NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

A BASELINE EXPERT CONTROL SYSTEM FOR
MARINE GAS TURBINE
COMPRESSOR SURGE

by

James A. Davitt

March 1989

Thesis Advisor David L. Smith

Approved for public release; distribution is unlimited.

REPORT DOCUMENTATION PAGE

Report Security Classification Unclassified		1b Restrictive Markings	
Security Classification Authority		3 Distribution Availability of Report	
Declassification Downgrading Schedule		Approved for public release; distribution is unlimited.	
Performing Organization Report Number(s)		5 Monitoring Organization Report Number(s)	
Name of Performing Organization Naval Postgraduate School	6b Office Symbol *(if applicable)* 52	7a Name of Monitoring Organization Naval Postgraduate School	
Address *(city, state, and ZIP code)* Monterey, CA 93943-5000		7b Address *(city, state, and ZIP code)* Monterey, CA 93943-5000	
Name of Funding Sponsoring Organization	8b Office Symbol *(if applicable)*	9 Procurement Instrument Identification Number	
Address *(city, state, and ZIP code)*		10 Source of Funding Numbers	
		Program Element No / Project No / Task No / Work Unit Accession No	

Title *(include security classification)* A BASELINE EXPERT CONTROL SYSTEM FOR MARINE GAS TURBINE COMPRESSOR SURGE

Personal Author(s) James A. Davitt

a Type of Report	13b Time Covered	14 Date of Report *(year, month, day)*	15 Page Count
Master's Thesis	From To	March 1989	82

Supplementary Notation The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

Cosati Codes			18 Subject Terms *(continue on reverse if necessary and identify by block number)*
eld	Group	Subgroup	expert system, marine gas turbine, casualty control

Abstract *(continue on reverse if necessary and identify by block number)*

United States Navy gas turbine ships are in need of casualty control system updating to reduce demanding conditions on ngineering watch standers, to increase equipment longevity, and reduce operating costs. This thesis presents a baseline omputer-based expert system controller concept developed for the critical casualty control problem of gas turbine ompressor surge. The controller design rests on the building-block components of real-time gas turbine simulation and ompressor surge characterization, which are discussed. The logic and rules for the expert system design are presented, as is dynamic investigation of the expert system diagnostic performance.

0 Distribution Availability of Abstract ☒ unclassified unlimited ☐ same as report ☐ DTIC users	21 Abstract Security Classification Unclassified	
2a Name of Responsible Individual David L. Smith	22b Telephone *(include Area code)* (408) 646-3383	22c Office Symbol 69Sm

D FORM 1473,84 MAR 83 APR edition may be used until exhausted security classification of this page

Approved for public release; distribution is unlimited.

A Baseline Expert Control System for Marine Gas Turbine
Compressor Surge

by

James A. Davitt
Lieutenant, United States Navy
B.S., United States Naval Academy, 1982

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN MECHANICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
March 1989

ABSTRACT

United States Navy gas turbine ships are in need of casualty control system updating to reduce demanding conditions on engineering watch standers, to increase equipment longevity, and reduce operating costs. This thesis presents a baseline computer-based expert system controller concept developed for the critical casualty control problem of gas turbine compressor surge. The controller design rests on the building-block components of real-time gas turbine simulation and compressor surge characterization, which are discussed. The logic and rules for the expert system design are presented, as is a dynamic investigation of the expert system diagnostic performance.

TABLE OF CONTENTS

LIST OF FIGURES

TABLE OF SYMBOLS

A. NOMENCLATURE

$A \equiv$ compressor inlet area

$\underline{A} \equiv$ state coefficient matrix

$\underline{B} \equiv$ input coefficient matrix

$\rho \equiv$ density

$\delta \equiv$ pressure correction factor

$\underline{f} \equiv$ vector of nonlinear state functions

$g_c \equiv$ gravity constant

$K \equiv$ flow coefficient

$m \equiv$ mass flow rate

$N \equiv$ rotational speed

$P \equiv$ pressure

$\pi \equiv$ pressure ratio

$Q \equiv$ torque

$\mathcal{R} \equiv$ gas constant

$T \equiv$ temperature

$\theta \equiv$ temperature correction factor

$u \equiv$ pertubed input

$U \equiv$ total input

$x \equiv$ perturbed state

$Y \equiv$ expansion factor

$X \equiv$ total state

B. SUBSCRIPTS

$a \equiv$ air

$e \equiv$ exhaust

$f \equiv$ fuel

$G \equiv$ gas generator

$i \equiv$ linearization point

$L \equiv$ load

$S \equiv$ shaft

$std \equiv$ standard

$1 \equiv$ compressor inlet

$2 \equiv$ compressor discharge

I. INTRODUCTION

With today's U.S. Navy committed to gas turbine propulsion, as evidenced by five classes of ships having General Electric LM-2500 engines, the need has arisen for improvements to these power plants to ensure their continued effective operation. In particular, engineering watchstanders and operators on these various platforms often work under demanding, stressful conditions which make it difficult to effectively and quickly diagnose potential casualties. Further, these often require rapid, precise, and spontaneous corrective action. A recent study conducted for the Navy Personnel Research and Development Center found that personnel were no longer able to fulfill the growing demands imposed upon them in their roles as operators of gas turbine propulsion units [Ref. 1: p.203]. Their greatest assistance needs were found to be in the areas of fault diagnosis and in the determination of proper corrective responses. The study went on to suggest that these two areas of difficulty may be effectively handled by the combined use of automation and computer expert systems. An improved, automated expert system casualty control system would not only reduce the demanding burdens on the operators, but would, for the same reasons, additionally increase equipment longevity and reduce overall operating costs.

Developing a marine gas turbine expert system capable of casualty control capabilities (that is, able to correctively diagnose severe machinary degradation and eliminate or minimize potential damage,) is a multifaceted project, undoubtedly requiring a vast amount of time and expertise. Before undertaking such a monumental task, the feasibility of such a system should be successfully demonstrated. One facet of the turbine casualty control problem is the problem of compressor surge. In support of this approach, a system validation was addressed by testing an expert system surge casualty

control design on a low power gas turbine plant. The low power gas turbine test bed was a critical part of the baseline casualty control expert system as it was assumed to provide half the system input data through direct measurements. The remaining system inputs were provided by a real time simulator, which was modelled after the plant. Comparison of the real and simulated plant data formed the basis for the expert system diagnosis. Consequently, in order to achieve a fully competent controller, the required real time simulator must be accurate and reliable over the gas turbine's entire operating range. Unlike data trends or plots commonly used in steady state condition monitoring applications, the justification of an accurate real time simulator lies in the fact that naval gas turbine vessels predominately operate in a dynamic, non steady state mode [Ref. 2: pp.286-287]. A control scheme was also required to utilize the output diagnostic information for proper casualty control. Several schemes to this are presented below, even though this was not an important part of the work presented.

An important element of the baseline expert system casualty control system was the software utilized. The construction of an expert system was simplified by the use of a user friendly expert system shell, such a tool is NASA's "C" Language Production System (CLIPS) [Ref. 3: p.1]. One of the appealing features of CLIPS, and all expert system programs, was its adaptibility. For example, rather than having to construct entirely new software when a problem expands or takes on new dimensions, expert system software written to solve one particular problem can be easily supplemented to accommodate additional constraints. Therefore, by successfully demonstrating the capability of a baseline expert control system for one particular casualty, as in the present study, the feasibility of a multiple casualty expert control system is realized through this expert system language adaptability feature.

II. EXPERT SYSTEM BACKGROUND

The specialized field of expert systems has developed from the Artificial Intelligence (A.I.) branch of Computer Science. Specifically, this branch deals with expanding a machine's ability to perceive and reason, or accomplish tasks that appear to require intelligence [Ref. 4: p.1]. Expert systems are computer programs that solve complex, real-world problems which require significant human expertise to interpret and solve. The solution of these involved problems is accomplished by the computer program's ability to simulate human reasoning and arrive at similar conclusions or solutions as would experts in the field which the problem exits. The program's expert reasoning capability is largely accomplished through utilizing a vast body of knowledge pertaining to the area of interest, and ranging from general to highly specific information. Generally speaking, an expert system consists of the following three parts [Ref. 5: p.2]:

1. A knowledge base consisting of facts and heuristics associated with the problem.
2. An inference or reasoning procedure for controlling knowledge information flow, by utilizing that information to draw conclusions.
3. A working memory or data base for data and problem status history.

The facts within the knowledge base are a commonly known, widely agreed upon collection of information; while the heuristics are rules-of-thumb known only by the "experts", obtained through expertise in the field of interest. Generally speaking, the larger and more complex expert system knowledge bases are, the more they are capable of solving large, more complex problems. Representation of a simplified expert system structure is shown in Figure 1.

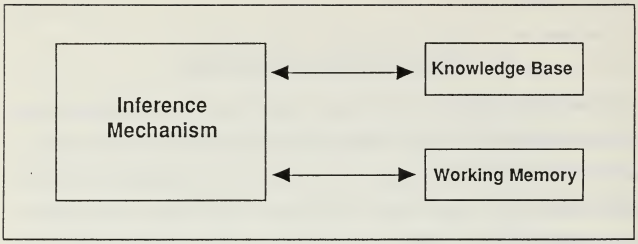

Figure 1. Expert System Components and Information Flow Direction

Three fundamental characteristics distinguish an expert system from conventional programs. First, while conventional programs make no distinction between knowledge and how knowledge is used, expert systems clearly seperate the two. Secondly, expert systems utilize "inexact" reasoning or information that may not be one hundred percent true. Lastly, expert systems can be easily amended or their knowledge base increased through incremental modifications, while changes to conventional programs are often painstaking [Ref. 6: p.5]. This last feature is appealing in the sense that an expert system used to diagnose and control one particular casualty can easily have its knowledge base broadened to accommodate further casualties added to the systems diagnostic and control menu.

Although all expert systems consist of the three aforementioned core features, expert systems differ from one another in the choice of solution direction, which is a function of the inference mechanism. Expert system problem solving techniques fall under the following categories: forward chaining, backward chaining, forward and backward processing combined, and event driven. For the purposes of brevity, only forward and backward chaining will be discussed because they are the two primary problem solving

methods. Forward and backward processing utilize a combination of backward and forward chaining to solve complex problems, and event driven processing is very similar to forward chaining.

Before discussing the differences of forward and backward chaining some basic expert system terminology should be understood. As discussed earlier, expert system knowledge bases are composed of facts and heuristics provided by the field experts; in expert programming language, this information is known as "rules", and input or created data are termed "facts". In forward chaining, the control strategy is initiated with a list of facts (or data) and utilizes rules (from the knowledge base) to arrive at a possible solution to the particular problem. Forward working systems are oftem characterized by being initiated by a small number of facts and are able to reach a large number of potential conclusions. A simple example of forward chaining is given below.

1. Car won't start (Input data)
2. If car won't start, then battery dead (Rule 1)
3. If battery dead, then need new battery (Rule 2)
4. Need new battery (Conclusion)

Forward chaining is often used for equipment or machinery diagnostic applications, and was used in the production of the present gas turbine baseline casualty control expert system.

Rather than starting with data, or a fact list, backward chaining commences with a particular goal or solution in mind, and works backwards. From existing rules it is determined what facts are necessary to obtain the particular stated goal. Existing facts are then checked to decide if the goal is correct, and if facts are not available the user is asked to answer questions to generate facts and obtain a conclusion [Ref. 7: p.22]. An example of backward chaining is given below.

1. Need new battery (Hypothesized Conclusion)
2. If need new battery, then battery is dead (Rule 1)

3. If battery is dead, then car won't start (Rule 2)
4. If car won't start, then need new battery (Rule 3)
5. Will the car start? (Data)

Just as some problem solving techniques work better in certain situations, the same holds true with expert systems; in some applications expert system utilization is infeasible or impractical, while in other cases the problem may be too complex or involved to be handled in an ordinary, non-expert system manner. The potential domain of expert system utilization is rather precise. The following list of expert system characteristic criteria should be met prior to undertaking development [Ref 8: p.6]:

1. Genuine experts exist in the field
2. Existing experts are much better than amateurs
3. Skill must be routinely taught to amateurs/novices
4. Task must be within reason
5. Task should not require common sense
6. Undertaking should have sufficient payoff to warrant construction

In the gas turbine arena there has been a number of recent expert system related developments initiated by the U.S. military. For example, in order to improve the reliability and maintainability of gas turbine engines in the United States Air Force inventory, the Air Force has been testing a new knowledge-based (expert system) diagnostic system which utilizes gas turbine vibration analysis data to diagnose rotor dynamic faults. The diagnostic concept developed was successfully demonstrated on a test rig when the integrated system with its implemented diagnostic logic and vibration data aquisition system successfully diagnosed five inputted faults including: rotor unbalance, misalignment, rub, increased support fexibility and accessory vibration [Ref. 8: p.7]. Additionally, the U.S. Army is developing an expert system which utilizes forward and backward reasoning to diagnose faults in a twin engine gas turbine helicopter power

train from instrument panel data readings, thereby decreasing the evergrowing workload demands on the single cockpit pilot [Ref. 9: p.19].

The Navy got involved in a shipboard gas turbine condition monitoring project in 1974 concerned with reliability of monitoring hardware, which had to improve by a factor of ten before the system could be used as an effective trouble shooting tool [Ref. 10: p.9]. In the last decade, however, technology and controller-sensor reliability have substantially increased. Expert system tools such as the NASA/Johnson Space Center's 'C' Language Production System (CLIPS) are currently available. CLIPS is a user friendly, forward chaining expert system computer language designed for writing expert system applications. CLIPS was designed for portability, as it may run on a range of microcomputers with little or no software modifications necessary. CLIPS can be easily integrated at virtually no cost to external systems, and was found to be quite compatable for use in a marine gas turbine expert control system. The use of CLIPS to develop a proper expert system will be discussed following a description of the compressor surge problem.

