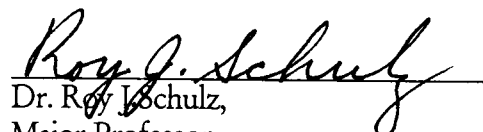
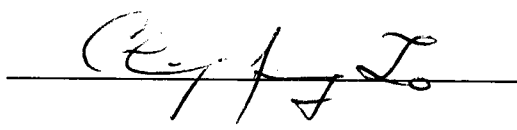


To the Graduate Council:

I am submitting herewith a thesis written by Thomas Paul Fetterhoff entitled "A Simulation Technique for Integrating Facility and Engine Models." I have examined the final copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Mechanical Engineering.


Dr. Roy J. Schulz,
Major Professor

We have read this thesis
and recommend its acceptance:


Frank A. Hollins

Accepted for the Council


Associate Vice Chancellor and
Dean of the Graduate School

A SIMULATION TECHNIQUE FOR
INTEGRATING FACILITY AND ENGINE
SYSTEM MODELS

A thesis

Presented for the

Master of Science

Degree

The University of Tennessee, Knoxville

Thomas Paul Fetterhoff

May 1998

ACKNOWLEDGMENTS

I wish to thank Dr Schulz for his help and guidance. I would also like to thank Dr. C.F. Lo and Dr. F. Collins for their service on this thesis committee. I would also like to thank Dr. Doug Garrard, Mr. Mark Burdett, Mr. Mark Chappell, and Mr. Ted D. Garretson for their support. I would like to give a special thanks to Col. Mike Holquist, USAF (retired) for making my attendance at the university possible. Finally, I would like to thank my wife for her patience and sacrifice during this time.

ABSTRACT

A simulation technique for integrating facility and engine system models is presented. Ground level test facility simulations are currently conducted with a very elementary representation of the test article. In most cases, the test article is treated as a "black box", in which simple parametric curves define the unit's output based on the given input. For example, in a turbine engine test facility model, the turbine engine is represented as an exhaust total pressure and temperature for a given inlet pressure and temperature and throttle angle. The simulation technique presented here integrates a component level model of a gas turbine engine with a facility model. The Navy Large Engine Environmental Test Facility was modeled in a graphical simulation environment called Dynamic System Simulation Software (SIMULINK[®]), into which was integrated a dynamic gas turbine engine model called the Advanced Turbine Engine Simulation Technique (ATEST). The results of the integration and coupling of two dynamic system models proved the feasibility of eventually developing a complete test facility-test article dynamic simulation and performance model.

TABLE OF CONTENTS

CHAPTER		PAGE
I.	INTRODUCTION	1
II.	STATEMENT OF THE PROBLEM	3
III.	THE NAVY LARGE ENGINE ENVIRONMENTAL TEST FACILITY	4
	Background	4
	Description of the Navy Large Engine Environmental Test Facility	5
IV.	MODELING USING SIMULINK®	6
	Software and Modeling	6
	S-functions	7
	Arnold Engineering Development Center Facility Models	7
V.	ADVANCED TURBINE ENGINE SIMULATION TECHNIQUE (ATEST)	9
	Background	9
	ATEST Architecture	9
	Engine Dependent Subroutines	10
	Component Subroutines	10
	Component Models	11
	Component Matching	11

VI.	SIMULATION INTEGRATION	13
	ATEST and NDP® Fortran	13
	Preparing ATEST for SIMULINK®	13
VII.	CONCLUSIONS AND RECOMMENDATIONS	17
	REFERENCES	19
	APPENDICES	22
	1. Figures	23
	2. Component Continuity, Momentum And Energy Equations	36
	3. Construction Of The Jacobian Matrix	39
	4. SIMULINK® Fortran Template	42
	5. SIMULINK® TEMPLATE For ATEST	47
	VITA	50

LIST OF FIGURES

FIGURE		PAGE
1.	ARTIST'S CONCEPT OF THE NAVY LARGE ENGINE ENVIRONMENTAL TEST FACILITY	24
2.	THE NAVY LARGE ENGINE ENVIRONMENTAL TEST FACILITY SCHEMATIC	25
3.	SIMULINK® TEST FACILITY AND CONTROLLER	26
4.	SIMULINK® PLANT MODEL	26
5.	SIMULINK® TEST ARTICLE AND ATMOSPHERIC EXHAUST MODELS	27
6.	SIMULINK® TEST ARTICLE MODEL	27
7.	TURBINE ENGINE	28
8.	ATEST ARCHITECTURE	29
9.	TYPICAL COMPONENT PERFORMANCE MAPS	30
10.	COMPONENT MODELS	31
11.	AIRFLOW VECTOR	32
12.	SIMULINK® TEST CELL MODEL	32
13.	ATEST AND SIMULATION OUTPUT DATA	33

14.	ENGINE RESPONSE TO PLANT PLA COMMAND	34
15.	ENGINE RESPONSE TO PLANT INLET TOTAL PRESSURE CHANGE	35

NOMENCLATURE

A	Area
C	Velocity
cbp	Cell bypass valve
cd	Cold dump valves
cm	Cold mixing valves
cmds	Commands
C-Plant	Aeropropulsion System Test Facility Air Processing Plant
cv	Control Volume
Demux	Algorithm subroutine that separates vector signal into scalar signal
e	Naperian Constant
f	Heat transfer film coefficient
F	Force
Gdn	Gas constant ratio, down stream volume
Gup	Gas constant ratio, upstream volume
h	Specific Enthalpy
hd	Hot dump valves
hm	Hot mixing valves
HPC	High Pressure Compressor
HPT	High Pressure Turbine

HT	Enthalpy
IGV	Inlet Guide Vane
λ_i	ATEST output Parameter
λ_{in}	Near Full Scale ATEST output Parameter
Lbm/s	Pound mass per second.
LPC	Low Pressure Compressor
LPT	Low Pressure Turbine
$\dot{m}$	Mass Flow rate
MD	Mass Flow rate
mhd	Mixing header dump valve
MMI	Man Machine Interface
MNdn	Mach Number, down stream volume
MNup	Mach Number, upstream volume
MRair	Gas constituent mass fraction of air
MRar	Gas constituent mass fraction of argon
MRco2	Gas constituent mass fraction of carbon dioxide
MRh2	Gas constituent mass fraction of hydrogen
MRh20	Gas constituent mass fraction of water
MRn2	Gas constituent mass fraction of nitrogen
MRo2	Gas constituent mass fraction of oxygen
Mux	Algorithm subroutine that groups scalars into a vector

N1	Low Pressure Rotor Speed
N2	High Pressure Rotor Speed
P	Pressure
P0dn	Total Pressure, down stream volume
P0up	Total pressure, upstream volume
P9	Total Pressure, engine exit
Pdn	Static Pressure, down stream volume
pl	Plenum (mixing header)
PL	Plant
PLA	Power Lever Angle
pos	Position
Pup	Static Pressure, upstream volume
$\dot{q}$	Heat Transfer
ρ	Density
Rdn	Specific gas constant, down stream volume
Rup	Specific gas constant, upstream volume
SFC	Specific Fuel Consumption
S-Function	User programmed SIMULINK [®] function
sc	Cold supply volume
sh	Hot supply volume
T	Temperature

t	Time
τ	Time Constant
T_{0dn}	Total temperature, down stream volume
T_{0up}	Total temperature, upstream volume
T_9	Total temperature, engine exit
T_{dn}	Static temperature, down stream volume
TEC	Test Environment Controller
T_F	Fluid Temperature
T_M	Metal Temperature
T_{up}	Static temperature, upstream volume
TYP_{dn}	Type, Down stream volume
TYP_{up}	Type, upstream volume
u	Specific internal energy
V	Volume
W_{1A}	Inlet air flow
W_9	Air flow, engine exit
W_{dn}	Air flow, down stream volume
W_{up}	Air flow, upstream volume

Abbreviations & Acronyms

AEDC	Arnold Engineering Development Center
------	---------------------------------------

ASTF	Aeropropulsion Systems Test Facility
ATEST	Advanced Turbine Engine Simulation Technique
ATMEX	ATEST “mex” file
BRAC	Base Realignment and Closure Commission
DOS	Disk Operating System
ETF	Engine Test Facility
MATLAB [®]	Software Produced by Math Works Inc. Called Matrix Laboratory
Mex	Fortran or C programs used as functions by MATLAB [®]
NLEETF	Navy Large Engine Environmental Test Facility
REPLICAS [®]	Relational Effective Procedures, and Logic for the Implementation of Computerized Analysis and Simulation
SIMULINK [®]	Dynamic System Simulation Software
SL2	Sea Level Test Cell Number 2
SL3	Sea Level Test Cell Number 3

I. INTRODUCTION

Ground level test facility design studies are currently conducted with simulations that use a very elementary representation of the test article, or device that is being tested in the facility. In most cases, the test article is treated as a “black box”, in which simple parametric curves define the unit’s output based on the given input. For example, in a turbine engine test facility model, a turbine engine is the test article and is represented by a set of values for its exhaust total pressures and temperatures, for a given set of inlet pressures and temperatures, and throttle angles.

