ,A6    ,A7    ,A8    ,A9    ,A28   ,A29   ,
     APS55  ,AM55  ,CVDNOZ,CVMNOZ,A8SAV ,A9SAV ,A28SAV,A29SAV
      COMMON / FRONT/
     1T1    ,P1    ,H1    ,S1    ,T2    ,P2    ,H2    ,S2    ,
     2T21   ,P21   ,H21   ,S21   ,T3    ,P3    ,H3    ,S3    ,
     3T4    ,P4    ,H4    ,S4    ,T5    ,P5    ,H5    ,S5    ,
     4T55   ,P55   ,H55   ,S55   ,BLF   ,BLC   ,BLDU  ,BLOB  ,
     5CNF   ,PRF   ,ETAF  ,WAFC  ,WAF   ,WA3   ,WG4   ,FAR4  ,
     6CNC   ,PRC   ,ETAC  ,WACC  ,WAC   ,ETAB  ,DPCOM ,DUMF  ,
     7CNHP  ,ETATHP,DHTCHP,DHTC  ,BLHP  ,WG5   ,FAR5  ,CS    ,
     8CNLP  ,ETATLP,DHTCLP,DHTF  ,BLLP  ,WG55  ,FAR55 ,HPEXT ,
     9AM    ,ALTP  ,ETAR  ,ZF    ,PCNF  ,ZC    ,PCNC  ,WFB   ,
     ATFFHP ,TFFLP ,PCBLF ,PCBLC ,PCBLDU,PCBLOB,PCBLHP,PCBLLP
      COMMON / COMB/PSI(15),DELT(15,15),ETA(15,15),NPS,NPT(15)
      COMMON /DERIVX/Z1UX,Z2UX,Z1UY,Z2UY,Z2UZ1,CPA,CPF,CPEX,AKEX,
     &REX,HUFAR,RUFAR,PUFAR,ZFU,PCNFU,XT4,ZCU,PCNCU,TFFHPU,
     &TFFLPU,XP21,XT21,H21U,S21U,XP3,XT3,H3U,S3U,WA3U,WFBU
      DIMENSION Q(9),DUMBO(15,15)
      DATA AWORD/6HCOCOMB/
      IF(IDERV.EQ.1)GO TO 150
      WORD=AWORD
      Q(2)=0.
      Q(3)=0.
      P3PSI=14.696*P3
      WA3C=WA3*SQRT(T3)/P3PSI
      IF(IDES.EQ.1) WA3CDS=WA3C
      DPCOM=DPCODS*(WA3C/WA3CDS)
      IF(DPCOM.GT.1.) DPCOM=1.
      P4=P3*(1.-DPCOM)
    1 IF(T4.GT.3999.)T4=3999.
      IF(T4.GE.1000.) GO TO 2
      T4=1000.
      IF(MODE.EQ.1) MAPEDG=1
    2 DTCO=T4-T3
      IF(IDES.NE.1) GO TO 3
      DTCOCF=DTCODS/DTCO
    3 DTCO=DTCOCF*DTCO
      P3PSIN=P3PSI
      CALL SEARCH(-1.,P3PSIN,DTCO,ETAB,DUMMY,
     1PSI(1),NPS,DELT(1,1),ETA(1,1),DUMBO(1,1),NPT(1),15,15,IGO)
      ETABUP=Z2UY
      ETABUD=Z2UZ1
      IF(IGO.EQ.7) CALL ERROR
```

Table 9. Sample Listing of Subroutine COCOMB (with Analytic Derivatives) (Subroutine Not Checked Out) (Continued).

```
 4    IF(IDES.NE.1)GO TO 5
      ETABCF=ETABDS/ETAB
 5    ETAB=ETABCF*ETAB
      HV4=((((((-.4594317E-19*T4)-.2034116E-15)*T4+.2783643E-11)*T4
     1+.2051501E-07)*T4-.2453116E-03)*T4-.9433296E-01)*T4+.1845537E+05
      CALL THERMO(P4,HA,T4,XX1,XX2,0,0,0,0)
      HA4=HA
      FAR4=(HA-H3)/(HV*ETAB)
      IF(FAR4.LT.0.) FAR4=0.
