AN EVALUATION OF MARINE PROPULSION ENGINES FOR SEVERAL NAVY SHIPS

by

Mark Thomas Stanko

B.S.M.E., University of Utah
1983

SUBMITTED TO THE DEPARTMENT OF OCEAN ENGINEERING AND MECHANICAL ENGINEERING IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREES OF

NAVAL ENGINEER

and

MASTER OF SCIENCE IN MECHANICAL ENGINEERING

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 1992

Signature of Author Mark T Stanko
Department of Ocean Engineering and
Mechanical Engineering
1992

Certified by D. G. Wilson
D. G. Wilson
Thesis Reader

Accepted by A. Douglas Carmichael
A. Douglas Carmichael
Thesis Supervisor and
Chairman, Departmental Graduate Committee
Department of Ocean Engineering

AN EVALUATION OF MARINE PROPULSION ENGINES FOR SEVERAL NAVY SHIPS

by

Mark Thomas Stanko

Submitted to the Departments of Ocean Engineering and Mechanical Engineering on May 1, 1992 in partial fulfillment of the requirements for the Degrees of Naval Engineer and Master of Science in Mechanical Engineering

ABSTRACT

The design of naval ships is a complex and iterative process. The propulsion system is selected early in the design cycle and it has significant impact on the ship design. A complete understanding of the marine propulsion engine alternatives is necessary to facilitate the design.

Five types of marine propulsion engines have been examined and compared. They include an LM-2500 marine gas turbine, an Intercooled Recuperative (ICR) marine gas turbine, a series of Colt-Pielstick PC4.2V medium speed diesels, a series of Colt-Pielstick PC2.6V medium speed diesels, and an Allison 571-KF marine gas turbine module power pak.

To facilitate an integrated propulsion systems study, an engine's computer model has been written that calculates the engine weight, volume, fuel consumption, and acquisition cost. Given user input for propulsor and transmission performance, the engine code will also calculate the required endurance fuel load in accordance with Navy standards. The Engine's computer code allows the user to employ different engine types for cruise and boost operating regimes. The model ensures that the engines are operated within their horsepower and RPM ratings and splits the propulsion load evenly when multiple engines are in use.

The engine's computer code will be integrated into a complete propulsion systems computer code. This will facilitate the analysis of various propulsion alternatives for Navy ships.

This thesis is one part of the three-part propulsion system study. The other two parts are the evaluation of transmissions for a ship's propulsion system, and the evaluation of propulsors for a ship's propulsion system.

Thesis Supervisor: A. Douglas Carmichael, Professor of Ocean Engineering
Thesis Reader: David Gordon Wilson, Professor of Mechanical Engineering

ACKNOWLEDGEMENTS

I wish to thank my advisor, Professor A. D. Carmichael. His support and guidance throughout this project have been both very helpful and illuminating.

TABLE OF CONTENTS

ABSTRACT 2

ACKNOWLEDGEMENTS 3

TABLE OF CONTENTS 4

LIST OF FIGURES 6

LIST OF TABLES 8

CHAPTER 1 INTRODUCTION 9

1.0 Background 9

1.1 The Propulsion System Study 10

1.2 Thesis Objective 12

CHAPTER 2 MARINE PROPULSION ENGINES 13

2.0 Overview 13

2.1 LM-2500 Marine Gas Turbine 14

2.2 ICR Marine Gas Turbine 19

2.3 Allison 571-KF Marine Gas Turbine Module Power Pak 22

2.4 Colt-Pielstick PC4.2V and PC2.6V Medium Speed Diesels 26

CHAPTER 3 COMPARISON OF THE MARINE PROPULSION ENGINES ...31

3.0 Overview 31

3.1 Boost Engine Options 31

3.2 PC4.2V Cruise Engine Options 34

3.3 PC2.6V Cruise Engine Options 35

CHAPTER 4 THE ELECTRIC PLANT GENERATOR SET CONSIDERATIONS 38

4.0 Overview 38

4.1 The DDG Electric Plant Options 38

4.2 The LX Electric Plant Options 41

CHAPTER 5 OVERVIEW OF THE INTEGRATED PROPULSION SYSTEM COMPUTER CODE STRUCTURE 43

5.0 Overview 43

5.1 The Integrated Codes Basic Structure 43

CHAPTER 6 DEVELOPMENT OF THE ENGINE FUNCTIONS FOR THE INTEGRATED PROPULSION SYSTEMS COMPUTER CODE 46

6.0 Overview 46

6.1 The Engine's Program Computer Code 47

CHAPTER 7 SUMMARY 56

REFERENCES 57

APPENDIX A SPECIFIC FUEL CONSUMPTION MODELLING 59

APPENDIX B INTAKE AND UPTAKE WEIGHT CALCULATIONS 73

APPENDIX C INTAKE AND UPTAKE VOLUME CALCULATIONS 75

APPENDIX D FUEL WEIGHT CALCULATIONS 79

APPENDIX E DETAILED LISTING OF COMPUTER CODE 83

APPENDIX F SAMPLE OF OUTPUT 133

LIST OF FIGURES

Figure 1 The LM-2500 Marine Gas Turbine 15

Figure 2 The Simple Cycle Gas Turbine 15

Figure 3 LM-2500 with assumed Cubic Loading 16

Figure 4 LM-2500 Bleed Air Discharge Pressure versus Engine BHP 17

Figure 5 LM-2500 Engine SFC for Cubic Loading 19

Figure 6 One of the Proposed ICR Marine Gas Turbine Designs 20

Figure 7 An ICR Cycle Gas Turbine 20

Figure 8 ICR Gas Turbine SFC for Cubic Loading 22

Figure 9 The Allison 571-KF Marine Gas Turbine 23

Figure 10 The 571-KF Marine Gas Turbine Power Pak Module 23

Figure 11 571-KF Engine SFC for Cubic Loading 25

Figure 12 Colt-Pielstick PC4.2V Diesel Engine 26

Figure 13 Available Colt-Pielstick Medium Speed Diesel Engine Options 27

Figure 14 Colt-Pielstick Diesel Engine Principle of Operations 29

Figure 15 PC4.2V Diesel Engine SFC for Cubic Loading 30

Figure 16 PC2.6V Diesel Engine SFC for Cubic Loading 30

Figure 17 Comparison of Boost Engines' SFC Performance versus BHP 33

Figure 18 Comparison of Selected Cruise Engines' SFC Performance versus BHP 36

Figure 19 SFC Performance for the DDG Generator Set Options 40

Figure 20 Proposed Integrated Propulsion System Computer Code Structure 44

Figure 21 The Engine's Program Computer Code Structure 46

APPENDIX A FIGURES

Figure 1 LM-2500 SFC Data Points 60

Figure 2 ICR Engines's SFC Performance versus BHP 62

Figure 3 Allison 571-KF Engine's Performance versus BHP 63

Figure 4 PC4.2V Engine's SFC Performance versus Operating Conditions 66

Figure 5 PC4.2V SFC Data Points 67

Figure 6 PC4.2V Fuel Rack Limitaitions 68

Figure 7 Extrapolation for PC4.2V Low Cylinder Power Operation 69

Figure 8 PC2.6V Engine's SFC Performance versus Operating Conditions 70

Figure 9 PC2.6V SFC Data Points 71

Figure 10 SFC Extraplolation for PC2.6V Low Cylinder Power Operation 71

Figure 11 Allison 501-K34 SFC Performance versus BHP 72

APPENDIX C FIGURES

Figure 1 Gas Turbine Engine Air Intake Cross Sectional Area 75

Figure 2 Typical Marine Diesel Air Intake System 76

Figure 3 Typical Marine Diesel Exhaust System 78

LIST OF TABLES

Table 1 Matrix of Ship Types and Propulsion System Components 11

Table 2 Navy Standard Day Conditions 13

Table 3 DEMA Standard Conditions 14

Table 4 Summary of Boost Engine Specifications 32

Table 5 Summary of Gas Turbine Acquisition Cost Components 33

Table 6 Summary of Diesel Acquisition Cost Components 33

Table 7 Summary of PC4.2V Cruise Engines' Specifications 34

Table 8 Summary of PC2.6V Cruise Engines' Specifications 35

Table 9 Summary of Specifications for DDG
Electric Plant Generator Set Options 39

Table 10 Summary of Specifications for LX
Electric Plant Generator Set Options 42

Table 11 Listing of the Engine Functions 47

Table 12 The plant_map[7][2] Integer Array 49

Table 13 Listing of all Preprocessor #define Variables 54

APPENDIX D

Table 1 Navy Standard Day Conditions 80

Chapter One

Introduction

1.0 Background

The design of naval ships is a complex and iterative process. The ship design process involves defining the requirements and constraints, selecting and combining candidate technologies, and applying selected design standards in order to meet the requirements for the ship. The term design spiral has been used to describe the iterative nature of this process cycle. The cycle must be repeated until the synergistic integration of the component technologies are deemed to satisfy the design requirements both technically and politically.

For naval ships, the combat systems and the propulsion plant are two major systems that have important impact on the ship design. A thorough understanding of the pros and cons of the candidate technologies for these systems is necessary to facilitate efficiency in the design spiral process.

The propulsion system is selected early in the design spiral. Once the component technologies for the propulsion system have been selected, there is little flexibility for change in the propulsion system. This further emphasizes the need to be able to define and trade-off the propulsion component technologies both qualitatively and quantitatively. The trade-off should result in the optimum solution given the constraints in the requirements statement.

There have been many advances and improvements in naval propulsion technologies. These advances have occurred in each of the three major propulsion system components which include the marine propulsion engines, the transmissions, and the propulsors. The evolving component technologies should lead to potentially promising propulsion systems for future naval ship designs. It can be expected that the

future propulsion systems will have significant performance and costs impacts on the ship designs.

A study has been initiated to investigate the different propulsion technologies. One of the goals of the study is to determine which of the technologies has the most promise and deserves the emphasis for future designs. The individual propulsion component technologies have been identified and characterized [1]. The study is entering its second phase which will define and determine the relative merits of the different propulsion technologies as applied to different classes of ships. A second goal of the study is to develop a computer model for quantifying and analyzing propulsion system alternatives.

1.1 The Propulsion System Study

The propulsion system study is centered around the development of a computer model for comparing alternative propulsion systems on a given ship class. The computer code, written in the computer language C, combines the characteristics of user selected individual propulsion system components and outputs the performance characteristics, weight, volume, and costs for the prospective propulsion system. Additionally, the impact of the propulsion system is determined for a selected ship class. For the study, two surface ship classes and one submarine are considered. The first surface ship class is the ARLEIGH BURKE (DDG 51) class destroyer. The second surface ship class is a naval amphibious ship class designated LX.

The propulsion system analysis has been split into three areas of responsibility: propulsion engines, transmissions, and propulsors. Each team member is responsible for ensuring that their propulsion component's characterization and computer modelling integrates into the overall propulsion system model. Table 1 presents the three by three matrix representation of the project.

	Destroyer	LX Amphibious	Submarine
Propulsion Engines	-LM-2500 -ICR Gas Turbine -PC4.2V Diesels -PC2.6V Diesels Allison 571-KF Power Pak	-LM-2500 -PC4.2V Diesels -PC2.6V Diesels	-Fuel Cells -Closed Brayton Cycle -Stirling Cycle -Semi-closed cycle diesel -Diesel/electric -Aluminum /Oxygen cell
Transmission	-Geared Mech. -Geared Mech. w/TOSI couple -Epicyclic - AC to AC	-Geared Mech. -Geared Mech. w/TOSI couple -Epicyclic - AC to AC	-Geared Mech.
Propulsor	<u>Propellers</u> -Fixed Pitch -CRP -Contrarotating -Fixed Pitch with preswirl stator -Ducted version of the above <u>Waterjets</u>	<u>Propellers</u> -Fixed Pitch -CRP	-Contrarotating

Table 1 Matrix of Ship Types and Propulsion System Components

1.2 Thesis Objective

This thesis focuses on the evaluation of marine propulsion engines for a surface ship's propulsion system. The development of the engine portion of the integrated propulsion system computer code is discussed. Five marine propulsion engines types are considered. They include an LM-2500 marine gas turbine, an Intercooled Recuperative (ICR) marine gas turbine, a series of Colt-Pielstick PC4.2V medium speed diesels, a series of Colt-Pielstick PC2.6V medium speed diesels, and an Allison 571-KF marine gas turbine module power pak. The user selected marine propulsion engines have their characteristics and performance specifications modeled and incorporated into the integrated propulsion system model. The engine computer code allows the user to employ different engine types for cruise and boost operating regimes. The model ensures that the engines are operated within their horsepower and RPM ratings and splits the propulsion load evenly when multiple engines are in use. The engine model also incorporates design standards and constraints to determine a given ship's fuel load requirements. The propulsion engines are compared in terms of weight, volume, fuel consumption, and capital cost.

This thesis is one part of the three-part propulsion system study. The other two parts are the evaluation of transmissions for a ship's propulsion system, and the evaluation of propulsors for a ship's propulsion system. Ultimately, the three parts will be combined and integrated to determine the impact of various propulsion systems on a DDG class ship and an LX class ship.

Chapter 2

MARINE PROPULSION ENGINES

2.0 Overview

This chapter provides a detailed description for each of the five types of selected marine propulsion engines. For this study, three marine gas turbines and two series of medium speed diesels were considered. The gas turbines include two simple cycle and one intercooled and recuperative cycle. The diesels include two different power sizes of engines. All of the engines are only available in discrete sizes as provided by the manufacturer.

The engines are classified as cruise or boost engines. The cruise engines are used to meet the ship cruise power requirement. The boost engines are used to meet the ship minimum sustained speed requirement. The propulsion plant configuration determines how a given engine is classified. In some configurations where there is only one engine type, the engine serves as both the cruise and boost engine. In configurations where there are more than one engine type, such as CODOG, the diesel is the cruise engine and the gas turbine is the boost engine. In a combined ICR and LM-2500 plant, the ICR would serve both cruise and boost operation while the LM-2500 would only serve boost operation.

The Navy standard day conditions shown in table 2 were assumed for all gas turbine performance calculations.

100°F Ambient Temperature
14.7 psia Ambient Pressure
4.0 in of H_2O Intake Loss
6.0 in of H_20 Exhaust Loss
40 % relative humidity
18,400 Btu/lbm Lower Heating Value
85°F Seawater Inlet Temperature

Table 2 Navy Standard Day Conditions

The Diesel Engine Manufacturer's Association (DEMA) standard conditions shown in table 3 were assumed for all diesel performance calculations. The diesel jacket water pumps and lube oil pumps will be motor driven, thus their impact on SFC will be accounted to the electric plant prime mover.

90°F Ambient Temperature
14.7 psia Ambient Pressure
No Engine Driven Jacket Water Pump
No Engine Driven Lube Oil Pump
18,360 Btu/lbm Lower Heating Value
85°F Intercooler Seawater Inlet Temperature

Table 3 DEMA Standard Conditions

2.1 LM-2500 Marine Gas Turbine

The LM-2500 marine gas turbine is a simple-cycle gas turbine or open Brayton-cycle engine. Figure 1 shows a picture of the engine and figure 2 shows a schematic of the simple-cycle gas turbine engine. The LM-2500 is currently rated for Navy use at a maximum of 26,250 BHP at 3600 RPM. The maximum output torque, limited by the output shaft flexible coupling, is 60,000 lb-ft. The engine is composed of two primary sections, the gas generator and the power turbine. The gas generator consist of a 16 stage axial compressor, an annular combustor , and a two stage high pressure turbine. The high pressure turbine exhaust to the low pressure or power turbine. The power turbine is aerodynamically coupled to the gas generator and consist of six stages.

Ideally the engine power versus engine RPM is scheduled to produce the optimum SFC performance through the required operating range. This is possible for electric drive applications. However for mechanical drive applications, the power versus RPM scheduling is constrained by the required propulsor RPM scheduling. In this case, an estimate for the required engine power versus RPM can be approximated by applying equation (1) which gives a cubic loading relationship.

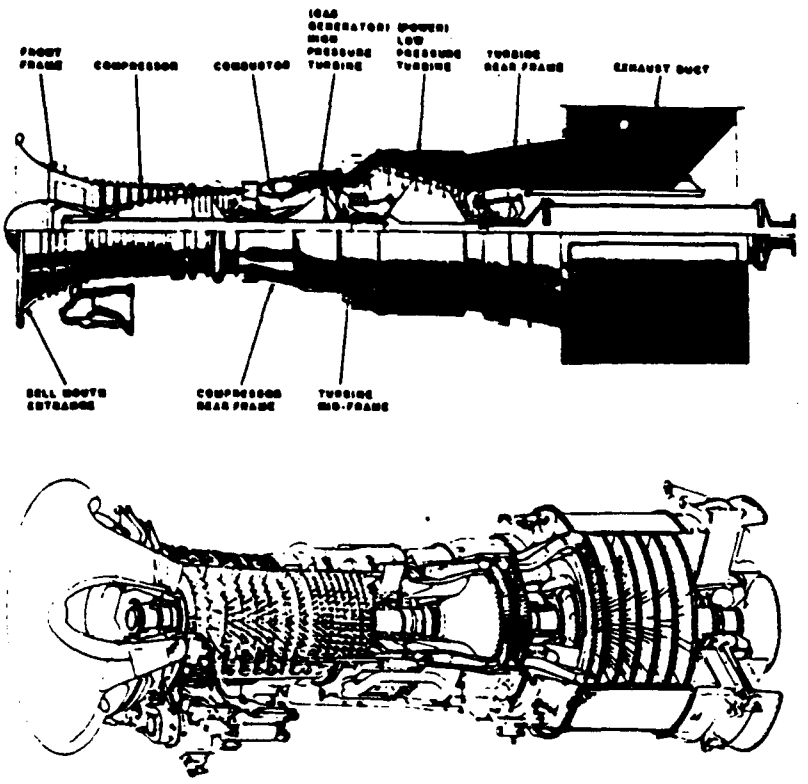

Figure 1 The LM-2500 Marine Gas Turbine [2]

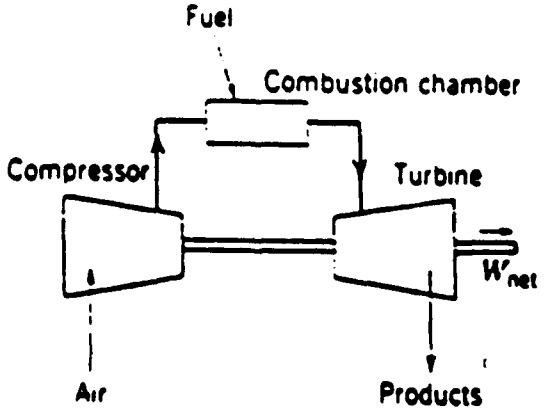

Figure 2 The Simple Cycle Gas Turbine [3]

$$(\%\ \text{Maximum BHP}) = (\%\ \text{Maximum RPM})^3 \qquad (1)$$

Figure 3 shows the power turbine output BHP versus power turbine speed (NPT) obtained by applying the cubic approximation in equation (1).

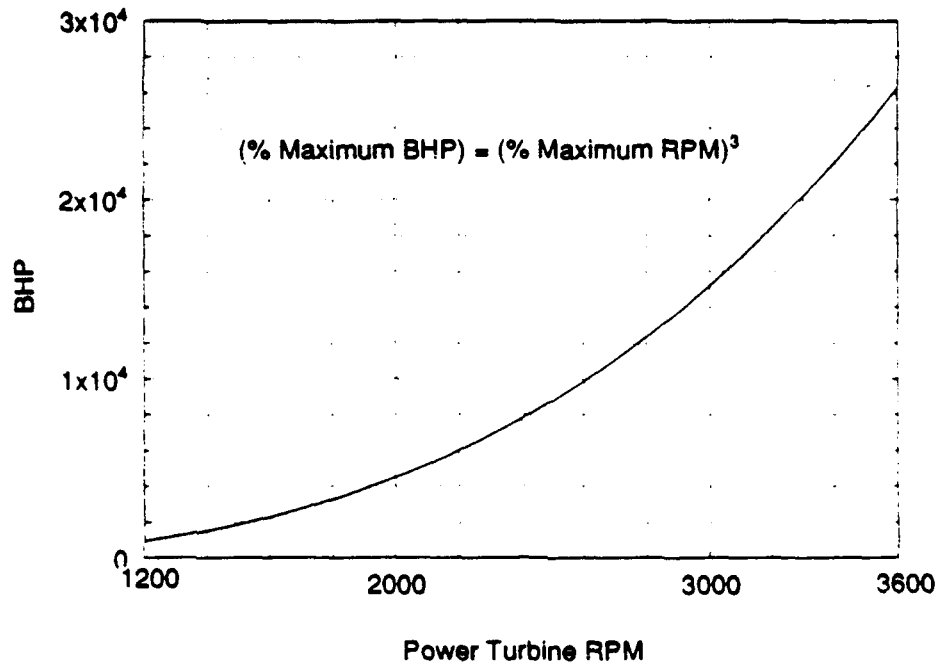

Figure 3 LM-2500 with assumed Cubic Loading

The axial flow 16 stage compressor develops an overall 17:1 pressure ratio. The inlet guide vanes and the first six rows of variable stator vanes adjust to prevent compressor stall during variable speed engine operation. Up to 12 percent of the total engine air flow may be drawn off of the 16th stage to supply customer bleed air [4]. Drawing bleed air off of the engine while maintaining a given power requirement will cause the power turbine inlet temperature (T48) to rise. Equation (2) gives the increase in T48 as a function of the ratio of the bleed air flow (WB3) to the total engine air flow (W2).

$$T48_{bleed} = T48_{no\text{-}bleed} * (1 + 1.167 * WB3/W2) \qquad (2)$$

The LM-2500 is rated to a maximum T48 of 1625^{O}F. If customer bleed is used during

high power operation, equation (2) must be applied to ensure that the engine is not outside of its T48 limit.

Since the LM-2500 is a variable speed engine, the compressor discharge bleed pressure (PE3D) varies with engine BHP. Figure 4 shows the relation of PE3D with engine BHP. Equation (3) must be applied to the values obtained from figure 4 to allow for the decrease in PE3D as a function of the ratio of the bleed air flow (WB3) to the total engine air flow (W2).

$$PE3D_{bleed} = PE3D_{no\text{-}bleed} * (1 - 1.25 * WB3/W2) \quad (3)$$

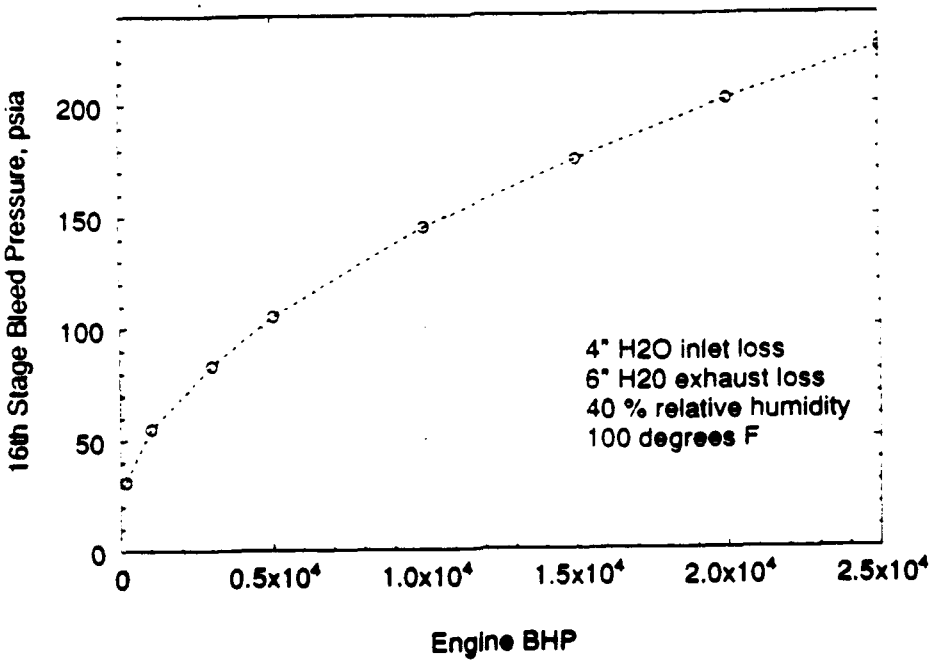

Figure 4 LM-2500 Bleed Air Discharge Pressure versus Engine BHP

It is important to note that the engine's customer bleed air pressure drops off at the low power end. This performance can have significant impact if the engine is designated as the primary source to provide customer bleed air for systems such as prairie air, masker air, start air, and anti-icing air. A standard bleed air system on the DD 963 class destroyer and CG-47 class cruiser requires a normal bleed air pressure of 75 psia [5]. Applying equation (3) to the 75 psia bleed air pressure results in a required 78.9 psia for

16th stage bleed air pressure. From figure 4, this 16th stage bleed air pressure, assuming cubic engine loading, requires the engine to be operating at 2715 BHP. For a typical destroyer with a typical propulsion plant this minimum engine BHP required to maintain normal bleed air pressure would result in a ship speed of 8 to 12 knots (ship dependant). Below this speed, the bleed air pressure would drop below normal values. To alleviate this problem, the LM-2500 is not used as the primary source of bleed air. Instead the ship service gas turbine generator (SSGTG) serves as the primary source of bleed air. The SSGTG is a constant speed engine that operates over a relatively small power range. This results in the SSGTG providing a relatively constant bleed air pressure at approximately 100 psia.

The annular combuster contains 30 fuel nozzles mounted in swirl cups that provide for mixing. The engine is configured for liquid fuel (DFM or JP5) use, though it can be configured for natural gas or other novel fuels. An inlet diffuser to the combuster ensures that relatively uniform flow is seen by the combuster even though the compressor discharge pressure may be varying significantly.

The two stage high pressure or gas generator turbine is used to drive the compressor and the accessory gear box. This portion of the turbine is subjected to the highest turbine temperatures and stresses. The blades uses special materials and film cooling through passages in the blades to allow for operating in the high temperatures.

The six stage low pressure or power turbine is aerodynamically coupled to the gas generator turbine. The gas generator can be scheduled so that the power turbine operates between 900 and 3600 RPM. However, since this project allows for Propulsion Derived Ship Service (PDSS) electric power to be driven off of the power turbine output shaft; the power turbine will be restricted to operate between 1200 and 3600 RPM. This RPM range is required for the PDSS units as discussed in detail by Hultgren [6].

The Specific Fuel Consumption (SFC) for the LM-2500, assuming cubic engine loading, is shown in figure 5. An engine SFC model was developed, that maps the

engine SFC as a function of engine bhp and output RPM. The engine cubic loading shown in figure 3 was applied to this model to develop figure 5. The details for the development of the LM-2500 SFC model can be found in appendix A.

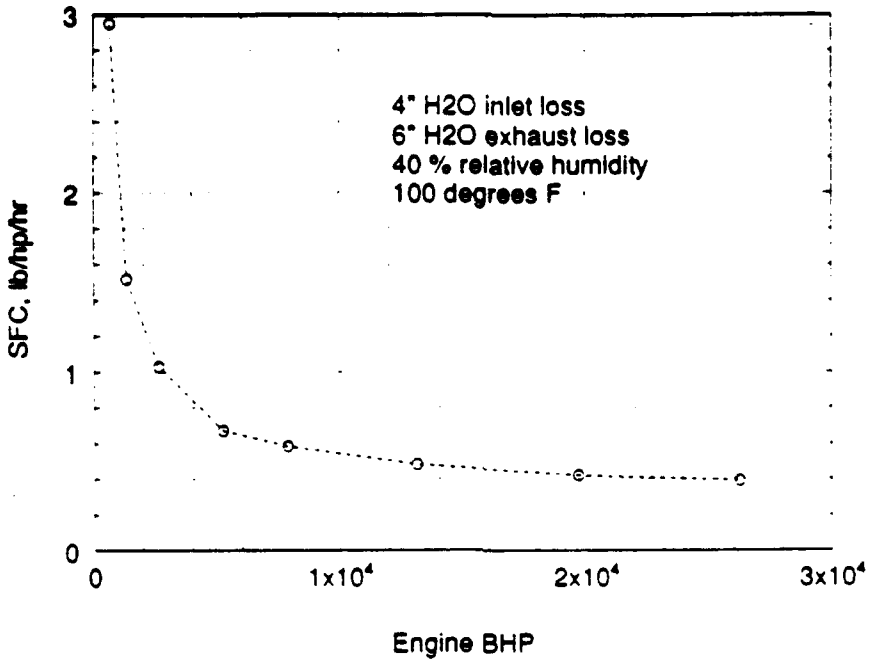

Figure 5 LM-2500 Engine SFC for Cubic Loading

2.2 ICR Marine Gas Turbine

The Intercooled and Recuperative (ICR) marine gas turbine is a modified open Brayton-cycle engine that includes intercooling and recuperation. The inclusion of intercooling and recuperation into the cycle is expected to provide a reduction in the SFC, compared to the simple gas turbine cycle, of 30 to 35 percent. The actual engine that will be used for Navy applications is not yet completely developed, however the required specifications for the engine have been published [7]. Figure 6 shows a picture of one of the proposed engine designs and figure 7 shows a schematic for that proposed ICR engine. The initial ICR engine will be rated for Navy use at a maximum of 26,400 BHP at 3600 RPM.

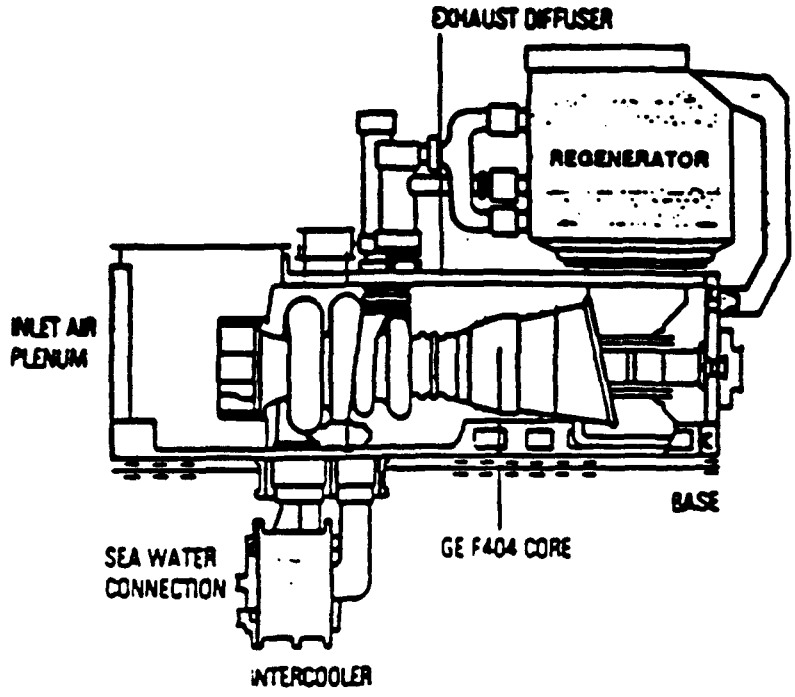

Figure 6 One of the Proposed ICR Marine Gas Turbine Designs[1]

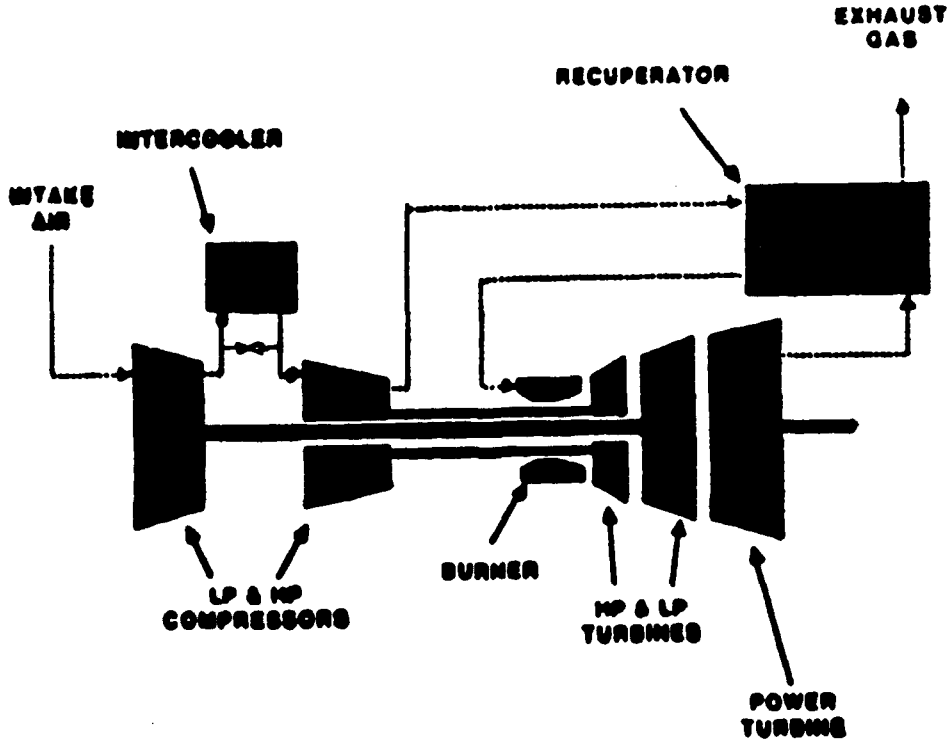

Figure 7 An ICR Cycle Gas Turbine [8]

The ICR gas turbine is a twin spooled gas generator coupled to a multistage power turbine. Each spool of the gas generator has an axial flow compressor coupled to a compressor turbine. Between the two compressors, the air temperature is reduced by the intercooler. The power turbine is expected to have variable stator blades to improve part-load performance. Figure 7 shows how the individual gas generator turbines are mechanically connected to their respective compressors via concentric shafting, and also implicitly shows via the required logical air flow how they are aerodynamically coupled. The low pressure gas generator turbine exhaust to the power turbine. The power turbine is aerodynamically coupled to the low pressure gas generator.