This current facility modeling approach gives little insight to the operation of the test article, the dynamic coupling between the test facility and the test article, and especially the interdependence of the facility and the test article. This current simulation technique yields information on test facility issues, but does not yield information concerning test article issues or their dynamic interactions. A separate model for the test article must be run and its output related to that of the facility model, in an attempt to provide this insight on coupling and interactions. Oversimplification of the test article model can lead to insufficient resolution of physical behavior of the test facility model based on its simulation output.

A Facility modeling technique that accurately modeled test article and test facility, as well as the dependencies between them, could be a useful tool in test facility studies, test period preparations, and test operations personnel training.

II. STATEMENT OF THE PROBLEM

This Thesis describes a feasibility study on an improved simulation technique that integrates a component level model of a gas turbine engine with a facility model. Both engine and facility models are dynamic models; that is, capable of describing the time dependent behavior of the entity modeled. In particular, the Navy Large Engine Environmental Test Facility (NLEETF) will be modeled in the graphical simulation environment called Dynamic System Simulation Software (SIMULINK[®]), and then integrated with a gas turbine engine simulation model called the Advanced Turbine Engine Simulation Technique (ATEST). SIMULINK[®] is an extension to software called MATLAB[®] (Matrix Laboratory) which is routinely used at the Arnold Engineering Development Center (AEDC) for simulation studies involving simulating dynamic systems. The thesis includes a discussion of these two technologies and the techniques used to couple and integrate them.

The Navy Large Engine Environmental Test Facility will be described in chapter III. The modeling using SIMULINK[®] is described in chapter IV. Chapter V will outline the ATEST engine model. The integration of the Navy Large Engine Environmental Test Facility and the ATEST turbine engine model is described in chapter VI. The summary, results, conclusions and recommendations are provided in Chapter VII.

III. THE NAVY LARGE ENGINE ENVIRONMENTAL TEST FACILITY

III.1. Background

In 1993 the Base Realignment and Closure Commission (BRAC) made the decision to close the aeropropulsion test facilities at the Naval Air Warfare Center, Aircraft Division, (NAWCAD) in Trenton, New Jersey. This decision also provided for the transfer of some of the unique turbine engine test capabilities of NAWCAD to AEDC. This transfer includes moving a small engine test cell and a turboshaft engine test cell to AEDC for installation in the Engine Test Facility (ETF), as well as the construction of two sea level ram inlet test cells, figure (1)¹. This new test facility at AEDC is called the Navy Large Engine Environmental Test Facility (NLEETF). The sea level ram inlet test cells are called SL2 and SL3 and are located in the NLEETF. To demonstrate the feasibility of the integration and coupling of a test facility dynamic model with a test article (engine) dynamic model, the present thesis will only be concerned with the dynamic simulation of the two sea level ram inlet test cells, with installed gas turbine engines.

¹ The Figures are located in Appendix 1.

III.2. Description of the Navy Large Engine Environmental Test Facility

These two test cells will have conditioned air supplied by the Aeropropulsion Systems Test Facility (ASTF) air supply plant (C plant). The Sea Level test facility supply duct is tied into leg 3 of the C plant. The duct supplies up to 600 lbm/s of conditioned air to the test cell area. The air is to be split into two legs, one hot and one cold, for temperature control. The hot leg will simply be the supply air from C plant. The cold air will be supplied by diverting air from the C plant supply and expanding it through expansion turbines located in the Navy Large Engine Environmental Test Facility. Test cell inlet temperatures and pressures are to be controlled by the mixing of these two legs. The test cell inlet temperature will be variable between -65°F and 260°F at pressures up to 30 psia. ASTF can provide supply air to the NLEETF, or one of the test cells in the ASTF complex, on a continuous basis. The NLEETF will also have dynamic soak capability independent of ASTF supply air. A facility schematic is provided as figure 2.

IV. MODELING USING SIMULINK®

IV.1. Software and Modeling

MATLAB® is a numeric computation and visualization program provided commercially by The Math Works Inc. MATLAB® is available for use in either Windows® or UNIX® environments. MATLAB® is used for numerical analysis, matrix computations, graphics and signal processing. MATLAB® is an interactive system for solving mathematical and engineering problems that would normally require programming in Fortran, C, or Basic. One the most important capabilities of MATLAB® is its ease of expansion. This expansion can be accomplished by using a variety of application-specific “toolboxes” or user programmed functions [1 and 2]².

Dynamic System Simulation Software (SIMULINK®) is an extension to MATLAB®. SIMULINK® allows you to both define and analyze dynamic system models. Models are easily defined by using block diagram windows. The modeler creates the model by mouse driven commands. The modeler can analyze the model from the MATLAB® command line or from SIMULINK® menus. MATLAB® and SIMULINK® used together can model dynamic systems [3 and 4].

² Numbers in brackets refer to similarly numbered references in the list of references.

IV.2. S-functions

MATLAB[®] and SIMULINK[®] can be supplemented with user programmed functions. These functions are called S-functions. S-Functions are called by simply entering their name in a system block. S-functions can be programmed in Fortran, C, or in the MATLAB[®] native language called M. Fortran code is compiled with the MATLAB[®] supplied "Fmex" utility and a compatible Fortran compiler to create a "mex" file that is compatible with MATLAB[®] and SIMULINK[®]. Only a limited number of Fortran compilers are supported by MATLAB[®] (version 4.3c) and SIMULINK[®] (version 1.3). AEDC is using Microway[®] NDP[®] Fortran (version 4.4).

IV.3. Arnold Engineering Development Center Facility Models

Engineers at Arnold Engineering Development Center (AEDC) are modeling the ground test facilities using MATLAB[®] (version 4.2c) and SIMULINK[®] (version 1.3). The first turbine engine test facility to be modeled in this graphical environment is the Navy Large Engine Environmental Test Facility. The model has not yet been finalized. The model depicted in figure 3 is envisioned as a test facility and test article simulation and as a training aid for training facility and test unit operators. This added capability is accomplished by accommodating inputs from hardware, such as the Test Environment Controller (TEC), Power Lever Angle (PLA) controller, and control room consoles, as well

as digital simulations of this hardware. Figures 3 through Figure 6 are provided to illustrate the hierarchy of the model in the SIMULINK[®] environment. Figure 3 is a high level graphical representation of the test facility including the controller. This figure contains the entire facility and test article models. TEC Sim, PLA Sim, and Console Sim blocks are where the models of the TEC, PLA controller, and control room consoles are located. Figure 4 is the graphical representation of the plant model. Figure 4 is the details of the block labeled "Plant Engine Sim" in Figure 3. The cold leg, supply leg, and mixing header blocks are where the corresponding models of these facility components are located. The Demux block breaks down an array of parameters into individual components. The Mux block combines a number of individual parameters into an array. The lines connecting the blocks in this model carry controller commands, position, temperature, and pressure indications between the blocks. The details of what is communicated between each of these blocks are not important for this discussion. The "engine model" block in figure 4 contains the test article and atmospheric exhaust models, figure 5. Figures 4 and 5 are successively lower levels of the facility model, Figure 3. The test article model, figure 6, is contained in the block labeled ATEST in figure 5. The block labeled ATEST, in figure 6, is the S-Function for ATEST. The blocks labeled in_1, in_2, in_3, out_1, out_2, and out_3 are input and output vectors. They are described in more detail in section VI.2. Each lower level block reveals more detail of the block in the next higher level.

V. ADVANCED TURBINE ENGINE SIMULATION TECHNIQUE (ATEST)

V.1. Background

The Advanced Turbine Engine Simulation Technique (ATEST) was developed at AEDC starting in 1986 [5 and 6], and development continues to the present. ATEST is a one-dimensional, time-dependant, component matching simulation that has been applied to a wide variety of turbine engine cycles. The current version, ATEST 3.0, is capable of simulating transient, steady state, start, and windmill turbine engine operation [7 and 8]. The engine simulated, for use in the facility model, is a dual spool augmented mixed flow turbofan. A schematic is provided as figure 7. The numbers in parenthesis on figure 7 are the station numbers assigned to the corresponding engine location. These station numbers are industry standard engine measurement locations.

V.2. ATEST Architecture

ATEST is a modular program written in Fortran 77. It consists of engine dependant subroutines, component and utility subroutines, and a main or supervisory program, (figure 8). This modularity makes it possible to change the simulated engine, or characteristics of the simulated engine, by modifying the appropriate modules or

subroutines. Component performance can be altered by simply replacing the block data associated with that component.