      WFBX=FAR4*WA3
      IF(MODE.NE.2) GO TO 8
      ERRW=(WFB-WFBX)/WFB
      DIR=SQRT(WFB/WFBX)
      CALL AFQUIR(Q(1),T4,ERRW,0.,20.,0.0001,DIR,T4T,IGO)
      GO TO (6,9,7),IGO
 6    T4=T4T
      GO TO 1
 7    CALL ERROR
 8    WFB=WFBX
 9    CALL THERMO(P4,H4,T4,S4,XX2,1,FAR4,0)
      WG4=WFB+WA3
      HUFAR4=HUFAR
      RUFAR4=RUFAR
      PUFAR4=PUFAR
      REX4=REX
      CPEX4=CPEX
      CPA4=CPA
      IF(IDES.EQ.1) WRITE(6,100) WA3CDS,ETABCF,DTCOCF
 100  FORMAT(17H0COMBUSTOR DESIGN,7X8H WA3CDS=,E15.8,8H ETABCF=,
     1E15.8,8H DTCOCF=,E15.8)
      GO TO 200
C       DERIVATIVE PATH
 150  P3PSIU=14.696*XP3*P3
      WA3CU=WA3C*(WA3U/WA3+0.5*XT3-XP3)
      DPCOMU=DPCOM*DPCOM*WA3CU/WA3C
      IF(DPCOM.EQ.1)DPCOMU=0.0
C     EXIT PRESSURE DERIVATIVE
      XP4=XP3-DPCOMU/(1.0-DPCOM)
      IF(T4.EQ.3999.)XT4=0.0
      IF(T4.EQ.1000.)XT4=0.0
      DTCOU=DTCOCF*(T4*XT4-T3*XT3)
C     EFFICIENCY DERIVATIVE
      ETABU=ETABCF*(ETABUP*P3PSIU+ETABUD*DTCOU)
C     FUEL HEATIN VALUE DERIVATIVE
      HVUT4=(((((-0.2756590E-18)*T4-0.1017058E-14)*T4
     &+0.1113457E-11)*T4+0.6154503E-07)*T4-0.4906232E-03)*T4
     &-0.9433296E-01
C     FUEL TO AIR RATIO DERIVATIVE
      FAR4U=FAR4*((CPA4/(HA4-H3)-HVUT4/HV4)*T4*XT4-
     &H3U/(HA4-H3)-ETAB)U/ETAB
      IF(FAR4.EQ.0)FAR4U=0.0
      IF(MODE.NE.2)GO TO 170
      FAR4U=-FAR4*WA3U/WA3
      XT4=T4*(H3U/(HA4-H3)+ETABU/ETAB)/(CPA4/(HA4-H3)-HVUT4/HV4)
 170  CONTINUE
      WFBU=FAR4*WA3U+WA3*FAR4U
      WG4U=WFBU+WA3U
```

Table 9. **Sample Listing of Subroutine COCOMB (with Analytic Derivatives) (Subroutine Not Checked Out) (Concluded).**

```
C      DERIVATIVE OF THE EXIT THERMODYNAMIC PROPERTIES
       H4U=CPEX4*T4*XT4+HUFAR4*FAR4U
       S4U=CPEX*XT4/(1.0+FAR4)-REX4*XP4+(PUFAR4-RUFAR4*
      &ALOG(P4))*FAR4U
   200 CALL COHPTB
       RETURN
       END
```

5. P4 = P3 * (1 - DPCOM)

$$\frac{dP4}{P4} = \frac{dP3}{P3} - \frac{dDPCOM}{1 - DPCOM}$$

XP4 = XP3 - DPCOMU/(1 - DPCOM)

6. IF(T4 .GT. 3999.) T4 = 3999.

IF(T4 .EQ. 3999.) XT4 = 0.0

7. IF(T4 .GE. 1000.) GO TO 2

T4 = 1000.