The ICR engine performs the intake air compression in two stages. The low pressure compressor heats the intake air as it performs the work necessary for the first stage compression. The air exits the low pressure compressor and is passed to a heat exchanger where the temperature is brought to near ambient conditions. This process step results in less work required by the high pressure compressor to complete the compression process.

The high pressure compressor discharge is drawn off of the engine and passed to the recuperator. The recuperator is a heat exchanger that transfers the heat energy from the exhaust gases coming off of the power turbine to the high pressure compressor discharge air before it enters the combuster. As mentioned, the expected overall SFC reduction has a result of combined intercooling and recuperation should be 30 to 35 percent.

For the ICR engine, the power versus engine RPM should ideally be scheduled to produce the optimum SFC performance through the required operating range. This is possible for electric drive applications. However for mechanical drive applications, the power versus RPM scheduling is constrained by the required propulsor RPM scheduling. As in the LM-2500 case, a first estimate for the required ICR engine power versus RPM can be approximated by applying equation (1) which gives a cubic loading relationship.

The ICR engine's specifications [7], assume that no customer bleed air will be provided by the engine. Based on this specification, it will be assumed that a SSGTG will provide customer bleed air if it is required for such services as prairie and masker air.

The SFC for the ICR gas turbine, assuming the prescribed cubic engine loading [7], is shown in figure 8. An ICR engine SFC model was developed that maps the engine SFC as a function of the percent of total engine power. The details of the ICR SFC model can be found in appendix A.

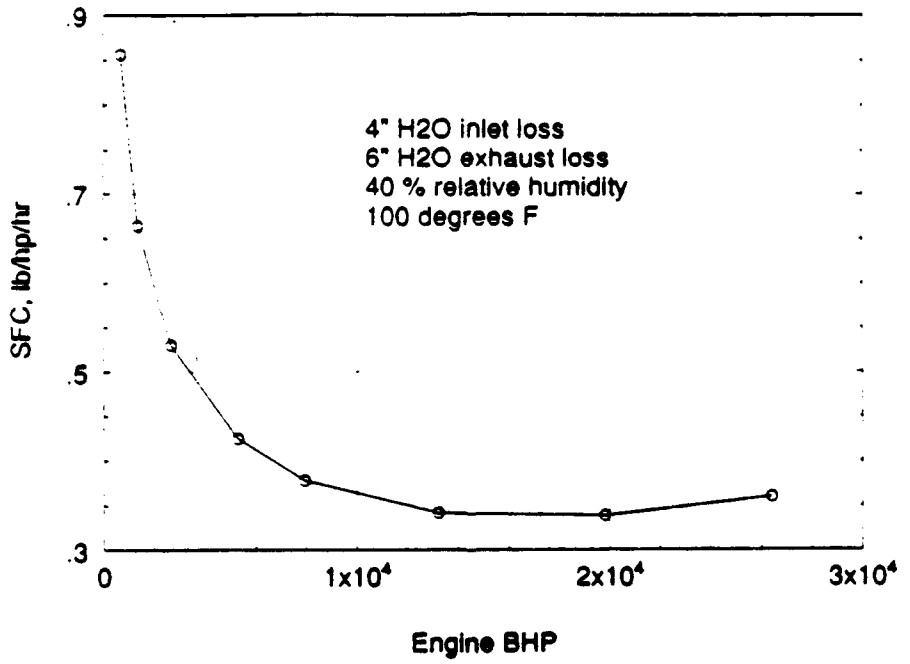

Figure 8 ICR Gas Turbine SFC for Cubic Loading

2.3 Allison 571-KF Marine Gas Turbine Module Power Pak

The Allison 571-KF marine gas turbine is a simple-cycle gas turbine or open Brayton-cycle engine. Figure 9 shows a picture of the engine. The Stewart

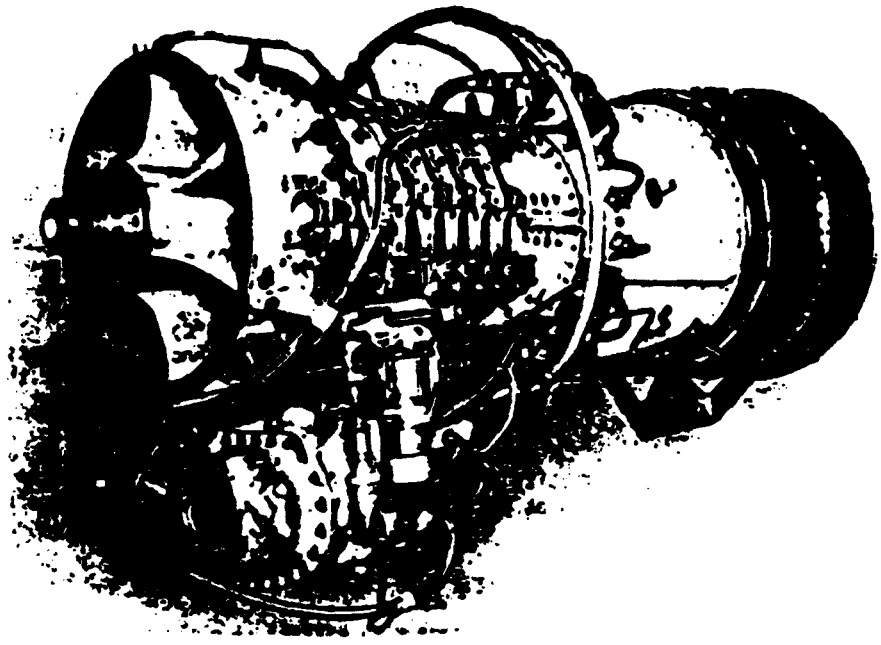

Figure 9 The Allison 571-KF Marine Gas Turbine [9]

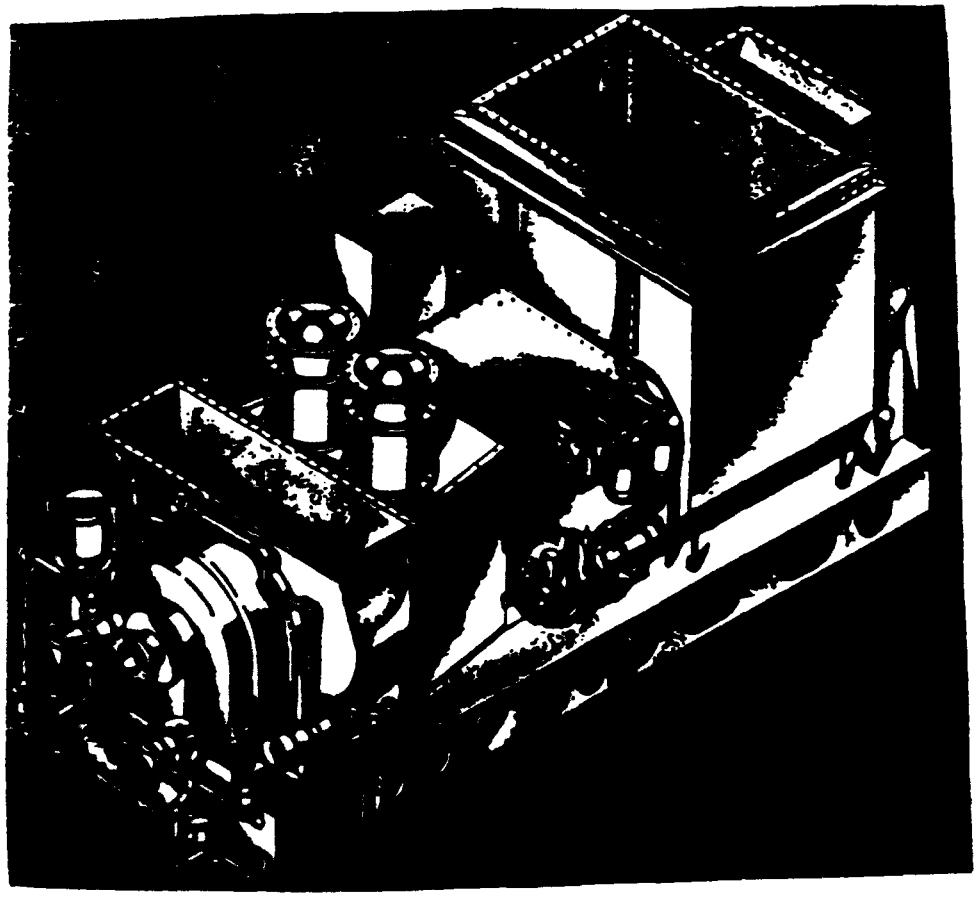

Figure 10 The 571-KF Marine Gas Turbine Power Pak Module [10]

Stevenson Company has incorporated the engine into a gas turbine module power pak shown in figure 10. The module power pak features the 571-KF engine with a reduction gear and supporting auxiliary equipment all enclosed in a module and mounted on a common pallet.

The 571-KF is currently rated for a maximum of 6000 BHP at 1800 RPM. The maximum output torque is 5,000 lb-ft. The engine is composed of two primary sections, the gas generator and the forward drive power turbine. The gas generator consist of a 13 stage axial compressor, an annular combustor , and a two stage high pressure turbine. The high pressure turbine exhaust to the low pressure or power turbine. The power turbine is aerodynamically coupled to the gas generator and consist of three stages.

Ideally the engine power versus engine RPM is scheduled to produce the optimum SFC performance through the required operating range. This is possible for electric drive applications. However for mechanical drive applications such as direct coupling to a waterjet, the power versus RPM scheduling is constrained by the required propulsor RPM scheduling. In this case, an estimate for the required engine power versus RPM can be approximated by applying equation (1) which gives a cubic loading relationship.

The axial flow 13 stage compressor develops an overall 12.7:1 pressure ratio. The inlet guide vanes and the first five rows of variable stator vanes adjust to prevent compressor stall during variable speed engine operation. Up to 5 percent of the total engine air flow may be drawn off of the 10th stage to supply customer bleed air [9].

The annular combuster contains 16 fuel nozzles mounted in swirl cups that provide for mixing. The engine is configured for liquid fuel (DFM or JP5) use, though it can be configured for natural gas or dual fuel (natural gas and liquid fuel). An inlet diffuser to the combuster ensures that relatively uniform flow is seen by the combuster even though the compressor discharge pressure may be varying significantly.

The two stage high pressure or gas generator turbine is used to drive the compressor and the accessory gear box. This portion of the turbine is subjected to the

highest turbine temperatures and stresses. The engine is rated for a maximum high pressure turbine temperature of 1477^{O}F. The turbine blades uses special materials and film cooling through passages in the blades to allow for operating at the high temperatures.

The three stage low pressure or power turbine is aerodynamically coupled to the gas generator turbine. The gas generator is normally scheduled so that the power turbine operates between 6000 and 12,000 RPM. The module reduction gear ratio is 6.66, so the output RPM from the module is 900 to 1800 RPM. The manufacturer advertises an optional system that will allow for output RPM down to zero. However, for applications requiring Propulsion Derived Ship Service (PDSS) electric power to be driven off of the power turbine output shaft; the power turbine will be restricted to operate between 1200 and 3600 RPM. This RPM range is required for the PDSS units as discussed in detail by Hultgren [6].

The Specific Fuel Consumption (SFC) for the 571-KF, assuming cubic engine loading, is shown in figure 11.

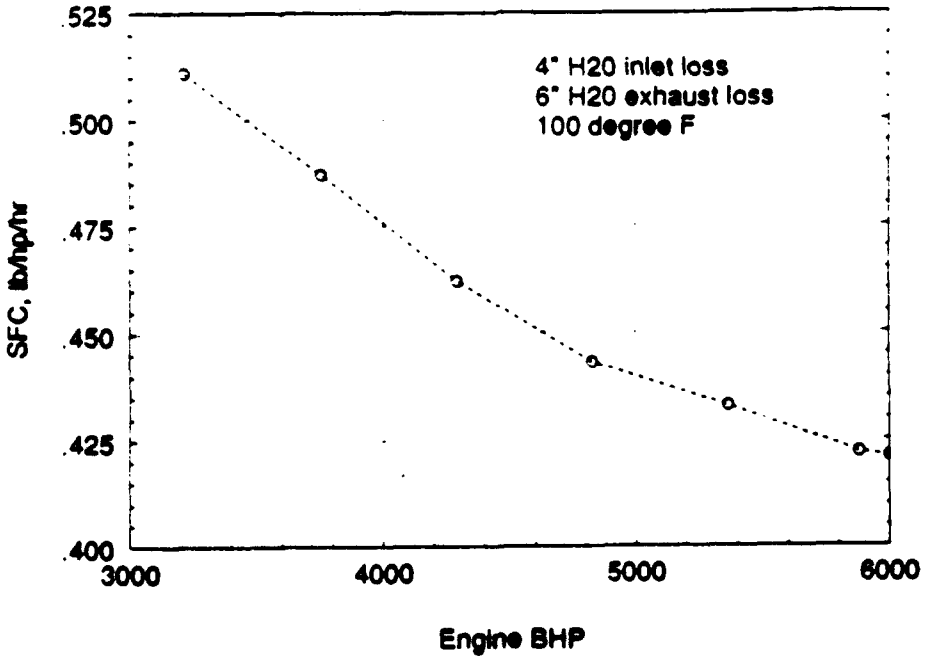

Figure 11 571-KF Engine SFC for Cubic Loading

An engine SFC model was developed that maps the engine SFC as a function of engine bhp. The details for the development of the 571-KF SFC model can be found in appendix A.

2.4 Colt-Pielstick PC4.2V and PC2.6V Medium Speed Diesels

The Colt-Pielstick medium speed diesels are open Diesel cycle, four stroke engines. Figure 12 shows a picture of the PC4.2V engine. The PC4.2V and PC2.6V engines are available in a range of power levels depending on the number of cylinders installed in the engine. Figure 13 shows the available engine ratings. Based on the projected ship propulsion power requirements for boost, the 16 cylinder PC4.2V engine will be the only diesel boost engine option. Based on the projected ship power requirements for cruise, the PC4.2V 10, 12, and 14 cylinder and the PC2.6V 10, 12, 14, and 16 cylinder engines will be considered for cruise engine options.

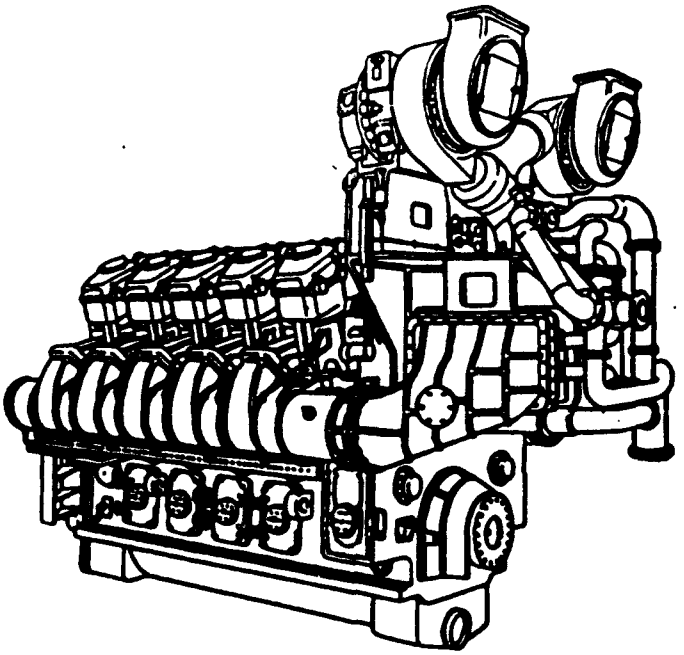

Figure 12 Colt-Pielstick PC4.2V Diesel Engine [11]

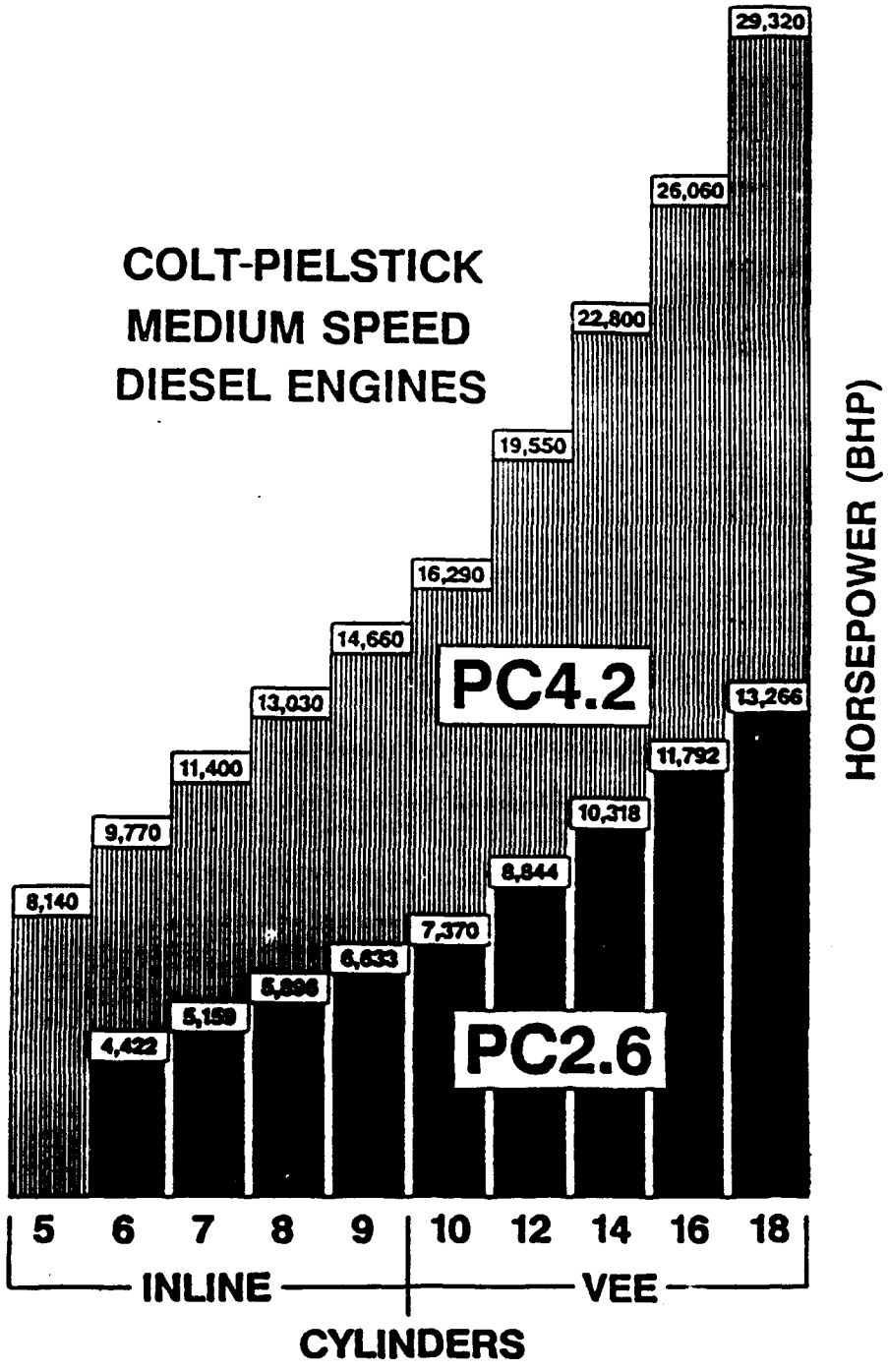

Figure 13 Available Colt-Pielstick Medium Speed Diesel Engine Options [11]

Figure 14 provides a description for the four stroke Colt-Pielstick diesel engine's principle of operation. The PC4.2V and PC2.6V are very similar in design and construction. The primary difference is in the power rating and thus size of the components. Both engines incorporate the vee cylinder arrangement. Direct reversing engines are available, but were not considered for this project. As a result either a reversing gear or a CRP propeller will have to be used for reversing. It is assumed that the diesels will be ran on DFM fuel at the conditions prescribed in table 3. The engines can be configured to run on various grades of heavy fuel. However, this requires upgrading the exhaust valves and engine fuel systems to run on the harsher fuel. The engines incorporate turbocharging with each bank of cylinders in the vee configuration served by its own turbocharger.

The PC4.2V diesel engine is rated at 1629 BHP per cylinder. The engine operates between 125 and 400 RPM. The SFC for the engine, assuming cubic engine loading, is shown in figure 15. An engine model was developed that maps the engine SFC as a function of BHP and RPM. The details for the development of the diesel SFC model can be found in appendix A.

The PC2.6V diesel engine is rated at 737 BHP per cylinder. The engine operates between 200 and 520 RPM. The SFC for the engine, assuming cubic engine loading, is shown in figure 16.

PRINCIPLE OF OPERATION
Diesel (Distillate or Heavy Fuels) Engines

These engines all operate on the "four stroke" cycle, that is, all events - air intake, compression, fuel admission, combustion, expansion, exhaust, and scavenging, occur during four strokes of the piston or two revolutions of the crankshaft (720°). See Fig. 1.

Starting at 0° TDC with the air intake valves already open, the piston moves down and air is admitted to the cylinder (1). Shortly after BDC, the air intake valves close, the cylinder is therefore sealed and compression takes place on the upward stroke (2). The heat of the air increases due to compression and fuel is injected shortly before TDC. The fuel is immediately ignited by the hot air, and combustion commences. Due to the head of combustion, the gases expand, resulting in an almost instantaneous increase in cylinder pressure. This forces the piston down on the power stroke (3).

Just before BDC, the exhaust valves open and remain open throughout the exhaust stroke (4) and into the intake stroke before closing. The air intake valves open during the exhaust stroke and both air and exhaust valves are open simultaneously for about 94° of rotation. This overlap allows incoming air to purge the cylinder of all exhaust gas and also aids in cooling the piston crown, upper cylinder and exhaust valves.

NOTE: Timing shown is nominal design. Individual applications may require modification. Each situation requires evaluation by Fairbanks Morse Engineering.

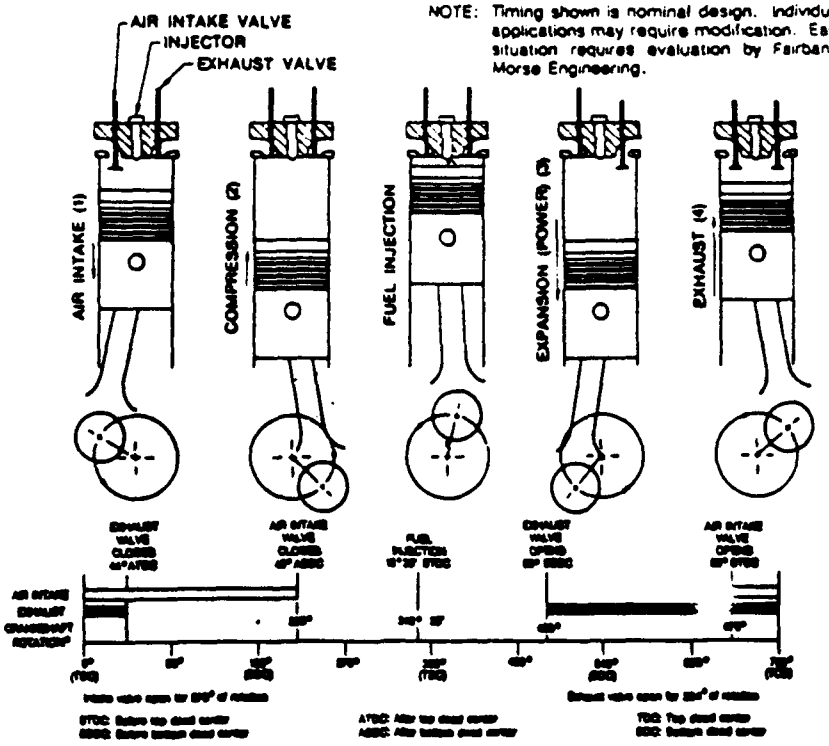

Fig. 1 Timing Diagram - PC4.2/2V

Figure 14 Colt-Pielstick Diesel Engine Principle of Operations [12]

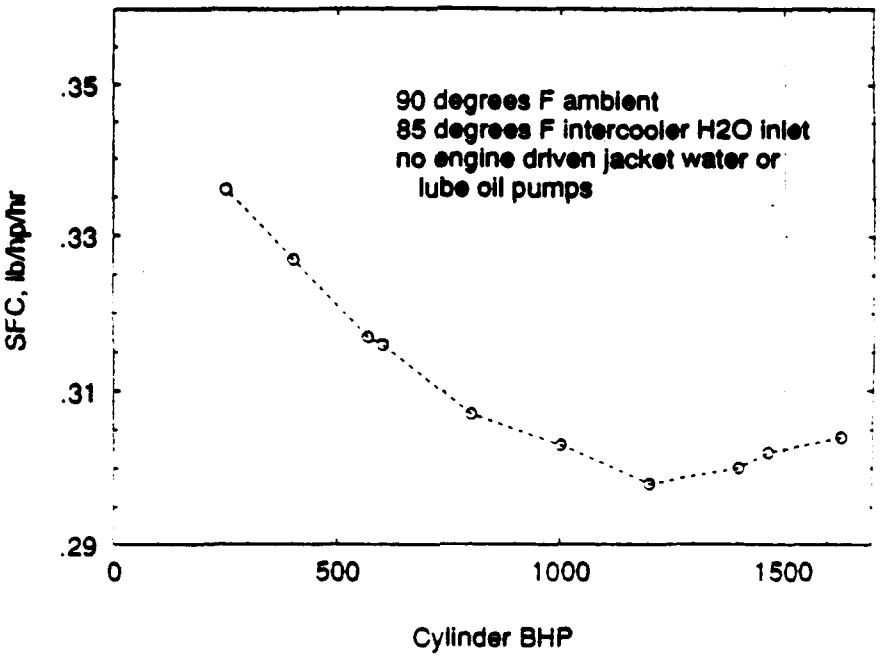

Figure 15 PC4.2V Diesel Engine SFC for Cubic Loading

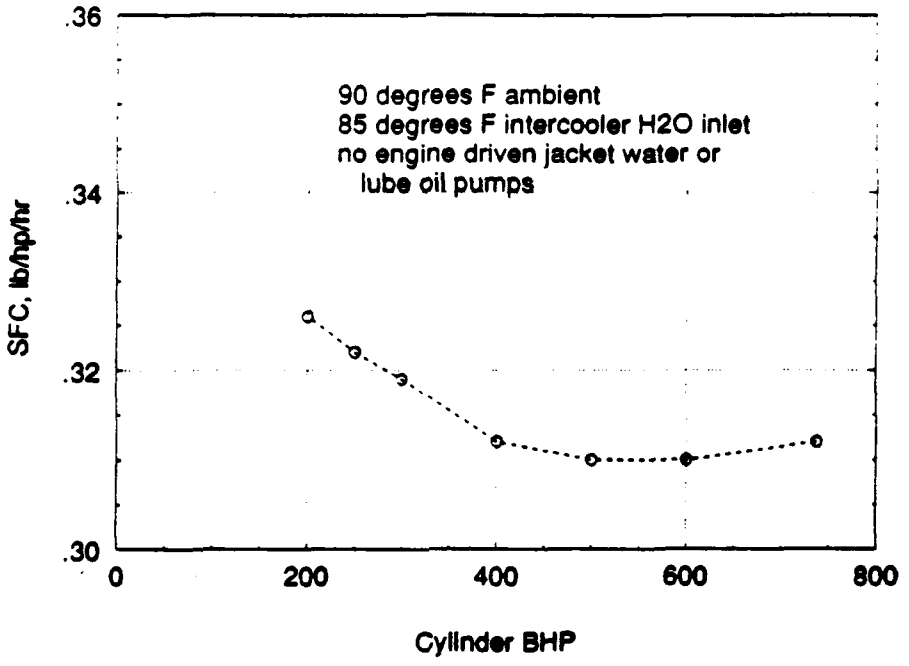

Figure 16 PC2.6V Diesel Engine SFC for Cubic Loading

Chapter 3

Comparison of The Marine Propulsion Engines

3.0 Overview

Determining the impact of a propulsion system on a ship requires that the analysis consider the synergistic integration of all the propulsion components. However, a comparison of the individual technologies within a component group can help to identify the specific individual promising technologies. With this motivation, the selected propulsion engines are compared.

3.1 Boost Engine Options

As described in chapter 2, the designated boost engines were selected to meet the projected power requirements for minimum sustained speed. Depending on the specific selection for the propulsion plant components, the engines may also be used for cruise operation. Table 4 provides a summary of the specifications for the boost engines.

The information in the table is used by the engine computer code to define the engine characteristics. The information for each engine is listed in the same order has it is stored in the array called "boost_engine_specs[14]". The program code is a value used by the computer code to identify the engine type and control branching logic.

Since the locations of the engines in the ship may vary depending on the propulsion system and the specific arrangement requirements, the intake and uptake ducting weight and volume will vary. Therefore the average weight per linear foot and the average cross sectional area per linear foot of ducting were calculated as detailed in appendix B and appendix C, respectively. Once the engine is placed and the length of ducting is known, the total weight and volume for the engine ducting can be calculated by multiplying by the known duct length.

	LM-2500	ICR	PC4.2 16 cyl	Allison 571-KF Power Pak
Maximum Engine Power, bhp	26,250	26,400	26,060	6000
Maximum Engine RPM	3600	3600	400	1800
Minimum Engine RPM	1200	1200	125	900
Number of Cylinders, Diesel's only			16	
Program Code	1	2	3	5
Weight of Engine, lbs	59,000	120,000	639,340	15,000
Linear Weight of Intake, lbs/ft	493.2	494.5	322.9	364.5
Linear Weight of Uptake, lbs/ft	799.0	801.1	500.5	590.5
Length of Engine, ft	26.5	26.5	42.7	15.8
Width of Engine, ft	8.7	8.7	17.0	5.7
Height of Engine, ft	10.4	22.2	26.2	7.7
Area Cross-section of Intake, ft^2	119.7	119.7	12.6	37.3
Area Cross-section of Uptakes, ft^2	162.5	162.5	19.6	82.3
Acquisition Cost, million 1991 $	4.5	6.5	7.8	3.5

Table 4 Summary of Boost Engines' Specifications

The acquisition cost for the Allison 571-KF Power Pak was quoted by the manufacturer [13] and includes all of the module components but does not include the cost for spare parts. The other engines' cost were reported from NAVSEA [14]. All gas turbine costs figures include the acquisition items shown in table 5. All diesel costs figures include the acquisition items shown in table 6.

Engine Cost
Attached Auxiliaries
Enclosure
Control Module
Removal Rails
INCO Spares
Tech Manuals

Table 5 Summary of Gas Turbine Acquisition Cost Components

Engine Cost
Control Room Console
Local Control Panel
Recommended Spares
Tech Manuals and Drawings
Field Support Service

Table 6 Summary of Diesel Acquisition Cost Components

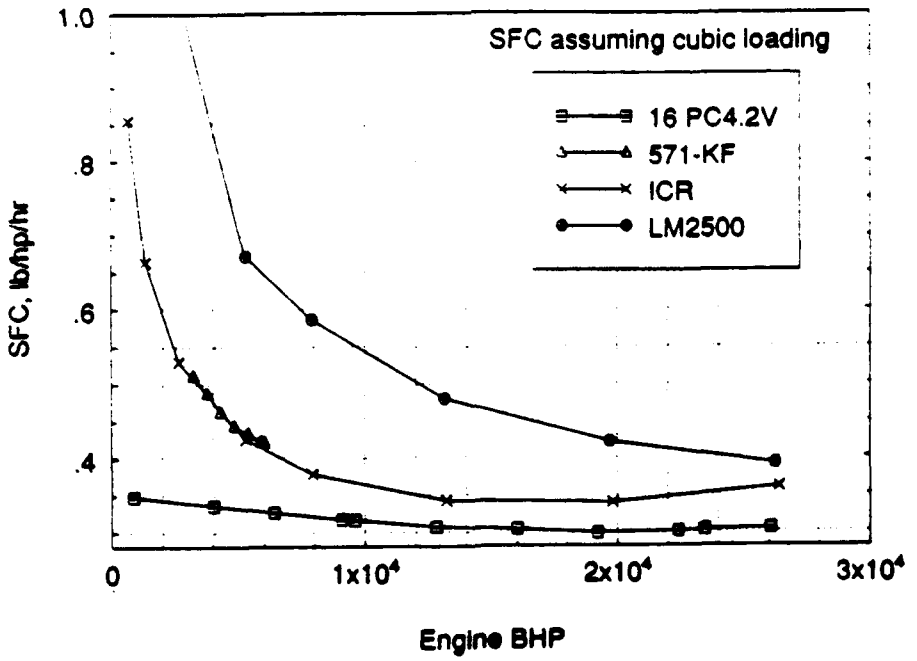

Figure 17 Comparison of Boost engines' SFC performance versus BHP

Figure 17 provides a comparison of the SFC for each of the boost engines as a function of BHP assuming cubic engine loading. It can be seen that the ICR engine's

SFC performance is very competitive with the diesel down to approximately 25 percent engine loading.