V.3. Engine Dependent Subroutines

The engine dependent subroutines consist of engine component data and engine cycle subroutines. The component data is taken from component performance maps similar to the maps depicted in figure 9. The engine cycle subroutine establishes the thermal states along the engine flow paths and the energy losses in the engine.

V.4. Component Subroutines

Compression, expansion, combustion and mixing calculations are completed in the engine component subroutines. The calculations for the inlet are done in a subroutine called INLET. Low and high-pressure compressor calculations are completed in COMPR. The augmentor and combustion chamber calculations are completed in DUCT. High and low pressure turbine calculations are done in TURB and the nozzle calculations are completed in NOZZLE. These subroutines are, collectively, a one-dimensional representation of an engine.

V.5. Component Models

Work, heat, or bleed in or out of a compression or expansion component is assumed to occur across an infinitesimally thin disk, or actuator disk. The lumped volume approach is used to account for dynamic volume effects on thermodynamic variables, (figure 10). The lumped volume approach is utilized so that thermodynamic properties can be considered constant throughout the volume. The continuity, momentum, and energy equations, as given in reference 5 are presented in appendix 2.

V.6. Component Matching

A system of nonlinear first order, differential equations defining conservation of mass flow, momentum flow, and energy flows within engine components relates the independent engine variables (shaft speed, component pressure ratios, bypass ratios, etc) to the dependent engine variables (mass, energy, momentum). These dependent parameters must be matched across the components. For example, the mass flow out of the combustor must match the mass flow into the high-pressure turbine. The work required to operate the high pressure compressor must equal the work output of the high pressure turbine, minus mechanical losses and power tradeoffs. The system of differential equations is converted into a system of non-linear algebraic equations by finite differencing techniques to relate the dependent and independent variables. A Jacobian matrix representing the system is approximated by perturbing the independent variables and calculating the change

in the dependent variables. Details of this process, as provided in reference 6, are presented in appendix 3. Solving the Jacobian matrix is an iterative process. ATEST uses a collection of algorithms called the Relational, Effective Procedures, and Logic for the Implementation of Computerized Analysis and Simulation (REPLICAS[®]) to provide convergence of these implicit relationships. REPLICAS[®] is based on a Newton-Raphson technique augmented by a Broyden's Method matrix updating algorithm [9]. REPLICAS[®] uses a scaling algorithm to reduce the matrix structure convergence problems associated with the Jacobian elements having large magnitude variations.

VI. SIMULATION INTEGRATION

VI.1. ATEST and NDP® Fortran

The ATEST program was previously recompiled for operation on personal computers using the Fortran compiler Microsoft PowerStation®. PowerStation®, however is not compatible with SIMULINK® (Version 1.3). NDP® Fortran is a Microsoft Disk Operation System (DOS) based compiler that generates code for the Intel Corporation 80386DX microprocessor [10 and 11]. This code also runs on 80486 and Pentium processors. To make ATEST compile in NDP® Fortran a number of format changes were made. While making these changes, it was imperative that the modularity of ATEST be maintained. It was also imperative that the ATEST engine simulation not be compromised. After ATEST was modified, transient and steady state test simulations were executed to ensure that the output had not been changed by the modifications.

VI.2. Preparing ATEST for SIMULINK®

For SIMULINK® to interface with and execute a Fortran program, it must be compiled using the SIMULINK® compatible compiler. The program must be in the format of the example supplied by MATLAB® as the file "usersys.f" (appendix 4). The ATEST version of this file is listed in appendix 5. Putting ATEST in this format allows

SIMULINK® to communicate with it. The inputs, from the SIMULINK® facility model to the ATEST test article model, are in 3 vectors totaling 61 inputs. The input vectors include two, thirty component, air flow vectors, figure 11, and Power Lever Angle (PLA). These air flow vectors are used to communicate aerothermal properties between the engine and facility models as well as between facility components. The outputs from ATEST are in 3 vectors totaling 65 outputs. The output vectors include two, thirty component, airflow vectors, and a five component engine parameter output vector. The facility model limits these inputs and outputs. Changing them requires modification of the SIMULINK® facility model. ATEST was reformatted to input and output these vectors instead of its normal input and output files. From the input vectors ATEST receives the "flight condition", which for an engine in a ground test facility is really a description of the flow in the inlet duct, and PLA settings. It then returns, through the computed engine output vectors, engine parameters and a description of the flow out of the engine. The "mex" file that results from compiling this file for use with SIMULINK® was then converted to an S-function. This S-function was then used to replace the simple parametric model in the SIMULINK® simulation of SL2 and SL3, when the facility model is completed.

A stand-alone test cell model, figure 12, was created to aid in the development of the S-function. The input and output vectors are the airflow vectors described previously. The slider gain is used to vary the PLA command. A comparison of ATEST output and output from the test cell model can be seen in figure 13. Figure 13 is a comparison of the SIMULINK® S-function model outputs of engine inlet mass flow (W1A), engine outlet

mass flow (W9), nozzle outlet total pressure (P9), nozzle outlet temperature (T9), and engine speeds (N1 and N2) with the same parameters as output by ATEST compiled and executed with NDP[®] Fortran. Each parameter (λ_i) has been non-dimensionalized by a near full-scale value of that parameter (λ_{in}). This method of presentation was chosen to protect the engine manufacturers data and allow the curves to be displayed in one figure. The curves in the figure depict the parameter non-dimensionalized value verses PLA. This same curve would be generated during performance testing in the test cell. The above-described comparison was used to verify that ATEST was functioning properly when compiled and executed inside the SIMULINK[®] environment.

Figure 14 is a time dependent plot of PLA, engine inlet mass flow (W1A), and nozzle outlet total pressure (P9). The vertical axis has been removed to protect the engine manufacturers data. In this figure, the engine model is responding to the facility PLA input. As PLA is ramped to slightly more than twice its original value, P9 and W1A grow to slightly more than double their initial values. The affect of the engine acceleration schedule can be seen in W1A in the first two seconds of the ramp. The inlet mass flow increases approximately fifty percent from a very small increase in PLA. Figure 15 is also a time dependent plot of PLA, engine inlet mass flow (W1A), and nozzle outlet total pressure (P9), however this time the engine is responding to inlet total pressure changes from the facility model. The vertical axis has been removed to protect the engine manufacturers data. This Plot shows small increases in W1A and P9 with a slight increase in total pressure at the inlet,

on the order of two psia. Comparisons to actual test facility data cannot be made because the facility is not yet operational.

VII. CONCLUSIONS AND RECOMMENDATIONS

ATEST was modified to be compatible with and to compile using Microway NDP[®] Fortran, and modified for compatibility with the SIMULINK[®] input and output vector formats. The program was then compiled to be linked to and operate in the SIMULINK[®] environment as an S-function. The incorporation of ATEST within the facility simulation gives the facility simulation access to two hundred engine and component parameters. The engine is now accurately simulated inside the test facility model. This allows more accurate facility modification studies. The SIMULINK[®] model can also be used to train engine and facility operators by running the program in conjunction the facility control computers.

It is recommended that ATEST models be incorporated into the models for different turbine engine test facilities as they are created. It is also recommended that the third output vector be enlarged to include more engine parameters including temperatures and pressures at the station numbers indicated on figure 7 as well as thrust, fuel flow, and specific fuel consumption (SFC). This will make additional engine data available to the model operator.

MATLAB[®] (version 5) supports only Microsoft PowerStation[®] Fortran (version 5.0) and MATLAB[®] (version 5.1) supports only Digital Visual Fortran. "Mex" files created for older versions of MATLAB[®] and SIMULINK[®] are not useable until compiled in the currently compatible compiler. These frequent changes in compiler compatibility create

additional upgrade costs. A component level SIMULINK[®] model should be made to alleviate some of these software version-dependent upgrade problems. A SIMULINK[®] engine model would also improve simulation times by moving the matrix operations from the “mex” file to MATLAB[®].

ATEST is a very versatile program applicable to many physical systems. Another approach to facility modeling would be to model the facility with ATEST. While this approach would not have the Graphical User Interface (GUI) it would execute faster and be less susceptible to software updating changes.

