Naval Ship Propulsion and Electric Power Systems Selection for Optimal Fuel Consumption

by

Emmanouil Sarris

Bachelor of Science in Marine Engineering
Hellenic Naval Academy, 2003

Submitted to the Department of Mechanical Engineering and the System Design and Management Program in Partial Fulfillment of the Requirements for the Degrees of

Naval Engineer and
Master of Science in Engineering and Management

at the
Massachusetts Institute of Technology
June 2011

Signature of Author.....
Department of Mechanical Engineering
System Design and Management Program
May 6,2011

Certified by.....
Mark S. Welsh
Professor of the Practice of Naval Construction and Engineering
Thesis Supervisor

Accepted by.....
Pat Hale
Director, Systems Design and Management Fellows Program
Engineering Systems Division

Accepted by.....
David E. Hardt
Chairman, Department Committee on Graduate Studies
Department of Mechanical Engineering

Page Intentionally Left Blank

Naval Ship Propulsion and Electric Power Systems Selection for Optimal Fuel Consumption

by

Emmanouil Sarris

Submitted to the Department of Mechanical Engineering and the System Design and Management Program on May 6, 2011 in Partial Fulfillment of the Requirements for the Degrees of

Naval Engineer
and
Master of Science in Engineering and Management

ABSTRACT

Although propulsion and electric power systems selection is an important part of naval ship design, respective decisions often have to be made without detailed ship knowledge (resistance, propulsors, etc.). Propulsion and electric power systems have always had to satisfy speed and ship-service power requirements. Nowadays, increasing fuel costs are moving such decisions towards more fuel-efficient solutions. Unlike commercial ships, naval ships operate in a variety of speeds and electric loads, making fuel consumption optimization challenging.

This thesis develops a flexible decision support tool in Matlab® environment, which identifies the propulsion and ship-service power generation systems configuration that minimizes fuel consumption for any ship based on its operating profile. Mechanical-driven propulsion systems with or without propulsion derived ship-service power generation, separate ship-service systems and integrated power systems are analyzed. Modeling includes hull resistance using the Holtrop-Mennen method requiring only basic hull geometry information, propeller efficiencies using the Wageningen B series and transmission and prime movers fuel efficiencies. Propulsion and ship-service power generation systems configuration is optimized using the genetic algorithm.

US Navy's Advanced Surface Ship Evaluation Tool (ASSET) model for the DDG-51 Flight I destroyer was used for modeling validation. Optimal fuel consumption results are compared against the existing configuration for the DDG-51 Flight I destroyer using a representative operating profile.

Thesis Supervisor: Mark S. Welsh
Title: Professor of the Practice of Naval Construction and Engineering

ACKNOWLEDGEMENTS

I would like to thank the Hellenic Navy for giving me the great opportunity to attend MIT and for financially supporting my studies.

I would like to express my sincere appreciation to the director of the Naval Construction and Engineering program, Captain Mark Welsh, USN and the director of the Systems Design and Management program, Pat Hale for their uninterrupted support throughout my graduate studies.

On a personal level, I would like to express my deepest appreciation to my significant other Demetra Paparounas for her inspiration, support and concern.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

NOMENCLATURE

$1+k$	Combined form factor
$1+k_1$	Hull form factor
$1+k_2$	Appendage resistance factor
A_{BT}	Transverse sectional area of the bulb at the position where still water intersects the stem
A_T	Immersed transom area
B	Beam
BHP	Brake Power
C_B	Block coefficient
C_F	Frictional resistance coefficient
C_M	Midship section coefficient
C_P	Prismatic coefficient
C_R	Residuary resistance coefficient
C_T	Total resistance coefficient
C_V	Viscous resistance coefficient
C_{WP}	Waterplane coefficient
D	Propeller diameter
EAR	Expanded area ratio
EHP	Effective power
fc	Fuel consumption
Fn	Froude number

g	Gravity acceleration (9.81m/sec^2)
h_B	Distance of the center of transverse area A_{BT} from the keel line
i_E	Half angle of entrance in degrees
J	Advance ratio
K_Q	Propeller torque coefficient
K_T	Propeller thrust coefficient
L	Ship's length between perpendiculars
lcb	Longitudinal position of the center of buoyancy forward of midships
LF	Engine load fraction
n	Propeller rotational speed
n_S	Number of propulsion shafts
P	Propeller pitch
Q	Propeller torque
R_A	Model-ship correlation resistance
R_{APP}	Appendage resistance
R_B	Pressure resistance of bulbous bow near the water surface
Re	Reynolds number
R_{TOTAL}	Total resistance
R_{TR}	Pressure resistance due to transom immersion
R_W	Wave resistance
S	Hull wetted surface area
S_{APP}	Total wetted appendage surface area
sfc	Specific fuel consumption
SHP	Shaft power

T	Propeller thrust
t	Thrust deduction factor
T_F	Forward draft
T_M	Mean draft
V	Speed
V_A	Speed of advance
w	Taylor wake fraction
Z	Propeller number of blades
η_b	Line shaft bearing efficiency
η_D	Total propulsive efficiency
η_H	Hull efficiency
η_O	Propeller open water efficiency
η_R	Relative rotative efficiency
η_s	Stern tube bearing and seal efficiency
η_{TR}	Power transmission efficiency
ν	Salt water kinematic viscosity ($1.1883*10^{-6}$ m^2/sec)
ρ	Sea water density (1025.26 kg/m^3 at 15^0C)
∇	Immersed volume

CHAPTER 1

INTRODUCTION

1.1 Motivation

Naval ships life cycle cost is of major consideration when designing a new vessel. Office of Naval Research (ONR) studies have shown that fuel and lubricants cost represents the 8.4% of the ship's life cycle cost (Surko and Osborne, 2005), making fuel efficiency a significant factor in the design process. Moreover, high volatility of oil prices indicates that fuel consumption cost may be even more important in the future.

Apart from fuel efficiency, the design team has to satisfy maximum speed and endurance range customer requirements, which are directly related with the mission profile of the naval ship. This poses a major design challenge and raises the question if ship fuel consumption can be optimized while meeting these requirements.

Fuel on board a ship is mainly consumed by propulsion and ship-service power generation systems. Due to their volume, weight and support requirements but also due to the intake and exhaust duct requirements, heat emissions management and shaft alleys arrangement restrictions, propulsion and ship-service power generation systems tend to have a major impact on the ship's general arrangements. This reveals, first, that the propulsion and ship-service power generation systems decisions have to be made relatively early in the design process and second, that these early decisions may be difficult, costly or even impossible to change during the later stages of the design. It is therefore crucial these decisions to be best informed from the beginning.

Design tools currently used at this stage at the Naval Construction and Engineering program at MIT fail to address this problem. The user has to manually select propulsion and ship-service power generation systems based on limited information on maximum engine power and full-load specific fuel consumption. The result is that although selected systems meet endurance and maximum speed and ship-service power requirements they scarcely represent fuel-efficient choices. Any kind of manual computation is challenging since unlike commercial vessels, naval ships operate in a wide range of speeds and ship-service loads. Moreover, even though naval ships are built to be able to sail at a maximum sustained speed and withstand all mission systems operating at the same time, this generally only represents a very small portion of the ship's operating profile.

It becomes obvious that there are potential significant gains in fuel costs by optimizing the selection of the propulsion and ship-service power generation systems for the expected ship operating profile. Furthermore, it will be beneficial if this information can become available during the early stages of the ship design.

1.2 Organization of the Thesis

This thesis develops a program in Matlab® environment, which can be used as decision support tool when selecting propulsion and ship-service power generation systems. This program, in paricular, is able to identify the most fuel-efficient propulsion and ship-service power generation systems for a given ship based on its operating profile and provide helpful fuel efficiency and engine usage related plots.

For this purpose certain parts of the ship design process were modeled. Propulsion related fuel consumption is driven by the need to propel the ship through the water at a desired speed. Chapter 2 discusses ship resistance estimation modeling, based on the Holtrop and Mennen statistical method (Holtrop and Mennen, 1982; Holtrop, 1984). Chapter 3 analyzes propulsor modeling. Representative twin screw propellers efficiencies are estimated by modeling and optimizing the Wageningen B propeller

series. Chapter 4 discusses the propulsion and electric power generation systems and their components. Emphasis is given so that fuel consumption related parameters are adequately estimated for partial-load operation. Chapter 5 describes the systems optimization set-up and DDG-51 Flight I destroyer optimal fuel consumption systems. Chapter 6 discusses conclusions and recommendations for future work. The three appendices include program library engine data, program user guide and Matlab® scripts respectively.

Chapters 2 to 4 are structured so that a brief description of the associated theory is given before modeling is discussed. Model validation is done using the US Navy's Advanced Surface Ship Evaluation Tool (ASSET) and the DDG-51 Flight I destroyer. The destroyer's optimal fuel consumption propulsion and ship-service power generation systems are presented at the end of Chapter 5.

1.3 Program Architecture

Naval ship design is a systems engineering process where decisions about propulsion and ship-service power generations systems are required to be taken during various stages of the design. Namely, this can be as early that the ship's hull may not be finalized or as late that a ship is considering a propulsion or ship-service power generation modification. In that respect ship propulsion and ship-service power generations systems selection resembles the agile development life cycle approach (INCOSE, 2010).

In order to reflect this need for adaptability the program was developed to be flexible enough to accommodate various stages of the design process. Namely, the program can estimate ship resistance using minimal hull information but can move directly to the propulsor modeling if dependable ship resistance information is available. Likewise, it can bypass propulsor modeling if the user opts to use specific shaft power information. This flexibility allows the program to be used during various stages of the naval ship design and with different levels of data fidelity. Program architecture along with associated Matlab® functions are presented schematically in Figure 1.

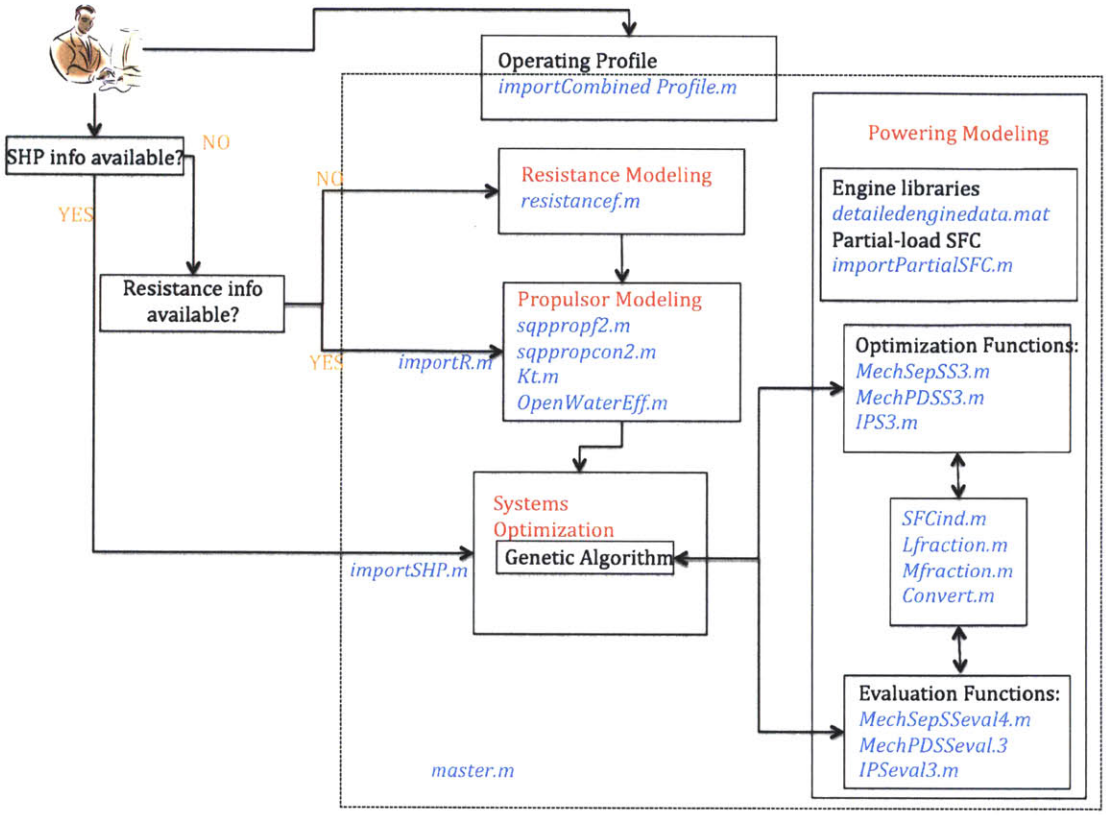


Figure 1 - Program Architecture

CHAPTER 2

RESISTANCE MODELING

2.1 Resistance Theory

The resistance of a ship is the force required to tow the ship at a given speed in smooth water assuming no interference from the towing ship (van Mannen and van Oossanen, 1988). The most important components of resistance can be identified to be:

1. Frictional resistance
2. Form resistance
3. Wave resistance

2.1.1 Frictional Resistance

It is the component of resistance caused by the motion of the hull through a viscous fluid. It has been proven to be a function of Reynolds number

$$\mathrm{Re} = \frac{VL}{\nu} \tag{2.1}$$

where V is the ship's speed, L is the ship's length and ν is the kinematic viscosity of the fluid.

2.1.2 Form Resistance

In a viscous fluid a body experiences energy losses due to flow separation before the rear stagnation point. This causes a turbulent wake (eddies), which adds an additional drag component. This resistance component is often called eddy resistance (van Mannen

and van Oossanen, 1988) or pressure drag due to the pressure drop at the back. The calculation of form drag is an extremely difficult process. Usually, its dependence on Re is neglected and is combined with wave resistance in what is called "residual resistance" (Gillmer and Johnson, 1982). The calculation of both wave and form resistance is achieved through model testing.

2.1.3 Wave Resistance

It is the component of resistance caused by the flow disturbance generated by a ship due to the existence of the free surface, which generates waves. As discussed previously, this part of the resistance is indistinguishable from the form resistance in model testing. The residual resistance coefficient C_R is assumed to depend only on the Froude number

$$Fn = \frac{V}{\sqrt{gL}} \tag{2.2}$$

where g is the acceleration of gravity.

As a result the total resistance coefficient can be approximated as:

$$C_T(\mathrm{Re},Fn) \approx C_F(\mathrm{Re}) + C_R(Fn) \tag{2.3}$$

where C_F is the frictional resistance coefficient and C_R is the residuary resistance coefficient.

This is the famous Froude hypothesis, which is widely for resistance estimation through tow tank experiments.

This part of resistance only represents the energy lost into wave generation from the ship. It has nothing to do with any additional resistance effect resulting from ambient waves.

2.1.4 Other Components of Resistance

In addition to those described above, there are other, usually smaller, components of total ship resistance. Some of them are:

1. Air resistance caused by the above-water part of the ship due to its motion through the air
2. Appendage resistance caused by the presence of appendages such as rudders, shafts and shaft struts. It comprises both frictional and form components.
3. Added resistance from present ambient waves due to diffraction effects.
4. Added resistance due to hull fouling. Increased surface roughness causes faster transition from laminar to turbulent flow, which increases frictional resistance.

2.2 Resistance Estimation Modeling using the Holtrop-Mennen Method

Literature research examined several available resistance prediction methods before deciding which is the most suitable to implement in the Matlab® environment. These methods were:

1. The Taylor Gertler Standard Series (Gertler, 1954)
2. DTMB Series 64 (Yeh, 1965)
3. NPL High Speed Round Bilge Series (Bailey, 1976)
4. VVT Mathematical Model (Lahtiharju, 1991)
5. Fung's Mathematical Model (Fung, 1991)
6. Holtrop and Mennen method (MARIN Mathematical Model) (Holtrop and Mennen, 1978, 1982; Holtrop, 1984)

The Holtrop and Mennen method represents a statistical analysis of the results of resistance and propulsion tests of 334 models and was preferred by virtue of range of application (Table 1), suitability for programming and minimal user input. Moody

(1996) highlighted the flexibility and relative accuracy of the Holtrop-Mennen method relative to the others.

	Lower Limit	Upper Limit
Cp	0.55	0.85
L/B	3.90	14.9
B/T	2.10	4.00
Fn	0.00	1.00

Table 1 - Range of application for Holtrop-Mennen method

Resistance estimation modeling based on Holtrop-Mennen is described in the subsequent paragraphs.

2.2.1 Total Resistance

The total resistance is subdivided into components as follows:

$$R_{TOTAL} = (1 + M_D)(R_F(1 + k_1) + R_{APP} + R_W + R_B + R_{TR} + R_A) \tag{2.4}$$

where:

R_F is the frictional resistance

$(1 + k_1)$ is the form factor of the hull

R_{APP} is the appendage resistance

R_W is the wave-making resistance

R_B is the additional pressure resistance of bulbous bow near the water surface

R_{TR} is the additional pressure resistance due to transom immersion

R_A is the model-ship correlation resistance

M_D is the power design margin

2.2.2 Frictional Resistance

The frictional resistance for each speed is calculated from:

$$R_F = 0.5\rho S V^2 C_F \tag{2.5}$$

where ρ is the sea water density and S is the hull wetted surface area.

The frictional resistance coefficient is calculated according to the ITTC-1957 formula:

$$C_F = \frac{0.075}{(\log \mathrm{Re} - 2)^2} \tag{2.6}$$

The hull wetted surface area can be approximated:

$$S = L(2T_M + B)\sqrt{C_M}\left(0.453 + 0.4425C_B - 0.2862C_M - 0.0003467\frac{B}{T_M} + 0.3696C_{WP}\right) + 2.38\frac{A_{BT}}{C_B} \tag{2.7}$$

where L is the length between perpendiculars, T_M is the mean draft, B is the beam, C_M is the midship section coefficient, C_B is the block coefficient and C_{WP} is the waterplane coefficient for the ship and A_{BT} is the transverse sectional area of the bulb at the position where still water intersects the stem.

2.2.3 Hull Form Resistance Factor

The hull form resistance factor is:

$$1 + k_1 = 0.93 + 0.487118c_{14}\left(\frac{B}{L}\right)^{1.06806}\left(\frac{T_M}{L}\right)^{0.46106}\left(\frac{L}{L_R}\right)^{0.121563}\left(\frac{L^3}{\nabla}\right)^{0.36486}(1 - C_P)^{-0.604247} \tag{2.8}$$

where ∇ is the ship's immersed volume and C_P is the ship's prismatic coefficient. c_{14} is assumed to be 1 for normal stern shape ships. L_R is defined as:

$$L_R = L\left(1 - C_P + \frac{0.06C_P lcb}{4C_P - 1}\right) \tag{2.9}$$

lcb is the longitudinal position of the center of buoyancy forward of midships.

2.2.4 Wave Resistance

Wave resistance for each speed is calculated by different formulas depending on the Froude number (Fn)

2.2.4.1 Fn<0.4

$$R_{W1} = c_1 c_2 c_5 \nabla \rho g e^{\left[m_1 Fn^{-0.9} + m_4 \cos(\lambda Fn^{-2}) \right]} \tag{2.10}$$

where:

$$c_1 = 2223105 c_7^{3.78613} \left(\frac{T_M}{B} \right)^{1.07961} (90 - i_E)^{-1.37165} \tag{2.11}$$

$$c_2 = e^{-1.89\sqrt{c_3}} \tag{2.12}$$

$$c_5 = 1 - \frac{0.8 A_T}{B T_M C_M} \tag{2.13}$$

c_2 accounts for the reduction of wave resistance due to the presence of a bulbous bow.

c_5 expresses transom stern influence on wave resistance.

$$m_1 = 0.0140407 \frac{L}{T_M} - 1.75254 \frac{\nabla^3}{L} - 4.79323 \frac{B}{L} - c_{16} \tag{2.14}$$

$$m_4 = 0.4 c_{15} e^{\left(-0.034 Fn^{-3.29}\right)} \tag{2.15}$$

$$\lambda = \begin{cases} 1.446 C_P - 0.03 \dfrac{L}{B} & \dfrac{L}{B} < 12 \\ 1.446 C_P - 0.36 & \dfrac{L}{B} > 12 \end{cases} \tag{2.16}$$

Other parameters can be determined from:

$$c_7 = \begin{cases} 0.229577\left(\frac{B}{L}\right)^{0.33333} & \frac{B}{L} < 0.11 \\ \frac{B}{L} & 0.11 < \frac{B}{L} < 0.25 \\ 0.5 - 0.0625\frac{L}{B} & \frac{B}{L} > 0.25 \end{cases} \tag{2.17}$$

$$i_E = 1 + 89e^{\left[-\left(\frac{L}{B}\right)^{0.80856}(1-C_{WP})^{0.30484}(1-C_P-0.0225lcb)^{0.6367}\left(\frac{L_R}{B}\right)^{0.34574}\left(\frac{100\nabla}{L^3}\right)^{0.16302}\right]} \tag{2.18}$$

i_E is the half angle of entrance in degrees . It represents the angle of the waterline at the bow. It can be either user input or calculated by the above formula:

$$c_3 = \frac{0.56A_{BT}^{1.5}}{BT\left(0.31\sqrt{A_{BT}} + T_F - h_B\right)} \tag{2.19}$$

h_B is the distance of the center of transverse area A_{BT} from the keel line. T_F is the forward draft of the ship. Model accepts values if

$$h_B < 0.6T_F \tag{2.20}$$

$$c_{16} = \begin{cases} 8.07981C_P - 13.8673C_P^{\,2} + 6.984388C_P^{\,3} & C_P < 0.8 \\ 1.73014 - 0.7067 & C_P > 0.8 \end{cases} \tag{2.21}$$

$$c_{15} = \begin{cases} -1.69385 & \frac{L^3}{\nabla} < 512 \\ -1.69385 + \frac{\left(\frac{L}{\nabla^{\frac{1}{3}}} - 8\right)}{2.36} & 512 < \frac{L^3}{\nabla} < 1727 \\ 0 & \frac{L^3}{\nabla} > 1727 \end{cases} \tag{2.22}$$

2.2.4.2 *Fn*>0.55

$$R_{W2} = c_{17}c_2c_5\nabla\rho g e^{\left[m_3 Fn^{-0.9} + m_4\cos(\lambda Fn^{-2})\right]} \tag{2.23}$$

where:

$$c_{17} = 6919.3C_M{}^{-1.3346}\left(\frac{\nabla}{L^3}\right)^{2.00977}\left(\frac{L}{B}-2\right)^{1.40692} \tag{2.24}$$

$$m_3 = -7.2035\left(\frac{B}{L}\right)^{0.326869}\left(\frac{T_M}{B}\right)^{0.605375} \tag{2.25}$$

2.2.4.3 $0.4 \le Fn \le 0.55$

Wave resistance is calculated using the following interpolation formula:

$$R_{W3} = R_{W1} + \frac{(10Fn-4)(R_{W2}-R_{W1})}{1.5} \tag{2.26}$$

2.2.5 Appendage Resistance

Model will attempt to calculate the appendage resistance for each speed using the Holtrop-Mennen formula:

$$R_{APP} = 0.5\rho S_{APP} V^2 (1+k_2) C_F \tag{2.27}$$

where S_{APP} is the total wetted appendage surface area and $(1+k_2)$ is the appendage resistance factor. Both quantities have to be calculated and input to the program by the user. This can be done using Table 2 if specific appendage information is available.

Appendage type	$(1+k_2)$ value
Rudder behind skeg	1.5-2.0
Rudder behind stern	1.3-1.5
Twin-screw balance rudders	2.8
Shaft brackets	3.0
Skeg	1.5-2.0
Strut bossings	3.0
Hull bossings	2.0
Shafts	2.0-4.0
Stabilizer fins	2.8
Dome	2.7
Bilge keels	1.4

Table 2 - Approximate $(1+k_2)$ values for different appendages (Holtrop and Mennen,1982)

The $(1+k_2)$ value is determined from:

$$1+k_2 = \frac{\sum_i (1+k_2)_i S_i}{S_{APP}} \tag{2.28}$$

$$S_{APP} = \sum_i S_i \tag{2.29}$$

where S_i and $(1+k_2)_i$ correspond for each individual appendage.

During early stages of the design specific appendage information is unlikely to be available. In this case, the calculated appendage resistance will be 0 and the program will prompt the user to enter the appendage resistance as a percentage of bare hull resistance.

PNA Vol. II (van Mannen and van Oossanen, 1988) discusses appendage resistance approximations (as a percentage of bare hull resistance) for different ship types.

2.2.6 Transom Stern Resistance

The additional pressure resistance due to immersed transom is calculated for each speed by the formula:

$$R_{TR} = 0.5\rho A_T V^2 c_6 \tag{2.30}$$

where A_T is the immersed transom area and

$$c_6 = \begin{cases} 0.2(1-0.2F_{nt}) & F_{nt} < 5 \\ 0 & F_{nt} > 5 \end{cases} \tag{2.31}$$

$$F_{nt} = \frac{V}{\sqrt{\frac{2gA_T}{B + BC_{WP}}}} \tag{2.32}$$

2.2.7 Bulbous Bow Resistance

Additional resistance due to the presence of a bulbous bow near the surface is calculated for each speed by the formula:

$$R_B = \frac{0.11e^{-3P_B^{-2}} F_{ni}^{\ 3} A_{BT}^{\ 1.5} \rho g}{1 + F_{ni}^{\ 2}} \tag{2.33}$$

P_B is a measure of bow emergence calculated by the formula:

$$P_B = \frac{0.56\sqrt{A_{BT}}}{T_F - 1.5h_B} \tag{2.34}$$

F_{ni} resembles the Froude number based on immersion:

$$F_{ni} = \frac{V}{\sqrt{g\left(T_F - h_B - 0.25\sqrt{A_{BT}}\right) + 0.15V^2}} \tag{2.35}$$

2.2.8 Correlation Resistance

The model-ship correlation allowance coefficient accounts for various differences between resistance measured by full-scale trials and resistance estimated by model testing (Gillmer and Johnson, 1982). The program gives the user the option to choose between Holtrop-Mennen's formula (2.36) and one from ASSET (2.38).

$$C_A = 0.006(L+100)^{-0.16} - 0.00205 + 0.003\sqrt{\frac{L}{7.5}}C_B{}^4 c_2(0.04 - c_4) \qquad (2.36)$$

where:

$$c_4 = \begin{cases} \frac{T_F}{L} & \frac{T_F}{L} < 0.04 \\ 0.04 & \frac{T_F}{L} \geq 0.04 \end{cases} \qquad (2.37)$$

$$C_A = \begin{cases} 0.0008 & L < 190 \\ 0.008289L^{-0.333} - 0.00064 & 190 < L < 960 \\ 0.0002 & L > 960 \end{cases} \qquad L \text{ in [ft]} \qquad (2.38)$$

The correlation resistance is given from:

$$R_A = 0.5\rho S V^2 C_A \qquad (2.39)$$

2.2.9 Power Design Margin

A power design margin is commonly used when approximate prediction methods are used for early or incomplete definitions of the ship design (Parsons, 2004). For the model the US Navy 10% power design margin for early parametric prediction before the plan and appendage configuration (DDS 051-1, 1984) was adopted.

2.2.10 Effective Power Estimation

Effective power (EHP) for each speed is calculated from:

$$EHP = R_{TOTAL} V \qquad (2.40)$$

2.3 Model Validation

Program was validated for correct method implementation using the numerical examples from the respective papers (Holtrop and Mennen, 1982; Holtrop, 1984).

Subsequently, the model had to be validated against other methods for predicting ship resistance. For this purpose the benchmark was DDG-51 Flight I destroyer resistance as this is calculated by ASSET using the Taylor series using the available ASSET's DDG-51 Flight I model file. The hull geometry parameters were input in the program for evaluation were:

Parameter	Symbol	Unit	Value
Length between perpendiculars	L	m	142.04
Beam	B	m	18.15
Draft fwd	T_F	m	6.67
Draft aft	T_A	m	6.67
Immersed Volume	∇	m^3	9019.4
Midship section coefficient	C_M		0.830
Waterplane coefficient	C_{WP}		0.788
Bulb transverse sectional area	A_{BT}	m^2	0
Center of bulb distance from keel line	h_B	m	0
Longitudinal center of buoyancy	lcb	% fwd MS	-0.964
Total appendage area [m^2]	S_{APP}	m^2	0
k+2 appendage factor	k+2		2.5
Half angle of entrance	i_E	deg	11.19
Immersed transom area	A_T	m^2	0

Table 3 - DDG-51 Flight I input for resistance estimation model evaluation

These hull geometry parameters were taken or calculated from the ASSET DDG-51 Flight I model file. Appendage area, transom area and bulbous bow- related parameters were set to 0 due to absence of specific information. $(1+k_2)$ factor was assumed to 2.5.

Resistance results for the speed range 1-30knots are depicted in Figure 2.

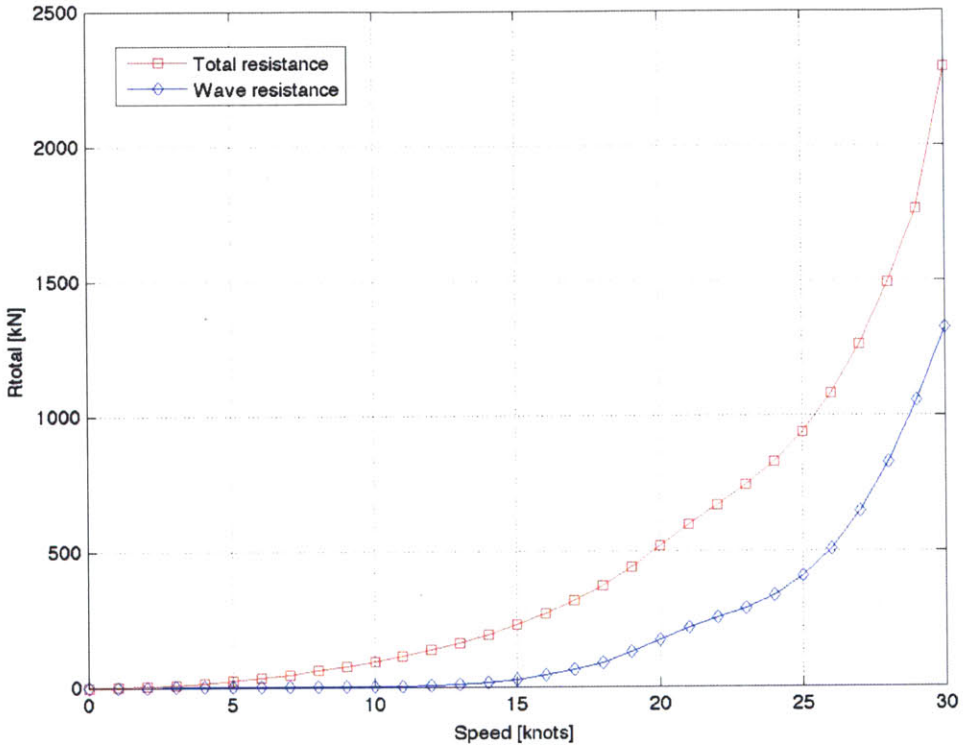


Figure 2 - DDG-51 Flight I Resistance Estimation

As expected no appendage, bulbous bow and transom elements of the resistance are present.

Total resistance was plotted against ASSET results for comparison in Figure 3.

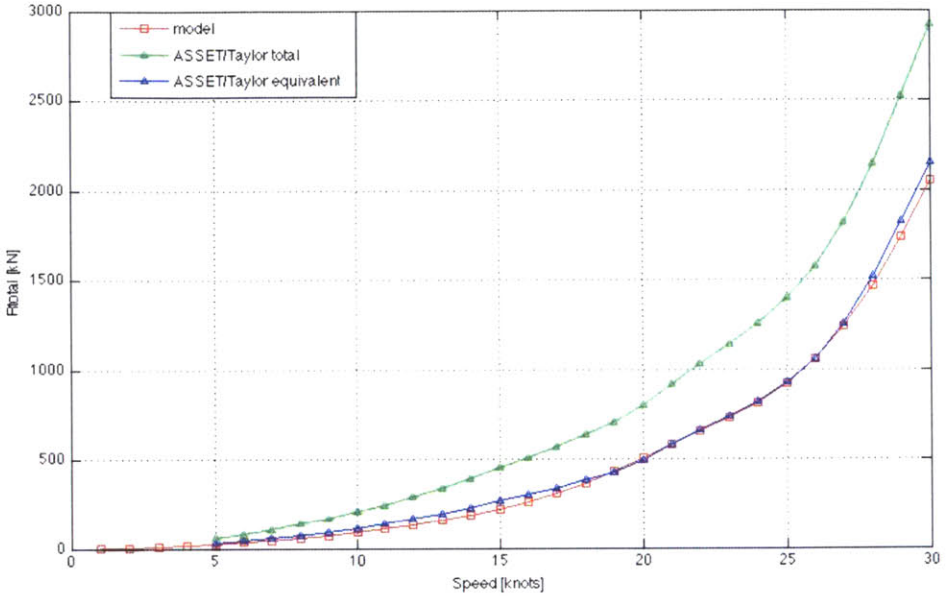


Figure 3 - Resistance estimation comparison between model and ASSET for DDG-51 Flight I

ASSET's total resistance includes appendage and air resistance as well as a resistance margin. To enable a straightforward comparison the equivalent resistance (frictional, wave and form) was plotted against the model's prediction without the power design margin. Results were within reasonable accordance to what ASSET provided.

CHAPTER 3

PROPULSOR MODELING

The screw propeller is the most common propulsor for marine vessels. Since it is also the most widely used propulsor for naval ships, it was straightforward to be chosen for propulsor modeling for the program. The purpose of the program was not to precisely define the geometry of the propeller but to estimate the associated propeller efficiencies so that resistance information can be adequately translated to shaft horsepower (SHP) requirements.

3.1 Propeller Theory

3.1.1 Introduction

A propeller generates thrust from lift on the blades that rotate at an angle of attack relative to a flow. Results of model test for propellers are available through methodical series charts. Basic propeller geometry is depicted in Figure 4.

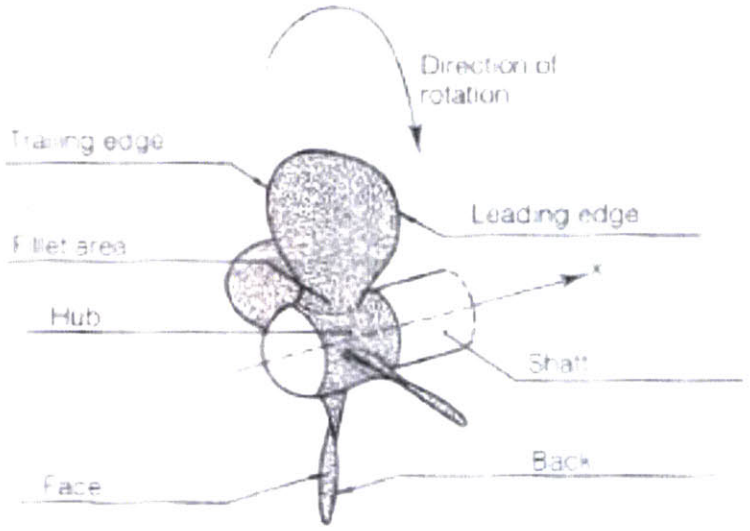


Figure 4 - Propeller Geometry (Woud and Stapersma, 2002)

Two types of propellers are of interest for this work:

- The fixed-pitch propeller. It is the least complicated propeller but requires shaft-reversing capability, which makes it suitable for electrical-driven propulsion systems.
- The controllable-pitch propeller. It achieves ship reversing while maintaining the same shaft rotation and better engine load management by varying the blades' angle of attack to the flow. This makes it ideal for mechanical-driven propulsion systems.

Despite the increased complexity of the controllable-pitch propeller, it can be generally modeled as fixed-pitch with an additional efficiency loss due to larger diameter hub, which is required to accommodate the controllable-pitch mechanism.

3.1.2 Propeller Parameters

(a) Pitch ratio $\frac{P}{D}$. Pitch is the distance the propeller advances during one revolution and D is the propeller diameter.

(b) Thrust coefficient:

$$K_T = \frac{T}{\rho n^2 D^4} \tag{3.1}$$

where T is the propeller thrust and n is the rotational speed.

(c) Torque coefficient:

$$K_Q = \frac{Q}{\rho n^2 D^5} \tag{3.2}$$

where Q is the propeller torque.

(d) Advance ratio:

$$J = \frac{V_A}{nD} \tag{3.3}$$

where V_A is the speed of advance. The propeller is located inside the ship's wake where water has a forward speed, tending to follow the ship's motion. This causes the propeller to advance at a lower speed relative to the water than the ship itself. The ship's wake is generally highly turbulent causing differences in flow velocity from point to point. Nevertheless, an average measure of the wake effect is done by the *Taylor wake fraction,* which is defined as:

$$w = \frac{V - V_A}{V} \tag{3.4}$$

(e) Open-water efficiency:

$$\eta_o = \frac{TV_A}{2\pi n Q_o} \tag{3.5}$$

where Q_o is the propeller open-water propeller torque (the torque in absence of ship's hull).

The action of the propeller behind the ship causes a pressure drop, which increases resistance. This can be quantified using the *thrust deduction factor*:

$$t = \frac{n_S T - R_{TOTAL}}{n_S T} \tag{3.6}$$

where n_S is the number of propulsion shafts.

Taylor wake fraction and thrust deduction factor define the hull efficiency since all differences between a towed and propelled hull are contained within those factors (Woud and Stapersma, 2002)

$$\eta_H = \frac{1-t}{1-w} \tag{3.7}$$

(f) Expanded area ratio:

$$EAR = \frac{A_E}{A_O} \tag{3.8}$$

where A_E is the area of the blades outside the hub expanded in a plane, and A_O is the area of the circle defined by the blades' tips circumference.

(g) Relative rotative efficiency (η_R) is defined as the ratio of the open-water power and the actual delivered power and accounts for the non-uniform velocity field in front of the propeller (Woud and Stapersma, 2002).

(h) Total propulsive efficiency is the product of open-water, relative rotative and hull efficiencies:

$$\eta_D = \eta_O \eta_R \eta_H \tag{3.9}$$

3.1.3 Propeller Cavitation

Cavitation is a phenomenon usually met in highly loaded propellers. It occurs when the absolute pressure on the aft side of the blade is reduced below the vapor pressure of

the water, generating vapor pockets. These pockets disrupt the flow reducing efficiency and if they collapse close to the blade surface, significant erosion may occur. Avoidance of cavitation is an important requirement in every propeller design.

3.2 Propeller Modeling

3.2.1 Hull Efficiency Parameters (*w, t*)

If hull resistance information is not available the program will calculate Taylor wake fraction and thrust deduction factor using the following Holtrop-Mennen formulas:

$$w = 0.3095C_B + 10C_V C_B - \frac{0.23D}{\sqrt{BT_M}} \tag{3.10}$$

$$t = 0.325C_B - \frac{0.1885D}{\sqrt{BT_M}} \tag{3.11}$$

where D is the propeller diameter and C_V is the viscous resistance coefficient given from:

$$C_V = (1+k)C_F + C_A \tag{3.12}$$

$1+k$ is a form factor combining the hull form and appendage form factors:

$$1+k = 1+k_1 + \left[(1+k_2)-(1+k_1)\right]\frac{S_{APP}}{S+S_{APP}} \tag{3.13}$$

3.2.2 Expanded Area Ratio

The Keller formula is used to determine the minimum expanded area ratio allowed to prevent cavitation (van Mannen and van Oossanen, 1988):

$$EAR_{\min} = \frac{(1.3+0.3Z)T}{(P_0 + \rho gh - P_V)D^2} \tag{3.14}$$

where Z is the number of blades for each propeller, $P_0 + \rho gh$ is the static pressure at the shaft centerline and P_V is the vapor pressure. This formula represents transom-stern naval vessels and $P_0 - P_V = 99047 \frac{N}{m^2}$ for salt water at 15^0C.

3.2.3 Thrust and Torque Coefficients and Open-Water Efficiency at Design Speed

Calculating the propeller's open-water efficiency for a particular operating speed requires knowledge of its geometry and operating characteristics. When carrying preliminary design studies, though, this information is not necessarily available. For this purpose, the thrust and torque coefficients polynomials for the Wageningen B series propellers, as developed by Oosterveld and van Oossanen (1973), were used. These polynomials express thrust and torque coefficients as a function of pitch over diameter ratio, expanded area ratio, advance coefficient and blade number (Z) for $\mathrm{Re} = 2 \cdot 10^6$. Although, Oosterveld and van Oossanen (1975) provide corrections for Reynolds higher numbers, these were not included in the program considering advice from Loukakis and Gelegenis (1988) that the polynomials at $\mathrm{Re} = 2 \cdot 10^6$ take into account approximately the full size propeller roughness effect. The Wageningen B series represents a fairly extensively used propeller series for preliminary ship design studies.