III. COMPRESSOR SURGE

Rather than examining a vast number of possible engine casualties, attention was focussed on one particular casualty which greatly effects the entire engine if not detected. Compressor surge was chosen for this effort because of its relation to compressor stall, since stall has been implicated for one of every five unscheduled engine removals in the LM-2500 gas turbine program [Ref. 11: p.13]. However, before an adequate casualty control expert system for the prevention of compressor surge can be developed, a clear understanding of the surge phenomenon must be obtained. Though often used interchangeably, compressor stall and surge are not one in the same. Basically, compressor stall can be descibed as a momentary disruption of the normal air flow rate through the compressor, while surge is a more severe form of stall. During surge, air flow through the compressor undergoes an actual reversal in flow direction. However, the mechanics of surge are quite involved and not yet fully understood; even so, stall has a clear mechanical relationship to surge and it is a widely held belief that stall may initiate a surge condition [Ref. 12: p.113].

The most common type of compressor stall is known as rotating stall. Rotating stall can be initiated by the stalling of a single blade. A slight distortion or uneven distribution of velocity at the inlet to the compressor may cause a blade's incidence angle to increase, changing from i to i' as shown in Figure 2.

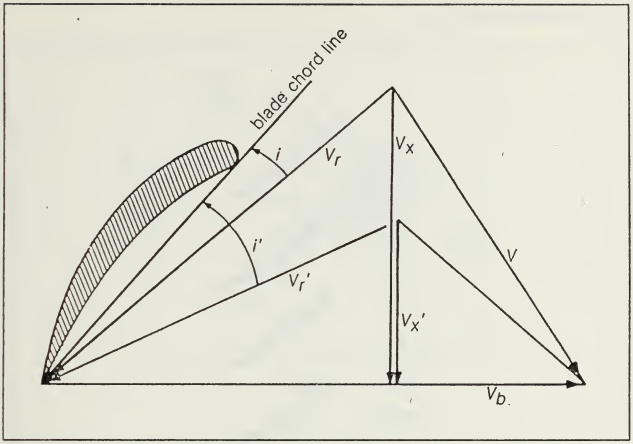

Figure 2. Increased Incidence Angle

At some increased incidence angle, which is difficult to define, flow separates from the suction side of the blade and the blade stalls [Ref. 13: p.67]. The stalled blade, due to the separated flow, then creates or causes an area of retarded or blocked flow which, in turn, increases the incidence angle on the blade adjacent to the stalled blade, in the direction opposite of blade rotation. The retarded flow, however, decreases the incidence angle of the blade next to the stalled blade in the direction of rotation, returning it to its normal angle of attack and restoring flow around the blade. The area of retarded flow travels from blade to blade, and the stalling and unstalling action propogates within the same row of rotating blades, creating the condition of rotating stall. Serious blade damage may result if the loading and unloading of the rotating blades approaches the

blades' natural frequency. The rotating or propagating stall, as it is sometimes refered to is illustrated in Figure 3.

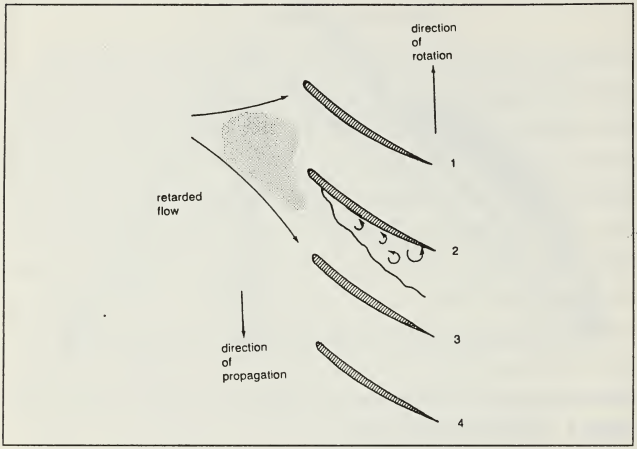

Figure 3. Rotating Stall

The most distinguishing characteristic between rotating stall and surge is the air mass flow rate. While the total mass flow rate during a rotating stall remains relatively constant, a surge condition exhibits an oscillation or pulsation of the air flow. A widely accepted theory on the surge phenomenon's trigger mechanism is lacking, although many contribute it to rotating stall [Ref. 12: p.113]. What the experts do agree on, however, is the necessary condition for surge; that being a positive instantaneous pressure ratio/mass flow gradient, which for a normal operating compressor is negative. This prerequisite surge condition and a description of what occurs during a surge cycle can be understood by examining the theoretical compressor curve shown in Figure 4.

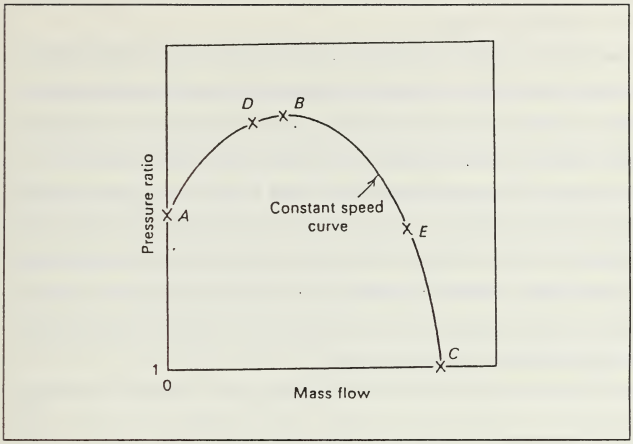

Figure 4. Theoretical Characteristic Curve

Consider a theoretical valve placed in the delivery line of a compressor running at constant speed. As the valve is slowly openned from zero flow (point A) the pressure ratio increases to a maximum (point B). Additional increases in mass flow, achieved by openning the valve further, result in a falling off of pressure ratio until it reaches zero (point C). This pressure ratio drop is caused by the deviation of the particular air mass flow rate from the design flow rate, which creates air-blade angle missmatching, and causes a decrease in compressor efficiency. [Ref. 12: p.111].

The portion of the curve between points E and C must be operationally avoided because of the much severe (negative) slope of the characteristic curve. In this region, pressure ratio drops rapidly with only a slight increase in air mass flow rate through the

compressor. Operating along this portion of the curve would result in a condition known as compressor choking.

Normally, the compressor would operate between points E and B, where the slope is negative and less severe, and where small deviations in mass flow rate do not cause drastic pressure ratio movement. The negatively sloped operating curve dictates an increase in pressure ratio for a drop in air mass flow rate through the compressor.

Contrarily, when operating between points B and A, where the characteristic curve slope is positive, any decrease in the flow rate will result in a decrease in pressure ratio. This action can be seen in Figure 4, by running along the curve from B to A and noting the relation between flow rate and pressure ratio. In actual (non theoretical) compressor operation, however, this portion of the curve from B to A is relatively unusable due to the surge phenomenon. As shown in Figure 5,

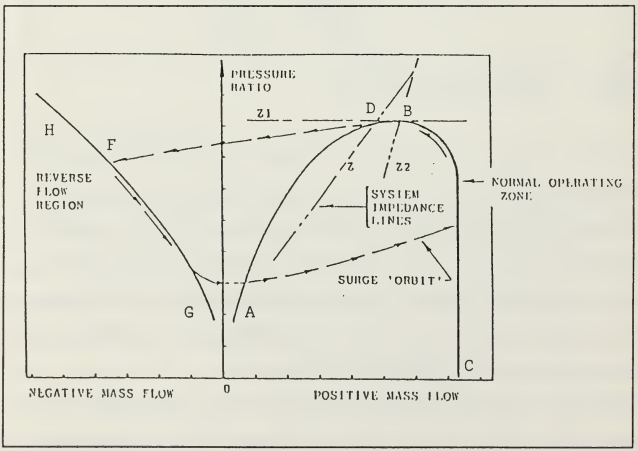

Figure 5. Surge Orbit

before the operating point is able to move down the positive slope portion of the curve, the mass air flow rate has already reversed its direction through the compressor, (moved from B to F) as depicted in the figure. This flow reversal can be explained in the following manner. Suppose, for example, the compressor is temporarily fixed at point D (Figure 4), again, noting that the slope of the characteristic curve in this region is positive. If, while operating at this point, a decrease in mass air flow is experienced, an accompanying drop in pressure ratio or delivery pressure would occur (as expected by the curve). If at that instant, though, the existing pressure immediately downstream of the compressor exit did not fall as quickly as that of the the air in the compressor, then the air will tend to take the path of least resistance, reverse its direction and flow backwards through the compressor - in the direction of the resulting negative pressure gradient. This reversal in flow direction causes a rapid drop in the pressure ratio (or delivery pressure), which creates a fall in the pressure downstream of the compressor, thus returning the airflow back to its normal direction through the compressor. The surge cycle is now in a position of repeating, provided the operating point remains on or is influenced to the positive side of the characteristic curve, and the delivery pressure falls at a greater rate than the pressure downsteam of the compressor. The surge cycle (or "orbit") described above is illustrated in Figure 5.

A well established method to prevent the onset of surge is to use a blow-off (pressure relief/recirculation) valve when the compressor's back pressure exceeds a preset limit. This is discussed in more detail in Chapter 6, but it is not a simple matter to implement in practice. The need for rapid control to recover may be met with an expert system that can precisely determine the onset of the casualty condition.

IV. PLANT DESCRIPTION

The test bed utilized in the development of the baseline expert surge control system was the gas turbine at the Naval Postgraduate School. This plant, which is commonly used for gas turbine laboratory experiments, consists primarily of a Boeing model 502-2E gas turbine engine coupled to a Clayton 17-300 water brake dynamometer. The test bed is illustrated in Figure 6.

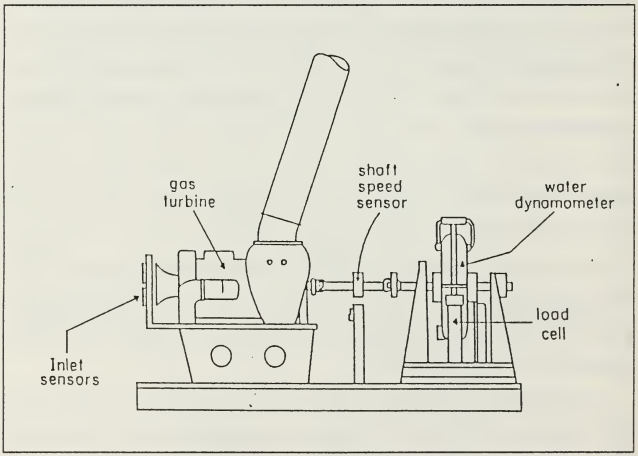

Figure 6. Test Bed Installation

The Boeing model 502-2E gas turbine engine is rated at 175 brake horsepower and consists of two primary sections: a gas producer and a power output section. The gas

producer contains a single stage, single entry, centrifugal compressor, two cross connected through-flow combustion chambers, a single-stage high pressure axial flow turbine (HPT), and an accessory drive section. The power output section consists of another single stage, axial flow, free power turbine (FPT), reduction gears and an output shaft. The power output shaft is aerodynamically coupled by a duct which directs the flow of gasses from the HPT to the FPT. This arrangement allows the HPT or gas producer to be controlled more or less independently of the output shaft speed. [Ref. 14: p.1]

The output shaft is mechanically coupled to a Clayton 17-300 water dynamometer which absorbs the power generated by the engine. Power absorbed is regulated by the quantity of water allowed in the dynamometer shell. This absorption action is achieved by a rotationg impeller attached to the shaft which produces a torque that is proportional to the water level in the dynamometer. The greater the amount of water, the greater the surface area acting on the impeller, and the greater the torque and power absorbed.

The gas turbine engine has two control modes; a marine emulator mode and a manual mode. In both modes engine control is divided into two areas; gas generator control and load control. The gas generator control is accomplished primarily through the use an analog flyball governor, which dictates the motion of a motor-driven throttle positioner. Required input to the flyball governor is the desired gas generator speed, which is translated to fuel flow output. The load control is a digital, proportional-plus-derivative controller which regulates the amount of water in the dynamometer through manipulating water inlet and outlet valves. [Ref. 15: p.16]. Required input for the load controller is desired dynamometer speed.

When using the manual control mode the operator simply inputs the desired gas generator speed and the desired dynamometer speed. When operating the engine utiliz-

ing the marine emulator mode, however, the operator simply inputs the desired dynamometer speed and the desired load or propeller pitch. The control system automatically determines the gas generator speed to fit the desired dynamometer speed and load. Again, the desired load is controlled by the regulated water level in the dynamometer.

Looking at the Boeing gas turbine power plant in comparison to those on board Navy gas turbine ships, two important features are similar, thus enhancing applicability of the baseline expert casualty control system. These two similar characteristics are the two engine control systems and the engines component configurations.

The marine emulator control mode is similar to the FFG-7 class propulsion control system, shown in Figure 7.