3.2 PC4.2V Cruise Engine Options

Table 7 summarizes the specifications for the PC4.2V engines that are considered for cruise only operation. As before, the information in the table is used by the engine computer code to define the engine characteristics. The information for each engine is listed in the same order has it is stored in the array called "cruise_engine_specs[14]". It is expected that only one PC4.2V cruise diesel will be required for cruise power.

	PC4.2V 10 cyl	PC4.2V 12 cyl	PC4.2V 14 cyl
Maximum Engine Power, bhp	16,290	19,550	22,800
Maximum Engine RPM	400	400	125
Minimum Engine RPM	125	125	125
Number of Cylinders, Diesel only	10	12	14
Program Code	3	3	3
Weight of Engine, lbs	425,500	507,060	577,600
Linear Weight of Intake, lbs/ft	255.3	279.6	302.0
Linear Weight of Uptake, lbs/ft	395.7	433.4	468.1
Length of Engine, ft	34.2	37.0	39.7
Width of Engine, ft	17.0	17.0	17.0
Height of Engine, ft	25.2	25.2	26.2
Area Cross-section of Intake, ft^2	12.6	12.6	12.6
Area Cross-section of Uptakes, ft^2	19.6	19.6	19.6
Acquisition Cost, million 1991 $	5.43	6.23	7.10

Table 7 Summary of PC4.2V Cruise Engines' Specifications

The average weight per linear foot and the average cross sectional area per linear foot of ducting were calculated as detailed in appendix B and appendix C, respectively. Once the engine is placed and the length of ducting is known, the total weight and volume for the engine ducting can be calculated by multiplying by the known duct length. The acquisition cost for the engines were reported from NAVSEA [14]. The diesel costs figures include the acquisition items shown in table 6.

3.3 PC2.6V Cruise Engine Options

Table 8 summarizes the specifications for the PC2.6V engines that are considered

	PC2.6V 10 cyl	PC2.6V 12 cyl	PC2.6V 14 cyl	PC2.6V 16 cyl
Maximum Engine Power, bhp	7370	8844	10,318	11,792
Maximum Engine RPM	520	520	520	520
Minimum Engine RPM	200	200	200	200
Number of Cylinders, Diesel only	10	12	14	16
Program Code	4	4	4	4
Weight of Engine, lbs	121,275	145,530	165,375	183,015
Linear Weight of Intake, lbs/ft	171.7	188.1	203.2	217.2
Linear Weight of Uptake, lbs/ft	266.1	291.5	314.9	336.6
Length of Engine, ft	20.0	24.2	26.6	29.0
Width of Engine, ft	11.0	11.0	11.0	11.0
Height of Engine, ft	12.3	14.9	14.9	14.9
Area Cross-section of Intake, ft^2	10.1	10.1	10.1	10.1
Area Cross-section of Uptakes, ft^2	11.0	11.0	11.0	13.1
Acquisition Cost, million 1991 $	2.92	3.40	3.38	3.87

Table 8 Summary of PC2.6V Cruise Engines' Specifications

for cruise only operation. As before, the information in the table is used by the engine computer code to define the engine characteristics. The information for each engine is listed in the same order has it is stored in the array called "cruise_engine_specs[14]". It is expected that two PC2.6V cruise diesels will meet the cruise power requirement in a split shaft operation configuration.

The average weight per linear foot and the average cross sectional area per linear foot of ducting were calculated as detailed in appendix B and appendix C, respectively. Once the engine is placed and the length of ducting is known, the total weight and volume for the engine ducting can be calculated by multiplying by the known duct length. The acquisition cost for the PC2.6V engines were quoted by the manufacturer [15]. The diesel costs figures include the acquisition items shown in table 6.

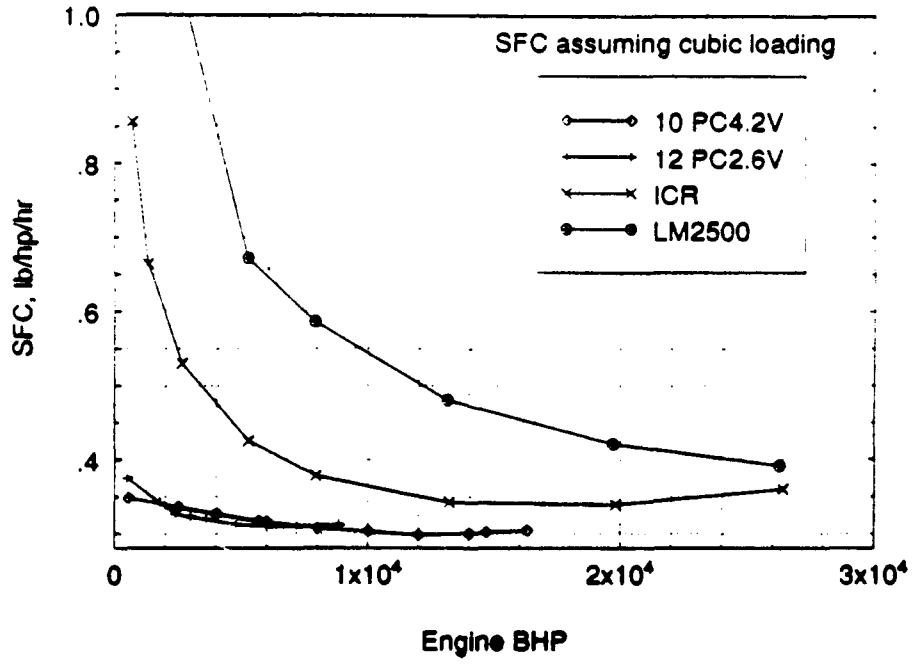

Figure 18 Comparison of Selected Cruise Engines' SFC performance versus BHP

Figure 18 provides a comparison of the SFC for four selected cruise engines as a function of BHP assuming cubic engine loading. It was assumed that the required average endurance power for cruise was 16,000 BHP. This power requirement can be

met by one 10 cylinder PC4.2V or two 12 cylinder PC2.6V diesel engines. It can be seen that at the assumed 16,000 cruise BHP, the ICR engine's SFC performance is very competitive with the diesels.

Chapter 4

The Electric Plant Generator Set Considerations

4.0 Overview

One of the functions of the engine computer model is to calculate the total fuel weight for the ship. The total required ship fuel weight is the sum of the propulsion plant endurance fuel weight and the electric plant endurance fuel weight. The specific details for performing the calculations can be found in NAVSEA Design Standard 200-1 [16] and are described in appendix D. Since the electric plant endurance fuel weight has to be calculated and since more than one type of generator set is available for consideration, the performance, weight, size, and cost must be known for the electric plant generator set options.

4.1 The DDG Electric Plant Options

For the DDG ship type, only two options for the electric plant generator sets were considered; the Allison 501-K34 SSGTG and the Propulsion Derived Ship Service generator (PDSS). Table 9 summarizes the specifications for each of the units. The linear average weights and linear average cross sectional areas for the intake and uptake of the 501-K34 were calculated as per appendix B and C. The PDSS has no intake or uptakes. The other information for the 501-K34 and the PDSS came from Hultgren[6]. The acquisition cost were provided by NAVSEA [14].

It is assumed that the DDG requires three 2500KW generators. The engine computer model allows for mixing PDSS and 501-K34 generator sets. However, as described in chapter 2, at least one 501-K34 gas turbine must be installed to meet the bleed air requirements. The average 24 hour electric load for the DDG is 2525KW [17].

Figure 19 shows the performance of the 501-K34 gas turbine with and without bleed air extraction and also shows the SFC performance for the two gas turbines that

	Allison 501-K34 Gen. Set		PDSS Gen Set Generator	PDSS Gen Set Gear	PDSS Gen. Set Converter
Generator Set Rating, KW	2500		2500	2500	2500
Weight of Gen Set, lbs	60,259		9000	15000	10,100
Linear Weight of Intake, lbs/ft	358.6		0	0	0
Linear Weight of Uptake, lbs/ft	579.4		0	0	0
Length of Gen Set, ft	28.0		7.0	9.0	8.0
Width of Gen Set, ft	9.0		4.0	9.0	4.0
Height of Gen Set, ft	8.0		4.0	4.5	7.5
Area Cross-section of Intake, ft^2	13.0		0	0	0
Area Cross-section of Uptakes, ft^2	13.0		0	0	0
Acquisition Cost, million 1991 $	2.3		1.0	1.0	1.5

Table 9 Summary of Specifications for DDG Electric Plant Generator Set options

would be used to drive a PDSS unit. The engine SFC model is discussed in appendix A. A simple example demonstrates the SFC performance savings provided by PDSS.

Assume that the average endurance power is 15,000 BHP and is being provided by one propulsion gas turbine. Also assume bleed air is required from one SSGTG and that the electric load of 2525KW is equally split between two generator units. From figure 19, the SFC for the 501-K34 supplying the required bleed air is 0.9 lb/hp/hr. If the second generator unit is a 501-K34 with no bleed air, its SFC is 0.8 lb/hp/hr. If the

second unit is a PDSS driven off of an ICR engine, its SFC would be 0.35 lb/hp/hr. If the second unit is a PDSS driven off of an LM-2500 unit, its SFC would be 0.45 lb/hp/hr.

Therefore, for 1260 KW being provided by the second generating unit; the 501-K34 with no bleed air would burn 1524 lb/hr of fuel. The PDSS driven off of the LM-2500 would only burn one half of that or 762 lb/hr. The PDSS driven off of the ICR would burn 592 lb/hr.

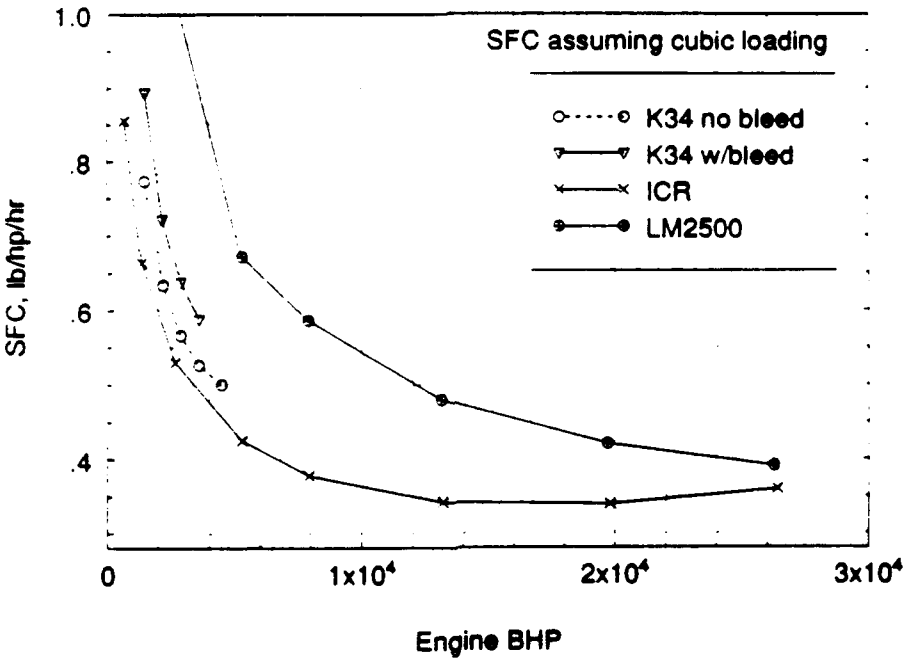

Figure 19 SFC Performance for the DDG Generator Set Options

4.2 The LX Electric Plant Options

For the proposed LX ship type, three options for the electric plant generator sets were considered; the Allison 501-K34 SSGTG, the Caterpillar 3612 Diesel generator, and the PDSS. Table 10 summarizes the specifications for each of the units. The linear average weights and linear average cross sectional areas for the intake and uptakes of the 501-K34 and the Cat 3612 were calculated as per appendix B and C. The PDSS has no intake or uptakes. The information for the 501-K34 and the PDSS came from Hultgren[6]. The information for the Cat 3612 came from the manufacturer [23]. The acquisition cost were provided by a manufacturer representative [27].

The LX is still in the conceptual stage, however the projected average 24 hour electric load is 3190 KW and the projected worst case loading is 7000KW [14]. Since standard practice is to assume one generator is not available, it will be assumed that four 2500KW generators will be required. The engine computer model will allow for mixing PDSS only with 501-K34 generator sets. It is asssumed that the LX will have no bleed air requirements. The Cat 3612 is assumed to have an SFC of 0.353 lb/hp/hr [14].

	Allison 501-K34 Gen. Set	CAT 3612 Gen. Set		PDSS Gen set Generator	PDSS Gen Set Gear	PDSS Gen. Set Converter
Generator Set Rating, KW	2500	3300		2500	2500	2500
Weight of Gen Set, lbs	60,259	102,000		9000	15000	10,100
Linear Weight of Intake, lbs/ft	358.6	115.8		0	0	0
Linear Weight of Uptake, lbs/ft	579.4	179.5		0	0	0
Length of Gen Set, ft	28.0	29.6		7.0	9.0	8.0
Width of Gen Set, ft	9.0	6.8		4.0	9.0	4.0
Height of Gen Set, ft	8.0	10.8		4.0	4.5	7.5
Area Cross-section of Intake, ft^2	23.8	8.7		0	0	0
Area Cross-section of Uptakes, ft^2	33.2	14.7		0	0	0
Acquisition Cost, million 1991 $	2.3	1.24		1.0	1.0	1.5

Table 10 Summary of Specifications for LX Electric Plant Generator Set Options

Chapter 5

Overview of the Integrated Propulsion System Computer Code Structure

5.0 Overview

The integrated propulsion system computer code, written in the computer language C, will determine the fuel performance, weight, volume, and costs of user selected propulsion systems for a destroyer type ship and an amphibious type ship. The computer code will consist of various functions developed for the analysis. The functions can be generally categorized. The five basic function categories are input/output, resistance, propulsors, transmissions, and engines. A brief overview of the proposed integrated propulsion system computer code will be provided so that the specific requirements for the engine functions of the computer code will be clear. A simple flow chart will be presented.

5.1 The Integrated Codes Basic Structure

Figure 20 presents a simple flow chart to describe the basic structure of the proposed integrated propulsions system computer code. The first task for the code is to perform the input operation. The input operation provides an interactive user interface that guides the user through the selection of the individual components for the propulsion system. A detailed description of this operation will be described in the next chapter.

Once the propulsion system has been defined, the computer code determines the maximum power that is available from the selected types and selected numbers of propulsions engines. This information is required early in the program in order for the resistance function to determine the limiting ship speed given the available power.

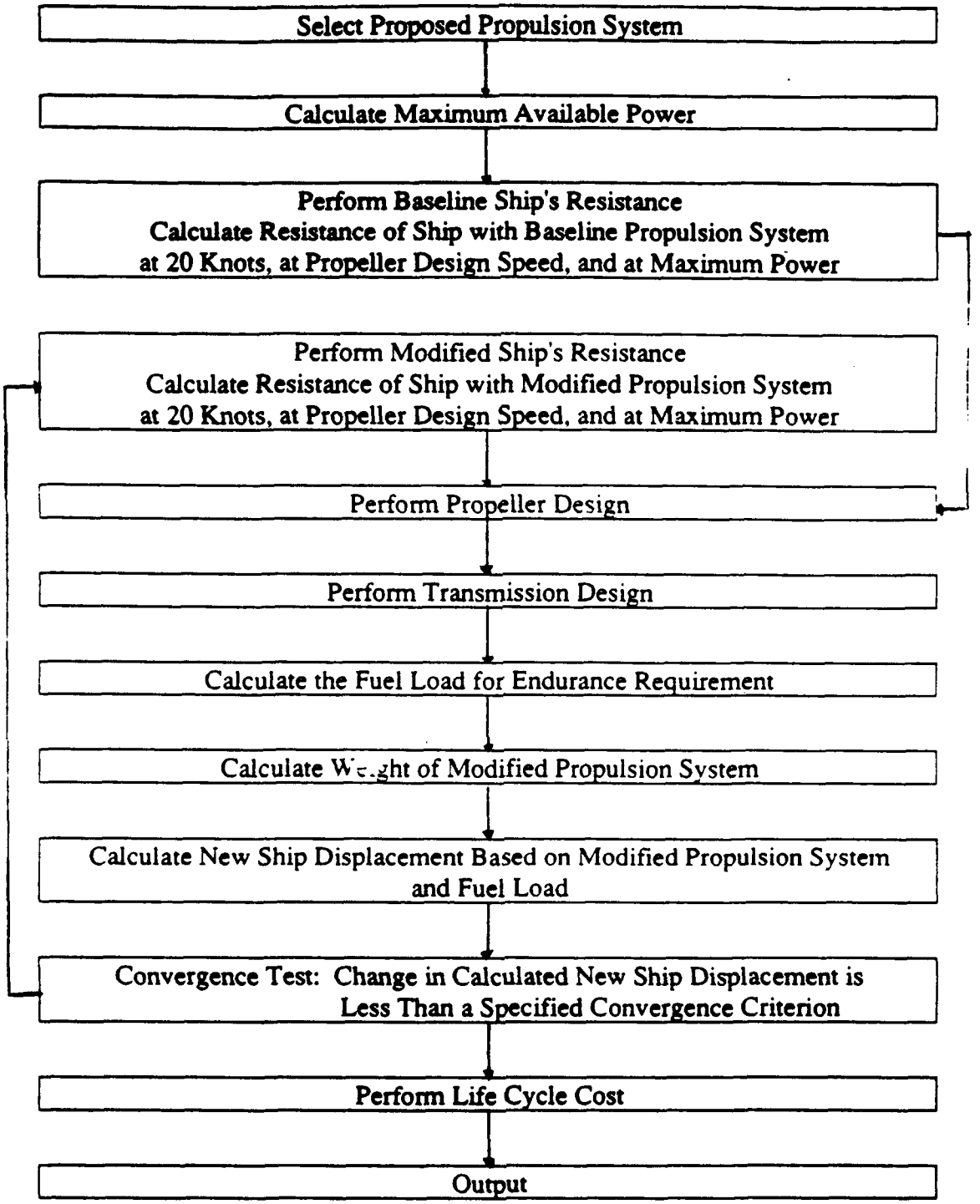

Figure 20 Proposed Integrated Propulsion System Computer Code Structure

The baseline ship's resistance function applies specific ship characteristics to determine the hull powering requirements. This information includes the baseline ship's displacement, length, beam, draft, and prismatic coefficient. The hull resistance is calculated at the endurance speed of 20 Knots, at the propeller cavitation limited design speed, and at the speed corresponding to the maximum available power. With the power requirements defined at the described conditions, the computer code steps through the design of the propeller, the design of the transmission, and the calculation of the fuel load. The weight of the fuel load and the weight of the new propulsion system components are summed to develop the new ship displacement. With a new ship displacement, the hull powering requirements must be recalculated and the described design loop repeated. The loop continues until a specified convergence criterion is satisfied.

After convergence of the resistance loop, the Life Cycle Cost (LCC) is calculated for the modified ship and propulsion system. The LCC calculation will determine the annual fuel consumed given an assumed operating profile.

The final task of the computer code is to output the weight, volume, performance, acquisition cost, and life cycle cost for the user selected propulsion system has applied to the specific ship type.

Chapter 6

Development of the Engine Functions for the Integrated Propulsion Systems Computer Code

6.0 Overview

This chapter provides a discussion of the engine related functions written to support the integrated propulsion systems computer code. The individual engine functions are combined to produce a stand-alone engine's program. Figure 21 presents a simple flow chart showing the basic structure of the stand-alone engine's program. The program employs user selected values for the required endurance EHP, Quasi-Propulsive Coefficient (QPC), propulsor RPM, transmission ratio, and transmission efficiency. These user input "dummy" values or "stub" functions will eventually be replaced by the appropriate functions in the integrated propulsion systems program.

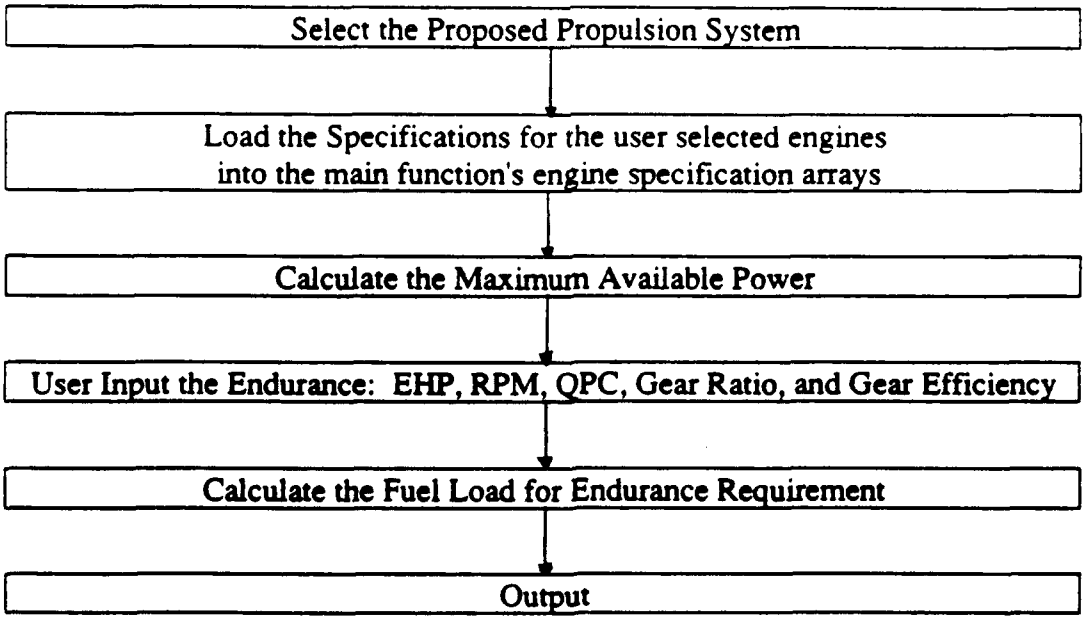

Figure 21 The Engine's Program Computer Code Structure

The stand-alone engine's program outputs the fuel load, the weight for all engines, the volume for all engines, the operating performance for all engines, and the acquisition cost for all engines.

6.1 The Engine's Program Computer Code

The individual functions that make-up the stand-alone engine's program are shown in table 11. The functions are shown in their function prototype format which shows each function's return type and each function's arguments. A discussion for each function will follow. The engine program functions are shown in appendix E.

int user_interface(void)
void mechanical_drive(void)
void electric_drive(void)
void mech_elec_hybrid_drive(void)
void multiple_waterjet(void)
void main_engine_specs(boost_engine_specs)
void endurance_engine_specs(boost_engine_specs, cruise_engine_specs)
double fuel_load(ehp_cruise, qpc_cruise, xmission_eff_cruise, rpm_prop_cruise, gear ratio, cruise_engine_specs, main_engine_max_pwr)
double engine_sfc(bhp, rpm, number_cylinders, eng_type)
double lm2500_map(engine_bhp, engine_rpm)
double pc42_sfc_map(engine_bhp, engine_rpm, number_cylinders)
double pc26_sfc_map(engine_bhp, engine_rpm, number_cylinders)

Table 11 Listing of the Engine Functions

The first five functions shown in table 11 perform the input operation. They provide an interactive user interface that guides the user through the selection of the individual components for the propulsion system. The user selected choices are stored in a 7 by 2 two dimensional array called plant_map which is shown in table 12. Each cell of the array contains information about the propulsion system. The individual cells, shown in table 12, display all the possible values for a given selection. However in some cases, depending on the selected drive system, not all of the possible values will available. The functions incorporate bounds checking to ensure that the user only inputs one of the accepted values. The plant_map array is declared has a global array so that any function can access the plant_map array to obtain information about the proposed propulsion system.

The user_interface function initiates the input process. However, the logic required to step through the entire input process depends on the type of drive system. Therefore the user_interface function calls one of the next four functions to complete the input process. At the end of the selection process, the user_interface function provides a summary of the selections and offers the choices of running the program, re-selecting the propulsion system, or quitting the program.

If the propulsion system is mechanical drive, the mechanical_drive function steps through the input process. The function's logic allows for two or three propulsion shafts. If three propulsion shafts are selected, the cross-connect gear option is not allowed. The user selects either the LM-2500, the ICR, or the 16 cylinder PC4.2V as the boost engine. If the LM-2500 is selected, then the user is given the opportunity to select a different type of engine for cruise. The cruise engine options are the ICR, the 10 cylinder PC4.2V, or the 10 cylinder PC2.6V. The fuel_load function, called later in the program, will check the power ratings on the cruise diesels and if necessary, it will automatically increase the number of cylinders to meet the endurance power requirement. The user then selects the number of cruise engines and the number of boost engines that will be

	PLANT_MAP [ ][0] CRUISE CONDITION	PLANT_MAP [ ][1] BOOST CONDITION
ROW[0]	NUMBER OF PROPULSORS USED FOR CRUISE: 1 = ONE 2 = TWO 3 = THREE 4 = FOUR 6 = SIX	NUMBER OF PROPULSORS USED FOR BOOST: 2 = TWO 3 = THREE 4 = FOUR 6 = SIX 8 = EIGHT
ROW[1]	PROPULSOR DESIGNATION CODE: 1 = FPP 2 = CRP 3 = CONTRA-ROTATING 4 = PRESWIRL STATOR 5 = DUCTED FPP 6 = DUCTED CRP 7 = DUCTED CONTRA 8 = DUCTED PRESWIRL 9 = WATERJET	HYBRID DRIVE XMISSION TYPE: 0 = NONE 1 = ELECTRIC TO MECH
ROW[2]	TRANSMISSION DESIGNATION CODE: 1 = MECHANICAL DRIVE 2 = ELECTRIC DRIVE 3 = HYBRID 4 = MULTIPLE WATERJET	TRANSMISSION TYPE CODE: 0 = NONE 1 = EPICYCLIC 2 = LTDR 3 = LTDR w/REVERSING 4 = LTDR W/CROSSCONNECT
ROW[3]	CRUISE ENGINE TYPE CODE: 0 = NONE 1 = ICR GAS TURBINE 2 = PC 4.2 DIESEL 3 = PC 2.6 DIESEL	BOOST ENGINE TYPE CODE: 0 = NONE 1 = LM-2500 GAS TURBINE 2 = ICR GAS TURBINE 3 = PC 4.2 DIESEL 4 = LM-2500 POWER PAK 5 = ALLISON 571-KF
ROW[4]	TOTAL NUMBER OF THE DESIGNATED BOOST ENGINE TYPE USED FOR CRUISE OPERATION: 0 = NONE 3 = THREE 1 = ONE 4 = FOUR 2 = TWO 6 = SIX	TOTAL NUMBER OF THE DESIGNATED BOOST ENGINE TYPE USED FOR MAXIMUM BOOST OPERATION: 1 = ONE 4 = FOUR 2 = TWO 6 = SIX 3 = THREE 8 = EIGHT
ROW[5]	TOTAL NUMBER OF THE DESIGNATED CRUISE ENGINE TYPE USED FOR CRUISE OPERATION: 0 = NONE 3 = THREE 1 = ONE 4 = FOUR 2 = TWO	TOTAL NUMBER OF THE DESIGNATED CRUISE ENGINE TYPE USED FOR MAXIMUM BOOST OPERATION: 0 = NONE 3 = THREE 1 = ONE 4 = FOUR 2 = TWO
ROW[6]	PDSS WILL BE USED FLAG: 0 = NO 1 = YES	THE NUMBER OF PDSS THAT WILL BE USED: 0 = NONE 4 = FOUR 1 = ONE 5 = SIX 2 = TWO 6 = EIGHT 3 = THREE

Table 12 The plant_map[7][2] Integer array.

used for cruise and boost operation. The program allows the ICR cruise type engine to be used for boost operation, but does not allow the diesel cruise type engines to be used for boost operation. Finally, if all the engine types are all gas turbine, the user is given the option to select PDSS driven off of the gas turbine output shafts.

If the propulsion system is electric drive, the electric_drive function steps through the input process. This function's logic only allows for two propulsion shafts. The user selects either an epicylic gear or direct drive between the motor and propulsor . Only one engine type is allowed for all operating regimes and the choices are LM-2500 or ICR. The user selects the number of engines that will be used for cruise operation and the number of engines that will be used for boost operation. The user is given the option to select PDSS driven off of the gas turbine output shafts.

If the propulsion system includes diesel electric for cruise, the mech_elec_hybrid_drive function steps through the input process. This function's logic only allows for two propulsion shafts and assumes that one LM-2500 mechanically drives each gear. The user has the option to also allow for one additional LM-2500 driving a generator that drives motors connected to each shaft's gear. The program assumes that the cruise diesel is a 10 cylinder PC4.2V. PDSS is not allowed.

A multiple waterjet propulsion system is allowed. The program assumes the Allison 571-KF will be the engine and that each engine is coupled to a waterjet. The user selects the number of engines used for cruise and boost. A PDSS option is provided.

With the proposed propulsion system stored in the plant_map array, the specifications for the selected engine types are required. The function call main_engine_specs will return the specifications for the boost engine. The specifications data for all the boost engine types are stored in a 14 by 5 two dimensional array called boost_engine_options. Table 4 of chapter 3 shows the contents of each cell of that array.

Each column of the array corresponds to one of the boost engine types. Each row of the array contain the specifications for engine ratings, weight, volume, and acquisition cost. The function uses the engine type code to select the column corresponding to the selected boost engine type. The boost engine specifications are returned to the main function in a 14 cell one dimensional array called boost_engine_specs.

The function call endurance_engine_specs will return the specifications for the cruise engine. The cruise engine specifications are returned to the main function in a 14 cell one dimensional array called cruise_engine_specs. The specifications data for the cruise engine types are stored in a 14 by 3 two dimensional array called cruise_engine_options. Each column of the array corresponds to one of the cruise engine types: ICR, 10 cylinder PC4.2V, or 10 cylinder PC2.6V. Each row of the array contains the specifications for engine ratings, weight, volume, and acquisition cost. The function uses the engine type code to select the column corresponding to the selected cruise engine type. If no cruise engine type has been selected, then the boost engine must also be used for cruise. In this case, the array called cruise_engine_specs is loaded with the boost_engine_specs array data.

The maximum available engine power used to determine the maximum speed is calculated by adding the boost engine rated power multiplied by the number of boost engines used for boost plus the cruise engine rated power multiplied by the number of cruise engines used for boost.

The next step in the engines program is to input the endurance EHP, the propulsor and transmission power efficiencies, and the RPM requirements. For the integrated propulsion systems program, this information will come from the resistance, propeller, and transmission functions. This information is passed to the fuel_load function.

The function called fuel_load calculates the required fuel weight to meet the endurance requirements for the ship. The details of the standards and requirements for the calculation are provided in appendix D. The function has two major branches that

provide separate logic for the Propulsion Derived Ship Service Generator (PDSS) not installed option and the PDSS installed option.

The PDSS not installed branch assumes that the cruise power required is equally split between each on-line cruise engine. Additionally, the average 24 hour electric load is assumed to be split equally between two generator sets. The average endurance BHP is calculated for each engine. As described in appendix D, the average endurance BHP applies a 10 percent margin to the power requirement. The function checks and ensures that the total installed cruise engine power will meet the average endurance BHP requirements.

As mentioned earlier, if the cruise engine type is selected by the user has either a PC4.2V diesel or a PC2.6V diesel, the program assumes that the 10 cylinder diesel size is adequately sized to meet the power demand. If the 10 cylinder diesel is not large enough to meet the average endurance power requirement per cruise engine, the function automatically increases the diesel to the next larger size. This process is repeated until either the diesel engine becomes large enough or the available diesel engine sizes are exhausted. This procedure should result in the optimum cruise diesel engine size. If a large enough diesel is found, then the cruise_engine_specs array is loaded with the new cruise engine specifications data. If the required power can not be met, an error statement is printed and the program is terminated.

With the required engine BHP and RPM for a specified engine type, the function called engine_sfc is used to calculate and return the uncorrected SFC. The uncorrected SFC is the SFC reported by the manufacturer's data at the specified engine BHP and RPM. Appendix D outlines the corrections to the manufacturer's SFC value required for Navy standard endurance fuel weight calculations. Appendix A provides a detailed explanation for the modelling of the individual engine's SFC performance. It should be noted that the individual engine SFC functions are written so has to only require the engine BHP, RPM, and the number of cylinders (in the case of the diesels). This makes

the individual engine SFC functions very portable, which will facilitate the Life Cycle Cost calculations.