The open-water efficiency can be calculated by the formula:

$$\eta_O = \frac{JK_T}{2\pi K_Q} \tag{3.15}$$

3.2.5 Open-Water Efficiency at Other Speeds

Having determined the propeller properties it is of great importance to estimate the open-water efficiency for the whole speed range. Solving for the thrust equation (3.6) becomes:

$$T = \frac{R_{TOTAL}}{n_S(1-t)} \tag{3.16}$$

Solving for speed of advance equation (3.4) becomes:

$$V_A = V(1-w) \tag{3.17}$$

Substituting equations (3.3), (3.16) and (3.17) into (3.1) we obtain:

$$K_T = \frac{R_{TOTAL}}{\rho n_S(1-t)(1-w)^2 D^2 V^2} J^2 \tag{3.18}$$

which represents the load curve for the ship. Thrust coefficient is also determined from the Wageningen B series polynomials as a function of the advance coefficient (J):

$$K_T = f_1(J) \tag{3.19}$$

The system of non-linear equations (3.17) and (3.18) is then numerically solved for each ship's speed. Finally, torque coefficient is calculated from Wageningen B series polynomial and open-water efficiency is estimated from (3.14).

3.2.6 Relative Rotative Efficiency

The relative rotative efficiency is calculated after the propeller selection and is given by the Holtrop-Mennen formula:

$$\eta_R = 0.9737 + 0.111(C_P - 0.0225lcb) - 0.06325\frac{P}{D} \tag{3.20}$$

3.3 Propeller Selection

A propeller is selected to optimize open-water efficiency at the design condition, typically the maximum ship's speed. This enables achieving the maximum speed requirement with the least installed power (BHP). The program assumes that the design

speed for propeller selection is the maximum speed that appears in the ship's speed profile.

3.3.1 Problem Formulation

The design variables are:

- Pitch over diameter ratio ($\frac{P}{D}$)
- Rotative speed (n)
- Diameter (D)
- Expanded area ratio (EAR)

The design parameters are:

- Number of propeller shafts (n_S)
- Number of blades for each propeller (Z)
- Taylor wake fraction (w)
- Thrust deduction factor (t)
- Maximum speed ($V_{\max}$)
- Total resistance at maximum speed (R_{TOTAL})

Table 4 depicts design variables and parameters bounds and values.

		Value/Initial Value	Lower Bound	Upper Bound	Set by
Design Variables	$\frac{P}{D}$	1.0	0.5	1.4	model
	n	2.0rps	1.0rps	4.5rps	model
	D	$\frac{D_{\min} + D_{\max}}{2}$	$D_{\min}$	$D_{\max}$	user
	EAR	0.7	0.3	1.05	model
Design Parameters	n_S	2			model
	Z	5			model
	w	ship specific			user/model
	t	ship specific			user/model
	$V_{\max}$	ship specific			user
	R_{TOTAL}	ship specific			user/model

Table 4 - Propeller optimization design variables and parameters

Model is set-up to work with two propulsion shafts and 5 blade propellers since the vast majority of naval ships share this configuration. Nonetheless, it can be very easily adapted to allow different propeller parameter choices. Design variable bounds are defined by the applicability of the Wageningen B series polynomials except from the propeller diameter, which is user input.

The design objective is to maximize the propeller open-water efficiency (η_o) such that:

- The propeller provides the required thrust (T) at the design speed (equality constraint)
- The propeller expanded area ratio exceeds the minimum allowed for the Keller cavitation formula ($EAR \geq EAR_{\min}$).

3.3.2 Optimization Algorithm

Selecting an appropriate algorithm is a challenging problem in optimization. This single-objective problem involves continuous bounded variables and has to satisfy non-linear constraints. Therefore, sequential quadratic programming (SQP) numerical optimization was the preferred choice. SQP was implemented by employing Matlab®'s *fmincon* build-in function and using the 'active-set' medium-scale algorithm.

Figure 5 shows an example of propeller open-water efficiency optimization for the DDG-51 Flight I destroyer converging quickly after 14 iterations.

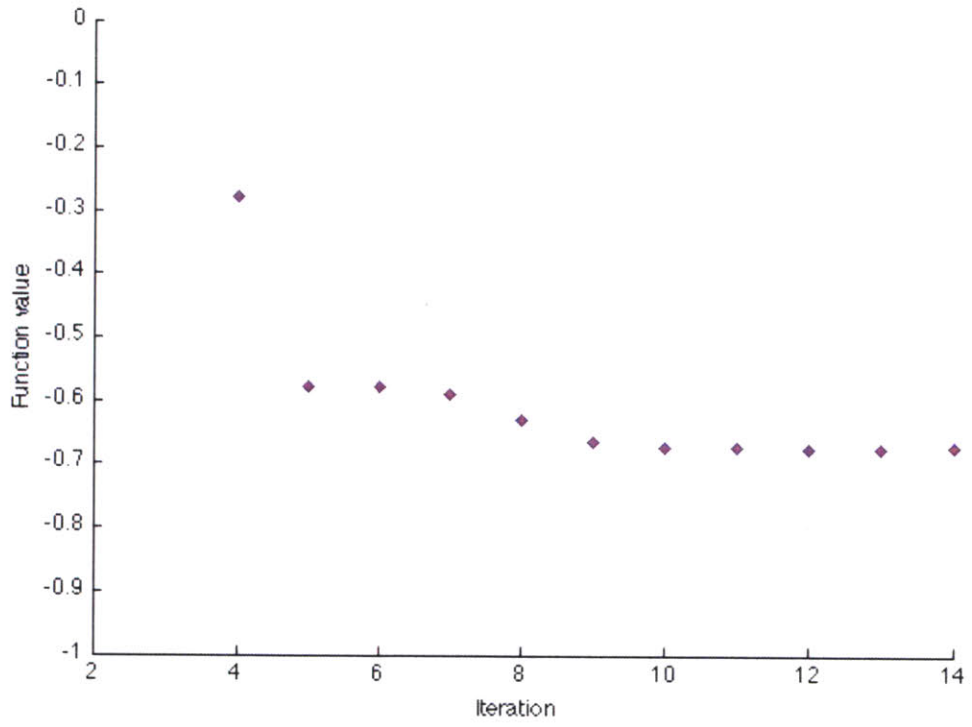


Figure 5 - Convergence for DDG-51 Flight I propeller open-water efficiency optimization

3.3.3 Global optimum

Finding the global optimum with confidence is often very difficult in engineering problems. Optimality conditions that are at the same time necessary and sufficient are

complicated and can seldom be verified in practice (Gill et al., 1986). Moreover, gradient-based techniques are sensitive to the search starting point, which can cause the algorithm to terminate at a local optimum.

In order to increase the likelihood that the program will be able to converge to the optimal propeller solution the Matlab®'s build-in *GlobalSearch* algorithm was used. This algorithm evaluates *fmincon* multiple times using various randomly scattered starting points (Ugray et al., 2007).

3.4 Model Validation

Propulsion factors estimated from Holtrop-Mennen formulas (w, t, η_R) were validated for correct implementation using the numerical examples from the respective paper (Holtrop, 1984). DDG-51 Flight I destroyer ASSET data was used to validate the method itself. Hull efficiency parameters (w, t) were calculated using equations (3.10) and (3.11) and compared against ASSET-derived values.

Parameter	Symbol	ASSET	Model
Taylor wake fraction	w	0.020	0.065
Thrust deduction factor	t	0.055	0.082
Hull efficiency	η_H	0.964	0.982

Table 5 – Hull efficiency parameters estimation comparison between model and ASSET for DDG-51 Flight I

Although there is an obvious deviation in the hull efficiency parameters prediction, hull efficiency estimation is within 2% from ASSET's results, which is satisfactory taking into account the applicability range of the Holtrop-Mennen equations. Relative rotative efficiency (η_R) requires pitch over diameter ratio information and thus was evaluated after the propeller selection.

Propeller modeling was also validated against the ASSET's propeller actual data. In order to be able to isolate and validate the propeller selection model only, ASSET's hull efficiency, resistance and propeller hub position data was used as input.

Parameter	Symbol	Value
Taylor wake fraction	w	0.020
Thrust deduction factor	t	0.055
Hub depth	h	5.68m
Minimum propeller diameter	$D_{\min}$	4.50m
Maximum propeller diameter	$D_{\max}$	5.18m
Total resistance at 30knots	R_{TOTAL}	2918.7kN

Table 6 - DDG-51 Flight I input for propeller selection model evaluation

Propeller diameter limits were chosen so that the upper limit matched the actual propeller diameter. Without imposing any torque constraints, the optimal open-water efficiency was naturally expected to be at the maximum possible diameter.

Model selected propeller compared to ASSET propeller as follows:

Parameter	Symbol	ASSET	Model
Propeller diameter	D	5.18m	5.18m
Pitch over diameter ratio	$\frac{P}{D}$	1.72	1.27
Rotative speed	n	159.3rpm	189.3rpm
Expanded area ratio	EAR	0.784	1.032
Open-water efficiency	η_O	0.692	0.674

Table 7 - Propeller parameters comparison between model and ASSET for DDG-51 Flight I at design speed

Although ASSET's design speed, determined by the installed power plant, is 30.55knots and model's is 30knots, which causes a noticeable difference in propeller thrust, the major differences in the table above can be attributed to ASSET's selected propeller not satisfying any cavitation criteria. Namely, using the Burrill's 10% cavitation criterion, the minimum EAR required is calculated by ASSET to be 1.249. It is possible that cavitation criteria are not taken into consideration since ASSET's model serves as an original DDG-51 Flight I representation and not a ship design study and

imposing early stage design cavitation limits is irrelevant. Nonetheless, preliminary ship design practice suggests that a cavitation criterion limiting expanded area ratio should be used. Under this constraint, the ASSET's propeller is expected to have lower open-water efficiency.

Open-water efficiency comparison for the whole speed range is depicted in Figure 6. Although this represents different propellers, model depicts increased open-water efficiency at the mid-speed range, as does ASSET.

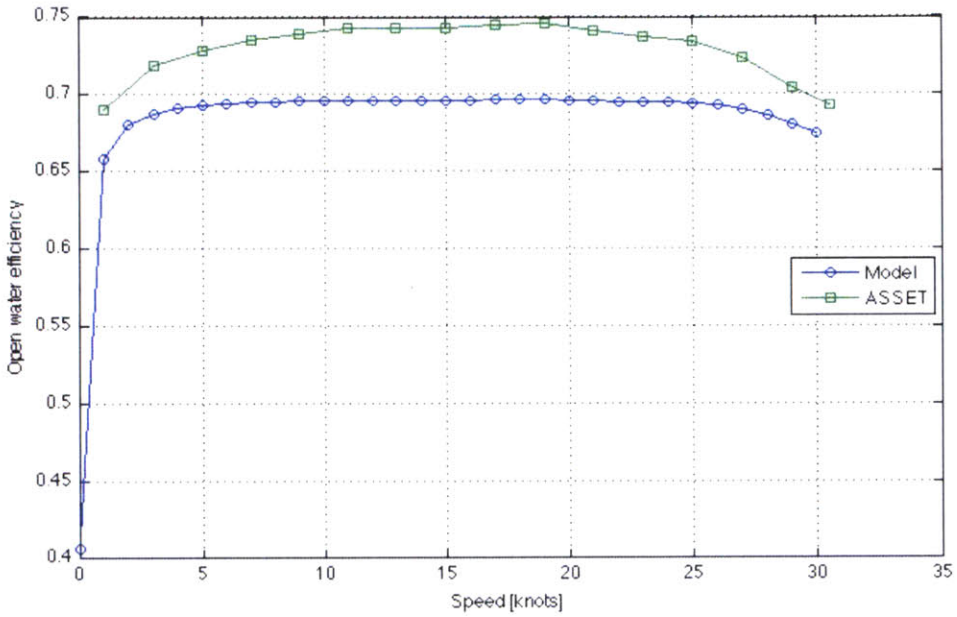


Figure 6 - Open-water efficiency comparison between model and ASSET for DDG-51 Flight I

Table 8 summarizes all efficiencies calculated using the propulsion model and ASSET for DDG-51 Flight I.

Parameter	Symbol	ASSET	Model	% Model departure
Open-water efficiency	η_O	0.692	0.674	-2.60
Relative rotative efficiency	η_R	0.985	0.961	-2.44
Hull efficiency	η_H	0.964	0.982	1.87
Total propulsive efficiency	η_D	0.657	0.636	-3.20

Table 8 – Propulsion efficiencies comparison between model and ASSET for DDG-51 Flight I at design speed

The purpose of this modeling process was not to design the exact propeller geometry that best suits a ship but to be able to estimate the propulsion efficiency with reasonable accuracy. DDG-51 Flight I evaluation showed that efficiency results are in general accordance to what ASSET provided.

CHAPTER 4

POWERING MODELING

4.1 Powering Theory

The propulsion system is primarily responsible for propelling the ship at the required speed. The electric power system is primarily responsible for supplying electric power for ship-service systems such as navigation, communication and air-conditioning. These two systems can be interrelated with the propulsion system contributing to electrical power generation and electrical power system providing propulsion through electric motors.

4.1.1 Resistance and Propulsion

Effective horsepower can be calculated using equation (2.40). The necessary shaft power to achieve the desired speed is related to the effective power with the formula:

$$SHP = \frac{EHP}{\eta_D \eta_s \eta_b} \tag{4.1}$$

where η_D is the total propulsive efficiency defined in chapter 3, η_s is the stern tube bearing and seal efficiency and η_b is the line shaft bearing efficiency (Parsons, 2004).

This can be translated to total engine brake power using the formula:

$$BHP = \frac{SHP}{\eta_{TR}} \tag{4.2}$$

where η_{TR} is the power transmission efficiency. This represents the efficiency of the reduction gear for mechanical-driven systems or the efficiency of the electrical power transmission for electrical-driven systems.

4.1.2 Main Components of the Propulsion System

The propulsion system generates the necessary thrust to enable the ship move through water at the desired speed. The main components of the propulsion system are:

- Prime mover. It converts the chemical energy contained in the fuel into mechanical energy. Typical prime movers are diesel engines, gas turbines and steam turbines.

- Transmission. It transfers the mechanical energy generated by the prime mover to the propulsor and transfers back the thrust generated by the propulsor to the ship's hull. Transmission type can be mechanical or electrical. Mechanical transmissions can be direct-drive or geared-drive when a gearbox is used to match the desired rotational speed of the propulsor. Electrical transmissions use the prime mover to power a generator and the electrical energy is transported to an electric motor, which powers the propulsor.

- Propulsor. It applies the transmitted mechanical energy to the sea and generates thrust. The most common type of propulsor is the propeller (fixed or controllable pitch). Waterjet is another type of propulsor where water is accelerated by a pump to generate thrust. Its use is limited in high-speed applications.

4.1.3 Main Components of the Electrical Power System

The electric power system generates, converts and distributes the necessary electric power for the ship-service systems. The main components of an electrical power system are:

- Prime mover. It is the same as for a propulsion system with the only difference that the mechanical energy is used to drive the generator.

- Generator. It converts mechanical energy into electric energy. Prime movers and generators are often combined into generator sets as manufacturers try to achieve best operating characteristics. A main propulsion shaft can also be used to drive a generator (shaft generator).

- Distribution and conversion. Switchboards receive, control and distribute electric energy from generators to loads and secondary electric energy supplies (Woud and Stapersma, 2002). Secondary electric energy is converted by transformers, rectifiers and converters.

4.1.4 Prime Movers

As discussed above, the prime movers are the components of the propulsion or electric power system that convert chemical energy to mechanical energy. The most commonly used prime movers are:

- Diesel engine (DE). It is a reciprocating internal combustion engine. Depending on the output maximum rotational speed diesel engines are identified as low-speed (80-300rpm), medium-speed (300-1000rpm) and high-speed (over 1000rpm). Power output ranges between 1MW for the smallest high-speed engines to 80MW for the biggest low-speed engines (Woud and Stapersma, 2002). The diesel engine is generally preferred for merchant ship applications mostly due to its high fuel efficiency. Its biggest drawback is its low power density for higher power applications.

- Gas Turbine (GT). It is a rotating internal combustion engine. It is modular and easily automated but exhibits low fuel efficiency especially at partial load operation that leads to high fuel consumption. It is most commonly used in naval ships and applications where space is a major concern due to its high power density.

- Steam turbine plant. It consists of boilers where steam is generated by external combustion or a nuclear reaction, turbines where the steam expands delivering output power, a condenser where the steam condenses to water to re-enter the power cycle and a pump which feeds water into the boilers. The steam turbine plant is not so popular due to high fuel consumption, low power density and high initial costs.

4.1.5 Fuel Consumption Parameters

Prime mover fuel consumption is calculated from the formula:

$$fc = sfc \cdot BHP \tag{4.3}$$

where *sfc* is the prime mover specific fuel consumption. *sfc* expresses fuel quantity burnt per generated power per hour, usually expressed in $\frac{gr}{kWh}$. It depends on the specific engine and at any given loading condition on the power the engine produces. Figure 7 depicts generic *sfc* curves for different types of engines as a function of their working load.

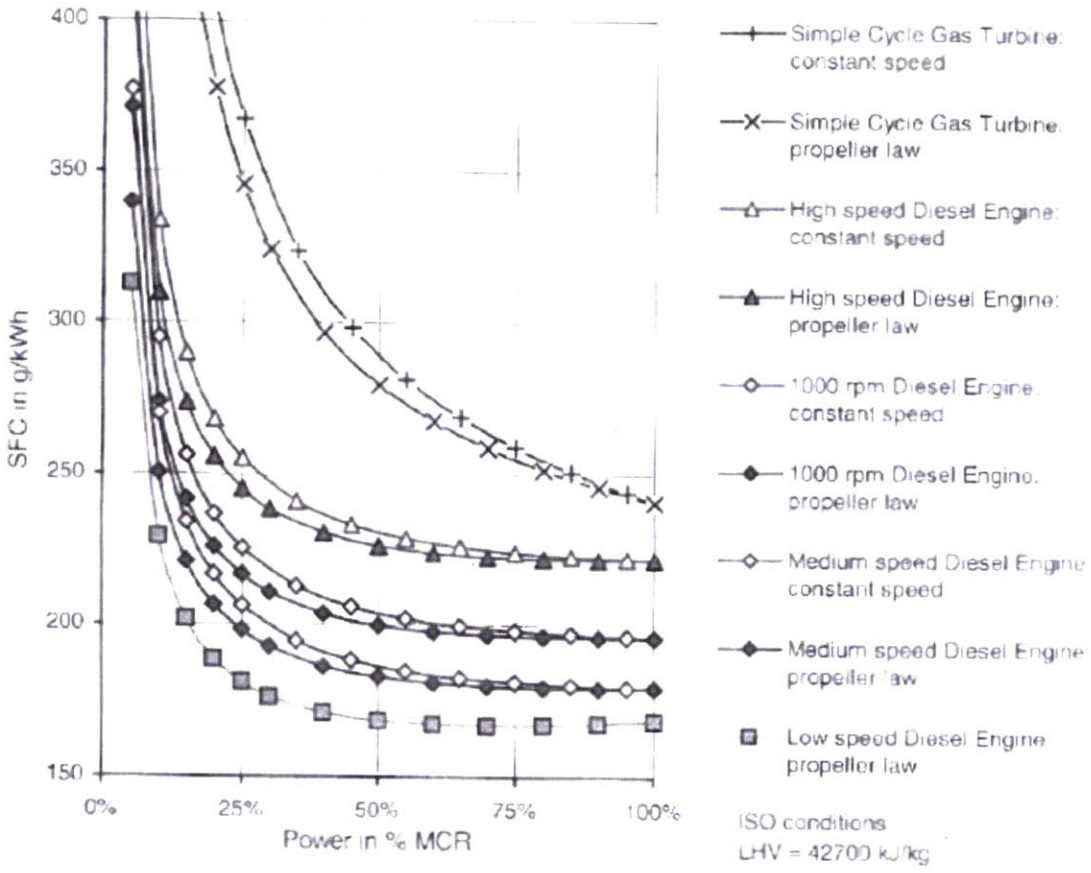


Figure 7 - Specific fuel consumption of prime movers (Woud and Stapersma, 2002)

4.2 Powering Modeling

4.2.1 Propulsion and Power Generation Systems

Taking into consideration:

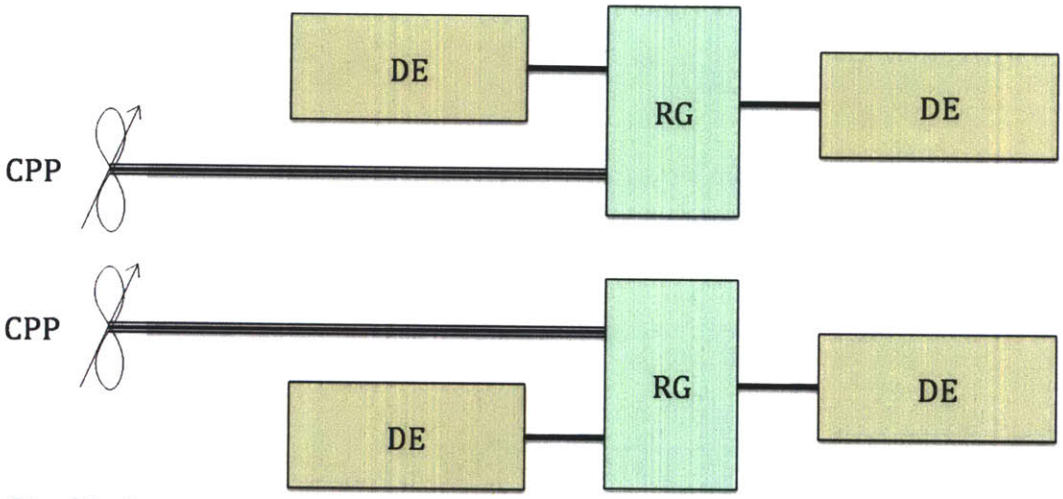
- the possible combinations between propulsion and electric power generation plants systems,
- the popularity of those systems in existing ship designs and design trends and
- the associated modeling complexity,

the following systems were selected for modeling:

4.2.1.1 Mechanical-Driven Propulsion with Separate Ship-Service Power Generation (Mech-SepSS)

This system comprises a distinct mechanically driven propulsion system and a distinct electric power generation system for ship-service. The following mechanically driven propulsion systems were modeled:

- Combined Diesel and Diesel (CODAD). Two diesel engines per main shaft exist. Each shaft can be driven by either of the two diesels or both.



DE=Diesel Engine
RG=Reduction Gear
CPP=Controllable-Pitch Propeller

Figure 8 - CODAD propulsion system representation

- Combined Gas and Gas (COGAG). Two gas turbines per main shaft exist. Each shaft can be driven by either of the two gas turbines or both.

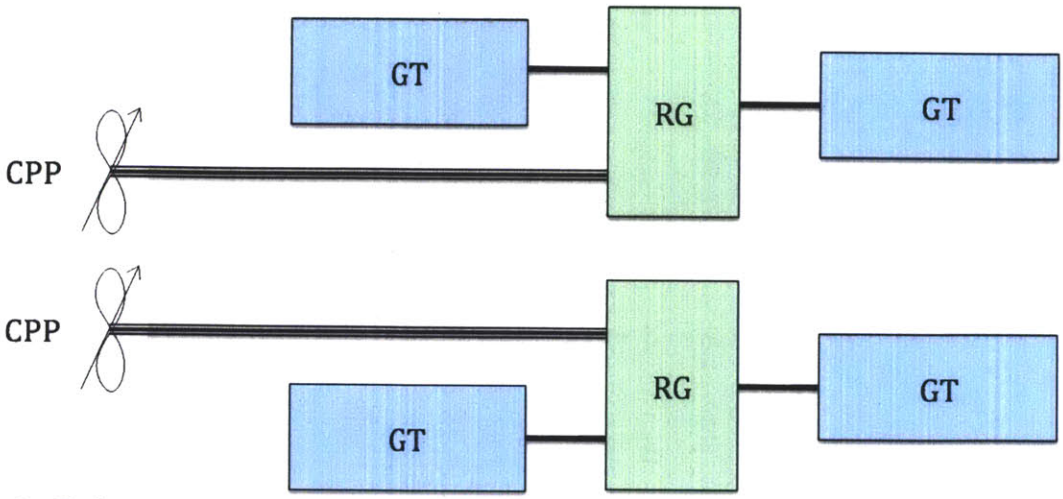
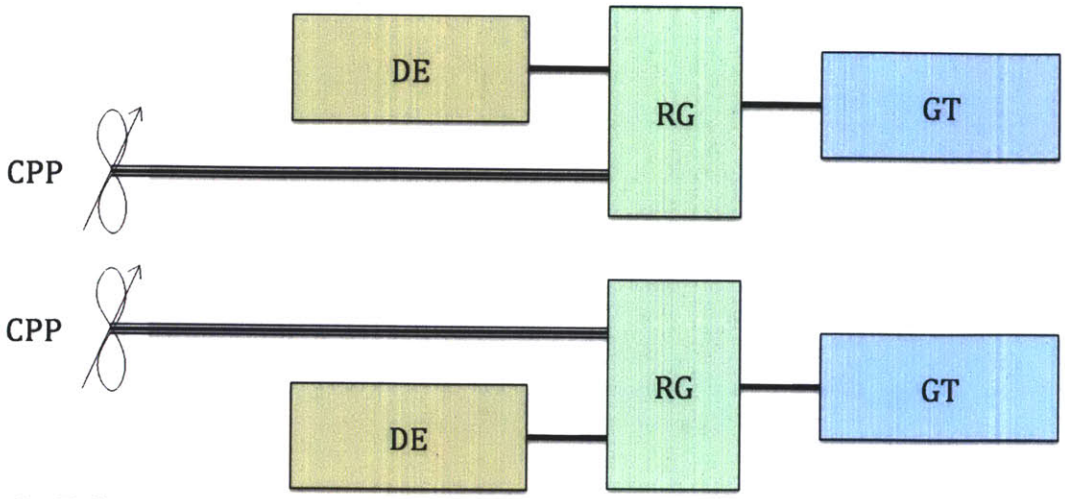


Figure 9 - COGAG propulsion system representation

- Combined Diesel and Gas (CODAG). One diesel engine and one gas turbine per main shaft exist. Each shaft can be driven by either the diesel or the gas turbine or both.

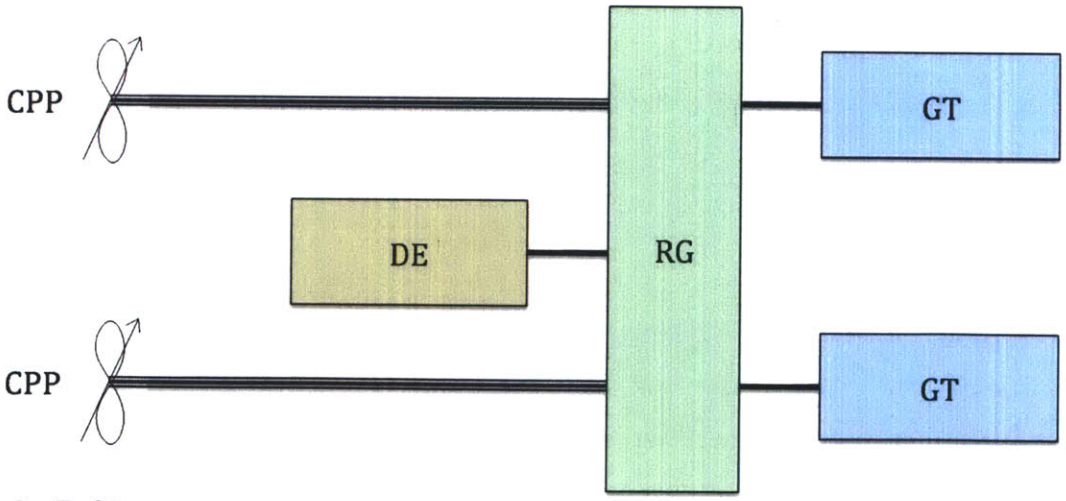
- Combined Diesel or Gas (CODOG). One diesel engine and one gas turbine per main shaft exist. Each shaft can be driven by either the diesel or the gas turbine.



GT=Gas Turbine
DE=Diesel Engine
RG=Reduction Gear
CPP=Controllable-Pitch Propeller

Figure 10 - CODOG/CODAG propulsion system representation

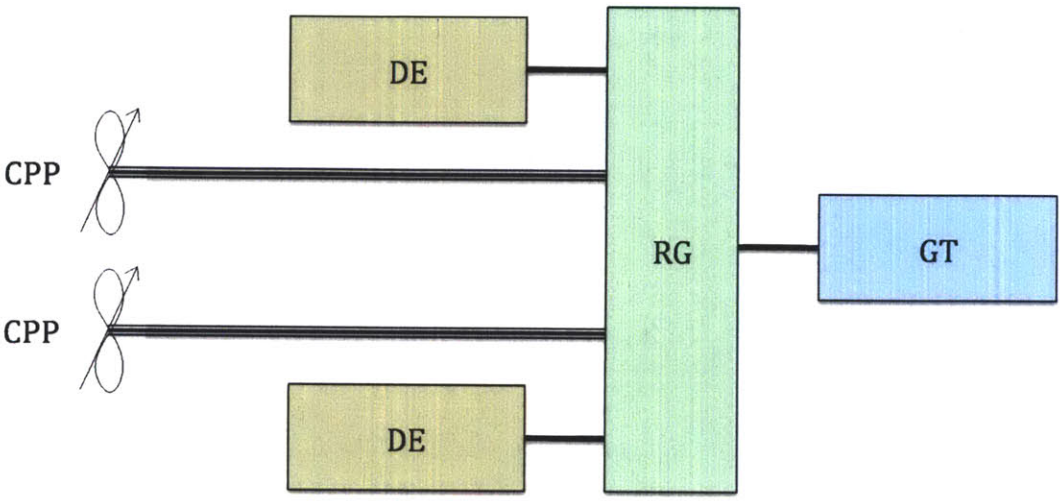
- Cross-connected propulsion system with one diesel and two gas turbines. One cross-connected reduction gear exists that allows all possible engine-working configurations.



GT=Gas Turbine
DE=Diesel Engine
RG=Reduction Gear
CPP=Controllable-Pitch Propeller

Figure 11 - Cross-connected propulsion system with 1DE and 2GTs

- Cross-connected propulsion system with two diesels and one gas turbine. One cross-connected reduction gear exists that allows all possible engine-working configurations.



GT=Gas Turbine
DE=Diesel Engine
RG=Reduction Gear
CPP=Controllable-Pitch Propeller

Figure 12 - Cross-connected propulsion system with 2DEs and 1GT

Ship-power generation systems were modeled with three or four generator sets. A typical separate ship-service power generation system with three generator sets is depicted in Figure 13.

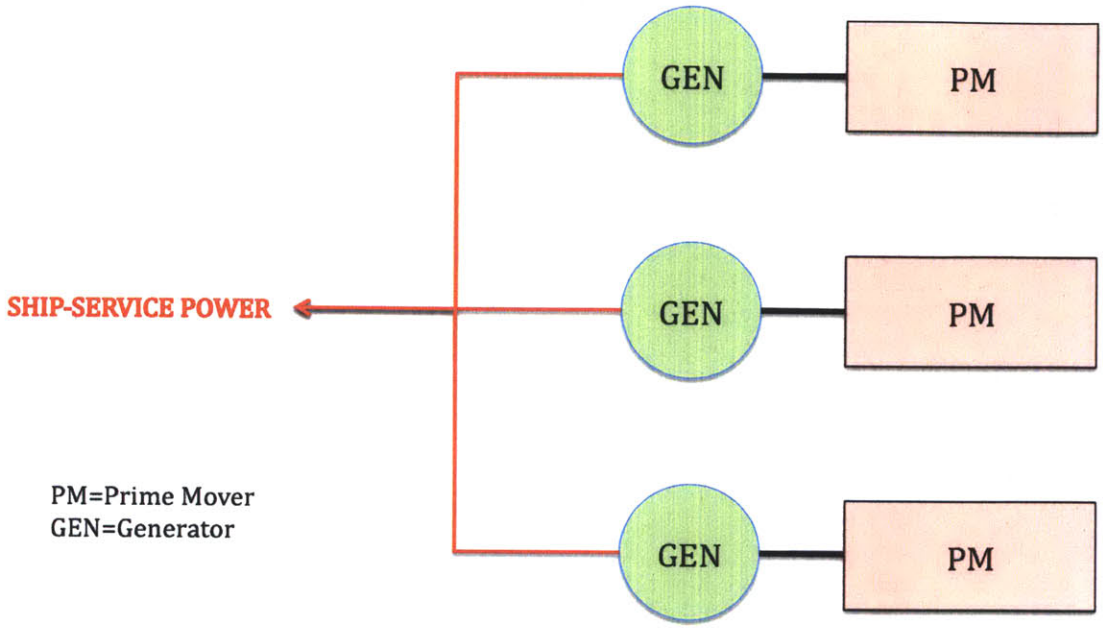


Figure 13 – Separate ship-service power generation representation

4.2.1.2 Mechanical-Driven Propulsion with Propulsion derived Ship-Service Power Generation (Mech-PDSS)

This system comprises a mechanically driven propulsion system and an electric power generation system whose generators are driven by the main propulsion shafts. For this purpose the ship-service power is called propulsion derived. In accordance with existing practices, a separate electric power generation may also exist to provide additional ship-service power. All mechanical-driven propulsion system options from paragraph 4.2.1.1 exist. If a separate ship-power generation system is present, it can be modeled with two or three generator sets. A CODAG/CODOG propulsion system with propulsion derived ship-service power is depicted in Figure 14.

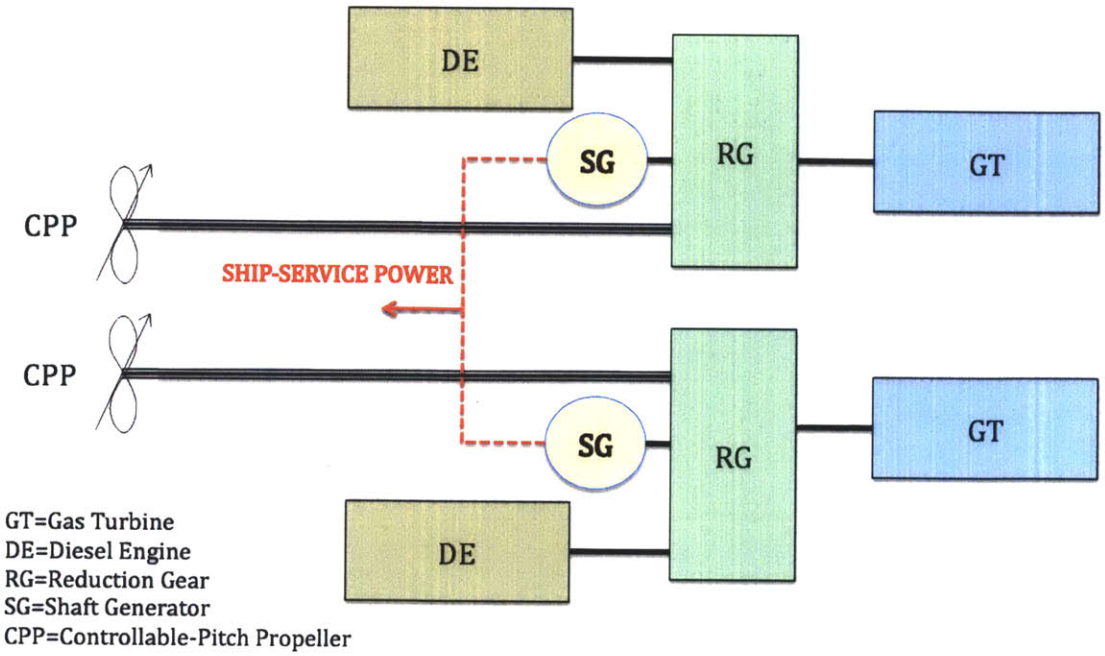


Figure 14 - CODAG/CODOG propulsion system with propulsion derived ship-service power

4.2.1.3 Integrated Power System (IPS)

In the IPS all available prime movers drive generators. The produced electric energy is used for propulsion and ship-service. The model can handle either a set of three identical prime movers or a set of two higher-power prime movers, which are called primary, and a set of two lower-power prime movers, which are called secondary. A power conversion module is responsible for converting the electric power for ship-service. The first case is depicted in Figure 15.

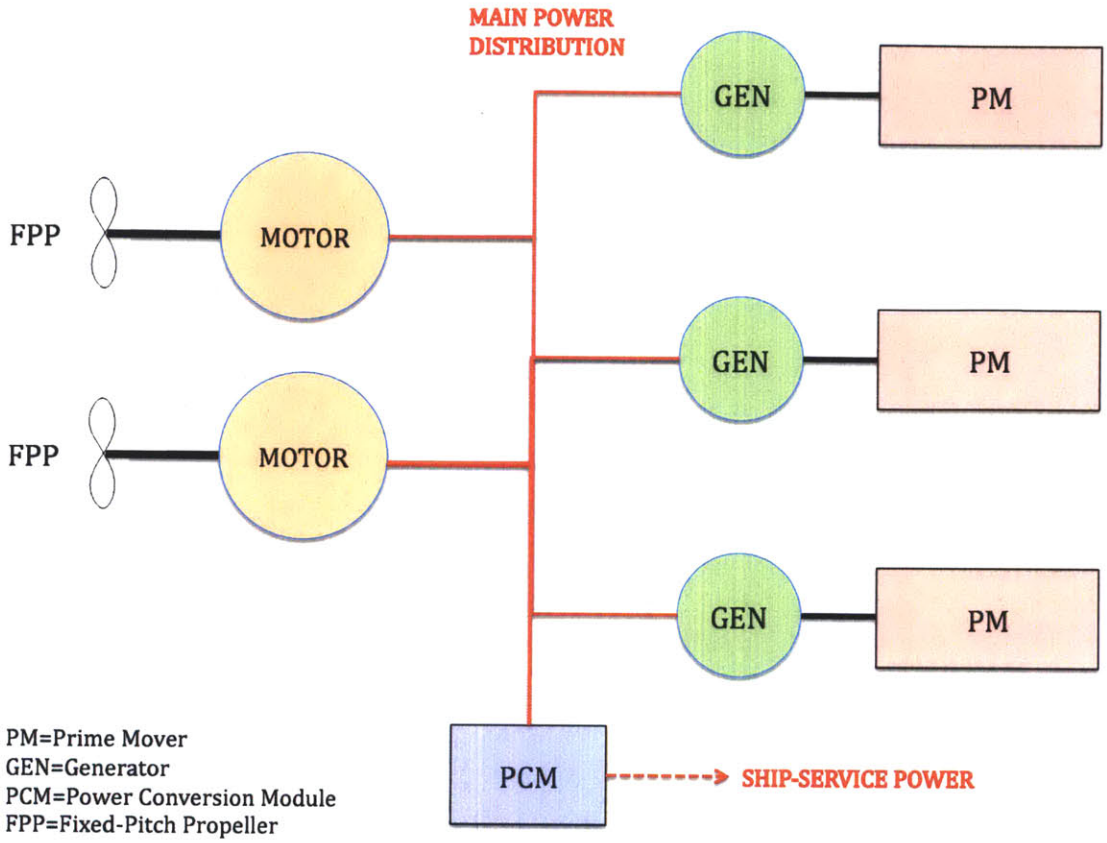


Figure 15 – IPS representation

4.2.2 Shaft Horsepower

As depicted in the propulsion and electric power systems representations above, mechanical-driven propulsion systems are paired with a set of controllable-pitch propellers while IPS are paired with a set of fixed-pitch propellers. The reason is that, unlike mechanical transmission, propulsion motors can easily reverse direction. The differences in propeller open-water efficiency between fixed- and controllable-pitch propellers are modeled by:

- Applying ASSET's efficiency multiplier ($\eta_{CP} = 0.997$) if a controllable-pitch propeller is present for the whole speed range:
- Assuming that the open-water efficiency at speeds less than 7knots remains constant for the controllable-pitch due to the pitch adjustment capability.