IF(T4 .EQ. 1000.) XT4 = 0

8. DTCO = (T4 - T3) * DTCOCF

dDTCO = (dT4 - dT3) * DTCOCF

DTCOU = DTCOCF * (T4 * XT4 - T3 * XT3)

9. The following combustor map derivatives are returned from subroutine SEARCH

$ETABUP = \partial ETAB/\partial P3PSIN)_{DTCO}$

$ETABUD = \partial ETAB/\partial DTCO)_{P3PSIN}$

$$dETAB = \frac{\partial ETAB}{\partial P3PSIN}\Big)_{DTCO} dP3PSIN + \frac{\partial ETAB}{\partial DTCO}\Big)_{P3PSIN} dDTCO$$

ETABU = ETABUP * P3PSIU + ETABUD * DTCOU

10. ETAB = ETABCF * ETAB

ETABU = ETABCF * ETABU

11. $HV4 = -0.4594317E-19 * T4^6 - 0.2034116E-15 * T4^5$
$+0.2783643E-11 * T4^4 + 0.2051501E-07 * T4^3$
$-0.2453116E-03 * T4^2 - 0.9433296E-01 * T4$
$+0.1845537E+05$

HVUT4 = (((((-0.2756590E-18) * T4 - 0.1017058E-14) * T4
+0.1113457E-11) * T4 + 0.6154503E-07) * T4
-0.4906232E-03) * T4 - -0.9433296E-01

12. FAR4 = (HA - H3)/(HV4 * ETAB)

$$\frac{dFAR4}{FAR4} = \frac{d(HA4-H3)}{HA4-H3} - \frac{d(HV4 * ETAB)}{HV4 * ETAB}$$

$$\frac{dFAR4}{FAR4} = \frac{dHA4}{HA4-H3} - \frac{dH3}{HA4-H3} - \frac{HV4 * dETAB}{HV4 * ETAB} - \frac{ETAB * dHV4}{HV4 * ETAB}$$

$$\frac{dFAR4}{FAR4} = \frac{CPA4\ dT4}{(HA4-H3)} - \frac{dH3}{(HA4-H3)} - \frac{dETAB}{ETAB} - \frac{dHV4}{dT4}\frac{dT4}{HV4}$$

FAR4U = FAR4 * (CPA4/(HA4-H3) - HVUT4/HV4) * T4 * XT4

-H3U/(HA4 - H3) - ETABU/ETAB

IF(FAR4 .EQ. 0) FAR4U = 0

13. IF(MODE .NE. 2) GO TO 170

FAR4 = WFB/WA3 (Constant WFB)

$$\frac{dFAR4}{FAR4} = -\frac{dWA3}{WA3}\left[= \frac{CPA4\ dT4}{(HA4-H3)} - \frac{dH3}{(HA4-H3)} - \frac{dETAB}{ETAB} - \frac{dHV4}{dT4}\ \frac{dT4}{Hv4}\right]$$

FAR4U = - FAR4 * WA3U/WA3

$$dT4 = \left(\frac{dH3}{(HA4-H3)} + \frac{dETAB}{ETAB} - \frac{dWA3}{WA3}\right) \Big/ \left(\frac{CPA4}{(HA4-H3)} - \frac{dHV4}{dT4}\ \frac{1}{HV4}\right)$$

XT4 = T4*(H3U/(HA4-H3)+ETABU/ETAB-WA3U/WA3)/(CPA4/(HA4-H3)-HVUT4/HV4)

14. 170 CONTINUE

WFB = WFBX = FAR4 * WA3

dWFB = FAR4 * dWA3 + WA3 * dFAR4

WFBU = FAR4 * WA3U + WA3 * FAR4U

15. WG4 = WFB + WA3

dWG4 = dWFB + dWA3

WG4U = WFBU + WA3U

16. The derivatives of the tnermodynamic properties of the exit are obtained as follows:

a) Enthalpy Derivative

H = (Ha + fHf)/(1 + f)

$$\left.\frac{\partial H}{\partial f}\right)_{P,T} = \frac{Hf - H}{1 + f} = HUFAR$$

$$dH = \left.\frac{\partial H}{\partial T}\right)_{P,f} dT + \left.\frac{\partial H}{\partial P}\right)_{T,f} dp + \left.\frac{\partial H}{\partial f}\right)_{P,T} df$$

$$\frac{dH}{dT} = \frac{1}{1+f}\frac{dHa}{dT} + \frac{f}{1+f}\frac{dHf}{dT} + e\frac{Hf-H}{1+f}\frac{df}{dT}$$

$$\frac{dH}{dT} = \frac{CPA}{1+f} + \frac{f}{1+f} CPf + \frac{Hf-H}{1+f}\frac{df}{dT}$$

$$\frac{dH}{dT} = \frac{CPA + f*CPf}{1+f} + \frac{Hf-H}{1+f}\frac{df}{dT}$$

$$dH = CpdT + \left(\frac{Hf-H}{1+f}\right) df$$

H4U = CPEX4 * T4 * XT4 + HUFAR4 * FAR4U

where HUFAR and CPEX4 are returned from the thermo subroutine PROCOM.