Once the uncorrected engine SFC is returned to the fuel_load function, the logic applied to determine the propulsion fuel weight is straight forward as outlined in appendix D.

The electric fuel weight is also calculated as outlined in appendix D. The DDG ship type will require bleed air from the Ship Service Gas Turbine Generator (SSGTG). The LX ship type will not require any bleed air. The percentage of time that bleed air will be extracted from the SSGTG is stored in the preprocessor #define variable called ENDUR_PCT_BLD. This predefined variable is set at 0.5 or 50 percent of the endurance time with bleed. The electric fuel weight for any percentage of bleed time can easily calculated by changing this predefined variable in the source code. The average electric SFC is equal to the bleed SFC multiplied by ENDUR_PCT_BLD plus the no-bleed SFC multiplied by one minus ENDUR_PCT_BLD.

The PDSS installed branch assumes that the cruise power required is equally split between each on-line cruise engine. However, the logic only allows the PDSS option if all the engine types are gas turbine. For most of the propulsion systems to be considered in the integrated systems study, one LM-2500 or one ICR engine will meet the cruise power requirement. In this case, the average 24 hour electric load is assumed to be split equally between one SSGTG and one PDSS generator set. Additionally, the SSGTG will provide bleed air has determined by the ENDUR_PCT_BLD variable's value.

It may be possible that for some of the propulsion systems to be considered in the integrated systems study, that more than one PDSS unit will be on-line during cruise. In this case, the average 24 hour electric load is assumed to be split equally between one SSGTG providing bleed air and the total number of on-line PDSS generator sets. The SSGTG will provide bleed air has determined by the ENDUR_PCT_BLD variable's

value. For the endurance time that does not require bleed air, the electric load will be assumed to be only split between the on-line PDSS units.

The function checks and ensures that the total installed cruise engine power will meet the average endurance BHP requirements plus the additional PDSS load requirements. If the cruise engine selections will not meet this requirement, an error statement is printed and the program is terminated. The rest of the calculation are the same as for the no PDSS option.

The fuel_load functions adds the propulsion fuel weight and the electric fuel weight and returns the endurance fuel weight load to the main function.

The last step is to output the results of the program. The program outputs the significant data used in the fuel calculations and the fuel calculation results. Additionally, the cruise engine's specifications and the boost engine's specifications are displayed. Appendix F shows the specific format of the output for three variations: cruise with no PDSS, cruise with one PDSS, and cruise with more than one PDSS.

Table 13 shows all the preprocessor #define statements used in the program.

INLET_LOSS 4.0 inches H_2O
EXHAUST_LOSS 6.0 inches H_2O
HUMIDITY 116.2 grains
RANGE 6000 Nautical Miles
ENDUR_SPD 20.0 Knots
ENDUR_PCT_BLD 0.50
AVG_ELEC_LOAD 2525.0 KW
TAIL_PIPE_ALLOWANCE 1.02

Table 13 Listing of All Preprocessor #define Variables

These predefined variable can easily be changed in the source code to change the value of the information they represent. The source code shown in appendix E contains the values shown in table 13. These values apply to the DDG. The source codes for the LX will require that some of these predefined variables are changed. Additionally, if diesel generating sets will be used for the LX, the LX source code will require their SFC performance to be included in the fuel_load function. The present source code assumes only Allison 501-K34 SSGTG's are used.

Chapter 7

Summary

Five types of marine propulsion engines have been examined and compared. They include an LM-2500 marine gas turbine, an Intercooled Recuperative (ICR) marine gas turbine, a series of Colt-Pielstick PC4.2V medium speed diesels, a series of Colt-Pielstick PC2.6V medium speed diesels, and an Allison 571-KF marine gas turbine module power pak.

To facilitate an integrated propulsion systems study, an engine's computer model has been written that calculates the engine weight, volume, fuel consumption, and acquisition cost. Given user input for propulsor and transmission performance, the engine code will also calculate the required endurance fuel load in accordance with Navy standards.

The Engine's computer code allows the user to employ different engine types for cruise and boost operating regimes. The model ensures that the engines are operated within their horsepower and RPM ratings and splits the propulsion load evenly when multiple engines are in use.

The engine's computer code will be integrated into a complete propulsion systems computer code. This will facilitate the analysis of various propulsion alternatives for Navy ships.

REFERENCES

[1] Simmons, L. D.,"Naval Propulsion Systems Phase 1: Survey of Alternative Technologies", IDA Paper P-2532, 1991

[2] Blank, D. A. and Bock, A. J., and Richardson, D. J.: *Introduction to Naval Engineering*, Annapolis : Naval Institute Press, 1985

[3] Richard E. and Gordon J. Van Wylen, *Introduction to Thermodynamics: Classical and Statistical*, Second Edition, John Wiley & Sons, New York, 1982

[4] LM-2500 Marine Gas Turbine Performance Data, MID-TD-2500-8, 1991

[5] U.S. Navy DD-963 Propulsion Plant Manual, S9234-AD-GTP-010, 1989

[6] Hultgren, K. J., " VSCF Cycloconverter Ships Service Power Equipment", Navy Engineers Journal, Jan, 1992

[7] ICR Engines, Solicitation N00024-91-PR-52146 , May 1991

[8] Reid, M.R., "Integrated Electric Drive Program Overview", overhead presentation material, March 1990

[9] Allison Gas Turbines Specification 905A, Model 571-K, October 1989

[10] Allison Gas Turbine Marine Propulsion Systems Brochure GTP 5291, 1984

[11] Fairbanks Morse Engine Division, Co. Sales Engineering Brochure, 1990

[12] Fairbanks Morse Engine Division, Co. Sales Engineering Data F1232-3, June 1990

[13] Stevens, K., Stewart Stevenson Co. *Sales Representative, phone* conversation, April 1992

[14] Brown, A., Goddard, C., and Doerry, N., NAVSEA O5Z, Advanced Surface Machinery System Program Office, various phone conversations and faxes, March and April, 1992

[15] Burnette, J., Fairbanks Morse Engine Division, Co. Sales Representative, phone conversation, April 1992

[16] NAVSEA Design Data Sheet 200-1, DDS 200-1, March 1982

[17] DDG 51 ASSET File, DTNRC, December 1988

[18] Allison Gas Turbine Performance Summary, Allison Gas Turbine Division, 1990

[19] Fairbanks Morse Engine Division, Co. Sales Engineering Data F1304-1, 1990

[20] Fairbanks Morse Engine Division, Co. Sales Engineering Data F1102-8, 1990

[21] Fairbanks Morse Engine Division, Co. Sales Engineering Data F1302-4A, 1984

[22] Halsey, J. E., Russom, D., "Design, Development, Testing, and Operational Experience of the Allison Model AG9130 Ship Service Gas Turbine Generator Set", ASME paper number 91-GT-46, 1991

[23] Caterpillar Inc., Co. Sales Engineering Data LEHW0569, April 1990

[24] ASSET Manual BCS 40530-11, Weight Module,

[25] Fairbanks Morse Engine Division, Co. Sales Engineering Data E3210-2, June 1980

[26] Fairbanks Morse Engine Division, Co. Sales Engineering Data E3200-3, Aug 1979

[27] Jarmon, R., Caterpillar Inc. Sales Representative, phone conversation, April 1992

Appendix A

Specific Fuel Consumption Modelling

Overview

The Specific Fuel Consumption (SFC), in units of lb/HP/hr, is modelled for each of the five discussed propulsion engine types and for the electric plant generator set engines using the standard conditions described in chapter 2. The manufacturers provided the SFC data for the engines with the exceptions of the ICR engine and the Allison 501-K34. The ICR engine SFC data is as specified in the Navy solicitation [7] for the ICR engine. The Allison 501-K34 SFC data was taken from a paper by Halsey[22]. For the ICR engine, the SFC data represents the maximum allowable SFC at the specified engine operating points. For all of the other engines, the SFC data has a manufacturer declared +/- 3 percent tolerance to allow for variations in individual engine performance.

LM-2500 SFC Model

The LM-2500 SFC data, provided by the manufacturer [4], is specified for discrete engine BHPs and RPMs. Figure 1 shows the specific SFC data points that were used to map the engine's SFC performance. It should be noted that other pertinent engine performance data is provided at these same operating points.

For the LM-2500, the engine exhaust parameters were also mapped at these data points to allow for the future consideration of bottoming cycles applied to the LM-2500 engine exhaust. The exhaust parameters mapped include: exhaust duct discharge total temperature (T8), exhaust duct discharge total pressure (P8), exhaust duct discharge flow (W8), and exhaust duct discharge specific heat (CP8).

The above mentioned LM-2500 engine performance parameters were incorporated into the engine portion of the integrated propulsion system computer code. The code's function call "lm2500_map (engine_bhp, engine_rpm)" will return the described engine performance parameters.

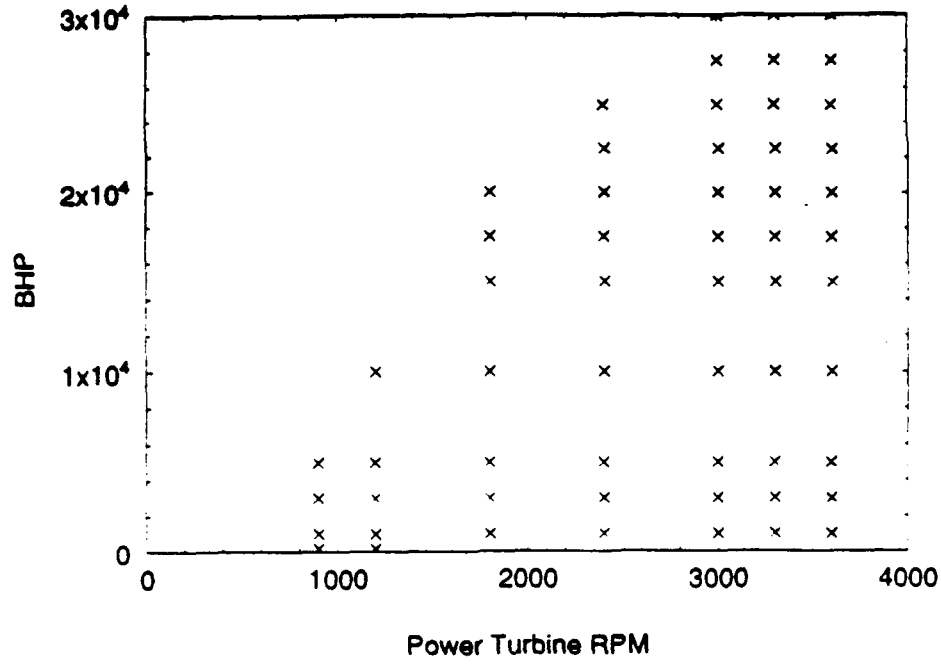

Figure 1 LM-2500 SFC Data Points

The lm2500_map function applies linear interpolation between the data points, as prescribed by the manufacturer, to determine the engine performance parameters for a given engine BHP and RPM. The function includes bounds checking to ensure that the specified engine BHP and RPM are within the limits of the data points shown in figure 1.

The engine performance parameters, as determined by the manufacturer's data at the specified data points, assume that there are no inlet losses and no exhaust losses and that the relative humidity is at zero percent. The following correction factors determined by equations (1) through (15) must be multiplied by the uncorrected engine parameter to correct for inlet and exhaust losses (in inches of H_2O) and for humidity (measured in grains) values greater than zero percent.

$$\text{inlet\_sfc\_factor} = 0.001125 * \text{INLET\_LOSS} + 1 \quad (1)$$

$$\text{exhaust\_sfc\_factor} = 0.001295 * \text{EXHAUST\_LOSS} + 1 \quad (2)$$

$$\text{humidity\_sfc\_factor} = 0.0000387 * \text{HUMIDITY} + 1 \quad (3)$$

$$\text{inlet\_T8\_factor} = 0.001875 * \text{INLET\_LOSS} + 1 \quad (4)$$

exhaust_T8_factor = 0.00098 * EXHAUST_LOSS + 1 (5)

humidity_T8_factor = -0.0000057 * HUMIDITY +1 (6)

inlet_W8_factor = -0.001375 * INLET_LOSS + 1 (7)

exhaust_W8_factor = 0.0003636 * EXHAUST_LOSS + 1 (8)

humidity_W8_factor = -0.00005 * HUMIDITY +1 (9)

inlet_P8_factor = 0.0 * INLET_LOSS + 1 (10)

exhaust_P8_factor = 0.00245 * EXHAUST_LOSS + 1 (11)

humidity_P8_factor = 0.0 * HUMIDITY +1 (12)

inlet_CP8_factor = 0.0004 * INLET_LOSS + 1 (13)

exhaust_CP8_factor = 0.002 * EXHAUST_LOSS + 1 (14)

humidity_CP8_factor = 0.000125 * HUMIDITY +1 (15)

ICR SFC Model

The ICR engine SFC data is specified in the Navy solicitation [7] for the ICR engine. The maximum allowable SFC as a function of the percentage of engine BHP is specified for seven engine BHP values. Figure 2 shows the seven specified SFC data points and the curve fit used to describe the ICR engine SFC as a function of the percentage of BHP. The computer software EASYPLOT^{TM} was used to find the least squares fit.

The ICR engine SFC performance function shown in figure 2 was incorporated into the engine portion of the integrated propulsion system computer code. The code's function call "engine_sfc(bhp, rpm, eng_type)" , with eng_type selected for ICR, will return the ICR engine SFC performance parameter.

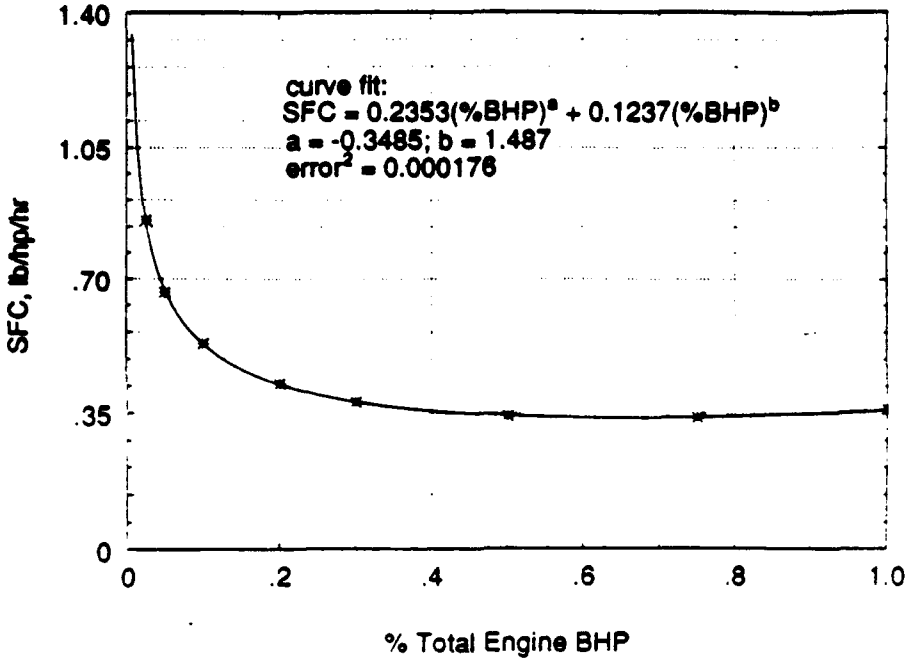

Figure 2 ICR Engine's SFC Performance versus BHP

Allison 571-KF SFC Model

The Allison 571-KF engine SFC data is provided by the manufacturer [18]. The engine SFC as a function of BHP is specified for seven discrete BHP values. Figure 3 shows the seven specified SFC data points and the curve fit used to describe the ICR engine SFC as a function of BHP. The computer software EASYPLOT^{TM} was used to find the least squares fit.

The 571-KF engine SFC performance function shown in figure 3 was incorporated into the engine portion of the integrated propulsion system computer code. The code's function call "engine_sfc(bhp, rpm, eng_type)" , with eng_type selected for 571-KF, will return the Allison 571-KF engine SFC performance parameter.

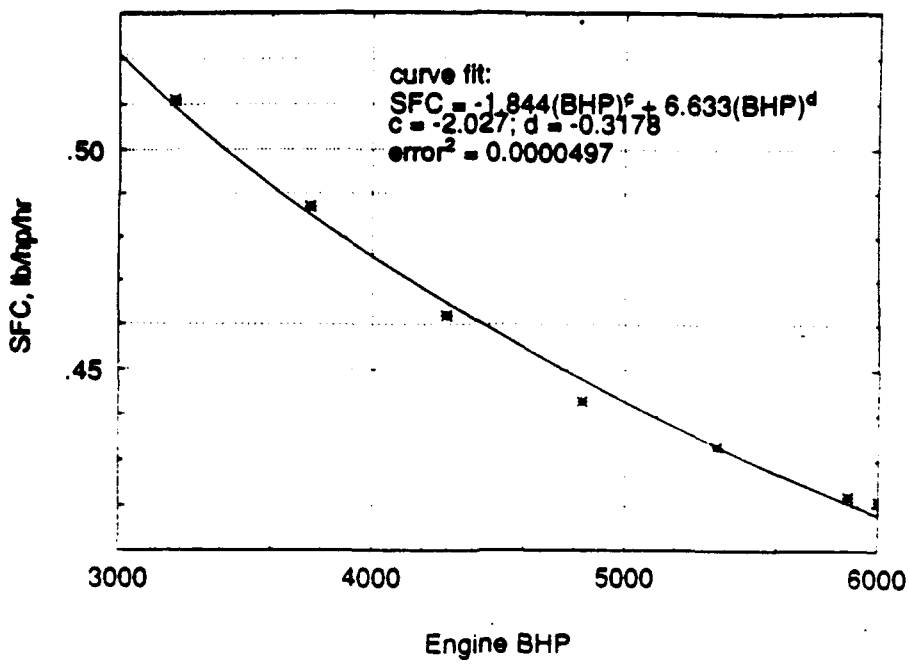

Figure 3 Allison 571-KF Engine's SFC Performance versus BHP

PC4.2V SFC Model

The PC4.2V SFC data, provided by the manufacturer [19], is shown in figure 4. In order to incorporate the graphical SFC data into the engine computer code, the engine SFC values at the discrete engine BHP and RPM data points shown in figure 5 were stored in an array. The engine code's function call "pc42_sfc_map(engine_bhp, engine_rpm, number_cylinders)" performs linear interpolation between the data points and returns the engine SFC parameter. The function includes bounds checking to ensure that the specified engine BHP and RPM are within the limits of the data points shown in figure 5. Additionally, the specified engine BHP and RPM are checked to ensure that they are not outside the fuel rack limitations shown is figure 6. Note that the SFC is determined on a per cylinder basis. Multiply by the number of cylinders to obtain the engine BHP.

For the life cycle cost analysis, the engine SFC for low engine BHP values is required. Figure 6 shows that the limiting BMEP per cylinder is 29 psi which equates to 44.27 BHP at 125 RPM. The SFC values, from figure 4, are not provided for this low

power operating regime. To estimate the SFC values for low cylinder BHP operation, the provided SFC data was extrapolated down to the low cylinder power operation as shown in figure7.

PC2.6V SFC Model

The PC2.6V SFC data, provided by the manufacturer [21], is shown in figure 8. In order to incorporate the graphical SFC data into the engine computer code, the engine SFC values at the discrete engine BHP and RPM data points shown in figure 9 were stored in an array. The engine code's function call "pc26_sfc_map(engine_bhp, engine_rpm, number_cylinders)" performs linear interpolation between the data points and returns the engine SFC parameter. The function includes bounds checking to ensure that the specified engine BHP and RPM are within the limits of the data points shown in figure 9. Note that the SFC is determined on a per cylinder basis. Multiply by the number of cylinders to obtain the engine BHP.

For the life cycle cost analysis, the engine SFC for low engine BHP values is required. The limiting BMEP per cylinder is 29 psi which equates to 25.8 BHP at 200 RPM. The SFC values, from figure 8, are not provide for this low power operating regime. To estimate the SFC values for low cylinder BHP operation, the provided SFC data was extrapolated down to the low cylinder power operation as shown in figure 10.

Allison 501-K34 SFC Model

The Allison 501-K34 SFC data is as specified by Halsey [22]. The engine SFC as a function of BHP, for both the no bleed air and the bleed air extraction conditions, is specified for several discrete BHP values. The computer software EASYPLOT^{TM} was used to find the best least squares fit. A fifth order polynomial provided the best fit. Figure 11 shows the discrete data points and the resulting curve fit. Equation (16) gives the no bleed air extraction SFC and equation (17) gives the bleed air extraction SFC,

both as a function of engine BHP. The engine computer code incorporated these equations for the SFC values of the 501-K34.

$$SFC_{no\text{-}bleed} = 3.12*10^{-19}BHP^5 + 1.332*10^{-16}BHP^4 - 4.17*10^{-11}BHP^3 + 3.06*10^{-7}BHP^2 - 9.14*10^{-4}BHP + 1.58 \quad (16)$$

$$SFC_{bleed} = 2.34*10^{-19}BHP^5 - 5.27*10^{-15}BHP^4 + 1.24*10^{-11}BHP^3 + 1.35*10^{-7}BHP^2 - 7.43*10^{-4}BHP + 1.67 \quad (17)$$

TYPICAL FUEL CONSUMPTION
- Marine Engines - PC4.2/2 - U.S. Units -

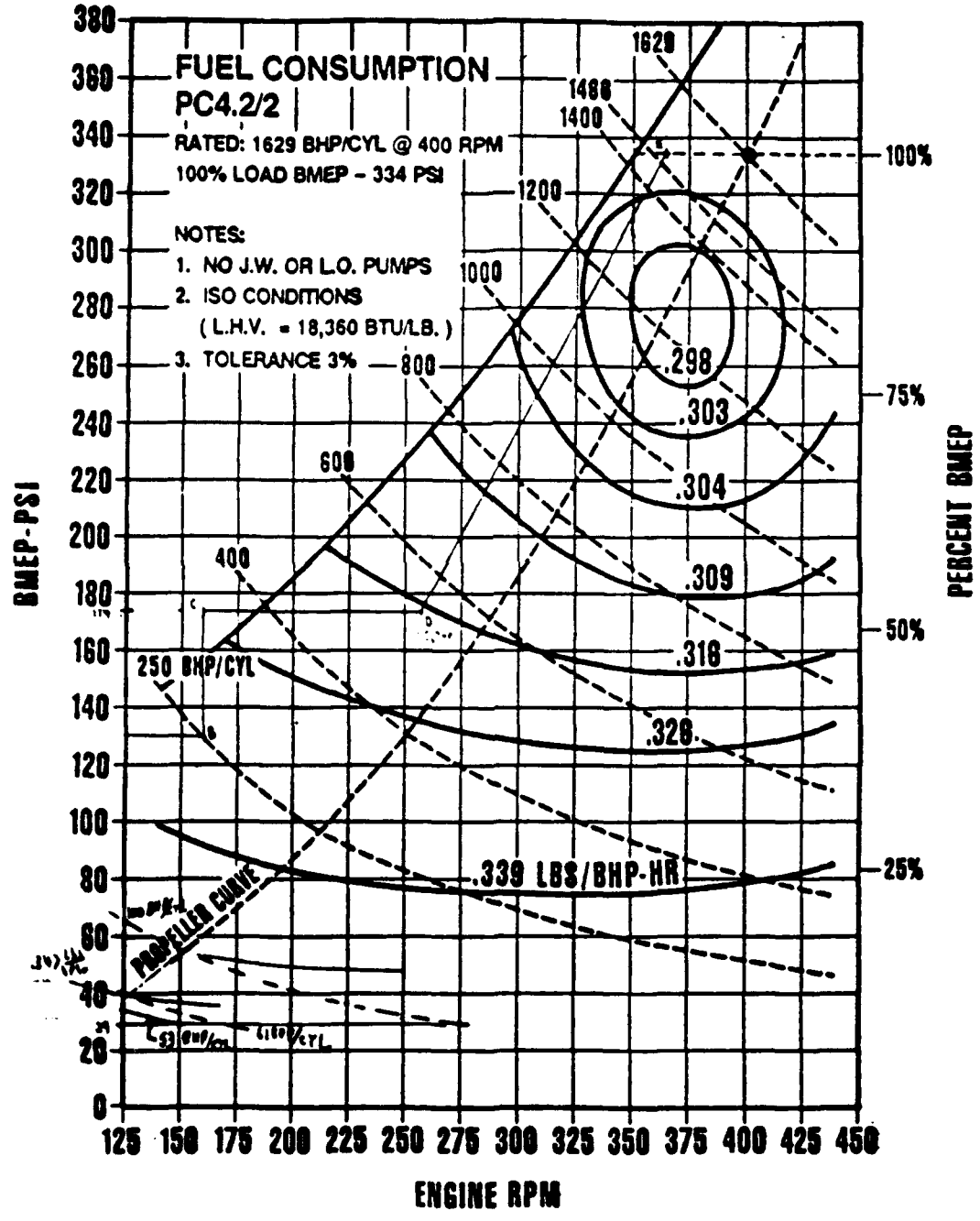

Figure 4 PC4.2V Engine's SFC Performance versus Operating Conditions [19]

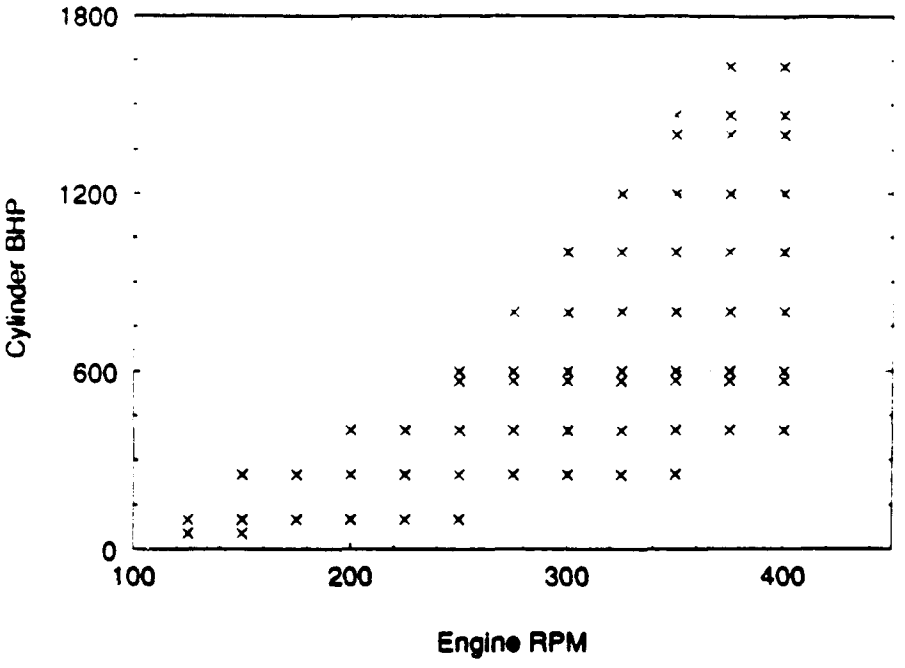

Figure 5 PC4.2V SFC Data Points

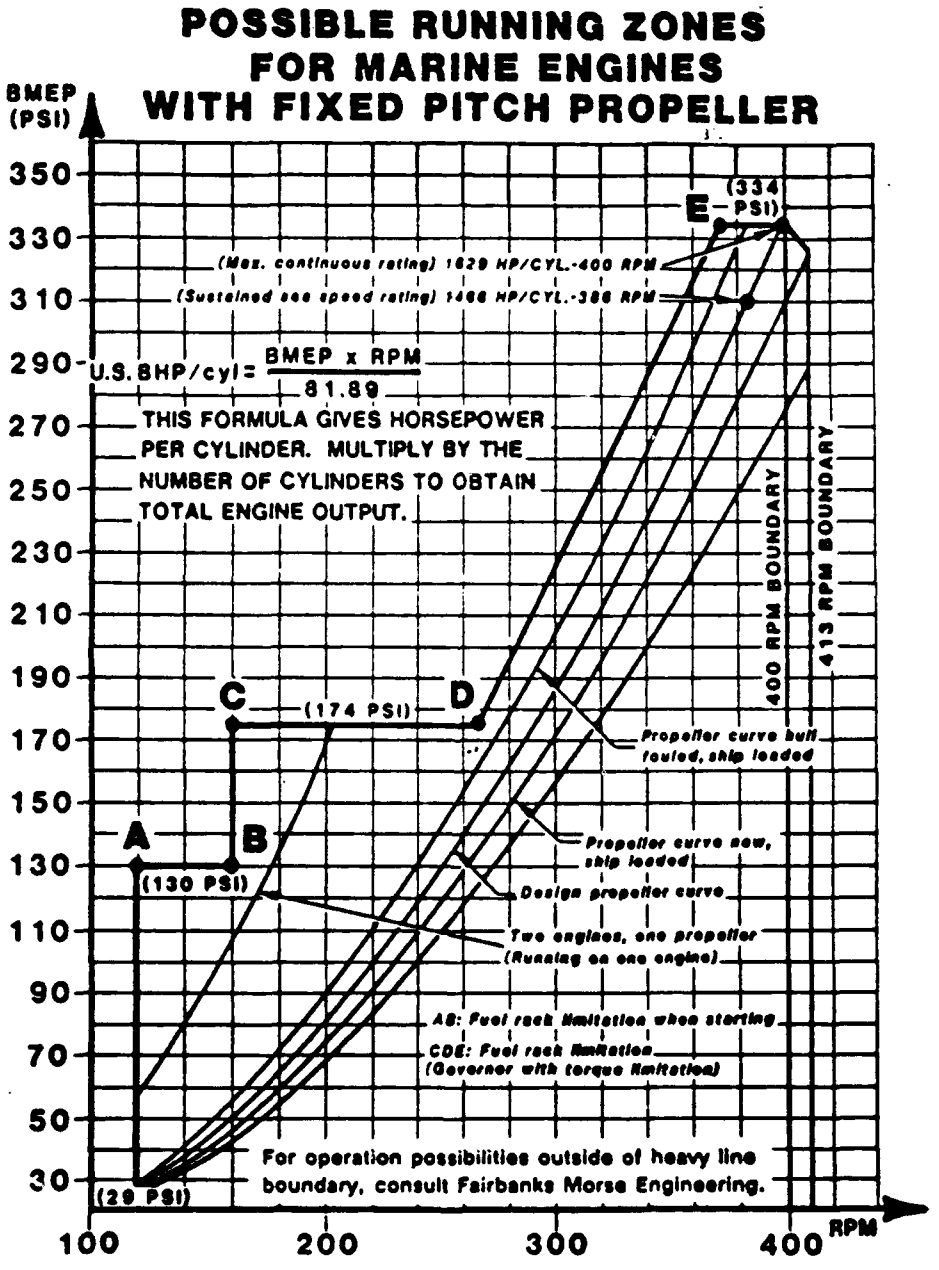

Figure 6 PC4.2V Fuel Rack Limitations [20]