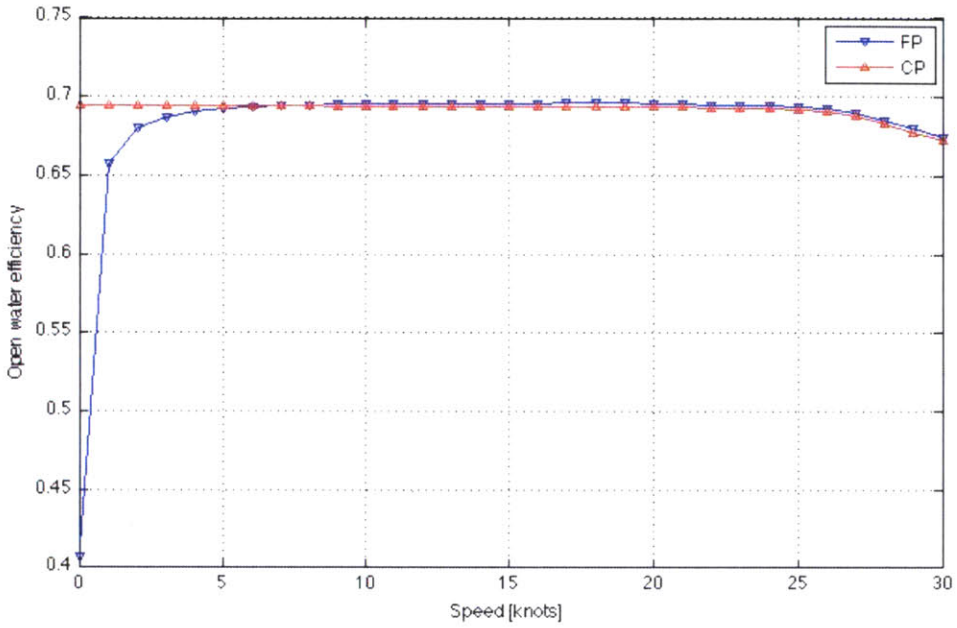


Figure 16 - Model open-water efficiency comparison between FP and CP propeller for DDG-51 Flight I

Total propulsive efficiency is calculated using equation (3.9) using the appropriate open-water efficiency.

Van Manen and van Oossanen (1988) propose sterntube and line bearing efficiencies as follows:

$$\eta_S \eta_b = \begin{cases} 0.98 \\ 0.97 \end{cases} \tag{4.4}$$

which corresponds to machinery being aft and amidships respectively. Lower efficiency represents higher bearing losses due to longer shaft lines. Common design practices suggest mechanical-driven propulsion systems to be placed amidships due to space and weight restrictions, while electric propulsion in IPS allows placement of the propulsion motors towards the stern. Consequently propulsion modeling assumes 0.98 sterntube and line bearing efficiency for IPS and 0.97 for mechanical-driven systems respectively.

The required shaft power (SHP) is calculated from the effective power (EHP) using equation (4.1).

4.2.3 Prime Movers

Due to their higher power density, gas turbines are an attractive solution for fast naval ships. Moreover, meeting increased range requirements relates to high fuel efficiency at the medium-speed range, which is strength of the diesel engine. On the other hand, steam turbine plants are becoming less popular and are still present mostly with nuclear power applications (aircraft carriers and submarines). Since the purpose of the program is to model fuel efficiency, the decision to model diesel and gas turbine engines only did not compromise adequate prime mover representation. In particular, diesel engine selection was limited to high-speed diesel engines only, due to their lower volume and weight advantages. Scarcity of engine fuel efficiency data and lack of consistency in installation power losses representation made compiling an exhaustive engine library impossible. On the other hand, a fairly representative one was derived based on ASSET's engine library. This ensured data consistency and adequate parameters representation.

Due to their operating differences, separate prime mover libraries were compiled for propulsion, ship-service, and integrated power systems and are included in Appendix A.

In order to produce the standard 60Hz current, generators and therefore ship-service engines have to operate at a standard rpm rate (600, 900, 1800, 3600, etc.). In order to increase engine options, diesel engine re-rating is assumed for engines whose maximum rpm is within a 5% range of a standard rating. Model uses ASSET's formula to calculate the rating factor:

$$R = \frac{n_{gen}}{n_{DE}} \tag{4.5}$$

where n_{gen} is the closest standard generator rating and n_{DE} is the diesel engine original maximum rpm. Likewise, the maximum engine power after re-rating is given from:

$$P_{RDE} = RP_{DE} \tag{4.6}$$

where P_{DE} is the maximum original engine power.

Prime movers for IPS are limited to guarantee that a representative system exists.

4.2.4 Transmission Efficiencies at Full-Load Operation

4.2.4.1 Mechanical Transmission

Since CP propellers provide the necessary reversing capability, there is no need to include reversing gear in the mechanical transmission. The power loss of a gearbox is subsequently based on reduction ratios and complexity. Woud and Stapersma (2002) suggest 1 to 2% losses for single step reduction gearboxes and 3% to 5% for more complex gearboxes. SNAME technical and Research bulletin (1975) proposes a 1% loss for each gear reduction. ASSET models transmission losses for full load operation to be 1.1% per reduction gear stage, but does not allow cross-connected gearboxes modeling.

Model assumes single reduction gears for diesel engines or gas turbines whose output shafts already includes a first stage reduction gear. Double reduction gears are modeled for all other gas turbines. If a cross-connected gear exists the increased complexity is modeled by applying an efficiency multiplier of 0.99. Modeling of mechanical transmission efficiencies is summarized in Table 9.

	Efficiency	**Simple gear**	**Cross-connected gear**
Driving Engine	Diesel	0.99	0.98
	Gas Turbine	0.98	0.97

Table 9 – Model mechanical transmission efficiencies at full-load operation

4.2.4.2 Electrical Transmission

Electrical transmission efficiencies generally increase with the rated installed power, but depend heavily on the type (AC or DC) of power generation and propulsion motor as depicted in Table 10.

Generation	Motor	Efficiency
AC	AC	88-95%
AC	DC	85-90%
DC	DC	80-86%

Table 10 - Total electrical transmission efficiencies (SNAME, 1974)

Although transmission efficiency is of high importance, selecting the transmission type (AC/DC) is not straightforward. High development rate of electric propulsion components and associated electrical power distribution options are only some of the factors that decide type of electric transmission to choose. Therefore, from a modeling standpoint, distinguishing electrical transmission types solely on total transmission efficiency did not make sense. Instead, the model assumes average values based on SNAME technical bulletin (1975) and ASSET for electrical transmission components. If required, the user can trivially reset those efficiencies for a given system evaluation.

	Efficiency
Generator	0.960
Motor drive	0.960
Propulsion motor	0.960
Total	0.885

Table 11 - Model electric components efficiencies at full-load operation

The generator efficiency is also used for separate ship-service power generation. In a propulsion derived ship-service (PDSS) power system, a variable-speed constant-frequency cycloconverter is assumed to adapt the output power from the shaft generator to the conventional AC ship-service bus. In the IPS a power conversion module (PCM) converts the main power to the desired ship-service voltage. Ship-service power conversion efficiencies are depicted in Table 12.

System	Basic Components	Efficiency
Mech-PDSS	Shaft Generator and Cycloconverter	0.915
IPS	Power Conversion Module (PCM)	0.965

Table 12 - Ship-service power conversion efficiencies

IPS power generation arrangement assumes independent generator sets. Therefore, no combining gear losses are modeled.

4.2.5 Prime Mover Operation

For the mechanical-driven propulsion systems the model allows three or four engines to be present. These are characterized as cruise or sprint engines depending on the speed regime they are intended to work. Cruise engines are typically operating in low speeds while sprint engines in high speeds. Depending on the reduction gear cruise engine may or may not contribute power at high speeds as well.

Cruise Engines		Sprint Engines		Propulsion System
Number	**Type**	**Number**	**Type**	
2	DE	2	DE	CODAD
2	GT	2	GT	COGAG
2	DE	2	GT	CODOG/CODAG
2	DE/GT	1	DE/GT	Cross-connected
1	DE/GT	2	DE/GT	Cross-connected

Table 13 - Propulsion systems' cruise/sprint engines identification

4.2.5.1 Mechanical-Driven Propulsion with Separate Ship-Service Power Generation (Mech-SepSS)

In order to model the prime mover operation the engine load fraction is defined from the formula:

$$LF = \frac{BHP_{USE}}{BHP_{AVAIL}} \tag{4.7}$$

where BHP_{USE} is the power an engine provides for a given condition and BHP_{AVAIL} is the maximum available engine power. From the prime mover perspective there are three cases:

- $LF = 0$. Prime mover is not operating.
- $0 < LF < 1$. Primer mover is operating at partial load.
- $LF = 1$. Prime mover is operating at full load.

Prime movers operation as described by the engine load fraction indicates the sequence, which engine go online as power requirements gradually increase. The operating philosophy adopted is reaching full-load operation before the next available engine set goes online, where the system allows. For redundancy purposes no less than two engines are allowed to operate at any time when ship-service power is exclusively provided by those engines.

For each propulsion system the prime mover operation modeling is described in Table 14.

Propulsion scheme	Prime mover	Speed →				
		LF				
CODAD	2 cruise DE	0-1				1
	2 sprint DE	0				0-1
COGAG	2 cruise GT	0-1				1
	2 sprint GT	0				0-1
CODOG	2 cruise DE	0-1				0
	2 sprint GT	0				0-1
CODAG	2 cruise DE	0-1				1
	2 sprint GT	0				0-1
Cross-connected 2cruise+1sprint	1st cruise	0-1	1	0	1	1
	2nd cruise	0	0-1	0	0	1
	sprint	0	0	0-1	0-1	0-1
Cross-connected 1cruise+2sprint	cruise	0-1		0	1	1
	1st sprint	0		0-1	0-1	1
	2nd sprint	0		0	0	0-1

Table 14 - Propulsion prime movers operation modeling

It is obvious that cruise engines operate at slower speeds and sprint engines engage at higher speeds. For the cross-connected configurations engine operation is such that better fuel efficiency cruise engines work at lower speeds and sprint engine contribute to go faster. As expected, cross-connected gears exhibit better operating flexibility.

Model allows three or four separate ship-service engines. In accordance with naval design practices, at least two of them are on line at any time and the necessary ship-service power can be met with one less engine operating for redundancy. Furthermore,

all operating ship-service engines are modeled to share the electric load until the 90% of their rated power. This imbalance factor accounts for transitional imbalances of electrical loads on individual ship-service systems.

Ship-Service Engines	Prime mover	Power / Load Fraction	
3	1st	0-0.9	
	2nd	0-0.9	
	3rd	0	
4	1st	0-0.9	0.6-0.9
	2nd	0-0.9	0.6-0.9
	3rd	0	0.6-0.9
	4th	0	0

Table 15 - Ship-service prime movers operation modeling

Due to the imbalance factor the maximum assumed load fraction is 0.9. Three generator sets operate only when the first two cannot meet the ship-service power requirements.

4.2.5.2 Mechanical-Driven Propulsion with Propulsion derived Ship-Service Power Generation (Mech-PDSS)

Model calculates usable propulsion derived power as the excess power that propulsion prime movers can generate at any given speed. A separate ship-service power generation system can be present or not. If it is, model allows two or three separate ship-service engines to be present. As discussed previously, only one or two of them are expected to provide all the separate ship-service power needed. On the other hand, single separate ship-service engine operation is allowed and no imbalance factor is imposed. Propulsion prime movers operation is modeled as depicted in Table 14. If separate ship-service prime movers exist they are modeled as shown in Table 16.

Ship-Service Engines	Prime mover	Power Load Fraction	
2	1st	0-1	
	2nd	0	
3	1st	0-1	1
	2nd	0	0-1
	3rd	0	0

Table 16 – Separate ship service prime movers operation modeling when propulsion derived ship service exists

4.2.5.3 Integrated Power System (IPS)

In the IPS all prime movers provide propulsion and ship-service power. No less than two prime movers are allowed to operate anytime for redundancy. Operation and load sharing are depicted in Table 17.

IPS scheme	Prime mover	Speed/Power Load Fraction					
3 Primary	1st	0-0.5		1		1	
	2nd	0-0.5		0-1		1	
	3rd	0		0		0-1	
2 Primary +2 Secondary	1st Secondary	0-1	1	1	0	1	1
	2nd Secondary	0-1	0	1	0	0	1
	1st Primary	0	0-1	0-1	1	1	1
	2nd Primary	0	0	0	0-1	0-1	0-1

Table 17 - IPS prime mover operation modeling

4.2.6 Fuel Consumption

Prime mover fuel consumption at full load operation is calculated from equation (4.3) using specific engine information included in Appendix A. Estimating fuel consumption for partial-load operation requires extensive specific fuel consumption information for each prime mover as well as the engine load fraction.

Instead of trying to model each engine's *sfc*, the generic curves from Woud and Stapersma (Figure 7) where used. Figure data were regenerated in Matlab® by using several data points and performing cubic spline interpolation. Modeling covered high-speed diesel engines and gas turbines at constant speed and at propeller law operation. Constant speed represents prime mover that is paired with a generator (separate ship-service power generation or IPS) while propeller law represents prime mover that is used for propulsion. Engine *sfc* is then scaled to the specific engine using the *sfc* at 100% load information included in Appendix A. Furthermore, Figure 7 represents *sfc* data for fuel with Lower Heating Value (LHV) of $42700\frac{KJ}{kg}$. Assuming that engine *sfc* does not change with the use of different fuel, partial-load *sfc* data is adjusted for the naval F-76 fuel specification using the formula:

$$sfc = sfc_{graph}\frac{LHV_{graph}}{LHV_{F-76}} \tag{4.8}$$

LHV for F-76 is $42800\frac{KJ}{kg}$.

Engine load fraction was determined from equation (4.5).

4.2.7 Transmission Efficiency Losses at Partial-Load Operation

SNAME technical bulletin (1975) includes transmission efficiencies losses for partial-load operation. Figures 17 and 18 depict losses for mechanical and electrical transmissions respectively.

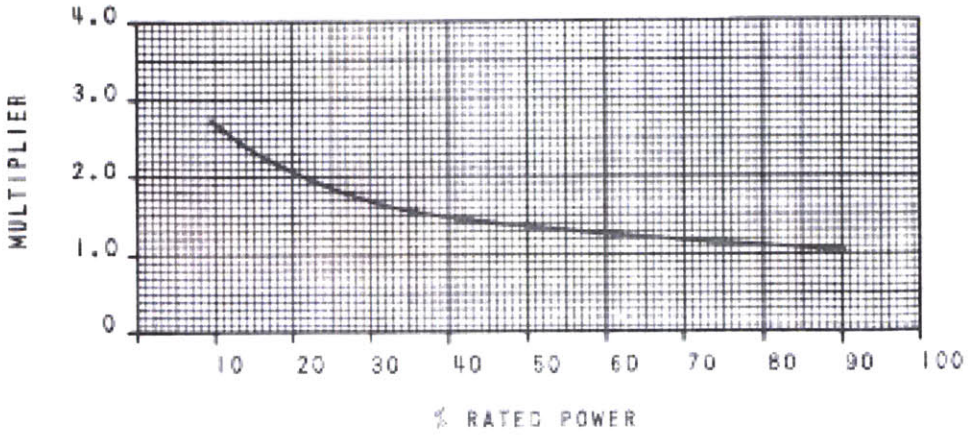


Figure 17 - Mechanical transmission loss multiplier for partial-load operation (SNAME, 1975)

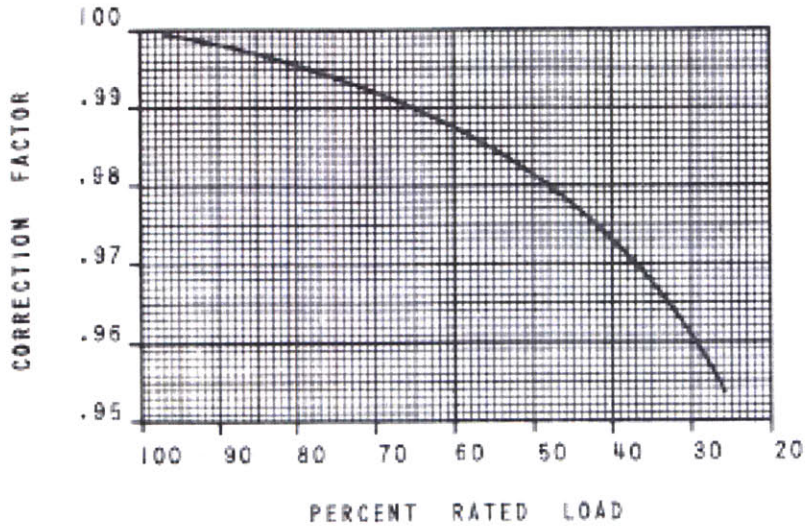


Figure 18 - Electrical transmission efficiency correction for partial-load operation (SNAME, 1975)

Figure data were regenerated in Matlab® by using several data points and performing cubic spline interpolation.

The mechanical transmission efficiency for partial-load operation is given from:

$$\eta_{TR/PL} = 1 - (1 - \eta_{TR})LM \tag{4.9}$$

where LM is the loss multiplier from Figure 17.

The electrical transmission efficiency for partial-load operation is given from:

$$\eta_{TR/PL} = CF\eta_{TR} \tag{4.10}$$

where CF is the correction factor from Figure 18.

Transmission efficiencies losses at partial load require the engines to operate higher to overcome those losses. The respective load fraction as defined in Equation (4.7) is therefore updated to adequately model increased fuel consumption due to partial load operation.

4.2.8 Power Service Margin

The power service margin (M_S) is used to ensure that the installed power can overcome added resistance from hull fouling, waves, wind, shallow water effects, etc (Parsons, 2004). Model incorporates this margin in the maximum required engine power estimation using the formula:

$$BHP_{\max}(1 - M_S) = \frac{SHP_{\max}}{\eta_{TR}} \tag{4.11}$$

Since power service margin application is case sensitive, it is left as a user input to the program.

4.3 Model Validation

Model was validated against ASSET by comparing propulsion fuel consumption and specific fuel consumption estimation for the DDG-51 Flight I destroyer. Results are depicted in Figures 19 and 20.

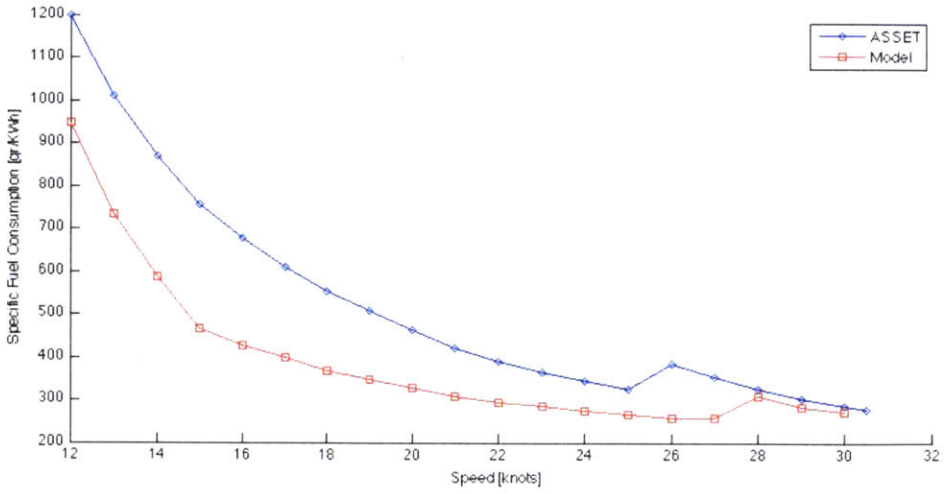


Figure 19 – Propulsion specific fuel consumption comparison between model and ASSET for DDG-51 Flight I

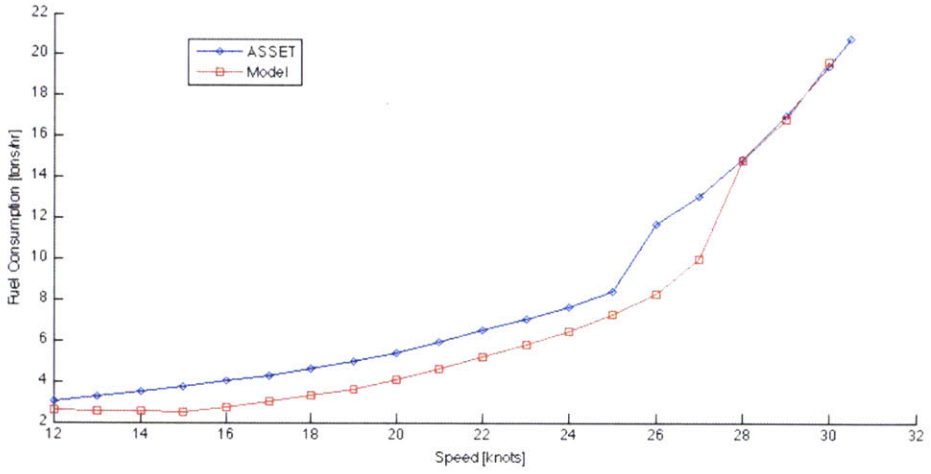


Figure 20 – Propulsion fuel consumption comparison between model and ASSET for DDG-51 Flight I

DDG-51 destroyers employ four GE LM-2500 gas turbines in a COGAG configuration. With such systems ships opt for trailing shaft operation for lower speeds to decrease fuel consumption. As a result, comparison seemed more relevant and was held for the speed range 12-30 knots.

Although exhibiting the same pattern, the first impression of the model is that it underestimates fuel consumption and specific fuel consumption in comparison to

ASSET. Moreover, the steep change at the area of 26-28knots, associated with the second set of gas turbines going on line, occurs later for the model (28knots vs. 26knots for ASSET). All these can be justified by taking into account that ASSET derives its results by adding margins for hull fouling, wave drag and machinery degradation. One would easily note that similar margins should be considered for the fuel consumption model. Incorporating power service margins for the whole speed range is debatable and since the purpose of the program is to identify the most fuel-efficient system, and not estimate its fuel consumption per se, no further effort was made to improve the fidelity of the model towards this dimension.

CHAPTER 5

SYSTEMS OPTIMIZATION

5.1 Problem Formulation

System optimization was performed independently for the three propulsion and power generation systems discussed in chapter 4. This allowed easier problem formulation for optimization and provided shorter computation times. Design parameters are common for the three systems' categories and are analyzed in 5.1.4. The design variables are integers since they represent number, type or combination of prime movers.

5.1.1 Mechanical-Driven Propulsion with Separate Ship-Service Power Generation (Mech-SepSS)

Design variables are:

- Cruise engines identification (ID_{CR})
- Sprint engines identification (ID_{SP})
- Cruise engines number (N_{CR})
- Sprint engines number (N_{SP})
- Combination identification for propulsion engines (ID_{COMB})
- Ship-service engine identification (ID_{SS})
- Ship-service engine number (N_{SS})

Design parameters are:

- Ship operating profile

- Shaft power profile
- Power service margin

		Lower Bound	Upper Bound
Design Variables	ID_{CR}	1	57
	ID_{SP}	1	57
	N_{CR}	1	2
	N_{SP}	1	2
	ID_{COMB}	0	1
	ID_{SS}	1	52
	N_{SS}	3	4

Table 18 - Mechanical-driven propulsion with separate ship-service power generation design variables' bounds

The design objective is to minimize average fuel consumption such that:

- Installed propulsion power meets the necessary shaft power required to achieve the maximum speed (inequality constraint)
- Installed ship-service power meets the maximum ship-service power required (inequality constraint)

5.1.2 Mechanical-Driven Propulsion with Propulsion derived Ship-Service Power Generation (Mech-PDSS)

Design variables are:

- Cruise engines identification (ID_{CR})
- Sprint engines identification (ID_{SP})
- Cruise engines number (N_{CR})
- Sprint engines number (N_{SP})
- Combination identification for propulsion engines (ID_{COMB})

- Separate ship-service system identification (ID_{SEP-SS})
- Ship-service engine identification (ID_{SS})
- Ship-service engine number (N_{SS})

Design parameters are:

- Ship operating profile
- Shaft power profile
- Power service margin

		Lower Bound	Upper Bound
Design Variables	ID_{CR}	1	57
	ID_{SP}	1	57
	N_{CR}	1	2
	N_{SP}	1	2
	ID_{COMB}	0	1
	ID_{SEP-SS}	0	1
	ID_{SS}	1	52
	N_{SS}	3	4

Table 19 - Mechanical-driven propulsion with propulsion derived ship-service power generation design variables' bounds

The design objective is to minimize average fuel consumption such that:

- Installed propulsion power meets the necessary shaft power required to achieve the maximum speed (inequality constraint)
- Installed total power meets the maximum total power required (inequality constraint)

5.1.3 Integrated Power System (IPS)

Design variables are:

- Primary engines identification number (ID_{PRI})
- Secondary engines identification number (ID_{SEC})
- Primary engines number (N_{PRI})

Design parameters are:

- Ship operating profile
- Shaft power profile
- Power service margin

		Lower Bound	Upper Bound
Design Variables	ID_{PRI}	1	13
	ID_{SEC}	1	13
	N_{PRI}	2	3

Table 20 – IPS design variables' bounds

The design objective is to minimize average fuel consumption such that installed total power meets the maximum total power required (inequality constraint)

5.1.4 Design Parameters

A ship's speed profile describes how much time the ship spends at each speed. Likewise, a ship's ship-service power profile describes how much time the ship utilizes certain amount of ship-service electric power. Although independent speed and ship-service power profiles suffice for the mechanical-driven propulsion system with separate ship-service power generation fuel consumption calculations, concurrent speed and ship-service power information is required to adequately model fuel consumption for the other two systems. As a result, program is set up to import a user-specified combined profile. Combined profile information is also used to extract maximum speed and ship-service power requirements.

The shaft power profile describes the necessary shaft power required to achieve each speed. It can be user input or estimated by the program as described in Chapter 4. When

the program estimates shaft power all relevant ship resistance and propeller parameters qualify as design parameters.

Examples of both combined operating and shaft power profiles are included in Appendix B.

The power service margin is used to determine the installed propulsive power that is required to meet the maximum speed as was described in 4.2.8.

5.2 Optimization Algorithm

5.2.1 Selection

Minimizing average fuel consumption represents a single-objective problem with integer design variables. Although a combinatorial approach would work for the IPS category (2*13*13=338 combinations), the same cannot be said for the rest (approximately 1.45 and 2.70 million combinations evaluations are required for mechanical-driven propulsion systems without or with propulsion derived ship-service power generation respectively). Moreover, the optimization algorithm should be able to handle robustly larger engine libraries and increased modeling combinations that may occur in the future.

The integer nature of the design variables did not allow optimization using gradient-based techniques. From the available heuristic optimization techniques the genetic algorithm was preferred by virtue of simplicity of application. It is a method based on natural selection where a population of design vectors (individuals) is chosen and a series of "genetic" operations are carried out to determine the optimal solution represented by the fittest individual.

5.2.2 Genetic Algorithm Implementation

The genetic algorithm was implemented using Matlab®'s *ga* build in function.

5.2.2.1 Representation

In the genetic algorithm language each design variable is a "gene" the string of design variables form a "chromosome". Integer design variables had to be encoded in binary format as depicted in Figures 21-23.

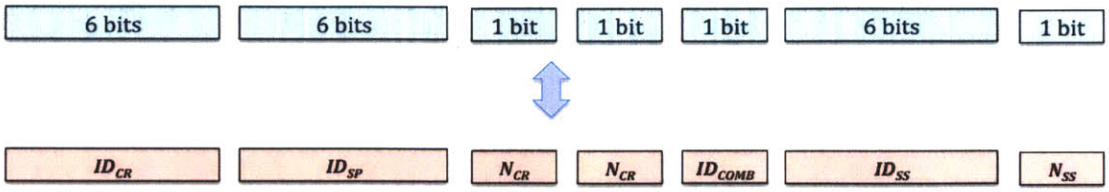


Figure 21 - Chromosome representation for Mech-SepSS

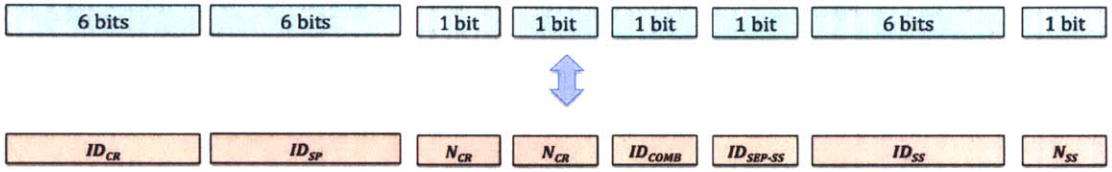


Figure 22 - Chromosome representation for Mech-PDSS

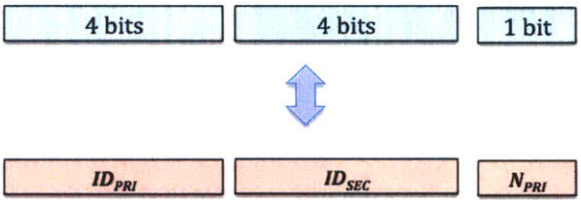


Figure 23 - Chromosome representation for IPS

Number of bits (n) was chosen to allow adequate design variables representation. For this purpose engine lists imported to Matlab® were modified to include exactly 2^i (i: integer) entries by repeating some entries. Although this affects initial population selection, it does not compromise overall algorithm performance.

5.2.2.2 Fitness Function and Constraints

Fitness function is defined as the average total fuel consumption plus penalties for not meeting the constraints for each case. Average total fuel consumption is the sum of average propulsive and ship-service fuel consumption. Average propulsive fuel consumption is the sum of fuel consumption for each speed weighted by the time spent in

each speed as described by the combined operating profile and average ship-service fuel consumption is the sum fuel consumption for each ship-service power segment weighted by the time spent in each segment as described by the combined operating profile.

Matlab®'s genetic algorithm with binary encoding cannot handle constraints outside the function that is evaluated. As a result, penalties to the fitness function were introduced for constraint violation. Furthermore, to enable feasibility for each function evaluation, a nominal total fuel consumption was calculated when the selected system violated the constraints based on the specific fuel consumption of the largest selected at maximum power. This fuel consumption represented full load operation on purpose to give the fitness function higher values. On top of that penalties were imposed depending on the system type as follows:

- Mech-SepSS:
  - Not enough installed propulsion power:

$$PEN = 0.5\left(SHP_{\max} - BHP_{INST}\eta_{TR}(1 - M_S)\right) \tag{5.1}$$

  - Not enough installed ship-service power:

$$PEN = 0.5\left(SSP_{\max} - SSP_{INST}\eta_{GEN}c_{imb}\right) \tag{5.2}$$

- Mech-PDSS:
  - Not enough total installed power:

$$PEN = 0.5\left(SHP_{\max} + SSP_{\max} - BHP_{INST}\eta_{TR}(1 - M_S) - SSP_{INST}\eta_{PDSS}\right) \tag{5.3}$$

  - Or total power suffices but not enough propulsion power:

$$PEN = 0.5\left(SHP_{\max} - BHP_{INST}\eta_{TR}(1 - M_S)\right) \tag{5.4}$$

- IPS:
  - Not enough total installed power:

$$PEN = 0.5\left(\frac{SHP_{\max}}{\eta_{IPS-PROP}} + \frac{SSP_{\max}}{\eta_{IPS-SS}} - BHP_{INST}(1 - M_S)\right) \quad (5.5)$$

where:

$SHP_{\max}$ is the maximum required shaft power.

$SSP_{\max}$ is the maximum required ship-service power.

BHP_{INST} is the total installed brake power except for CODOG where it is the sprint engines installed power.

SSP_{INST} is the installed separate ship-service power (not propulsion derived) subtracting one generator set for redundancy.

η_{TR} is the mechanical transmission efficiency

η_{GEN} is the generator efficiency

c_{imb} is the imbalance factor

η_{PDSS} is the shaft generator and cycloconverter efficiency for propulsion derived ship-service power.

$\eta_{IPS-PROP}$ is the propulsion total efficiency for IPS

η_{IPS-SS} is the ship-service total efficiency for IPS

M_S is the power service margin

5.2.2.3 Genetic Operators and Algorithm Tuning

The following genetic operators had to be set for binary encoding and algorithm tuning:

- Population options:
    - Population type. It specifies the data type of the input to the fitness function.
    - Population size. It specifies how many individuals are in each generation. With a large population size, the genetic algorithm

searches the design space more thoroughly, thereby increasing the probability of finding the global minimum. However, a large population size also causes the algorithm to run more slowly.

- Initial population. It specifies an initial population for the algorithm.

- Selection option. It specifies how the algorithm selects individuals as parents to create the next generation.

- Reproduction options:

  - Elite count. It specifies the number of the best individuals the algorithm carries unchanged to the next generation.

  - Crossover fraction. It specifies the fraction of the next generation, other than elite children, that are produced by crossover.

- Mutation option. It specifies how the algorithm makes small random changes in the individuals in the population to create mutation children. Mutation provides genetic diversity and enables the genetic algorithm to search a broader design space.

- Crossover option. It specifies how the algorithm combines two individuals to form a crossover child for the next generation.

The following table summarizes genetic operators final settings in Matlab®:

Matlab® option	Handle/Value	Comment	
CreationFcn	@gacreationuniform	Population type restriction	
CrossoverFcn	@crossoverscattered	Population type restriction	
CrossoverFraction	0.9	Algorithm tuning	
EliteCount	1	Algorithm tuning	
MutationFcn	@mutationuniform	Population type restriction	
PopulationSize	400	Mech-SepSS	Algorithm tuning
	400	Mech-PDSS	
	40	IPS	
PopulationType	bitstring	Population type restriction	
SelectionFcn	@selectionstochunif	Algorithm tuning	

Table 21 - Genetic Algorithm operators' settings in Matlab®

5.2.2.4 Convergence and Global Optimum

Figure 24 shows algorithm convergence for a mechanical-driven propulsion with separate ship-service power generation system optimization for the DDG-51 Flight I destroyer.

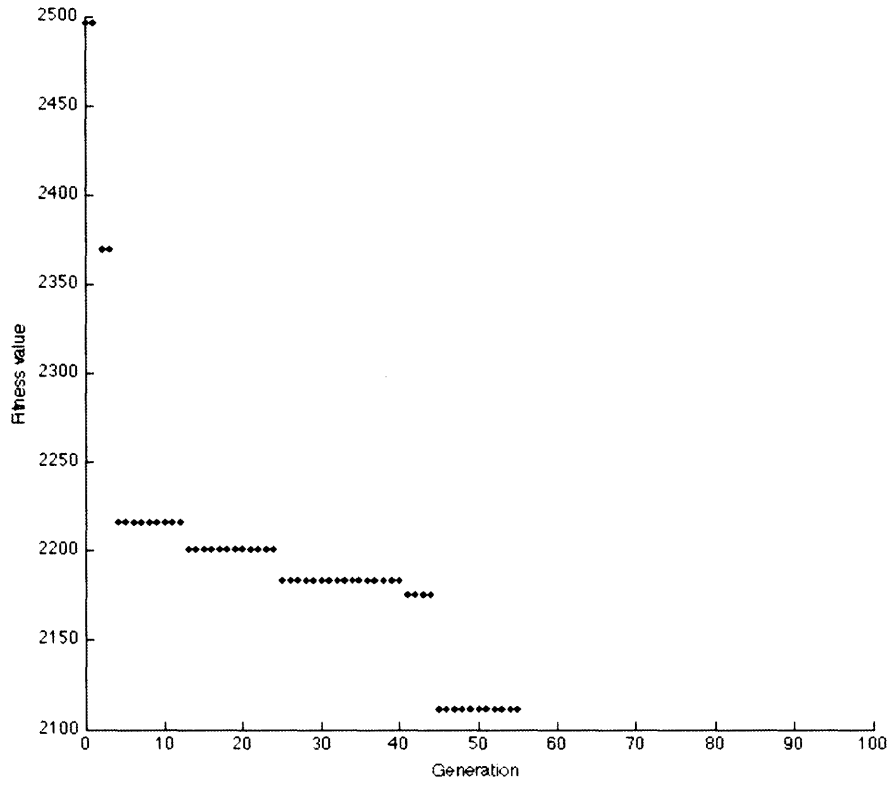


Figure 24 - Genetic algorithm best fitness convergence for the DDG-51 Flight I destroyer

Convergence history indicates appropriate fitness function and penalties scaling. Similar convergence patterns were exhibited for the other two systems.

Although the genetic algorithm searches more thoroughly the design space due to the embedded randomness, there is no guarantee that the global optimum will be found. Increased population size yield better results with the expense of increased run times. Genetic algorithm computational effort for individual system categories optimization is depicted in Table 22.

System	Function Evaluations	Time
Mech-SepSS	~20000	~2min
Mech-PDSS	~20000	~10min
IPS	~2000	~80sec

Table 22 - Computational effort for systems optimization

Optimization time is noticeably affected by the population size and the time required for each function evaluation. Mechanical-driven propulsion with propulsion derived ship service power systems were the most computationally expensive to evaluate.

The algorithm set up described above is specific enough for the particular system category but generic enough to accommodate different design parameters. Taking also into account the inherent difficulty of a global optimum search using a genetic algorithm there was need to robustly address the global optimum search problem. As discussed above, increasing the genetic algorithm's population size provided a more complete design space search. Although this came with a significant increase in computational effort, global optimality was still not guaranteed. In order to address this problem, an option for the program was set up to perform a series of ten, less demanding, algorithm evaluations for each system category. Leveraging the randomness of the initial population formation the program exhibited superior performance in attaining the global optimum. A robust global optimum search for all three, system categories required around 45 minutes.

5.3 DDG-51 Flight I Destroyer Systems Selection for Optimal Fuel Consumption

DDG-51 Flight I systems selection for optimal fuel consumption was done using ASSET's shaft power information and a zero power service margin. Combined operating profile is based on the NAVSEA speed profile for DDG-51 (NAVSEA, 2002) and the ship-service power profile is normally distributed around the mean of 2.4MW with a

standard deviation of 0.4MW for all speeds. All relevant information is included in Appendix B.

5.3.1 Optimization Results

Systems selection for optimal fuel consumption for each system category is depicted in Table 23.

System	Configuration		Average Fuel Consumption
Mech-SepSS	Cruise Engines	2xMTU 20V956 (DIESEL/3885kW)	2112 kg/hr
	Sprint Engines	2xGE LM6000PC (GT/37285kW)	
	Configuration	CODOG	
	Ship-service engines	4xMTU 16V396* (DIESEL/1487kW)	
Mech-PDSS	Cruise Engines	2xMTU 20V956 (DIESEL/3885kW)	2145 kg/hr
	Sprint Engines	2xGE LM6000PC (GT/37285kW)	
	Configuration	CODOG	
	Separate ship-service engines	3xMTU 12V538* (DIESEL/1830kW)	
IPS	Primary engines	2xGE LM6000PC (GT/37285kW)	2812 kg/hr
	Secondary engines	2xRRA 601KF9G (GT/6617kW)	

Table 23 - DDG-51 Flight I destroyer optimization results

Figures 25 and 26 depict fuel consumption and specific fuel consumption respectively as a function of speed. Specific fuel consumption is calculated for the average ship-service power (~2.4MW).