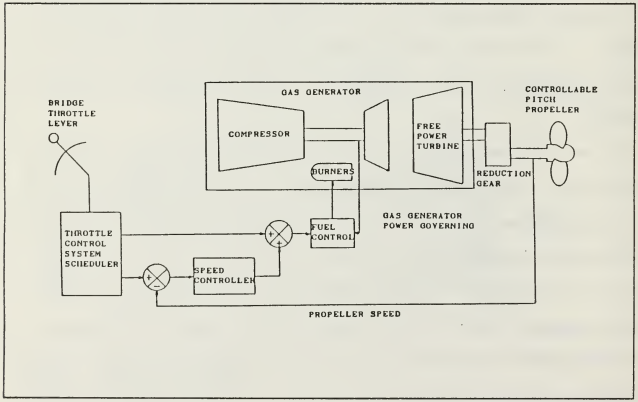

Figure 7. Simplified Propulsion Control System

On the FFG-7 class, the propulsion control system (PCS) can control the gas turbine engine power and propeller pitch via three control modes: local manual, remote manual or programmed control. In the programmed control mode, which is the primary operating mode, gas turbine power and propeller pitch are automatically controlled by a single control lever. Similarly, the NPS marine emulator control system has load control capability, (akin to the propeller pitch system), and output shaft speed control.

The other characteristic which is common to both operating plants is engine component composition. Having already descibed the makeup of the NPS Boeing model 502-6A, the G.E. LM-2500 gas turbine engine has a similar component arrangement. Like the Boeing, the LM-2500 consists of a two major sections; the gas generator and power turbine. The gas generator portion consists of an axial flow compressor, combustor and a high pressure turbine. The compressor and high pressure turbine are connected by a single shaft, (as with the Boeing). The exhaust gases from the high pressure turbine are directed through the free power turbine, with no mechanical connection existing between the two turbines, again similar to the coupling on the Boeing engine.

V. REAL TIME SIMULATOR

The real time simulator for the N.P.S. Boeing gas turbine engine is largely the product of previous work conducted by faculty and students, the following desciption of the real time simulator's composition is primarily a brief review of that work. A more thorough explanation can be attained from [Ref. 16]. An understanding of the mechanics behind the real time simulator was considered important due to the thought that a similar structured simulator could be constructed for the LM-2500 gas turbine engine, and because the real time simulator is such an integral part of the casualty control expert system.

Requirements for the real time simulator were accuracy and quickness. These two requirements do not necessarily complement one another, because a simple model may be quick, but have questionable accuracy. It was, however, felt that sufficient accuracy could be attained from a quasi-linear model, even though a large quantity of literature exists on the incompatability of dynamic gas turbine engines and linear models. In wrestling with the nonlinear, dynamic environment a sequential state space linearization technique was utilized in developing the model of the engine. The procedure utilized is summarized in the list below. [Ref. 16]

1. Determine the nonlinear relations between the individual components inputs and outputs as shown in Figure 8.
2. Linearize the models of each component.
3. Create an overall linearized system model by combining the linearized components.
4. Test the model accuracy and speed with simulated system performance.

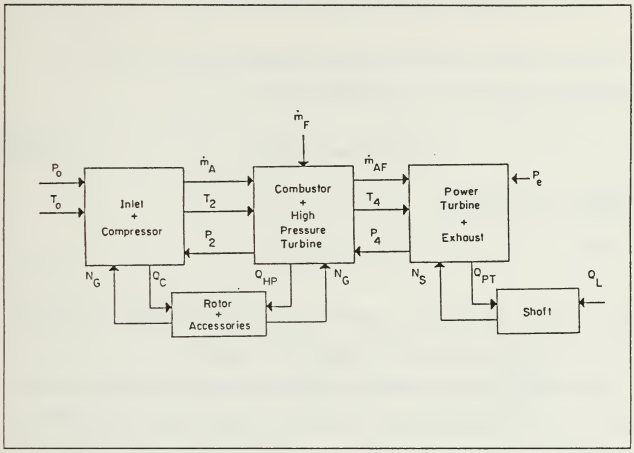

Figure 8. Cause and Effect Plant Model

Component modelling was accomplished primarily with simple algebraic equations. Instantaneous, non-dynamic component outputs were modelled as complete quadratic functions in terms of input variables, with instantaneous constants determined experimentally with steady state data.

$$\dot{N}_G = (Q_{HP} - Q_C)/J_G \tag{5.1}$$

Components with considerable time lags, such as shaft rotor and combustor were modelled with rate equations, as above. The dynamic constants in these equations were determined from dynamic experimentation. An example equation for a non-dynamic variable is given below: [Ref. 16]

$$Q_C = aN_G^2 + bP_2^2 + cN_GP_2 + dN_G + eP_2 + f \tag{5.2}$$

Component output equations were linearized using perturbation analysis. The perturbed version of equation (5.1) is as follows:

$$\partial \dot{N}_G = (\partial Q_{HP} - \partial Q_C)/J_G \tag{5.3}$$

Perturbations were defined as:

$$\partial N_G = N_G - N_{Gi} \tag{5.4}$$

Using steady state operating data, linearization points were established in order to linearize the equations in terms of the perturbed values. [Ref. 16]

A state space form was utilized in combining the linearized component model equations to create the overall linear system model. Through numerical and physical experimention, a minimum list of variables or states were determined which were necessary to determine the dynamic and steady state operating characteristics. These states were found to be shaft speed, gas generator speed and the combustion energy state.

$$X(1) = N_G \tag{5.5}$$

$$X(2) = N_S \tag{5.6}$$

$$X(3) = E \tag{5.7}$$

Equally important as the states in attaining a desired plant performance are the input variables, these were selected as fuel flow rate and load torque. Since establishing the nonlinear relation between the states and inputs is too complex a task, the dynamic behavior of the plant may be attained through a series of linearizations of the nonlinear

first order differential equation expressing the relation between the states and inputs as shown: [Ref. 16]

$$\dot{\underline{X}} = \underline{A}\underline{x} + \underline{B}\underline{u} \tag{5.8}$$

where:

$$x = \partial X \tag{5.9}$$

$$u = \partial U \tag{5.10}$$

Expansion of equation (5.8) yields:

$$\begin{bmatrix} \partial \dot{N}_G \\ \partial \dot{N}_S \\ \partial \dot{E} \end{bmatrix} = \begin{bmatrix} A(1,1) & A(1,2) & A(3,1) \\ A(2,1) & A(2,2) & A(3,2) \\ A(3,1) & A(2,3) & A(3,3) \end{bmatrix} \begin{bmatrix} \partial N_G \\ \partial N_S \\ \partial E \end{bmatrix} + \begin{bmatrix} B(1,1) & B(1,2) \\ B(2,1) & B(2,2) \\ B(3,1) & B(3,2) \end{bmatrix} \begin{bmatrix} \partial \dot{m}_f \\ \partial Q_L \end{bmatrix} \tag{5.11}$$

The "A" and "B" matrixes were attained analytically by utilizing knowledge of the states at desired linearization points and using the quadratic component equations to balance the turbine to steady state and determine the remaining plant variables. The attained plant variables were then substituted into linearized component equations to determine matrix entry values. A set of matricies was established for the entire operating envelope, to reveal how the plant's linearization varied. The B matrix was found to be constant and curve fits were conducted to establish equations for the A matrix elements in terms of the states (gas generator and shaft speeds). The B matrix values and the A matrix equations can be seen in the real time simulator program contained in Appendix A. [Ref. 16: p.5]

Testing of the simulator clearly indicated that the sequential linearization technique provided a better response then the single linearization method, and the results were

quite good when compared with actual data when tested at constant gas generator speeds while shaft speed was varied. To estimate computational time constraints a timing study was conducted using the code contained in Appendix A. Plant input values for load and fuel flow rate were not included in the program as they would be directly input into the program through a "read" statement during each program iteration. Read time was assumed to be negligible. Using a 32 bit processor a 100 second simulated run with a 0.001 second time step took roughly 13 secords or 0.13 seconds of computation time for each second of simulation time [Ref. 16: p.6.].

In order to test the expert surge control system, the code in Appendix A had to be modified. This modification included changing the FORTRAN code to C code, inputting values for load and fuel flow rate, and inserting into the program equations to generate the compressor discharge pressure and compressor air mass flow rate from the state values at the time of interest. When considering how the load and fuel flow values should be input, the thought of reading the two values from a data file was ruled out due to the enormous size the data file would have to be to accommodate even a simulated ten second run. The problem was handled by approximating the input time histories using simple linear equations derived from actual dynamic data taken off a strip chart. Since naval gas turbine plants do not operate dynamically at constant gas generator (compressor) speeds when operating in the speed control or power control mode, the baseline expert surge control system testing should be accomplished while the compressor was in a dynamic state. This testing constraint was accomplished by varying the load at constant power turbine speed. This arrangement is akin to the speed control mode naval gas turbine ships operate under for speeds between zero and ten knots. In this speed interval (0-10 kts) shaft and power turbine speed is held constant, and ship speed changes are accomplished by varying the propellor pitch. The change in load is satisfied by an increase or decrease in gas generator speed.

The equations for compressor outlet pressure and air mass flow rate were required to be included in the simulator code because, along with compressor speed, they are critical variables in determining the compressor operating point in order to access potential surge onset. Equations for outlet pressure and flow rate were computed using a curve fitting technique from steady state taken taken over a wide range of operating conditions. The non-normalized equations for compressor outlet pressure (psia) and air mass flow rate (lbm/sec) are given below:

$$P_2 = 14.0[0.504 - 1.44N_G + 0.0134N_S + 1.81N_G^2] \tag{5.12}$$

$$\dot{m}_A = 2.36[-0.176 + 1.16N_G + 0.022(P_2/14.0)] \tag{5.13}$$

When testing the modified real time simulator code with strip chart data, dynamic data under constant shaft speed and load varying, a scaling factor error was encountered, and the time step had to be changed from 0.001 to 0.000368 which yielded very good agreement with recorded data measurements. On a 16 bit processor, using an optimizing, turbo C compiler, a simulated ten second run took 47 seconds of computation time. Or for each second of simulated time 4.7 seconds of computation time was required. The modified C language version of the real time simulator code can be found in Appendix B.

VI. EXPERT SYSTEM DESIGN

Having been introduced to expert systems, the surge phenomena, the Boeing 502-6 gas turbine engine, and the background of the Boeing real time simulator, the framework of the baseline expert system to diagnose and prevent compressor surge onset can be discussed. The basic principle of the system is illustrated in Figure 9.

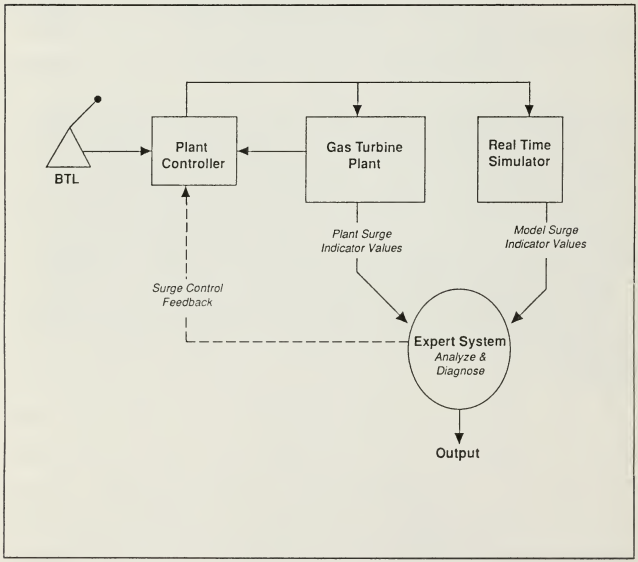

Figure 9. Expert System Framework

As illustrated, the same inputs were provided to both the engine (or plant) and the real time simulator (or model). Output was received from the plant and model by a central processor (or microcomputer). There, the two sets of data were massaged into identical indicators and tolerance windows were applied to the model indicators. The data sets were then compared and, based on deviation analysis, a control decision was reached. With these basic functions in mind a five step approach was used in developing the anti-surge casualty control expert system, as listed below:

1. Select key engine health indicators, including measured and calculated parameters.
2. Determine and apply indicator tolerances.
3. Compare key engine health indicators with the same parameters provided by the real time simulator, and determine respective health indicator deviations.
4. Evaluate and analyze the indicator deviations.
5. Based on the deviation analysis determine corrective control action.

The remainder of this chapter is devoted to discussing the composition of these steps and their integration into the total expert system package.

As was described in Chapter Three, the compressor surge phenomena is a complex condition, not yet fully understood. Nevertheless, there was one facet of the surge condition discussed earlier that the experts did agree on, that being the principle condition necessery for surge onset, namely, a positive instantaneous pressure ratio mass flow gradient. In other words, if the compressor operating point lies on the positive portion of the compressor's characteristic curve (at a given compressor speed) the main prerequisite for surge has been met. The key surge indicators, therefore, were chosen to be those parameters which established position on the standard compressor map; namely, compressor speed, air mass flow rate through the compressor, and compressor inlet and discharge pressures; which together determine pressure ratio. A hypothetical compressor performance map is shown in Figure 10.