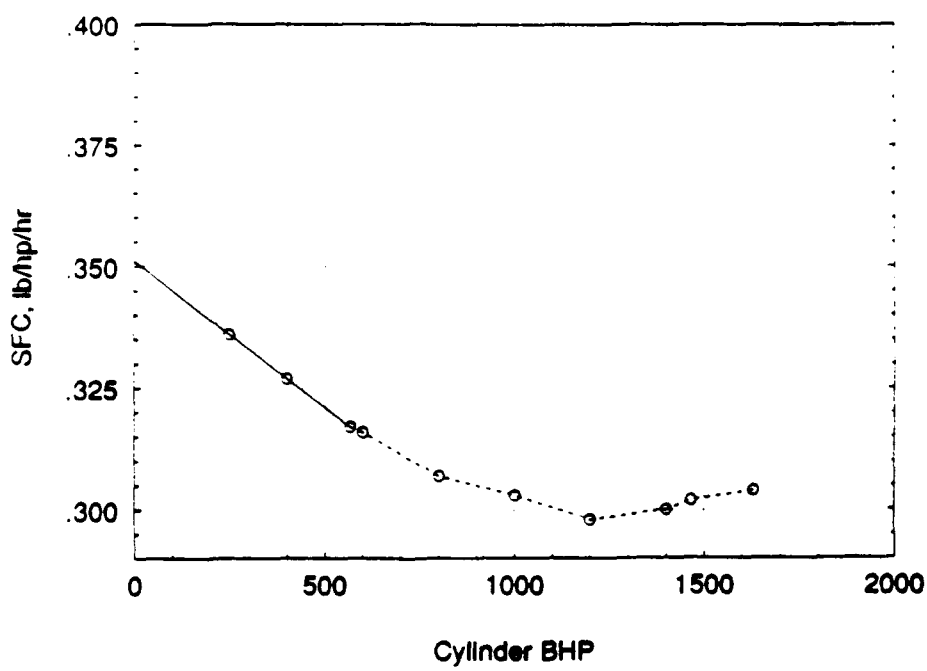

Figure 7 SFC Extrapolation for PC4.2V Low Cylinder Power Operation

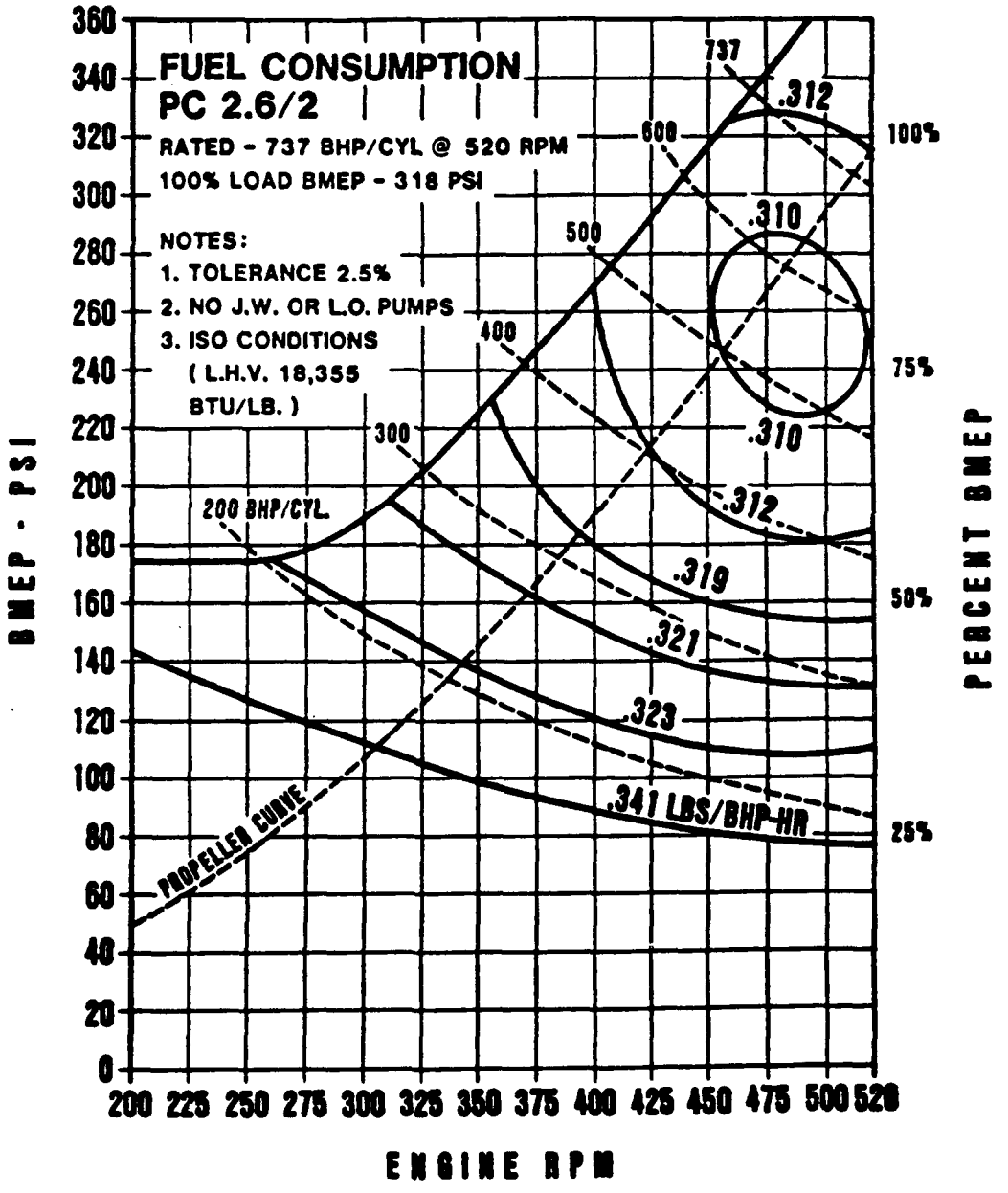

Figure 8 PC2.6V Engine's SFC Performance versus Operating Conditions [21]

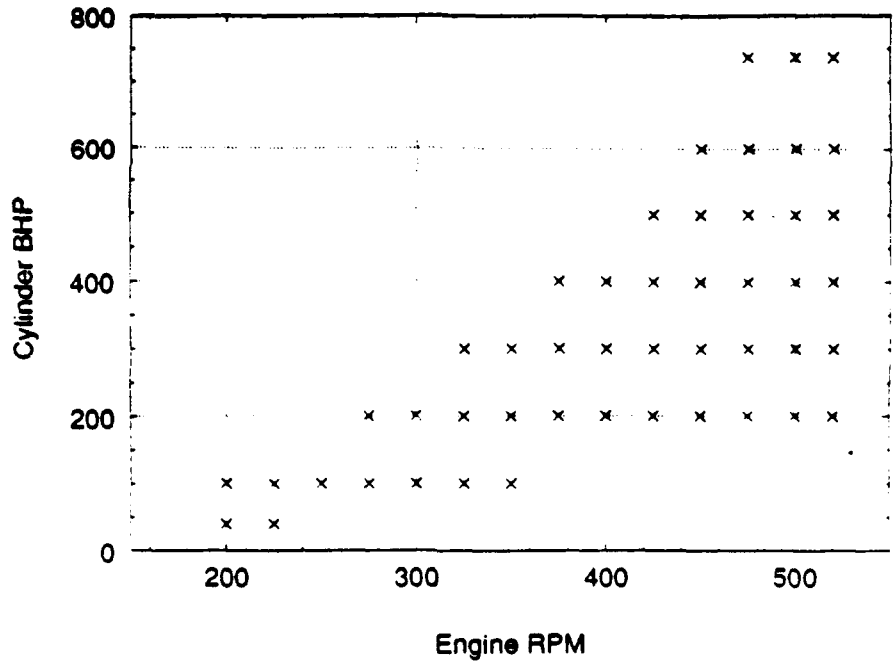

Figure 9 PC2.6V SFC Data Points

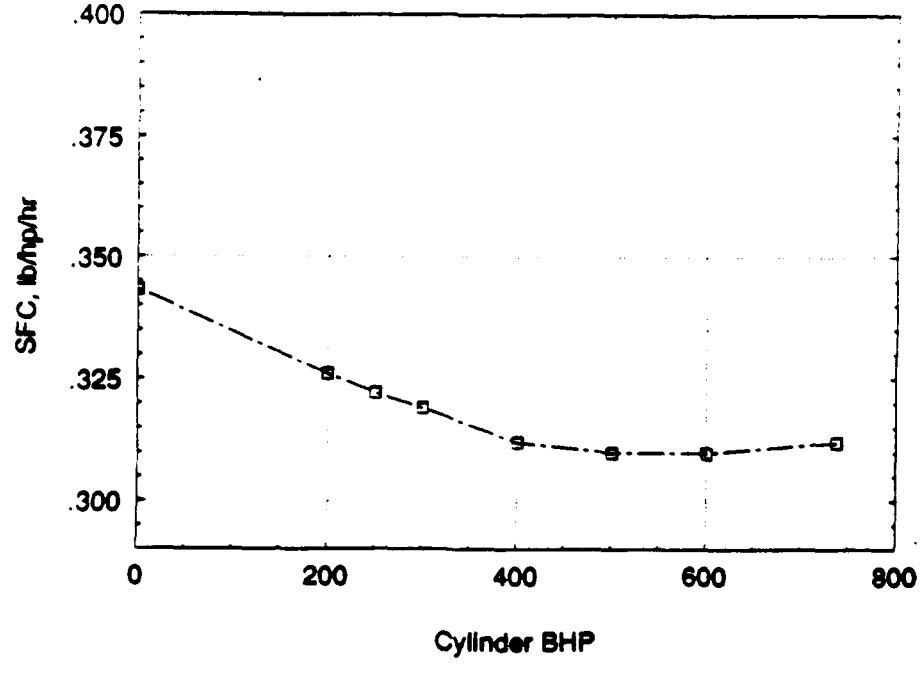

Figure 10 SFC Extrapolation for PC2.6V Low Cylinder Power Operation

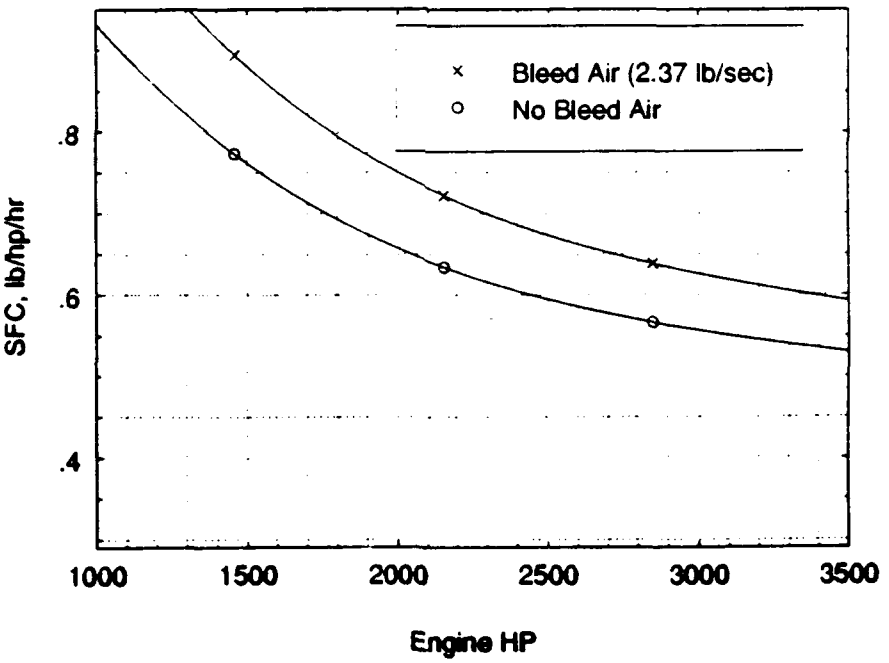

Figure 11 Allison 501-K34 SFC Performance versus BHP

Appendix B

Intake and Uptake Weight Calculation

Intake Weight Calculation

The intake weight estimating algorithm is based on the procedure used in ASSET [24]. The intake is assumed to include the moisture separators, louvers, built-in plenums, and acoustical insulation required to support both propulsion air and module cooling air. The intake weight algorithm provides the weight per linear foot of ducting for gas turbines and diesels.

The gas turbine intake weight algorithm is based on the intake component weights from the FFG-7 and the DD-963. These component weights were used to develop the total gas turbine intake weight per linear foot of ducting, W_{gt_intake} , for a single gas turbine engine at rated power, P_{gt} , has:

$$W_{gt\_intake} = 0.013 * (P_{gt}\wedge 2 + 7.5 * 10\wedge 8)\wedge 0.5 \text{ , lb/ft} \qquad (1)$$

The diesel intake weight algorithm is based on manufacturers data from Colt-Pielstick and DeLaval. The total diesel intake weight per linear foot of ducting, W_{dsl_intake} , for a single diesel engine at rated power, P_{dsl} , is:

$$W_{dsl\_intake} = 2.0 * P_{dsl}\wedge 0.5 \text{ , lb/ft} \qquad (2)$$

The rated power for each engine was entered into the appropriate equation to determine the linear weight of the intake.

Uptake Weight Calculation

Has before, the uptake weight estimating algorithm is also based on the procedure used in ASSET [24]. The uptake weight algorithm provides the weight per linear foot of ducting for gas turbines and diesels.

The gas turbine uptake weight algorithm is based on the uptake weights from the FFG-7 and the DD-963. These weights were used to develop the total gas turbine

uptake weight per linear foot of ducting, W_{gt_uptake}, for a single gas turbine engine at rated power, P_{gt}, has:

$$W_{gt\_uptake} = 0.021 * (P_{gt}\text{^}2 + 7.5 * 10\text{^}8)\text{^}0.5 \text{ , lb/ft} \quad (3)$$

The diesel uptake weight algorithm is based on the same manufacturers data from Colt-Pielstick and DeLaval. The total diesel uptake weight per linear foot of ducting, W_{dsl_uptake}, for a single diesel engine at rated power, P_{dsl}, is:

$$W_{dsl\_uptake} = 3.1 * P_{dsl}\text{^}0.5 \text{ , lb/ft} \quad (4)$$

The rated power for each engine was entered into the appropriate equation to determine the linear weight of the uptake.

Dividing equation (3) by equation (1) shows that the gas turbine uptake is 1.62 times heavier per linear foot than its associated gas turbine intake. Similarly, dividing equation (4) by equation (2) shows that the diesel uptake is 1.55 times heavier per linear foot than its associated diesel intake.

Appendix C

Intake and Uptake Volume Calculation

Intake Cross Sectional Area Determination

The volume requirement for the intake ducting of a given engine will be assumed to be equal to the average cross sectional area of the ducting times the length of the intake ducting.

The required average cross sectional area for the intake ducting of a gas turbine engine is assumed to be equal to: 1) the area defined by the air intake flange connection on the gas turbine module enclosure plus, 2) the area from an additional one foot wide rectangular ring based on the enclosure flange dimensions to allow for structure and acoustical insulation on the ducting coming down through the ship plus, 3) five square feet to allow for the module enclosure cooling air ducting. These areas are shown in figure 1.

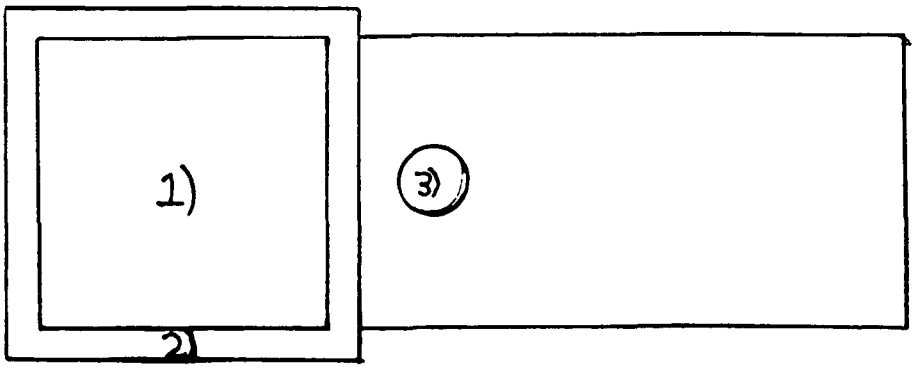

Figure 1 Gas Turbine Engine Air Intake Cross Sectional Area

The dimensions for the air intake flange connections of the LM-2500 and ICR gas turbine module enclosures are equivalent and are 8.67 ft by 8.75 ft. The air intakes are sized to allow for the necessary air flow and to allow for the engine removal up the intake ducting. The dimensions for the Allison 571-KF air intake flange are 2.34 ft by 5.45 ft. All three gas turbines require a significant amount of cooling air flow through the module enclosure allowed for by the five square feet of ducting.

The required average cross sectional area for the intake ducting of a diesel engine is assumed to be equal to: 1) the area defined by the air intake flange connection on the diesel engine plus, 2) the area from an additional one foot wide annulus based around the intake flange dimension to allow for structure and acoustical insulation or silencers on the ducting coming down through the ship. Figure 2 shows a typical marine diesel air intake system. The diameter of the pipe is specified by the manufacturer for each engine but in general the diameter is on the order of 16 to 26 inches.

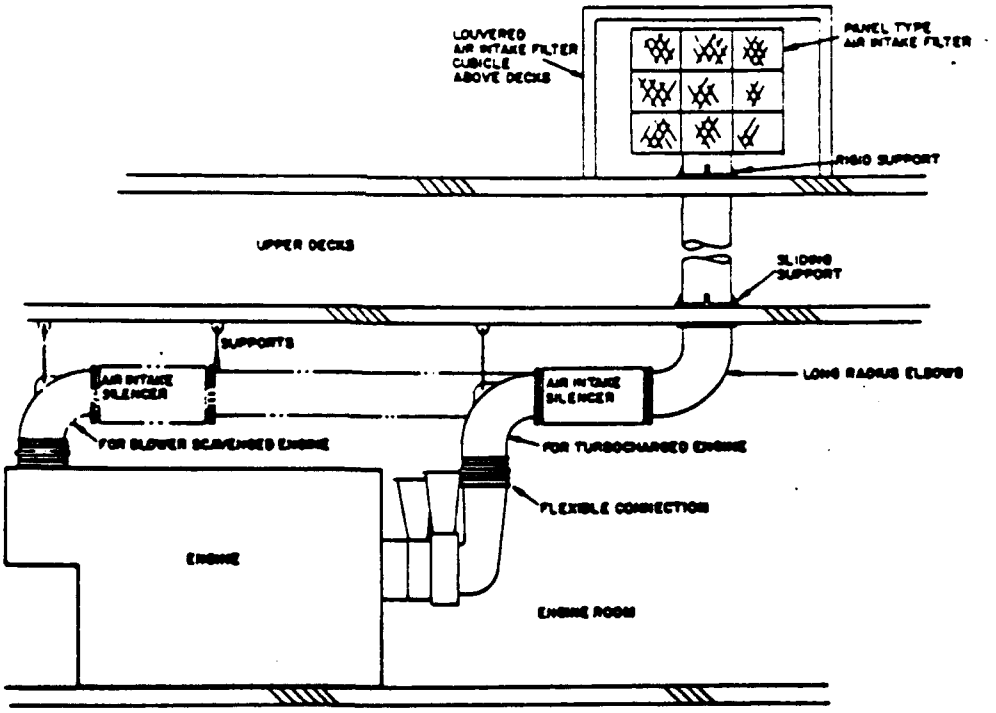

Figure 2 Typical Marine Diesel Air Intake System [25]

Uptake Cross Sectional Area Determination

The volume requirement for the uptake ducting of a given engine will be assumed to be equal to the average cross sectional area of the ducting times the length of the uptake ducting.

The required average cross sectional area for the uptake ducting of a gas turbine engine is assumed to be equal to: 1) the area defined by the exhaust flange connection on the gas turbine module enclosure plus, 2) the area from an additional 1.5 foot wide rectangular ring based on the exhaust flange dimensions to allow for structure, acoustical insulation, and thermal insulation on the ducting going up through the ship plus, 3) five square feet to allow for the exhausting of the module enclosure cooling air.

The dimensions for the exhaust flange connections of the LM-2500 and ICR gas turbine module enclosures are equivalent and are 8.67 ft by 10.5 ft. The dimensions for the Allison 571-KF exhaust flange are 6.14 ft by 5.45 ft. All three gas turbines mix the module cooling air exhaust with the engine exhaust and this extra flow is allowed for by the addition of the five square feet to the exhaust ducting.

The required average cross sectional area for the uptake ducting of a diesel engine is assumed to be equal to: 1) the area defined by the exhaust flange connection on the diesel engine plus, 2) the area from an additional 1.5 foot wide annulus based around the intake flange dimension to allow for structure, acoustical insulation or silencers, and thermal insulation on the ducting going up through the ship. Figure 3 shows a typical marine diesel exhaust system. The diameter of the pipe is specified by the manufacturer for each engine but in general the diameter is on the order of 16 to 30 inches.

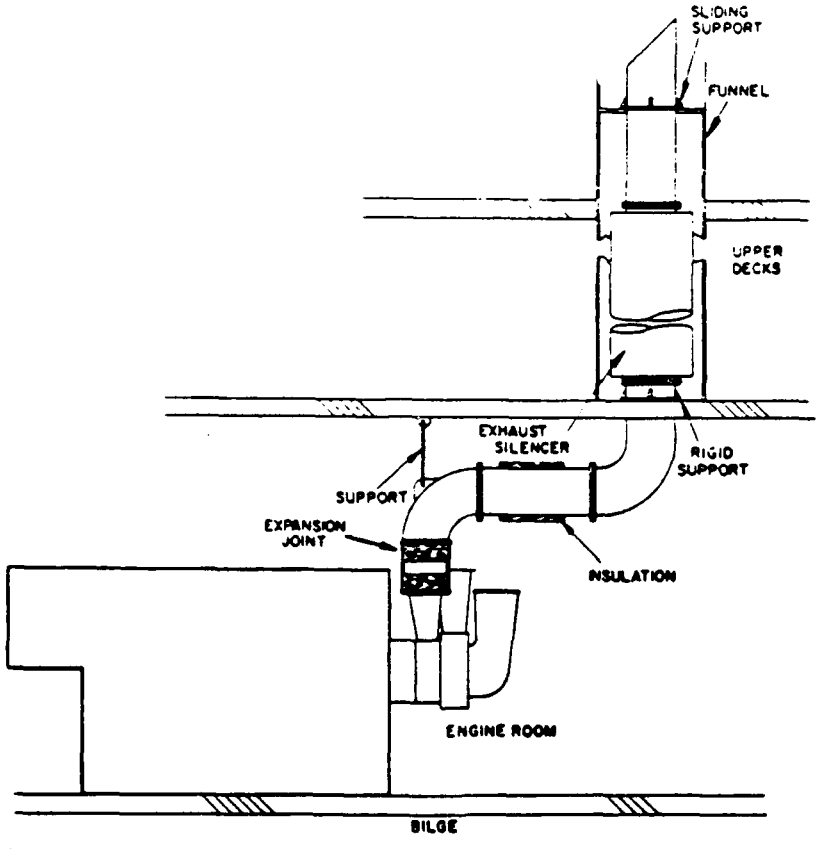

Figure 3 Typical Marine Diesel Exhaust System [26]

Appendix D

Fuel Weight Calculation

The calculation to determine the required fuel weight to meet a given ship's endurance range and speed constraints is based on a standardized Navy procedure described in reference [16]. That standard procedure is discussed here. Key terms defined in the reference and used in this discussion are underlined.

Calculation of Propulsion Fuel Load

The design endurance power is the Effective HorsePower (EHP) required to meet the endurance speed constraint. A 10 percent margin is applied to the endurance EHP to allow for adverse sea conditions and average bottom fouling over a 2-year period. The average endurance power is the propulsion power requirement based on the 10 percent margin applied to the endurance EHP.

The required engine Brake HorsePower (BHP) for endurance is determined by applying the appropriate Propulsive Coefficient (PC) to the average endurance power. If more than one engine is used for endurance, it is assumed that the load will be split equally between the on-line engines. If Propulsion Derived Ship Service (PDSS) electric power will be used, its additional power requirement must be added to the engine BHP. The engine endurance RPM may also be required to determine the engine Specific Fuel Consumption (SFC). Additionally, if customer bleed air from the engine will be used; its impact on engine SFC must be included. Unless otherwise specified, prairie and masker systems shall be considered in operation 50 percent of the time for ships so fitted.

For all calculations on SFC, the Navy standard day conditions shown in table 1 were employed.

100°F Ambient Temperature
14.7 psia Ambient Pressure
4.0 in of H_2O Intake Loss
6.0 in of H_20 Exhaust Loss
18,400 Btu/lbm Lower Heating Value
85°F Seawater Inlet Temperature

Table 1 Navy Standard Day Conditions

With the engine BHP, RPM, and the standard conditions; the engine SFC at endurance can be determined. There are two margins that are applied to the S . The specified fuel consumption is the SFC times a correction factor to allow a tolerance for instrumentation inaccuracy and design changes during the construction period. The correction factor is 1.04 if the average endurance power is one-third or less of the total rated BHP of all propulsion engines, 1.03 if between one-third and two-thirds, and 1.02 if greater than two-thirds. The average endurance fuel consumption is the specified fuel consumption increased by 5 percent to allow for plant deterioration over a two year period.

The burnable propulsion endurance fuel is the sum of the individual engines' average endurance fuel consumption multiplied by the engine BHP and the time at endurance. If bleed air is extracted from the engine for only a percentage of the endurance time, t_{end} , the different SFC for the bleed and no-bleed operation must be accounted for as shown in equation (1).

$$\text{Fuel Wt} = t_{end} * \text{BHP} (SFC_{bleed} * \text{bleed \%time} + SFC_{no\text{-}bleed} * \text{no-bleed \%time}) \quad (1)$$

The propulsion endurance fuel load is the burnable propulsion endurance fuel divided by the tailpipe allowance. The tailpipe allowance allows for the unavailable fuel remaining in the tank below the suction tailpipes. If the majority of the tanks are broad and shallow, the factor is 0.95; if narrow and deep, it is 0.98.

Calculation of Ship Service Electrical Power Fuel Load

The non-PDSS ships service power generators will also impact the fuel load and must be considered in the total fuel weight.

The average 24 hour electrical load must be determined for the ship. It is assumed that the average 24 hour electrical load will be split equally among all the on-line generators, PDSS and non-PDSS, at endurance. If PDSS electrical power will be used, its power requirement is added to the propulsion engine BHP. The stand alone generator set engines are the only engines used to determine the electrical power fuel load. Once again, if customer bleed air from the generator set's engine will be used; its impact on engine SFC must be included. Unless otherwise specified, prairie and masker systems shall be considered in operation 50 percent of the time for ships so fitted.

With the generator set's engine BHP, RPM, and the standard conditions; the engine SFC at endurance can be determined. Once again, there are two margins that are applied to the SFC. The specified electrical fuel consumption is the SFC times a correction factor to allow a tolerance for instrumentation inaccuracy and machinery changes. The correction factor is 1.04 if the average 24 hour electrical load is one-third or less of the (total number of generators minus one) times the generator rating, 1.03 if between one-third and two-thirds, and 1.02 if greater than two-thirds. The average electrical endurance fuel consumption is the specified fuel consumption increased by 5 percent to allow for plant deterioration over a two year period.

The burnable electrical endurance fuel is the sum of the individual engines' average endurance fuel consumption multiplied by the engine BHP and the time at endurance. If bleed air is extracted from the engine for only a percentage of the endurance time, the different SFC for the bleed and no-bleed operation must be accounted for as shown in equation (1). The electrical endurance fuel load is the burnable electrical endurance fuel divided by the tailpipe allowance. Again, the tailpipe allowance allows for the unavailable fuel remaining in the tank below the suction tailpipes.

The endurance fuel load is the sum of the propulsion endurance fuel load and the electrical endurance fuel load. If any other equipment uses fuel, such as an installed donkey boiler, its fuel requirements must also be included in the endurance fuel load.