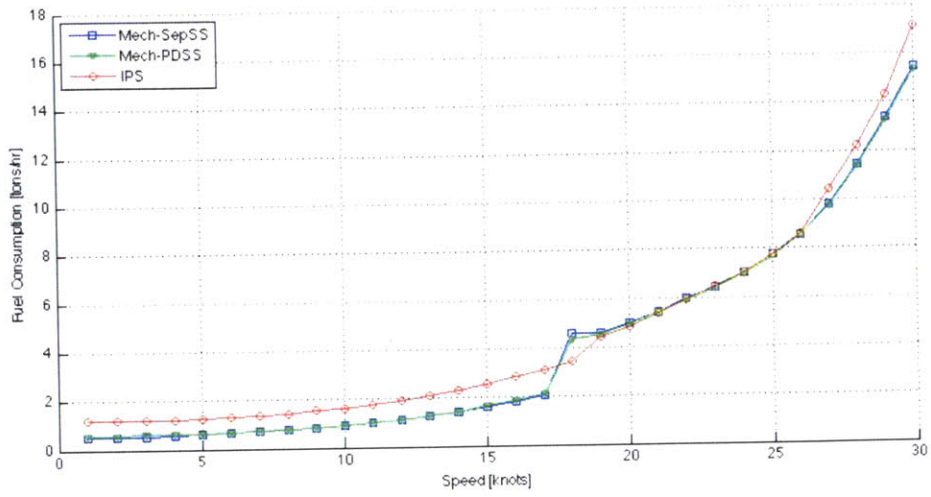


Figure 25 - Optimal fuel consumption profiles for the DDG-51 Flight I destroyer

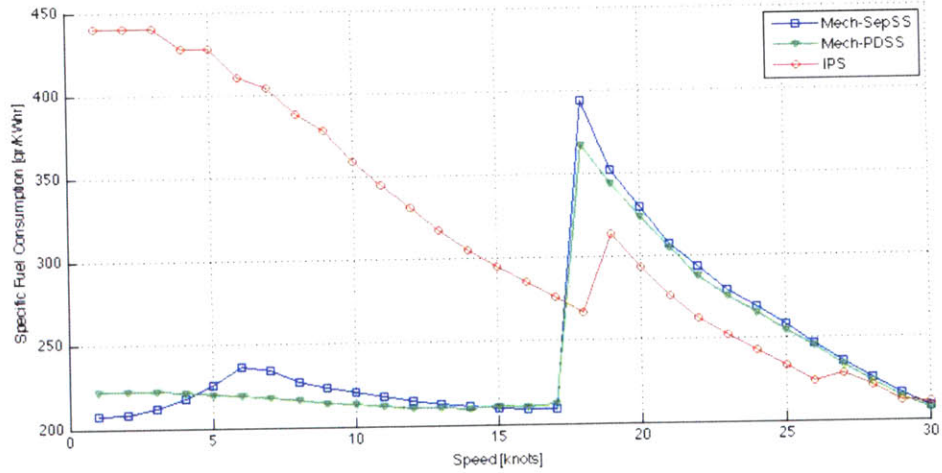


Figure 26 - Optimal specific fuel consumption profiles for the DDG-51 Flight I destroyer

It is obvious that the optimal IPS configuration is less fuel-efficient than the mechanical-driven options. This can be attributed to:

- IPS prime movers are all gas turbines, which exhibit high specific fuel consumption at partial load operation, which occurs at lower speeds (<17knots).

- IPS electrical transmission exhibits higher losses comparing to mechanical transmission meaning more engine power is required at any time. Therefore, although IPS specific fuel consumption is better for the speed range 18-29knots, fuel consumption remains comparable to the mechanical-driven systems.

- IPS average fuel consumption departure is increased since the destroyer spends most of the time at the lower speed range, as depicted in Figure 27, where its system is less fuel-efficient.

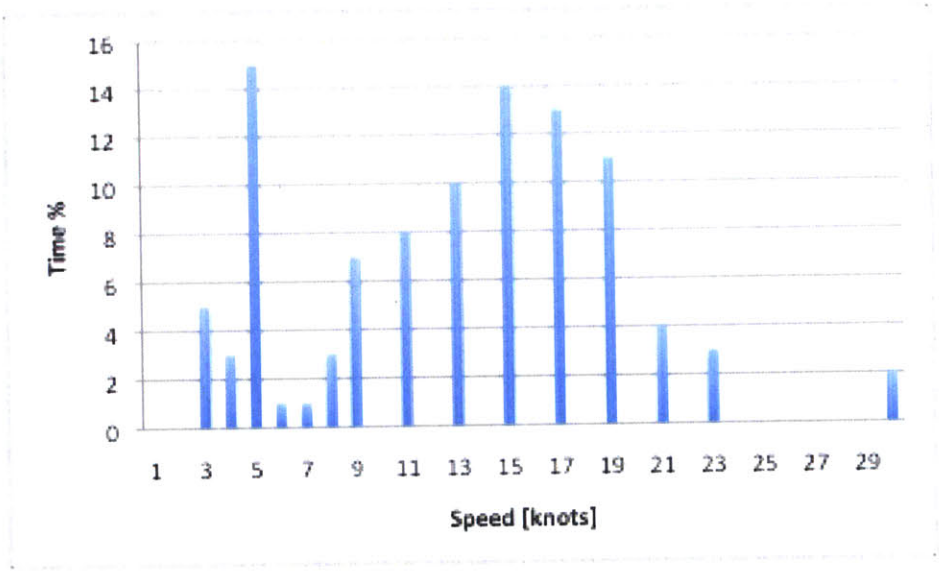


Figure 27 - DDG-51 Flight I destroyer speed operating profile

5.3.2 Optimal Fuel Consumption vs. Current System

Current DDG-51 Flight I propulsion and electric power generation system comprises a mechanical-driven COGAG system with three separate gas turbines for ship-service power generation. Current and optimal fuel consumption systems are depicted in Table 24.

	System	Configuration		Average Fuel Consumption
Actual	Mech-SepSS	Cruise Engines	2xGE LM2500 (GT/19220kW)	3448 kg/hr
		Sprint Engines	2xGE LM2500 (GT/19220kW)	
		Configuration	COGAG	
		Ship-service engines	3xDDA 501-K34G (GT/3362kW)**	
Optimal fuel consumption	Mech-SepSS	Cruise Engines	2xMTU 20V956 (DIESEL/3885kW)	2112 kg/hr
		Sprint Engines	2xGE LM6000PC (GT/37285kW)	
		Configuration	CODOG	
		Ship-service engines	4xMTU 16V396* (DIESEL/1487kW)	

Table 24 - DDG-51 Flight I destroyer actual vs. fuel consumption optimal systems

* Denotes re-rated diesel engine rpm and power.

** Actual DDA 501-K34 is approximated with DDA 501-K34G due to ship-service engine library limitations.

Current DDG-51 Flight I destroyer propulsion and ship service system can be represented for fuel consumption evaluation for mechanical-driven systems with separate ship-service power generation with the following bit string.

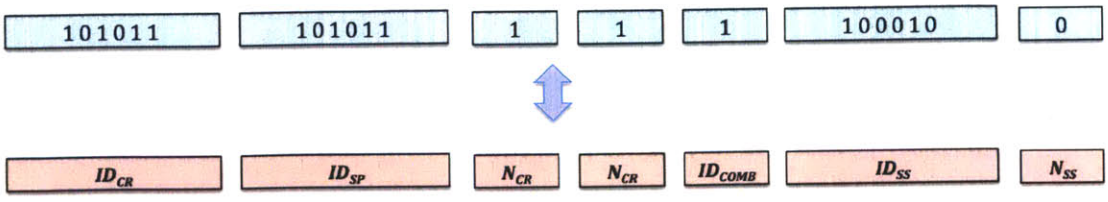


Figure 28 - DDG-51 Flight I destroyer design variable binary encoding

Figures 29 and 30 depict fuel consumption and specific fuel consumption respectively as a function of speed. Specific fuel consumption is calculated for the average ship-service power (~2.4MW).

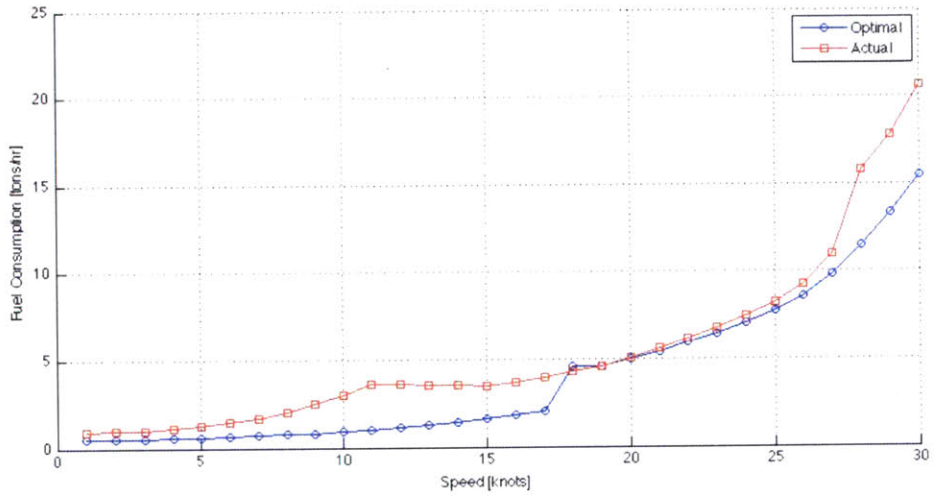


Figure 29 - DDG-51 Flight I current vs. optimal fuel consumption profile

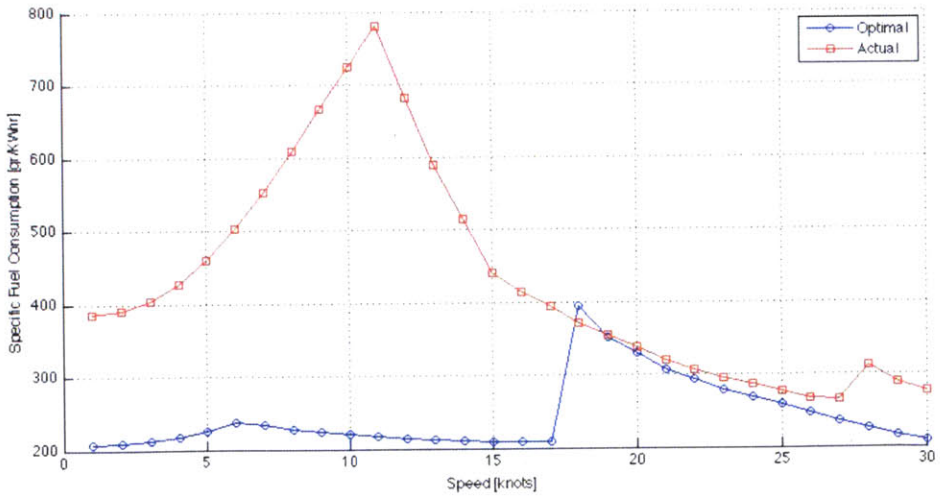


Figure 30 -DDG-51 Flight I current vs. optimal specific fuel consumption profile

As expected the optimal configuration is more fuel-efficient especially at the lower speed range. This is explained because:

- Optimal configuration's cruise engines are diesels, which are the most fuel-efficient at partial load operation required for the lower speed range.

- Optimal configuration's ship-service engines are diesels, which are the most fuel-efficient at partial load operation required for average ship-service loads. Departure is enhanced at the lower speed range, where the accounted average electrical load becomes a significant percentage of total generated power.

All the above contribute to an average fuel consumption difference of 1336 tons/hr for the given combined operating profile.

5.3.3 Sensitivity Analysis

Post-processing sensitivity analysis targets in capturing the effects of changing design variables, design parameters and constraints. As already discussed, this problem involves integer design variables while some design parameters are in vector and matrix formats. Taking into account this apparent difficulty of applying traditional gradient-based techniques, performing a detailed sensitivity analysis for the example of DDG-51 Flight I destroyer was out of the scope of this thesis. Nonetheless, it was interesting to visualize how sensitive the optimal solution was to the given combined operating profile by solving the same DDG-51 Flight I destroyer problem for another operating profile. The new combined profile keeps the same ship-service power profile but has a right-skewed speed profile as depicted in Figure 31. According to new speed profile the ship is expected to spend more time at the higher speed range.

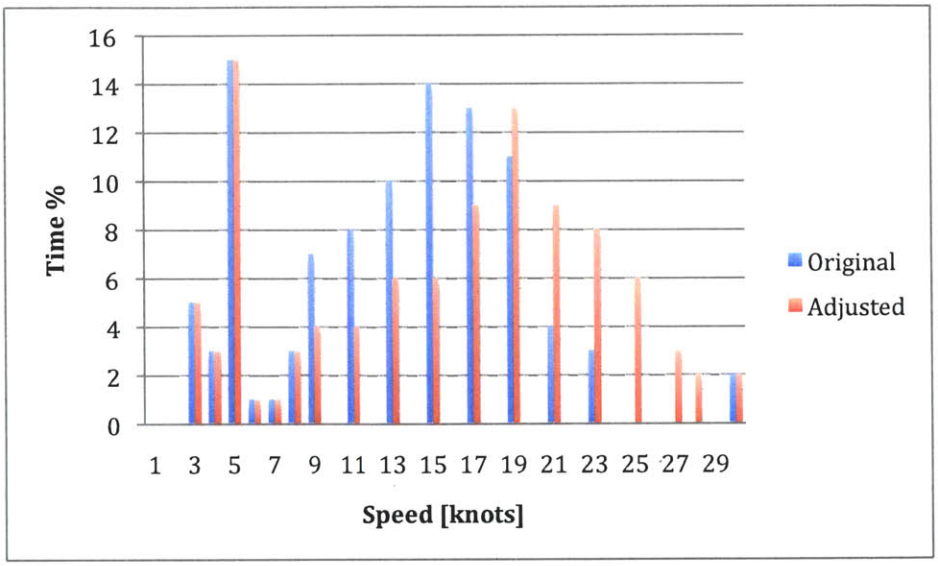


Figure 31 - DDG-51 Flight I destroyer given and adjusted speed operating profiles

Systems selection for optimal fuel consumption for the new speed profile is depicted in Table 25.

System	Configuration		Average Fuel Consumption
Mech-SepSS	Cruise Engines	1xMTU 20V956 (DIESEL/3885kW)	3182 kg/hr
	Sprint Engines	2xGE LM6000PC (GT/37285kW)	
	Configuration	Cross-Connected	
	Ship-service engines	4xMTU 16V396* (DIESEL/1487kW)	
Mech-PDSS	Cruise Engines	1xMTU 20V956 (DIESEL/3885kW)	3191 kg/hr
	Sprint Engines	2xGE LM6000PC (GT/37285kW)	
	Configuration	Cross-Connected	
	Separate ship-service engines	3xMTU 12V538* (DIESEL/1830kW)	
IPS	Primary engines	2xGE LM6000PC (GT/37285kW)	3981 kg/hr
	Secondary engines	2xRRA 601KF9G (GT/6617kW)	

Table 25 - DDG-51 Flight I destroyer optimization results for the new operating profile

Although engine selection remained similar the mechanical-driven systems now employ a cross-connected reduction gear for optimal fuel efficiency. Comparison of average fuel consumption for each system is given is Table 26.

	Original Operating Profile		New Operating Profile	
	Average Fuel Consumption	**Departure from Optimal**	**Average Fuel Consumption**	**Departure from Optimal**
Mech-SepSS	2112	0.0%	3182	0.0%
Mech-PDSS	2145	1.6%	3191	0.3%
IPS	2812	33.1%	3981	25.1%

Table 26 - Average fuel consumption results comparison for DDG-51 Flight I operating profiles

It is easily noticed that the new speed profile increases average fuel consumption but by a different amount for each system. This is illustrated in Figures 32 and 33, which depict fuel consumption and specific fuel consumption respectively as a function of speed. Specific fuel consumption is calculated for the average ship-service power (~2.4MW).

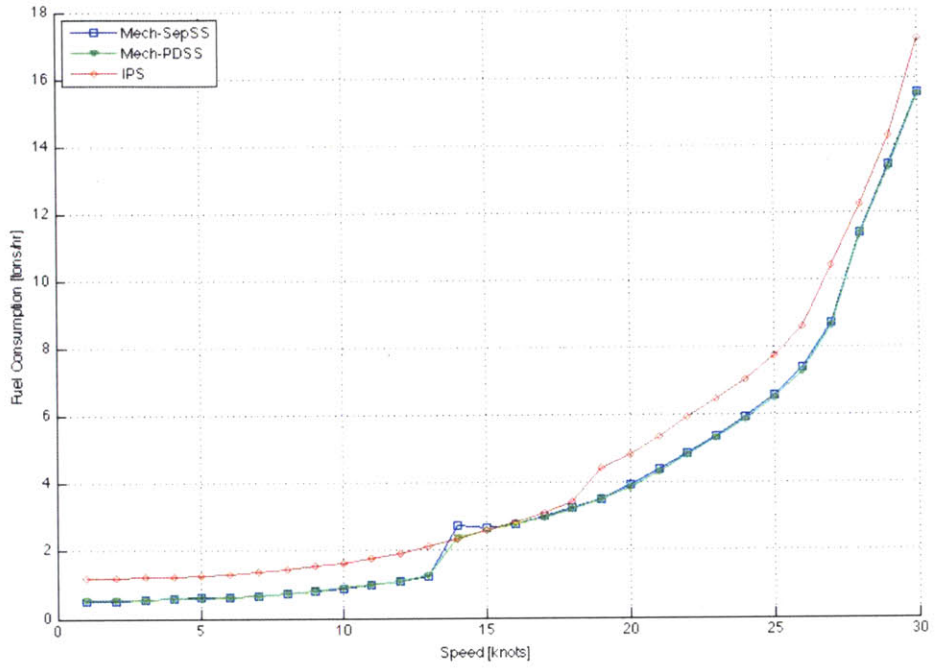


Figure 32 – Optimal fuel consumption profiles for the DDG-51 Flight I destroyer for the new operating profile

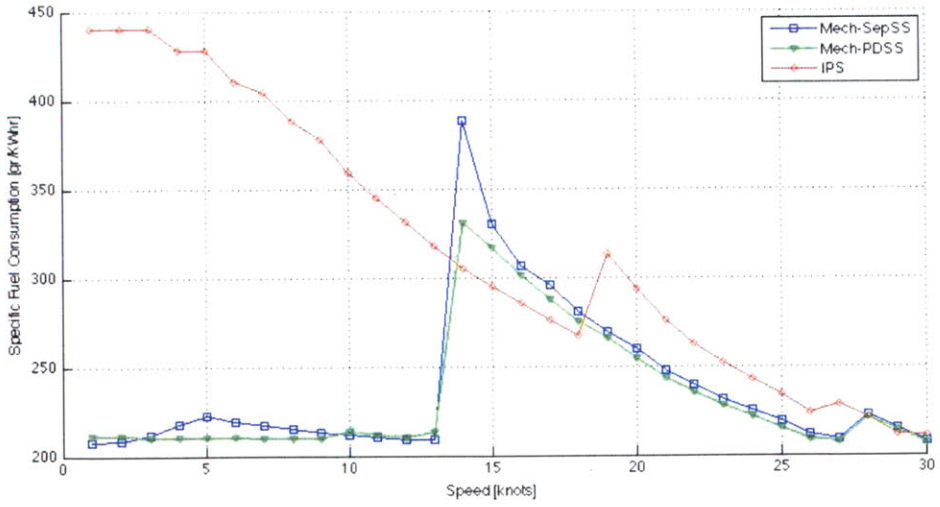


Figure 33 - Optimal fuel consumption profiles for the DDG-51 Flight I destroyer for the new operating profile

CHAPTER 6

CONCLUSIONS

6.1 Conclusions

Fuel efficiency has always been a major issue for the marine industry as has been in most commercial applications due to cost as well as CO_2 emissions. Current economic climate and increasing oil prices have started to drive naval ship design into more fuel-efficient solutions as well. Naval ship design requirements such as redundancy, high top speed and high endurance range have always increased the complexity of the propulsion and ship-service power generation systems architecture. This multi-disciplinary systems selection process was optimized for minimal fuel consumption over the whole ship speed range by managing modeling complexity and employing a heuristic optimization technique.

Resistance and hull efficiency parameters are estimated using the Holtrop and Mennen statistical method, which allows minimal user input. Propulsor modeling's purpose of the program is not to precisely define the geometry of the propeller but to estimate the associated propeller efficiencies so that resistance information can be adequately translated to shaft power (SHP) requirements. Nevertheless, the program's flexibility allows the design team to import specific resistance or shaft power data if those are available to them.

Traditional naval ships speed profiles are not ideal for adequate propulsion derived ship-service power and integrated power systems modeling. Instead, a two-dimension speed and ship-service power operating profile information is required. In case ship-

service power operating profile is not available the approach of a normally distributed ship-service power profile around the average ship-service power is recommended.

Fuel consumption and specific fuel consumption generated plots are very useful in comparing different systems fuel consumption profiles and visualizing where efficiency gains can be achieved. Furthermore, specific fuel consumption plots along with engine operation load fraction plots can be utilized to define optimal efficiency transit speeds.

Systems transmission efficiencies influence fuel consumption calculations since the prime movers have to work at higher load to overcome those losses. If more detailed transmission efficiencies information becomes available the program can trivially be updated to include this information.

As an example, the program identified the optimal fuel consumption system for the DDG-51 Flight I destroyer to be over 38% more efficient than the existing configuration highlighting the potential for fuel efficiency improvements in naval ships.

This program attempts to support management decisions during the naval ship design process by modeling engineering principles and employing heuristic optimization techniques. Using this program to determine fuel-efficient choices is expected to improve the decision-making process during the propulsion and ship-service power generation systems selection.

6.2 Recommendations for Future Work

As with every program, user feedback will indicate shortcomings and areas for improvement. Possible areas of future work identified include increasing modeling fidelity and introducing multiple objectives for optimization.

Future work may include increasing the prime mover libraries entries and especially those for integrated power systems. Caution should be exercised though in maintaining consistency in the engine installation losses accounting. Representing each engine's

specific fuel consumption independently would also improve partial-load engine operation modeling.

Although still not very popular, hybrid propulsion systems exhibit some of the advantages of both mechanical and electrical transmissions and are of increasing interest in modern applications. Modeling those systems would add significant value in systems' fuel consumption comparison studies. Waterjet propulsors may also be considered due to their superior performance in high-speed applications.

Integrated power systems provide the flexibility of allocating the generated power between propulsion and ship-service. As the ship-service loads on board naval ships are constantly increasing it may occur that the integrated power system is designed to compromise speed over ship-service power and vice versa, not having to satisfy both requirements at the same time. It would be interesting to assess how this aspect affects fuel consumption.

At the moment the program indicates the most fuel-efficient configuration for each of the three major systems categories (mechanical-driven propulsion with or without propulsion derived ship-service power and integrated power systems). Although the user can intervene to the prime mover libraries to avoid a specific engine selection, if for example this does not meet weight restrictions, it would be very useful if the program would indicate numerous configurations close to the optimal solution. This would improve scoring-based decision-making methods such as the Pugh matrix.

This thesis dealt with the propulsion and ship-service power generation systems selection as a single-objective problem by optimizing fuel consumption, undoubtedly the most computationally expensive problem for those systems selection. Nonetheless, there are many other objectives that the design team usually considers such as weight and volume restrictions, endurance range, acquisition and maintenance costs. Introducing one or more of those objectives in the optimization process would be extremely beneficial for every design team.

REFERENCES

[1] Bailey, D., "The NPL High Speed Round Bilge Displacement Hull Series", Royal Institute of Naval Architects, *Maritime Technology Monograph No4*, 1976

[2] Carderock Division, Naval Surface Warfare Center, "Advanced Surface Ship Evaluation Tool v.5.3", Help Files

[3] Fung, S.C., "Resistance and Powering Prediction for Transom Stern Hull Forms During Early Stage Ship Design", *Transactions of the Society of Naval Architects and Marine Engineers*, Vol. 99, 1991, pp. 29-73

[4] Gertler, M., "A Reanalysis of the Original Test Data for the Taylor Standard Series", Navy Department, The David W. Taylor Model Basin Washington DC, Report 806, 1954

[5] Gill, P.E, Murray, W. and Wright, M.H., "Practical Optimization", 1986

[6] Gillmer, G.C. and Johnson, B., "Introduction to Naval Architecture", 1982

[7] Holtrop, J. and Mennen, G.G.J., "A Statistical Power Prediction Method", *International Shipbuilding Progress*, Vol. 25, 1977, pp. 253-256

[8] Holtrop, J. and Mennen, G.G.J., "An Approximate Power Prediction Method", *International Shipbuilding Progress*, Vol. 29, No. 335, 1982, pp. 166-170

[9] Holtrop, J., "A Statistical Re-analysis of Resistance and Propulsion Data", *International Shipbuilding Progress*, Vol. 31, No. 363, 1984, pp. 272-276

[10] International Council on Systems Engineering (INCOSE), "Systems Engineering Handbook", 2010

[11] Lahtiharju, E and Karppinen, T., "Resistance and Seakeeping Characteristics of Fast Transom Stern Hulls with Systematically Varied Form", *Transactions of the Society of Naval Architects and Marine Engineers*, Vol. 99, 1991

[12] Loukakis, T.A and Gelegenis, C.J., "A New Form of Optimization Diagrams for Preliminary Propeller Design", *Transactions of the Royal Institute of Naval Architects*, 1988

[13] "Marine Diesel Power Plant Performance Practices", SNAME T&R Bulletin No. 3-27, 1975

[14] Moody, R.D., "Preliminary Power Prediction During Early Design Stages of a Ship", *Cape Technikon Theses & Dissertations*, Paper 175, 1996

[15] NAVSEA Design Data Sheet DDS 051-1, 1984

[16] NAVSEA, "Speed-Time Profile Guide for Surface Ships", FY 2002

[17] Oosterveld, M.W.C. and Van Oossanen, P., "Further Computer-Analyzed Data of the Wageningen B-Screw Series", *International Shipbuilding Progress*, Vol. 22, No. 251, 1975

[18] Oosterveld, M.W.C. and Van Oossanen, P., "Representation of Propeller Characteristics Suitable for Preliminary Ship Design Studies", *International Conference on Computer Applications in Shipbuilding*, 1973

[19] Parsons, M.G., "Parametric Design" in Ship Design and Construction Vol. I, Chapter 11, SNAME, 2004

[20] Surko, S. and Osborne, M., "Operating Speed Profiles and the Ship Design Cycle", *Naval Engineers Journal*, Vol. 117, Issue 3, 2005, pp.79-85

[21] Ugray, Zsolt, Lasdon L., Plummer J. C., Glover F., Kelly J., and Martí R., "Scatter Search and Local NLP Solvers: A Multistart Framework for Global Optimization" *INFORMS Journal on Computing*, Vol. 19, No. 3, 2007, pp. 328–340.

[22] Van Mannen, J.D. and Van Oossanen, P., "Resistance" and "Propulsion" in Principles of Naval Architecture Vol. II, SNAME, 1988

[23] Woud, H.K. and Stapersma, D., "Design of Propulsion and Electric Power Generation Systems", 2002

[24] Yeh, H., "Series 64 Resistance Experiments on High-Speed Displacement Platforms", Marine Technology, 2, 1965, pp. 248-272

APPENDIX

A

ENGINE LIBRARIES

	Manufacturer	Model	Type	rpm	Power	SFC	Weight
					kW	$\frac{gr}{KWh}$	$tons$
1	Caterpillar	CAT D399V16	DIESEL	1225	839	232.2	7.82
2	Caterpillar	CAT 3408B-TA	DIESEL	1800	317	216.5	1.74
3	Caterpillar	CAT 3512V12	DIESEL	1800	876	215.9	6.46
4	Caterpillar	CAT 3516V16	DIESEL	1800	1268	214.7	8.15
5	Cummins	C KTA 2300G	DIESEL	1800	701	228.1	4.05
6	Cummins	C KTA 38-GC1	DIESEL	1800	723	214.1	7.41
7	Detroit Diesel	DD 12V71RC	DIESEL	1800	343	228.7	3.18
8	Detroit Diesel	DD 16V71T	DIESEL	1800	350	231.1	3.20
9	Detroit Diesel	DD 12V71	DIESEL	1800	261	228.7	2.40
10	Detroit Diesel	DD 16V92T	DIESEL	1800	537	220.8	2.65
11	Detroit Diesel	DD 12V149TI	DIESEL	1800	701	237.2	6.33
12	Detroit Diesel	DD 16V149TI	DIESEL	1800	1053	252.4	13.79
13	Isotta Fraschini	IF ID36SS6V-AM	DIESEL	1800	447	231.1	1.89
14	Isotta Fraschini	IF ID36SS3V-AM	DIESEL	1800	597	223.2	3.02
15	MTU	MTU 6V331	DIESEL	2180	485	206.8	1.85
16	MTU	MTU 6V396	DIESEL	1845	582	206.8	2.06

	Manufacturer	Model	Type	rpm	Power	SFC	Weight
					kW	$\frac{gr}{KWh}$	$tons$
17	MTU	MTU 8V331	DIESEL	2250	671	206.8	2.31
18	MTU	MTU 8V396	DIESEL	1845	783	206.8	2.57
19	MTU	MTU 12V331	DIESEL	2180	969	206.8	3.21
20	MTU	MTU 12V396	DIESEL	1845	1163	206.8	3.57
21	MTU	MTU 16V396	DIESEL	1845	1562	206.8	4.70
22	MTU	MTU 12V538	DIESEL	1710	1652	209.9	5.15
23	MTU	MTU 16V538	DIESEL	1710	2215	209.9	6.70
24	MTU	MTU 20V538	DIESEL	1710	2770	209.9	9.00
25	MTU	MTU 16V956	DIESEL	1455	3110	209.9	13.65
26	MTU	MTU 16V1163	DIESEL	1160	3281	209.9	14.45
27	MTU	MTU 20V956	DIESEL	1455	3885	209.9	16.22
28	MTU	MTU 20V1163	DIESEL	1160	4101	209.9	16.96
29	Paxman Valenta	PVAL BPCI16V	DIESEL	1460	1778	212.9	2.40
30	Paxman Valenta	PVAL DPCI16V	DIESEL	1500	2013	212.9	9.77
31	Paxman Valenta	PVAL CPCI16V	DIESEL	1400	2189	212.9	8.96
32	Paxman Valenta	PVAL RP200M16V	DIESEL	1500	2498	214.1	9.31
33	Paxman Vega	PVEGA CPCI16V	DIESEL	1250	1193	212.9	5.18
34	Paxman Vega	PVEGA DPCI16V	DIESEL	1800	1305	212.9	4.96
35	Paxman Vega	PVEGA BPCI16V	DIESEL	1600	1357	212.9	1.14
36	Paxman Vega	PVEGA APCI16V	DIESEL	1800	1715	212.9	1.16
37	Detroit Diesel Allison	DDA 501-K17	GT	13820	2834	331.7	0.58
38	Detroit Diesel	DDA 501-K34	GT	14300	3430	287.5	0.58

	Manufacturer	Model	Type	rpm	Power	SFC	Weight
					kW	$\frac{gr}{KWh}$	$tons$
	Allison						
39	Detroit Diesel Allison	DDA 570-KA	GT	11500	4448	289.7	0.61
40	Detroit Diesel Allison	DDA 571-K	GT	11500	4746	276.5	0.68
41	Detroit Diesel Allison	DDA 501-K34G	GT	1800	3362	293.4	3.22
42	General Electric	GE LM500	GT	7000	3356	292.7	0.58
43	General Electric	GE LM500G	GT	1800	3940	284.7	2.29
44	General Electric	GE LM2500	GT	3600	19220	249.2	3.20
45	General Electric	GE LM2500-1A	GT	3600	21681	234.0	3.20
46	General Electric	GE LM2500-PLUS	GT	3600	26100	226.1	3.39
47	General Electric	GE LM2500-20	GT	3600	12304	259.6	3.20
48	General Electric	GE LM2500-21	GT	3600	16033	249.2	3.20
49	General Electric	GE LM2500-30	GT	3600	19575	239.0	3.20
50	General Electric	GE LM5000	GT	3600	29157	235.3	4.88
51	General Electric	GE LM6000PC	GT	3600	37285	206.8	5.59
52	Solar	SOLAR MERC 50G	GT	1800	3846	217.2	7.35
53	Solar	SOLAR MERC 50	GT	14186	3925	212.9	6.51
54	Rolls Royce/DDA	RR DDA-SPEY	GT	4800	9873	258.1	2.52
55	Rolls Royce	RRA T406G	GT	1800	3654	269.6	2.24
56	Rolls Royce	RRA 601KF9G	GT	1800	6617	262.5	4.18
57	Rolls Royce	RR MT30	GT	3600	36000	213.2	6.20

Table 27 - Program Propulsion Engines Library

	Manufacturer	Model	Type	rpm	Power	SFC	Weight
					kW	$\frac{gr}{KWh}$	$tons$
1	Caterpillar	CAT 3406C	DIESEL	1800	272	212.3	1.32
2	Caterpillar	CAT 3408C	DIESEL	1800	339	220.2	1.66
3	Caterpillar	CAT 3412C	DIESEL	1800	485	225.1	2.32
4	Caterpillar	CAT 3508	DIESEL	1800	638	222.0	5.21
5	Caterpillar	CAT 3508B	DIESEL	1800	746	212.3	4.88
6	Caterpillar	CAT 3512B	DIESEL	1800	1119	213.5	5.89
7	Caterpillar	CAT 3516B	DIESEL	1800	1491	226.3	7.79
8	Caterpillar	CAT 3408B-TA	DIESEL	1800	317	216.5	1.74
9	Caterpillar	CAT 3512V12	DIESEL	1800	876	215.9	6.46
10	Caterpillar	CAT 3516V16	DIESEL	1800	1268	214.7	8.15
11	Caterpillar	CAT C18 #E	DIESEL	1800	567	217.2	1.59
12	Cummins	C KTA 38-M2	DIESEL	1800	895	210.5	4.22
13	Cummins	C KTA 50-M2	DIESEL	1800	1193	207.4	4.86
14	Cummins	C KTA 2300G	DIESEL	1800	701	228.1	4.05
15	Cummins	C KTA 38-GC1	DIESEL	1800	723	214.1	7.41
16	Detroit Diesel	DD 12V71RC	DIESEL	1800	343	228.7	3.18
17	Detroit Diesel	DD 16V71T	DIESEL	1800	350	231.1	3.20
18	Detroit Diesel	DD 12V71	DIESEL	1800	261	228.7	2.40
19	Detroit Diesel	DD 16V92T	DIESEL	1800	537	220.8	2.65
20	Detroit Diesel	DD 12V149TI	DIESEL	1800	701	237.2	6.33
21	Detroit Diesel	DD 16V149TI	DIESEL	1800	1053	252.4	13.79
22	Isotta Fraschini	IF ID36SS6V-AM	DIESEL	1800	447	231.1	1.89

	Manufacturer	Model	Type	rpm	Power	SFC	Weight
					kW	$\frac{gr}{KWh}$	$tons$
23	Isotta Fraschini	IF ID36SS3V-AM	DIESEL	1800	597	223.2	3.02
24	MTU	MTU 6V396	DIESEL	1800*	554	206.8	2.06
25	MTU	MTU 8V396	DIESEL	1800*	745	206.8	2.57
26	MTU	MTU 12V396	DIESEL	1800*	1107	206.8	3.57
27	MTU	MTU 16V396	DIESEL	1800*	1487	206.8	4.70
28	MTU	MTU 12V538	DIESEL	1800*	1830	209.9	5.15
29	MTU	MTU 16V538	DIESEL	1800*	2454	209.9	6.70
30	MTU	MTU 20V538	DIESEL	1800*	3070	209.9	9.00
31	MTU	MTU 16V1163	DIESEL	1200*	3511	209.9	14.45
32	MTU	MTU 20V1163	DIESEL	1200*	4389	209.9	16.96
33	Paxman Vega	PVEGA DPCI16V	DIESEL	1800	1305	212.9	4.96
34	Paxman Vega	PVEGA APCI16V	DIESEL	1800	1715	212.9	1.16
35	Detroit Diesel Allison	DDA 501-K34G	GT	1800	3362	293.4	3.22
36	General Electric	GE LM500G	GT	1800	3940	284.7	2.29
37	General Electric	GE LM1600G #E	GT	3600	10089	265.2	4.23
38	General Electric	GE LM2500	GT	3600	19220	249.2	3.20
39	General Electric	GE LM2500-1A	GT	3600	21681	234.0	3.20
40	General Electric	GE LM2500-PLUS	GT	3600	26100	226.1	3.39
41	General Electric	GE LM2500+ #E	GT	3600	24608	228.7	2.36
42	General Electric	GE LM2500-20	GT	3600	12304	259.6	3.20
43	General Electric	GE LM2500-21	GT	3600	16033	249.2	3.20
44	General Electric	GE LM2500-30	GT	3600	19575	239.0	3.20

	Manufacturer	Model	Type	rpm	Power	SFC	Weight
					kW	$\frac{gr}{KWh}$	$tons$
45	General Electric	GE LM5000	GT	3600	29157	235.3	4.88
46	General Electric	GE LM6000PC	GT	3600	37285	206.8	5.59
47	General Electric	GE LM6000PD #E	GT	3600	48471	214.1	1.05
48	Solar	SOLAR MERC 50G	GT	1800	3846	217.2	7.35
49	Rolls Royce/DDA	RR DDA-SPEY	GT	4800	9873	258.1	2.52
50	Rolls Royce	RRA T406G	GT	1800	3654	269.6	2.24
51	Rolls Royce	RRA 601KF9G	GT	1800	6617	262.5	4.18
52	Rolls Royce	RR MT30	GT	3600	36000	213.2	6.20

Table 28 - Model ship-service engines library

* denotes re-rated engine rpm and power.

	Manufacturer	Model	Type	rpm	Power	SFC	Weight
					kW	$\frac{gr}{KWh}$	$tons$
1	General Electric	GE LM1600G #E	GT	3600	10089	265.2	4.23
2	General Electric	GE LM2500-1A	GT	3600	21681	234.0	3.20
3	General Electric	GE LM2500+ #E	GT	3600	24608	228.7	2.36
4	General Electric	GE LM6000PC	GT	3600	37285	206.8	5.59
5	General Electric	GE LM6000PD #E	GT	3600	48471	214.1	1.05
6	Rolls Royce	RR MT30	GT	3600	36000	213.2	6.20
7	Detroit Diesel Allison	DDA 501-K34G	GT	1800	3362	293.4	3.22
8	General Electric	GE LM500G	GT	1800	3940	284.7	2.29
9	Solar	SOLAR MERC 50G	GT	1800	3846	217.2	7.35
10	Solar	SOLAR MERC 50	GT	14186	3925	212.9	6.51
11	Rolls Royce	RRA T406G	GT	1800	3654	269.6	2.24
12	Rolls Royce	RRA 601KF9G	GT	1800	6617	262.5	4.18
13	Caterpillar	CAT C18 #E	DIESEL	1800	567	217.2	1.59

Table 29 - Model IPS engines library

APPENDIX

B

USER GUIDE

B1. Import User files

Program is set up to import a series of files. Although these files have to be in windows formatted text (.txt) format the user is encouraged to do prior editing in Microsoft Excel® for convenience.

B1.1 Combined Operating Profile

This profile expresses how much time the ship spends at each speed and each ship-service power segment. Rows correspond to the same speed expressed in knots. Columns correspond to the same ship-service power segment express in average kW. After importing, it is used to retrieve maximum speed and ship-service power requirements.

Number of rows and number of columns can change to accommodate specific user needs.