b) Gas Constant Derivative

AMW = 28.97 - 0.946186 * FARX

REX = 1.986375/AMW = 1.986375/(28.97-0.946186 * FAR4)

The above equations define the molecular weight and gas constant and appear in subroutine PROCOM.

$$\left.\frac{\partial REX}{\partial f}\right)_{P,T} = \frac{0.946186*REX}{AMW}$$

$$RUFAR4 = \left.\frac{\partial REX}{\partial f}\right)_{P,T}$$

where RUFAR4 is returned from thermo subroutine PROCOM

c) Entropy Derivative

$$s = \phi - R * ALOG(P) = S(P,T,f)$$

$$\text{where } \phi = \frac{\phi a + f\phi f}{1+f} = \phi(T,f)$$

$$\left.\frac{\partial\phi}{\partial f}\right)_T = \frac{\phi f - \phi}{1+f} = PUFAR$$

$$d\phi = \left.\frac{\partial\phi}{\partial T}\right)_f dT + \left.\frac{\partial\phi}{\partial f}\right)_T df$$

$$d\phi = \frac{d\phi_a + fd\phi f}{1+f} + \left.\frac{\partial\phi}{\partial f}\right)_T df$$

But $d\phi = cpdT/T$ (by definition)

$$d\phi = \frac{Cpa + fCpf}{1+f} \; \frac{dT}{T} + \left.\frac{\partial\phi}{\partial f}\right)_T df$$

$$d\phi = \frac{Cp}{1+f} \; \frac{dT}{T} + \left.\frac{\partial\phi}{\partial f}\right)_T df$$

$$dS = d\phi - \frac{Rdp}{p} - ALOG(P) \frac{\partial R}{\partial f} \; df$$

$$dS = \frac{Cp}{1+f} \; \frac{dT}{T} + \left.\frac{\partial\phi}{\partial f}\right)_T df - \frac{Rdp}{P} - \frac{\partial R}{\partial f} ALOG(P) \; df$$

S4U = CPEX4 * XT4/(1+FAR4) + PUFAR4 * FAR4U

- REX4 * XP4 - RUFAR4 * ALOG(P4) * FAR4U

where PUFAR4 is calculated in the thermo subroutine PROCOM

5.0 COST ADVANTAGES OF THE ANALYTICAL DERIVATIVE METHOD

5.1 SELECTION OF FLIGHT ENVELOPE AND MATRIX OF TEST POINTS

The engine data matrix used in the current Analytical Derivative program is shown in Figure 6. The data matrix is a modification of a similar matrix found to be useful in a typical military customer deck. It contains a total of 411 operating points. The figure shows the flight envelope with the matrix of data superimposed. Each of the locations indicted by a circle consists of seven points while those identified by a triangle consist of five points. All of the designated locations were run at the following power levels:

(1) Maximum Afterburner Power

(2) Roughly 50 Percent Afterburner Fuel Flow

(3) Intermediate (Max Dry) Power

(4) Idle

In addition, the locations indicated by triangles were run at one part power level roughly equally-spaced between Idle and Intermediate (maximum dry) power. At the locations designated by circles, three Intermediate part power levels were run so as to divide the range between Idle and Intermediate power into four approximately equally spaced intervals.