Appendix E

Detailed Listing of The Computer Code

The following pages contain the developed engine's computer code.

```
* This is the stand alone engine's program used to determine the */
/* the engine performance and specs for a propulsion system. */

#include <stdio.h>
#include <math.h>

#define INLET_LOSS 4.0        /* in inches of water, max allowed 12 */
#define EXHAUST_LOSS 6.0  /* in inches of water, max allowed 20 */
#define HUMIDITY 116.2        /* in grains, max allowed 350 */
                              /* 116.2 grains is equal to 40% relative*/
#define RANGE 4429.0
#define ENDUR_SPD 20.0
#define ENDUR_PCT_BLD 0.50  /* percent of endurance time with bleed */
                                    /* air supplying prairie + masker */

#define AVG_ELEC_LOAD 2525.0 /* Avg. KW load for the DDG */
#define TPA 1.02          /* Tail Pipe Allowance */

int plant_map [7][2];

char buff[11];          /* general purpose string buffer */

/*The following are the function declarations.  */

int user_interface(void);
void mechanical_drive(void);
void electric_drive(void);
void mech_elec_hybrid_drive(void);
void multiple_waterjet(void);
void main_engine_specs();
void endurance_engine_specs();
double fuel_load (double, double, double, double, double,
                        double *cruise_engine_specs,double);
double engine_sfc(double,double,double,double);
double lm2500_map(double,double);
double pc42_sfc_map(double,double,double);
double pc26_sfc_map(double,double,double);


void main()

{
int program_continue_flag;

double weight_fuel,          /* Declare the variables seen by main. */
ehp_cruise,
qpc_cruise,
xmission_eff_cruise,
rpm_prop_cruise,
gear_ratio;

double boost_engine_specs[14],  /* Declare engine arrays. */
```

```
cruise_engine_specs[14];


/* ------------------------------------------------------------------------ */
/* -- Call the program interface function and provide the - */
/* -- option to quit the program or to reselect options. ----- */
/* ------------------------------------------------------------------------ */

program_continue_flag = user_interface();

while (program_continue_flag == 2)
        {
  /* ------------------------------------------------------------------------ */
  /* reset plant_map to zero before re-calling user_interface */
  /* ------------------------------------------------------------------------ */
        plant_map [0][0] = 0;
        plant_map [1][0] = 0;
        plant_map [2][0] = 0;
        plant_map [3][0] = 0;
        plant_map [4][0] = 0;
        plant_map [5][0] = 0;
        plant_map [6][0] = 0;
        plant_map [0][1] = 0;
        plant_map [1][1] = 0;
        plant_map [2][1] = 0;
        plant_map [3][1] = 0;
        plant_map [4][1] = 0;
        plant_map [5][1] = 0;
        plant_map [6][1] = 0;

        program_continue_flag = user_interface();
  }
if (program_continue_flag == 3)
  {
  goto end;
  }

/* ------------------------------------------------------------ */
/* -- Commence the resist loop  ---------------- */
/* ------------------------------------------------------------ */
/* ------------------------------------------------------------ */


main_engine_specs(boost_engine_specs);

endurance_engine_specs(boost_engine_specs,cruise_engine_specs);

/* These statements are used to obtain the dummy arguments which are */
/* used to pass arguments to the fuel_load function. In the */
/* integrated program, these will be replaced by functions. */

gets(buff);             /* this clears all preceding input so it won't interfere with the following input */

printf("Enter the following in decimal format.\n ");
```

```
printf("Enter the cruise EHP:");
gets(buff);
sscanf(buff,"%lf", &ehp_cruise);
printf("Enter the cruise QPC:");
gets(buff);
sscanf(buff,"%lf", &qpc_cruise);
printf("Enter the cruise transmission efficiency:");
gets(buff);
sscanf(buff,"%lf", &xmission_eff_cruise);
printf("Enter the cruise propulsor RPM:");
gets(buff);
sscanf(buff,"%lf", &rpm_prop_cruise);
printf("Enter the gear ratio:\n");
gets(buff);
sscanf(buff,"%lf", &gear_ratio);


weight_fuel = fuel_load(ehp_cruise, qpc_cruise, xmission_eff_cruise,
                        rpm_prop_cruise, gear_ratio,
                        cruise_engine_specs, boost_engine_specs[0]);

if(weight_fuel == -1.0)
 {
 goto end;
 }


/* ------------------------------------------------ */
/* --- Perform the Output          ---------- */
/* ------------------------------------------------ */

printf("The Total Fuel Weight = %6.1f LTONS.\n\n", weight_fuel);


printf("The cruise type engine Specs:\n\n");
printf("Max Engine Power    = %8.2f.\n", cruise_engine_specs[0]);
printf("Max Engine RPM      = %8.2f.\n", cruise_engine_specs[1]);
printf("Min Engine RPM      = %8.2f.\n", cruise_engine_specs[2]);
printf("Number of Cylinders = %8.2f.\n", cruise_engine_specs[3]);
printf("Engine Type Code    = %8.2f.\n", cruise_engine_specs[4]);
printf("Engine Weight       = %8.2f lb.\n", cruise_engine_specs[5]);
printf("Linear Weight Intake = %8.2f lb.\n", cruise_engine_specs[6]);
printf("Linear Weight Uptake = %8.2f lb.\n", cruise_engine_specs[7]);
printf("Engine Length, ft   = %8.2f.\n", cruise_engine_specs[8]);
printf("Engine Width, ft    = %8.2f.\n", cruise_engine_specs[9]);
printf("Engine Height, ft   = %8.2f.\n", cruise_engine_specs[10]);
printf("Cross Section Intake = %8.2f ft^2.\n", cruise_engine_specs[11]);
printf("Cross Section Uptake = %8.2f ft^2.\n", cruise_engine_specs[12]);
printf("Acquisition Cost    = %8.2f $mil,1991.\n\n", cruise_engine_specs[13]);


printf("The boost type engine Specs:\n\n");
printf("Max Engine Power    = %8.2f BHP.\n", boost_engine_specs[0]);
printf("Max Engine RPM      = %8.2f.\n", boost_engine_specs[1]);
```

```
printf("Min Engine RPM      = %8.2f.\n", boost_engine_specs[2]);
printf("Number of Cylinders = %8.2f.\n", boost_engine_specs[3]);
printf("Engine Type Code    = %8.2f.\n", boost_engine_specs[4]);
printf("Engine Weight       = %8.2f lb.\n", boost_engine_specs[5]);
printf("Linear Weight Intake = %8.2f lb.\n", boost_engine_specs[6]);
printf("Linear Weight Uptake = %8.2f lb.\n", boost_engine_specs[7]);
printf("Engine Length, ft   = %8.2f.\n", boost_engine_specs[8]);
printf("Engine Width, ft    = %8.2f.\n", boost_engine_specs[9]);
printf("Engine Height, ft   = %8.2f.\n", boost_engine_specs[10]);
printf("Cross Section Intake = %8.2f ft^2.\n", boost_engine_specs[11]);
printf("Cross Section Uptake = %8.2f ft^2.\n", boost_engine_specs[12]);
printf("Acquisition Cost    = %8.2f $mil.1991.\n", boost_engine_specs[13]);


end:;

}

int user_interface(void)

/* This is the user interface function used to define  */
/* the propulsion system.                               */

{
  int program_continue_flag = 1;

  printf("Select one of the four propulsion options.\n");
  printf("1  Mechanical Drive.\n");
  printf("2  Electric Drive.\n");
  printf("3  Mechanical with Electric Hybrid drive.\n");
  printf("4  Multiple Dispersed Waterjets.\n");
  scanf("%d", &plant_map[2][0]);

  while (plant_map[2][0] < 1 || plant_map[2][0] > 4)
        {
        printf("ERROR!  Enter 1, 2, 3, or 4.\n\n");
        scanf("%d", &plant_map[2][0]);
        }

  if (plant_map[2][0] == 1)
    {
    mechanical_drive();
    }
  if (plant_map[2][0] == 2)
    {
    electric_drive();
    }
  if (plant_map[2][0] == 3)
    {
    mech_elec_hybrid_drive();
    }
  if (plant_map[2][0] == 4)
    {
    multiple_waterjet();
```

```
    }

/* ------------------------------------------------------------ */
/* ----- Display the selections.                 ---------- */
/* ------------------------------------------------------------ */
printf("-----------------------------------------\n\n ");
printf("\n\nYour selections are summarized as follows:\n\n");

   printf("# Cruise Propulsors= %d\t\t", plant_map[0][0]);
   printf("# Boost Propulsors = %d\n", plant_map[0][1]);
/* ------------------------------------------------------------------------- */
   if (plant_map[1][0] == 1)
   {
   printf("Propulsor Type    = FPP\t");
   }
   if (plant_map[1][0] == 2)
   {
   printf("Propulsor Type    = CRP\t");
   }
   if (plant_map[1][0] == 3)
   {
   printf("Propulsor Type    = Contra\t");
   }
   if (plant_map[1][0] == 4)
   {
   printf("Propulsor Type    = Preswirl Stator\t");
   }
   if (plant_map[1][0] == 5)
   {
   printf("Propulsor Type    = Ducted FPP\t");
   }
   if (plant_map[1][0] == 6)
   {
   printf("Propulsor Type    = Ducted CRP\t");
   }
   if (plant_map[1][0] == 7)
   {
   printf("Propulsor Type    = Ducted Contra\t");
   }
   if (plant_map[1][0] == 8)
   {
   printf("Propulsor Type    = Ducted Preswirl\t");
   }
   if (plant_map[1][0] == 9)
   {
   printf("Propulsor Type    = Waterjet\t");
   }
/* ------------------------------------------------------------ */

   if (plant_map[1][1] == 0)
   {
   printf("Hybrid Trans Type  = None\n");
   }
   if (plant_map[1][1] == 1)
```

```
    {
    printf("Hybrid Trans Type  = Electric\n");
    }

/* ------------------------------------------------------ */
    if (plant_map[2][0] == 1)
    {
    printf("Transmission Type  = Mech\t");
    }
    if (plant_map[2][0] == 2)
    {
    printf("Transmission Type  = Elec\t");
    }
    if (plant_map[2][0] == 3)
    {
    printf("Transmission Type  = Hybrid\t");
    }
    if (plant_map[2][0] == 4)
    {
    printf("Transmission Type  = Multi-Jet\t");
    }
/* ------------------------------------------------------ */


    if (plant_map[2][1] == 0)
    {
    printf("XMission per shaft = None\n");
    }
    if (plant_map[2][1] == 1)
    {
    printf("XMission per shaft = Epicyclic\n");
    }
    if (plant_map[2][1] == 2)
    {
    printf("XMission per shaft = LTDR\n");
    }
    if (plant_map[2][1] == 3)
    {
    printf("XMission per shaft = LTDR w/Reversing\n");
    }
    if (plant_map[2][1] == 4)
    {
    printf("XMission per shaft = LTDR w/cross-connect\n");
    }

/* ------------------------------------------------------ */
    if (plant_map[3][0] == 0)
    {
    printf("Cruise Eng Type    = none\t");
    }
    if (plant_map[3][0] == 1)
    {
    printf("Cruise Eng Type    = ICR\t");
    }
```

```
   if (plant_map[3][0] == 2)
   {
   printf("Cruise Eng Type   = PC4.2\t");
   }
   if (plant_map[3][0] == 3)
   {
   printf("Cruise Eng Type   = PC2.6\t");
   }
/* ------------------------------------------------------- */
   if (plant_map[3][1] == 0)
   {
   printf("Boost Eng Type    = None\n");
   }
   if (plant_map[3][1] == 1)
   {
   printf("Boost Eng Type    = LM-2500\n");
   }
   if (plant_map[3][1] == 2)
   {
   printf("Boost Eng Type    = ICR\n");
   }
   if (plant_map[3][1] == 3)
   {
   printf("Boost Eng Type    = PC4.2 w/16 Cyl.\n");
   }
   if (plant_map[3][1] == 4)
   {
   printf("Boost Eng Type    = LM-2500 Power PAK\n");
   }
   if (plant_map[3][1] == 5)
   {
   printf("Boost Eng Type    = Allison 571-KF\n");
   }

/* ------------------------------------------------------- */
   printf("Total # boost eng\t\t Total # boost eng\n");
   printf("   used for cruise = %d\t\t", plant_map[4][0]);
   printf("   used for boost  = %d\n", plant_map[4][1]);
   printf("Total # cruise eng\t\t Total # cruise eng\n");
   printf("   used for cruise = %d\t\t", plant_map[5][0]);
   printf("   used for boost  = %d\n", plant_map[5][1]);
/* ------------------------------------------------------- */

   if (plant_map[6][0] == 0)
   {
   printf("PDSS flag setting  = NO\t\t");
   }
   if (plant_map[6][0] == 1)
   {
   printf("PDSS flag setting  = YES\t\t");
   }
/* ------------------------------------------------------- */
   printf("Number of PDSS     = %d\n", plant_map[6][1]);
   printf("-----------------------------------------\n\n ");
```

```
  printf("Select an option:\n");
  printf("1  Run the program with your selected propulsion system.\n");
  printf("2  Reselect the propulsion system.\n");
  printf("3  Quit the program.\n\n");
  scanf("%d", &program_continue_flag);
  while ( program_continue_flag < 1 || program_continue_flag > 3)
    {
        printf("ERROR!  Enter 1, 2, or 3.\n\n");
        scanf("%d", &program_continue_flag);
    }
        return(program_continue_flag);

}

/* ------------------------------------------------------------------------------------------ */

void mechanical_drive(void)
{
  int n_engine_cruising;

  printf("Select the number of propulsors used for cruise.\n\n");
  printf("1  One propulsor.\n");
  printf("2  Two propulsors.\n");
        printf("3  Three propulsors.\n\n");
        scanf("%d", &plant_map[0][0]);

  while (plant_map [0][0] < 1 || plant_map [0][0] > 3)
        {
        printf("ERROR!  Enter  1, 2, or 3.\n\n");
        scanf("%d", &plant_map[0][0]);
        }

  printf("Select the number of propulsors used for boost.\n\n");
  printf("2  Two propulsors.\n");
  printf("3  Three propulsors.\n\n");
  scanf("%d", &plant_map[0][1]);

  while (plant_map [0][1] < 2 || plant_map [0][1] > 3)
        {
        printf("ERROR!  Enter  2 or 3.\n\n");
        scanf("%d", &plant_map[0][1]);
        }

  printf("Select one of the propulsor options.\n\n");
  printf("1  FPP\n");
  printf("2  CRP\n");
  printf("3  Contra-rotating\n");
  printf("4  Preswirl Stator\n");
  printf("5  Ducted FPP\n");
  printf("6  Ducted CRP\n");
  printf("7  Ducted Contra-rotating\n");
  printf("8  Ducted Preswirl Stator\n");
  printf("9  Waterjet\n\n");
  scanf("%d", &plant_map[1][0]);
```

```
  while (plant_map[1][0] < 1 || plant_map[1][0] > 10)
        {
        printf("ERROR!  Enter 1 through 9.\n\n");
        scanf("%d", &plant_map[1][0]);
        }


/* ------------------------------------------------------------------------------- */
/* Specify the transmission type to support either the two propulsor  */
/* or the three propulsor options.                                                 */
/* ------------------------------------------------------------------------------- */

    printf("Select the desired mechanical transmission options.\n\n");
    if (plant_map[0][1] == 3)
      {

      printf("1  Epicyclic on each shaft.\n");
      printf("2  LTDR on each shaft.\n");
      printf("3  LTDR with reversing mechanism on each shaft.\n");
      scanf("%d", &plant_map[2][1]);

      while (plant_map[2][1] < 1 || plant_map[2][1] > 3)
            {
            printf("ERROR!  Enter 1, 2, or 3.\n\n");
            scanf("%d", &plant_map[2][1]);
            }
      }
     else
            {
            printf("1  Epicyclic on each shaft.\n");
             printf("2  LTDR on each shaft.\n");
             printf("3  LTDR with reversing mechanism on each shaft.\n");
              printf("4  LTDR on each shaft with cross-connect.\n");
              scanf("%d", &plant_map[2][1]);

      while (plant_map[2][1] < 1 || plant_map[2][1] > 4)
            {
            printf("ERROR!  Enter 1, 2, 3, or 4.\n\n");
            scanf("%d", &plant_map[2][1]);
            }
      }
/* ------------------------------------------------------------------------ */
/* -----   Specify the engine type[s] and arrangements.   /
/* ------------------------------------------------------------------------ */

    printf("The following questions are used to determine the engine\n");
    printf("type[s] and alignments to operate the propulsion system.\n");
    printf("Note:  if you choose the LM-2500 for boost, you will also be\n");
    printf("given the opportunity to select an alternative\n");
    printf("engine type for cruise.\n");
    printf("If you choose one of the other boost engine types,\n");
    printf("it will be assumed that your selected boost engine \n");
```

```
printf("type  will be used for both cruise and boost operation.\n\n");


printf("Select one of the boost engine types.\n");
printf("1  LM-2500 Gas Turbine\n");
printf("2  ICR Gas Turbine\n");
printf("3  PC 4.2 Diesel,16 cyl rated at 26060 HP.\n\n");
scanf("%d", &plant_map[3][1]);

while (plant_map[3][1] < 1 || plant_map[3][1] > 3)
        {
        printf("ERROR!  Enter 1, 2, or 3.\n\n");
        scanf("%d", &plant_map[3][1]);
        }

if(plant_map[3][1] == 1)
{

 /* ------------------------------------------------------------------- */
 /* If the boost engine type is LM-2500, there will be    */
/* a cruise engine type option.                                  */
/* ------------------------------------------------------------------- */

 printf("Select one of the cruise engine types.\n");

 printf("0  NONE\n");
 printf("1  ICR Gas Turbine\n");
 printf("2  PC 4.2 Diesel\n");
 printf("3  PC 2.6 Diesel\n\n");
scanf("%d", &plant_map[3][0]);

 while (plant_map[3][0] < 0 || plant_map[3][0] > 3)
        {
        printf("ERROR!  Enter 0, 1, 2, or 3.\n\n");
        scanf("%d", &plant_map[3][0]);
        }

 if(plant_map[3][0] == 0)
 {
  /* ------------------------------------------------------------------- */
  /* No cruise engine type was selected. The LM-2500 engine   */
  /* must provide for both the boost and the cruise operation.    */
  /* ------------------------------------------------------------------- */

  printf("Select the total number of LM-2500 engines\n");
  printf("used during maximum boost operation.\n");
  printf("2  Two\n");
  printf("3  Three\n");
  printf("4  Four\n");
  scanf("%d", &plant_map[4][1]);

  while (plant_map[4][1] < 2 || plant_map[4][1] > 4)
        {
        printf("ERROR!  Enter 2, 3, or 4.\n\n");
```

```
        scanf("%d", &plant_map[4][1]);
        }

 printf("Select the total number of LM-2500 engines used\n");
 printf("during cruise operation.\n");
 printf("1  One\n");
printf("2  Two\n");
 printf("3  Three\n");
printf("4  Four\n\n");
 scanf("%d", &plant_map[4][0]);

 while (plant_map[4][0] < 1 || plant_map[4][0] > 4)
        {
        printf("ERROR!  Enter 1, 2, 3, or 4.\n\n");
        scanf("%d", &plant_map[4][0]);
        }
}

/* ---------------------------------------------------------- */
/* If the cruise engine type is ICR, allow it to */
/* be used for boost i.e. COGAG.
/* ---------------------------------------------------------- */

if(plant_map[3][0] == 1)
  {
  printf("Select the total number of LM-2500 boost engines\n");
  printf("used during maximum boost operation.\n");
  printf("2  Two\n");
  printf("3  Three\n");
  printf("4  Four\n");
  scanf("%d", &plant_map[4][1]);

  while (plant_map[4][1] < 2 || plant_map[4][1] > 4)
        {
        printf("ERROR!  Enter 2, 3, or 4.\n\n");
        scanf("%d", &plant_map[4][1]);
        }

  printf("You have selected an ICR cruise engine type that\n");
  printf("may also be used for boost operation.\n");

  printf("Select the total number of ICR engines used\n");
  printf("during maximum boost operation.\n");
  printf("0  None\n");
  printf("1  One\n");
  printf("2  Two\n");
  printf("3  Three\n");
  printf("4  Four\n");
  scanf("%d", &plant_map[5][1]);

  while (plant_map[5][1] < 0 || plant_map[5][1] > 4)
        {
        printf("ERROR!  Enter 0, 1, 2, 3, or 4.\n\n");
        scanf("%d", &plant_map[5][1]);
```

```
        }

 printf("Select the total number of ICR engines online\n");
 printf("during cruise operation.\n\n");
 printf("1  One\n");
 printf("2  Two\n");
 printf("3  Three\n");
 printf("4  Four\n");
 scanf("%d", &plant_map[5][0]);

 while (plant_map[5][0] < 1 || plant_map[5][0] > 4)
        {
        printf("ERROR!  Enter 1, 2, 3, or 4.\n\n");
        scanf("%d", &plant_map[5][0]);
        }

}

if(plant_map[3][0] == 2 || plant_map[3][0] == 3)
{

/* ------------------------------------------------------------------------ */
/* If the cruise engine type is diesel, do not allow it *         /
/* to be used for boost i.e. CODAG.  Do not allow          */
/* CODAG on a shaft. Additionally, do not allow CODAG   */
/* even when the diesel will be on its own shaft.          */
/* The diesel only shaft will be trailed at boost.         */
/* ------------------------------------------------------------------------ */


printf("Select the total number of LM-2500 boost engines\n");
printf("used during maximum boost operation.\n");
printf("2  Two\n");
printf("3  Three\n");
printf("4  Four\n");
scanf("%d", &plant_map[4][1]);

while (plant_map[4][1] < 2 || plant_map[4][1] > 4)
        {
        printf("ERROR!  Enter 2, 3, or 4.\n\n");
        scanf("%d", &plant_map[4][1]);
        }

printf("Select the total number of Diesel cruise engines\n");
printf("used during cruise operation.\n\n");
printf("1  One\n");
printf("2  Two\n");
printf("3  Three\n");
printf("4  Four\n");
scanf("%d", &plant_map[5][0]);

while (plant_map[5][0] < 1 || plant_map[5][0] > 4)
        {
        printf("ERROR!  Enter 1, 2, 3, or 4.\n\n");
```

```
        scanf("%d", &plant_map[5][0]);
        }
        }


}
else
{
  /* ------------------------------------------------------------------------ */
  /* If the boost engine type is ICR or diesel, there will be          */
  /* no cruise engine type. The boost engine type must provide */
  /* for both the boost and the cruise operation.                      */
  /* ------------------------------------------------------------------------ */

  plant_map[3][0] = 0;

  printf("Select the total number of boost engines used\n");
  printf("during maximum boost operation.\n");
  printf("2  Two\n");
  printf("3  Three\n");
  printf("4  Four\n");
  scanf("%d", &plant_map[4][1]);

  while (plant_map[4][1] < 2 || plant_map[4][1] > 4)
        {
        printf("ERROR!  Enter 2, 3, or 4.\n\n");
        scanf("%d", &plant_map[4][1]);
        }

  printf("Select the total number of boost engines used\n");
  printf("during cruise operation.\n\n");
  printf("1  One\n");
  printf("2  Two\n");
  printf("3  Three\n");
  printf("4  Four\n\n");
  scanf("%d", &plant_map[4][0]);

  while (plant_map[4][0] < 1 || plant_map[4][0] > 4)
        {
        printf("ERROR!  Enter 1, 2, 3, or 4.\n\n");
        scanf("%d", &plant_map[4][0]);
        }

}
 /* ------------------------------------------- */

 /* ------------------------------------------- */

  while ( plant_map[0][1] > (plant_map[4][1] + plant_map[5][1]))
    {
     printf("Ensure that the sum of the number of boost engines and\n");
    printf("the number of cruise engines used for boost is equal\n");
    printf("to or greater than the total number of propulsors\n");
```

```
        printf("that are used for maximum boost.\n");


        printf("Re-select the total number of boost engines online\n");
        printf("during maximum boost operation.\n\n");
        printf("1  One\n");
        printf("2  Two\n");
        printf("3  Three\n");
        printf("4  Four\n");
        scanf("%d", &plant_map[4][1]);

        while (plant_map[4][1] < 1 || plant_map[4][1] > 4)
              {
              printf("ERROR!  Enter 1, 2, 3, or 4.\n\n");
              scanf("%d", &plant_map[4][1]);
              }

        if (plant_map[3][1] == 1 && plant_map[3][0] == 1)
         {
         printf("Re-select the total number of ICR engines used\n");
         printf("during maximum boost operation.\n");
         printf("0  None\n");
         printf("1  One\n");
        printf("2  Two\n");
         printf("3  Three\n");
        printf("4  Four\n");
         scanf("%d", &plant_map[5][1]);

         while (plant_map[5][1] < 0 || plant_map[5][1] > 4)
               {
               printf("ERROR!  Enter 0, 1, 2, 3, or 4.\n\n");
               scanf("%d", &plant_map[5][1]);
               }
          }
}


/* ------------------------------------------------------------------------------- */
/* ----- PDSS option only considered if all engine type[s] are only  */
/* ----- gas turbine.                                                              */
/* ------------------------------------------------------------------------------- */

         if (plant_map[3][1] == 1 || plant_map[3][1] == 2)
         {
         if (plant_map[3][0] == 0 || plant_map[3][0] == 1)
                  {
                  printf("Will PDSS be driven off any of the gas turbine\n");
                     printf("output shafts.\n\n");
                  printf("0  NO\n");
                  printf("1  Yes\n");
                  scanf("%d", &plant_map[6][0]);
                  while (plant_map[6][0] < 0 || plant_map[6][0] > 1)
                    {
                    printf("ERROR!  Enter 0 or 1.\n\n");
```

```
                    scanf("%d", &plant_map[6][0]);
                     }
                   }
          }

if (plant_map[6][0] == 1)
      {
      /* ------------------------------------------------------------ */
      /* Incorporate logic so that the minimum number of */
      /* PDSS units allowed is equal to the number of     */
      /* engines online during cruise.                    */
      /* ------------------------------------------------------------ */

      n_engine_cruising = plant_map[4][0] + plant_map [5][0];

      if(n_engine_cruising == 1)
       {
       printf("Select the total number of pdss units desired.\n");
       printf("1  One pdss unit.\n");
       printf("2  Two pdss units.\n");
       printf("3  Three pdss units.\n");
       printf("4  Four pdss units.\n");
       scanf("%d", &plant_map[6][1]);

       while (plant_map[6][1] > plant_map[4][1] + plant_map[5][0])
        {
        printf("ERROR!  Ensure the total number of PDSS units is less\n");
        printf("than the total number of gas turbines. Re-enter.\n\n");
        scanf("%d", &plant_map[6][1]);
        }

        while (plant_map[6][1] < 1 || plant_map[6][1] > 4)
         {
         printf("ERROR!  Enter 1, 2, 3, or 4.\n\n");
         scanf("%d", &plant_map[6][1]);
         }
          }

      if(n_engine_cruising == 2)
       {
       printf("Select the total number of pdss units desired.\n");
       printf("2  Two pdss units.\n");
       printf("3  Three pdss units.\n");
       printf("4  Four pdss units.\n");
       scanf("%d", &plant_map[6][1]);

       while (plant_map[6][1] > plant_map[4][1] + plant_map[5][0])
        {
        printf("ERROR!  Ensure the total number of PDSS units is less\n");
        printf("than the total number of gas turbines. Re-enter.\n\n");
        scanf("%d", &plant_map[6][1]);
        }

          while (plant_map[6][1] < 2 || plant_map[6][1] > 4)
```

```
            {
            printf("ERROR!  Enter 2, 3, or 4.\n\n");
            scanf("%d", &plant_map[6][1]);

            }

        if(n_engine_cruising == 3)
        {
        printf("Select the total number of pdss units desired.\n");
        printf("3  Three pdss units.\n");
        printf("4  Four pdss units.\n");
        scanf("%d", &plant_map[6][1]);

        while (plant_map[6][1] > plant_map[4][1] + plant_map[5][0])
          {
          printf("ERROR!  Ensure the total number of PDSS units is less\n");
          printf("than the total number of gas turbines. Re-enter.\n\n");
          scanf("%d", &plant_map[6][1]);
          }

           while (plant_map[6][1] < 3 || plant_map[6][1] > 4)
           {
           printf("ERROR!  Enter 3 or 4.\n\n");
           scanf("%d", &plant_map[6][1]);
           }
        }
      }

}



void electric_drive(void)
{
  int n_engine_cruising;

  /* Set the number of propulsors used for cruise = 2. */

  plant_map[0][0] = 2;

  /* Set the number of propulsors used for boost = 2. */

  plant_map[0][1] = 2;


  printf("Select one of the propulsor options.\n\n");
  printf("1  FPP\n");
  printf("2  CRP\n");
  printf("3  Contra-rotating\n");
  printf("4  Preswirl Stator\n");
  printf("5  Ducted FPP\n");
  printf("6  Ducted CRP\n");
  printf("7  Ducted Contra-rotating\n");
  printf("8  Ducted Preswirl Stator\n");
```

```
  printf("9  Waterjet\n\n");
  scanf("%d", &plant_map[1][0]);

  while (plant_map[1][0] < 1 || plant_map[1][0] > 10)
        {
        printf("ERROR!  Enter 1 through 9.\n\n");
        scanf("%d", &plant_map[1][0]);
        }

/* ------------------------------------------------------------------------- */
/* Specify the transmission type between the motor and propulsor. */
/* ------------------------------------------------------------------------- */

  printf("Select the transmission type between the motors and\n");
  printf("the propulsors.\n");
  printf("0  None\n");
  printf("1  Epicyclic\n");
  scanf("%d", &plant_map[2][1]);

  while (plant_map[2][1] < 0 || plant_map[2][1] > 1)
        {
        printf("ERROR!  Enter 0 or 1.\n\n");
        scanf("%d", &plant_map[2][1]);
        }
/* ------------------------------------------------------------------------- */
 * -----  Specify the engine type[s] and arrangements.  ---------- */
 * ------------------------------------------------------------------------- */

  printf("Select one of the propulsion engine types.\n");
  printf("1  LM-2500 Gas Turbine\n");
  printf("2  ICR Gas Turbine\n");
  printf("4  LM-2500 Power-PAK, includes generator\n\n");
  scanf("%d", &plant_map[3][1]);

  while (plant_map[3][1] != 1 && plant_map[3][1] != 2)
    {
    if (plant_map[3][1] != 4)
        {
        printf("ERROR!  Enter 1, 2, or 4.\n\n");
        scanf("%d", &plant_map[3][1]);
        }
    else
    {
     break;
     }
  }

    printf("Select the total number of engines online\n");
    printf("during maximum boost operation.\n\n");
    printf("1  One\n");
    printf("2  Two\n");
    printf("3  Three\n");
    printf("4  Four\n");
    scanf("%d", &plant_map[4][1]);
```

```
while (plant_map[4][1] < 1 || plant_map[4][1] > 4)
        {
        printf("ERROR!  Enter 1, 2, 3, or 4.\n\n");
        scanf("%d", &plant_map[4][1]);
        }

printf("Select the total number of engines online\n");
printf("during cruise operation.\n\n");
printf("1  One\n");
printf("2  Two\n");
printf("3  Three\n");
printf("4  Four\n\n");
scanf("%d", &plant_map[4][0]);

while (plant_map[4][0] < 1 || plant_map[4][0] > 4)
        {
        printf("ERROR!  Enter 1, 2, 3, or 4.\n\n");
        scanf("%d", &plant_map[4][0]);
        }

/* ------------------------------------------------------------------------------- */
/* ----- PDSS option only considered if all engine type[s] are only  */
/* ----- gas turbine.  For electric drive this is always true.                */
/* ------------------------------------------------------------------------------- */


printf("Will PDSS be driven off any of the gas turbine\n");
printf("output shafts.\n\n");
printf("0  NO\n");
printf("1  Yes\n");
scanf("%d", &plant_map[6][0]);
while (plant_map[6][0] < 0 || plant_map[6][0] > 1)
  {
  printf("ERROR!  Enter 0 or 1.\n\n");
  scanf("%d", &plant_map[6][0]);
  }


if (plant_map[6][0] == 1)
                        {
  /* ------------------------------------------------------------------ */
  /* Incorporate logic so that the minimum number of */
  /* PDSS units allowed is equal to the number of      */
  /* engines online during cruise.                      */
  /* ------------------------------------------------------------------ */

  n_engine_cruising = plant_map[4][0] + plant_map [5][0];

  if(n_engine_cruising == 1)
   {
   printf("Select the total number of pdss units desired.\n");
   printf("1  One pdss unit.\n");
   printf("2  Two pdss units.\n");
```

```
printf("3  Three pdss units.\n");
printf("4  Four pdss units.\n");
scanf("%d", &plant_map[6][1]);

while (plant_map[6][1] > plant_map[4][1] + plant_map[5][0])
 {
 printf("ERROR!  Ensure the total number of PDSS units is less\n");
 printf("than the total number of gas turbines. Re-enter.\n\n");
 scanf("%d", &plant_map[6][1]);
 }

  while (plant_map[6][1] < 1 || plant_map[6][1] > 4)
            {
            printf("ERROR!  Enter 1, 2, 3, or 4.\n\n");
            scanf("%d", &plant_map[6][1]);
            }
}

if(n_engine_cruising == 2)
{
printf("Select the total number of pdss units desired.\n");
printf("2  Two pdss units.\n");
printf("3  Three pdss units.\n");
printf("4  Four pdss units.\n");
scanf("%d", &plant_map[6][1]);

while (plant_map[6][1] > plant_map[4][1] + plant_map[5][0])
 {
 printf("ERROR!  Ensure the total number of PDSS units is less\n");
 printf("than the total number of gas turbines. Re-enter.\n\n");
 scanf("%d", &plant_map[6][1]);
 }

   while (plant_map[6][1] < 2 || plant_map[6][1] > 4)
   {
   printf("ERROR!  Enter 2, 3, or 4.\n\n");
   scanf("%d", &plant_map[6][1]);
   }
}

if(n_engine_cruising == 3)
{
 printf("Select the total number of pdss units desired.\n");
printf("3  Three pdss units.\n");
printf("4  Four pdss units.\n");
scanf("%d", &plant_map[6][1]);

while (plant_map[6][1] > plant_map[4][1] + plant_map[5][0])
 {
 printf("ERROR!  Ensure the total number of PDSS units is less\n");
 printf("than the total number of gas turbines. Re-enter.\n\n");
 scanf("%d", &plant_map[6][1]);
 }
```

```
        while (plant_map[6][1] < 3 || plant_map[6][1] > 4)
          {
          printf("ERROR!  Enter 3 or 4.\n\n");
          scanf("%d", &plant_map[6][1]);
          }
       }
      }


}

void mech_elec_hybrid_drive(void)
{

  /* Set the number of propulsors used for cruise = 2. */

  plant_map[0][0] = 2;

  /* Set the number of propulsors used for boost = 2. */

  plant_map[0][1] = 2;


  printf("Select one of the propulsor options.\n\n");
  printf("1  FPP\n");
  printf("2  CRP\n");
  printf("3  Contra-rotating\n");
  printf("4  Preswirl Stator\n");
  printf("5  Ducted FPP\n");
  printf("6  Ducted CRP\n");
  printf("7  Ducted Contra-rotating\n");
  printf("8  Ducted Preswirl Stator\n");
  printf("9  Waterjet\n\n");
  scanf("%d", &plant_map[1][0]);

  while (plant_map[1][0] < 1 || plant_map[1][0] > 10)
        {
        printf("ERROR!  Enter 1 through 9.\n\n");
        scanf("%d", &plant_map[1][0]);
        }

/* ------------------------------------------------------------------------ */
/* Specify the type of transmission for the mechanical drive. */
/* ------------------------------------------------------------------------ */
  /* Set the mechanical transmission type = LTDR. */

  plant_map[2][1] = 2;


/* ------------------------------------------------------------------------ */
/* Specify the type of transmission for the electric hybrid drive. */
/* Set the electric hybrid transmission type = electric.            */
/* ------------------------------------------------------------------------ */
```

```
  plant_map[1][1] = 1;

/* ------------------------------------------------------------------------- */
/* -----  Specify the engine type[s] and arrangements.  ---------- */
/* ------------------------------------------------------------------------- */

  printf("The boost engine type is LM-2500.  The program\n");
  printf("already assumes one LM-2500 ir  its directly into\n");
  printf("each mechanical drive LTDR gear.  You must decide\n");
  printf("if a boost engine will also be on the electric hybrid drive.\n");

  printf("Select one of the boost engine types for the.\n");
  printf("electric hybrid drive.\n");
  printf("0  None\n");
  printf("1  LM-2500 Gas Turbine\n\n");
  scanf("%d", &plant_map[3][1]);

  while (plant_map[3][1] != 0 && plant_map[3][1] != 1)
     {
     printf("ERROR!  Enter 0. or 1.\n\n");
     scanf("%d", &plant_map[3][1]);
     }
  if (plant_map[3][1] == 1)
    {
    plant_map[4][1] = 3;
    }
  else
    {
    plant_map[3][1] = 1;
    plant_map[4][1] = 2;
    }

  /* Set the cruise engine type to be PC4.2.          */
  /* Program assumes it will not be used for boost. */

  plant_map[3][0] = 2;
  plant_map[5][0] = 1;

}



void multiple_waterjet(void)
{
  int n_engine_cruising;

  printf("Select the number of waterjets used for cruise.\n");

  printf("2  Two waterjets.\n");
  printf("4  Four waterjets.\n");
  printf("6  Six waterjets.\n");
  scanf("%d", &plant_map[0][0]);

  while (plant_map[0][0] != 2 && plant_map[0][0] != 4)
```

```
    {
    if (plant_map[0][0] != 6)
        {
        printf("ERROR!  Enter 2, 4, or 6.\n\n");
        scanf("%d", &plant_map[0][0]);
        }
    else
        {
        break;
        }
    }

  printf("Select the number of waterjets used for maximum boost.\n");

  printf("6  Six waterjets.\n");
  printf("8  Eight waterjets.\n");
  scanf("%d", &plant_map[0][1]);

  while (plant_map[0][1] != 6 && plant_map[0][1] != 8)
    {
    printf("ERROR!  Enter 6 or 8.\n\n");
    scanf("%d", &plant_map[0][1]);
    }

  plant_map[1][0] = 9;
  plant_map[2][0] = 4;
  plant_map[3][1] = 5;
  plant_map[4][0] = plant_map[0][0];
  plant_map[4][1] = plant_map[0][1];

/* ------------------------------------------------------------ */
/* ----- PDSS option                                            */
/* ------------------------------------------------------------ */

  printf("Will PDSS be driven off of the gas turbine\n");
  printf("output shafts.\n\n");
  printf("0  NO\n");
  printf("1  Yes\n");
  scanf("%d", &plant_map[6][0]);
  while (plant_map[6][0] < 0 || plant_map[6][0] > 1)
    {
    printf("ERROR!  Enter 0 or 1.\n\n");
    scanf("%d", &plant_map[6][0]);
    }

  if (plant_map[6][0] == 1)
    {

      /* ------------------------------------------------------ */
      /* Incorporate logic so that the minimum number of */
      /* PDSS units allowed is equal to the number of   */
      /* engines online during cruise.                  */
      /* ------------------------------------------------------ */
```

```
    n_engine_cruising = plant_map[4][0] + plant_map [5][0];


    printf("Select the total number of pdss units desired.\n");
    printf("1  One pdss unit.\n");
    printf("2  Two pdss units.\n");
    printf("3  Three pdss units.\n");
    printf("4  Four pdss units.\n");
    printf("6  Six pdss units.\n");
    printf("8  Eight pdss units.\n");
    scanf("%d", &plant_map[6][1]);

    while (plant_map[6][1] > plant_map[4][1] + plant_map[5][0])
      {
      printf("ERROR!  Ensure the total number of PDSS units is less\n");
      printf("than the total number of gas turbines.\n\n");
      scanf("%d", &plant_map[6][1]);
      }

    while (plant_map[6][1] < n_engine_cruising)
      {
      printf("ERROR!  Ensure the total number of PDSS units is\n");
      printf(">= the total number of gas turbines used at cruise.\n\n");
      scanf("%d", &plant_map[6][1]);
      }

    while (plant_map[6][1] < 1 || plant_map[6][1] > 8)
      {
      if(plant_map[6][1] != 6 && plant_map[6][1] != 8)

         {
         printf("ERROR!  Enter 1, 2, 3, 4, 6, or 8.\n\n");
        scanf("%d", &plant_map[6][1]);
         }
      else
        {
        break;
        }
      }
    }

}

/* ---------------------------------------------------------------- */
/* ---------------------------------------------------------------- */
/* ---------------------------------------------------------------- */

void main_engine_specs(boost_engine_specs)

double boost_engine_specs[14];  /* Declare engine arrays. */


{
double boost_engine_options[14][5] =
```

```
 {
 {26250.0, 26400.0, 26060.0, 26250.0, 6000.0},
 {3600.0, 3600.0, 400.0, 3600.0, 1800.0},
 {1200.0, 1200.0, 125.0, 1200.0, 900.0},
 {0.0, 0.0, 16.0, 0.0, 0.0},
 {1.0, 2.0, 3.0, 1.0, 5.0},
 {59000.0, 120000.0, 639340.0, 59000.0, 15000.0},
 {493.2, 494.5, 322.9, 493.2, 364.5},
 {799.0, 801.1, 500.5, 799.0, 590.5},
 {26.5, 26.5, 42.7, 26.5, 15.8},
 {8.7, 8.7, 17.0, 8.7, 5.7},
 {10.4, 22.2, 26.2, 10.4, 7.7},
 {119.7, 119.7, 12.6, 119.7, 37.3},
 {162.5, 162.5, 19.6, 162.5, 82.3},
 {4.5, 6.5, 7.8, 4.5, 3.5},
 };


int i, j;

for (i = 1; i < 6; i++)
 {
 if(i == plant_map[3][1])
  {
  for(j = 0; j < 14; j++)
   {
   boost_engine_specs[j] = boost_engine_options[j][(i-1)];
   }
  break;
  }
 }


}


/* ---------------------------------------------------------------- */
/* ---------------------------------------------------------------- */

void endurance_engine_specs(boost_engine_specs,cruise_engine_specs)

double boost_engine_specs[14],  /* Declare engine arrays. */
cruise_engine_specs[14];

{
double cruise_engine_options[14][3] =
 {
  {26400.0, 16290.0, 7370.0},
  {3600.0, 400.0, 520.0},
  {1200.0, 125.0, 200.0},
  {0.0, 10.0, 10.0},
  {2.0, 3.0, 4.0},
```

```
  {120000.0, 425000.0, 121275.0},
  {494.5, 255.3, 171.7},
  {801.1, 395.7, 266.1},
  {26.5, 34.2, 20.0},
  {8.7, 17.0, 11.0},
  {22.2, 25.2, 12.3},
  {119.7, 12.6, 10.1},
  {162.5, 19.6, 11.0},
  {6.5, 5.43, 2.92},
  };

int i, j;

if(plant_map[3][0] == 0)
  {
  for(j = 0; j < 14; j++)
     {
     cruise_engine_specs[j] = boost_engine_specs[j];
     }
  }
else
  {
  for (i = 1; i < 4; i++)
    {
    if(i == plant_map[3][0])
     {
     for(j = 0; j < 14; j++)
       {
       cruise_engine_specs[j] = cruise_engine_options[j][(i-1)];
       }
     break;
     }
    }
  }

}


/* ---------------------------------------------------------------- */
/* ---------------------------------------------------------------- */

double fuel_load (ehp_cruise, qpc_cruise, xmission_eff_cruise,
                  rpm_prop_cruise, gear_ratio,
                cruise_engine_specs,main_engine_max_pwr)

double ehp_cruise,
qpc_cruise,
xmission_eff_cruise,
rpm_prop_cruise,
gear_ratio;

double cruise_engine_specs[14];

double main_engine_max_pwr;
```

```
{

double weight_fuel;

double avg_endur_bhp_per_engine,
rpm_engine_cruise,
uncorrected_sfc_per_engine,
specified_fuel_rate,
avg_endur_fuel_rate,
propulsion_endur_fuel_weight,
f_ratio,
f_factor;

double avg_24hr_elec_load = AVG_ELEC_LOAD,
k34_bhp_per_eng,
no_bleed_sfc,
bleed_sfc,
avg_sfc,
elec_endur_fuel_weight,
n_online_eng,
pdss_eff = 1.0,
hp_per_pdss,
hp_per_pdss_no_bld,
hp_per_pdss_bld,
avg_endur_bhp_per_engine_no_bld,
avg_endur_bhp_per_engine_bld,
uncorrected_sfc_per_engine_no_bld,
uncorrected_sfc_per_engine_bld,
specified_fuel_rate_no_bld,
specified_fuel_rate_bld,
avg_endur_fuel_rate_no_bld,
avg_endur_fuel_rate_bld;


int i;


/* ---------------------------------------- */
/* Perform the no PDSS loop.      */
/* ---------------------------------------- */

if(plant_map[6][0] == 0)
 {
 /* The program will assume that the total power required  */
 /* for cruise is split equally amongst all cruise engines. */

 n_online_eng = (plant_map[4][0] + plant_map[5][0]);

 avg_endur_bhp_per_engine = 1.10 * ehp_cruise /
                    (qpc_cruise * xmission_eff_cruise *
                     n_online_eng);

/* ---------------------------------------------------------------------------- */
```

```
/* Check that the designated cruise engine will meet the cruise   */
/* power requirements.  If the cruise engine type is a diesel,    */
/* allow for the increase in power rating as defined in the loop. */
/* ------------------------------------------------------------------------- */

 if(cruise_engine_specs[0] < avg_endur_bhp_per_engine)
  {
  if(plant_map[3][0] != 2 && plant_map[3][0] != 3)
   {
   printf("ERROR! The number of engines you selected\n");
   printf("for cruise does not meet the power requirement.\n");
   printf("The total engine BHP requirement for cruise\n");
   printf("is = %7.1f BHP.\n", (avg_endur_bhp_per_engine *
                                (plant_map[4][0]+plant_map[5][0])));
   printf("Re-run the program with more cruise engine power.\n\n");
   return(-1.0);

   }

  if(plant_map[3][0] == 2)
   {
   cruise_engine_specs[3] = 12.0;
   cruise_engine_specs[0] = cruise_engine_specs[0] *12.0/10.0;
   if(cruise_engine_specs[0] > avg_endur_bhp_per_engine)
    {
    cruise_engine_specs[4] = 3.0;
    cruise_engine_specs[5] = 507060.0;
    cruise_engine_specs[6] = 279.6;
    cruise_engine_specs[7] = 433.4;
    cruise_engine_specs[8] = 37.0;
    cruise_engine_specs[9] = 17.0;
    cruise_engine_specs[10] = 25.2;
    cruise_engine_specs[11] = 12.6;
    cruise_engine_specs[12] = 19.6;
    cruise_engine_specs[13] = 6.23;
    goto pc42end;
    }

   else
    {
    cruise_engine_specs[3] = 14.0;
    cruise_engine_specs[0] = cruise_engine_specs[0] *14.0/12.0;
    }
   if(cruise_engine_specs[0] > avg_endur_bhp_per_engine)
    {
    cruise_engine_specs[4] = 3.0;
    cruise_engine_specs[5] = 577600.0;
    cruise_engine_specs[6] = 302.0;
    cruise_engine_specs[7] = 468.1;
    cruise_engine_specs[8] = 39.7;
    cruise_engine_specs[9] = 17.0;
    cruise_engine_specs[10] = 26.2;
    cruise_engine_specs[11] = 12.6;
    cruise_engine_specs[12] = 19.6;
```

```
    cruise_engine_specs[13] = 7.10;
    }
   else
    {
    printf("ERROR! The number of diesel engine[s] you selected\n");
    printf("for cruise do not meet the power requirement.\n");
    printf("The total engine BHP requirement for cruise\n");
    printf("is = %7.1f BHP.\n", (avg_endur_bhp_per_engine *
                          (plant_map[4][0]+plant_map[5][0])));
    printf("The largest available PC4.2 cruise diesel\n");
    printf("has 14 cylinders and is rated at 22800 BHP.  \n");
    printf("Re-run the program with more cruise engine power.\n\n");
    return(-1.0);
    }
pc42end::
   }

  if(plant_map[3][0] == 3)
  {
  cruise_engine_specs[3] = 12.0;
  cruise_engine_specs[0] = cruise_engine_specs[0] *12.0/10.0;
  if(cruise_engine_specs[0] > avg_endur_bhp_per_engine)
   {
   cruise_engine_specs[4] = 4.0;
   cruise_engine_specs[5] = 145530.0;
   cruise_engine_specs[6] = 188.1;
   cruise_engine_specs[7] = 291.5;
   cruise_engine_specs[8] = 24.2;
   cruise_engine_specs[9] = 11.0;
   cruise_engine_specs[10] = 14.9;
   cruise_engine_specs[11] = 10.1;
   cruise_engine_specs[12] = 11.0;
   cruise_engine_specs[13] = 3.40;
   goto pc26end;
   }

  else
   {
   cruise_engine_specs[3] = 14.0;
   cruise_engine_specs[0] = cruise_engine_specs[0] *14.0/12.0;
   }

  if(cruise_engine_specs[0] > avg_endur_bhp_per_engine)
   {
   cruise_engine_specs[4] = 4.0;
   cruise_engine_specs[5] = 165375.0;
   cruise_engine_specs[6] = 203.2;
   cruise_engine_specs[7] = 314.9;
   cruise_engine_specs[8] = 26.6;
   cruise_engine_specs[9] = 11.0;
   cruise_engine_specs[10] = 14.9;
   cruise_engine_specs[11] = 10.1;
   cruise_engine_specs[12] = 11.0;
   cruise_engine_specs[13] = 3.38;
```

```
      goto pc26end;
      }

    else
     {
     cruise_engine_specs[3] = 16.0;
     cruise_engine_specs[0] = cruise_engine_specs[0] *16.0/14.0;
     }

    if(cruise_engine_specs[0] > avg_endur_bhp_per_engine)
     {
     cruise_engine_specs[4] = 4.0;
     cruise_engine_specs[5] = 183015.0;
     cruise_engine_specs[6] = 217.2;
     cruise_engine_specs[7] = 336.6;
     cruise_engine_specs[8] = 29.0;
     cruise_engine_specs[9] = 11.0;
     cruise_engine_specs[10] = 14.9;
     cruise_engine_specs[11] = 10.1;
     cruise_engine_specs[12] = 13.1;
     cruise_engine_specs[13] = 3.87;
     }

    else
     {
      printf("ERROR! The number of diesel engine[s] you selected\n");
     printf("for cruise do not meet the power requirement.\n");
     printf("The total engine BHP requirement for cruise\n");
     printf("is = %7.1f BHP.\n", (avg_endur_bhp_per_engine *
                          (plant_map[4][0]+plant_map[5][0])));
     printf("The largest available PC2.6 cruise diesel\n");
     printf("has 16 cylinders and is rated at 11792 BHP.  \n");
     printf("Re-run the program with more cruise engine power.\n\n");
     return(-1.0);
     }
  pc26end::

    }
   }


/* ------------------------------------------------------- */
/* ------------------------------------------------------- */

  rpm_engine_cruise = rpm_prop_cruise * gear_ratio;

  uncorrected_sfc_per_engine =
    engine_sfc(avg_endur_bhp_per_engine, rpm_engine_cruise,
                                  cruise_engine_specs[3],  cruise_engine_specs[4]);

  /* ------------------------------------------------------- */

  if(uncorrected_sfc_per_engine == -1.0)
    {
```

```
 return(-1.0);
 }
/* ------------------------------------------------------- */


f_ratio = avg_endur_bhp_per_engine * (plant_map[4][0] + plant_map[5][0])/
       (main_engine_max_pwr * plant_map[4][1] +
                                    cruise_engine_specs[0] * plant_map[5][1]);

if (f_ratio <= (1.0/3.0))
 {
 f_factor = 1.04;
 }
else
 {
 f_factor = 1.03;
 }

if (f_ratio >= (2.0/3.0))
 {
 f_factor = 1.02;
 }

specified_fuel_rate = f_factor * uncorrected_sfc_per_engine;

avg_endur_fuel_rate = 1.05 * specified_fuel_rate;

/* ------------------------------------------------------------------- */
/* The factor TPA in the propulsion_endur_fuel_weight */
/* formula accounts for tail pipe allowance.          */
/* ------------------------------------------------------------------- */


propulsion_endur_fuel_weight =
 RANGE/ENDUR_SPD * TPA * avg_endur_fuel_rate *
 avg_endur_bhp_per_engine * n_online_eng/
 2240.0;

/* ------------------------------------------------------------------- */
/* Calculate the electric fuel wt.  For the DDG, the   */
/* avg_24hr_elec_load = 2525.0.  With two SSGTG's      */
/* rated at 2500 KW, assume both will be online        */
/* to provide bleed air and equally split elec load    */
/* Note: .7457 KW = 1HP.                               */
/* ------------------------------------------------------------------- */

k34_bhp_per_eng = avg_24hr_elec_load/2.0/0.7457;

no_bleed_sfc = 3.12*pow10(-19) * pow(k34_bhp_per_eng,5) +
          1.332*pow10(-16) * pow(k34_bhp_per_eng,4) -
          4.17*pow10(-11) * pow(k34_bhp_per_eng,3) +
          3.06*pow10(-7) * pow(k34_bhp_per_eng,2) -
          9.14*pow10(-4) * k34_bhp_per_eng + 1.58;
```

```
bleed_sfc =   2.34*pow10(-19) * pow(k34_bhp_per_eng,5) -
          5.27*pow10(-15) * pow(k34_bhp_per_eng,4) +
          1.24*pow10(-11) * pow(k34_bhp_per_eng,3) +
          1.35*pow10(-7) * pow(k34_bhp_per_eng,2) -
          7.43*pow10(-4) * k34_bhp_per_eng + 1.67;

avg_sfc = (no_bleed_sfc * (1.0 - ENDUR_PCT_BLD) +
          bleed_sfc * ENDUR_PCT_BLD);



/* ------------------------------------------------------------------ */
/* The factor TPA in the elec_endur_fuel_weight      */
/* formula accounts for tail pipe allowance.         */
/* 1.05 = plant deterioration factor.                */
/* 1.03 = precalculated f_factor.                    */
/* ------------------------------------------------------------------ */

elec_endur_fuel_weight =
 RANGE/ENDUR_SPD * TPA * (k34_bhp_per_eng * 2.0) *
 (1.05 * 1.03 * avg_sfc)/2240.0;


printf("\n\nNumber of engines used for cruise = %3.1f\n", n_online_eng);
printf("avg_endur_bhp_per_eng = %8.2f BHP\n", avg_endur_bhp_per_engine);
printf("uncorrect_sfc_per_eng = %6.5f LB/HP/HR\n",
                        uncorrected_sfc_per_engine);
printf("propulsion f_factor = %5.4f\n", f_factor);
printf("Tail Pipe Allowance = %6.3f\n", TPA);
printf("pct of endurance time w/bleed = %6.3f\n", ENDUR_PCT_BLD);
printf("avg_endur_fuel_rate = %6.5f LB/HP/HR\n", avg_endur_fuel_rate);
printf("K34_BHP_per_eng = %8.2f\n", k34_bhp_per_eng);
printf("The elec avg_sfc = %7.5f LB/HP/HR.\n\n", avg_sfc);


printf("The electric fuel wt = %7.1f LTONS.\n",
                    elec_endur_fuel_weight);

printf("The propulsion fuel wt = %7.1f LTONS.\n",
                    propulsion_endur_fuel_weight);

weight_fuel = propulsion_endur_fuel_weight + elec_endur_fuel_weight;

} /* note this brace is for the end of the no PDSS loop */



/* ------------------------------------------------------------------ */
/* ------------------------------------------------ */
/* Perform the  overall PDSS loop.  */
/* ------------------------------------------------ */
```

```
if(plant_map[6][0] == 1)
 {
 /* The program will assume that the total power required  */
 /* for cruise is split equally amongst all cruise engines. */


 n_online_eng = plant_map[4][0] + plant_map[5][0];


 /* ------------------------------------------------------------------- */
 /* ------------------------------------------------------------------- */
 /* Perform this if loop for PDSS if there's only one cruise    */
 /* engine online.                                               */
 /* ------------------------------------------------------------------- */

 if(n_online_eng == 1)
  {
  /* ------------------------------------------------------------------- */
  /* Assume that the electric load is split equally    */
  /* between one PDSS and one SSGTG.  The SSGTG will   */
  /* supply bleed air has determined by ENDUR_PCT_BLD.  */
  /* ------------------------------------------------------------------- */

  hp_per_pdss = avg_24hr_elec_load/
                (2.0 * 0.7457 * pdss_eff);

  k34_bhp_per_eng = hp_per_pdss;

  avg_endur_bhp_per_engine = 1.10 * ehp_cruise /
                (qpc_cruise * xmission_eff_cruise) +
                 hp_per_pdss;

/* ------------------------------------------------------------------- */
/* Check that the designated cruise engine will meet the cruise  */
/* power + PDSS load requirements.                              */
/* ------------------------------------------------------------------- */

  if(cruise_engine_specs[0] < avg_endur_bhp_per_engine)
   {
   printf("ERROR! The number of engines you selected\n");
   printf("for cruise does not meet the power requirement.\n");
   printf("The total cruise power + PDSS load requirement\n");
   printf("is = %7.1f BHP.\n", avg_endur_bhp_per_engine);
   printf("Re-run the program with more cruise engine power.\n\n");
   return(-1.0);
   }

/* ------------------------------------------------------------ */

  rpm_engine_cruise = rpm_prop_cruise * gear_ratio;

  uncorrected_sfc_per_engine =
   engine_sfc(avg_endur_bhp_per_engine, rpm_engine_cruise,
                                cruise_engine_specs[3],  cruise_engine_specs[4]);
```

```
/* ---------------------------------------------------------- */

if(uncorrected_sfc_per_engine == -1.0)
 {
 return(-1.0);
 }
/* ---------------------------------------------------------- */


f_ratio = avg_endur_bhp_per_engine * n_online_eng/
      (main_engine_max_pwr * plant_map[4][1] +
                                  cruise_engine_specs[0] * plant_map[5][1]);

if (f_ratio <= (1.0/3.0))
 {
 f_factor = 1.04;
 }
else
 {
 f_factor = 1.03;
 }

if (f_ratio >= (2.0/3.0))
 {
 f_factor = 1.02;
 }

specified_fuel_rate = f_factor * uncorrected_sfc_per_engine;

avg_endur_fuel_rate = 1.05 * specified_fuel_rate;

/* ---------------------------------------------------------------------- */
/* The factor TPA in the propulsion_endur_fuel_weight */
/* formula accounts for tail pipe allowance.          */
/* ---------------------------------------------------------------------- */

propulsion_endur_fuel_weight =
 RANGE/ENDUR_SPD * TPA * avg_endur_fuel_rate *
 avg_endur_bhp_per_engine * (plant_map[4][0] + plant_map[5][0])/
 2240.0;

/* ---------------------------------------------------------------------- */
/* Calculate the electric fuel wt.  For the DDG, the   */
/* avg_24hr_elec_load = 2525.0.  With two SSGTG's      */
/* rated at 2500 KW, assume two will be online equally */
/* sharing the load.  Note: .7457 KW = 1HP.            */
/* ---------------------------------------------------------------------- */


no_bleed_sfc = 3.12*pow10(-19) * pow(k34_bhp_per_eng,5) +
          1.332*pow10(-16) * pow(k34_bhp_per_eng,4) -
          4.17*pow10(-11) * pow(k34_bhp_per_eng,3) +
          3.06*pow10(-7) * pow(k34_bhp_per_eng,2) -
```

```
        9.14*pow10(-4) * k34_bhp_per_eng + 1.58;

bleed_sfc =   2.34*pow10(-19) * pow(k34_bhp_per_eng,5) -
        5.27*pow10(-15) * pow(k34_bhp_per_eng,4) +
        1.24*pow10(-11) * pow(k34_bhp_per_eng,3) +
        1.35*pow10(-7) * pow(k34_bhp_per_eng,2) -
        7.43*pow10(-4) * k34_bhp_per_eng + 1.67;

avg_sfc = (no_bleed_sfc * (1.0 - ENDUR_PCT_BLD) +
        bleed_sfc * ENDUR_PCT_BLD);


/* ------------------------------------------------------------------ */
/* The factor TPA in the elec_endur_fuel_weight      */
/* formula accounts for tail pipe allowance.        */
/* 1.05 = plant deterioration factor.               */
/* 1.03 = precalculated f_factor.                   */
/* ------------------------------------------------------------------ */

elec_endur_fuel_weight =
 RANGE/ENDUR_SPD * TPA * (k34_bhp_per_eng * 2.0) *
 (1.05 * 1.03 * avg_sfc)/2240.0;

printf("\n\nNumber of engines used for cruise = %3.1f\n", n_online_eng);
printf("HP per PDSS = %8.2f BHP\n", hp_per_pdss);
printf("K34_BHP_per_eng = %8.2f BHP\n", k34_bhp_per_eng);
printf("avg_endur_bhp_per_eng = %8.2f BHP\n", avg_endur_bhp_per_engine);
printf("uncorrect_sfc_per_eng = %6.5f LB/HP/HR\n",
                        uncorrected_sfc_per_engine);
printf("propulsion f_factor = %5.4f\n", f_factor);
printf("Tail Pipe Allowance = %6.3f\n", TPA);
printf("pct of endurance time w/bleed = %6.3f\n", ENDUR_PCT_BLD);
printf("avg_endur_fuel_rate = %6.5f LB/HP/HR\n", avg_endur_fuel_rate);
printf("The elec avg_sfc = %7.5f LB/HP/HR.\n\n", avg_sfc);

printf("The electric fuel wt = %7.1f LTONS.\n",
                    elec_endur_fuel_weight);

printf("The propulsion fuel wt = %7.1f LTONS.\n\n",
                    propulsion_endur_fuel_weight);

weight_fuel = propulsion_endur_fuel_weight + elec_endur_fuel_weight;

}   /* note this brace is for the end of n_online_eng=1 loop */



/* ------------------------------------------------------------------------ */
/* ------------------------------------------------------------------------ */
/* Perform this if loop for PDSS if there's more than one cruise */
/* engine online.                                                 */
/* ------------------------------------------------------------------------ */
```

```
if(n_online_eng > 1)
  {
  /* ------------------------------------------------------------------------ */
  /* Assume that the electric load is split equally     */
  /* between the online PDSS units and the one SSGTG,   */
  /* when that SSGTG is supplying bleed air has         */
  /* determined by ENDUR_PCT_BLD.  When the SSGTG is not*/
  /* required to supply bleed air has determined by     */
  /* (1.0 - ENDUR_PCT_BLD), assume the SSGTG is offline.*/
  /* Hence the electric load is split between the       */
  /* online PDSS units only.                            */
  /* ------------------------------------------------------------------------ */

  hp_per_pdss_no_bld = avg_24hr_elec_load/
                       (n_online_eng * 0.7457 * pdss_eff);


  hp_per_pdss_bld = avg_24hr_elec_load/
                    ((n_online_eng + 1.0) * 0.7457 * pdss_eff);

  k34_bhp_per_eng = hp_per_pdss_bld;

  avg_endur_bhp_per_engine_no_bld = 1.10 * ehp_cruise /
        (n_online_eng * qpc_cruise * xmission_eff_cruise) +
         hp_per_pdss_no_bld;

  avg_endur_bhp_per_engine_bld = 1.10 * ehp_cruise /
        (n_online_eng * qpc_cruise * xmission_eff_cruise) +
         hp_per_pdss_bld;

/* ------------------------------------------------------------------------ */
/* Check that the designated cruise engine will meet the cruise   */
/* power + highest PDSS load requirements.                        */
/* ------------------------------------------------------------------------ */

  if(cruise_engine_specs[0] < avg_endur_bhp_per_engine_no_bld)
    {
    printf("ERROR! The number of engines you selected\n");
    printf("for cruise does not meet the power requirement.\n");
    printf("The total cruise power + PDSS load requirement\n");
    printf("is = %7.1f BHP.\n", avg_endur_bhp_per_engine);
    printf("Re-run the program with more cruise engine power.\n\n");
    return(-1.0);
    }

/* ------------------------------------------------------------- */

  rpm_engine_cruise = rpm_prop_cruise * gear_ratio;

  uncorrected_sfc_per_engine_no_bld =
   engine_sfc(avg_endur_bhp_per_engine_no_bld, rpm_engine_cruise,
                                    cruise_engine_specs[3],  cruise_engine_specs[4]);

  uncorrected_sfc_per_engine_bld =
```

```
 engine_sfc(avg_endur_bhp_per_engine_bld, rpm_engine_cruise,
                                  cruise_engine_specs[3],  cruise_engine_specs[4]);


/* ------------------------------------------------------- */

if(uncorrected_sfc_per_engine_no_bld == -1.0)
 {
 return(-1.0);
 }
if(uncorrected_sfc_per_engine_bld == -1.0)
 {
 return(-1.0);
 }
/* ------------------------------------------------------- */


f_ratio = avg_endur_bhp_per_engine_no_bld * n_online_eng/
      (main_engine_max_pwr * plant_map[4][1] +
                                  cruise_engine_specs[0] * plant_map[5][1]);

if (f_ratio <= (1.0/3.0))
 {
 f_factor = 1.04;
 }
else
 {
 f_factor = 1.03;
 }

if (f_ratio >= (2.0/3.0))
 {
 f_factor = 1.02;
 }

specified_fuel_rate_no_bld = f_factor *
                        uncorrected_sfc_per_engine_no_bld;

specified_fuel_rate_bld = f_factor *
                        uncorrected_sfc_per_engine_bld;


avg_endur_fuel_rate_no_bld = 1.05 * specified_fuel_rate_no_bld;

avg_endur_fuel_rate_bld = 1.05 * specified_fuel_rate_bld;

/* ------------------------------------------------------------------- */
/* The factor TPA in the propulsion_endur_fuel_weight */
/* formula accounts for tail pipe allowance.          */
/* ------------------------------------------------------------------- */

propulsion_endur_fuel_weight =

 RANGE/ENDUR_SPD/2240.0 * TPA * n_online_eng *
```

```
((ENDUR_PCT_BLD * avg_endur_fuel_rate_bld *
                    avg_endur_bhp_per_engine_bld) +

                    ( (1.0 - ENDUR_PCT_BLD) * avg_endur_fuel_rate_no_bld *
                    avg_endur_bhp_per_engine_no_bld));

/* ------------------------------------------------------------------ */
/* Calculate the electric fuel wt. For the DDG, the   */
/* avg_24hr_elec_load = 2525.0.  With one SSGTG       */
/* rated at 2500 KW, assume it will be online only    */
/* to provide bleed air.  When it is online it will   */
/* also share the elec load.  Note: .7457 KW = 1HP.   */
/* ------------------------------------------------------------------ */



bleed_sfc =   2.34*pow10(-19) * pow(k34_bhp_per_eng,5) -
          5.27*pow10(-15) * pow(k34_bhp_per_eng,4) +
          1.24*pow10(-11) * pow(k34_bhp_per_eng,3) +
          1.35*pow10(-7) * pow(k34_bhp_per_eng,2) -
          7.43*pow10(-4) * k34_bhp_per_eng + 1.67;


/* ------------------------------------------------------------------ */
/* The factor TPA in the elec_endur_fuel_weight       */
/* formula accounts for tail pipe allowance.          */
/* 1.05 = plant deterioration factor.                 */
/* 1.03 = precalculated f_factor.                     */
/* ------------------------------------------------------------------ */

elec_endur_fuel_weight =
 RANGE/ENDUR_SPD *ENDUR_PCT_BLD/2240.0 * TPA * k34_bhp_per_eng *
 (1.05 * 1.03 * bleed_sfc);


printf("\n\nNumber of engines used for cruise = %3.1f\n", n_online_eng);
printf("HP per PDSS w/no-bleed = %8.2f BHP\n", hp_per_pdss_no_bld);
printf("HP per PDSS w/bleed = %8.2f BHP\n", hp_per_pdss_bld);
printf("K34_BHP_per_eng w/bleed = %8.2f BHP\n", k34_bhp_per_eng);
printf("avg_endur_bhp_per_eng_no_bld = %8.2f BHP\n",
                    avg_endur_bhp_per_engine_no_bld);
printf("avg_endur_bhp_per_eng_bld = %8.2f BHP\n",
                    avg_endur_bhp_per_engine_bld);
printf("uncorrect_sfc_per_eng_no_bld = %6.5f LB/HP/HR\n",
                    uncorrected_sfc_per_engine_no_bld);
printf("uncorrect_sfc_per_eng_bld = %6.5f LB/HP/HR\n",
                    uncorrected_sfc_per_engine_bld);
printf("propulsion f_factor = %5.4f\n", f_factor);
printf("Tail Pipe Allowance = %6.3f\n", TPA);
printf("pct of endurance time w/bleed = %6.3f\n", ENDUR_PCT_BLD);
printf("avg_endur_fuel_rate_no_bld = %6.5f LB/HP/HR\n",
                     avg_endur_fuel_rate_no_bld);
printf("avg_endur_fuel_rate_bld = %6.5f LB/HP/HR\n",
```

```
                        avg_endur_fuel_rate_bld);
 printf("K34 w/bleed SFC = %7.5f LB/HP/HR.\n\n", bleed_sfc);

 printf("The electric fuel wt = %7.1f LTONS.\n",
                    elec_endur_fuel_weight);

 printf("The propulsion fuel wt = %7.1f LTONS.\n\n",
                    propulsion_endur_fuel_weight);

 weight_fuel = propulsion_endur_fuel_weight + elec_endur_fuel_weight;

 }        /* note this brace is for the end of n_online_eng>1 loop */


 }        /* note this brace is for the end of the PDSS loop */



return (weight_fuel);
}

/* ------------------------------------------------ */
/* ------------------------------------------------ */

double engine_sfc(bhp, rpm, number_cylinders, eng_type)


double bhp,
rpm,
number_cylinders,
eng_type;

{

double pct_bhp,
sfc,


if(eng_type == 1)
  {
  sfc = lm2500_map(bhp,rpm);
  }
if(eng_type == 2)
  {
  pct_bhp = bhp/26400.0;

  sfc = 0.2353 * pow(pct_bhp,-0.3485) +
      0.1237 * pow(pct_bhp, 1.4870);
  }
if(eng_type == 3)
  {
  sfc = pc42_sfc_map(bhp,rpm,number_cylinders);
  }
if(eng_type == 4)
```

```
 {
 sfc = pc26_sfc_map(bhp,rpm,number_cylinders);
 }
if(eng_type == 5)
 {
 sfc = -1.844 * pow(bhp,-2.0270) +
      6.633 * pow(bhp,-0.3178);
 }


return(sfc);
}

/* ------------------------------------------------- */
/* ------------------------------------------------- */


double lm2500_map(engine_bhp, engine_rpm)

double engine_bhp,
engine_rpm;

{
/* This function provides the LM2500 engine parameters of: */
/*  SFC = specific fuel consumption in Lb/Hp-Hr  */
/*  T8 = exhaust duct discharge total temp in deg R */
/*  W8 = exhaust duct discharge flow in Lb/sec */
/*  P8 = exhaust duct discharge total pressure in PSIA */
/*  CP8 =exhaust duct discharge specific heat in BTU/Lb-Deg. R */
/* It performs linear interpolation on a given engine BHP and NPT */
/* using the tabular engine performance map for 100 Deg. F.  It */
/* also accounts for three losses defined in the define statements. */


        unsigned int lower_row, /*bottom bhp bound for interpolation*/
        higher_row,          /*upper bhp bound for interpolation*/
        left_column,         /*left rpm bound for interpolation*/
        right_column,         /*right rpm bound for interpoaltion*/
        param;              /*counter used to recall parameters*/


        float bhp_ratio,     /*bhp ratio for linear interpolation*/
        rpm_ratio,          /*rpm ratio for linear interpolation*/
        param_lower_rpm,    /*engine parameter at desired bhp and*/
                                /*lower rpm interpolation bound*/
        param_upper_rpm;    /*engine parameter at desired bhp and*/
                               /*upper rpm interpolation bound*/

        /*declare the parameter correction factor variables for inlet,*/
        /*exhaust, and humidity losses*/

        double inlet_sfc_fac, inlet_t8_fac, inlet_w8_fac, inlet_p8_fac,
        inlet_cp8_fac, exhaust_sfc_fac, exhaust_t8_fac,
```

```
        exhaust_w8_fac, exhaust_p8_fac, exhaust_cp8_fac,
        humidity_sfc_fac, humidity_t8_fac, humidity_w8_fac,
        humidity_p8_fac, humidity_cp8_fac, sfc_factor, t8_factor,
        w8_factor, p8_factor, cp8_factor;

/* ------------------------------------------------------------ */
        /*declare and initialize the arrays*/

        /*declare the array that will output the five operating*/
        /*parameters of SFC, T8, W8, P8, and CP8 for the given */
        /* engine BHP and NPT.  The array will store the parameters*/
          /* in the order listed above.*/

        float parameter_engine [5];

        /* declare and initialize the array that defines the BHP */
        /* values for the tabular rows */

        float bhp [12] = {182.0, 1000.0, 3000.0, 5000.0, 10000.0,
                15000.0, 17500.0, 20000.0, 22500.0,
                25000.0, 27500.0, 30000.01};

        /* declare and initialize the array that defines the rpm */
        /* values for the tabular columns */

        float rpm [7] = {900.0, 1200.0, 1800.0, 2400.0, 3000.0,
                3300.0, 3600.01};

         * declare and initialize the array that maps the five */
        . * engine parameters. */

float engine_map [5][12][7] =
{
{
  {4.4071, 4.8394, 0.0, 0.0, 0.0, 0.0, 0.0},
  {1.6962, 1.6021, 1.6040, 1.7396, 1.9825, 2.1302, 2.3004},
 {.9678, .90312, .86855, .87937, .92378, .9569, .99805},
 {.85648, .73477, .6692, .66606, .68551, .70115, .72322},
 {0.0, .65875, .54401, .50957, .50801, .51213, .51859},
 {0.0, 0.0, .50343, .46102, .44704, .44637, .44825},
 {0.0, 0.0, .49543, .44506, .42874, .42696, .42701},
 {0.0, 0.0, .49107, .43374, .41471, .41157, .41097} ,
  {0.0, 0.0, 0.0, .4265, .40395, .40007, .39883},
  {0.0, 0.0, 0.0, .42436, .39543, .3902, .38784},
  {0.0, 0.0, 0.0, 0.0, .39174, .38384, .38035},
  {0.0, 0.0, 0.0, 0.0, .39124, .38182, .37709}
  },
  {
  {1278.0, 1284.6, 0.0, 0.0, 0.0, 0.0, 0.0},
  {1285.4, 1287.9, 1303.2, 1321.8, 1341.2, 1354.3, 1365.3},
  {1239.9, 1266.1, 1317.3, 1341.7, 1356.8, 1362.1, 1367.1},
  {1326.5, 1268.9, 1277.5, 1308.5, 1335.5, 1347.5, 1360.4},
  {0.0, 1448.6, 1351.5, 1333.3, 1354.8, 1369.2, 1383.0},
  {0.0, 0.0, 1438.6, 1398.5, 1393.4, 1403.0, 1414.8},
```

```
{0.0, 0.0, 1489.0, 1423.0, 1413.7, 1419.8, 1427.7},
{0.0, 0.0, 1543.2, 1449.7, 1432.2, 1434.6, 1441.3},
{0.0, 0.0, 0.0, 1482.6, 1450.7, 1452.1, 1458.1},
{0.0, 0.0, 0.0, 1533.0, 1470.3, 1466.6, 1469.4},
{0.0, 0.0, 0.0, 0.0, 1508.9, 1491.4, 1488.1},
{0.0, 0.0, 0.0, 0.0, 1561.0, 1534.2, 1524.6}
},
{
{22.577, 22.564, 0.0, 0.0, 0.0, 0.0, 0.0},
{42.681, 39.839, 39.058, 41.82, 47.104, 50.103, 53.693},
{73.76, 65.353, 57.942, 56.859, 59.057, 61.114, 63.768},
{93.72, 84.48, 74.214, 71.539, 70.446, 71.301, 72.832},
{0 0, 115.82, 102.0, 95.618, 92.482, 91.733, 91.667},
{0.0, 0.0, 123.02, 114.2, 109.91, 108.34, 107.39},
{0.0, 0.0, 132.18, 122.71, 117.47, 115.86, 114.74},
{0.0, 0.0, 140.33, 130.66, 124.79, 122.99, 121.71},
{0.0, 0.0, 0.0, 137.85, 131.85, 129.67, 128.12},
{0.0, 0.0, 0.0, 143.51, 138.45, 136.19, 134.45},
{0.0, 0.0, 0.0, 0.0, 143.49, 141.79, 140.27},
{0.0, 0.0, 0.0, 0.0, 147.39, 146.09, 144.76}
},
{
{14.699, 14.699, 0.0, 0.0, 0.0, 0.0, 0.0},
{14.71, 14.706, 14.707, 14.709, 14.714, 14.714, 14.716},
{14.734, 14.725, 14.72, 14.72, 14.722, 14.723, 14.726},
{14.761, 14.745, 14.735, 14.732, 14.733, 14.734, 14.734},
{0.0, 14.804, 14.773, 14.765, 14.76, 14.759, 14.762},
{0.0, 0.0, 14.816, 14.797, 14.789, 14.786, 14.788},
{0.0, 0.0, 14.841, 14.814, 14.804, 14.802, 14.799},
{0.0, 0.0, 14.863, 14.833, 14.82, 14.815, 14.814},
{0.0, 0.0, 0.0, 14.852, 14.835, 14.832, 14.828},
{0.0, 0.0, 0.0, 14.871, 14.852, 14.845, 14.843},
{0.0, 0.0, 0.0, 0.0, 14.866, 14.861, 14.857},
{0.0, 0.0, 0.0, 0.0, 14.883, 14.879, 14.873}
},
{
{.2614, .26166, 0.0, 0.0, 0.0, 0.0, 0.0},
{.26176, .26189, .26251, .2632, .26392, .26441, .26482},
{.26015, .26127, .26342, .26441, .26498, .26516, .26533},
{.26381, .26158, .26205, .26336, .26445, .26491, .2654},
{0.0, .26917, .2655, .26488, .26579, .26639, .26694},
{0.0, 0.0, .26935, .26791, .26779, .2682, .26869},
{0.0, 0.0, .2715, .26906, .26879, .26907, .26941},
{0.0, 0.0, .27378, .27029, .26971, .26984, .27014},
{0.0, 0.0, 0.0, .27176, .27062, .27072, .27099},
{0.0, 0.0, 0.0, .27392, .27155, .27146, .27161},
{0.0, 0.0, 0.0, 0.0, .27326, .27261, .27251},
{0.0, 0.0, 0.0, 0.0, .2755, .27448, .27414}
}
};

/* ------------------------------------------------------ */
for (lower_row = 0; lower_row < 11; lower_row++)
```