Table 30 depicts the annotated combined profile used for the DDG-51 Flight I destroyer. Table 31 depicts the excel version of this profile. The ship service power is normally distributed around the mean (2377kW) using the Excel® formula:

$$H2 = \$P2*(\$D\$1-\$C\$1)*NORMDIST(H\$1,2377,408,FALSE) \quad (B.1)$$

		SHIP-SERVICE POWER [MW]													% TIME
		MIN-1.3	1.3-1.5	1.5-1.7	1.7-2.1	1.9-2.1	2.1-2.3	2.3-2.5	2.5-2.7	2.7-2.9	2.9-3.1	3.1-3.3	3.3-3.5	3.5-MAX	
SPEED [knots]	0-2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	3	0.0	0.1	0.2	0.4	0.6	0.9	1.0	0.8	0.6	0.3	0.1	0.0	0.0	5.0
	4	0.0	0.0	0.1	0.2	0.4	0.5	0.6	0.5	0.3	0.2	0.1	0.0	0.0	3.0
	5	0.0	0.2	0.5	1.1	1.9	2.7	2.9	2.5	1.7	0.9	0.4	0.1	0.0	15.0
	6	0.0	0.0	0.0	0.1	0.1	0.2	0.2	0.2	0.1	0.1	0.0	0.0	0.0	1.0
	7	0.0	0.0	0.0	0.1	0.1	0.2	0.2	0.2	0.1	0.1	0.0	0.0	0.0	1.0
	8	0.0	0.0	0.1	0.2	0.4	0.5	0.6	0.5	0.3	0.2	0.1	0.0	0.0	3.0
	9	0.0	0.1	0.2	0.5	0.9	1.2	1.4	1.2	0.8	0.4	0.2	0.1	0.0	7.0
	10	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	11	0.0	0.1	0.3	0.6	1.0	1.4	1.6	1.3	0.9	0.5	0.2	0.1	0.0	8.0
	12	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	13	0.0	0.1	0.3	0.7	1.3	1.8	2.0	1.7	1.1	0.6	0.3	0.1	0.0	10.0
	14	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	15	0.0	0.2	0.4	1.0	1.8	2.5	2.7	2.4	1.6	0.9	0.4	0.1	0.0	14.0
	16	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	17	0.0	0.1	0.4	0.9	1.7	2.3	2.5	2.2	1.5	0.8	0.3	0.1	0.0	13.0
	18	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	19	0.0	0.1	0.4	0.8	1.4	2.0	2.1	1.9	1.3	0.7	0.3	0.1	0.0	11.0
	20	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	21	0.0	0.0	0.1	0.3	0.5	0.7	0.8	0.7	0.5	0.2	0.1	0.0	0.0	4.0
	22	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	23	0.0	0.0	0.1	0.2	0.4	0.5	0.6	0.5	0.3	0.2	0.1	0.0	0.0	3.0
	24	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	25	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	26	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	27	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	28	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	29	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	30	0.0	0.0	0.1	0.1	0.3	0.4	0.4	0.3	0.2	0.1	0.1	0.0	0.0	2.0
% TIME		0.3	1.0	2.9	6.6	11.7	16.4	18.0	15.5	10.5	5.6	2.4	0.8	0.2	100.0

Table 30 - DDG-51 Flight I combined operating profile

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1		**1200**	**1400**	**1600**	**1800**	**2000**	**2200**	**2400**	**2600**	**2800**	**3000**	**3200**	**3400**	**3600**		
2	**1**	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	**2**	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	**3**	0.0	0.1	0.2	0.4	0.6	0.9	1.0	0.8	0.6	0.3	0.1	0.0	0.0	5.0	5.0
5	**4**	0.0	0.0	0.1	0.2	0.4	0.5	0.6	0.5	0.3	0.2	0.1	0.0	0.0	3.0	3.0
6	**5**	0.0	0.2	0.5	1.1	1.9	2.7	2.9	2.5	1.7	0.9	0.4	0.1	0.0	15.0	15.0
7	**6**	0.0	0.0	0.0	0.1	0.1	0.2	0.2	0.2	0.1	0.1	0.0	0.0	0.0	1.0	1.0
8	**7**	0.0	0.0	0.0	0.1	0.1	0.2	0.2	0.2	0.1	0.1	0.0	0.0	0.0	1.0	1.0
9	**8**	0.0	0.0	0.1	0.2	0.4	0.5	0.6	0.5	0.3	0.2	0.1	0.0	0.0	3.0	3.0
10	**9**	0.0	0.1	0.2	0.5	0.9	1.2	1.4	1.2	0.8	0.4	0.2	0.1	0.0	7.0	7.0
11	**10**	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
12	**11**	0.0	0.1	0.3	0.6	1.0	1.4	1.6	1.3	0.9	0.5	0.2	0.1	0.0	8.0	8.0
13	**12**	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
14	**13**	0.0	0.1	0.3	0.7	1.3	1.8	2.0	1.7	1.1	0.6	0.3	0.1	0.0	10.0	10.0
15	**14**	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
16	**15**	0.0	0.2	0.4	1.0	1.8	2.5	2.7	2.4	1.6	0.9	0.4	0.1	0.0	14.0	14.0
17	**16**	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
18	**17**	0.0	0.1	0.4	0.9	1.7	2.3	2.5	2.2	1.5	0.8	0.3	0.1	0.0	13.0	13.0
19	**18**	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
20	**19**	0.0	0.1	0.4	0.8	1.4	2.0	2.1	1.9	1.3	0.7	0.3	0.1	0.0	11.0	11.0
21	**20**	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
22	**21**	0.0	0.0	0.1	0.3	0.5	0.7	0.8	0.7	0.5	0.2	0.1	0.0	0.0	4.0	4.0
23	**22**	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
24	**23**	0.0	0.0	0.1	0.2	0.4	0.5	0.6	0.5	0.3	0.2	0.1	0.0	0.0	3.0	3.0
25	**24**	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
26	**25**	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
27	**26**	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
28	**27**	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
29	**28**	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
30	**29**	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
31	**30**	0.0	0.0	0.1	0.1	0.3	0.4	0.4	0.3	0.2	0.1	0.1	0.0	0.0	2.0	2.0
32		0.3	1.0	2.9	6.6	11.7	16.4	18.0	15.5	10.5	5.6	2.4	0.8	0.2	100.0	100.0

Table 31 - Excel® file for importing combined operating profile (DDG-51 example)

B.1.2 Resistance

If resistance information is known user can import this information to the program using the associated text file. Table 32 depicts resistance data in Excel® format for the DDG-51 Flight I destroyer. Program is set up to interpolate/extrapolate data to fill missing information. Speed is in knots and resistance in kN.

	A	B
1	speed	R_import
2	0	0
3	1	
4	2	
5	3	
6	4	
7	5	58.78
8	6	81.528
9	7	107.603
10	8	136.932
11	9	169.457
12	10	205.128
13	11	244.479
14	12	290.004
15	13	339.643
16	14	394.317
17	15	453.786
18	16	508.563
19	17	567.943
20	18	635.269
21	19	704.494
22	20	796.974
23	21	917.679
24	22	1030.169
25	23	1139.45
26	24	1256.071
27	25	1396.454
28	26	1574.162
29	27	1821.969
30	28	2147.16
31	29	2520.554
32	30	2918.698
33	31	3318.8
34	32	3670.584
35	33	3991.137
36	34	
37	35	

Table 32 - Excel® file for importing resistance data (DDG-51 example)

B.1.3 Shaft Power

If shaft power information is known user can import this information to the program using the associated text file. Table 33 depicts shaft power data in Excel® format for the DDG-51 Flight I destroyer. Program is set up to interpolate/extrapolate data to fill missing information. Speed is in knots and shaft power in kW.

	A	B
1	speed	R_import
2	0	0
3	1	
4	2	
5	3	
6	4	
7	5	218
8	6	362
9	7	555
10	8	805
11	9	1118
12	10	1500
13	11	1963
14	12	2539
15	13	3221
16	14	4027
17	15	4967
18	16	5928
19	17	7025
20	18	8381
21	19	9732
22	20	11615
23	21	14112
24	22	16640
25	23	19269
26	24	22197
27	25	25783
28	26	30380
29	27	36849
30	28	45600
31	29	56172
32	30	68109
33	31	80837
34	32	92859
35	33	104512
36	34	
37	35	

Table 33 - Excel® file for importing shaft power data (DDG-51 example)

B.2 Program User Input

First user selects if shaft power information is available (Figure 34).

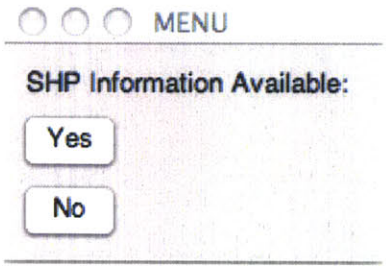


Figure 34 - Matlab® GUI (Shaft power information availability)

If it is not the program will ask if resistance information is available (Figure 35).

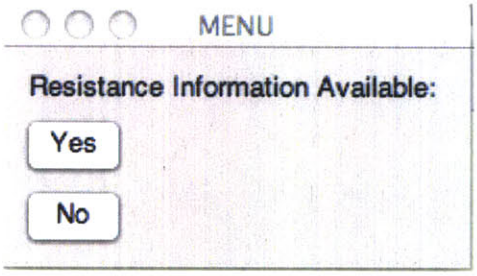


Figure 35 - Matlab® GUI (Resistance information availability)

Depending on those selections there are the following options:

B.2.1 Shaft Power Information Available

Program will ask user to input the power service margin (Figure 36).

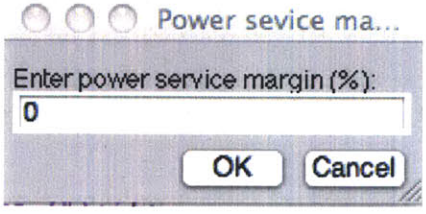


Figure 36 - Matlab® GUI (Power service margin input)

B.2.2 Shaft Power Information not Available but Resistance Information Available

Program will ask some hull related information required for the propeller efficiencies estimation (Figure 37).

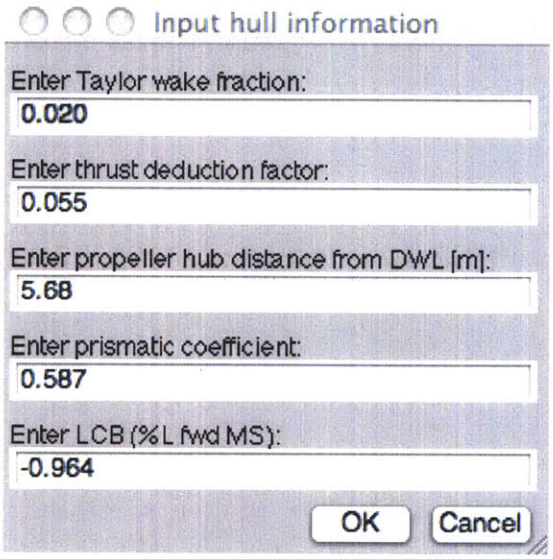


Figure 37 - Matlab® GUI (Hull information input)

Subsequently the program will ask for propeller limits (Figure 38).

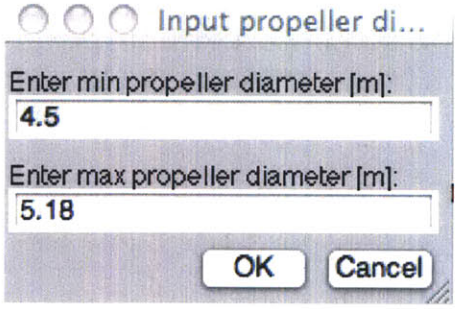


Figure 38 - Matlab® GUI (Propeller limits input)

After propeller selection is over user will be asked to enter the power service margin (Figure 36).

B.2.3 No Shaft Power or Resistance Information Available

Program will ask for hull geometry information to enable resistance calculations (Figure 39).

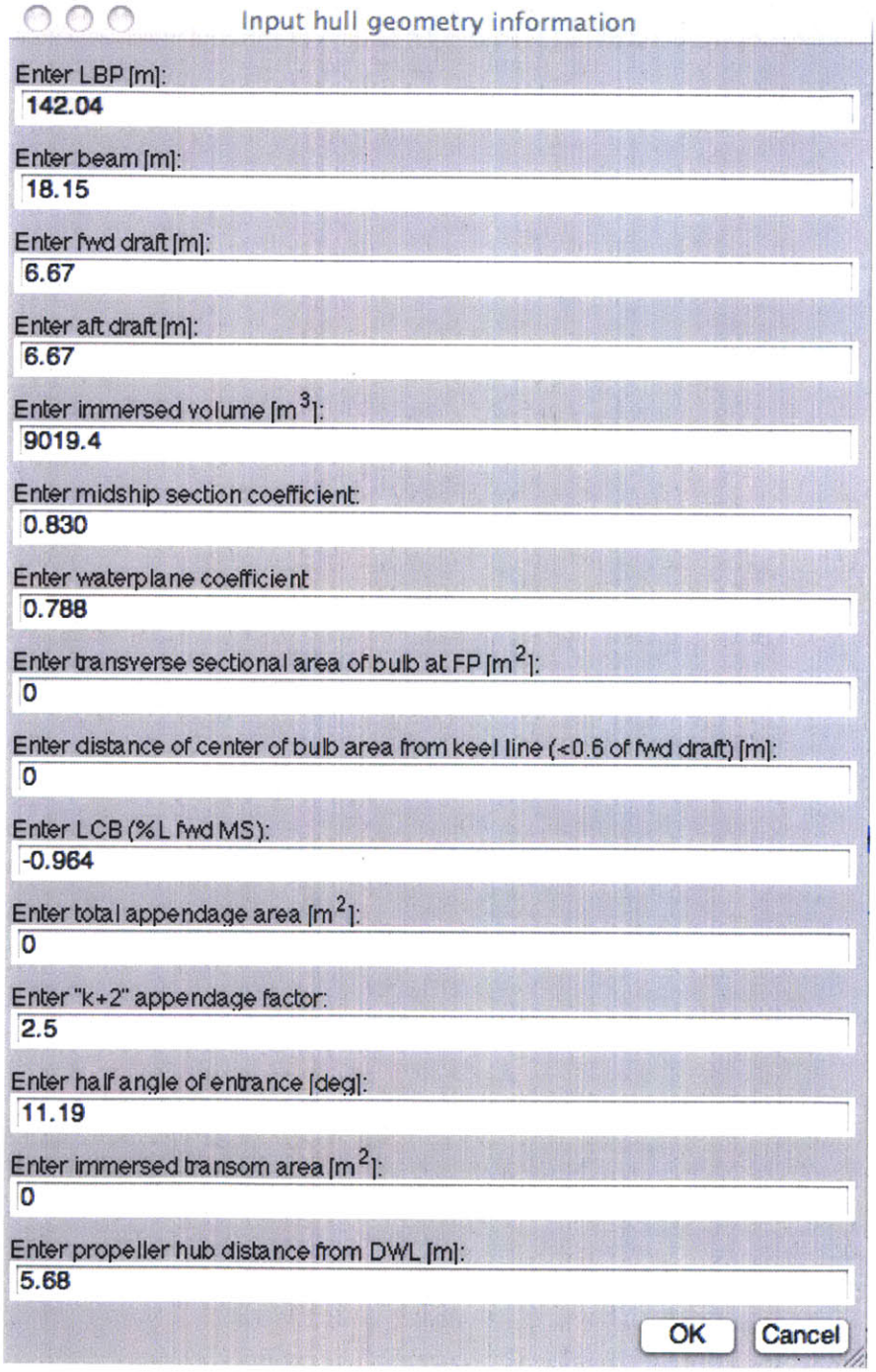


Figure 39 - Matlab® GUI (Hull geometry information input)

Subsequently program will prompt the user to select a correlation allowance estimation method (Figure 40).

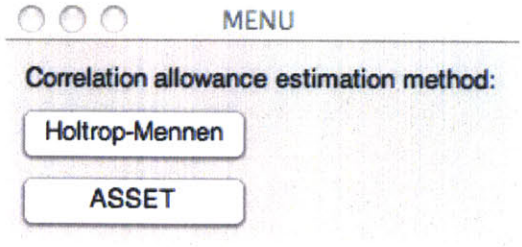


Figure 40 - Matlab® GUI (Correlation allowance method selection)

If program calculates the appendage resistance to be zero it will ask the user whether he wants to enter an appendage resistance estimation as a percentage of bare hull resistance (Figure 41).

Figure 41 - Matlab® GUI (Appendage resistance percentage dialogue)

If user selects to do an input window will ask for this information (Figure 42). Guidance in selecting such percentages is given in PNA Vol. II (Van Mannen and Van Oossanen, 1988).

Figure 42 - Matlab® GUI (Appendage resistance percentage input)

After resistance calculations are over the program will move to the propeller selection and ask the user for propeller limits (Figure 38).

After propeller selection is over user will be asked to enter the power service margin (Figure 36).

B.2.4 Optimization Menu

The optimization menu enables the user to carry single optimization runs for each system category or multiple optimization runs for all categories. While the first three options will give relative quick results there is a possibility they miss the global optimum. The latter option is an automated way of carrying out multiple runs and evaluates the optimal configuration from each system category. It is the recommended option for a global optimum search.

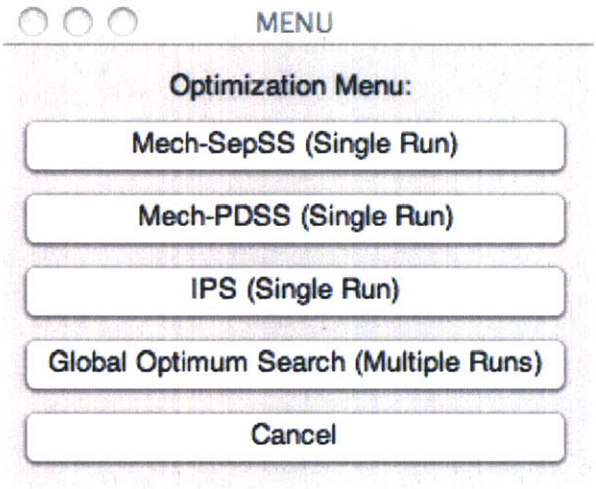


Figure 43 - Matlab® GUI (Optimization menu)

B.3 Program Output

B.3.2 Printed Reports

Printed output in the Matlab® Command Window includes resistance results (), propeller selection results () and optimization results ().

RESISTANCE RESULTS:

Speed	Rw[kN]	Rapp[kN]	Rtr[kN]	Rb[kN]	Rtotal[kN]
0	0.0	0.0	0.0	0.0	0.0
1	0.0	0.2	0.0	0.0	1.4
2	0.0	0.7	0.0	0.0	5.2
3	0.0	1.4	0.0	0.0	11.1
4	0.0	2.5	0.0	0.0	19.0
5	0.0	3.8	0.0	0.0	29.0
6	0.0	5.3	0.0	0.0	40.9
7	0.0	7.1	0.0	0.0	54.8
8	0.1	9.2	0.0	0.0	70.6
9	0.3	11.5	0.0	0.0	88.5
10	0.8	14.2	0.0	0.0	108.7
11	2.2	17.2	0.0	0.0	131.5
12	4.7	20.6	0.0	0.0	157.6
13	9.2	24.5	0.0	0.0	187.7
14	16.3	29.1	0.0	0.0	222.7
15	27.1	34.4	0.0	0.0	263.6
16	42.1	40.6	0.0	0.0	311.2
17	61.7	47.7	0.0	0.0	365.9
18	88.6	56.2	0.0	0.0	430.7
19	126.3	66.5	0.0	0.0	509.7
20	171.6	78.1	0.0	0.0	599.1
21	215.3	89.8	0.0	0.0	688.4
22	252.8	100.8	0.0	0.0	772.4
23	289.9	111.9	0.0	0.0	857.6
24	337.4	124.8	0.0	0.0	956.4
25	406.3	141.1	0.0	0.0	1081.6
26	506.3	162.3	0.0	0.0	1244.3
27	645.4	189.6	0.0	0.0	1453.6
28	828.5	223.7	0.0	0.0	1715.2
29	1056.0	264.7	0.0	0.0	2029.6
30	1325.6	312.3	0.0	0.0	2633.4

Figure 44 - Matlab® resistance results (DDG-51 Flight I example)

SELECTED PROPELLER CHARACTERISTICS:

D: 5.18m
P/D: 1.27
RPM: 189.2
EAR: 1.032
Open Water Efficiency: 0.674
Taylor wake fraction: 0.020
Thrust deduction factor: 0.055
Relative rotative efficiency: 0.961

Figure 45 - Matlab® propeller results (DDG-51 Flight I example)

```
MECHANICAL-DRIVEN PROPULSION WITH SEPARATE SHIP-SERVICE POWER
                    OPTIMAL CONFIGURATION

Propulsion system: CODOG

Cruise engines:        2 MTU 20V956 DIESEL (MTU) / 3885kW
Sprint engines:        2 GE LM6000PC GT (General Electric) / 37285kW
Ship service engines: 4 MTU 16V396* DIESEL (MTU) / 1487kW


Average fuel consumption for given combined profile:  2112 kg/hr
```

Figure 46 - Matlab® optimal configuration results (DDG-51 Flight I – Mech-SepSS optimization example)

```
MECHANICAL-DRIVEN PROPULSION WITH PROPULSION-DERIVED SHIP-SERVICE POWER
                    OPTIMAL CONFIGURATION

Propulsion system: CODOG

Cruise engines:        2 MTU 20V956 DIESEL (MTU) / 3885kW
Sprint engines:        2 GE LM6000PC GT (General Electric) / 37285kW
Ship service engines: 3 MTU 12V538* DIESEL (MTU) / 1830kW

Average fuel consumption for given combined profile:  2145 kg/hr
```

Figure 47 - Matlab® optimal configuration results (DDG-51 Flight I – Mech-PDSS optimization example)

```
                INTEGRATED POWER SYSTEM (IPS)
                    OPTIMAL CONFIGURATION

Primary engines  :        2 GE LM6000PC GT (General Electric) / 37285kW
Secondary engines:        2 RRA 601KF9G GT-G (Rolls Royce) / 6617kW

Average fuel consumption for given combined profile: 2812.1 kg/hr
```

Figure 48 - Matlab® optimal configuration results (DDG-51 Flight I – IPS optimization example)

B.3.3 Graphic Reports

Graphic output includes resistance results and shaft power results.

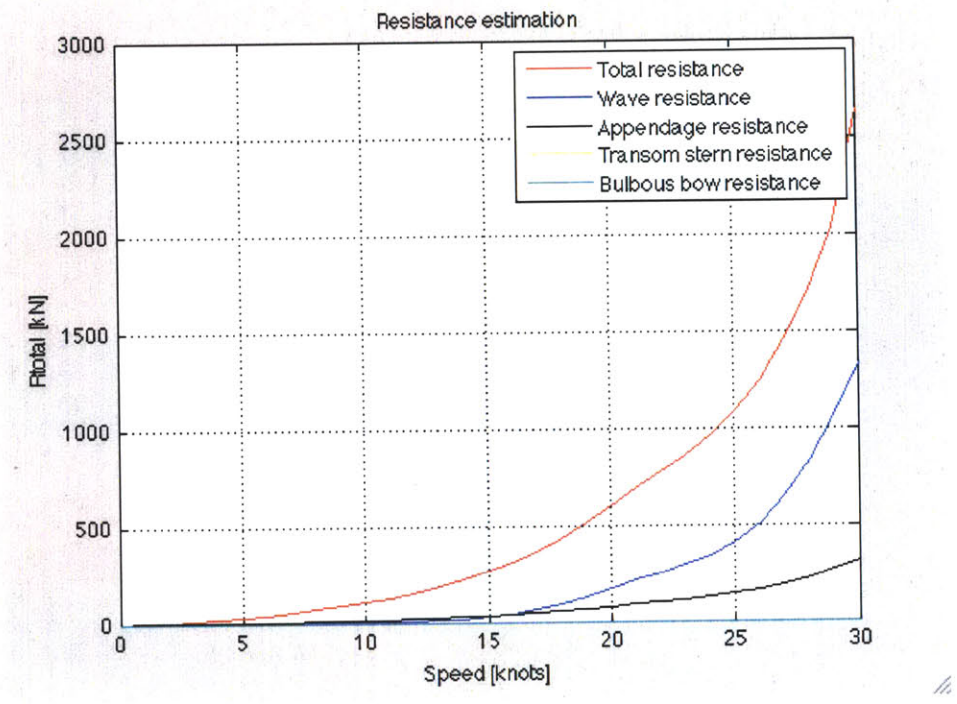


Figure 49 - Matlab® resistance plot (DDG-51 Flight I example)

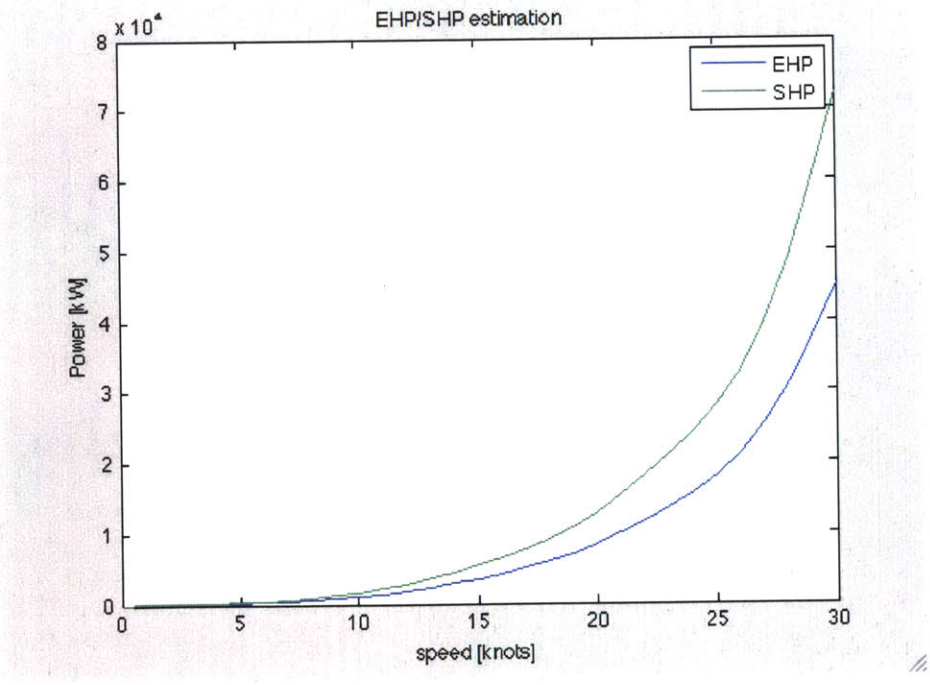


Figure 50 - Matlab® EHP/SHP plot (DDG-51 Flight I example)

Fuel consumption and engine operating load fraction plots are generated for the optimal system selected. Next figures depict such plots for the three different systems categories.

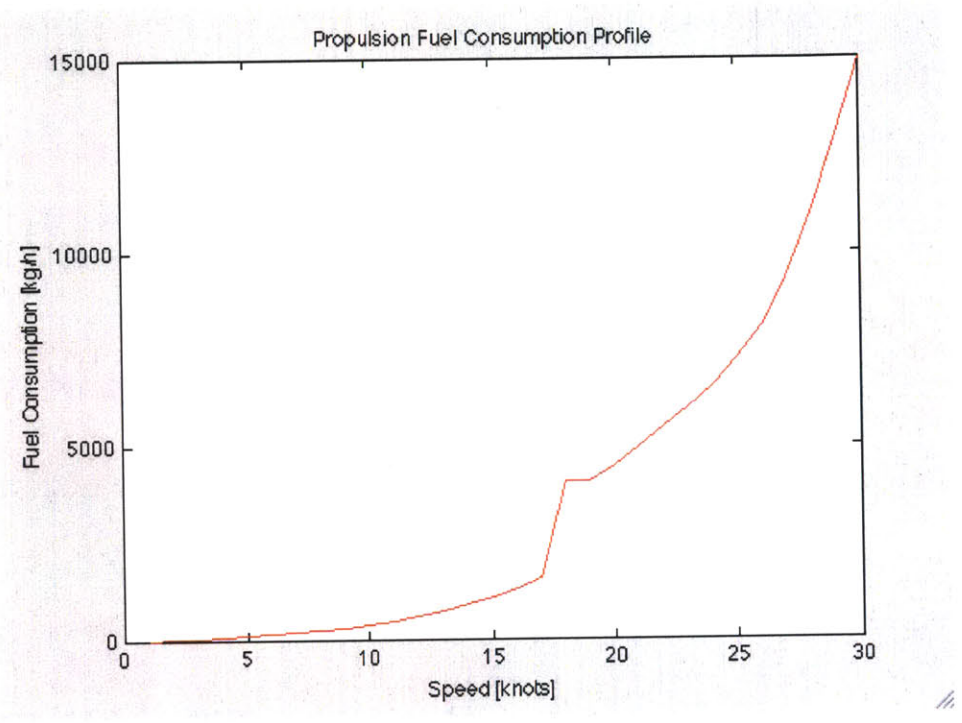


Figure 51 - Matlab® optimal propulsion fuel consumption profile plot (DDG-51 Flight I – Mech-SepSS optimization example)

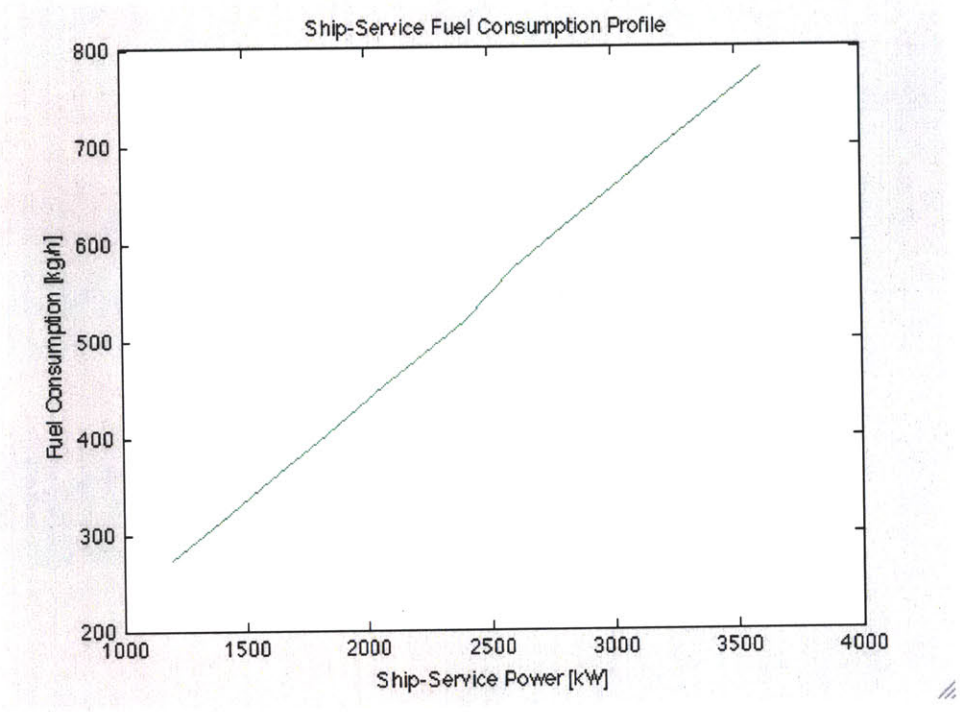


Figure 52 - Matlab® optimal ship-service fuel consumption profile plot (DDG-51 Flight I – Mech-SepSS optimization example)

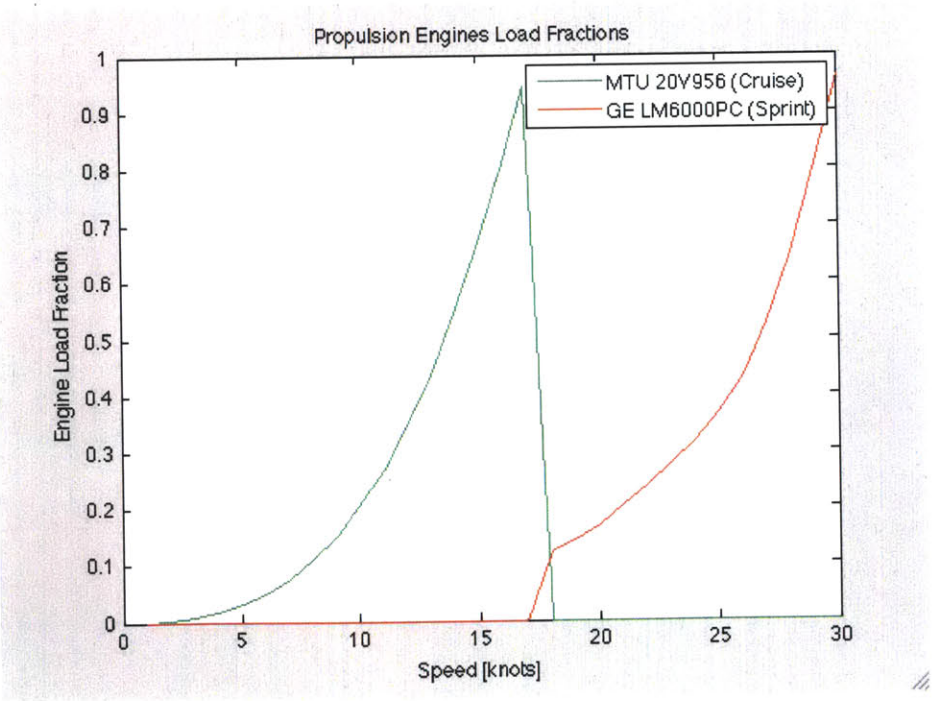


Figure 53 - Matlab® propulsion engines operating load fractions plot (DDG-51 Flight I – Mech-SepSS optimization example)

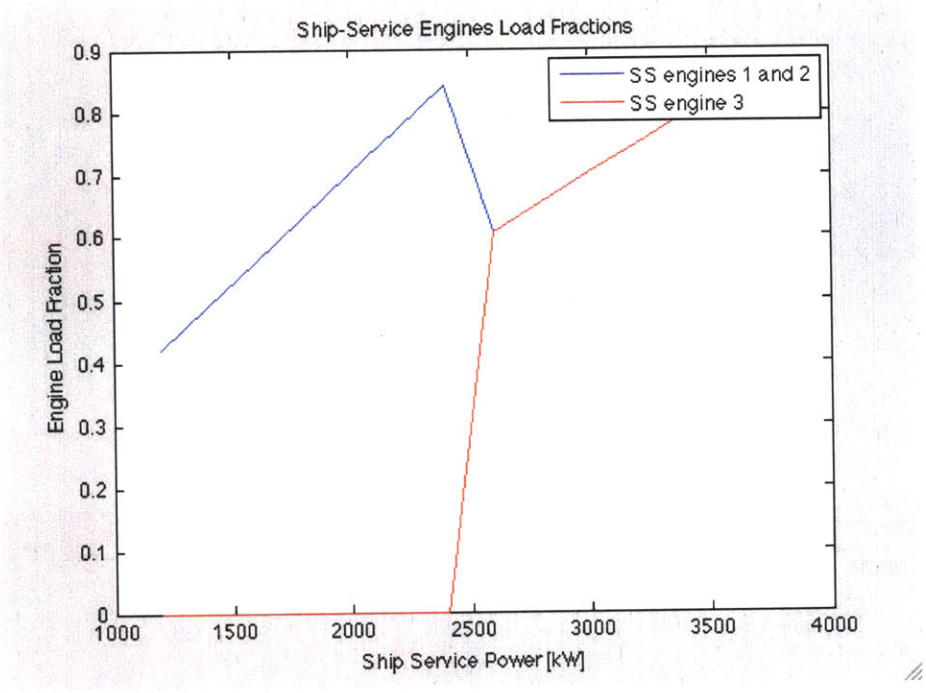


Figure 54 - Matlab® ship-service engines operating load fractions plot (DDG-51 Flight I – Mech-SepSS optimization example)

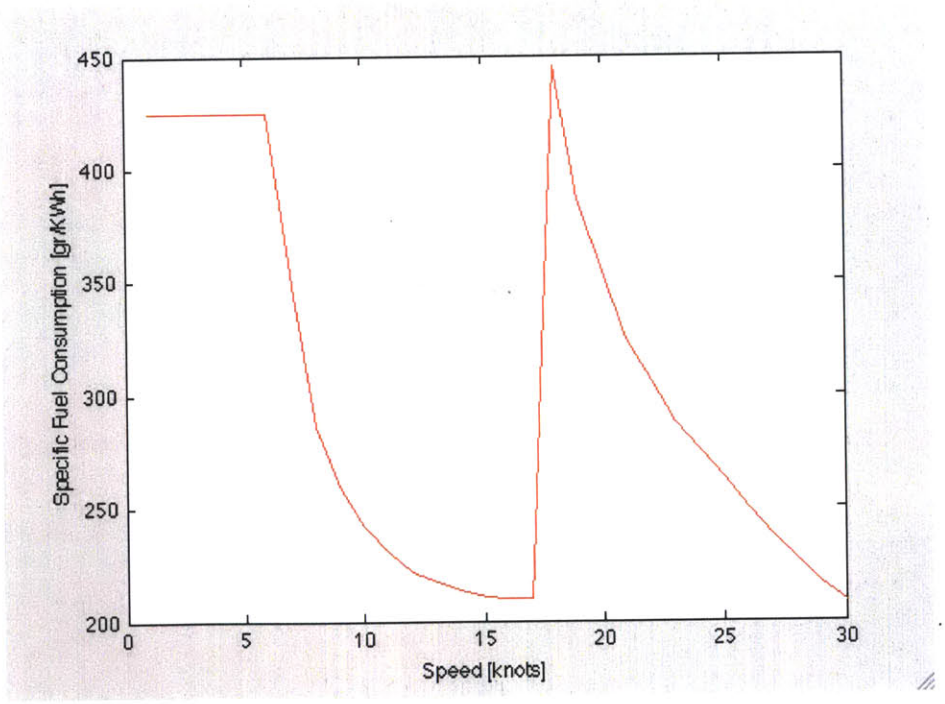


Figure 55 - Matlab® optimal propulsion specific fuel consumption plot (DDG-51 Flight I – Mech-SepSS optimization example)

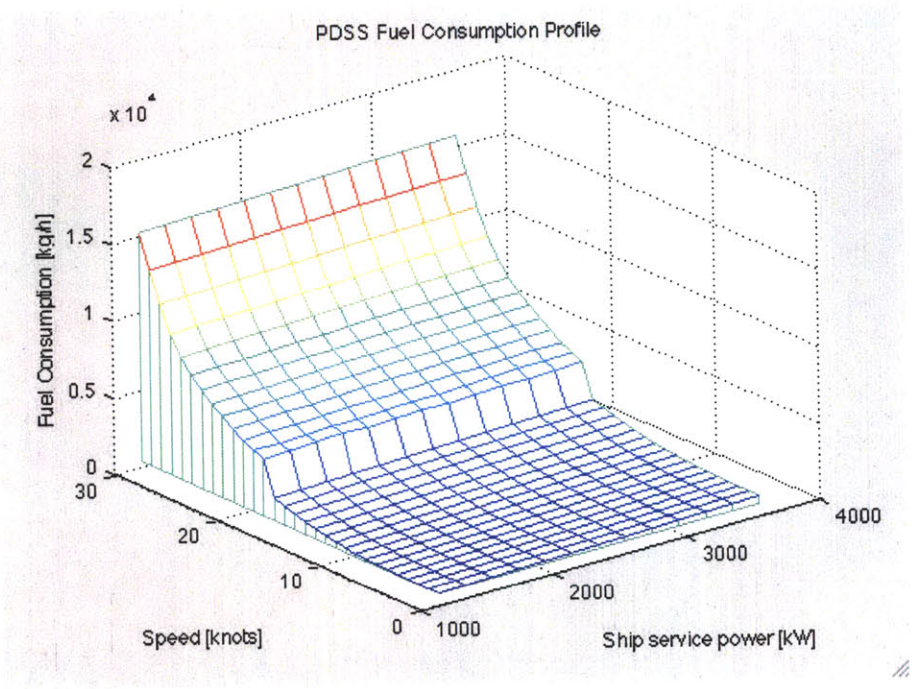


Figure 56 - Matlab® optimal fuel consumption profile plot (DDG-51 Flight I – Mech-PDSS optimization example)

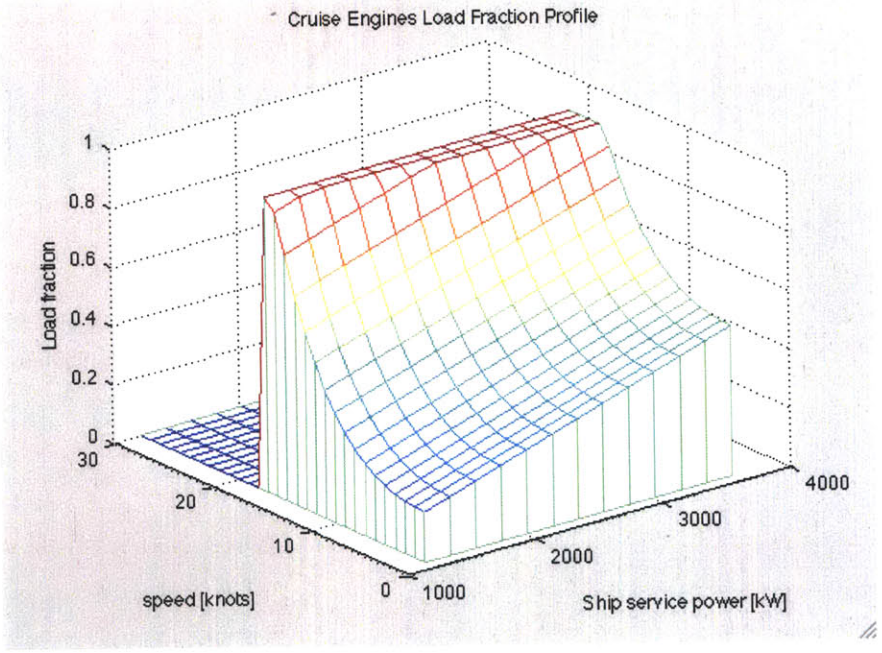


Figure 57 - Matlab® propulsion cruise engines operating load fractions plot (DDG-51 Flight I – Mech-PDSS optimization example)