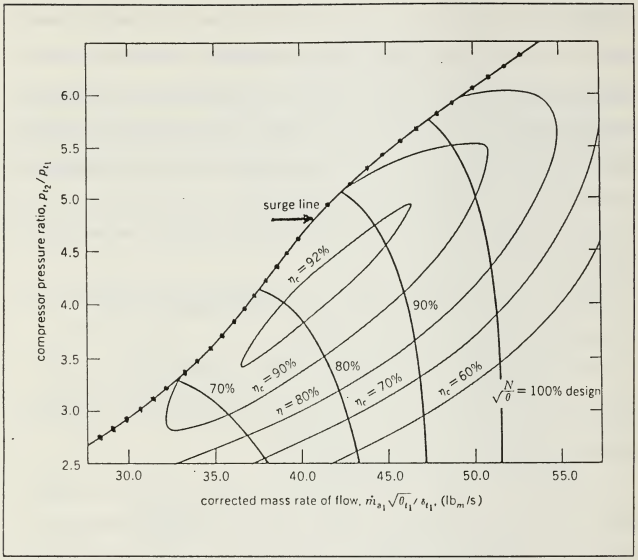

Figure 10. Hypothetical Compressor Performance Map

It should be noted that standard correction factors for temperature and pressure have been applied to accommodate compressor operations under varying conditions. It is conceivably possible for a compressor not to go into surge immediately when the compressor's operating point has crossed the surge line as shown in Figure 10 due to a particular engine's air flow swallowing capacity achieved by components downstream of the compressor (thus maintaining the required negative pressure gradient through the compressor). However, under normal design conditions it is simply a matter of time

before a reduction in air mass flow rate is introduced and the related pressure reduction causes the pressure gradient to shift, resulting in the undesirable flow reversal. Further, this is why compressors normally operate in the negative slope regions of their characteristic curves (figure 4), where a decrease in air mass flow rate is accompanied by a rise in delivery pressure, thus ensuring stability of operation [Ref. 12: p.112]. The swallowing capacity of the engine downstream of the compressor is important in preventing surge from occurring, and will be utilized and discussed in the control process. With this in mind, the key factor used in the baseline expert system for diagnosing compressor surge potential onset was a significant disagreement between the ideal behavior expressed by the model, and the actual measured engine behavior. If disagreement existed it was further analyzed to determine if the actual operating point was located closer to the surge line then called for by the model (indicating surge potential) or further from the surge line (indicating a non-surge potential condition).

Since the desired outputs of the real time simulator have been identified, the task has become one of modifying the model to provide compressor speed, mass air flow and outlet pressures from the states. This operation was previously discussed at the end of Chapter Five (equations (5.12) and (5.13)).

The actual plant, however, does not allow as much flexibility as the model in that all of the required surge indicator outputs are not directly accessible through sensor measurements. Although compressor speed and inlet and outlet pressures can be measured, the air flow rate through the compressor must be arrived at indirectly. Therefore, it was assumed that the aerothermal indicators which the actual plant sends to the processor would be compressor speed, compressor discharge pressure, compressor inlet pressure, and nozzle pressure. Output from the simulator included compressor model speed, compressor model air mass flow rate, and compressor model discharge pressure.

The first step in the data preparation process was to establish identical indicators from model and plant. Namely, actual (plant) compressor speed would be compared to model speed, actual compressor pressure ratio would be compared with model (simulator) pressure ratio, and actual air mass flow rate compared to model flow rate. To accomplish this, compressor pressure ratio for both model and the plant, as well as plant air mass flow rate through the compressor, needed to be computed. Since model compressor outlet pressure was computed by the simulator, model pressure ratio was easily determined by dividing the model outlet pressure with the actual measured compressor inlet pressure. Similarly, plant pressure ratio was computed by dividing measured compressor outlet pressure by measured inlet pressure. Since plant inlet and nozzle pressures were measured directly, plant air mass flow rate could be calculated utilizing a simplified version of the mass flow rate equation for compressible flow through a nozzle [Ref. 17: p.237]. (Note that equation simplification was allowed since the differential pressure over inlet pressure ratio was less than 0.10):

$$\dot{m}_a = YKA\sqrt{2g_c\rho(P_1 - P_N)} \tag{6.1}$$

Applying the standard temperature and pressure correction factors to equation (6.1) yields:

$$\dot{m}_a \frac{\sqrt{\theta}}{\delta} = YKA\sqrt{2g_c\rho(P_1 - P_N)}\sqrt{\frac{T_1}{T_{std}}}\,\frac{P_{std}}{P_1} \tag{6.2}$$

Also, since:

$$\rho = \frac{P_1}{\mathcal{R}T_1} \tag{6.3}$$

equation (6.2) can be rearranged to yield:

$$\dot{m}_a \frac{\sqrt{\theta}}{\delta} = YKA \frac{\sqrt{(2g_c)}}{\sqrt{\mathcal{R}}} \frac{P_{std}}{\sqrt{T_{std}}} \frac{\sqrt{P_1(P_1 - P_N)}}{P_1} \tag{6.4}$$

Final simplification of equation (6.4) yields:

$$\dot{m}_a \frac{\sqrt{\theta}}{\delta} = C\sqrt{\frac{(P_1 - P_N)}{P_1}} \tag{6.5}$$

where C is a constant equalling 15.0706 for the Boeing gas turbine's compressor.

Having established the selected compressor surge indicators (or those variables needed to calculate them) for both plant and model, the next procedure required was to have both sets of inputs read and entered into the expert systems's fact list.

The real time simulator inputs were directly entered into the expert system's fact list through file manipulation once the model values for compressor speed, flow rate and compressor outlet pressure were computed. The measured plant inputs, for baseline development purposes, were input manually into a text file, and read by the main program after the above mentioned model surge indicator values had been computed. The repetitive-cycle sequence of events then went as follows:

1. main simulator code computes model compressor speed, air mass flow rate and oulet pressure from existing plant initial conditions and plant inputs (load and fuel flow rate),
2. main simulator code reads measured plant surge inputs of compressor speed,inlet, outlet and nozzle pressures from text file (plant.txt),
3. all seven indicators (model's and plant's) are sent to a data file (turb.dat),
4. the embedded CLIPS subroutine (surge.clp) then reads the data file and enters the input into its fact list,
5. the two sets of indicators are manipulated, compared/analyzed and
6. finally, diagnosis displayed and control action taken.

In the present system a timed, analog-to-digital board was assumed to be used in conjunction with the input output capabilities of CLIPS for the purpose of entering the actual plant data directly into the CLIPS subroutine's fact list. The main simulator code

(rltms.c) can be referred to in Appendix C. The main simulator code (rltms.c) is similar to the C version of the real time simulator program (rts.c) contained in Appendix B, except for the required CLIPS interface commands and the additional commands to handle the reading and transfer of the measured plant data.

Notice that the above control procedure relies on an ability of the expert system to detect significant deviations between the model and actual data. Consequently, this implementation forces the programmer to decide on allowable operating windows or tolerances for the three health indicator values used. At this point it becomes necessary to develop an understanding of the differences in purpose and development between an engine health monitoring system and an expert system for casualty control.

In developing an engine health monitoring system, one of the first requirements is to develop what are known as an operating baselines. Operating baselines are a product of countless data recording, taken under various operating conditions. The data is then plotted to recognize the degree of scatter for the various parameters, and the collective curves are fitted to form the baselines. With the operating baselines used as references, the engine health monitoring system develops trend plots. What is important, however, and distinquishes a health monitoring system from a casualty control system, is that it is not essential that trend plots are developed with zero deviation from the baseline for each parameter, but rather the percentage deviation change from the baseline with time [Ref. 18: p.3]. The reason percentage deviation with time, and not zero deviation from the baseline is important is that the main function of an engine health monitoring system is trend assessment. In an expert system for casualty control, the real time simulator takes the place of the operating baselines. While operating baselines provide a general reference for various parameters, their use is restricted to steady state (stable) conditions [Ref. 18: p.5]. A real time simulator, however, provides a specific set of data which decribes the exact, dynamic state of the engine at a particular instant in time. It can be

used under steady state or dynamic conditions. Unlike the health monitoring system, response time is critical with a casualty control system, so deviations from an operating parameters provided by the real time simulator need to be carefully and quickly scrutinized to determine if immediate control action is required. The problem then becomes one of determining acceptable deviations from simulated operating points before control action is mandated.

To accurately define acceptable tolerances, a number of factors have to be considered. Some of these factors include: accuracy of the real time simulator (model), accuracy of aerothermal and mechanical sensors, and effectiveness of data filters. In developing the present expert casualty control system for surge control, a large amount of effort was not spent on accurately determining allowable deviation., This was largely due to the fact that certain required equipment (such as sensor data filters) were not installed. and that the modeling methods used seemed to agree with uncertainty calculations on engine parameters calculated during numerous gas turbine laboratory experiments conducted at the facility. In determining the individual allowable deviations. each parameter's operating range was computed, and then multiplied by 0.03. A three percent value was used because greater differences is operating conditions would be considered extreme for gas turbine engines. Compressor speed tolerance determined was (plus or minus) 350 rpm, pressure ratio tolerance was 0.035, and air mass flow rate's window was found to be 0.06 lbm/sec.

Having established the tolerances for the simulated (model) surge indicators, the next step was to apply these deviations to the model's indicators to acheive allowable tolerance windows. In other words, deviations were both added and subtracted to the model's pressure ratio, speed, and flow rate to produce a high model rpm value, a low model rpm value, a high model pressure ratio value, a low model pressure ratio value, a high model flow rate value, and a low model flow rate value.

Having applied the tolerance windows, thereby establishing allowable parameter limits, the next step was to compare the actual plant indicators to their model counterpart indicators to determine if the measured values fell within the established operating windows. In accomplishing this the comparison process was broken down into three broad cases; plant compressor speed being within model speed limits, plant compressor speed being greater than model limits and compressor speed being less than model (simulator) limits. In all cases the same surge onset indicator was looked for, that being, if plant-model indicator deviation was determined, did the plant's compressor characteristics (speed, pressure ratio, and air mass flow rate) position the compressor operating point closer or further from the surge line than did the model's characteristic indicators.

Before describing in detail the logic surrounding these three broad cases some background on the Boeing 502-2E compressor map, its associated surge line and the applied surge control line needs to be introduced. For the present study, efforts were made to locate an original compressor map for the Boeing model 502-E gas turbine engine's compressor without success. Owing to the fact that the engine is over thirty years old, the historical archives department at Boeing was contacted, but an original compressor map could not be found. To circumvent this problem, actual pressure ratio and air mass flow rate data was plotted at various steady state compressor speeds under various loadings (see Figure 11).

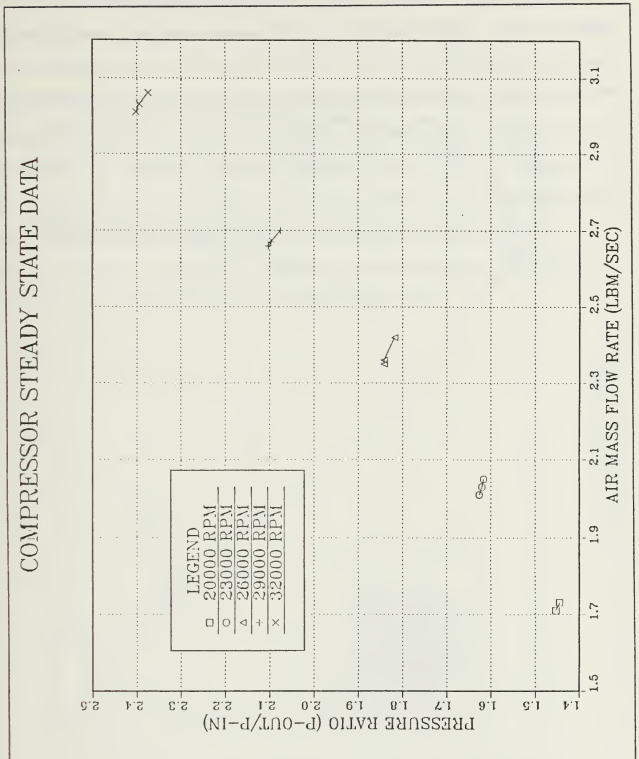

Figure 11. Steady State Compressor Data

By combining Figure 11 with some fabricated data points (based on general compressor mapping criteria) a compressor map for the Boeing engine was contructed. Figure 12 shows this construction.

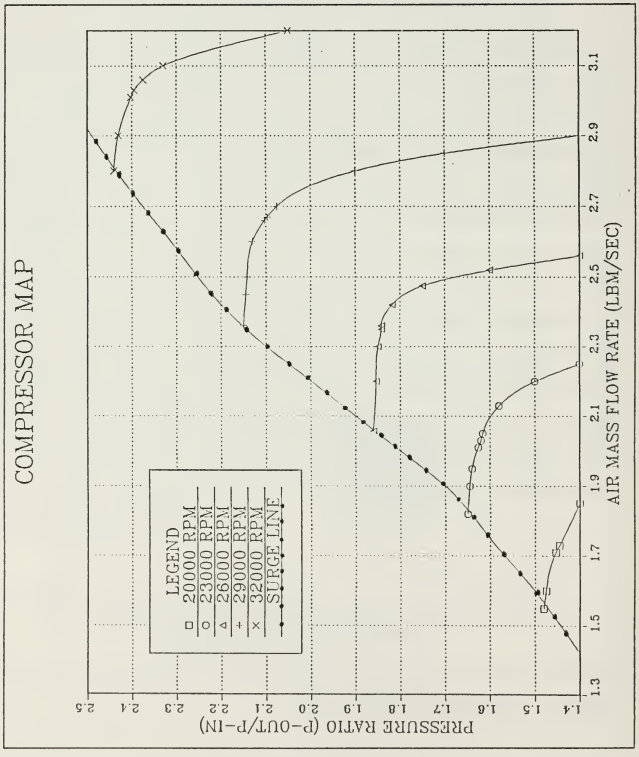

Figure 12. Constructed Boeing Compressor Map

Figure 12 shows a hypothetical surge line on the NPS gas turbine compressor. The surge line being the line that connects the surge points at various compressor speeds. Generally, the surge point is located at the compressor characteristic curve's peak (as implied in Chapter Three) where the slope changes. Here, however, allowances have been made for the centrifugal compressor stability. Figure 13 shows the compressor map with the surge control line added. The surge control line in the baseline expert surge control system was a fixed straight line which ran as parallel as possible to the compressor surge line. The distance between the surge control line and the surge line determined the amount of surge protection offered by the system.