REFERENCES

REFERENCES

1. The Math Works, Inc., MATLAB User's Guide, August 1992, Natick, Mass.
2. The Math Works, Inc., MATLAB External Interface Guide, November 1994, Natick, Mass.
3. The Math Works, Inc., SIMULINK® 1.3 Users Guide, March 1992, Natick, Mass.
4. The Math Works, Inc., SIMULINK® 1.3 Release Notes, January 1995, Natick, Mass.
5. Chappell, M. A., 1998, Sverdrup Technologies Incorporated, AEDC Group, Arnold Engineering Development Center, Arnold AFB, TN, Personal Communication.
6. Chappell, M. A., and Blevins, E.G., 31 December 1985, "Advanced Turbine Engine Simulation Technique Development and Applications to Testing." AIAA-86-1731, AIAA/ASME/SAE/ASEE 22nd Joint Propulsion Conference 16-18 June 1986, Huntsville, Alabama.
7. Chappell, M. A., and McLaughlin, P. W., 1983, "Approach to Modeling Continuous Turbine Engine Operation from Startup to Shutdown," Journal of Propulsion and Power, Vol. 9, No.3, May – June 1993, pp.466-471.
8. Saville, S. A. and Garrard. G. D., 1996, "Application of Transient and Dynamic Simulation to the U.S. Army T55-L-712 Helicopter Engine." Presented at the ASME International Gas Turbine and Aeroengine Congress and Exposition, Birmingham, UK, June 1996.

9. McLaughlin, P. W., "REPLICAS – A New Continuous System Simulation Language,"
Proceeding of the 15th Annual Simulation Symposium, Association for Computing
Machinery, Tampa, FL, March 1982, pp, 1-15.
10. Microway, NDP 386/486/Pentium DOS User's Tools, Quick View Manuals, 1994
Kingston, Mass.
11. Microway, NDP Library Reference Manual and NDP Fortnum Language Reference
Manuel, 1994, Kingston, Mass.

APPENDICES

APPENDIX 1

FIGURES

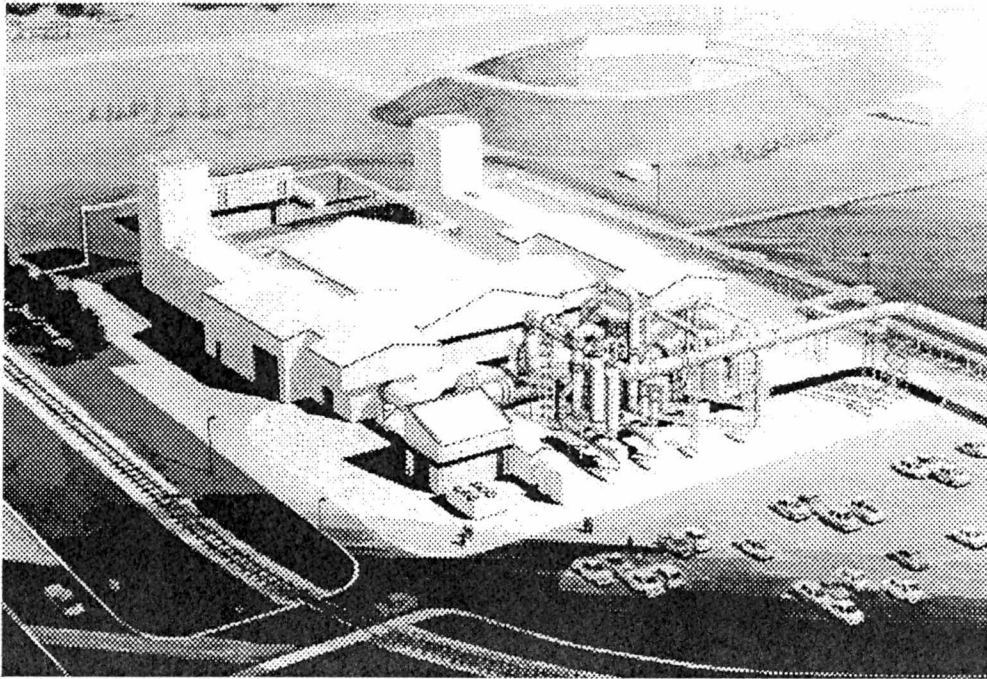
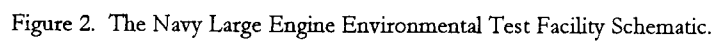


Figure 1. Artist's Concept of the Navy Large Engine Environmental Test Facility.



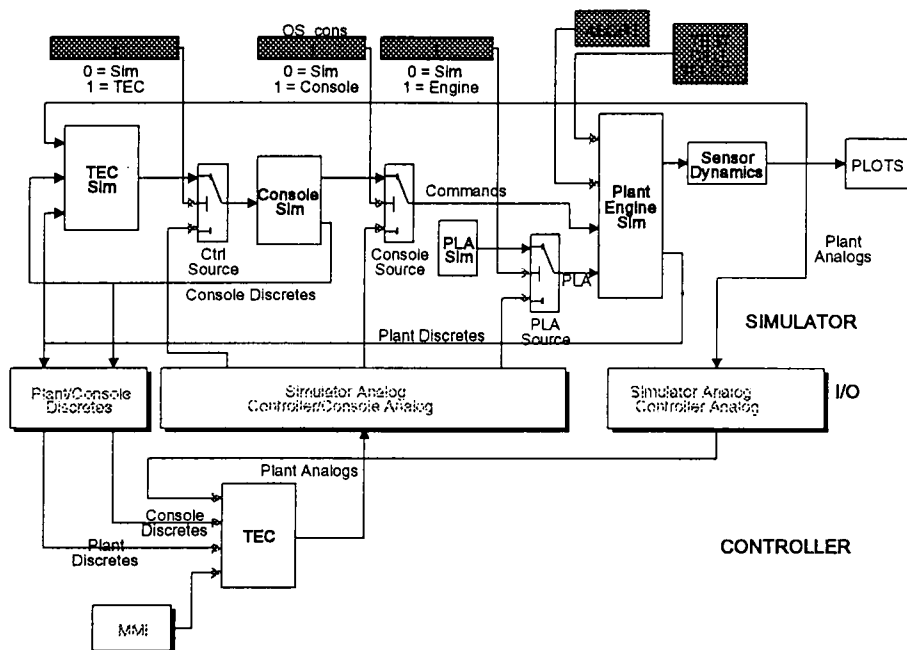


Figure 3. SIMULINK® Test Facility and Controller.

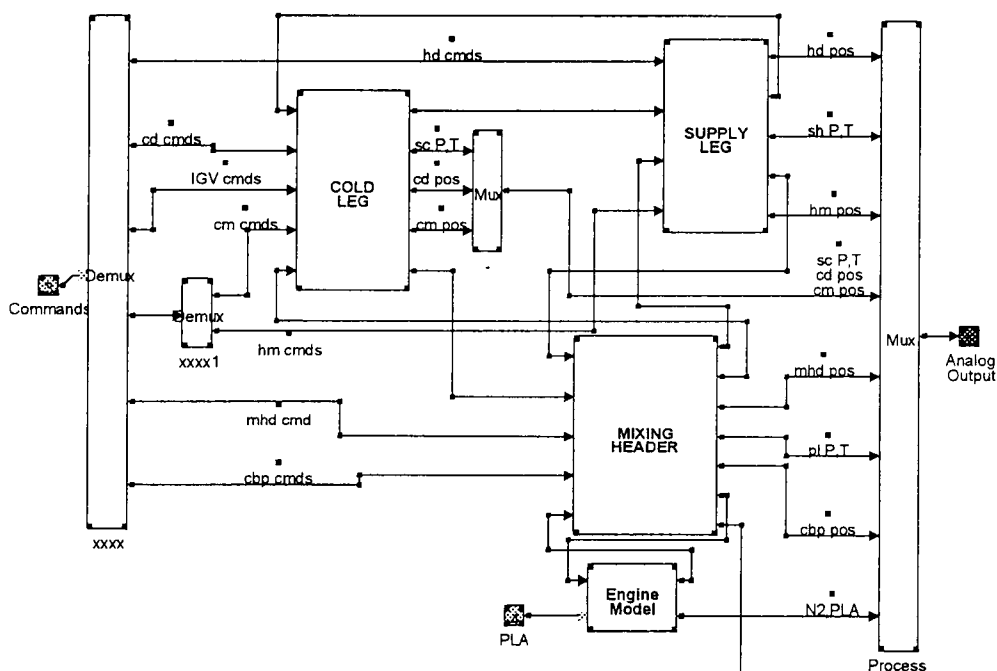


Figure 4. SIMULINK® Plant Model.

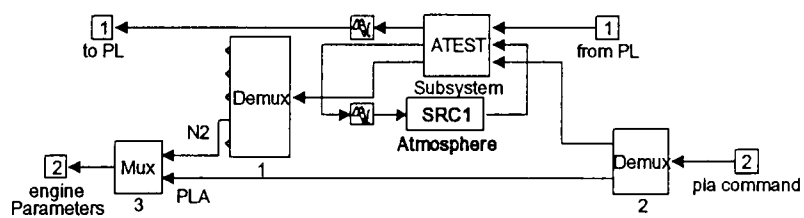


Figure 5. SIMULINK® Test Article and Atmospheric Exhaust Models.