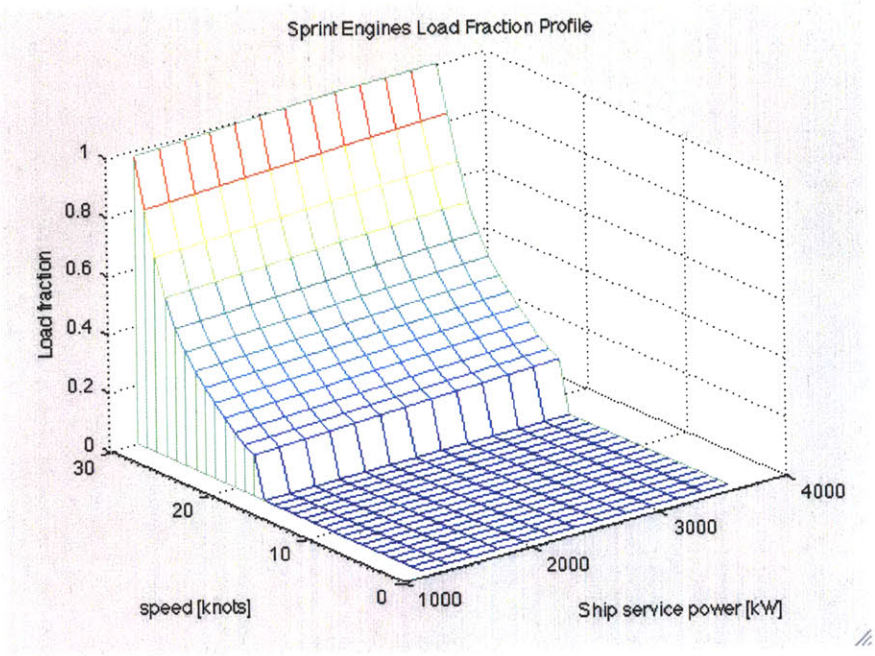


Figure 58 - Matlab® propulsion sprint engines operating load fractions plot (DDG-51 Flight I – Mech-PDSS optimization example)

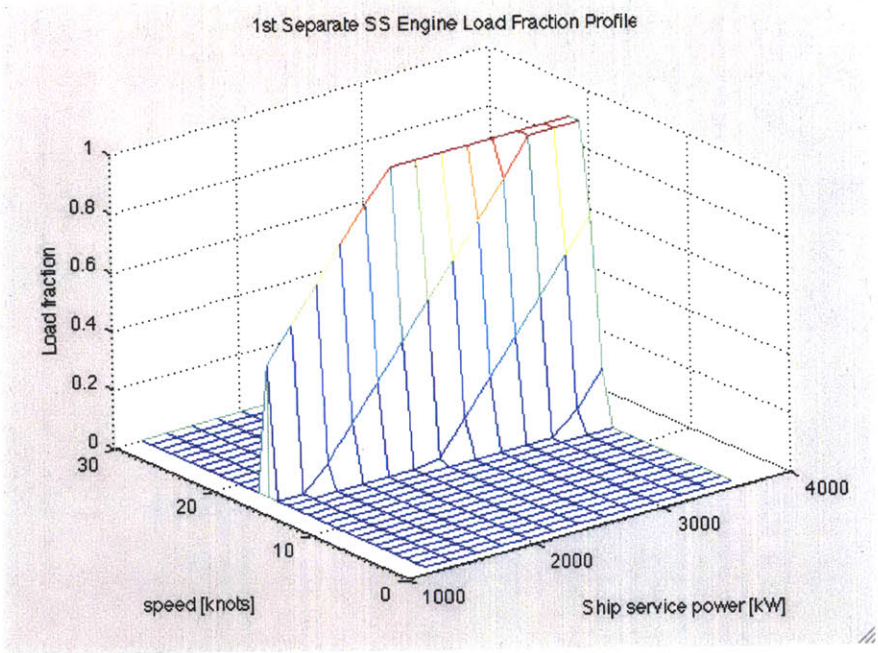


Figure 59 - Matlab® 1st ship-service engine operating load fractions plot (DDG-51 Flight I – Mech-PDSS optimization example)

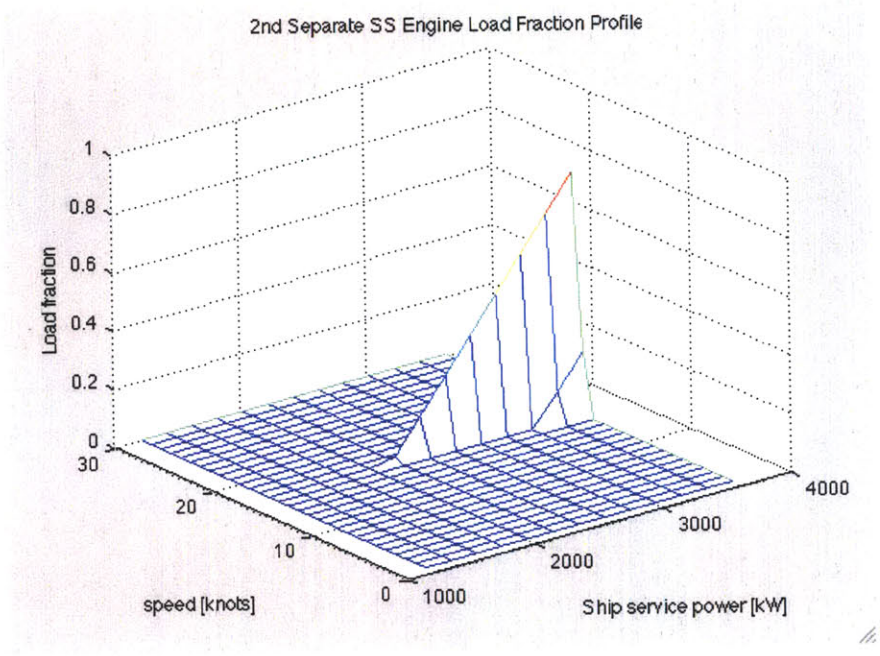


Figure 60 - Matlab® 2nd ship-service engine operating load fractions plot (DDG-51 Flight I – Mech-PDSS optimization example)

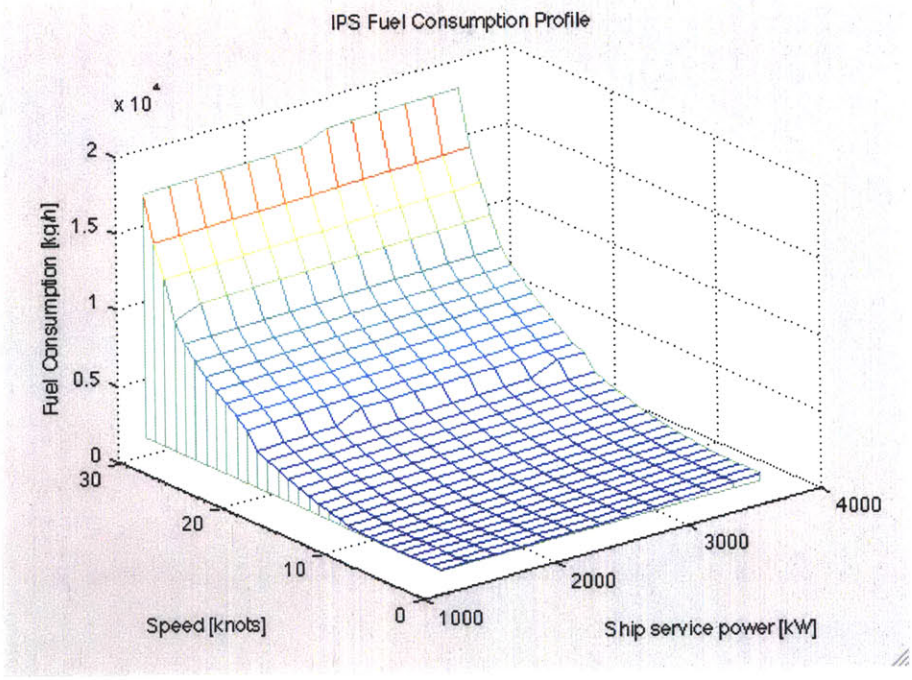


Figure 61 - Matlab® optimal fuel consumption profile plot (DDG-51 Flight I – IPS optimization example)

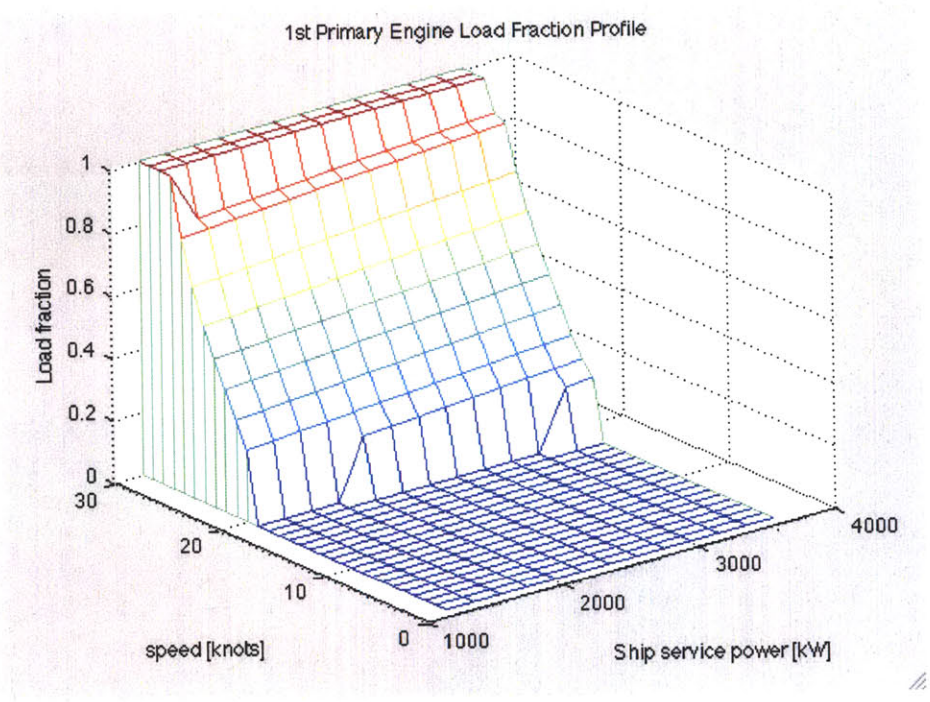


Figure 62 - Matlab® 1st primary engine operating load fractions plot (DDG-51 Flight I – IPS optimization example)

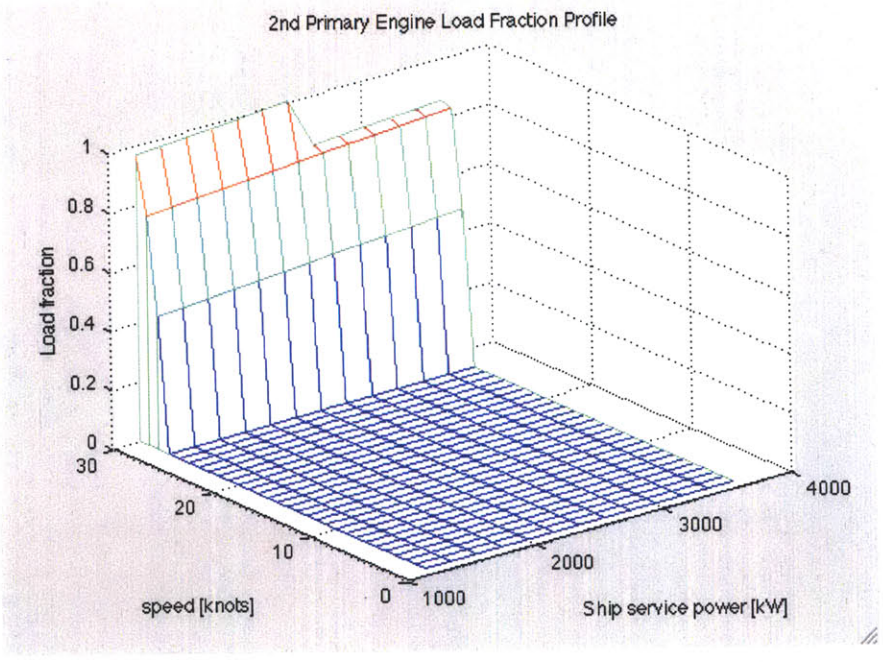


Figure 63 - Matlab® 1st primary engine operating load fractions plot (DDG-51 Flight I – IPS optimization example)

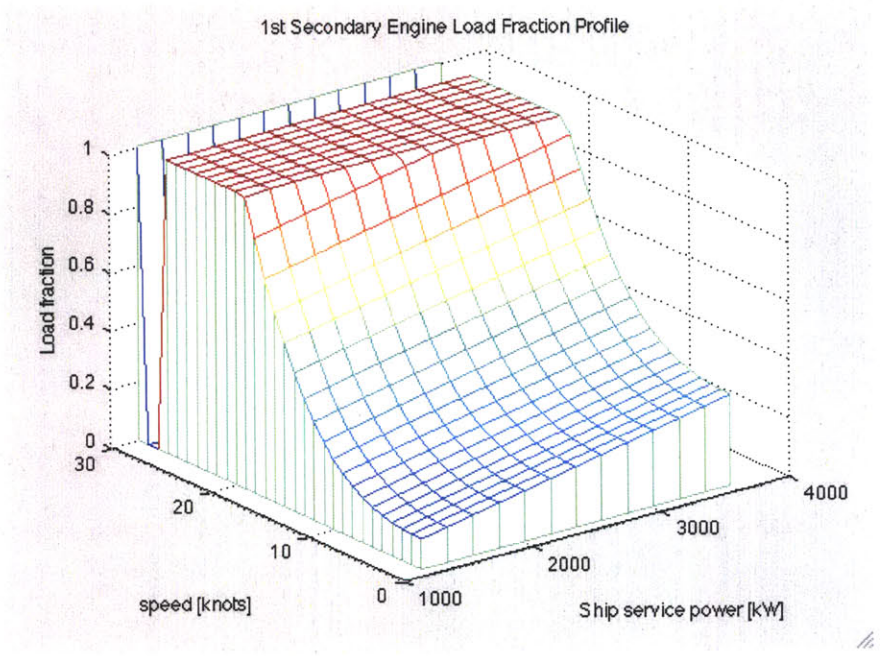


Figure 64 - Matlab® 1st secondary engine operating load fractions plot (DDG-51 Flight I – IPS optimization example)

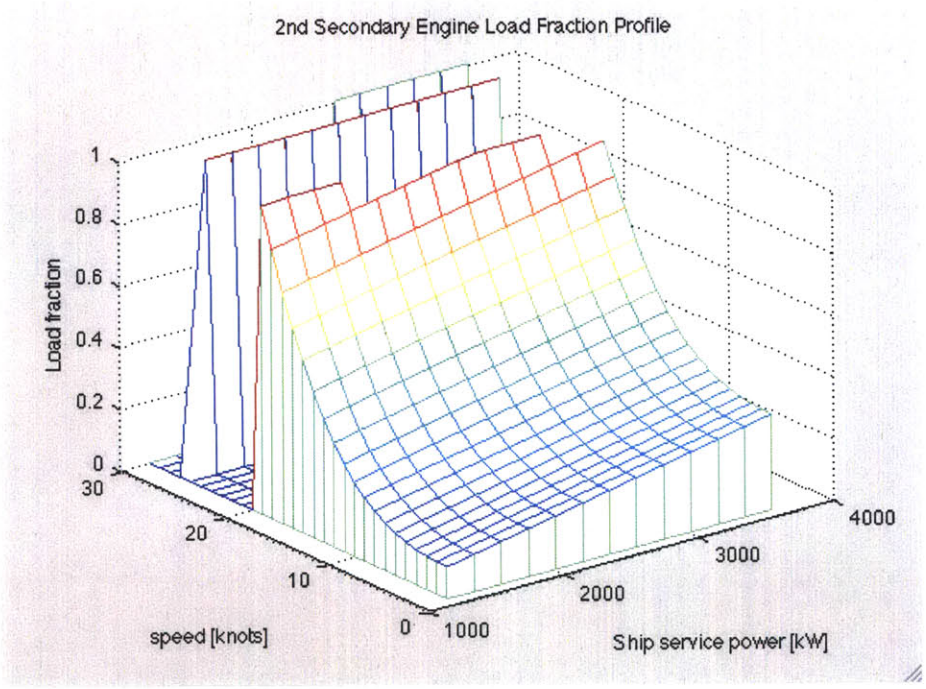


Figure 65 - Matlab® 2nd secondary engine operating load fractions plot (DDG-51 Flight I – IPS optimization example)

Although associated Matlab® script exists, multiple secondary plots were not included in the program running configuration to avoid clutter.

APPENDIX

C

MATLAB® SCRIPTS

SystemsSelection.m:

```
% Program selects propulsion-ship service configuration for given hull
characteristics
% Salt water properties:
% Salinity: 3.5%
% Temperature: 15deg C (59deg F)

clear;clc;close all

global rho g k_visc L B T_f T S S_app A_trans Vol C_b C_wp c_a c_1 c_2
c_5...
    c_15 c_17 m1 m3 lamda h_b A_bt P_b k_plus_1 k1_plus_1 k2_plus_1 D Z...
    n_shafts Va EAR PD h R_dec Rtotal dv Thrust t w v_iter R_iter
profile_data...
    dspeed dss SHP gen_sfc_FL partial_sfc LHV_correction speed M_s
loss_mult ...
    propeng_type propeng_sfc propeng_power propeng_name  propeng_manuf ...
    sseng_type sseng_sfc sseng_power sseng_name sseng_manuf...
    ipseng_type ipseng_sfc ipseng_power ipseng_name ipseng_manuf SHP_1
SHP_2 SHP_3

rho=1025.26; % Sea water density [kg/m^3]
g=9.81; % Gravity accelaration [m/sec^2]
k_visc=1.1883e-6; % Kinematic viscocity [m^2/sec]

% Import speed and ship-service profile information
importCombinedProfile('profile_data.txt')
disp('-Speed and ship service profile information imported.')
dspeed = profile_data(end-1,1); % Maximum sustained speed [knots]
dv = .5144*dspeed; % Maximum ship-service load [kW]
dss = profile_data(1,end-2);

SHP_dec = menu('SHP Information Available: ','Yes','No');
if SHP_dec == 1
    % Import SHP information
    % Import SHP from txt file (1st col:speed[knots] starting with 0knots,
2nd col:SHP[kW])
    importSHP('SHP_data.txt')
```

```
    % Disable interpolation warnings
    warning off MATLAB:interp1:NaNinY
    warning off MATLAB:chckxy:IgnoreNaN

    % Interpolate for missing SHP information
    SHP = zeros(dspeed+1,1);
    for i = 1:dspeed+1
        if isnan(SHP_import(i)) == 1
            SHP(i) = interp1(speed,SHP_import,speed(i),'spline');
        else
            SHP(i) = SHP_import(i);
        end
        if SHP(i) < 0
            SHP(i) = 0;
        end
    end

    clear i
    disp('-SHP information imported.')
else
    R_dec = menu('Resistance Information Available: ','Yes','No');
    if R_dec == 1
        % Import resistance data from txt file (1st col:speed[knots]
starting with 0knots, 2nd col:R[kN])
        importR('R_data.txt')

        % Interpolate for missing resistance information
        Rtotal = zeros(dspeed+1,1);
        for i = 1:dspeed+1
            if isnan(R_import(i)) == 1
                Rtotal(i) = interp1(speed,R_import,speed(i),'spline');
            else
                Rtotal(i) = R_import(i);
            end
            if Rtotal(i) < 0
                Rtotal(i) = 0;
            end
        end

        % Enable previously disabled warnings
        warning on MATLAB:interp1:NaNinY
        warning on MATLAB:chckxy:IgnoreNaN

        clear i
        disp('-Resistance information imported.')

        v = .5144*speed(1:dspeed+1); % [m/sec]
        EHP = Rtotal.*v; % EHP [kW], Rtotal is in [kN]

        % Enter Taylor wake fraction and thrust deduction factor
        prompt = {'Enter Taylor wake fraction:','Enter thrust deduction
 factor:'...
            ,'Enter propeller hub distance from DWL [m]:','Enter prismatic
 coefficient:','Enter LCB (%L fwd MS):'};
        name = 'Input hull information';
        num_lines = 1;
        def = {'0.020','0.055','5.68','0.587','-0.964'};
```

```
        options.Resize = 'on';
        options.WindowStyle = 'normal';
        hull = inputdlg(prompt,name,num_lines,def,options);
        w = str2double(hull{1});
        t = str2double(hull{2});
        h = str2double(hull{3});
        C_p = str2double(hull{4});
        lcb = str2double(hull{5});
        clear prompt name def options num_lines hull_eff

        n_shafts = 2; % Number of shafts, default=2
        Thrust = 1000*Rtotal(end)/(1-t)/n_shafts; % Thrust per shaft [N],
Rtotal is in [kN]
        Va = dv*(1-w); % Speed of advance [m/sec]

    else
        % Resistance estimation using Holtrop-Mennen method
        speed = zeros(dspeed+1,1);
        speed(2:end) = profile_data(2:end-1,1); % [knots]
        v = .5144*speed; % [m/sec]

        % Enter hull information
        prompt = {'Enter LBP [m]:','Enter beam [m]:','Enter fwd draft
[m]:','Enter aft draft [m]:'...
            ,'Enter immersed volume [m^3]:','Enter midship section
coefficient:','Enter waterplane coefficient'...
            ,'Enter transverse sectional area of bulb at FP [m^2]:','Enter
distance of center of bulb area from keel line (<0.6 of fwd draft) [m]:'...
            ,'Enter LCB (%L fwd MS):','Enter total appendage area
[m^2]:','Enter "k+2" appendage factor:','Enter half angle of entrance
[deg]:'...
            ,'Enter immersed transom area [m^2]:','Enter propeller hub
distance from DWL [m]:'};
        name = 'Input hull geometry information';
        num_lines = 1;
        def =
{'142.04','18.15','6.67','6.67','9019.4','0.830','0.788','0','0','-
0.964','0','2.5','11.19','0','5.68'};
        options.Resize = 'on';
        options.WindowStyle = 'normal';
        options.Interpreter='tex';
        hull = inputdlg(prompt,name,num_lines,def,options);
        L = str2double(hull{1}); % LBP [m]
        B = str2double(hull{2}); % Beam [m]
        T_f = str2double(hull{3}); % Fwd draft [m]
        T_a = str2double(hull{4}); % Aft draft [m]
        Vol = str2double(hull{5}); % Immersed hull volume [m^3]
        C_m = str2double(hull{6}); % Midship section coefficient
        C_wp = str2double(hull{7}); % Waterplane coefficient
        A_bt = str2double(hull{8}); % Transverse sectional area of bulb at
FP [m^2]
        h_b = str2double(hull{9}); % Distance of center of bulb area from
keel line (<0.6 of fwd draft) [m]
        lcb = str2double(hull{10}); % LCB (%L fwd MS)
        S_app = str2double(hull{11}); % Total appendage area [m^2]
        k2_plus_1 = str2double(hull{12}); % "k+2" appendage factor
        i_E = str2double(hull{13}); % Half angle of entrance [deg]
        A_trans = str2double(hull{14}); % Immersed transom area [m^2]
```

```
        h = str2double(hull{15}); % Propeller hub distance from DWL [m]
        clear prompt name def options num_lines hull_eff

        % Hull geometry calculations
        T = (T_f+T_a)/2; % Mean draft
        C_b = Vol/(L*B*T); % Block coefficient
        C_p = C_b/C_m; % Prismatic coefficient
        T_over_L = T/L;
        B_over_L = B/L;
        L_over_B = 1/B_over_L;
        Lcube_over_Vol = L^3/Vol;

        S = L*(2*T+B)*C_m^.5*(.453+.4425*C_b-.2862*C_m-
.003467*B/T+.3696*C_wp)+2.38*A_bt/C_b; % Hull wetted surface
        Lr = L*(1-C_p+.06*C_p*lcb/(4*C_p-1)); % Form resistance factor

        if isnan(i_E) == 1
            i_E = 1+89*exp(-(L/B)^.80856*(1-C_wp)^.30484*(1-C_p-
.0225*lcb)^.6367*(Lr/B)^.34574*(100*Vol/L^3)^.16302); % Half angle of
entrance regression formula
        end

        c_stern = 0;
        c_14 = 1+.011*c_stern;
        k1_plus_1 =
.93+.487118*c_14*B_over_L^1.06806*T_over_L^.46106*(L/Lr)^.121563*Lcube_over
_Vol^.36486*(1-C_p)^(-0.604247); % Hull form factor
        if B_over_L < .11
            c_7 = .229577*B_over_L^.33333;
        elseif B_over_L > .25
            c_7 = .5-.0625*(L/B);
        else
            c_7 = B_over_L;
        end
        if L_over_B < 12
            lamda = 1.446*C_p-.03*L_over_B;
        else
            lamda = 1.446*C_p-.36;
        end
        c_3 = .56*A_bt^1.5/(B*T*(.31*A_bt^.5+T_f-h_b));
        c_1 = 2223105*c_7^3.78613*(T/B)^1.07961*(90-i_E)^(-1.37565);
        c_2 = exp(-1.89*c_3^.5); % Accounts for the reduction of wave
resistance due to the action of bulbous bow
        c_5 = 1-.8*A_trans/(B*T*C_m); % Expresses influence of a transom
stern on the wave resistance
        if Lcube_over_Vol < 512
            c_15 = -1.69385;
        elseif Lcube_over_Vol > 1726.91
            c_15 = 0;
        else
            c_15 = -1.69385+(L/Vol^(1/3)-8)/2.36;
        end
        if C_p < .8
            c_16 = 8.07981*C_p-13.8673*C_p^2+6.984388*C_p^3;
        else
            c_16 = 1.73014-.7067*C_p;
        end
```

```
        m1 = .0140407*L/T-1.75254*Vol^(1/3)/L-4.79323*B_over_L-c_16;
        c_17 = 6919.3*C_m^(-1.3346)*Lcube_over_Vol^(-2.00977)*(L/B-
2)^1.40692;
        m3 = -7.2035*B_over_L^.326869*(T/B)^.605375;
        P_b = .56*A_bt^.5/(T_f-1.5*h_b);

        % Model-ship correlation allowance
        corr = menu('Correlation allowance estimation method:','Holtrop-
Mennen','ASSET');
        if corr == 1
            Fdraft_over_L = T_f/L;
            if Fdraft_over_L <=.04
                c_4 = Fdraft_over_L;
            else
                c_4 = .04;
            end
            c_a = .006*(L+100)^(-.16)-
.00205+.003*sqrt(L/7.5)*C_b^4*c_2*(.04-c_4);
        else
            L_ft = L*3.28084; % Length [ft]
            if L_ft < 190
                c_a = .0008;
            elseif L_ft >960
                c_a = .0002;
            else
                c_a = .008289/L_ft^.333-.00064;
            end
        end

        Rw = zeros(length(v),1);
        Rapp = zeros(length(v),1);
        Rtr = zeros(length(v),1);
        Rb = zeros(length(v),1);
        Rtotal = zeros(length(v),1);
        Fr_plot = zeros(length(v),1);

        for i = 2:length(v)
            [R_tcalc,R_wcalc,R_appcalc,R_trcalc,R_bcalc] =
resistancef(v(i)); % [N]
            Rtotal(i) = R_tcalc/1000; % [kN]
            Rw(i) = R_wcalc/1000; % [kN]
            Rapp(i) = R_appcalc/1000; % [kN]
            Rtr(i) = R_trcalc/1000; % [kN]
            Rb(i) = R_bcalc/1000; % [kN]
            Fr_plot(i) = v(i)/(g*L)^.5;
        end

        % Prompt for appendgage resistance
        if sum(Rapp) == 0
            default = 'No';
            app1 = questdlg('Would you like to enter appendage resistance
as a percentage of bare hull resistance?','Appendage
resistance','Yes','No',default);
            if strcmp(app1,'Yes') == 1
                % Enter percentage
                prompt = {'Enter percentage of bare hull resistance (%):'};
                name = 'Appendage resistance';
```

```
                num_lines = 1;
                def = {'15'};
                options.Resize = 'on';
                options.WindowStyle = 'modal';
                app2 = inputdlg(prompt,name,num_lines,def,options);
                app_factor = .01*str2double(app2);
                for i = 2:length(v)
                    Rapp(i) = app_factor*Rtotal(i);
                    Rtotal(i) = Rtotal(i)+Rapp(i);
                end
                clear app2
            end
        end
        clear app1

        % Apply power design margin
        M_d = 0.1; % USN Early parametric prediction
        Rtotal(i) = (1+M_d)*Rtotal(i);
        EHP = Rtotal.*v; % EHP [kW], Rtotal is in [kN]

        % Resistance results
        fprintf('\n')
        disp('-------------------------------------------------------------
----------')
        disp('RESISTANCE RESULTS:')
        fprintf('\n')
        disp('Speed   Rw[kN]  Rapp[kN]  Rtr[kN]   Rb[kN]  Rtotal[kN]')
        for i=1:size(speed,1)
        fprintf('%2.0f     %6.1f  %5.1f     %5.1f     %5.1f   %6.1f\n',
speed(i), Rw(i), Rapp(i), Rtr(i), Rb(i), Rtotal(i));
        end
        fprintf('\n')

        figure;

plot(speed,Rtotal,'r',speed,Rw,'b',speed,Rapp,'k',speed,Rtr,'y',speed,Rb,'c
'),xlabel('Speed [knots]'),ylabel('Rtotal [kN]'),title('Resistance
estimation'),
        legend('Total resistance','Wave resistance','Appendage
resistance','Transom stern resistance','Bulbous bow resistance'),grid on

    end

    % Propeller selection

    % Enter propeller diameter limits
    prompt = {'Enter min propeller diameter [m]:','Enter max propeller
diameter [m]:'};
    name = 'Input propeller diameter limits';
    num_lines = 1;
    def = {'4.5','5.18'};
    options.Resize = 'on';
    options.WindowStyle = 'modal';
    prop_lim = inputdlg(prompt,name,num_lines,def,options);
    Dmin = str2double(prop_lim{1});
    Dmax = str2double(prop_lim{2});
    clear prompt name def options num_lines prop_lim
```

```
    Z = 5; % Number of blades, default=5
    n_shafts = 2; % Number of shafts, default=2

    % fmincon setup
    % Design variables: P/D, rps, D, EAR
    lb = [.5 ; 1 ; Dmin ; .3];
    ub = [1.5 ; 4.5 ; Dmax ; 1.05];
    x0 = [1 ; 2 ; (Dmin+Dmax)/2; .7];
    A = [];
    b = [];
    Aeq = [];
    beq = [];
    nonlcon = @sqppropcon2;
    options = optimset('Algorithm','active-set','TolFun',1e-6,'TolCon',1e-
6,'TolX',1e-6,'Display','off');

    problem =
createOptimProblem('fmincon','objective',@sqppropf2,'x0',x0,'lb',lb,'ub',ub
,'nonlcon',nonlcon,'options',options);

    % Global search optimization solver
    gs = GlobalSearch('Display','off');
    [x,f] = run(gs,problem);

    PD = x(1);
    rps = x(2);
    D = x(3);
    EAR = x(4);
    n0 = -f;

    clear x0 x options nonlcon lb ub f gs Dmin Dmax

    nr = .9737+.111*(C_p-.0225*lcb)-.06325*PD; % Relative rotative
efficiency

    disp('-----------------------------------------------------------------
------')
    fprintf('SELECTED PROPELLER CHARACTERISTICS:\n\n')
    fprintf('D:    %1.2fm\n', D);
    fprintf('P/D:  %1.2f\n', PD);
    fprintf('RPM:  %3.1f\n', rps*60);
    fprintf('EAR:  %1.3f\n', EAR);
    fprintf('Open Water Efficiency:        %1.3f\n', n0);
    fprintf('Taylor wake fraction:         %1.3f\n', w);
    fprintf('Thrust deduction factor:      %1.3f\n', t);
    fprintf('Relative rotative efficiency: %1.3f\n\n', nr);

    % Open water efficiency for whole speed range
    n0 = zeros(dspeed+1,1);
    x0 = .5;
    for i = 1:length(v)
        R_iter = Rtotal(i)*1000; % [N]
        v_iter = v(i);
        options = optimset('Display','off');
        [x,fval] = fsolve(@Kt,x0,options);
```

```
        n0(i) = OpenWaterEff(x);
    end
    clear options x x0
    n_cpp = 0.997; % CPP efficiency multiplier. Represents decreased
efficiency due to larger hub diameter
    n0_cpp = n0*n_cpp;
    n0_cpp(1:9) = n0(8); % Assume constant open-water efficiency for speeds
less than 7knots

%     figure % Plot open-water efficiency for FPP and CPP (approximate)
%     plot(speed(1:dspeed+1),n0,speed(1:dspeed+1),n0_cpp,'r'),xlabel('Speed
[knots]'),ylabel('Open water efficiency'),legend('FP','CP')

    SHP = EHP.*(1-w)./(nr*n0*(1-t)*n_shafts); % SHP per shaft [kW]

    figure
    plot(speed(1:dspeed+1),EHP,'b',speed(1:dspeed+1),2*SHP,'g')
    xlabel('speed [knots]'),ylabel('Power [kW]')
    legend('EHP','SHP'),title('EHP/SHP estimation')

end


% Enter power service margin
prompt = {'Enter power service margin (%):'};
name = 'Power sevice margin for maximum speed';
num_lines = 1;
def = {'0'};
options.Resize = 'on';
options.WindowStyle = 'modal';
margin_s = inputdlg(prompt,name,num_lines,def,options);
M_s = .01*str2double(margin_s);
clear prompt name def options num_lines margin_s


% Partial-load prime mover sfc
% Import file with sfc information at partial load from Woud, Stapersma
% (2002)
% Txt file is expected to have 15 rows of data
% Subscript-Engine Type Correspondence:
% 1= 1000rpm Diesel Engine at Propeller Law
% 2= 1000rpm Diesel Engine at Constant Speed
% 3= High Speed Diesel Engine at Propeller Law
% 4= High Speed Diesel Engine at Constant Speed
% 5= Gas Turbine at Propeller Law
% 6= Gas Turbine at Constant Speed
importPartialSFC('sfc.txt')

% Pre-allocate
partial_sfc = zeros(6,96);
x = zeros(1,15);
y = zeros(1,15);

% Perform cubic spline interpolation
for i=1:15
    x(i)=data(i,1);
```

```
    y(i)=data(i,2);
end
pp = spline(x,y);
ppp(1:6,:) = pp;

for j=1:6
    j_x=1+2*(j-1);
    j_y=2+2*(j-1);
    for i=1:15
        x(i)=data(i,j_x);
        y(i)=data(i,j_y);
    end
    pp=spline(x,y);
    ppp(j,:) = pp;
    partial_sfc(j,:)=ppval(pp,5:100);

    % Use to plot datapoints and check piecewise polynomial fits the
    % datapoints
%     figure
%     plot(x,y,'o',5:100,partial_sfc(j,:)),xlabel('Load fraction
[%]'),ylabel('sfc [gr/kWh]'),title('sfc at partial load')
end

LHV_correction = 42700/42800; % Adjustment for F-76 fuel
gen_sfc_FL = partial_sfc(:,96)*LHV_correction; % Full-load sfc

clear data textdata pp j i x y j_x j_y colheaders


% Load engine data
load detailedengdata


% Transmission efficiency loss due to partial load operation
trans_los_matrix = [10 2.7 ; 20 2.1 ; 30 1.7 ; 40 1.5 ; 70 1.2 ; 90 1.05 ;
100 1];
pp_mech = spline(trans_los_matrix(:,1),trans_los_matrix(:,2));
loss_mult=ppval(pp_mech,5:100);

% % Use to plot datapoints and check piecewise polynomial fits the
datapoints
%
% figure;
%
plot(trans_los_matrix(:,1),trans_los_matrix(:,2),'o',5:100,loss_mult),xlabe
l('Load fraction [%]'),ylabel('Loss multiplier'),title('Transmission loss
multiplier for partial load operation')


sys_dec = menu('Optimization Menu: ','Mech-SepSS (Single Run)','Mech-PDSS
(Single Run)','IPS (Single Run)','Global Optimum Search (Multiple
Runs)','Cancel');
tic
switch sys_dec
    case 1
        % Mech-SepSS
```

```
        n_s = 0.97; % Sterntube and line bearings efficiencies
        if SHP_dec == 2
            SHP_1 = SHP/n_cpp*n0/n0_cpp; % SHP including CPP open-water eff
adjustment [kW]
        else
            SHP_1 = SHP;
        end
        SHP_1 = SHP_1/n_s; % SHP including sterntube and line bearings
efficiencies [kW]

        % GA optimization using matlab's build-in functions
        A = [];
        b = [];
        Aeq = [];
        beq = [];
        nonlcons = [];

        LB = [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0];
        UB = [1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1];

        options =
gaoptimset('PopulationType','bitstring','PopulationSize',400,'CreationFcn',
@gacreationuniform...

,'MutationFcn',@mutationuniform,'CrossoverFcn',@crossoverscattered,'TolFun'
,1e-6,'Generations',100,...

'SelectionFcn',@selectionstochunif,'CrossoverFraction',0.9,'EliteCount',1,'
StallGenLimit',50,'Display','off','PlotFcns',@gaplotbestf);

        warning off globaloptim:constrvalidate:unconstrainedMutationFcn %
Suppress warning

        x = ga(@MechSepSSf,22,A,b,Aeq,beq,LB,UB,nonlcons,options);

        MechSepSSevalf(x); % Evaluates optimal configuration for reporting
purposes
    case 2
        % Mech-PDSS
        n_s = 0.97; % Sterntube and line bearings efficiencies
        if SHP_dec == 2
            SHP_2 = SHP/n_cpp*n0/n0_cpp; % SHP including CPP open-water eff
adjustment [kW]
        else
            SHP_2 = SHP;
        end
        SHP_2 = SHP_2/n_s; % SHP including sterntube and line bearings
efficiencies [kW]

        % GA optimization using matlab's build-in functions
        A = [];
        b = [];
        Aeq = [];
        beq = [];
        nonlcons = [];

        LB = [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0];
```

```
        UB = [1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1];

        options =
gaoptimset('PopulationType','bitstring','PopulationSize',400,'CreationFcn',
@gacreationuniform...

,'MutationFcn',@mutationuniform,'CrossoverFcn',@crossoverscattered,'TolFun'
,1e-6,'Generations',100,...

'SelectionFcn',@selectionstochunif,'CrossoverFraction',0.9,'EliteCount',1,'
StallGenLimit',50,'Display','off','PlotFcns',@gaplotbestf);

        warning off globaloptim:constrvalidate:unconstrainedMutationFcn %
Suppress warning

        x = ga(@MechPDSSf,23,A,b,Aeq,beq,LB,UB,nonlcons,options);

        MechPDSSevalf(x); % Evaluates optimal configuration for reporting
purposes
    case 3
        % IPS

        n_s = 0.98; % Sterntube and line bearings efficiencies
        SHP_3 = SHP/n_s; % SHP including sterntube and line bearings
efficiencies [kW]

        % GA optimization using matlab's build-in functions
        A = [];
        b = [];
        Aeq = [];
        beq = [];
        nonlcons = [];

        LB = [0 0 0 0 0 0 0 0 0];
        UB = [1 1 1 1 1 1 1 1 1 ];

        options =
gaoptimset('PopulationType','bitstring','PopulationSize',40,'CreationFcn',@
gacreationuniform...

,'MutationFcn',@mutationuniform,'CrossoverFcn',@crossoverscattered,'TolFun'
,1e-6,'Generations',100,...

'SelectionFcn',@selectionstochunif,'CrossoverFraction',0.9,'EliteCount',1,'
StallGenLimit',50,'Display','off','PlotFcns',@gaplotbestf);

        warning off globaloptim:constrvalidate:unconstrainedMutationFcn %
Suppress warning

        x = ga(@IPSf,9,A,b,Aeq,beq,LB,UB,nonlcons,options);

        IPSevalf(x); % Evaluates optimal configuration for reporting
purposes
    case 4
        % Mech-SepSS
        n_s = 0.97; % Sterntube and line bearings efficiencies
```

```
        if SHP_dec == 2
            SHP_1 = SHP/n_cpp*n0/n0_cpp; % SHP including CPP open-water eff
adjustment [kW]
        else
            SHP_1 = SHP;
        end
        SHP_1 = SHP_1/n_s; % SHP including sterntube and line bearings
efficiencies [kW]

        % GA optimization using matlab's build-in functions
        A = [];
        b = [];
        Aeq = [];
        beq = [];
        nonlcons = [];

        LB = [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0];
        UB = [1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1];

        options =
gaoptimset('PopulationType','bitstring','PopulationSize',400,'CreationFcn',
@gacreationuniform...

,'MutationFcn',@mutationuniform,'CrossoverFcn',@crossoverscattered,'TolFun'
,1e-6,'Generations',100,...

'SelectionFcn',@selectionstochunif,'CrossoverFraction',0.9,'EliteCount',1,'
StallGenLimit',30,'Display','off','PlotFcns',@gaplotbestf);

        warning off globaloptim:constrvalidate:unconstrainedMutationFcn %
Suppress warning

        results_msep1 = zeros(10,22);
        results_msep2 = zeros(10,1);
        for i=1:10
            [x,fval] =
ga(@MechSepSSf,22,A,b,Aeq,beq,LB,UB,nonlcons,options);

            results_msep1(i,:) = x;
            results_msep2(i) = fval;
        end
        fc = min(results_msep2);
        x_ind = find(results_msep2==fc);
        x_ind = x_ind(1);
        x_msep = results_msep1(x_ind,:);


[opt_msep_fc_prop,opt_msep_sfc_prop,opt_msep_fc_avgss,opt_msep_sfc_avgss]=M
echSepSSevalf(x_msep); % Evaluates optimal configuration for reporting
purposes


        % Mech-PDSS
        n_s = 0.97; % Sterntube and line bearings efficiencies
        if SHP_dec == 2
            SHP_2 = SHP/n_cpp*n0/n0_cpp; % SHP including CPP open-water eff
```

```
adjustment [kW]
        else
            SHP_2 = SHP;
        end
        SHP_2 = SHP_2/n_s; % SHP including sterntube and line bearings
efficiencies [kW]

        % GA optimization using matlab's build-in functions
        A = [];
        b = [];
        Aeq = [];
        beq = [];
        nonlcons = [];

        LB = [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0];
        UB = [1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1];

        options =
gaoptimset('PopulationType','bitstring','PopulationSize',400,'CreationFcn',
@gacreationuniform,...

'MutationFcn',@mutationuniform,'CrossoverFcn',@crossoverscattered,'TolFun',
1e-6,'Generations',100,...

'SelectionFcn',@selectionstochunif,'CrossoverFraction',0.9,'EliteCount',1,'
StallGenLimit',30,'Display','off','PlotFcns',@gaplotbestf);

        results_mpd1 = zeros(10,23);
        results_mpd2 = zeros(10,1);
        for i=1:10
            [x,fval] =
ga(@MechPDSSf,23,A,b,Aeq,beq,LB,UB,nonlcons,options);

            results_mpd1(i,:) = x;
            results_mpd2(i) = fval;
        end
        fc = min(results_mpd2);
        x_ind = find(results_mpd2==fc);
        x_ind = x_ind(1);
        x_mpd = results_mpd1(x_ind,:);

        [opt_mpd_fc,opt_mpd_sfc]=MechPDSSevalf(x_mpd); % Evaluates optimal
configuration for reporting purposes


        % IPS
        n_s = 0.98; % Sterntube and line bearings efficiencies
        SHP_3 = SHP/n_s; % SHP including sterntube and line bearings
efficiencies [kW]

        % GA optimization using matlab's build-in functions
        A = [];
        b = [];
        Aeq = [];
        beq = [];
        nonlcons = [];
```

```
        LB = [0 0 0 0 0 0 0 0 0];
        UB = [1 1 1 1 1 1 1 1 1];

        options =
gaoptimset('PopulationType','bitstring','PopulationSize',40,'CreationFcn',@
gacreationuniform,...

'MutationFcn',@mutationuniform,'CrossoverFcn',@crossoverscattered,'TolFun',
1e-6,'Generations',100,...

'SelectionFcn',@selectionstochunif,'CrossoverFraction',0.9,'EliteCount',1,'
StallGenLimit',20,'Display','off','PlotFcns',@gaplotbestf);

        results_ips1 = zeros(10,9);
        results_ips2 = zeros(10,1);
        for i=1:10
            [x,fval] = ga(@IPSf,9,A,b,Aeq,beq,LB,UB,nonlcons,options);

            results_ips1(i,:) = x;
            results_ips2(i) = fval;
        end
        fc = min(results_ips2);
        x_ind = find(results_ips2==fc);
        x_ind = x_ind(1);
        x_ips = results_ips1(x_ind,:);

        [opt_ips_fc,opt_ips_sfc]=IPSevalf(x_ips); % Evaluates optimal
configuration for reporting purposes

        save optimal

    case 5
        disp('-----------------------------------------------------------------
----------')
        disp('-End of program.')
        disp('-----------------------------------------------------------------
----------')

end
toc
```