THIS
PAGE
IS
MISSING
IN
ORIGINAL
DOCUMENT

```
    /* Calcutate the corrections for inlet, exhaust, and humidity.  */

     inlet_sfc_fac = 0.001125 * INLET_LOSS + 1;
     inlet_t8_fac = 0.001875 * INLET_LOSS + 1;
     inlet_w8_fac = -0.001375 * INLET_LOSS + 1;
     inlet_p8_fac = 0.0 * INLET_LOSS + 1;
     inlet_cp8_fac = 0.00040 * INLET_LOSS + 1;
    exhaust_sfc_fac = 0.001295 * EXHAUST_LOSS + 1;
    exhaust_t8_fac = 0.00098 * EXHAUST_LOSS + 1;
    exhaust_w8_fac = 0.0003636 * EXHAUST_LOSS + 1;
     exhaust_p8_fac = 0.00245 * EXHAUST_LOSS + 1;
     exhaust_cp8_fac = 0.0020 * EXHAUST_LOSS + 1;
     humidity_sfc_fac = 0.0000387 * HUMIDITY + 1;
     humidity_t8_fac = -0.0000057 * HUMIDITY + 1;
     humidity_w8_fac = -0.00005 * HUMIDITY + 1;
     humidity_p8_fac = 0.0 * HUMIDITY + 1;
     humidity_cp8_fac = 0.000125 * HUMIDITY + 1;

     sfc_factor = inlet_sfc_fac * exhaust_sfc_fac *  humidity_sfc_fac;
     t8_factor = inlet_t8_fac * exhaust_t8_fac  *  humidity_t8_fac;
    w8_factor = inlet_w8_fac * exhaust_w8_fac *  humidity_w8_fac;
    p8_factor = inlet_p8_fac * exhaust_p8_fac *  humidity_p8_fac;
   cp8_factor = inlet_cp8_fac * exhaust_cp8_fac * humidity_cp8_fac;
     /* ------------------------------------------------------------------------------ */

    return(parameter_engine [0] * sfc_factor);

}


/* ------------------------------------------------------------ */
/* ------------------------------------------------------------ */


double pc42_sfc_map(engine_bhp,engine_rpm,number_cylinders)

double engine_bhp,
engine_rpm,
number_cylinders;

{
/* This function provides the PC 4.2 engine parameter of: */
/*   SFC = specific fuel consumption in Lb/Hp-Hr.  */

/* It performs linear interpolation on a given engine BHP and RPM */
/* using the tabular engine performance map for ISO conditions. */
/* The SFC performance maps are based on a per cylinder basis.  */
/* The total engine_bhp = cylinder_bhp * number_cylinders.  */


unsigned int lower_row, /*bottom bhp bound for interpolation*/
higher_row,          /*upper bhp bound for interpolation*/
left_column,         /*left rpm bound for interpolation*/
right_column;         /*right rpm bound for interpoaltion*/
```

```
float cylinder_bhp,
  fuel_rack_limit_rpm,
bhp_ratio,          /*bhp ratio for linear interpolation*/
rpm_ratio,          /*rpm ratio for linear interpolation*/
param_lower_rpm,    /*engine parameter at desired bhp and*/
                        /*lower rpm interpolation bound*/
param_upper_rpm;    /*engine parameter at desired bhp and*/
                        /*upper rpm interpolation bound*/



/*declare the variable that will output the engine operating*/
/*parameter of SFC for the given engine BHP and RPM.*/


float parameter_engine;

  /* ------------------------------------------------------------ */
  /* Declare and initialize the arrays */

/* declare and initialize the array that defines the BHP */
/* values for the tabular rows */

float bhp [12] = {53.0, 100.0, 250.0, 400.0, 567.0,
        600.0, 800.0, 1000.0, 1200.0,
        1400.0, 1466.0, 1629.0};

/* declare and initialize the array that defines the rpm */
/* values for the tabular columns */

float rpm [12] = {125.0, 150.0, 175.0, 200.0, 225.0,
        250.0, 275.0, 300.0, 325.0, 350.0, 375.0, 400.0};

/* declare and initialize the array that maps the */
/* engine SFC parameters. */

float engine_map [12][12] =
  {

    {.348, .349, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0},
    {.344, .345, .345, .346, .347, .347, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0},
    {0.0, .331, .333, .335, .336, .338, .339, .341, .343, .345, 0.0, 0.0},
    {0.0, 0.0, 0.0, .323, .325, .327, .329, .331, .333, .334, .335, .339},
    {0.0, 0.0, 0.0, 0.0, 0.0, .316, .317, .318, .320, .323, .326, .331},
    {0.0, 0.0, 0.0, 0.0, 0.0, .314, .315, .316, .317, .321, .324, .327},
    {0.0, 0.0, 0.0, 0.0, 0.0, 0.0, .308, .307, .307, .308, .309, .313},
    {0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, .304, .303, .303, .304, .305} ,
    {0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, .303, .298, .298, .301},
    {0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, .304, .300, .300},
    {0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, .304, .303, .301},
    {0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, .306, .304},
```

```
};

/* ---------------------------------------------------- */

cylinder_bhp = engine_bhp/number_cylinders;

/* Ensure that the BHP and RPM are within the PC4.2 fuel */
/* rack limitations. */

if(cylinder_bhp >= 567.0)
  {
  fuel_rack_limit_rpm = 0.1105 * cylinder_bhp + 204.33;
  if(engine_rpm < fuel_rack_limit_rpm)
    {
    printf("This engine BHP of %7.1f and RPM of %5.0f\n", engine_bhp, engine_rpm);
    printf("are outside the engine fuel rack limitations.\n");
    printf("Re-examine the BHP and RPM operating regime.\n\n");
    return(-1.0);
    }
  }

/* ---------------------------------------------------- */
for (lower_row = 0; lower_row < 11; lower_row++)
  {
  higher_row = lower_row + 1;

  if (bhp [lower_row] <= cylinder_bhp
                          && cylinder_bhp < bhp [higher_row])
    {
    bhp_ratio = (cylinder_bhp - bhp [lower_row])/
                              (bhp [higher_row] - bhp [lower_row]);
    break;
    }
  }
/* ---------------------------------------------------- */
for (left_column = 0; left_column < 12; left_column++)
  {
  right_column = left_column + 1;

  if (rpm [left_column] <= engine_rpm
                    && engine_rpm < rpm [right_column])
    {
    rpm_ratio = (engine_rpm - rpm [left_column])/
                          (rpm [right_column] - rpm [left_column]);
    break;
    }
  }
/* ---------------------------------------------------- */

if (engine_map [higher_row][left_column] == 0
      || engine_map [lower_row][right_column] == 0)
  {
  printf("This BHP of %7.1f and RPM of %5.0f\n", engine_bhp, engine_rpm);
```

```
    printf("are outside the ISO conditions engine map interpolation zone.\n");
    printf("Re-examine  the PC4.2 BHP and RPM operating regime.\n\n");
    return(-1.0);
    }

  /* ------------------------------------------------------------ */
    param_lower_rpm = bhp_ratio*
                            (engine_map [higher_row][left_column]-
                             engine_map [lower_row][left_column])+
                             engine_map [lower_row][left_column];

     param_upper_rpm = bhp_ratio*
                            (engine_map [higher_row][right_column]-
                             engine_map [lower_row][right_column])+
                             engine_map [lower_row][right_column];

                            parameter_engine = rpm_ratio*
                            (param_upper_rpm - param_lower_rpm) + param_lower_rpm;



     /* ------------------------------------------------------------ */

     return(parameter_engine);
}



 * ------------------------------------------------------------ */
/* ------------------------------------------------------------ */

double pc26_sfc_map(engine_bhp,engine_rpm,number_cylinders)

double engine_bhp,
engine_rpm,
number_cylinders;

{
/* This program provides the PC 2.6V engine parameter of: */
/*   SFC = specific fuel consumption in Lb/Hp-Hr.  */

/* It performs linear interpolation on a given engine BHP and RPM */
/* using the tabular engine performance map for ISO conditions. */
/* The SFC performance maps are based on a per cylinder basis.  */
/* The total engine_bhp = cylinder_bhp * NUMBER_CYLINDERS.  */


  unsigned int lower_row, /*bottom bhp bound for interpolation*/
  higher_row,          /*upper bhp bound for interpolation*/
  left_column,         /*left rpm bound for interpolation*/
  right_column;         /*right rpm bound for interpoaltion*/


  float cylinder_bhp,
```

```
  bhp_ratio,        /*bhp ratio for linear interpolation*/
  rpm_ratio,        /*rpm ratio for linear interpolation*/
  param_lower_rpm,  /*engine parameter at desired bhp and*/
                      /*lower rpm interpolation bound*/
  param_upper_rpm;  /*engine parameter at desired bhp and*/
                      /*upper rpm interpolation bound*/


/*declare the variable that will output the engine operating*/
/*parameter of SFC for the given engine BHP and RPM.*/

  float parameter_engine;

  /* ------------------------------------------------------------ */
  /* Declare and initialize the arrays */

  /* declare and initialize the array that defines the BHP */
  /* values for the tabular rows */

  float bhp [8] = {40.0, 100.0, 200.0, 300.0, 400.0,
                   500.0, 600.0, 737.01};

  /* declare and initialize the array that defines the rpm */
  /* values for the tabular columns */

  float rpm [14] = {200.0, 225.0, 250.0, 275.0, 300.0, 325.0,
              350.0, 375.0, 400.0, 425.0, 450.0, 475.0, 500.0, 520.01};

  /* declare and initialize the array that maps the */
  /* engine SFC parameters. */

  float engine_map [8][14] =
        {
        {.375, .375, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
        0.0, 0.0},
        {.349, .353, .355, .355, .356, .358, .359, 0.0, 0.0, 0.0, 0.0, 0.0,
        0.0, 0.0},
        {0.0, 0.0, 0.0, .324, .325, .326, .326, .328, .328, .329, .330, .331,
        .333, .336},
        {0.0, 0.0, 0.0, 0.0, 0.0, .321, .320, .319, .319, .319, .32, .32,
        .321, .321},
        {0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, .316, .314, .312, .312, .312,
        .312, .314},
        {0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, .311, .310, .310,
        .310, .311},
        {0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, .311, .310,
        .310, .310},
        {0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, .312,
        .312, .312},
  };
```

```
/* ------------------------------------------------------ */

cylinder_bhp = engine_bhp/number_cylinders;

/* ------------------------------------------------------ */
for (lower_row = 0; lower_row < 11; lower_row++)
  {
  higher_row = lower_row + 1;

  if (bhp [lower_row] <= cylinder_bhp
                                  && cylinder_bhp < bhp [higher_row])
    {
    bhp_ratio = (cylinder_bhp - bhp [lower_row])/
                                          (bhp [higher_row] - bhp [lower_row]);
    break;
    }
  }
/* ------------------------------------------------------ */
for (left_column = 0; left_column < 12; left_column++)
  {
  right_column = left_column + 1;

  if (rpm [left_column] <= engine_rpm
                                  && engine_rpm < rpm [right_column])
    {
    rpm_ratio = (engine_rpm - rpm [left_column])/
                                  (rpm [right_column] - rpm [left_column]);
    break;
    }
  }
/* ------------------------------------------------------ */

 if (engine_map [higher_row][left_column] == 0
                || engine_map [lower_row][right_column] == 0)
  {
  printf("This BHP of %7.1f and RPM of %5.0f\n", engine_bhp, engine_rpm);
  printf("are outside the ISO conditions engine map interpolation zone.\n");
  printf("Re-examine the PC2.6 BHP and RPM operating regime.\n\n");
  return(-1.0);
  }

/* ------------------------------------------------------ */

  {
   param_lower_rpm = bhp_ratio*
                (engine_map [higher_row][left_column]-
                 engine_map [lower_row][left_column])+
                 engine_map [lower_row][left_column];

    param_upper_rpm = bhp_ratio*
                         (engine_map [higher_row][right_column]-
                          engine_map [lower_row][right_column])+
                          engine_map [lower_row][right_column];
```

```
        parameter_engine = rpm_ratio*
                           (param_upper_rpm - param_lower_rpm) + param_lower_rpm;
    }

    /* ------------------------------------------------------------ */

    return(parameter_engine);

}
```