All points were run at standard day conditions, and the ram recovery was in accordance with MIL Spec 5008B.

5.2 COST COMPARISON FOR INTERNAL DECKS

Computation cost comparisons were made between runs using analytical derivatives and runs using a reference deck which had not been modified for analytical derivatives. Valid comparisons could not be made using the same deck, even though numerical derivative capability was retained in the modified deck, since the computation and storage of partial derivatives in the low-level subroutines extended the time for computing base passes significantly. The comparisons were made using the 411-point matrix of flight conditions, with both the primary (Engine 1) and alternate (Engine 2) engine models.

Processor time was measured over an interval that included all the actual engine cycle calculations, but not the program initialization or input-output processing. The analytical derivative model required 62 percent less time than the numerical derivative model using Engine 1, and

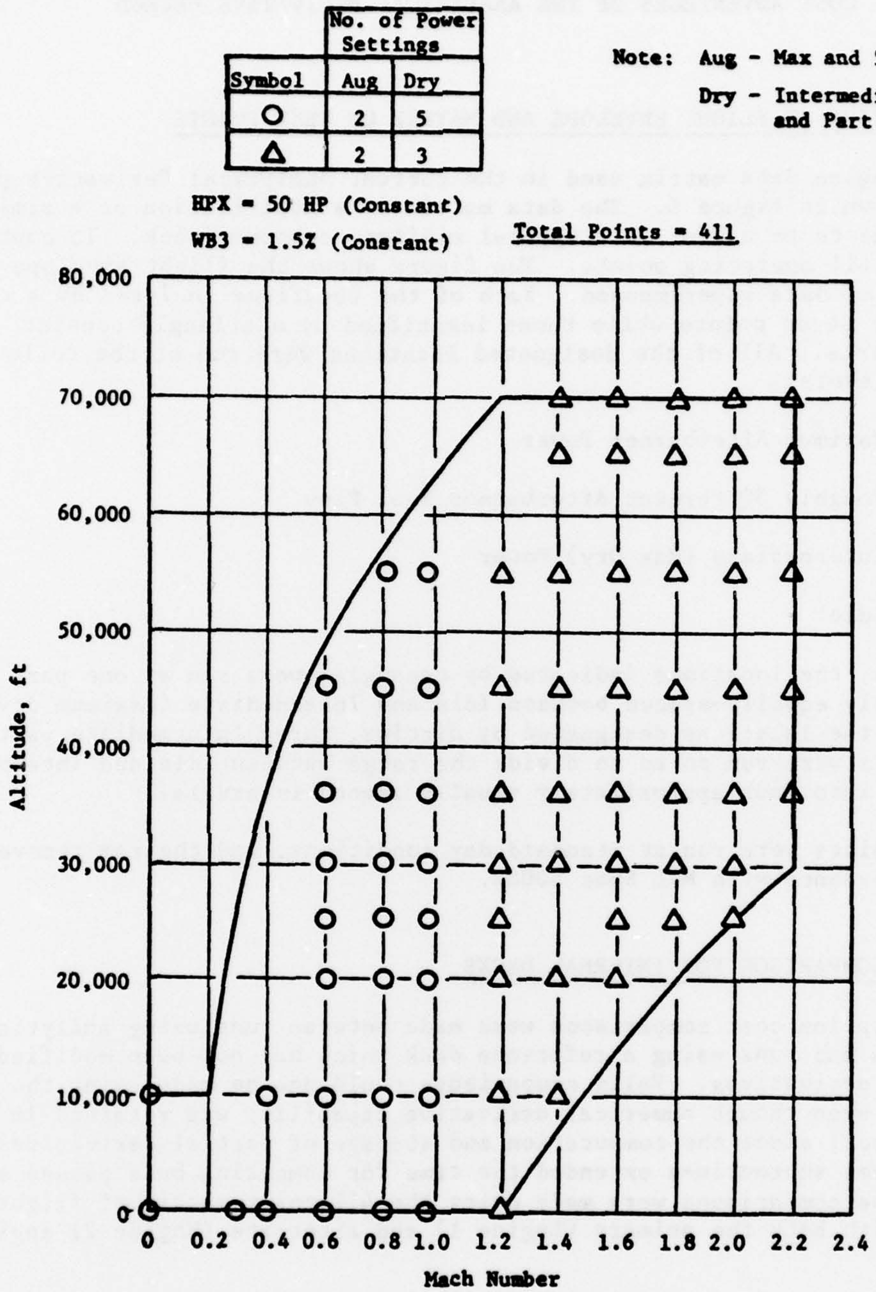


Figure 6. Engine Data Matrix.