Function *importCombinedProfile.m*:

```
function importCombinedProfile(fileToRead1)
% Imports data from the specified file
% FILETOREAD1:  file to read
% Auto-generated by MATLAB

% Import the file
rawData1 = importdata(fileToRead1);

[~,name] = fileparts(fileToRead1);
newData1.(genvarname(name)) = rawData1;

% Create new variables in the base workspace from those fields.
vars = fieldnames(newData1);
for i = 1:length(vars)
    assignin('base', vars{i}, newData1.(vars{i}));
end
```

Function *importR.m*:

```
function importR(fileToRead1)
%  Imports data from the specified file
%  FILETOREAD1:  file to read
%  Auto-generated by MATLAB

% Import the file
newData1 = importdata(fileToRead1);

% Create new variables in the base workspace from those fields.
for i = 1:size(newData1.colheaders, 2)
    assignin('base', genvarname(newData1.colheaders{i}),
newData1.data(:,i));
end
```

Function *ImportSHP.m*:

```
function importSHP(fileToRead1)
%  Imports data from the specified file
%  FILETOREAD1:  file to read
%  Auto-generated by MATLAB

% Import the file
newData1 = importdata(fileToRead1);

% Create new variables in the base workspace from those fields.
for i = 1:size(newData1.colheaders, 2)
    assignin('base', genvarname(newData1.colheaders{i}),
newData1.data(:,i));
end
```

Function *importPartialSFC.m*:

```
function importPartialSFC(fileToRead1)
%  Imports data from the specified file
%  FILETOREAD1:  file to read
%  Auto-generated by MATLAB

% Import the file
newData1 = importdata(fileToRead1);

% Create new variables in the base workspace from those fields.
vars = fieldnames(newData1);
for i = 1:length(vars)
    assignin('base', vars{i}, newData1.(vars{i}));
end
```

Function *resistancef.m*:

```
function [Rtotal,R_w,R_app,R_tr,R_b] = resistancef(v)
% Function calculates resistance using Holtrop-Mennen equations

global rho g k_visc L B T_f S S_app A_trans Vol C_wp c_a c_1 c_2 c_5 c_15
c_17 m1 m3 lamda h_b A_bt P_b k1_plus_1 k2_plus_1


Fr = v/(g*L)^.5; % Froude number
Re = L*v/k_visc; % Reynolds number

% Frictional resistance estimation
c_f = .075/(log10(Re)-2)^2;
R_f = .5*rho*v^2*S*c_f;

% Appendage resistance estimation
R_app = .5*rho*v^2*S_app*k2_plus_1*c_f;

% Wave resistance estimation
d = -.9;
m4 = c_15*.4*exp(-.034/Fr^3.29);
R_w1 = c_1*c_2*c_5*Vol*rho*g*exp(m1*Fr^d+m4*cos(lamda/Fr^2));
R_w2 = c_17*c_2*c_5*Vol*rho*g*exp(m3*Fr^d+m4*cos(lamda/Fr^2));
if Fr < .4
    R_w = R_w1;
elseif Fr > .55
    R_w = R_w2;
else
    R_w = R_w1+(10*Fr-4)*(R_w2-R_w1)/1.5;
end

% Transom resistance estimation
F_nt = v/(2*g*A_trans/(B+B*C_wp))^.5;
if F_nt < 5
    c_6 = .2*(1-.2*F_nt);
else
    c_6 = 0;
end
R_tr = .5*rho*v^2*A_trans*c_6;

% Bulbous bow resistance estimation (presence near surface)
F_ni = v/(g*(T_f-h_b-.25*A_bt^.5)+.15*v^2)^.5;
R_b = .11*exp(-3*P_b^(-2))*F_ni^3*A_bt^1.5*rho*g;

% Model-Ship correlation allowance resistance
R_a = .5*rho*v^2*S*c_a;

% Total resistance
Rtotal = R_f*k1_plus_1+R_app+R_w+R_b+R_tr+R_a;

end
```

Function *sqppropf2.m*:

```
function J1 = sqppropf2(x)
% Function is used with propeller SQP optimization
% Represents objective function for the optimization problem

global Va D Z w t R_dec dv k1_plus_1 k2_plus_1 S_app S C_b B T L k_visc c_a

% Function input variables
PD = x(1); % P/D
rps = x(2); % Revolution of Propeller per Second [1/sec]
D = x(3); % Propeller diameter [m]
EAR = x(4); % Expanded Area Ratio

if R_dec == 2
    k_plus_1 = k1_plus_1+(k2_plus_1-k1_plus_1)*S_app/(S+S_app);
    Re = L*dv/k_visc; % Reynolds number
    c_f = .075/(log10(Re)-2)^2;
    c_v = k_plus_1*c_f+c_a;
    w = .3095*C_b+10*c_v*C_b-.23*D/(B*T)^.5;
    t = .325*C_b-.1885*D/(B*T)^.5;
    Va = dv*(1-w); % Speed of advance [m/sec]
end

J = Va/(rps*D); % Advance coefficient

% K_t regression equation implementation
K_t=8.80496e-3;
K_t=K_t-.204554*J;
K_t=K_t+.166351*PD;
K_t=K_t+.158114*PD^2;
K_t=K_t-.147581*J^2*EAR;
K_t=K_t-.481497*J*PD*EAR;
K_t=K_t+.415437*PD^2*EAR;
K_t=K_t+.0144043*Z;
K_t=K_t-.0530054*J^2*Z;
K_t=K_t+.0143481*PD*Z;
K_t=K_t+.0606826*J*PD*Z;
K_t=K_t-.0125894*EAR*Z;
K_t=K_t+.0109689*J*EAR*Z;
K_t=K_t-.133698*PD^3;
K_t=K_t+6.38407e-3*PD^6;
K_t=K_t-1.32718e-3*J^2*PD^6;
K_t=K_t+.168496*J^3*EAR;
K_t=K_t-.0507214*EAR^2;
K_t=K_t+.0854559*J^2*EAR^2;
K_t=K_t-.0504475*J^3*EAR^2;
K_t=K_t+.010465*J*PD^6*EAR^2;
K_t=K_t-6.48272e-3*J^2*PD^6*EAR^2;
K_t=K_t-8.417281e-3*PD^3*Z;
K_t=K_t+.0168424*J*PD^3*Z;
K_t=K_t-1.02296e-3*J^3*PD^3*Z;
K_t=K_t-.0317791*PD^3*EAR*Z;
K_t=K_t+.018604*J*EAR^2*Z;
K_t=K_t-4.10798e-3*PD^2*EAR^2*Z;
K_t=K_t-6.06848e-4*Z^2;
K_t=K_t-4.9819e-3*J*Z^2;
```

```
K_t=K_t+2.5983e-3*J^2*Z^2;
K_t=K_t-5.60528e-4*J^3*Z^2;
K_t=K_t-1.63652e-3*J*PD^2*Z^2;
K_t=K_t-3.28787e-4*J*PD^6*Z^2;
K_t=K_t+1.16502e-4*J^2*PD^6*Z^2;
K_t=K_t+6.90904e-4*EAR*Z^2;
K_t=K_t+4.21749e-3*PD^3*EAR*Z^2;
K_t=K_t+5.65229e-5*J^3*PD^6*EAR*Z^2;
K_t=K_t-1.46564e-3*PD^3*EAR^2*Z^2;

% K_q regression equation implementation
K_q=3.79368e-3;
K_q=K_q+8.86523e-3*J^2;
K_q=K_q-.032241*J*PD;
K_q=K_q+3.44778e-3*PD^2;
K_q=K_q-.0408811*PD*EAR;
K_q=K_q-.108009*J*PD*EAR;
K_q=K_q-.0885381*J^2*PD*EAR;
K_q=K_q+.188561*PD^2*EAR;
K_q=K_q-3.70871e-3*J*Z;
K_q=K_q+5.13696e-3*PD*Z;
K_q=K_q+.0209449*J*PD*Z;
K_q=K_q+4.74319e-3*J^2*PD*Z;
K_q=K_q-7.23408e-3*J^2*EAR*Z;
K_q=K_q+4.38388e-3*J*PD*EAR*Z;
K_q=K_q-.0269403*PD^2*EAR*Z;
K_q=K_q+.0558082*J^3*EAR;
K_q=K_q+.0161886*PD^3*EAR;
K_q=K_q+3.18086e-3*J*PD^3*EAR;
K_q=K_q+.015896*EAR^2;
K_q=K_q+.0471729*J*EAR^2;
K_q=K_q+.0196283*J^3*EAR^2;
K_q=K_q-.0502782*PD*EAR^2;
K_q=K_q-.030055*J^3*PD*EAR^2;
K_q=K_q+.0417122*J^2*PD^2*EAR^2;
K_q=K_q-.0397722*PD^3*EAR^2;
K_q=K_q-3.50024e-3*PD^6*EAR^2;
K_q=K_q-.0106854*J^3*Z;
K_q=K_q+1.10903e-3*J^3*PD^3*Z;
K_q=K_q-3.13912e-4*PD^6*Z;
K_q=K_q+3.5985e-3*J^3*EAR*Z;
K_q=K_q-1.42121e-3*PD^6*EAR*Z;
K_q=K_q-3.83637e-3*J*EAR^2*Z;
K_q=K_q+.0126803*PD^2*EAR^2*Z;
K_q=K_q-3.18278e-3*J^2*PD^3*EAR^2*Z;
K_q=K_q+3.34268e-3*PD^6*EAR^2*Z;
K_q=K_q-1.83491e-3*J*PD*Z^2;
K_q=K_q+1.12451e-4*J^3*PD^2*Z^2;
K_q=K_q-2.97228e-5*J^3*PD^6*Z^2;
K_q=K_q+2.69551e-4*J*EAR*Z^2;
K_q=K_q+8.3265e-4*J^2*EAR*Z^2;
K_q=K_q+1.55334e-3*PD^2*EAR*Z^2;
K_q=K_q+3.02683e-4*PD^6*EAR*Z^2;
K_q=K_q-1.843e-4*EAR^2*Z^2;
K_q=K_q-4.25399e-4*PD^3*EAR^2*Z^2;
K_q=K_q+8.69243e-5*J^3*PD^3*EAR^2*Z^2;
K_q=K_q-4.659e-4*PD^6*EAR^2*Z^2;
K_q=K_q+5.54194e-5*J*PD^6*EAR^2*Z^2;
```

```
% Open Water Efficiency calculation
n0 = J*K_t/(2*pi*K_q);

J1 = -n0;

end
```

Function *sqppropcon2.m*:

```
function [c,ceq] = sqppropcon2(x)
% Function is used with propeller SQP optimization
% Represents constraints for the optimization problem

global rho Thrust Va Z n_shafts g h w t R_dec dv k1_plus_1 k2_plus_1 S_app
S C_b B T Rtotal L k_visc c_a

PD = x(1); % P/D
rps = x(2); % Revolution of Propeller per Second [1/sec]
D = x(3); % Propeller diameter [m]
EAR = x(4); % Expanded Area Ratio


if R_dec == 2
    k_plus_1 = k1_plus_1+(k2_plus_1-k1_plus_1)*S_app/(S+S_app);
    Re = L*dv/k_visc; % Reynolds number
    c_f = .075/(log10(Re)-2)^2;
    c_v = k_plus_1*c_f+c_a;
    w = .3095*C_b+10*c_v*C_b-.23*D/(B*T)^.5;
    t = .325*C_b-.1885*D/(B*T)^.5;
    Va = dv*(1-w); % Speed of advance [m/sec]
    Thrust = 1000*Rtotal(end)/(1-t)/n_shafts; % Thrust per shaft [N],
Rtotal is in [kN]
end


J = Va/(rps*D); % Advance coefficient

% K_t regression equation implementation
K_t=8.80496e-3;
K_t=K_t-.204554*J;
K_t=K_t+.166351*PD;
K_t=K_t+.158114*PD^2;
K_t=K_t-.147581*J^2*EAR;
K_t=K_t-.481497*J*PD*EAR;
K_t=K_t+.415437*PD^2*EAR;
K_t=K_t+.0144043*Z;
K_t=K_t-.0530054*J^2*Z;
K_t=K_t+.0143481*PD*Z;
K_t=K_t+.0606826*J*PD*Z;
K_t=K_t-.0125894*EAR*Z;
K_t=K_t+.0109689*J*EAR*Z;
K_t=K_t-.133698*PD^3;
K_t=K_t+6.38407e-3*PD^6;
K_t=K_t-1.32718e-3*J^2*PD^6;
K_t=K_t+.168496*J^3*EAR;
K_t=K_t-.0507214*EAR^2;
K_t=K_t+.0854559*J^2*EAR^2;
K_t=K_t-.0504475*J^3*EAR^2;
K_t=K_t+.010465*J*PD^6*EAR^2;
K_t=K_t-6.48272e-3*J^2*PD^6*EAR^2;
K_t=K_t-8.417281e-3*PD^3*Z;
K_t=K_t+.0168424*J*PD^3*Z;
K_t=K_t-1.02296e-3*J^3*PD^3*Z;
```

```
K_t=K_t-.0317791*PD^3*EAR*Z;
K_t=K_t+.018604*J*EAR^2*Z;
K_t=K_t-4.10798e-3*PD^2*EAR^2*Z;
K_t=K_t-6.06848e-4*Z^2;
K_t=K_t-4.9819e-3*J*Z^2;
K_t=K_t+2.5983e-3*J^2*Z^2;
K_t=K_t-5.60528e-4*J^3*Z^2;
K_t=K_t-1.63652e-3*J*PD^2*Z^2;
K_t=K_t-3.28787e-4*J*PD^6*Z^2;
K_t=K_t+1.16502e-4*J^2*PD^6*Z^2;
K_t=K_t+6.90904e-4*EAR*Z^2;
K_t=K_t+4.21749e-3*PD^3*EAR*Z^2;
K_t=K_t+5.65229e-5*J^3*PD^6*EAR*Z^2;
K_t=K_t-1.46564e-3*PD^3*EAR^2*Z^2;

% Equality constraint
ceq = K_t-Thrust/(rho*D^4*rps^2);



% Inequality constraint
c = (1.3+0.3*Z)*Thrust/(99047+rho*g*h)/D^2-EAR;

end
```

Function *Kt.m*:

```
function K_t = Kt(x)
% Function calculates thrust coefficient for input J(advance coefficient)

global rho Z EAR PD D t w v_iter R_iter

J=x;

% K_t regression equation implementation
K_t=8.80496e-3;
K_t=K_t-.204554*J;
K_t=K_t+.166351*PD;
K_t=K_t+.158114*PD^2;
K_t=K_t-.147581*J^2*EAR;
K_t=K_t-.481497*J*PD*EAR;
K_t=K_t+.415437*PD^2*EAR;
K_t=K_t+.0144043*Z;
K_t=K_t-.0530054*J^2*Z;
K_t=K_t+.0143481*PD*Z;
K_t=K_t+.0606826*J*PD*Z;
K_t=K_t-.0125894*EAR*Z;
K_t=K_t+.0109689*J*EAR*Z;
K_t=K_t-.133698*PD^3;
K_t=K_t+6.38407e-3*PD^6;
K_t=K_t-1.32718e-3*J^2*PD^6;
K_t=K_t+.168496*J^3*EAR;
K_t=K_t-.0507214*EAR^2;
K_t=K_t+.0854559*J^2*EAR^2;
K_t=K_t-.0504475*J^3*EAR^2;
K_t=K_t+.010465*J*PD^6*EAR^2;
K_t=K_t-6.48272e-3*J^2*PD^6*EAR^2;
K_t=K_t-8.417281e-3*PD^3*Z;
K_t=K_t+.0168424*J*PD^3*Z;
K_t=K_t-1.02296e-3*J^3*PD^3*Z;
K_t=K_t-.0317791*PD^3*EAR*Z;
K_t=K_t+.018604*J*EAR^2*Z;
K_t=K_t-4.10798e-3*PD^2*EAR^2*Z;
K_t=K_t-6.06848e-4*Z^2;
K_t=K_t-4.9819e-3*J*Z^2;
K_t=K_t+2.5983e-3*J^2*Z^2;
K_t=K_t-5.60528e-4*J^3*Z^2;
K_t=K_t-1.63652e-3*J*PD^2*Z^2;
K_t=K_t-3.28787e-4*J*PD^6*Z^2;
K_t=K_t+1.16502e-4*J^2*PD^6*Z^2;
K_t=K_t+6.90904e-4*EAR*Z^2;
K_t=K_t+4.21749e-3*PD^3*EAR*Z^2;
K_t=K_t+5.65229e-5*J^3*PD^6*EAR*Z^2;
K_t1=K_t-1.46564e-3*PD^3*EAR^2*Z^2;

K_t2 = R_iter/(2*rho*(1-t)*(1-w)^2*D^2*v_iter^2)*J^2;

K_t = K_t1 - K_t2;
```

Function *OpenWaterEff.m*:

```
function n0 = OpenWaterEff(x)
% Function is used with propeller SQP optimization
% Represents objective function for the optimization problem

global Z EAR PD

% Function input variables
J = x;

% K_t regression equation implementation
K_t=8.80496e-3;
K_t=K_t-.204554*J;
K_t=K_t+.166351*PD;
K_t=K_t+.158114*PD^2;
K_t=K_t-.147581*J^2*EAR;
K_t=K_t-.481497*J*PD*EAR;
K_t=K_t+.415437*PD^2*EAR;
K_t=K_t+.0144043*Z;
K_t=K_t-.0530054*J^2*Z;
K_t=K_t+.0143481*PD*Z;
K_t=K_t+.0606826*J*PD*Z;
K_t=K_t-.0125894*EAR*Z;
K_t=K_t+.0109689*J*EAR*Z;
K_t=K_t-.133698*PD^3;
K_t=K_t+6.38407e-3*PD^6;
K_t=K_t-1.32718e-3*J^2*PD^6;
K_t=K_t+.168496*J^3*EAR;
K_t=K_t-.0507214*EAR^2;
K_t=K_t+.0854559*J^2*EAR^2;
K_t=K_t-.0504475*J^3*EAR^2;
K_t=K_t+.010465*J*PD^6*EAR^2;
K_t=K_t-6.48272e-3*J^2*PD^6*EAR^2;
K_t=K_t-8.417281e-3*PD^3*Z;
K_t=K_t+.0168424*J*PD^3*Z;
K_t=K_t-1.02296e-3*J^3*PD^3*Z;
K_t=K_t-.0317791*PD^3*EAR*Z;
K_t=K_t+.018604*J*EAR^2*Z;
K_t=K_t-4.10798e-3*PD^2*EAR^2*Z;
K_t=K_t-6.06848e-4*Z^2;
K_t=K_t-4.9819e-3*J*Z^2;
K_t=K_t+2.5983e-3*J^2*Z^2;
K_t=K_t-5.60528e-4*J^3*Z^2;
K_t=K_t-1.63652e-3*J*PD^2*Z^2;
K_t=K_t-3.28787e-4*J*PD^6*Z^2;
K_t=K_t+1.16502e-4*J^2*PD^6*Z^2;
K_t=K_t+6.90904e-4*EAR*Z^2;
K_t=K_t+4.21749e-3*PD^3*EAR*Z^2;
K_t=K_t+5.65229e-5*J^3*PD^6*EAR*Z^2;
K_t=K_t-1.46564e-3*PD^3*EAR^2*Z^2;

% K_q regression equation implementation
K_q=3.79368e-3;
K_q=K_q+8.86523e-3*J^2;
K_q=K_q-.032241*J*PD;
K_q=K_q+3.44778e-3*PD^2;
```

```
K_q=K_q-.0408811*PD*EAR;
K_q=K_q-.108009*J*PD*EAR;
K_q=K_q-.0885381*J^2*PD*EAR;
K_q=K_q+.188561*PD^2*EAR;
K_q=K_q-3.70871e-3*J*Z;
K_q=K_q+5.13696e-3*PD*Z;
K_q=K_q+.0209449*J*PD*Z;
K_q=K_q+4.74319e-3*J^2*PD*Z;
K_q=K_q-7.23408e-3*J^2*EAR*Z;
K_q=K_q+4.38388e-3*J*PD*EAR*Z;
K_q=K_q-.0269403*PD^2*EAR*Z;
K_q=K_q+.0558082*J^3*EAR;
K_q=K_q+.0161886*PD^3*EAR;
K_q=K_q+3.18086e-3*J*PD^3*EAR;
K_q=K_q+.015896*EAR^2;
K_q=K_q+.0471729*J*EAR^2;
K_q=K_q+.0196283*J^3*EAR^2;
K_q=K_q-.0502782*PD*EAR^2;
K_q=K_q-.030055*J^3*PD*EAR^2;
K_q=K_q+.0417122*J^2*PD^2*EAR^2;
K_q=K_q-.0397722*PD^3*EAR^2;
K_q=K_q-3.50024e-3*PD^6*EAR^2;
K_q=K_q-.0106854*J^3*Z;
K_q=K_q+1.10903e-3*J^3*PD^3*Z;
K_q=K_q-3.13912e-4*PD^6*Z;
K_q=K_q+3.5985e-3*J^3*EAR*Z;
K_q=K_q-1.42121e-3*PD^6*EAR*Z;
K_q=K_q-3.83637e-3*J*EAR^2*Z;
K_q=K_q+.0126803*PD^2*EAR^2*Z;
K_q=K_q-3.18278e-3*J^2*PD^3*EAR^2*Z;
K_q=K_q+3.34268e-3*PD^6*EAR^2*Z;
K_q=K_q-1.83491e-3*J*PD*Z^2;
K_q=K_q+1.12451e-4*J^3*PD^2*Z^2;
K_q=K_q-2.97228e-5*J^3*PD^6*Z^2;
K_q=K_q+2.69551e-4*J*EAR*Z^2;
K_q=K_q+8.3265e-4*J^2*EAR*Z^2;
K_q=K_q+1.55334e-3*PD^2*EAR*Z^2;
K_q=K_q+3.02683e-4*PD^6*EAR*Z^2;
K_q=K_q-1.843e-4*EAR^2*Z^2;
K_q=K_q-4.25399e-4*PD^3*EAR^2*Z^2;
K_q=K_q+8.69243e-5*J^3*PD^3*EAR^2*Z^2;
K_q=K_q-4.659e-4*PD^6*EAR^2*Z^2;
K_q=K_q+5.54194e-5*J*PD^6*EAR^2*Z^2;

% Open Water Efficiency calculation
n0 = J*K_t/(2*pi*K_q);
```

Function *SFCind.m*:

```
function [ret_ind] = SFCind(eng_id,system)
%% Function chooses which sfc curve to follow depending on engine ID number

global propeng_type sseng_type ipseng_type

if strcmp(system,'prop') == 1
    type = propeng_type(eng_id);
    if strcmp(type,'DIESEL') == 1
        scf_ind = 3;
    else
        scf_ind = 5;
    end
elseif strcmp(system,'ss') == 1
    type = sseng_type(eng_id);
    if strcmp(type,'DIESEL') == 1
        scf_ind = 4;
    else
        scf_ind = 6;
    end
else
    type = ipseng_type(eng_id);
    if strcmp(type,'DIESEL') == 1
        scf_ind = 4;
    else
        scf_ind = 6;
    end
end

ret_ind = scf_ind;
```

Function *Lfraction.m*:

```
function [LF_out] = Lfraction(LF)
%% Function makes load fraction values integers and be over 5%
if LF < .05
    LF_out = 5; % Load fraction range 5-100%
elseif LF > 1
    LF_out = 100;
else
    LF_out = int8(100*LF); % Load fraction is integer
end
```

Function *Mfraction.m*:

```
function [MF_out] = Mfraction(MF)
%% Function makes motor load fraction values integers and be over 25%
if MF < .25
    MF_out = 25; % Motor load fraction range 25-100%
else
    MF_out = int8(100*MF); % Motor load fraction is integer
end
```

enginedata.m:

```
%% Engine data allocation
load enginedata

propeng_sfc = propengdata(:,6);
propeng_power = propengdata(:,5);
propeng_name = propengtextdata(2:end,2);
propeng_manuf = propengtextdata(2:end,3);
propeng_type = propengtextdata(2:end,4);

sseng_sfc = ssengdata(:,6);
sseng_power = ssengdata(:,5);
sseng_name = ssengtextdata(2:end,2);
sseng_manuf = ssengtextdata(2:end,3);
sseng_type = ssengtextdata(2:end,4);

ipseng_sfc = ipsengdata(:,6);
ipseng_power = ipsengdata(:,5);
ipseng_name = ipsengtextdata(2:end,2);
ipseng_manuf = ipsengtextdata(2:end,3);
ipseng_type = ipsengtextdata(2:end,4);

clear propengdata propengtextdata ssengdata ssengtextdata ipsengdata
ipsengtextdata

save detailedengdata
```

Function *Convert.m*:

```
function [ output ] = Convert( input_string, position, number_of_bits )
%% Takes a binary row vector (input_string) as input and converts it to its
decimal equivalent
% position is where in input to begin the conversion
% number_of_bits is the number of bits starting at position to convert
% The decimal equivalent is output in the parameter output
output=0;
% Extract from the total string the individual bits that are
% needed_
portion = input_string(position:position+number_of_bits-1);
% Compact the string so that the spaces between binary digits are removed_
s = strrep(int2str(portion),' ','');_
% Create in output the actual decimal equivalent
j=1;_L=length(s);
while(j<=L)
    output=output * 2 + bin2dec(s(j));
    j=j+1;
end
end
```

Function *MechSepf.m*:

```
function f = MechSepSSf(binarystr)
%% Function evaluates mechanical driven propulsion systems with separate
ship service systems

global profile_data dspeed dss SHP_1 gen_sfc_FL partial_sfc LHV_correction
loss_mult M_s...
    propeng_type sseng_type propeng_sfc sseng_sfc propeng_power sseng_power


%% Propulsion engines type and number selection
eng_id_cr = 1+Convert( binarystr, 1, 6 ); % Cruise engine selection ID
eng_id_sp = 1+Convert( binarystr, 7, 6 ); % Sprint engine selection ID
eng_num_cr = 1+Convert( binarystr, 13, 1 ); % Number of cruise engines

% Determine number of sprint engines
if eng_num_cr == 1
    eng_num_sp = 2;
else
    eng_num_sp = 1+Convert( binarystr, 14, 1 );
end

comb_id = Convert( binarystr, 15, 1 ); % OR/AND (0/1)


%% Mechanical driven propulsion system identification
% CODAG systems represent cross-connected configurations with 3 prime
movers
if strcmp(propeng_type(eng_id_cr),'DIESEL')==1 &&
strcmp(propeng_type(eng_id_sp),'DIESEL')==1 && eng_num_cr == eng_num_sp
    sys_type = 'CODAD';
elseif strcmp(propeng_type(eng_id_cr),'DIESEL')==1 &&
strcmp(propeng_type(eng_id_sp),'DIESEL')==0 && eng_num_cr == eng_num_sp &&
comb_id == 0
    sys_type = 'CODOG';
elseif strcmp(propeng_type(eng_id_cr),'DIESEL')==1 &&
strcmp(propeng_type(eng_id_sp),'DIESEL')==0 && eng_num_cr == eng_num_sp &&
comb_id == 1
    sys_type = 'CODAG';
elseif strcmp(propeng_type(eng_id_cr),'DIESEL')==0 &&
strcmp(propeng_type(eng_id_sp),'DIESEL')==0 && eng_num_cr == eng_num_sp
    sys_type = 'COGAG';
else
    sys_type = 'Cross-connected';
end


%% Propulsive power availability
% All configurations
usable_HP_cr = eng_num_cr*propeng_power(eng_id_cr); % [kW]
usable_HP_sp = (1-
strcmp(sys_type,'CODOG'))*usable_HP_cr+eng_num_sp*propeng_power(eng_id_sp);
% [kW], Only sprint engines if CODOG

% Cross-connected configurations
usable_cross_work = zeros(eng_num_cr+eng_num_sp,1);
```

```
if strcmp(sys_type,'Cross-connected') == 1
    for i = 1:eng_num_cr;
    usable_cross_work(i) = propeng_power(eng_id_cr);
    end
    for i = eng_num_cr+1:eng_num_cr+eng_num_sp
        usable_cross_work(i) = propeng_power(eng_id_sp);
    end
end
clear i


%% Mechanical transmission efficiencies
% Full-load transmission efficiencies
% GT transmission losses bigger due to 2 stage reduction required
if strcmp(sys_type,'Cross-connected')
    trans_mult = .99;
else
    trans_mult = 1;
end

if strcmp(propeng_type(eng_id_cr),'GT') == 1
    n_trans_cr = 0.98*trans_mult;
else
    n_trans_cr = 0.99*trans_mult;
end

if strcmp(propeng_type(eng_id_sp),'GT') == 1
    n_trans_sp = 0.98*trans_mult;
else
    n_trans_sp = 0.99*trans_mult;
end


%% Propulsion system fuel consumption calculations
penalty_prop = 0;
i_speed = length(profile_data(2:end-1,1));
if SHP_1(dspeed+1) > usable_HP_sp*n_trans_sp*(1-M_s) % Propulsion plant
provides required HP to achieve maximum sustained speed
    work_fc_prop = propeng_sfc(eng_id_sp)*usable_HP_sp*ones(i_speed,1); %
Assumes working fuel consumption is fuel consumption at max power
    penalty_prop = .5*(SHP_1(dspeed+1)-usable_HP_sp*n_trans_sp*(1-M_s)); %
Penalty for fitness function
else
    % Pre-allocate
    prop_power = zeros(i_speed,1);
    work_fc_prop = zeros(i_speed,1);
    system = 'prop';
    for i = 1:i_speed
        prop_power(i) = SHP_1(i+1); % SHP_1 starts with speed=0, prop_power
starts with speed=1knot
        % load_fraction_2 expresses increased transmission losses at
partial load
        if strcmp(sys_type,'Cross-connected') == 0
            if prop_power(i) < usable_HP_cr*n_trans_cr
                sfc_ind = SFCind(eng_id_cr,system);
                load_fraction = prop_power(i)/(usable_HP_cr*n_trans_cr);
                sfc_fraction = Lfraction(load_fraction);
```

```
                load_fraction2 = n_trans_cr/(1-(1-
n_trans_cr)*loss_mult(sfc_fraction-4))*load_fraction;
                work_sfc =
propeng_sfc(eng_id_cr)/gen_sfc_FL(sfc_ind)*partial_sfc(sfc_ind,sfc_fraction
-4)*LHV_correction;
                work_fc_prop(i) = load_fraction2*usable_HP_cr*work_sfc; %
[gr/hr]

            else
                if strcmp(sys_type,'CODOG') == 1
                    load_fraction =
prop_power(i)/(usable_HP_sp*n_trans_sp);
                    sfc_fraction = Lfraction(load_fraction);
                    load_fraction2 = n_trans_sp/(1-(1-
n_trans_sp)*loss_mult(sfc_fraction-4))*load_fraction;
                    work_sfc =
propeng_sfc(eng_id_sp)/gen_sfc_FL(5)*partial_sfc(5,sfc_fraction-
4)*LHV_correction;
                    work_fc_prop(i) = load_fraction2*usable_HP_sp*work_sfc;
% [gr/hr]
                else % CODAD, CODAG or COGAG
                    % Sprint engines at partial load, cruise engines
                    % at 100% load
                    sfc_ind = SFCind(eng_id_sp,system);
                    load_fraction = (prop_power(i)-
usable_HP_cr*n_trans_cr)/(usable_HP_sp*n_trans_sp);
                    sfc_fraction = Lfraction(load_fraction);
                    load_fraction2 = n_trans_sp/(1-(1-
n_trans_sp)*loss_mult(sfc_fraction-4))*load_fraction;
                    work_sfc =
propeng_sfc(eng_id_sp)/gen_sfc_FL(sfc_ind)*partial_sfc(sfc_ind,sfc_fraction
-4)*LHV_correction;
                    work_fc_prop(i) =
propeng_sfc(eng_id_cr)*usable_HP_cr+load_fraction2*usable_HP_sp*work_sfc; %
[gr/hr]
                end
            end
        else % Cross-connected configurations
            if eng_num_cr == 1 % 1 cruise engine and 2 sprint engines
                if prop_power(i) < usable_cross_work(1)*n_trans_cr
                    % Cruise engine at partial load
                    sfc_ind = SFCind(eng_id_cr,system);
                    load_fraction =
prop_power(i)/(usable_cross_work(1)*n_trans_cr);
                    sfc_fraction = Lfraction(load_fraction);
                    load_fraction2 = n_trans_cr/(1-(1-
n_trans_cr)*loss_mult(sfc_fraction-4))*load_fraction;
                    work_sfc =
propeng_sfc(eng_id_cr)/gen_sfc_FL(sfc_ind)*partial_sfc(sfc_ind,sfc_fraction
-4)*LHV_correction;
                    work_fc_prop(i) =
load_fraction2*usable_cross_work(1)*work_sfc; % [gr/hr]
                elseif prop_power(i) < usable_cross_work(2)*n_trans_sp
                    % 1st sprint engine at partial load
                    sfc_ind = SFCind(eng_id_sp,system);
                    load_fraction =
prop_power(i)/(usable_cross_work(2)*n_trans_sp);
                    sfc_fraction = Lfraction(load_fraction);
```

```
                    load_fraction2 = n_trans_sp/(1-(1-
n_trans_sp)*loss_mult(sfc_fraction-4))*load_fraction;
                    work_sfc =
propeng_sfc(eng_id_sp)/gen_sfc_FL(sfc_ind)*partial_sfc(sfc_ind,sfc_fraction
-4)*LHV_correction;
                    work_fc_prop(i) =
load_fraction2*usable_cross_work(2)*work_sfc; % [gr/hr]
                elseif prop_power(i) <
usable_cross_work(1)*n_trans_cr+usable_cross_work(2)*n_trans_sp
                    % Cruise engine at full load, 1st sprint engine
                    %  at partial load
                    sfc_ind = SFCind(eng_id_sp,system);
                    load_fraction = (prop_power(i)-
usable_cross_work(1)*n_trans_cr)/(usable_cross_work(2)*n_trans_sp);
                    sfc_fraction = Lfraction(load_fraction);
                    load_fraction2 = n_trans_sp/(1-(1-
n_trans_sp)*loss_mult(sfc_fraction-4))*load_fraction;
                    work_sfc =
propeng_sfc(eng_id_sp)/gen_sfc_FL(sfc_ind)*partial_sfc(sfc_ind,sfc_fraction
-4)*LHV_correction;
                    work_fc_prop(i) =
propeng_sfc(eng_id_cr)*usable_cross_work(1)+load_fraction2*usable_cross_wor
k(2)*work_sfc; % [gr/hr]
                elseif prop_power(i) <
sum(usable_cross_work(2:3))*n_trans_sp
                    % 1st sprint engine at full load, 2nd sprint engine
                    % at partial load
                    sfc_ind = SFCind(eng_id_sp,system);
                    load_fraction = (prop_power(i)-
usable_cross_work(2)*n_trans_sp)/(usable_cross_work(3)*n_trans_sp);
                    sfc_fraction = Lfraction(load_fraction);
                    load_fraction2 = n_trans_sp/(1-(1-
n_trans_sp)*loss_mult(sfc_fraction-4))*load_fraction;
                    work_sfc =
propeng_sfc(eng_id_sp)/gen_sfc_FL(sfc_ind)*partial_sfc(sfc_ind,sfc_fraction
-4)*LHV_correction;
                    work_fc_prop(i) =
propeng_sfc(eng_id_sp)*usable_cross_work(2)+load_fraction2*usable_cross_wor
k(3)*work_sfc; % [gr/hr]
                else
                    % Cruise engine and 1st sprint engine at full load, 2nd
                    % sprint engine at partial load
                    sfc_ind = SFCind(eng_id_sp,system);
                    load_fraction = (prop_power(i)-
usable_cross_work(1)*n_trans_cr-
usable_cross_work(2)*n_trans_sp)/(usable_cross_work(3)*n_trans_sp);
                    sfc_fraction = Lfraction(load_fraction);
                    load_fraction2 = n_trans_sp/(1-(1-
n_trans_sp)*loss_mult(sfc_fraction-4))*load_fraction;
                    work_sfc =
propeng_sfc(eng_id_sp)/gen_sfc_FL(sfc_ind)*partial_sfc(sfc_ind,sfc_fraction
-4)*LHV_correction;
                    work_fc_prop(i) =
propeng_sfc(eng_id_cr)*usable_cross_work(1)+propeng_sfc(eng_id_sp)*usable_c
ross_work(2)+load_fraction2*usable_cross_work(3)*work_sfc; % [gr/hr]
                end
            else % 2 cruise engines and 1 sprint engine
                if prop_power(i) < usable_cross_work(1)*n_trans_cr
```

```
                    % Cruise engine at partial load
                    sfc_ind = SFCind(eng_id_cr,system);
                    load_fraction =
prop_power(i)/(usable_cross_work(1)*n_trans_cr);
                    sfc_fraction = Lfraction(load_fraction);
                    load_fraction2 = n_trans_cr/(1-(1-
n_trans_cr)*loss_mult(sfc_fraction-4))*load_fraction;
                    work_sfc =
propeng_sfc(eng_id_cr)/gen_sfc_FL(sfc_ind)*partial_sfc(sfc_ind,sfc_fraction
-4)*LHV_correction;
                    work_fc_prop(i) =
load_fraction2*usable_cross_work(1)*work_sfc; % [gr/hr]
                elseif prop_power(i) <
sum(usable_cross_work(1:2))*n_trans_cr
                    % 1st cruise engine at full load, 2nd cruise engine at
                    % partial load
                    sfc_ind = SFCind(eng_id_cr,system);
                    load_fraction = (prop_power(i)-
usable_cross_work(1)*n_trans_cr)/(usable_cross_work(2)*n_trans_cr);
                    sfc_fraction = Lfraction(load_fraction);
                    load_fraction2 = n_trans_cr/(1-(1-
n_trans_cr)*loss_mult(sfc_fraction-4))*load_fraction;
                    work_sfc =
propeng_sfc(eng_id_cr)/gen_sfc_FL(sfc_ind)*partial_sfc(sfc_ind,sfc_fraction
-4)*LHV_correction;
                    work_fc_prop(i) =
propeng_sfc(eng_id_cr)*usable_cross_work(1)+load_fraction2*usable_cross_wor
k(2)*work_sfc; % [gr/hr]
                elseif prop_power(i) < usable_cross_work(3)*n_trans_sp
                    % Sprint engine at partial load
                    sfc_ind = SFCind(eng_id_sp,system);
                    load_fraction =
prop_power(i)/(usable_cross_work(3)*n_trans_sp);
                    sfc_fraction = Lfraction(load_fraction);
                    load_fraction2 = n_trans_sp/(1-(1-
n_trans_sp)*loss_mult(sfc_fraction-4))*load_fraction;
                    work_sfc =
propeng_sfc(eng_id_sp)/gen_sfc_FL(sfc_ind)*partial_sfc(sfc_ind,sfc_fraction
-4)*LHV_correction;
                    work_fc_prop(i) =
load_fraction2*usable_cross_work(3)*work_sfc; % [gr/hr]
                elseif prop_power(i) <
usable_cross_work(1)*n_trans_cr+usable_cross_work(3)*n_trans_sp
                    % 1st cruise engine at partial load, sprint engine
                    % at 100% load
                    sfc_ind = SFCind(eng_id_sp,system);
                    load_fraction = (prop_power(i)-
usable_cross_work(1)*n_trans_cr)/(usable_cross_work(3)*n_trans_sp);
                    sfc_fraction = Lfraction(load_fraction);
                    load_fraction2 = n_trans_sp/(1-(1-
n_trans_sp)*loss_mult(sfc_fraction-4))*load_fraction;
                    work_sfc =
propeng_sfc(eng_id_sp)/gen_sfc_FL(sfc_ind)*partial_sfc(sfc_ind,sfc_fraction
-4)*LHV_correction;
                    work_fc_prop(i) =
propeng_sfc(eng_id_cr)*usable_cross_work(1)+load_fraction2*usable_cross_wor
k(3)*work_sfc; % [gr/hr]
                else
```

```
                    % Both ruise engines at full load,
                    % sprint engine at partial load
                    sfc_ind = SFCind(eng_id_sp,system);
                    load_fraction = (prop_power(i)-
sum(usable_cross_work(1:2))*n_trans_cr)/(usable_cross_work(3)*n_trans_sp);
                    sfc_fraction = Lfraction(load_fraction);
                    load_fraction2 = n_trans_sp/(1-(1-
n_trans_sp)*loss_mult(sfc_fraction-4))*load_fraction;
                    work_sfc =
propeng_sfc(eng_id_sp)/gen_sfc_FL(sfc_ind)*partial_sfc(sfc_ind,sfc_fraction
-4)*LHV_correction;
                    work_fc_prop(i) =
propeng_sfc(eng_id_cr)*sum(usable_cross_work(1:2))+load_fraction2*usable_cr
oss_work(3)*work_sfc; % [gr/hr]
                end
            end
        end
    end
end


%% Ship service engine type and number selection
ss_id = 1+Convert( binarystr, 16, 6 ); % Ship service engine selection ID
ss_num = 3+Convert( binarystr, 22, 1 ); % Number of ship service engines (3
or 4 allowed)

%% Ship service power and efficiency calculations

n_gen = 0.96;
system = 'ss';
sfc_ss_ind = SFCind(ss_id,system);

c_imb = .9; % Load factor for imbalances of electrical loads on individual
ship service systems

% Ship service power availability for different number of engines on-line
% N-1 engines
usable_HP_ss_work = zeros(ss_num-1,1); % 1st col=number working, 2nd
col=usable HP
for i_gen = 1:ss_num-1
    usable_HP_ss_work(i_gen) = i_gen*sseng_power(ss_id);
end


%% Ship service fuel consumption calculations
penalty_ss = 0;
% On-line engines equally share load
i_sspower = length(profile_data(1,2:end-2));

if dss > usable_HP_ss_work(end)*n_gen*c_imb % N-1 ss engines meeting max
demand
    work_fc_ss =
propeng_sfc(ss_id)*usable_HP_ss_work(end)*ones(1,i_sspower); % Assume
working fuel consumption is fuel consumption at max ss power
    penalty_ss = .5*(dss-usable_HP_ss_work(end)*n_gen*c_imb); % Penalty for
fitness function
```

```
else
    % Pre-allocate
    ss_power = profile_data(1,2:end-2); % Required ss power
    work_fc_ss = zeros(1,i_sspower);
    ss_num_work = zeros(1,i_sspower);
    if ss_num == 3 % 2 generators on line should meet the required ss power
profile
        for j = 1:i_sspower
            ss_num_work(j) = 2;
            load_fraction = ss_power(j)/(usable_HP_ss_work(2)*n_gen);
            sfc_fraction = Lfraction(load_fraction);
            work_sfc_ss =
sseng_sfc(ss_id)/gen_sfc_FL(sfc_ss_ind)*partial_sfc(sfc_ss_ind,sfc_fraction
-4)*LHV_correction;
            work_fc_ss(j) = load_fraction*usable_HP_ss_work(2)*work_sfc_ss;
% [gr/hr]
        end
    elseif ss_num == 4 % 2 or 3 generators on line should meet the required
ss power profile
        for j = 1:i_sspower
            if ss_power(j) < usable_HP_ss_work(2)*n_gen*c_imb % 2 engines
on line meet the required ss power profile
                ss_num_work(j) = 2;
                load_fraction = ss_power(j)/(usable_HP_ss_work(2)*n_gen);
                sfc_fraction = Lfraction(load_fraction);
                work_sfc_ss =
sseng_sfc(ss_id)/gen_sfc_FL(sfc_ss_ind)*partial_sfc(sfc_ss_ind,sfc_fraction
-4)*LHV_correction;
                work_fc_ss(j) =
load_fraction*usable_HP_ss_work(2)*work_sfc_ss; % [gr/hr]
            else % 3 engines on line meet the required ss power profile
                ss_num_work(j) = 3;
                load_fraction = ss_power(j)/(usable_HP_ss_work(3)*n_gen);
                sfc_fraction = Lfraction(load_fraction);
                work_sfc_ss =
sseng_sfc(ss_id)/gen_sfc_FL(sfc_ss_ind)*partial_sfc(sfc_ss_ind,sfc_fraction
-4)*LHV_correction;
                work_fc_ss(j) =
load_fraction*usable_HP_ss_work(3)*work_sfc_ss; % [gr/hr]
            end
        end
    end
end


%% Total fuel consumption for the given operating profile
fc_total_prop = sum(profile_data(2:end-1,end).*work_fc_prop/1000)/100; %
Propulsion fuel consumption [kg/hr]
fc_total_ss = sum(profile_data(end,2:end-2).*work_fc_ss/1000)/100; % Ship
service fuel consumption [kg/hr]
fc_total = fc_total_prop+fc_total_ss; % Total fuel consumption [kg/hr]

f = fc_total+penalty_prop+penalty_ss; % Objective function (minimize)
```