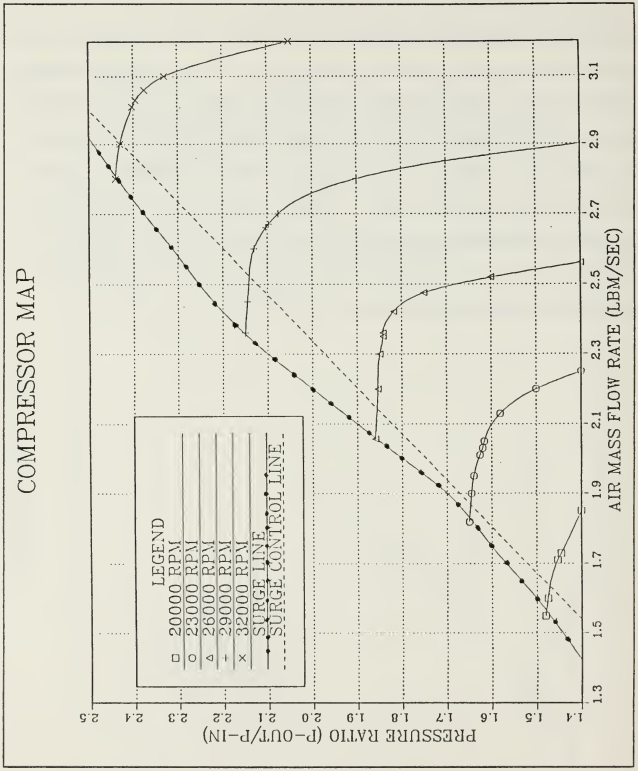

Figure 13. Compressor Map with Surge Control Line

Having defined all the applicable surge related terms and having been introduced to the NPS Boeing compressor map used in this study, the three previously mentioned indicator comparison cases can now be examined. The following flow chart of the embedded CLIPS subroutine (surge.clp) visually explains the logic and reasoning used for the three cases analyzed, and will be of assistance in understanding the embedded expert system program. The first two pages of the flow chart apply to the equal speed situation, the next two pages apply to the case of compressor speed being greater than model limits and the last flow chart is concerned with compressor speed being lower than model limits.

Note 1. * respresent simulator or model terms

Note 2. Data file "Turb.dat" contains the following data in the order listed.

Note 3. K_1 represents the slope of the surge control line

Turb.dat

N_G

P_2

P_1

P_N

N_G^*

m_a^*

P_2^*

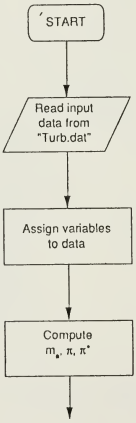

Figure 14. Flow Chart pg.1

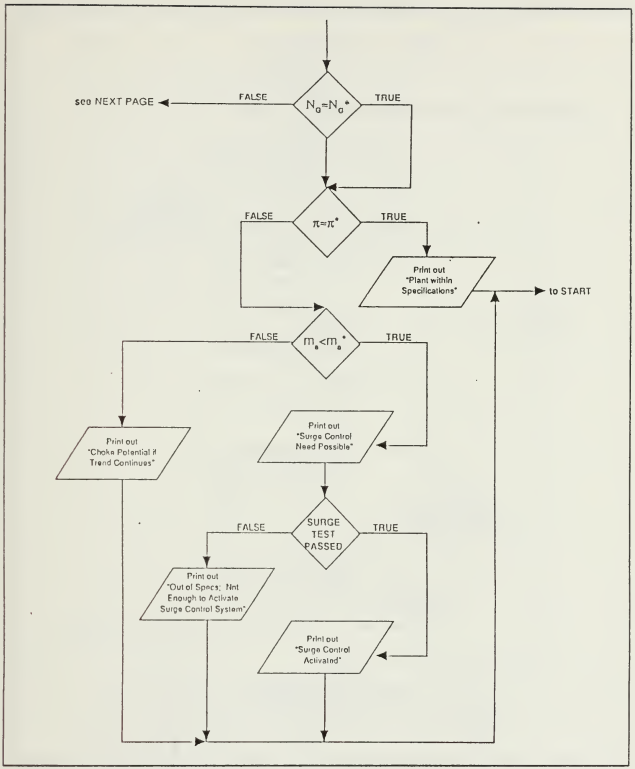

Figure 15. Flow Chart pg.2

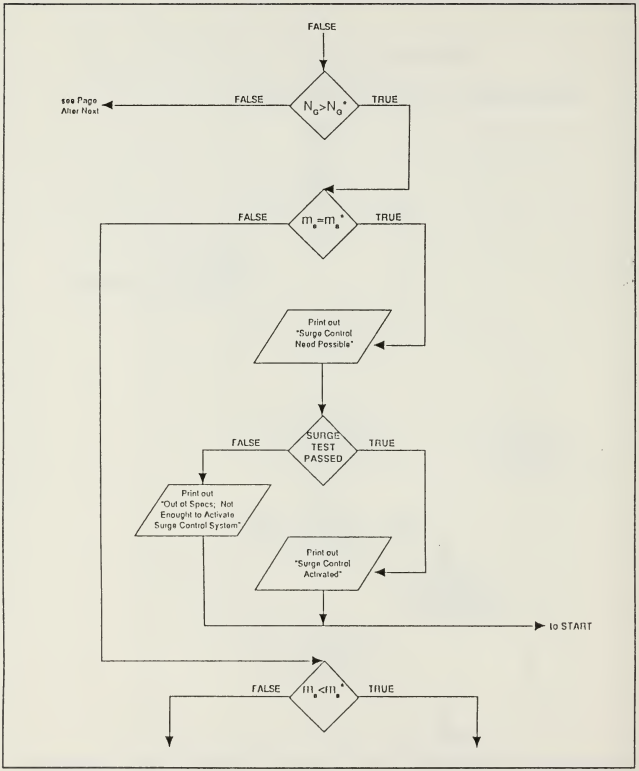

Figure 16. Flow Chart pg.3

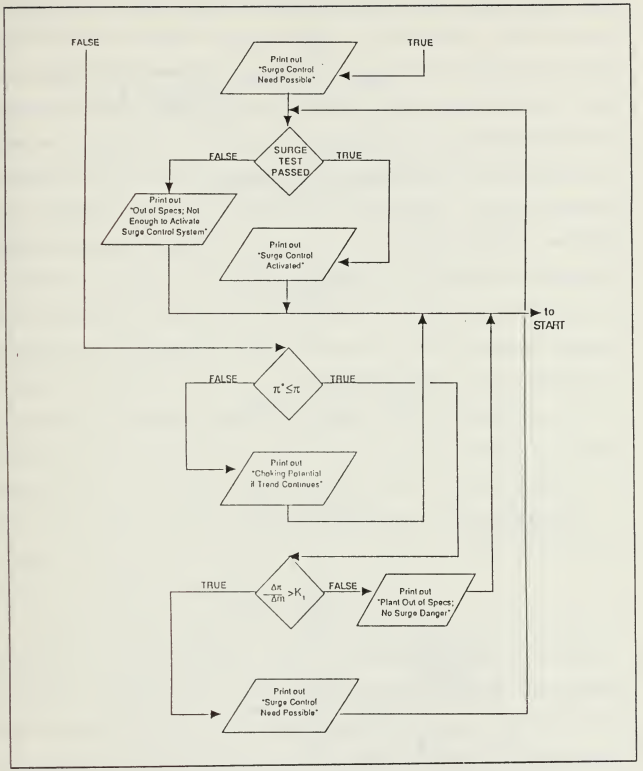

Figure 17. Flow Chart pg.4

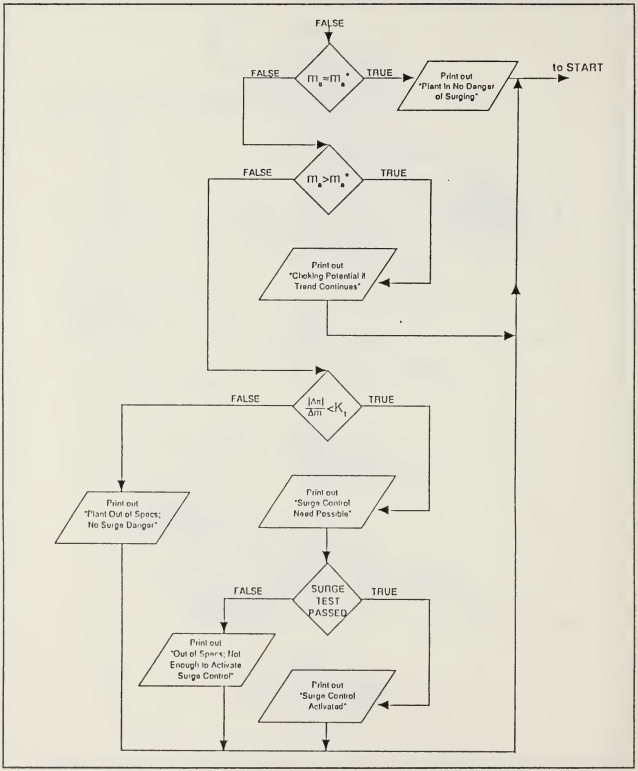

Figure 18. Flow Chart pg.5

The case of plant speed being within model speed limits will be examined first. This general case can be further broken down into three subcategories: plant speed and pressure ratio within model limits ("equal"), plant speed equal to model speed with high measured air mass flow rate, and plant speed equal to model speed with low air mass flow rate. Taking the first subcategory, if compressor speed and pressure ratio are determined to be within model limits, then the flow rate must also be within limits, therefore, a display will result stating "plant within specifications." If pressure ratios are different at the same speed, then air mass flow rates will have to be different, and either actual flow rate is higher or lower than model limits. So, if plant air mass flow rate is lower than the model's low flow rate limit, compressor choking could occur if flow rate continued to drop. Accordingly, if these characteristics exhibitted themselves a message stating, "choke potential if trend continues" would be displayed. On the other hand if the expert system code determined that speeds were equal but compressor mass air flow rate was lower than model limits, the plant would be operating closer to the surge line than the model, and the message "surge control need possible" would be displayed. The plant characteristics would then be utilized in the previosly discussed surge test to determine if the compressor is operating beyond the surge control line. If it is, "surge control activated" will be displayed.

The general case of actual compressor speed falling outside model speed limits is a more complicated scenario; however, similar reasoning was used to determine appropriate outputs, only now, instead of examining two variable combinations, three variables must be analyzed. Again, compressor speed is used to distinguish between cases. This time subcategories are broken into actual compressor speed being lower than model speed limits, and measured speed being higher than model speed limits. With actual compressor speed being higher than model speed limits the next variable examined is flow rate. If the compressor's measured flow rate is within limits, this would imply that

the actual pressure ratio would be higher than the model's (while operating in the negative slope region of the compressor map), and that the operating point would be closer to the surge control line. This would generate the message "surge control need possible," and the measured plant parameters would be used to check if the surge control line was crossed. If the air mass flow rate was not equal, while the plant compressor speed was higher than the model's, the flow rate cases of higher and lower-than model flow rate would have to be examined seperately. Higher measured compressor speed and lower measured air mass flow rate would again have an operating point closer to the surge control line then the model's operating point, and the warning "surge need possible" would flash on the screen. Additionally, with the warning comes the checking of the actual operating point to see if the compressor is in surge danger by determining if the control line has been crossed. However, if the measured air mass flow rate was greater than the model's, while the actual compressor speed remains greater, the outcome is not as straight-forward as previous cases. In this case, the subcategory would be further broken down into measured pressure ratio being greater than model limits, and actual pressure ratio being less than or equal to model pressure ratio limits. The proper diagnosis will be determined by the magnitudes of the differences between measured and actual pressure ratios and air mass flow rates. If, for the case of high measured pressure ratio, the pressure ratio difference (measured minus model) divided by the flow rate difference is greater than the slope of the surge control line, then the actual compressor operating point is closer to the control line. Being closer to the surge line, the expert system would initiate the standard "surge control need possible" warning, and the actual operating point would be checked to determine if surge control was indeed required. However, if while the measured speed and flow rate were greater, the actual pressure ratio was less than or equal to the model's pressure ratio the compressor would not be

in surge danger, but could possibly be in a compressor choking situation, and an appropriate message would be displayed.

Lastly, the general case of measured compressor speed being less than model speed needed to be investigated. As with the case of measured compressor speed being greater than model limits, three variables need to be scrutinized. The same general principles and reasoning that were used in the later case (measured speed greater) were again applied to arrive at proper compressor diagnosis and determine if compressor control action was required.

The "surge test", which was referred to in the above case descriptions, is simply a check to see if the plant's operating point has crossed the surge control line. The concept behind the surge test is straightforward, and in some respects resembles the overview of the simplified universal anti-surge control system given in [Ref. 19: pp.84-85]. The mechanics of the surge test descibed below utilizes the equation of the surge control line along with the measured plant parameters. Writing the surge control line in stardard line equation form (Y = m X + b) yields:

$$\frac{P_2}{P_1} = 0.7534\dot{m}_a + 0.2398 \tag{6.6}$$

Subtracting one from both sides and multiplying through by compressor inlet pressure equation (6.6) becomes:

$$P_2 - P_1 = 0.7534\dot{m}_a P_1 - 0.7602P_1 \tag{6.7}$$

Substituting equation (6.5) for mass flow rate, and organizing terms yields:

$$P_2 - P_1 = P_1(11.354\sqrt{\frac{(P_1 - P_v)}{P_1}} - 0.7602) \tag{6.8}$$

If, after measured plant values for the inlet, outlet, and nozzle pressures have been substituted into equation (6.8) the left hand side of the equation is determined to be greater than the left hand side than the compressor's operating point has indeed crossed the surge control line, and surge control action is required. As is the case with the universal anti-surge control system decribed in [Ref. 19: p.83], the baseline expert surge control system ultimately provides surge protection by not allowing the compressor to operate to the right of the surge control line, thus establishing a surge margin of safety. It has been assumed, due to the relative close proximity of the control line to the surge line, that the real time simulator operating points would not cross the surge control line, and yet come close enough to it, in order to ensure maximum compressor operating efficiency. Under current system configuration, however, if the simulator happened to track through the surge control line, the surge control system would not be activated as long as the plant surge indicators remain within model limits.