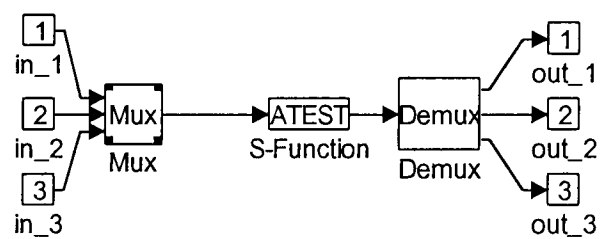


Figure 6. SIMULINK® Test Article Model.

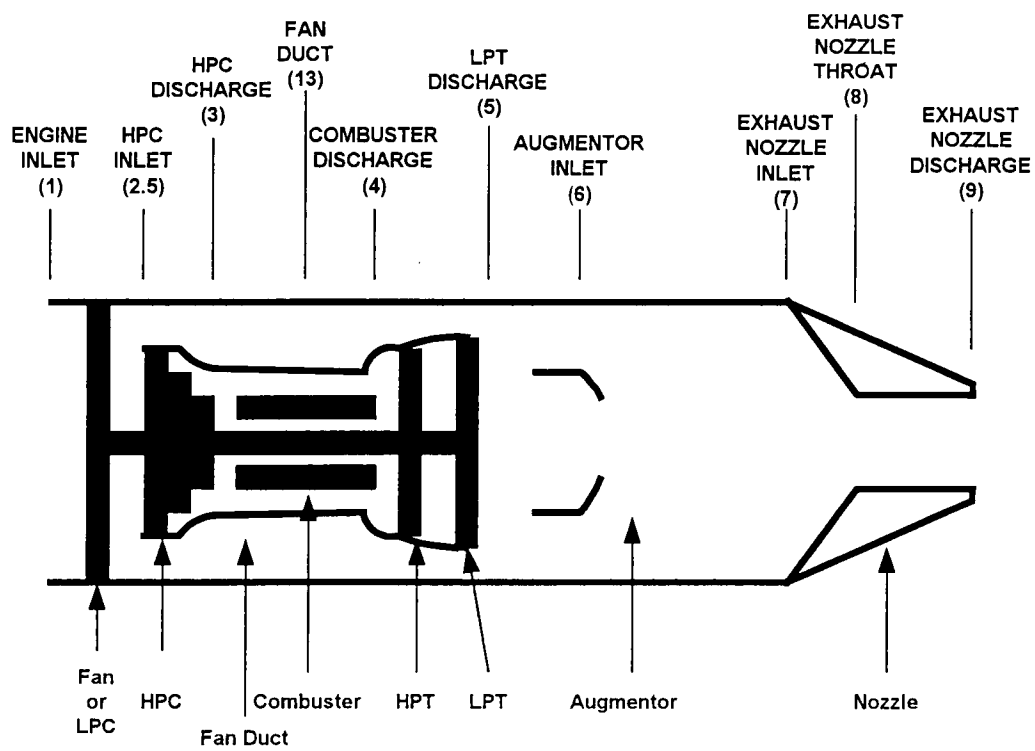


Figure 7. Turbine Engine.

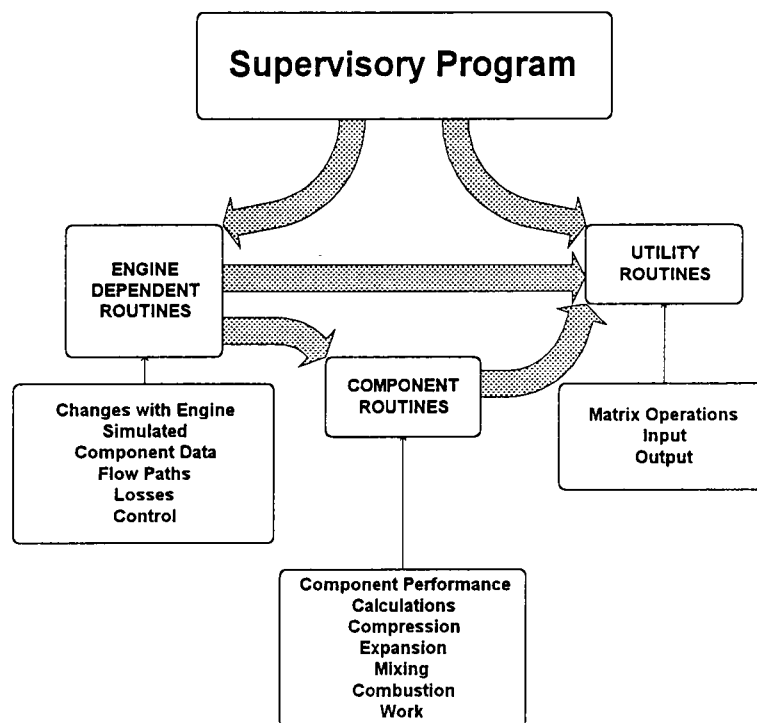


Figure 8. ATEST Architecture.

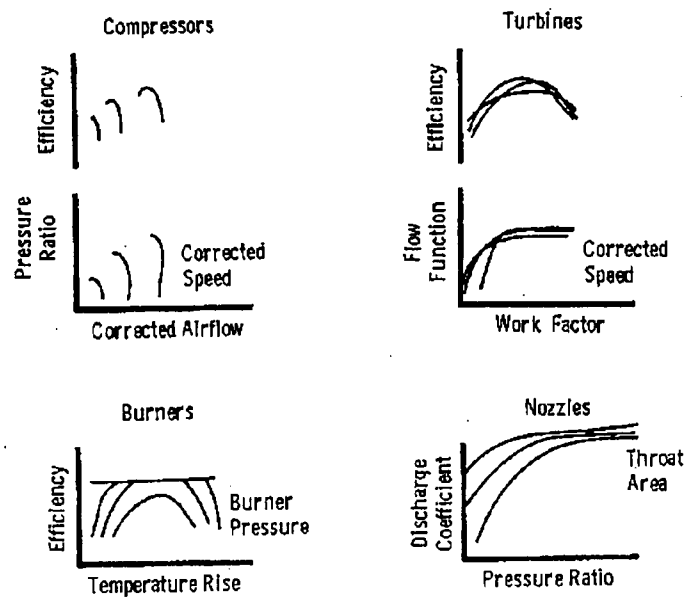


Figure 9. Typical Component Performance Maps.

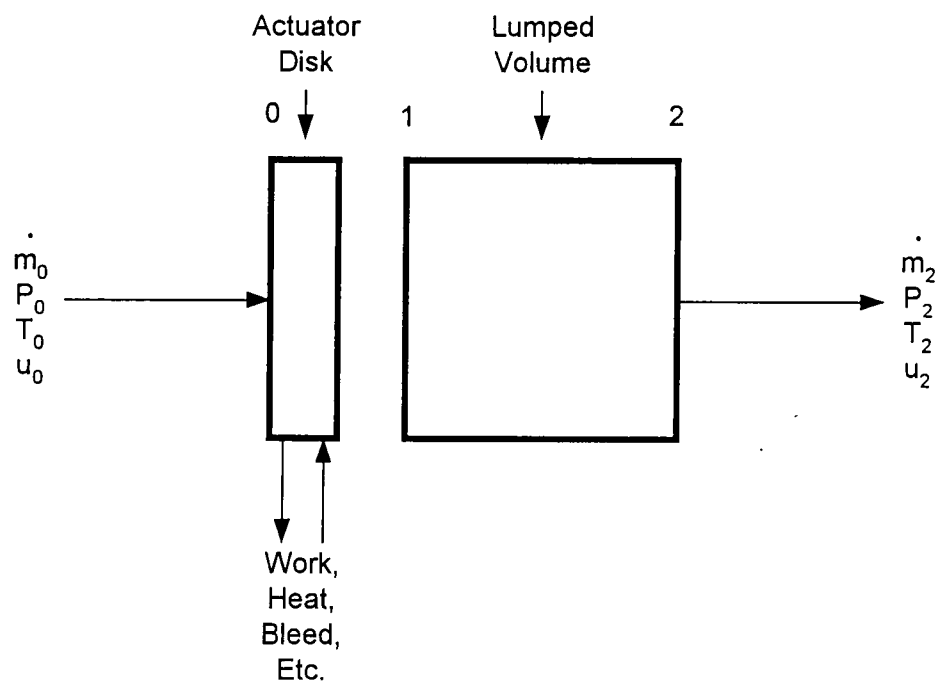


Figure 10. Component Models.