61 percent less time using the Engine 2. Processor time is not a complete measure of the relative total cost of running the models, as the analytical derivative model requires 25% more computer memory.

The majority of the time gain (54%) was directly attributable to the inclusion of the analytic derivatives. About 8% was due to the rewriting and inclusion of a new and faster version of the subroutine COFFEX. This was a relatively new subroutine, and the coding had not been optimized. The remaining subroutines were not new and the coding had already been optimized.

A total cost comparison was made between the two models, each run from loader output files without any subroutine recompilation, and using Engine 2. The total cost of the run using the analytical derivative deck was 44% lower than the run using the numerical derivative deck (38% due to analytic derivatives and 6% due to the new subroutine COFFEX. The total cost includes not only the processor time and memory residence charges associated with performing the actual cycle calculations, but also the model initialization, input processing, and output formatting and conversion charges. The output processing cost alone has been measured to be 30% of the total cost of running the analytical derivative deck.

It is estimated that the first method (i.e., the complete set of analytic derivatives) if completely implemented internally would result in an (AEG Evendale) computations cost savings of about $100,000 annually. However, the many second order effects which are included in production cycle decks result in nonstandard logic in the higher level subroutines. For this reason, about $10,000 per deck is estimated for inclusion of the complete derivative set. In addition, continued modification of component logic would require continual updating of derivatives and a cost gain would be hard to realize.

The second method (low level derivative) would result in a cost savings of approximately $75,000/year. Moreover, it would require only about $500 in total cost per deck to implement the method. Method 2 is the one of choice for all but the most heavily used parametric customer decks.

5.3 COST COMPARISON FOR EXTERNAL (CUSTOMER) DECKS

A total cost comparison between the base deck and the deck with full analytical derivatives included was made at Wright Patterson Air Force Base on a CDC 6600/CYBER-74 computer. Both Engine 1 and Engine 2 were compared. Engine 1 showed a 51.6% savings in cost while Engine 2 showed a 52.2% savings. If these numbers are taken as fairly typical, a cost savings of about 52% should result from the inclusion of the method in an external customer deck. The difference in cost savings between the external and internal decks can be attributed to the greater number of output parameters printed out with internal decks.

As described in the previous section, the majority of this gain (i.e., approximately 45%) is due to analytic derivatives. The remaining 6% is due to the inclusion of the new faster version of the subroutine COFFEX.

6.0 ASSESSMENT OF THE CENERALITY AND UTILITY OF THE APPROACH

The analytic derivative method can be applied to any cycle deck which uses numerical derivatives. Based upon the experience at AEG, the best approach is to differentiate that portion of the deck for whch the coding remains constant; for example, the subroutines used to evaluate thermodynamic properties are usually good candidates since their structure is fairly constant and they are called frequently. This approach should result in the greatest savings per unit cost.

For the Air Force Deck (SMOTE) the deck coding appears to be relatively constant (based upon a comparison of the current coding and that given in Reference 3). This is due largely to the neglect of second order effects such as tip clearances, etc. Since this deck represents the principal engine simulation model used by the Air Force, the entire deck should be differentiated.