The most common type of surge protection used on compressors (not the LM-2500), is through the use of blow-off or recirculation valves [Ref. 20: p.40]. These systems protect against surge by measuring existing compressor conditions and checking that pre-established limits are not exceeded. If these limits are compromised the blow-off (or recirculation) valve is automatically openned, and operating limits are restored to exceptable levels. Two typical types of blow-off surge protection systems from [Ref. 21: p.184] are shown in Figure 19 and Figure 20.

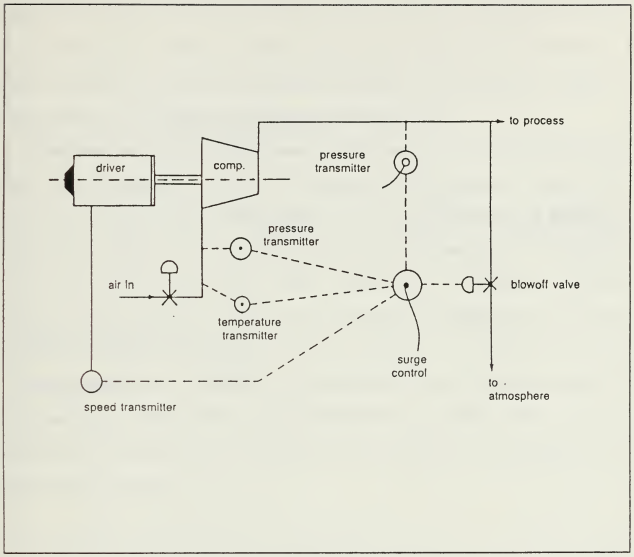

Figure 19. Pressure Oriented Anti-Surge Control System

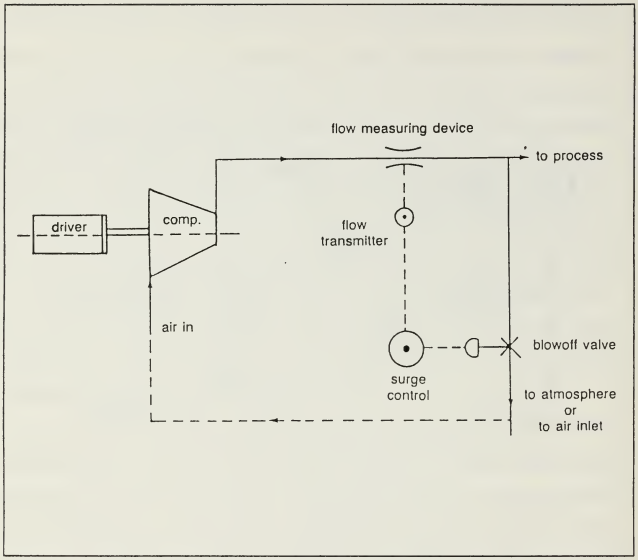

Figure 20. Flow-Oriented Anti-Surge Control System

The principle behind the blow-off (and recirculation) valve is simply that by opening the valve, which is located just downstream of the compressor, positive air flow is maintained through the compressor because the negative pressure gradient is controlled.

An anti-surge control device which could be easily incorporated into the baseline expert control system is a version of the blow-off valve principle. The difference between the ordinary blow-off valve surge controller and the baseline expert sytem is that the latter system, through the use of the real time simulator, is able to provide a continuous

display of plant-model deviation. Additionally, it may utilize simulator data as feedback information to the plant controller and blow-off valve to vector the compressor's measured operating point to within agreement of the model's location on the compressor map. And lastly, the expert surge controller, by the nature of expert systems, could be easily expanded to provide additional engine casualty protection.

In summary, the expert system performs the following tasks in the sequence listed: Plant inputs are received by the plant and real time simulator, the plant responds and the real time simulator predicts the nominal compressor operating point, the measured surge indicator data is read by the embedded expert analysis routine where it is massaged and compared with surge indicator limits, and finally, the comparison analysis compressor status is displayed. If the actual operating point was determined to be closer to the surge control line than the model called for, a surge test is conducted to determine if the control line has been crossed. If the surge control line has indeed been compromised, the system would provide simulator feedback to the plant controller and the blow-off valve, which would work in conjuction with each other to reestablish agreement between plant and model.

VII. RESULTS

Discussion of results will be broken down into three areas: real time simulator testing and results, CLIPS subroutine testing and results, and integrated system testing and results.

As discussed in Chapter Five, the testing of the real time simulator was accomplished for accuracy and timing under basically four different conditions. Initially the code was tested for accuracy under constant gas generator (or compressor) speeds and agreement between predicted model values and measured steady state values were quite good. Then using the same simulator structure the programming code was changed from DSL to Fortran 77, and a timing study was conducted on a 32 bit processor. Timing test results revealed that 0.13 seconds of computing time was required for each second of simulated time. Because a CLIPS to Fortran interface package would have been required to make the Fortran version of the simulator interactive with the CLIPS language subroutine, the obvious alternative was to change the Fortran version of the simulator to C, as the interface functions between C and CLIPS exist in the CLIPS code.

Testing of the real time simulator in C, under transient gas generator conditions, with load and fuel flow equations included in the code to estimate system inputs yielded close agreement with actual measured data after the program's time step was modified from 0.001 to 0.000368. Figure 21 illustrates how the simulator output tracked right along measured data taken at the various compressor speeds shown. On a 16 bit processor, using an optimizing, turbo C compiler, a simulated ten second run took 47 seconds of computation time. Or, for each second of simulated time 4.7 seconds of computation time was required.

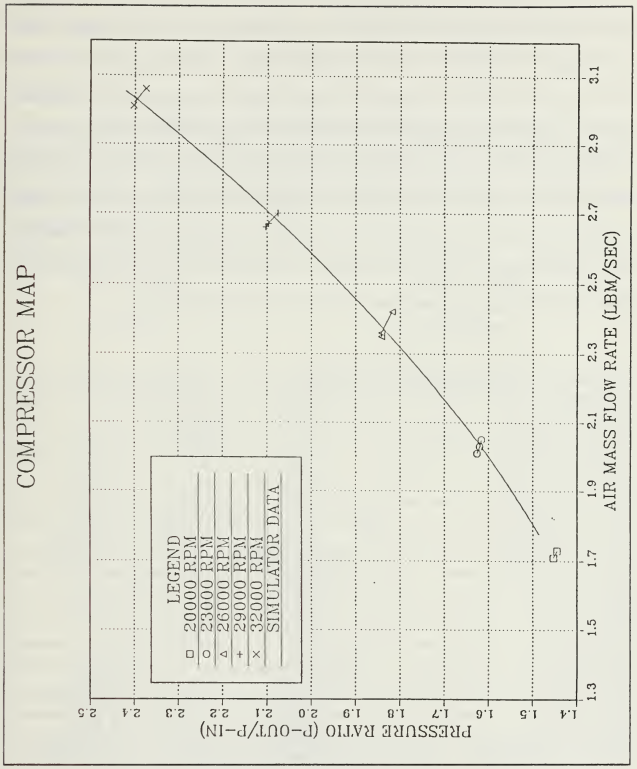

Figure 21. Simulator Test Results

The next portion of the baseline system which had to be tested was the CLIPS subroutine code (surge.clp) which would be embedded into the main C simulator code. This testing was done in the CLIPS executable mode, in order that each rule of the subroutine could be tested seperately to ensure actual output agreed with anticipated output. In order to accomplish this twelve different data sets were constructed, and individually read into the programs fact list during code rule testing. Since it was known ahead of time how many rules should be triggered or fired, and also what the output should be, based on the particular data set entered, it was relatively simple to determine if the program was functioning as anticipated. The twelve sets of data combinations of compressor speed, pressure ratio, and air flow rate are listed in Table 1.

Table 1. CLIPS SUBROUTINE TESTING

Nominal	rpm =	pr =	
Choke Potential 1.	rpm =	pr <	
Choke Potential 2.	rpm >	pr <	air >
Choke Potential 3.	rpm <		air >
Out, No Danger 1.	rpm <	pr <	air >
Out, No Danger 2.	SLOPE	TEST 1	LOW
Out, No Danger 3.	SLOPE	TEST 2	HIGH
Out, No Danger 4.	SURGE	TEST	FAILED
Slope Test 1.	rpm >	pr > or =	air >
Slope Test 2.	rpm <		air <
Surge Test 1.	rpm =	pr >	
Surge Test 2.	rpm >		air =
Surge Test 3.	rpm >		air <
Surge Test 4.	SLOPE	TEST 1	HIGH
Surge Test 5.	SLOPE	TEST 2	LOW
Surge Control Req'd	SURGE	TEST	PASSED.

It should be noted, however, that input data was read by the subroutine as compressor speed, and inlet, outlet, and nozzle pressures. For illustration purposes, the computation

which occured in the initial part of the subroutine, (flow rate and pressure ratio) were performed before this test was conducted. Individual response tests conducted were successful in that all anticipated responses agreed with those anticipated from the data sets input.

The last testing conducted was the integrated testing phase, during which all components of the baseline expert diagnosis system were tested together, during a dynamic response simulation. The framework for the dynamic integrated expert system testing is shown in Figure 22.

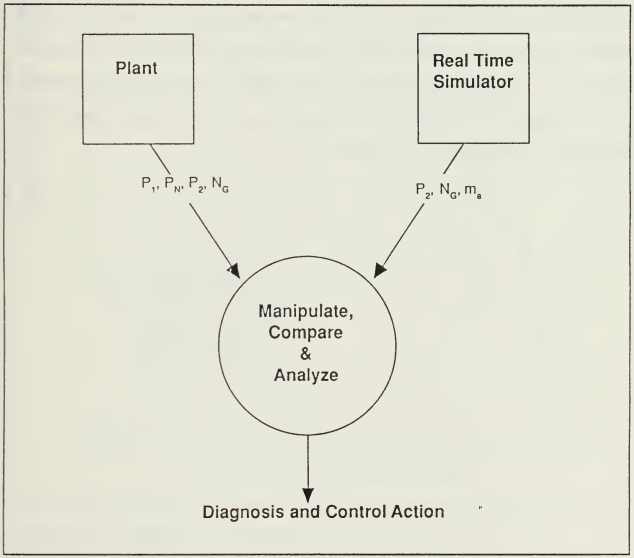

Figure 22. Dynamic Response Testing Framework

In achieving this goal, an engine history test file was fabricated containing twenty sets of data that would be independently read by the system during normal operation at half second intervals. (Recall that the system receives four measured data values for each run loop.) The fabricated data was constructed in such a way that different system diagnoses could be dynamically tested, while keeping the inputs realistic. The integrated testing run was to simulate a change in gas generator speed, created by a changing load, over a time span of ten seconds, with outputs provided by the simulator every half second. The data sampling rate of a half second was determined by actual system load and fuel flowrate input measurements taken from a strip chart recording, which were then molded into input equations and utilized in the simulator code, as previously discussed in Chapter Five. A graphical representation of the dynamic, integrated testing results is contained in Figure 23, with the output display response corresponding to each of the twenty measured operating points given below.

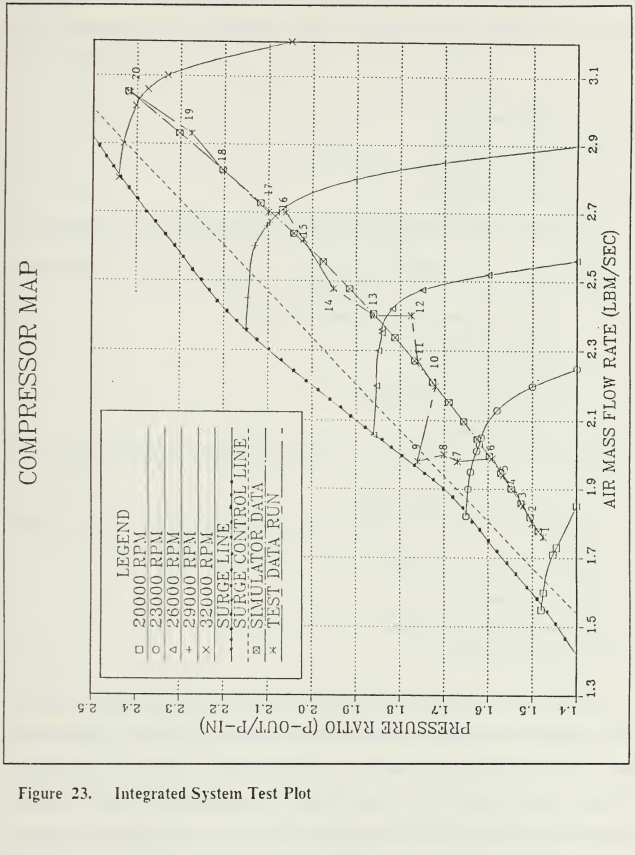

Figure 23. Integrated System Test Plot

Response one below corresponds to the measured operating point at the bottom left of Figure 23, and similarly response number twenty at the top right of the plot.