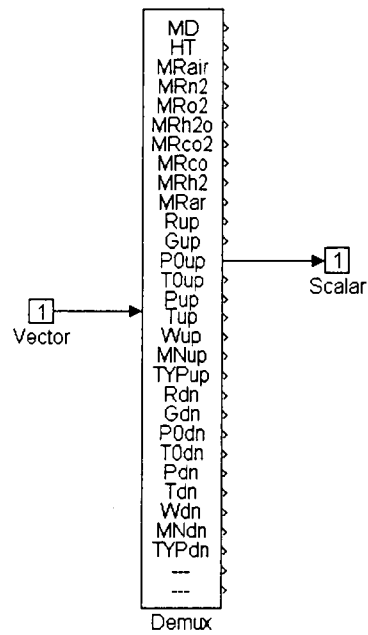


Figure 11. Air Flow Vector.

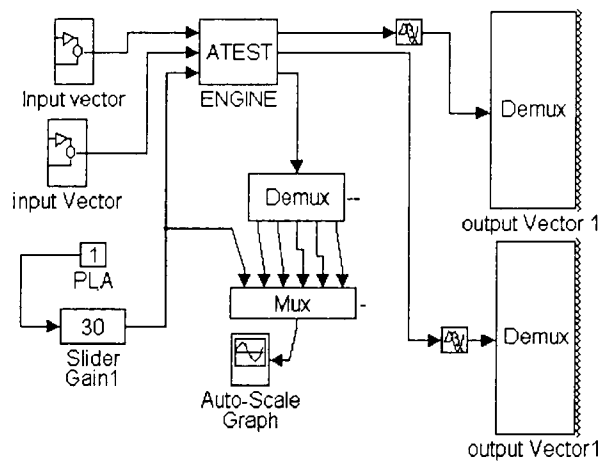


Figure 12. SIMULINK® Test Cell Model.

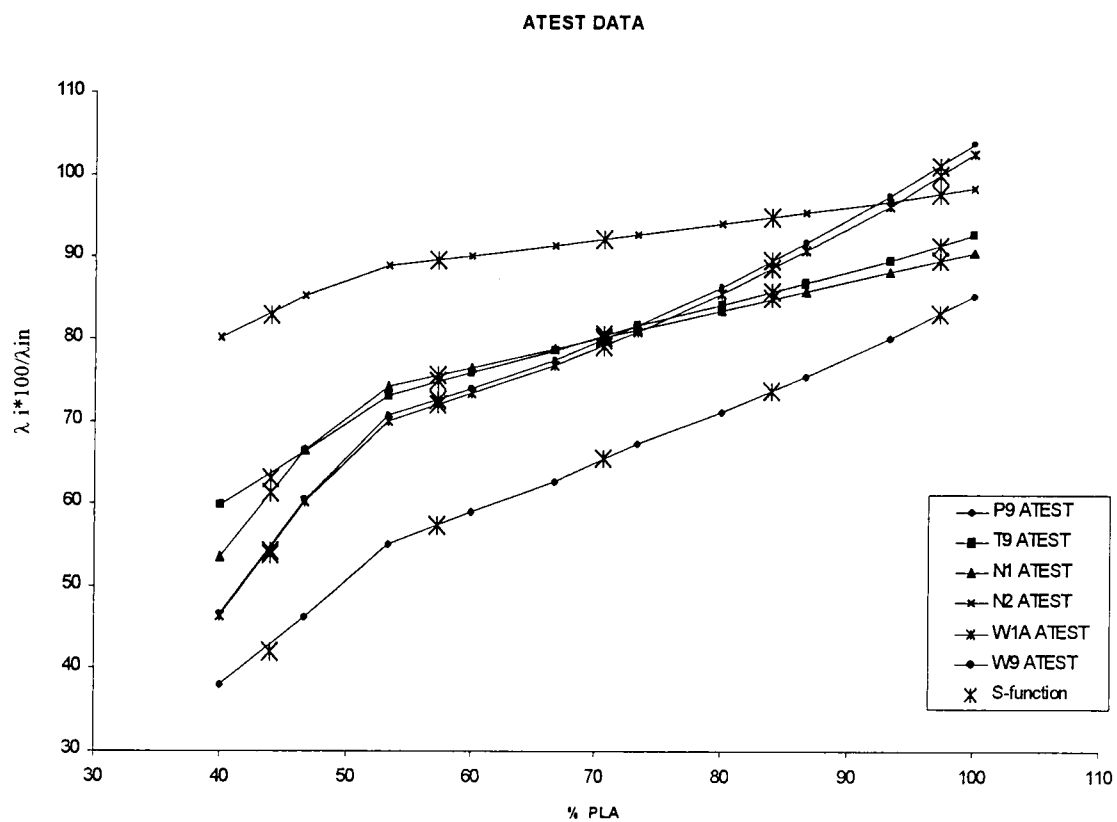


Figure 13. ATEST and Simulation Output Data.

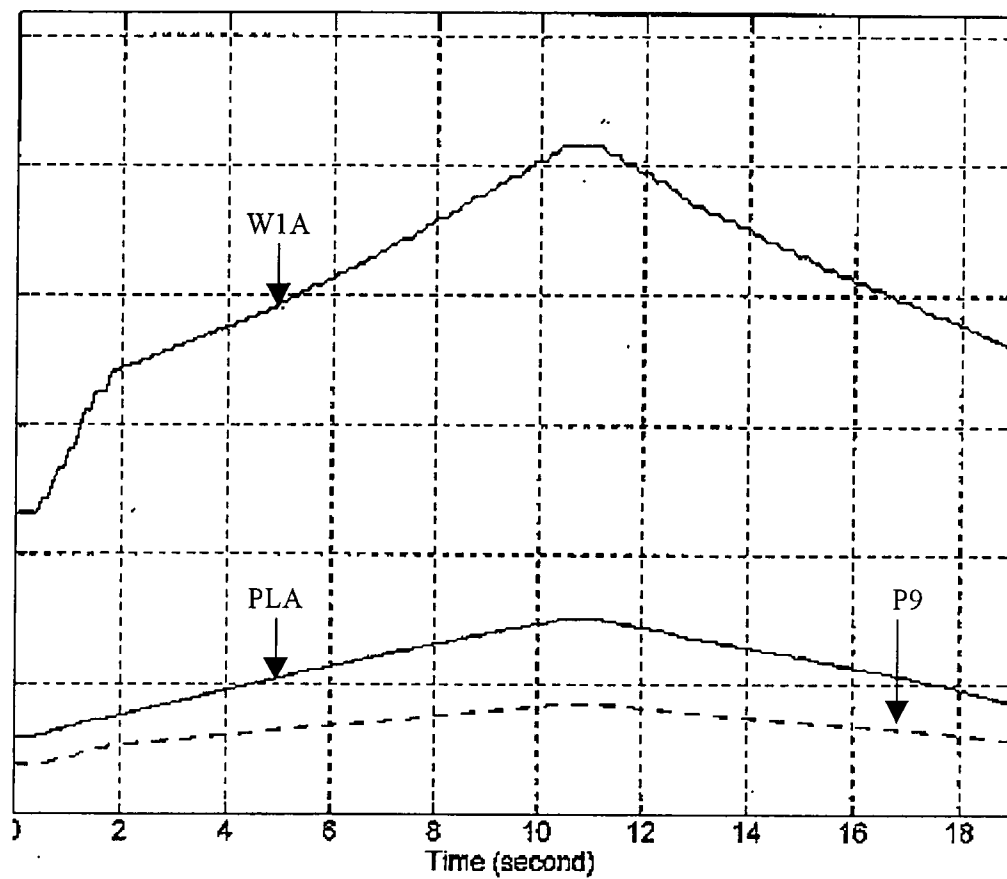


Figure 14. Engine Response to Plant PLA Command.