Appendix F

Sample of Output

The following pages provide samples of output from the engine's program. The program was executed with three variations: cruise with no PDSS, cruise with only one on-line PDSS, and cruise with more than one on-line PDSS. Each of these variations has a slightly different output that shows the power split between engines.

Select one of the four propulsion options.
1 Mechanical Drive.
2 Electric Drive.
3 Mechanical with Electric Hybrid drive.
4 Multiple Dispersed Waterjets.
1
Select the number of propulsors used for cruise.

1 One propulsor.
2 Two propulsors.
3 Three propulsors.

2
Select the number of propulsors used for boost.

2 Two propulsors.
3 Three propulsors.

2
Select one of the propulsor options.

1 FPP
2 CRP
3 Contra-rotating
4 Preswirl Stator
5 Ducted FPP
6 Ducted CRP
7 Ducted Contra-rotating
8 Ducted Preswirl Stator
9 Waterjet

2
Select the desired mechanical transmission options.

1 Epicyclic on each shaft.
2 LTDR on each shaft.
3 LTDR with reversing mechanism on each shaft.
4 LTDR on each shaft with cross-connect.
2
The following questions are used to determine the engine type[s] and alignments to operate the propulsion system.
Note: if you choose the LM-2500 for boost, you will also be given the opportunity to select an alternative engine type for cruise.
If you choose one of the other boost engine types, it will be assumed that your selected boost engine type will be used for both cruise and boost operation.

Select one of the boost engine types.
1 LM-2500 Gas Turbine
2 ICR Gas Turbine
3 PC 4.2 Diesel,16 cyl rated at 26060 HP.

```
1
Select one of the cruise engine types.
0  NONE
1  ICR Gas Turbine
2  PC 4.2 Diesel
3  PC 2.6 Diesel

0
Select the total number of LM-2500 engines
used during maximum boost operation.
2  Two
3  Three
4  Four
4
Select the total number of LM-2500 engines used
during cruise operation.
1  One
2  Two
3  Three
4  Four

2
Will PDSS be driven off any of the gas turbine
output shafts.

0  NO
1  Yes
0
-------------------------------------------


Your selections are summarized as follows:

# Cruise Propulsors= 2          # Boost Propulsors = 2
Propulsor Type     = CRP        Hybrid Trans Type  = None
Transmission Type  = Mech       XMission per shaft = LTDR
Cruise Eng Type    = none       Boost Eng Type     = LM-2500
Total # boost eng               Total # boost eng
   used for cruise = 2             used for boost  = 4
Total # cruise eng              Total # cruise eng
   used for cruise = 0             used for boost  = 0
PDSS flag setting  = NO         Number of PDSS     = 0
-------------------------------------------

 Select an option:
1  Run the program with your selected propulsion system.
2  Reselect the propulsion system.
3  Quit the program.
```

```
1
Enter the following in decimal format.
  Enter the cruise EHP:9360.0
Enter the cruise QPC:.718
Enter the cruise transmission efficiency:.955
Enter the cruise propulsor RPM:88.9
Enter the gear ratio:
22.39


Number of engines used for cruise = 2.0
avg_endur_bhp_per_eng =  7507.77 BHP
uncorrect_sfc_per_eng = 0.61055 LB/HP/HR
propulsion f_factor = 1.0400
Tail Pipe Allowance =  1.020
pct of endurance time w/bleed =  0.500
avg_endur_fuel_rate = 0.66672 LB/HP/HR
K34_BHP_per_eng =  1693.04
The elec avg_sfc = 0.76595 LB/HP/HR.

The electric fuel wt =   282.8 LTONS.
The propulsion fuel wt =  1009.5 LTONS.
The Total Fuel Weight = 1292.4 LTONS.

The cruise type engine Specs:

Max Engine Power      = 26250.00.
Max Engine RPM        =  3600.00.
Min Engine RPM        =  1200.00.
Number of Cylinders   =     0.00.
Engine Type Code      =     1.00.
Engine Weight         = 59000.00 lb.
Linear Weight Intake  =   493.20 lb.
Linear Weight Uptake  =   799.00 lb.
Engine Length, ft     =    26.50.
Engine Width, ft      =     8.70.
Engine Height, ft     =    10.40.
Cross Section Intake  =   119.70 ft^2.
Cross Section Uptake  =   162.50 ft^2.
Acquisition Cost      =     4.50 $mil,1991.
```

The boost type engine Specs:

Max Engine Power	=	26250.00 BHP.
Max Engine RPM	=	3600.00.
Min Engine RPM	=	1200.00.
Number of Cylinders	=	0.00.
Engine Type Code	=	1.00.
Engine Weight	=	59000.00 lb.
Linear Weight Intake	=	493.20 lb.
Linear Weight Uptake	=	799.00 lb.
Engine Length, ft	=	26.50.
Engine Width, ft	=	8.70.
Engine Height, ft	=	10.40.
Cross Section Intake	=	119.70 ft^2.
Cross Section Uptake	=	162.50 ft^2.
Acquisition Cost	=	4.50 $mil,1991.

exit
Select one of the four propulsion options.
1 Mechanical Drive.
2 Electric Drive.
3 Mechanical with Electric Hybrid drive.
4 Multiple Dispersed Waterjets.
1
Select the number of propulsors used for cruise.

1 One propulsor.
2 Two propulsors.
3 Three propulsors.

1
Select the number of propulsors used for boost.

2 Two propulsors.
3 Three propulsors.

3
Select one of the propulsor options.

1 FPP
2 CRP
3 Contra-rotating
4 Preswirl Stator
5 Ducted FPP
6 Ducted CRP
7 Ducted Contra-rotating
8 Ducted Preswirl Stator
9 Waterjet

2
Select the desired mechanical transmission options.

1 Epicyclic on each shaft.
2 LTDR on each shaft.
3 LTDR with reversing mechanism on each shaft.
1
The following questions are used to determine the engine type[s] and alignments to operate the propulsion system.
Note: if you choose the LM-2500 for boost, you will also be given the opportunity to select an alternative engine type for cruise.
If you choose one of the other boost engine types, it will be assumed that your selected boost engine type will be used for both cruise and boost operation.

Select one of the boost engine types.
1 LM-2500 Gas Turbine
2 ICR Gas Turbine
3 PC 4.2 Diesel,16 cyl rated at 26060 HP.

```
1
Select one of the cruise engine types.
0  NONE
1  ICR Gas Turbine
2  PC 4.2 Diesel
3  PC 2.6 Diesel

0
Select the total number of LM-2500 engines
used during maximum boost operation.
2  Two
3  Three
4  Four
3
Select the total number of LM-2500 engines used
during cruise operation.
1  One
2  Two
3  Three
4  Four

1
Will PDSS be driven off any of the gas turbine
output shafts.

0  NO
1  Yes
1
Select the total number of pdss units desired.
1  One pdss unit.
2  Two pdss units.
3  Three pdss units.
4  Four pdss units.
3
-------------------------------------------



Your selections are summarized as follows:

# Cruise Propulsors= 1          # Boost Propulsors = 3
Propulsor Type     = CRP        Hybrid Trans Type  = None
Transmission Type  = Mech       XMission per shaft = Epicyclic
Cruise Eng Type    = none       Boost Eng Type     = LM-2500
Total # boost eng               Total # boost eng
   used for cruise = 1             used for boost  = 3
Total # cruise eng              Total # cruise eng
   used for cruise = 0             used for boost  = 0
PDSS flag setting  = YES                Number of PDSS     = 3
-------------------------------------------

 Select an option:
1  Run the program with your selected propulsion system.
2  Reselect the propulsion system.
3  Quit the program.
```

```
1
Enter the following in decimal format.
  Enter the cruise EHP:9360
Enter the cruise QPC:.718
Enter the cruise transmission efficiency:.96
Enter the cruise propulsor RPM:89.0
Enter the gear ratio:
22.4


Number of engines used for cruise = 1.0
HP per PDSS =  1693.04 BHP
K34_BHP_per_eng =  1693.04 BHP
avg_endur_bhp_per_eng = 16630.37 BHP
uncorrect_sfc_per_eng = 0.49099 LB/HP/HR
propulsion f_factor = 1.0400
Tail Pipe Allowance =  1.020
pct of endurance time w/bleed =  0.500
avg_endur_fuel_rate = 0.53616 LB/HP/HR
The elec avg_sfc = 0.76595 LB/HP/HR.

The electric fuel wt =   282.8 LTONS.
The propulsion fuel wt =   899.1 LTONS.

The Total Fuel Weight = 1182.0 LTONS.

The cruise type engine Specs:

Max Engine Power      = 26250.00.
Max Engine RPM        =  3600.00.
Min Engine RPM        =  1200.00.
Number of Cylinders   =     0.00.
Engine Type Code      =     1.00.
Engine Weight         = 59000.00 lb.
Linear Weight Intake  =   493.20 lb.
Linear Weight Uptake  =   799.00 lb.
Engine Length, ft     =    26.50.
Engine Width, ft      =     8.70.
Engine Height, ft     =    10.40.
Cross Section Intake  =   119.70 ft^2.
Cross Section Uptake  =   162.50 ft^2.
Acquisition Cost      =     4.50 $mil,1991.
```

The boost type engine Specs:

Max Engine Power	=	**26250.00 BHP.**
Max Engine RPM	=	**3600.00.**
Min Engine RPM	=	**1200.00.**
Number of Cylinders	=	0.00.
Engine Type Code	=	1.00.
Engine Weight	=	59000.00 lb.
Linear Weight Intake	=	493.20 lb.
Linear Weight Uptake	=	799.00 lb.
Engine Length, ft	=	26.50.
Engine Width, ft	=	8.70.
Engine Height, ft	=	10.40.
Cross Section Intake	=	119.70 ft^2.
Cross Section Uptake	=	162.50 ft^2.
Acquisition Cost	=	4.50 $mil,1991.

```
C:\TC>tc
Select one of the four propulsion options.
1  Mechanical Drive.
2  Electric Drive.
3  Mechanical with Electric Hybrid drive.
4  Multiple Dispersed Waterjets.
1
Select the number of propulsors used for cruise.

1  One propulsor.
2  Two propulsors.
3  Three propulsors.

2
Select the number of propulsors used for boost.

2  Two propulsors.
3  Three propulsors.

2
Select one of the propulsor options.

1  FPP
2  CRP
3  Contra-rotating
4  Preswirl Stator
5  Ducted FPP
6  Ducted CRP
7  Ducted Contra-rotating
8  Ducted Preswirl Stator
9  Waterjet

2
Select the desired mechanical transmission options.

1  Epicyclic on each shaft.
2  LTDR on each shaft.
3  LTDR with reversing mechanism on each shaft.
4  LTDR on each shaft with cross-connect.
2
The following questions are used to determine the engine
type(s) and alignments to operate the propulsion system.
Note:  if you choose the LM-2500 for boost, you will also be
given the opportunity to select an alternative
engine type for cruise.
If you choose one of the other boost engine types,
it will be assumed that your selected boost engine
type  will be used for both cruise and boost operation.

Select one of the boost engine types.
1  LM-2500 Gas Turbine
2  ICR Gas Turbine
3  PC 4.2 Diesel,16 cyl rated at 26060 HP.
```

```
1
Select one of the cruise engine types.
0  NONE
1  ICR Gas Turbine
2  PC 4.2 Diesel
3  PC 2.6 Diesel

0
Select the total number of LM-2500 engines
used during maximum boost operation.
2  Two
3  Three
4  Four
4
Select the total number of LM-2500 engines used
during cruise operation.
1  One
2  Two
3  Three
4  Four

2
Will PDSS be driven off any of the gas turbine
output shafts.

0  NO
1  Yes
1
Select the total number of pdss units desired.
2  Two pdss units.
3  Three pdss units.
4  Four pdss units.
4
-------------------------------------------


Your selections are summarized as follows:

# Cruise Propulsors= 2          # Boost Propulsors = 2
Propulsor Type     = CRP        Hybrid Trans Type  = None
Transmission Type  = Mech       XMission per shaft = LTDR
Cruise Eng Type    = none       Boost Eng Type     = LM-2500
Total # boost eng               Total # boost eng
   used for cruise = 2             used for boost  = 4
Total # cruise eng              Total # cruise eng
   used for cruise = 0             used for boost  = 0
PDSS flag setting  = YES                Number of PDSS     = 4
-------------------------------------------

 Select an option:
1  Run the program with your selected propulsion system.
2  Reselect the propulsion system.
3  Quit the program.
```

```
1
Enter the following in decimal format.
  Enter the cruise EHP:9360.0
Enter the cruise QPC:.718
Enter the cruise transmission efficiency:.955
Enter the cruise propulsor RPM:89.0
Enter the gear ratio:
22.4


Number of engines used for cruise = 2.0
HP per PDSS w/no-bleed =  1693.04 BHP
HP per PDSS w/bleed =  1128.69 BHP
K34_BHP_per_eng w/bleed =  1128.69 BHP
avg_endur_bhp_per_eng_no_bld =  9200.81 BHP
avg_endur_bhp_per_eng_bld =  8636.46 BHP
uncorrect_sfc_per_eng_no_bld = 0.56387 LB/HP/HR
uncorrect_sfc_per_eng_bld = 0.57940 LB/HP/HR
propulsion f_factor = 1.0400
Tail Pipe Allowance =  1.020
pct of endurance time w/bleed =  0.500
avg_endur_fuel_rate_no_bld = 0.61575 LB/HP/HR
avg_endur_fuel_rate_bld = 0.63270 LB/HP/HR
K34 w/bleed SFC = 1.01307 LB/HP/HR.

The electric fuel wt =    62.4 LTONS.
The propulsion fuel wt =  1122.3 LTONS.

The Total Fuel Weight = 1184.7 LTONS.

The cruise type engine Specs:

Max Engine Power      = 26250.00.
Max Engine RPM        =  3600.00.
Min Engine RPM        =  1200.00.
Number of Cylinders   =     0.00.
Engine Type Code      =     1.00.
Engine Weight         = 59000.00 lb.
Linear Weight Intake  =   493.20 lb.
Linear Weight Uptake  =   799.00 lb.
Engine Length, ft     =    26.50.
Engine Width, ft      =     8.70.
Engine Height, ft     =    10.40.
Cross Section Intake  =   119.70 ft^2.
Cross Section Uptake  =   162.50 ft^2.
Acquisition Cost      =     4.50 $mil,1991.
```

The boost type engine Specs:

Max Engine Power	= 26250.00	BHP.
Max Engine RPM	= 3600.00.	
Min Engine RPM	= 1200.00.	
Number of Cylinders	= 0.00.	
Engine Type Code	= 1.00.	
Engine Weight	= 59000.00	lb.
Linear Weight Intake	= 493.20	lb.
Linear Weight Uptake	= 799.00	lb.
Engine Length, ft	= 26.50.	
Engine Width, ft	= 8.70.	
Engine Height, ft	= 10.40.	
Cross Section Intake	= 119.70	ft^2.
Cross Section Uptake	= 162.50	ft^2.
Acquisition Cost	= 4.50	$mil,1991.