1. plant within specification
2. plant within specification
3. plant within specification
4. plant within specification
5. plant within specification
6. plant within specification
7. surge control need possible...out of specs; not enough to activate surge control system
8. surge control need possible...out of specs; not enough to activate surge control system
9. surge control need possible...surge control activated
10. plant within specification
11. plant within specification
12. choke potential if trend continues
13. plant within specification
14. surge control need possible...out of specs; not enough to activate surge control system
15. plant out of specs, no surge danger
16. surge control need possible...out of specs; not enough to activate surge control system
17. plant in no danger of surging
18. plant within specification
19. plant in no danger of surging
20. plant within specification

As can be seen from Figure 23, the baseline expert system output agrees with what one would anticipate by comparing the plant and model operating points. Additionally, during the integrated system testing phase a timing study was conducted to determine the time required to cycle through one half second sampling/computing loop. This loop contained simulator data computation, model and measured surge indicator reading, indicator comparison and analysis, and finally diagnosis output. It was found that to run

an executable (.exe) file consisting of twenty interations or loops, on a 16 bit processor using an optimizing Turbo C compiler, it took approximately 110 seconds; or 5.5 seconds per half second system loop.

VIII. CONCLUSIONS AND RECOMMENDATIONS

Before examining and discussing specific sytem testing results, some general observations can be made regarding the baseline expert surge control system development:

1. A very limited amount of literature exists in the marine gas turbine expert system arena,
2. No literature or research work could be found in the more specific combined topic of compressor surge and expert systems,
3. The majority of expert system gas turbine research is being conducted in the steady state, industrial gas turbine environment,
4. The limited amount of reseach taking place in the area of expert systems and marine gas turbines is primarily directed towards engine monitoring vice engine control or engine diagnosis and control, and
5. The phenomenon of compressor surge is still not completely understood, even by experts in the turbomachinery field.

Focussing attention closer to the actual baseline system development itself, the following observations were made while constructing the individual system components:

1. The sequential linearization technique used in the development of the system's real time simulator is an effective means of determining the gas turbines state, from which other parameters can be calculated,
2. The executable version of CLIPS was an efficient and effective tool, which greatly simplified expert system code development,
3. CLIPS integration capabilities are noteworthy, thus eliminating embedded application difficulties, and
4. CLIPS was very portable and worked flawlessly with a Boreland turbo C compiler.

Specific conclusions focussing on individual component and integrated system testing results revolve around two issues, accuracy and timing. Overall, the integrated system achieved the desired output, however, two areas of concern arise. The first being that, in order to be an effective system to prevent compressor surging, the system's loop (or one complete cycle) execution time has to be in the order of less than one half a second. Currently it is taking the system 5.5 seconds of execution time for every half

second of simulated time. Clearly, under current configuration this is an unworkable, non- real time system. Considerable decreases in execution time possibly may be achieved by moving the baseline system to a 32 bit processor vice the slower 16 bit. Lastly, in regards to system accuracy, but also related to timing concern, is the scaling factor problem that was encompassed when the simulator code was tested under varying gas generator speeds. Although simulator data agreed with measured data after the simulator code's iteration step size was changed, further simulator code testing needs to be accomplished under various gas generator operating conditions.

With these concerns in mind the following recommendations concerning continued interest in the development of an expert casualty control system for marine gas turbines are proposed:

1. Conduct baseline expert surge control system testing on a 32 bit processor,
2. Conduct further simulator code testing under various operating conditions, specifically varying gas generator speed,
3. Develop specifics on the surge control strategy, enabling simulator data to be used in the control process by the plant controller and surge protection blow-off valve,
4. Install the baseline expert surge control system on the NPS Boeing gas turbine for online system testing,
5. Futher investigate specifics for possible incorporation of an expert surge control system on the LM-2500 engine, and
6. Expand the baseline expert system's knowledge base to facilitate additional gas turbine engine casualties.

Lastly, in a recent artical published by the American Society of Naval Engineers, the author beckoned engineers, naval architects,and managers to [Ref. 22: p.39]:

1. "Examine the field of expert systems,
2. Get people involved,
3. Pick the most promising problems, document them, and distill the expert knowledge required to solve them
4. Make this technology availiable,
5. Build, test, and use the systems to save time and money, now and in the future."

If for nothing else, this thesis has answered his call.

APPENDIX A. FORTRAN REAL TIME SIMULATOR CODE

```
      REAL NG,NS,NGN,MF,MFI
C   establish the initial conditions
      NG  = 349000.0
      NS  = 570.0
      E   = 193.5
      MFI = 193.5
      QLI = 405.0
C   establish the initial offset values-note that these are not the
C   not the perturbations. at any time step they are equal to the
C   previous offset from the initial condition plus the pertibations
      DNG=0.0
      DNS=0.0
      DE=0.0
C   set the normalizing values
      NGN=26000
      NSN=1750
c
c   the 100 do loop is the simulation loop
      DO 100 n=1,100000
C
      A11=-1.0*EXP(-0.6993*(NS/NSN)+5.583*(NG/NGN)-3.2433)
c
      A12=-1.0*EXP(-2.8415*(NS/NSN)+7.9978*(NG/NGN)-4.4662)
c
      A13=EXP(-0.4579*(NS/NSN)+1.189*(NG/NGN)+6.8305)
c
      A21=0.1531*(NS/NSN)*(NS/NSN)-0.9535*(NG/NGN)*(NS/NSN)
     $       +1.5745*(NG/NGN)*(NG/NGN)+0.5181*(NS/NSN)
     $            -1.6232*(NG/NGN)+0.46015
c
      A22=.056875*(NS/NSN)-1.3166*(NG/NGN)+.3962
c
      A23=EXP(0.92011*(NS/NSN)-4.2549*(NG/NGN)+6.2290)
c
      A31=0.0
      A32=0.0
      A33=-10.0
      B11=0.0
      B12=0.0
      B21=0.0
      B22=-14.17
      B31=10.0
      B32=0.0
c
c   compute plant input offset
      DMF=MF-MFI
      DQL=QL-QLI
      DNGDOT=A11*DNG+A12*DNS+A13*DE
```

```
      DNSDOT=A21*DNG+A22*DNS+A23*DE+B22*DQL
      DEDOT=A31*DNG+A32*DNS+A33*DE+B31*DMF
c
c   compute integral of the rates
c   use rectangular integration methos with a time step of 0.001 sec
      DNG=DNG+DNGDOT*0.001
      DNS=DNS+DNSDOT*0.001
      DE=DE+DEDOT*0.001
c
c   compute the state variables
      NG=NG+DNG
      NS=NS+DNS
      E=E+DE
c
  100 CONTINUE
c
      STOP
      END
```

APPENDIX B. C VERSION OF REAL TIME SIMULATOR

This program computes compressor speed, compressor outlet pressure and compressor air mass flow rate.

```
# include <stdio.h>
# include <math.h>
main ()
{
 /* establish starting point*/
 int n,ct,t;
 double ng,ngn,ns,nsn,e,mfi,qli,mf,ql,z;
 double dng,dns,de,a11,a12,a13;
 double a,b,c,a21,a31,a32,a33,a22,a23;
 double b22,b31;
 double dmf,dql,dngdot,dnsdot,dedot;
 double p2n,p2nn,mn,p2,mdot;
 ng=20220.0;
 ns=1510.0;
 e=79.0;
 mfi=79.0;
 qli=65.0;
 z=0.0;
 t=0;
 /* initial mf and ql values */
 mf=79.0;
 ql=65.0;
 /* initial offset values */
 start:
 ct=0;
 dng=0.0;
 dns=0.0;
 de=0.0;
 /* normalizing values */
 ngn=26000.0;
 nsn=1750.0;
 /* simulation loop */
 n=100;
 while (n)
 {
  ct=ct+1;
  a11=-1.0*exp((-0.6993*(ns/nsn))+(5.583*(ng/ngn))-3.2433);
  a12=-1.0*exp((-2.8415*(ns/nsn))+(7.9978*(ng/ngn))-4.4662);
  a13=exp((-0.4579*(ns/nsn))+(1.189*(ng/ngn))+6.8305);
  a=(.1531*(ns/nsn)*(ns/nsn))-(.9535*(ng/ngn)*(ng/ngn));
  b=(1.5745*(ng/ngn)*(ng/ngn))+(0.5181*(ns/nsn));
  c=(-1.6232*(ng/ngn))+0.46015;
  a21=a+b+c;
  a22=(0.056875*(ns/nsn))-(1.3166*(ng/ngn))+0.3962;
```

```
a23=exp((0.92011*(ns/nsn))-(4.2549*(ng/ngn))+6.2290);
a31=0.0;
a32=0.0;
a33=-10.0;
b22=-14.17;
b31=10.0;
/* compute plant input offset */
dmf=mf-mfi;
dq1=q1-q1i;
dngdot=(a11*dng)+(a12*dns)+(a13*de);
dnsdot=(a21*dng)+(a22*dns)+(a23*de)+(b22*dq1);
dedot=(a31*dng)+(a32*dns)+(a33*de)+(b31*dmf);
/* compute integral of the rates */
/* use rectangular integration method */
dng=dng+(dngdot*.000368);
dns=dns+(dnsdot*.000368);
de=de+(dedot*.000368);
/* compute the state variables */
ng=ng+dng;
ns=ns+dns;
e=e+de;
/* compute p2 and mdot from state variables */
p2nn=0.504-(1.44*ng/26000.0)+(0.0134*ns/1500.0);
p2n=p2nn+(1.81*(ng/26000.0)*(ng/26000.0));
mn=-0.176+(1.16*ng/26000.0)+(0.022*p2n);
p2=p2n*14.0;
mdot=mn*2.36;
z=(ct/100.0)+t;
q1=(26.0*z)+65.0;
mf=(8.7*z)+79.0;
n=n-1;
}
t=t+1;
printf ("%d n",t);
printf ("%6.3f n",ng);
printf ("%2.4f n",mdot);
printf ("%3.4f n",p2);
/* start over */
mfi=mf;
e=mf;
q1i=q1;
if (t<10)
 {
  goto start;
 }
}
```

APPENDIX C. CLIPS INTERACTIVE SIMULATOR C CODE

This program is similar to the C simulator code in Appendix B only the CLIPS interaction commands have been added.

```
# include <stdio.h>
# include <math.h>
# include "clips.h"
main ()
{
 /* establish starting point*/
 char turb[60], plant[60], surge[60];
 int n,ct;
 double ng,ngn,ns,nsn,e,mfi,qli,mf,ql,z,t;
 double dng,dns,de,a11,a12,a13;
 double a,b,c,a21,a31,a32,a33,a22,a23;
 double b22,b31;
 double dmf,dql,dngdot,dnsdot,dedot;
 double p2n,p2nn,mn,p2,mdot;
 float rpm,pdis,pnoz,pin;
 FILE *input_file;
 FILE *output_file;
 sprintf(turb,"%s","turb.dat");
 sprintf(plant,"%s","plant.txt");
 sprintf(surge,"%s","surge.clp");
 init_clips();
 load_rules(surge);
 ng=20220.0;
 ns=1510.0;
 e=79.0;
 mfi=79.0;
 qli=65.0;
 z=0.0;
 t=0.0;
 /* initial mf and ql values */
 mf=79.0;
 ql=65.0;
 input_file = fopen(plant,"r");
 /* initial offset values */
 start:
 ct=0;
 dng=0.0;
 dns=0.0;
 de=0.0;
 /* normalizing values */
 ngn=26000.0;
 nsn=1750.0;
 /* simulation loop */
 n=100;
```

```
while (n)
{
 ct=ct+1;
 a11=-1.0*exp((-0.6993*(ns/nsn))+(5.583*(ng/ngn))-3.2433);
 a12=-1.0*exp((-2.8415*(ns/nsn))+(7.9978*(ng/ngn))-4.4662);
 a13=exp((-0.4579*(ns/nsn))+(1.189*(ng/ngn))+6.8305);
 a=(.1531*(ns/nsn)*(ns/nsn))-(.9535*(ng/ngn)*(ng/ngn));
 b=(1.5745*(ng/ngn)*(ng/ngn))+(0.5181*(ns/nsn));
 c=(-1.6232*(ng/ngn))+0.46015;
 a21=a+b+c;
 a22=(0.056875*(ns/nsn))-(1.3166*(ng/ngn))+0.3962;
 a23=exp((0.92011*(ns/nsn))-(4.2549*(ng/ngn))+6.2290);
 a31=0.0;
 a32=0.0;
 a33=-10.0;
 b22=-14.17;
 b31=10.0;
 /* compute plant input offset */
 dmf=mf-mfi;
 dql=ql-qli;
 dngdot=(a11*dng)+(a12*dns)+(a13*de);
 dnsdot=(a21*dng)+(a22*dns)+(a23*de)+(b22*dql);
 dedot=(a31*dng)+(a32*dns)+(a33*de)+(b31*dmf);
 /* compute integral of the rates */
 /* use rectangular integration method */
 dng=dng+(dngdot*.000368);
 dns=dns+(dnsdot*.000368);
 de=de+(dedot*.000368);
 /* compute the state variables */
 ng=ng+dng;
 ns=ns+dns;
 e=e+de;
 /* compute p2 and mdot from state variables */
 p2nn=0.504-(1.44*ng/26000.0)+(0.0134*ns/1500.0);
 p2n=p2nn+(1.81*(ng/26000.0)*(ng/26000.0));
 mn=-0.176+(1.16*ng/26000.0)+(0.022*p2n);
 p2=p2n*14.0;
 mdot=mn*2.36;
 z=(ct/200.0)+t;
 ql=(26.0*z)+65.0;
 mf=(8.7*z)+79.0;
 n=n-1;
}
t=t+.5;
output_file = fopen(turb,"w");
fscanf (input_file,"%f",&rpm);
fscanf (input_file,"%f",&pdis);
fscanf (input_file,"%f",&pin);
fscanf (input_file,"%f",&pnoz);
fprintf (output_file,"%6.4f n",rpm);
fprintf (output_file,"%6.4f n",pdis);
fprintf (output_file,"%6.4f n",pin);
fprintf (output_file,"%6.4f n",pnoz);
fprintf (output_file,"%6.4f n",ng);
fprintf (output_file,"%6.4f n",mdot);
fprintf (output_file,"%6.4f n",p2);
```

```
 fclose (output_file);
 reset_clips();
 assert("start-up");
 run(-1);
 /* start over */
 mfi=mf;
 e=mf;
 qli=ql;
 if (t<10.0)
  {
   goto start;
  }
 fclose (input_file);
}
usrfuncs()
 {}
```