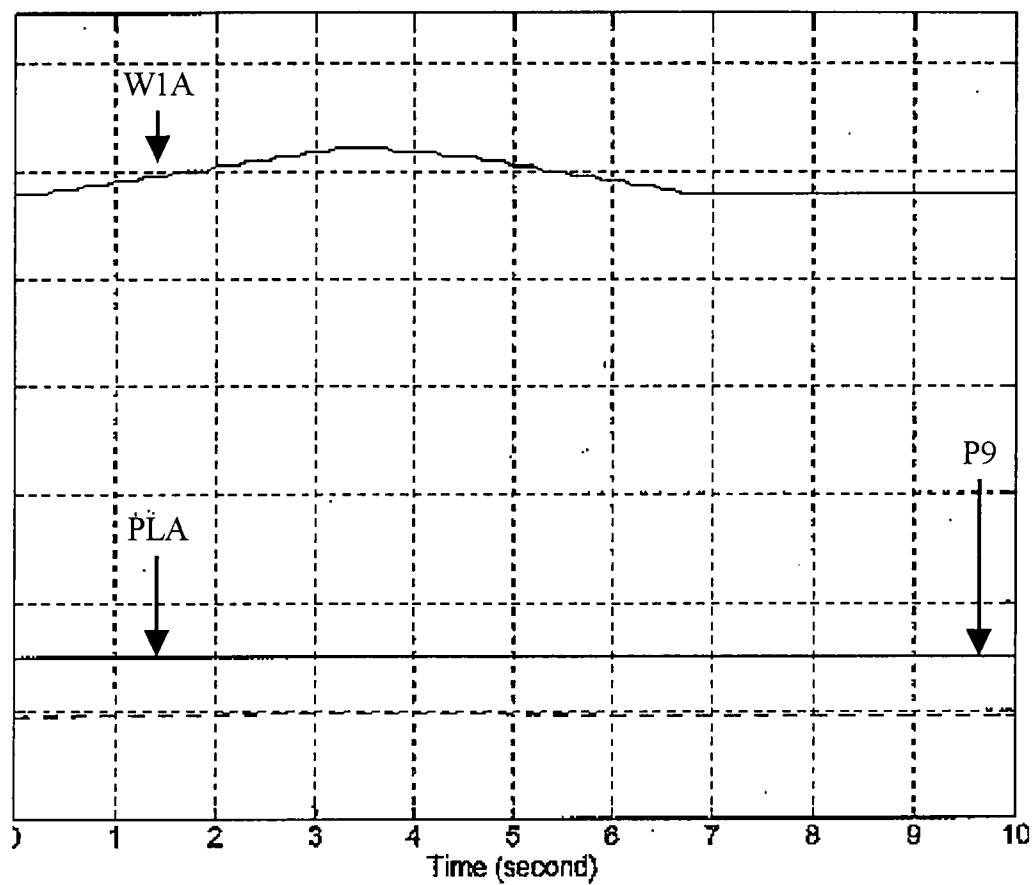


Figure 15. Engine Response to Plant Inlet Total Pressure Change.

APPENDIX 2

COMPONENT CONTINUITY, MOMENTUM AND ENERGY EQUATIONS

APPENDIX 2

COMPONENT CONTINUITY, MOMENTUM AND ENERGY EQUATIONS

The time dependent continuity equation for a component is

$$\frac{\partial \rho}{\partial t} = \frac{1}{V}(\dot{m}_1 - \dot{m}_2).$$

The steady state continuity equation reduces to

$$\dot{m}_2 = \dot{m}_1$$

The time dependent energy equation is

$$\frac{\partial \rho u}{\partial t} = \frac{1}{V}(\dot{m}_1 h_1 - \dot{m}_2 h - \dot{q}).$$

where the heat transfer between the metal and the fluid is

$$\dot{q} = -fA(T_M - T_F).$$

The metal temperature is a function of the fluid temperature.

$$T_M = T_F(1 - e^{-t/\tau}).$$

The steady state energy equation is

$$\dot{m}_1 h_1 = \dot{m}_2 h.$$

The time dependent momentum equation is

$$\Sigma F_x = \frac{\partial}{\partial t} \int_{cv} C_x \rho dV + \dot{m}_2 C_2 - \dot{m}_1 C_1 .$$

The quasi steady state momentum equation is

$$\Sigma F_x = \dot{m}_2 C_2 - \dot{m}_1 C_1 .$$

APPENDIX 3

CONSTRUCTION OF THE JACOBIAN MATRIX

APPENDIX 3

CONSTRUCTION OF THE JACOBIAN MATRIX

Consider the system of nonlinear equations:

$$f_1(x, y, z) = A$$

$$f_2(x, y, z) = B$$

$$f_3(x, y, z) = C$$

Where A, B, and C are constants.

Perturbing each Independent variable.

$$x' = x + \Delta x$$

$$y' = y + \Delta y$$

$$z' = z + \Delta z$$

Where Δx , Δy , and Δz are Small.

Substitute x' , y' , z' into f_1 , f_2 , and f_3 .

$$\begin{array}{lll} f_1(x', y, z) = A_x & f_1(x, y', z) = A_y & f_1(x, y, z') = A_z \\ f_2(x', y, z) = B_x, & f_2(x, y', z) = B_y, & f_2(x, y, z') = B_z. \\ f_3(x', y, z) = C_x & f_3(x, y', z) = C_y & f_3(x, y, z') = C_z \end{array}$$

Define the following:

$$\begin{aligned}\Delta A_x &= A_x - A & \Delta A_y &= A_y - A & \Delta A_z &= A_z - A \\ \Delta B_x &= B_x - B, & \Delta B_y &= B_y - B, & \Delta B_z &= B_z - B. \\ \Delta C_x &= C_x - C & \Delta C_y &= C_y - C & \Delta C_z &= C_z - C\end{aligned}$$

Then the Jacobian Matrix is approximated by:

$$\begin{bmatrix} \frac{\partial f_1}{\partial x} & \frac{\partial f_1}{\partial y} & \frac{\partial f_1}{\partial z} \\ \frac{\partial f_2}{\partial x} & \frac{\partial f_2}{\partial y} & \frac{\partial f_2}{\partial z} \\ \frac{\partial f_3}{\partial x} & \frac{\partial f_3}{\partial y} & \frac{\partial f_3}{\partial z} \end{bmatrix} \approx \begin{bmatrix} \frac{\Delta A_x}{\Delta_x} & \frac{\Delta A_y}{\Delta_y} & \frac{\Delta A_z}{\Delta_z} \\ \frac{\Delta B_x}{\Delta_x} & \frac{\Delta B_y}{\Delta_y} & \frac{\Delta B_z}{\Delta_z} \\ \frac{\Delta C_x}{\Delta_x} & \frac{\Delta C_y}{\Delta_y} & \frac{\Delta C_z}{\Delta_z} \end{bmatrix}.$$

APPENDIX 4
SIMULINK® FORTRAN TEMPLATE

APPENDIX 4

SIMULINK® FORTRAN TEMPLATE

```

      SUBROUTINE SIZES(SIZE)
C    .. Array arguments ..
      INTEGER*4      SIZE(*)
C=====
C    Purpose:
C      Function to set size vector
C    Input arguments:
C      None
C    Output arguments:
C      SIZE      a vector of model sizes
C    Description:
C      Sizes returns a vector which determines model characteristics.
C      This vector contains the sizes of the state vector and other
C      parameters. More precisely,
C      SIZE(1) number of continuous states
C      SIZE(2) number of discrete states
C      SIZE(3) number of outputs
C      SIZE(4) number of inputs
C      SIZE(5) number of discontinuous roots in the system
C      SIZE(6) set to 1 if the system has direct feedthrough of its
C               inputs, otherwise 0
C      Copyright (c) 1990-93 by the MathWorks, Inc.
C=====
C    .. Parameters ..
      INTEGER*4      NSIZES
      PARAMETER      (NSIZES=6)
C    .. Local scalars ..
C    .. Local arrays ..
C    .. Executable statements ..

      RETURN
      END

      SUBROUTINE INITCOND(X0)
C    .. Array arguments ..
      REAL*8         X0(*)
C=====
C    Purpose:
C      Function to set initial condition vector
C    Input arguments:
C      None
C    Output arguments:
C      X0      a vector of initial conditions
C=====
C    .. Local scalars ..
C    .. Local arrays ..
C    .. Executable statements ..

      RETURN
      END

```

```

      SUBROUTINE DERIVS(T, X, U, DX)
C    .. Scalar arguments ..
      REAL*8      T
C    .. Array arguments ..
      REAL*8      X(*), U(*), DX(*)
C=====
C    Purpose:
C      Function to return derivatives
C    Input arguments:
C      T      time
C      X      state vector
C      U      input vector
C    Output arguments:
C      DX     state vector derivatives
C    Remark:
C      The state vector is partitioned into continuous and discrete
C      states. The first states contain the continuous states, and
C      the last states contain the discrete states.
C=====
C    .. Local scalars ..
C    .. Local arrays ..
C    .. Executable statements ..

      RETURN
      END

      SUBROUTINE DSTATES(T, X, U, XNEW)
C    .. Scalar arguments ..
      REAL*8      T
C    .. Array arguments ..
      REAL*8      X(*), U(*), XNEW(*)
C=====
C    Purpose:
C      Function to perform discrete state update
C    Input arguments:
C      T      time
C      X      state vector
C      U      input vector
C    Output arguments:
C      XNEW   next state values
C    Remark:
C      The state vector is partitioned into continuous and discrete
C      states. The first states contain the continuous states, and
C      the last states contain the discrete states.
C=====
C    .. Local scalars ..
C    .. Local arrays ..
C    .. Executable statements ..

      RETURN
      END

      SUBROUTINE OUTPUT(T, X, U, Y)
C    .. Scalar arguments ..
      REAL*8      T
C    .. Array arguments ..
      REAL*8      X(*), U(*), Y(*)
C=====
C    Purpose:
C      Function to return continuous outputs
C    Input arguments:
C      T      time