It is difficult to estimate cost savings, since different computer installations use different cost algorithms. However, it is possible to estimate processor time gains. An approximate formula for estimating processor time gain which can be obtained from analytic derivatives was derived during this program. If no other information is available, processor time gains can be used as a rough estimate of cost gain. The derivation of this formula together with several examples of its use is given below.

In order to estimate the balanced point time gain due to the inclusion of analytical derivatives in a gas turbine engine cycle deck, the following information is required:

1. The number of base points per cycle balance point.

2. The number of derivative points per cycle balance point.

3. Ratio of the time required to execute an analytical derivavtive pass to the time required for a base point.

Items 1 and 2 can be obtained from the iteration history of the deck averaged over a typical flight envelope. The third item may be estimated based upon past experience. For example, for the parametric VCE deck, the ratio is known to be about one-tenth (0.1).

The estimating formula is obtained as follows:

Let NBP = number of base points

TBP = processor time required to execute one base point

NDP = number of derivative points per cycle balance point

TDP = processor time required to execute one analytic derivative point

Assume 1. The time required to execute a base point is 10% greater after analytic derivative logic is included (due largely to curve and thermo derivatives which must be evaluated on base points).

2. The time required to execute a base point is the same as the time required for one numerical <u>derivative</u> pass.

Then,

$$\frac{\text{Time for Balanced Pt. with Analytic Derivatives}}{\text{Time Required with Numerical Derivatives}}$$

$$= \frac{NBP * TBP * 1.1 + NDP * TDP}{NBP * TBP + NDP * TBP}$$

After simplification,

$$\text{Time Ratio} = \frac{1.1 + \left(\frac{TDP}{TBP}\right) * \left(\frac{NDP}{NBP}\right)}{1 + \left(\frac{NDP}{NBP}\right)}$$

The maximum possible time savings can be obtained by assuming that the time for an analytical derivative pass is zero, and that the time to run the base point is unchanged by the inclusion of the analytical derivative code (i.e., the 10% increase in base point run time does not exist). With these two restrictions, the above equation reduced to:

$$\text{Maximum Time Ratio} = \frac{1}{1 + \left(\frac{NDP}{NBP}\right)}$$