APPENDIX D. EMBEDDED CLIPS SUBROUTINE CODE

This CLIPS program which will be embedded in the main C code. Based on inputs from plant and model it reached a diagnostic control decision.

```
(defrule open-file
   (start-up)
=>
   (open "turb.dat" data)
   (assert (read-file)))
(defrule read-turb-file
   ?read-file <- (read-file)
=>
   (retract ?read-file)
   (assert (rpm =(read data)))
   (assert (pdis =(read data)))
   (assert (pin =(read data)))
   (assert (pnoz =(read data)))
   (assert (rpm-m =(read data)))
   (assert (air-m =(read data)))
   (assert (pdis-m =(read data)))
   (assert (computation))
   (close))
(defrule computation
   ?computation <- (computation)
   ?rpms-m <- (rpm-m ?rpm-m)
   ?pnozs <- (pnoz ?pnoz)
   ?pdiss <- (pdis ?pdis)
   ?pins <- (pin ?pin)
   ?pdiss-m <- (pdis-m ?pdis-m)
   ?airs-m <- (air-m ?air-m)
=>
   (retract ?computation)
   (retract ?rpms-m)
   (retract ?pnozs)
   (retract ?pdiss)
   (retract ?pins)
   (retract ?pdiss-m)
   (retract ?airs-m)
   (assert (h-rpm-m =(+ ?rpm-m 350.0)))
   (assert (l-rpm-m =(- ?rpm-m 350.0)))
   (assert (pr =(/ ?pdis ?pin)))
   (bind ?pr-m (/ (+ ?pdis-m ?pin)?pin))
   (assert (h-pr-m =(+ ?pr-m .035)))
   (assert (l-pr-m =(- ?pr-m .035)))
   (assert (air =(* 15.0706 (sqrt (/ (- ?pin ?pnoz) ?pin)))))
   (assert (h-air-m =(+ ?air-m .06)))
   (assert (l-air-m =(- ?air-m .06)))
   (bind ?t-rhs (* (- (* (sqrt (/ (- ?pin ?pnoz) ?pin)) 11.354) .7602) ?pin))
```

```
   (bind ?t-lhs (- ?pdis ?pin))
   (assert (air-m ?air-m))
   (assert (pr-m ?pr-m))
   (assert (t-rhs ?t-rhs))
   (assert (t-lhs ?t-lhs)))
(defrule rpm-check
   (h-rpm-m ?h-rpm-m)
   (l-rpm-m ?l-rpm-m)
   (h-pr-m ?h-pr-m)
   (l-pr-m ?l-pr-m)
   (rpm ?rpm&:(>= ?h-rpm-m ?rpm)&:(>= ?rpm ?l-rpm-m))
   (pr ?pr&:(>= ?h-pr-m ?pr)&:(>= ?pr ?l-pr-m))
=>
   (fprintout t "plant within specification "crlf))
(defrule pr-low
   (h-rpm-m ?h-rpm-m)
   (l-rpm-m ?l-rpm-m)
   (l-pr-m ?l-pr-m)
   (h-air-m ?h-air-m)
   (rpm ?rpm&:(>= ?h-rpm-m ?rpm)&:(>= ?rpm ?l-rpm-m))
   (pr ?pr&:(> ?l-pr-m ?pr))
   (air ?air&:(> ?air ?h-air-m))
=>
   (fprintout t "choke potential if trend continues"crlf))
(defrule pr-high
   (h-rpm-m ?h-rpm-m)
   (l-rpm-m ?l-rpm-m)
   (h-pr-m ?h-pr-m)
   (l-air-m ?l-air-m)
   (rpm ?rpm&:(>= ?h-rpm-m ?rpm)&:(>= ?rpm ?l-rpm-m))
   (pr ?pr&:(> ?pr ?h-pr-m))
   (air ?air&:(< ?air ?l-air-m))
=>
   (fprintout t "surge control need possible"crlf)
   (assert (surge test)))
(defrule surge-test
   (surge test)
   (t-rhs ?t-rhs)
   (t-lhs ?t-lhs&:(> ?t-lhs ?t-rhs))
=>
   (fprintout t "surge control activated"crlf))
(defrule surge-test-fail
   (surge test)
   (t-rhs ?t-rhs)
   (t-lhs ?t-lhs&:(<= ?t-lhs ?t-rhs))
=>
   (fprintout t "out of specs;not enough to activate  controlsystem"crlf))
(defrule rpm-greater
   (h-rpm-m ?h-rpm-m)
   (h-air-m ?h-air-m)
   (l-air-m ?l-air-m)
   (rpm ?rpm&:(> ?rpm ?h-rpm-m))
   (air ?air&:(>= ?h-air-m ?air)&:(>= ?air ?l-air-m))
=>
   (fprintout t "surge control need possible"crlf)
```

```
   (assert(surge test)))
(defrule rpm-greater-low-air
   (h-rpm-m ?h-rpm-m)
   (l-air-m ?l-air-m)
   (h-air-m ?h-air-m)
   (rpm ?rpm&:(> ?rpm ?h-rpm-m))
   (air ?air&:(> ?l-air-m ?air))
=>
   (fprintout t "surge control need possible"crlf)
   (assert (surge test)))
(defrule rpm-greater-hi-air
   (h-rpm-m ?h-rpm-m)
   (h-air-m ?h-air-m)
   (h-pr-m ?h-pr-m)
   (air-m ?air-m)
   (pr-m ?pr-m)
   (rpm ?rpm&:(> ?rpm ?h-rpm-m))
   (air ?air&:(> ?air ?h-air-m))
   (pr ?pr&:(>= ?pr ?h-pr-m))
=>
   (bind ?pr-diff (- ?pr ?pr-m))
   (bind ?slopes (/ ?pr-diff (- ?air ?air-m)))
   (assert (slope-check ?slopes)))
(defrule slope-check-low
   (slope-check ?slopes&:(>= .7534 ?slopes))
=>
   (fprintout t "plant out of specs, no surge danger"crlf))
(defrule slope-check-high
   (slope-check ?slopes&:(< .7534 ?slopes))
=>
   (fprintout t "surge control need possible"crlf)
   (assert (surge test)))
(defrule rpm-greater-hi-air-low-pr
   (h-rpm-m ?h-rpm-m)
   (h-air-m ?h-air-m)
   (h-pr-m ?h-pr-m)
   (rpm ?rpm&:(> ?rpm ?h-rpm-m))
   (air ?air&:(> ?air ?h-air-m))
   (pr ?pr&:(>= ?h-pr-m ?pr))
=>
   (fprintout t "choking potential if trend continues"crlf))
(defrule rpm-less-air-equal
   (l-rpm-m ?l-rpm-m)
   (h-air-m ?h-air-m)
   (l-air-m ?l-air-m)
   (rpm ?rpm&:(> ?l-rpm-m ?rpm))
   (air ?air&:(>= ?h-air-m ?air)&:(>= ?air ?l-air-m))
=>
   (fprintout t "plant in no danger of surging"crlf))
(defrule rpm-less-hi-air
   (l-rpm-m ?l-rpm-m)
   (h-air-m ?h-air-m)
   (rpm ?rpm&:(> ?l-rpm-m ?rpm))
   (air ?air&:(> ?air ?h-air-m))
=>
```

```
   (fprintout t "chocking potential if trend continues"crlf))
(defrule rpm-less-low-air
   (l-air-m ?l-air-m)
   (l-rpm-m ?l-rpm-m)
   (pr-m ?pr-m)
   (pr ?pr)
   (air-m ?air-m)
   (rpm ?rpm&:(> ?l-rpm-m ?rpm))
   (air ?air&:(> ?l-air-m ?air))
=>
   (bind ?pr-dif (abs (- ?pr-m ?pr)))
   (bind ?slope (/ ?pr-dif (- ?air-m ?air)))
   (assert (slope ?slope)))
(defrule low-slope
   (slope ?slope&:(> .7534 ?slope))
=>
   (fprintout t "surge control need possible"crlf)
   (assert (surge test)))
(defrule high-slope
   (slope ?slope&:(<= .7534 ?slope))
=>
   (fprintout t "plant out of specs; no surge danger"crlf))
```

LIST OF REFERENCES

1. Malkoff, D.B., "Innovations in the Control of Gas Turbine Propulsion Systems," paper presented at the Society of Naval Architects and Marine Engineers Spring Meeting, Norfolk, VA, 21-24 May 1985

2. Rubis, C.J., "Governing Ship Propulsion Gas Turbine Engines," *Society of Naval Architects and Marine Engineers Transactions*, v.94, pp.283-308, 1986

3. Artificial Intelligence Section, NASA Johnson Space Center, *CLIPS Users Manual*, Version 4.2, June 1988

4. Rowe, N.C., *Artificial Intelligence Through Prolog*, Prentice Hall, Englewood Cliffs, NJ, 1988

5. National Bureau of Standards Information Report 83-2637, *An Overview of Expert Systems*, by W.B. Gevarter, April 1983

6. Society of Automotive Engineers Technical Paper Series 8603344, *Expert Systems: Misconceptions and Reality*, by D.D. Dankel, February 1986

7. Society of Automotive Engineers Technical Paper Series 860334, *Embedding Artificial Intelligence in Existing Applications*, by W. Moseley and M.B. Zeagler, February 1986

8. American Institute of Aeronautics and Astronautics Technical Report 86-1678, *Knowledge-Based Systems for Rotordynamic Diagnostics,* by B.B. Aggarwal and J.C. Giordano, June 1986

9. Simmons, D.W. and Hamilton, T.P., "HELIX: A Casual Model-Based Diagnostic Expert System," *Journal of the American Helicopter Society,* v.32, pp. 19-25, January 1987

10. American Society of Mechanical Engineers Technical Report 80-GT-158, *U.S. Navy, LM2500 Gas Turbine Condition Monitoring Development Experience,* by M.G. Krandl and D.A. Groghan, December 1980

11. Coward, L., "Gas Turbine Stall," *Deckplate,* v.8, pp.13-16, July-August 1988

12. Cohen, H., Rogers, G.F., and Saravanamutto, H.I., *Gas Turbine Theory,* 2d ed., Longman, 1972

13. Dixon, S.L. *Thermodynamics of Turbomachinery,* Pergamon Press, Oxford, U.K. 1978

14. Boeing Airplane Company, *Boeing Model 502-6A Instruction Book,* Boeing Document 12840, Seattle, WA, 25 February 1953

15. Miller, R.L., *Marine Gas Turbine Modeling for Optimal Control,* Master's Thesis, Naval Postgraduate School, Monterey, CA, December 1986

16. Smith, D.L., "Sequential Linearization as an Approach to Real Time Marine Gas Turbine Simulation," Naval Postgraduate School, Monterey, CA, Publication Pending

17. Holman, J.P., *Experimental Methods for Engineers,* McGraw-Hill, New York, NY, 1984

18. American Society of Mechanical Engineers Technical Report 82-GT-297, *Development of a Low Cost Performance Monitoring System for Use on Board Naval Vessels,* by H.I. Saravanamutto, 1982

19. Kolnsberg, A., "Reasons for Centifugal Compressor Surging and Surge Control," *Journal of Engineering for Power,* v.101, pp.79-86, January 1979

20. Buzzard, W.S., "Controlling Centrifugal Compressors," *Instrumentation Technology,* pp.79-86, January 1979

21. Boyce, M.P., *Gas Turbine Engine Handbook,* Gulf Publishing Company, Houston, TX, 1982

22. Hartman, P.J. "Practical Application of Artificial Intelligence in Naval Engineering." *Naval Engineers Journal,* pp.32-40, November 1988

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria, VA 22304-6145	2
2.	Library, Code 0142 Naval Postgraduate School Monterey, CA 93943-5002	2
3.	Professor David L. Smith, Code 69Sm Department of Mechanical Engineering Naval Postgraduate School Monterey, CA 93943	5
4.	LT James A. Davitt 9820 Beach Blvd. Panama City Beach, FL 32407	3
5.	Mr. Dan Groghan SEA Code 05R3 Naval Sea Systems Command Washington, D.C. 20362	2
6.	Mr. T. Bowen Code 2721 David Taylor Naval Ship Research and Development Center Annapolis, MD 21402	1
7.	Department Chairman, Code 69 Department of Mechanical Engineering Naval Postgraduate School Monterey, CA 93943	1
8.	CLIPS User Help Desk M30 Computer Sciences Corporation 16511 Space Center Boulevard Houston, TX 77058	1
9.	Mr. J. Alan Davitt 39 Huntersfield Rd Delmar, NY 12054	1