```

```

C      X      state vector
C      U      input vector
C      Output arguments:
C      Y      output vector
C      Remark:
C      The state vector is partitioned into continuous and discrete
C      states. The first states contain the continuous states, and
C      the last states contain the discrete states.
C=====
C      .. Local scalars ..
C      .. Local arrays ..
C      .. Executable statements ..

      RETURN
      END

      SUBROUTINE DOUTPUT(T, X, U, Y)
C      .. Scalar arguments ..
      REAL*8      T
C      .. Array arguments ..
      REAL*8      X(*), U(*), Y(*)
C=====
C      Purpose:
C      Function to return discrete outputs
C      Input arguments:
C      T      time
C      X      state vector
C      U      input vector
C      Output arguments:
C      Y      output vector
C      Remark:
C      The state vector is partitioned into continuous and discrete
C      states. The first states contain the continuous states, and
C      the last states contain the discrete states.
C      This procedure is called only if it is a sample hit.
C=====
C      .. Local scalars ..
C      .. Local arrays ..
C      .. Executable statements ..

      RETURN
      END

      SUBROUTINE TSAMPL(T, X, U, TS, OFFSET)
C      .. Scalar arguments ..
      REAL*8      T,TS,OFFSET
C      .. Array arguments ..
      REAL*8      X(*), U(*)
C=====
C      Purpose:
C      Function to return the sample and offset times
C      Input arguments:
C      T      time
C      X      state vector
C      U      input vector
C      Output arguments:
C      TS      sample time
C      OFFSET  offset time
C=====
C      .. Local scalars ..
C      .. Local arrays ..
C      .. Executable statements ..

```

```
RETURN
END
```

```
SUBROUTINE SINGUL(T, X, U, SING)
C  .. Scalar arguments ..
REAL*8      T
C  .. Array arguments ..
REAL*8      X(*), U(*), SING(*)
C=====
C  Purpose:
C    Function to return singularities
C  Input arguments:
C    T      time
C    X      state vector
C    U      input vector
C  Output arguments:
C    SING   singularities
C=====
C  .. Local scalars ..
C  .. Local arrays ..
C  .. Executable statements ..
```

```
RETURN
END
```

APPENDIX 5

SIMULINK[®] TEMPLATE FOR ATEST

APPENDIX 5

SIMULINK® TEMPLATE FOR ATEST

```

C
C      ATMEX.F      ATEST ENGINE MODEL FOR SIMULINK
C
C
C
C=====
C=====
C=====
      SUBROUTINE SIZES(SIZE)
C=====
      INTEGER*4      SIZE(*)
      INTEGER*4      NSIZES
      PARAMETER      (NSIZES=6)

      SIZE(1) = 0    ! number of continuous states
      SIZE(2) = 0    ! number of discrete states
      SIZE(3) = 65   ! number of outputs
      SIZE(4) = 61   ! number of inputs
      SIZE(5) = 0    ! number of discontinuous roots in the system
      SIZE(6) = 1    ! 1 if direct feedthrough, else 0
      SIZE(7) = 0    ! number of "extra" arguments (0-10)

      RETURN
      END
C=====
C=====
C=====
      SUBROUTINE OUTPUT(T, X, U, Y)
C=====
C      .. Scalar arguments ..
      REAL*8      T
C      .. Array arguments ..
      REAL*8      X(*), U(*), Y(*)
C=====
C=====
      MODIFIED ATEST GOES HERE
C=====
C=====
      RETURN
      END

C=====
C=====
C=====
      SUBROUTINE INITCOND(X0)
C=====
      REAL*8      X0(*)

      RETURN
      END

C=====
C=====
C=====

```

```

      SUBROUTINE DERIVS(T, X, U, DX)
C=====
C      .. Scalar arguments ..
      REAL*8      T
C      .. Array arguments ..
      REAL*8      X(*), U(*), DX(*)

      RETURN
      END
C=====
C=====
C=====
      SUBROUTINE DSTATS(T, X, U, XNEW)
      REAL*8      T
      REAL*8      X(*), U(*), XNEW(*)
      INTEGER*4 COEFFS(10),NCOEFFS
      COMMON /EXTRAS/COEFFS, NCOEFFS
      RETURN
      END
C=====
C=====
C=====
      SUBROUTINE DOUTPUT(T, X, U, Y)
      REAL*8      T
      REAL*8      X(*), U(*), Y(*)
      INTEGER*4 COEFFS(10),NCOEFFS
      COMMON /EXTRAS/COEFFS, NCOEFFS
      RETURN
      END
C=====
C=====
C=====
      SUBROUTINE TSAMPL(T, X, U, TS, OFFSET)
      REAL*8      T,TS,OFFSET
      REAL*8      X(*), U(*)
      INTEGER*4 COEFFS(10),NCOEFFS
      COMMON /EXTRAS/COEFFS, NCOEFFS
      RETURN
      END
C=====
C=====
C=====
      SUBROUTINE SINGUL(T, X, U, SING)
      REAL*8      T
      REAL*8      X(*), U(*), SING(*)
      INTEGER*4 COEFFS(10),NCOEFFS
      COMMON /EXTRAS/COEFFS, NCOEFFS
      RETURN
      END
C=====
C=====

```

VITA

Mr. Thomas P. Fetterhoff is an Aerospace Engineer in the Plans and Programs Division, Arnold Engineering Development Center, Arnold Air Force Base, Tennessee.

Mr. Fetterhoff was born in Dauphin, Pennsylvania on July 10, 1962. He attended the Punxsutawney Area High School, in Punxsutawney, Pennsylvania.

After graduating high school in 1980, Mr. Fetterhoff attended the Pennsylvania State University. While attending the Pennsylvania State University: in 1982 he attended the Army Reserve Officers Training School, Basic Camp at Fort Knox, Kentucky; in 1982 and 1983 he worked for Plateau Coal Corporation; and in 1984 and 1985 he interned with the Pennsylvania Department of Transportation. In 1986 he graduated from the Pennsylvania State University with a Bachelor of Science Degree in Aerospace Engineering.

From 1986 to 1990 Mr. Fetterhoff was employed by the Department of the Navy at the Naval Aviation Depot, Naval Air Station Alameda, California. He was assigned to the T56 Turbo-Prop Engine Design and Maintenance Cognizant Engineering group as the T56 Series IV Project Engineer and the T56 Series II, III, and IV Reliability Centered Maintenance Manager. In 1989, he was assigned as the Aircraft Engine and Air Refueling Store Capability Manager and Depot Maintenance Manager Logistics. During this time, he

was responsible for managing all aspects of near and long term planning, depot capability establishment, and the transition of workload from commercial to organic for all Engines and Refueling Stores.

From 1990 to 1994 Mr. Fetterhoff was employed by the Department of the Navy at the Naval Air Warfare Center, Aircraft Division Trenton New Jersey. He was assigned to the Test Operations Division as a Test Engineer. He later transferred to the Propulsion Engineering Directorate, Systems Development and Evaluation Department, Small Engine Division as a Senior Project Engineer. He served as the Project or Test Engineer on the following altitude, sea level, and environmental tests; F405 and F110 Accelerated Simulated Mission Endurance Tests, F404 Quality Assurance Test, T800 Icing and Corrosion Tests, various F107 and F112 Cruise Missile Tests, and various Helicopter Gear Train Tests. He was then assigned to the V22 Integrated Test Team, comprised of both government and contractor personnel, as the propulsion and power systems Flight Test Engineer. He was responsible for the integration of the drive system, engine, fuel system, engine inlet, IR suppressor, and auxiliary power unit into the aircraft.

In 1994 Mr. Fetterhoff became a Department of the Air Force employee in the Commercial Engines Group of the Aeropropulsion Systems Test Division, Test Operations Directorate, Arnold Engineering Development Center, Arnold Air Force Base, Tennessee. Mr. Fetterhoff served as the Project manager for the \$30M Pratt & Whitney PW4000 Growth

Engine Testing, the \$6.6M TF33 JP8 Performance and cold start testing, and the \$2.5M Allison Engine Company AE3007 Testing. Mr. Fetterhoff has also been a member of the Rolls-Royce Trent 800 and Westinghouse Combustor Test teams.

In August 1997 Mr. Fetterhoff was assigned to the Plans and Programs Division at Arnold Engineering Development Center. He is the Long Range Requirements and Facility Planner for the Air Breathing Propulsion Test Facilities. He is responsible for ensuring that new test technologies and test facilities are available to meet the test requirements of future Military and Commercial Air Breathing Propulsion Systems.

Mr. Fetterhoff expects to receive his Masters in Mechanical Engineering from The University of Tennessee in December 1997. He has begun the coursework for a Doctorate in Aerospace Engineering.

Mr. Fetterhoff is a member of the American Institute of Aeronautics and Astronautics (AIAA), The Society of Automotive Engineers (SAE), and American Helicopter Society (AHS).