These relationships have been plotted in Figure 7. The percent reduction in run time in the present deck is about 62% over the 411 point flighr matrix. The ratio of derivative passes to base points averages about 2.3. The approximately relationship shown in Figure 7 would predict a reduction of about 60% which is within two percentage points of the actual value. The maximum time gain possible would be 70%. Therefore, about 90% (62%/70%) of the possible time reduction has been obtained with the present deck setup. For engines and/or flight envelopes having a greater ratio of derivatives passes to base points, considerably greater time gains can be achieved with analytic derivatives.

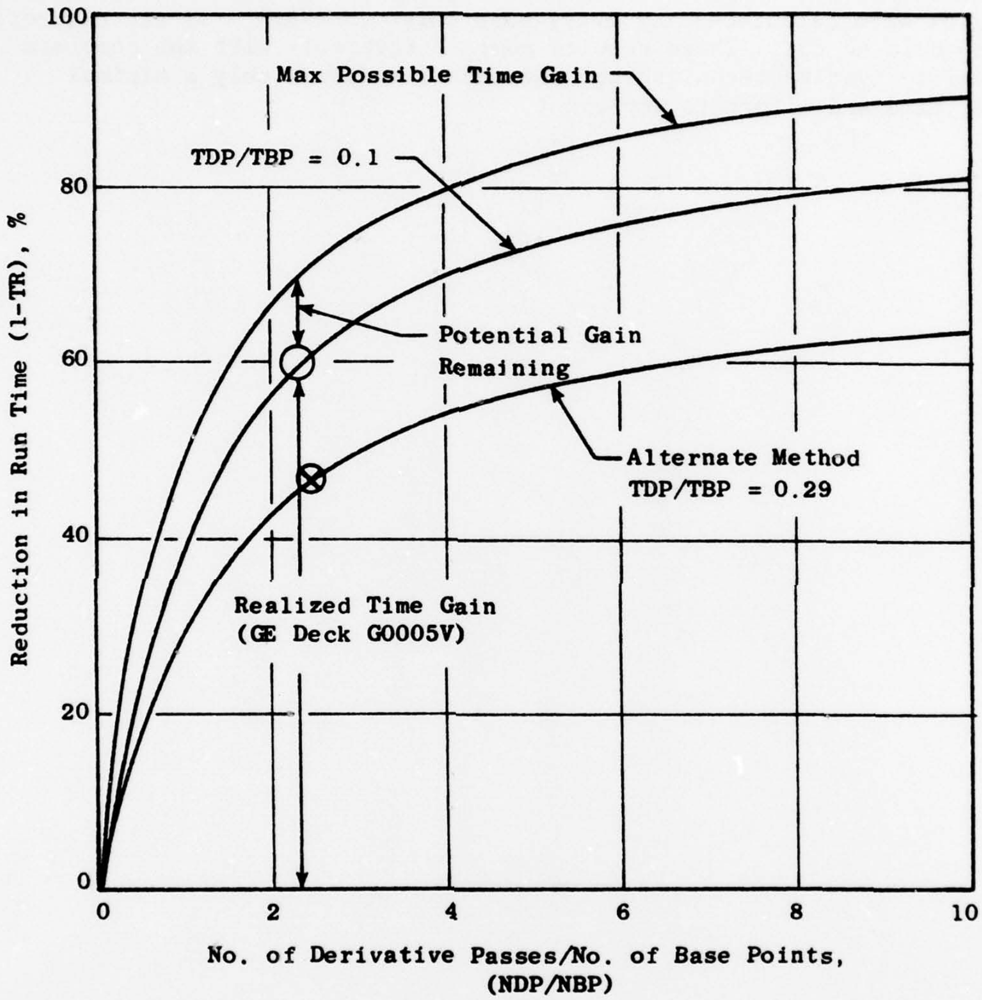


Figure 7. Estimated Reduction in Run Time Due to Analytic Derivatives.

For iteration algorithms which evaluate derivatives after every base point, the ratio of derivative passes to base point passes is equal to the number of independent variables. For the VCE computer program file, the number of independent variables is 9. Had derivatives been taken after every point, the reduction in time would have been 80% or about five to one.

For the second method (low level derivative set), it was determined that the ratio between the time for a derivative pass to a base point pass is 0.29. The curve for TDP/TBP = 0.29 is shown on Figure 7; and it indicates that if derivatives were calculated for every base point (NDP/NBP = 9) the theoretical gain would be 63%. These results compare favorably with the complete analytical derivative technique and the method requires only a minimal amount of time and effort to implement.

7.0 CONCLUSIONS

It can be concluded that in any engine simulation model using numerical derivatives to obtain balanced cycle points, a considerable savings in both computer time and cost can be obtained by the inclusion of analytic derivatives. The greatest return per dollar of cost results when only that portion of the model for which the coding remains relatively constant is differentiated. For the Air Force engine simulation model (SMOTE), the entire deck should be differentiated since coding changes do not appear to occur frequently.

The cost savings obtained during this study were 44% of a deck run internally and 52% of the same deck run externally (the difference is due to the use of a larger set of output parameters on the internal deck).

REFERENCES

1. Powell, H.N.; Shafter, A.; Suciu, S.N.; "Properties of Combustion Gases, System C_n H_{2n}-Air, Volume 1. Thermodynamic Properties; Volume 2. Chemical Composition of Equilibrium Mixtures," Published by McGraw Hill.

2. McKinney, John S., "Simulation of Turbofan Engine. Part I. Description of Method and Balancing Technique," Report No. AFAPL-TR-67-125, Pt. 1, Air Force Systems Command, November 1967. (Available from DDC as AD-825197.)

3. McKinney, John S., "Simulation of Turbofan Engine. Part II. User's Manual and Computer Program Listing," Report No. AFAPL-TR-67-125, Pt. 2, Air Force Systems Command, November 1967. (Available from DDC as AD-825198.)

☆U.S.Government Printing Office: 1979 — 657-002/388