Function *MechPDSSf.m*:

```
function f = MechPDSSf(binarystr)
%% Function evaluates mechanical-driven propulsion systems with propulsion
derived ship-service power
% Separate ship service systems can still be present

global profile_data dspeed dss SHP_2 gen_sfc_FL partial_sfc LHV_correction
loss_mult M_s...
    propeng_type sseng_type propeng_sfc sseng_sfc propeng_power sseng_power


%% Propulsion engines type and number selection
eng_id_cr = 1+Convert( binarystr, 1, 6 ); % Cruise engine selection ID
eng_id_sp = 1+Convert( binarystr, 7, 6 ); % Sprint engine selection ID

if propeng_power(eng_id_sp) < propeng_power(eng_id_cr)
    temp = eng_id_cr;
    eng_id_cr = eng_id_sp;
    eng_id_sp = temp;
end

eng_num_cr = 1+Convert( binarystr, 13, 1 ); % Number of cruise engines

% Determine number of sprint engines
if eng_num_cr == 1
    eng_num_sp = 2;
else
    eng_num_sp = 1+Convert( binarystr, 14, 1 );
end

comb_id = Convert( binarystr, 15, 1 ); % OR/AND (0/1)

%% Mechanical driven propulsion system identification
% CODAG systems represent cross-connected configurations with 3 prime
movers
if strcmp(propeng_type(eng_id_cr),'DIESEL')==1 &&
strcmp(propeng_type(eng_id_sp),'DIESEL')==1 && eng_num_cr == eng_num_sp
    sys_type = 'CODAD';
elseif strcmp(propeng_type(eng_id_cr),'DIESEL')==1 &&
strcmp(propeng_type(eng_id_sp),'DIESEL')==0 && eng_num_cr == eng_num_sp &&
comb_id == 0
    sys_type = 'CODOG';
elseif strcmp(propeng_type(eng_id_cr),'DIESEL')==1 &&
strcmp(propeng_type(eng_id_sp),'DIESEL')==0 && eng_num_cr == eng_num_sp &&
comb_id == 1
    sys_type = 'CODAG';
elseif strcmp(propeng_type(eng_id_cr),'DIESEL')==0 &&
strcmp(propeng_type(eng_id_sp),'DIESEL')==0 && eng_num_cr == eng_num_sp
    sys_type = 'COGAG';
else
    sys_type = 'Cross-connected';
end
```

```
%% Propulsive power availability
% All configurations
usable_HP_cr = eng_num_cr*propeng_power(eng_id_cr); % [kW]
usable_HP_sp = (1-
strcmp(sys_type,'CODOG'))*usable_HP_cr+eng_num_sp*propeng_power(eng_id_sp);
% [kW], Only sprint engines if CODOG

% Cross-connected configurations
usable_cross_work = zeros(eng_num_cr+eng_num_sp,1);
if strcmp(sys_type,'Cross-connected') == 1
    for i_cross = 1:eng_num_cr;
    usable_cross_work(i_cross) = propeng_power(eng_id_cr);
    end
    for i_cross = eng_num_cr+1:eng_num_cr+eng_num_sp
        usable_cross_work(i_cross) = propeng_power(eng_id_sp);
    end
end


%% Mechanical transmission efficiencies
% Full-load transmission efficiencies
% GT transmission losses bigger due to 2 stage reduction required
if strcmp(sys_type,'Cross-connected')
    trans_mult = .99;
else
    trans_mult = 1;
end

if strcmp(propeng_type(eng_id_cr),'GT') == 1
    n_trans_cr = 0.98*trans_mult;
else
    n_trans_cr = 0.99*trans_mult;
end

if strcmp(propeng_type(eng_id_sp),'GT') == 1
    n_trans_sp = 0.98*trans_mult;
else
    n_trans_sp = 0.99*trans_mult;
end


%% Separate ship service engine type and number selection if present
sep_ss_dec = Convert( binarystr, 16, 1 ); % Separate ship service system:
0/1 = No/Yes
sep_ss_id = 1+Convert( binarystr, 17, 6 ); % Separate ship service engine
selection ID
sep_ss_num = 2+Convert( binarystr, 23, 1 ); % Number of separate ship
service engines (2 or 3 allowed)

%% Ship service power and efficiency calculations
% Ship service power generation efficiencies
n_gen = 0.96;
system = 'ss';
sfc_ss_ind = SFCind(sep_ss_id,system);

% Ship service power availability for different number of engines on-line
```

```
% N-1 engines
usable_HP_ss_work = zeros(sep_ss_num-1,1);
usable_HP_ss_sep = 0;
if sep_ss_dec == 1
    for i_gen = 1:sep_ss_num-1
        usable_HP_ss_work(i_gen) = sseng_power(sep_ss_id);
    end
    usable_HP_ss_sep = sum(usable_HP_ss_work);
end


% Propulsion derived ship service efficiency
n_ss_pd = 0.915; % VSCF cycloconverter efficiency, ASSET value


%% Propulsion and propulsion derived ship service systems fuel consumption
calculations
% Load_fraction_2: Engine working load fraction (taking into account
transmission losses at partial load)
i_speed = length(profile_data(2:end-1,1));
i_sspower = length(profile_data(1,2:end-2));
% Check that propulsion and ship-service configuration provides the
required HP
if usable_HP_sp*(1-
M_s)*min([n_trans_cr,n_trans_sp])+usable_HP_ss_sep*n_ss_pd <
dss+SHP_2(dspeed+1) || usable_HP_sp*(1-M_s)*min([n_trans_cr,n_trans_sp]) <
SHP_2(dspeed+1)
    work_fc =
(propeng_sfc(eng_id_sp)*usable_HP_sp+propeng_sfc(sep_ss_id)*usable_HP_ss_se
p)*ones(i_speed,i_sspower);
    if usable_HP_sp*(1-
M_s)*min([n_trans_cr,n_trans_sp])+usable_HP_ss_sep*n_ss_pd <
dss+SHP_2(dspeed+1)
        penalty = 0.5*(dss+SHP_2(dspeed+1)-usable_HP_sp*(1-
M_s)*min([n_trans_cr,n_trans_sp])+usable_HP_ss_sep*n_ss_pd);
    elseif usable_HP_sp*(1-M_s)*min([n_trans_cr,n_trans_sp]) <
SHP_2(dspeed+1)
        penalty = 0.5*(SHP_2(dspeed+1)-usable_HP_sp*(1-
M_s)*min([n_trans_cr,n_trans_sp]));
    end
else
    penalty = 0;
    system = 'prop';
    % Pre-allocate
    prop_power = zeros(i_speed,1);
    ss_power = profile_data(1,2:end-2);
    work_fc = zeros(i_speed,i_sspower); % lin=prop, col=ss
    usable_HP_pd = zeros(i_speed,i_sspower); % Usable PD HP for each speed/
ss power demand combination
    for i = 1:i_speed
        prop_power(i) = SHP_2(i+1); % SHP_2 starts with speed=0, prop_power
starts with speed=1knot
        for j = 1:i_sspower
            if strcmp(sys_type,'Cross-connected') == 0
                if prop_power(i)/n_trans_cr+ss_power(j)/n_ss_pd <
usable_HP_cr
                    % Cruise engines at partial load
```

```
                    usable_HP_pd(i,j) = usable_HP_cr-
prop_power(i)/n_trans_cr;
                    sfc_ind = SFCind(eng_id_cr,system);
                    load_fraction =
(prop_power(i)/n_trans_cr+ss_power(j)/n_ss_pd)/usable_HP_cr;
                    sfc_fraction = Lfraction(load_fraction);
                    load_fraction2 = n_trans_cr/(1-(1-
n_trans_cr)*loss_mult(sfc_fraction-4))*load_fraction;
                    work_sfc =
propeng_sfc(eng_id_cr)/gen_sfc_FL(sfc_ind)*partial_sfc(sfc_ind,sfc_fraction
-4)*LHV_correction;
                    % Total working fuel consumption
                    work_fc(i,j) = load_fraction2*usable_HP_cr*work_sfc; %
[gr/hr]
                elseif prop_power(i)/n_trans_cr+ss_power(j)/n_ss_pd <
usable_HP_cr+usable_HP_ss_work(1) && prop_power(i)/n_trans_cr <
usable_HP_cr
                    % Cruise engines at full load, ss engine working at
partial load
                    usable_HP_pd(i,j) = usable_HP_cr-
prop_power(i)/n_trans_cr;
                    load_fraction = (ss_power(j)-
usable_HP_pd(i,j)*n_ss_pd)/n_gen/usable_HP_ss_work(1);
                    sfc_fraction = Lfraction(load_fraction);
                    work_sfc =
sseng_sfc(sep_ss_id)/gen_sfc_FL(sfc_ss_ind)*partial_sfc(sfc_ss_ind,sfc_frac
tion-4)*LHV_correction;
                    % Total working fuel consumption
                    work_fc(i,j) =
usable_HP_cr*propeng_sfc(eng_id_cr)+load_fraction*usable_HP_ss_work(1)*work
_sfc; % [gr/hr]
                elseif prop_power(i)/n_trans_cr+ss_power(j)/n_ss_pd <
usable_HP_cr+sum(usable_HP_ss_work) && prop_power(i)/n_trans_cr <
usable_HP_cr
                    % Cruise and 1st ss engines at full load, 2nd ss engine
at partial load
                    usable_HP_pd(i,j) = usable_HP_cr-
prop_power(i)/n_trans_cr;
                    load_fraction = (ss_power(j)-usable_HP_pd(i,j)*n_ss_pd-
usable_HP_ss_work(1)*n_gen)/n_gen/usable_HP_ss_work(2);
                    sfc_fraction = Lfraction(load_fraction);
                    work_sfc =
sseng_sfc(sep_ss_id)/gen_sfc_FL(sfc_ss_ind)*partial_sfc(sfc_ss_ind,sfc_frac
tion-4)*LHV_correction;
                    % Total working fuel consumption
                    work_fc(i,j) =
usable_HP_cr*propeng_sfc(eng_id_cr)+usable_HP_ss_work(1)*propeng_sfc(sep_ss
_id)+load_fraction*usable_HP_ss_work(2)*work_sfc; % [gr/hr]
                elseif prop_power(i)/n_trans_sp+ss_power(j)/n_ss_pd <
usable_HP_sp && prop_power(i)/n_trans_sp < usable_HP_sp
                    if strcmp(sys_type,'CODOG')
                        % Sprint engines at partial load
                        usable_HP_pd(i,j) = usable_HP_sp-
prop_power(i)/n_trans_sp;
                        load_fraction =
(prop_power(i)/n_trans_sp+ss_power(j)/n_ss_pd)/usable_HP_sp;
                        sfc_fraction = Lfraction(load_fraction);
                        load_fraction2 = n_trans_sp/(1-(1-
```

```
n_trans_sp)*loss_mult(sfc_fraction-4))*load_fraction;
                        work_sfc =
propeng_sfc(eng_id_sp)/gen_sfc_FL(5)*partial_sfc(5,sfc_fraction-
4)*LHV_correction;
                        % Total working fuel consumption
                        work_fc(i,j) =
load_fraction2*usable_HP_sp*work_sfc; % [gr/hr]
                    else
                        % Cruise engines at full load, sprint engines at
partial load
                        usable_HP_pd(i,j) = usable_HP_sp-
prop_power(i)/n_trans_sp;
                        sfc_ind = SFCind(eng_id_sp,system);
                        load_fraction = ((prop_power(i)-
usable_HP_cr*n_trans_cr)/n_trans_sp+ss_power(j)/n_ss_pd)/(usable_HP_sp-
usable_HP_cr);
                        sfc_fraction = Lfraction(load_fraction);
                        load_fraction2 = n_trans_sp/(1-(1-
n_trans_sp)*loss_mult(sfc_fraction-4))*load_fraction;
                        work_sfc =
propeng_sfc(eng_id_sp)/gen_sfc_FL(sfc_ind)*partial_sfc(sfc_ind,sfc_fraction
-4)*LHV_correction;
                        % Total working fuel consumption
                        work_fc(i,j) =
propeng_sfc(eng_id_cr)*usable_HP_cr+load_fraction2*usable_HP_sp*work_sfc; %
[gr/hr]
                    end
                elseif prop_power(i)/n_trans_sp+ss_power(j)/n_ss_pd <
usable_HP_sp+usable_HP_ss_work(1) && prop_power(i)/n_trans_sp <
usable_HP_sp
                    if strcmp(sys_type,'CODOG')
                        % Sprint engines at full load, ss engine at partial
load
                        usable_HP_pd(i,j) = usable_HP_sp-
prop_power(i)/n_trans_sp;
                        load_fraction = (ss_power(j)-
usable_HP_pd(i,j)*n_ss_pd)/n_gen/usable_HP_ss_work(1);
                        sfc_fraction = Lfraction(load_fraction);
                        work_sfc =
sseng_sfc(sep_ss_id)/gen_sfc_FL(sfc_ss_ind)*partial_sfc(sfc_ss_ind,sfc_frac
tion-4)*LHV_correction;
                        % Total working fuel consumption
                        work_fc(i,j) =
usable_HP_sp*propeng_sfc(eng_id_sp)+load_fraction*usable_HP_ss_work(1)*work
_sfc; % [gr/hr]
                    else
                        % Cruise and sprint engines at full load, ss engine
at partial load
                        usable_HP_pd(i,j) = usable_HP_cr*(n_trans_cr-
n_trans_sp)/n_trans_sp+usable_HP_sp-prop_power(i)/n_trans_sp;
                        load_fraction = (ss_power(j)-
usable_HP_pd(i,j)*n_ss_pd)/n_gen/usable_HP_ss_work(1);
                        sfc_fraction = Lfraction(load_fraction);
                        work_sfc =
sseng_sfc(sep_ss_id)/gen_sfc_FL(sfc_ss_ind)*partial_sfc(sfc_ss_ind,sfc_frac
tion-4)*LHV_correction;
                        % Total working fuel consumption
                        work_fc(i,j) =
```

```
usable_HP_sp*propeng_sfc(eng_id_sp)+(usable_HP_sp-
usable_HP_cr)*propeng_sfc(eng_id_cr)+load_fraction*usable_HP_ss_work(1)*wor
k_sfc; % [gr/hr]
                    end
                elseif prop_power(i)/n_trans_sp+ss_power(j)/n_ss_pd <
usable_HP_sp+sum(usable_HP_ss_work) && prop_power(i)/n_trans_sp <
usable_HP_sp
                    if strcmp(sys_type,'CODOG')
                        % Sprint and 1st ss engines at full load, 2nd ss
engine at partial load
                        usable_HP_pd(i,j) = usable_HP_sp-
prop_power(i)/n_trans_sp;
                        load_fraction = (ss_power(j)-
usable_HP_pd(i,j)*n_ss_pd-
usable_HP_ss_work(1)*n_gen)/n_gen/usable_HP_ss_work(2);
                        sfc_fraction = Lfraction(load_fraction);
                        work_sfc =
sseng_sfc(sep_ss_id)/gen_sfc_FL(sfc_ss_ind)*partial_sfc(sfc_ss_ind,sfc_frac
tion-4)*LHV_correction;
                        % Total working fuel consumption
                        work_fc(i,j) =
usable_HP_sp*propeng_sfc(eng_id_sp)+usable_HP_ss_work(1)*sseng_sfc(sep_ss_i
d)+load_fraction*usable_HP_ss_work(2)*work_sfc; % [gr/hr]
                    else
                        % Cruise, sprint and 1st ss engines at full load,
2nd ss engine at partial load
                        usable_HP_pd(i,j) = usable_HP_cr*(n_trans_cr-
n_trans_sp)/n_trans_sp+usable_HP_sp-prop_power(i)/n_trans_sp;
                        load_fraction = (ss_power(j)-
usable_HP_pd(i,j)*n_ss_pd-
usable_HP_ss_work(1)*n_gen)/n_gen/usable_HP_ss_work(2);
                        sfc_fraction = Lfraction(load_fraction);
                        work_sfc =
sseng_sfc(sep_ss_id)/gen_sfc_FL(sfc_ss_ind)*partial_sfc(sfc_ss_ind,sfc_frac
tion-4)*LHV_correction;
                        % Total working fuel consumption
                        work_fc(i,j) =
usable_HP_sp*propeng_sfc(eng_id_sp)+(usable_HP_sp-
usable_HP_cr)*propeng_sfc(eng_id_cr)+usable_HP_ss_work(1)*sseng_sfc(sep_ss_
id)+load_fraction*usable_HP_ss_work(2)*work_sfc; % [gr/hr]
                    end
                end
            else % Cross-connected configurations
                if eng_num_cr == 1 % 1 cruise engine and 2 sprint engines
                    if prop_power(i)/n_trans_cr+ss_power(j)/n_ss_pd <
usable_cross_work(1)
                        % Cruise engine at partial load
                        usable_HP_pd(i,j) = usable_cross_work(1)-
prop_power(i)/n_trans_cr;
                        sfc_ind = SFCind(eng_id_cr,system);
                        load_fraction =
(prop_power(i)/n_trans_cr+ss_power(j)/n_ss_pd)/usable_cross_work(1);
                        sfc_fraction = Lfraction(load_fraction);
                        load_fraction2 = n_trans_cr/(1-(1-
n_trans_cr)*loss_mult(sfc_fraction-4))*load_fraction;
                        work_sfc =
propeng_sfc(eng_id_cr)/gen_sfc_FL(sfc_ind)*partial_sfc(sfc_ind,sfc_fraction
-4)*LHV_correction;
```

```
                        % Working fuel consumption
                        work_fc(i,j) =
load_fraction2*usable_cross_work(1)*work_sfc; % [gr/hr]
                    elseif prop_power(i)/n_trans_cr+ss_power(j)/n_ss_pd <
usable_cross_work(1)+usable_HP_ss_work(1) && prop_power(i)/n_trans_cr <
usable_cross_work(1)
                        % Cruise engine at full load and 1st ss engine at
                        % partial load
                        usable_HP_pd(i,j) = usable_cross_work(1)-
prop_power(i)/n_trans_cr;
                        load_fraction = (ss_power(j)-
usable_HP_pd(i,j)*n_ss_pd)/n_gen/usable_HP_ss_work(1);
                        sfc_fraction = Lfraction(load_fraction);
                        work_sfc =
sseng_sfc(sep_ss_id)/gen_sfc_FL(sfc_ss_ind)*partial_sfc(sfc_ss_ind,sfc_frac
tion-4)*LHV_correction;
                        % Total working fuel consumption
                        work_fc(i,j) =
usable_cross_work(1)*propeng_sfc(eng_id_cr)+load_fraction*usable_HP_ss_work
(1)*work_sfc; % [gr/hr]
                    elseif prop_power(i)/n_trans_cr+ss_power(j)/n_ss_pd <
usable_cross_work(1)+sum(usable_HP_ss_work) && prop_power(i)/n_trans_cr <
usable_cross_work(1)
                        % Cruise and 1st ss engines at full load, 2nd ss
engine at partial load
                        usable_HP_pd(i,j) = usable_cross_work(1)-
prop_power(i)/n_trans_cr;
                        load_fraction = (ss_power(j)-
usable_HP_pd(i,j)*n_ss_pd-
usable_HP_ss_work(1)*n_gen)/n_gen/usable_HP_ss_work(2);
                        sfc_fraction = Lfraction(load_fraction);
                        work_sfc =
sseng_sfc(sep_ss_id)/gen_sfc_FL(sfc_ss_ind)*partial_sfc(sfc_ss_ind,sfc_frac
tion-4)*LHV_correction;
                        % Total working fuel consumption
                        work_fc(i,j) =
usable_cross_work(1)*propeng_sfc(eng_id_cr)+usable_HP_ss_work(1)*sseng_sfc(
sep_ss_id)+load_fraction*usable_HP_ss_work(2)*work_sfc; % [gr/hr]
                    elseif prop_power(i)/n_trans_sp+ss_power(j)/n_ss_pd <
usable_cross_work(2)
                        % 1st sprint engine at partial load
                        usable_HP_pd(i,j) = usable_cross_work(2)-
prop_power(i)/n_trans_sp;
                        sfc_ind = SFCind(eng_id_sp,system);
                        load_fraction =
(prop_power(i)/n_trans_sp+ss_power(j)/n_ss_pd)/usable_cross_work(2);
                        sfc_fraction = Lfraction(load_fraction);
                        load_fraction2 = n_trans_sp/(1-(1-
n_trans_sp)*loss_mult(sfc_fraction-4))*load_fraction;
                        work_sfc =
propeng_sfc(eng_id_sp)/gen_sfc_FL(sfc_ind)*partial_sfc(sfc_ind,sfc_fraction
-4)*LHV_correction;
                        % Working fuel consumption
                        work_fc(i,j) =
load_fraction2*usable_cross_work(2)*work_sfc; % [gr/hr]
                    elseif prop_power(i)/n_trans_sp+ss_power(j)/n_ss_pd <
usable_cross_work(2)+usable_HP_ss_work(1) && prop_power(i)/n_trans_sp <
usable_cross_work(2)
```

```
                        % 1st sprint engine at full load, 1st ss engine at
partial load
                        usable_HP_pd(i,j) = usable_cross_work(2)-
prop_power(i)/n_trans_sp;
                        load_fraction = (ss_power(j)-
usable_HP_pd(i,j)*n_ss_pd)/n_gen/usable_HP_ss_work(1);
                        sfc_fraction = Lfraction(load_fraction);
                        work_sfc =
sseng_sfc(sep_ss_id)/gen_sfc_FL(sfc_ss_ind)*partial_sfc(sfc_ss_ind,sfc_frac
tion-4)*LHV_correction;
                        % Total working fuel consumption
                        work_fc(i,j) =
usable_cross_work(2)*propeng_sfc(eng_id_sp)+load_fraction*usable_HP_ss_work
(1)*work_sfc; % [gr/hr]
                    elseif prop_power(i)/n_trans_sp+ss_power(j)/n_ss_pd <
usable_cross_work(2)+sum(usable_HP_ss_work) && prop_power(i)/n_trans_sp <
usable_cross_work(2)
                        % 1st sprint and 1st ss engines at full load, 2nd
ss engine at partial load
                        usable_HP_pd(i,j) = usable_cross_work(2)-
prop_power(i)/n_trans_sp;
                        load_fraction = (ss_power(j)-
usable_HP_pd(i,j)*n_ss_pd-
usable_HP_ss_work(1)*n_gen)/n_gen/usable_HP_ss_work(2);
                        sfc_fraction = Lfraction(load_fraction);
                        work_sfc =
sseng_sfc(sep_ss_id)/gen_sfc_FL(sfc_ss_ind)*partial_sfc(sfc_ss_ind,sfc_frac
tion-4)*LHV_correction;
                        % Total working fuel consumption
                        work_fc(i,j) =
usable_cross_work(2)*propeng_sfc(eng_id_sp)+usable_HP_ss_work(1)*sseng_sfc(
sep_ss_id)+load_fraction*usable_HP_ss_work(2)*work_sfc; % [gr/hr]
                    elseif prop_power(i)/n_trans_sp+ss_power(j)/n_ss_pd <
sum(usable_cross_work(1:2))
                        % Cruise engine at full load, 1st sprint engine at
partial load
                        usable_HP_pd(i,j) =
usable_cross_work(1)*(n_trans_cr-
n_trans_sp)/n_trans_sp+sum(usable_cross_work(1:2))-
prop_power(i)/n_trans_sp;
                        sfc_ind = SFCind(eng_id_sp,system);
                        load_fraction = ((prop_power(i)-
usable_cross_work(1)*n_trans_cr)/n_trans_sp+ss_power(j)/n_ss_pd)/usable_cro
ss_work(2);
                        sfc_fraction = Lfraction(load_fraction);
                        load_fraction2 = n_trans_sp/(1-(1-
n_trans_sp)*loss_mult(sfc_fraction-4))*load_fraction;
                        work_sfc =
propeng_sfc(eng_id_sp)/gen_sfc_FL(sfc_ind)*partial_sfc(sfc_ind,sfc_fraction
-4)*LHV_correction;
                        % Total working fuel consumption
                        work_fc(i,j) =
propeng_sfc(eng_id_cr)*usable_cross_work(1)+load_fraction2*usable_cross_wor
k(2)*work_sfc; % [gr/hr]
                    elseif prop_power(i)/n_trans_sp+ss_power(j)/n_ss_pd <
sum(usable_cross_work(1:2))+usable_HP_ss_work(1) &&
prop_power(i)/n_trans_sp < sum(usable_cross_work(1:2))
                        % Cruise and 1st sprint engines at full load, 1st
```

```
ss engine at partial load
                        usable_HP_pd(i,j) =
usable_cross_work(1)*(n_trans_cr-
n_trans_sp)/n_trans_sp+sum(usable_cross_work(1:2))-
prop_power(i)/n_trans_sp;
                        load_fraction = (ss_power(j)-
usable_HP_pd(i,j)*n_ss_pd)/n_gen/usable_HP_ss_work(1);
                        sfc_fraction = Lfraction(load_fraction);
                        work_sfc =
sseng_sfc(sep_ss_id)/gen_sfc_FL(sfc_ss_ind)*partial_sfc(sfc_ss_ind,sfc_frac
tion-4)*LHV_correction;
                        % Total working fuel consumption
                        work_fc(i,j) =
propeng_sfc(eng_id_cr)*usable_cross_work(1)+usable_cross_work(2)*propeng_sf
c(eng_id_sp)+load_fraction*usable_HP_ss_work(1)*work_sfc; % [gr/hr]
                    elseif prop_power(i)/n_trans_sp+ss_power(j)/n_ss_pd <
sum(usable_cross_work(1:2))+sum(usable_HP_ss_work) &&
prop_power(i)/n_trans_sp < sum(usable_cross_work(1:2))
                        % Cruise, 1st sprint and 1st ss engines at full
load, 2nd ss engine at partial load
                        usable_HP_pd(i,j) =
usable_cross_work(1)*(n_trans_cr-
n_trans_sp)/n_trans_sp+sum(usable_cross_work(1:2))-
prop_power(i)/n_trans_sp;
                        load_fraction = (ss_power(j)-
usable_HP_pd(i,j)*n_ss_pd-
usable_HP_ss_work(1)*n_gen)/n_gen/usable_HP_ss_work(2);
                        sfc_fraction = Lfraction(load_fraction);
                        work_sfc =
sseng_sfc(sep_ss_id)/gen_sfc_FL(sfc_ss_ind)*partial_sfc(sfc_ss_ind,sfc_frac
tion-4)*LHV_correction;
                        % Total working fuel consumption
                        work_fc(i,j) =
propeng_sfc(eng_id_cr)*usable_cross_work(1)+usable_cross_work(2)*propeng_sf
c(eng_id_sp)+usable_HP_ss_work(1)*sseng_sfc(sep_ss_id)+load_fraction*usable
_HP_ss_work(2)*work_sfc; % [gr/hr]
                    elseif prop_power(i)/n_trans_sp+ss_power(j)/n_ss_pd <
sum(usable_cross_work(2:3))
                        % 1st sprint engine at full load, 2nd at partial
                        % load
                        usable_HP_pd(i,j) = sum(usable_cross_work(2:3))-
prop_power(i)/n_trans_sp;
                        sfc_ind = SFCind(eng_id_sp,system);
                        load_fraction = ((prop_power(i)-
usable_cross_work(2)*n_trans_sp)/n_trans_sp+ss_power(j)/n_ss_pd)/usable_cro
ss_work(3);
                        sfc_fraction = Lfraction(load_fraction);
                        load_fraction2 = n_trans_sp/(1-(1-
n_trans_sp)*loss_mult(sfc_fraction-4))*load_fraction;
                        work_sfc =
propeng_sfc(eng_id_sp)/gen_sfc_FL(sfc_ind)*partial_sfc(sfc_ind,sfc_fraction
-4)*LHV_correction;
                        % Total working fuel consumption
                        work_fc(i,j) =
propeng_sfc(eng_id_sp)*usable_cross_work(2)+load_fraction2*usable_cross_wor
k(3)*work_sfc; % [gr/hr]
                    elseif prop_power(i)/n_trans_sp+ss_power(j)/n_ss_pd <
sum(usable_cross_work(2:3))+usable_HP_ss_work(1) &&
```

```
prop_power(i)/n_trans_sp < sum(usable_cross_work(2:3))
                        % Both sprint engines at full load, 1st ss engine
at partial load
                        usable_HP_pd(i,j) = sum(usable_cross_work(2:3))-
prop_power(i)/n_trans_sp;
                        load_fraction = (ss_power(j)-
usable_HP_pd(i,j)*n_ss_pd)/n_gen/usable_HP_ss_work(1);
                        sfc_fraction = Lfraction(load_fraction);
                        work_sfc =
sseng_sfc(sep_ss_id)/gen_sfc_FL(sfc_ss_ind)*partial_sfc(sfc_ss_ind,sfc_frac
tion-4)*LHV_correction;
                        % Total working fuel consumption
                        work_fc(i,j) =
sum(usable_cross_work(2:3))*propeng_sfc(eng_id_sp)+load_fraction*usable_HP_
ss_work(1)*work_sfc; % [gr/hr]
                    elseif prop_power(i)/n_trans_sp+ss_power(j)/n_ss_pd <
sum(usable_cross_work(2:3))+sum(usable_HP_ss_work) &&
prop_power(i)/n_trans_sp < sum(usable_cross_work(2:3))
                        % Both sprint and 1st ss engines at full load, 2nd
                        % ss engine at partial load
                        usable_HP_pd(i,j) = sum(usable_cross_work(2:3))-
prop_power(i)/n_trans_sp;
                        load_fraction = (ss_power(j)-
usable_HP_pd(i,j)*n_ss_pd-
usable_HP_ss_work(1)*n_gen)/n_gen/usable_HP_ss_work(2);
                        sfc_fraction = Lfraction(load_fraction);
                        work_sfc =
sseng_sfc(sep_ss_id)/gen_sfc_FL(sfc_ss_ind)*partial_sfc(sfc_ss_ind,sfc_frac
tion-4)*LHV_correction;
                        % Total working fuel consumption
                        work_fc(i,j) =
sum(usable_cross_work(2:3))*propeng_sfc(eng_id_sp)+usable_HP_ss_work(1)*sse
ng_sfc(sep_ss_id)+load_fraction*usable_HP_ss_work(2)*work_sfc; % [gr/hr]
                    elseif prop_power(i)/n_trans_sp+ss_power(j)/n_ss_pd <
sum(usable_cross_work)
                        % Cruise and 1st sprint engine at full load, 2nd
                        % sprint engine at partial load
                        usable_HP_pd(i,j) =
usable_cross_work(1)*(n_trans_cr-
n_trans_sp)/n_trans_sp+sum(usable_cross_work)-prop_power(i)/n_trans_sp;
                        sfc_ind = SFCind(eng_id_sp,system);
                        load_fraction = ((prop_power(i)-
usable_cross_work(1)*n_trans_cr-
usable_cross_work(2)*n_trans_sp)/n_trans_sp+ss_power(j)/n_ss_pd)/usable_cro
ss_work(3);
                        sfc_fraction = Lfraction(load_fraction);
                        load_fraction2 = n_trans_sp/(1-(1-
n_trans_sp)*loss_mult(sfc_fraction-4))*load_fraction;
                        work_sfc =
propeng_sfc(eng_id_sp)/gen_sfc_FL(sfc_ind)*partial_sfc(sfc_ind,sfc_fraction
-4)*LHV_correction;
                        % Total working fuel consumption
                        work_fc(i,j) =
propeng_sfc(eng_id_cr)*usable_cross_work(1)+propeng_sfc(eng_id_sp)*usable_c
ross_work(2)+load_fraction2*usable_cross_work(3)*work_sfc; % [gr/hr]
                    elseif prop_power(i)/n_trans_sp+ss_power(j)/n_ss_pd <
sum(usable_cross_work)+usable_HP_ss_work(1) && prop_power(i)/n_trans_sp <
sum(usable_cross_work)
```

```
                        % Cruise and both sprint engines at full load, 1st
ss engine at partial load
                        usable_HP_pd(i,j) =
usable_cross_work(1)*(n_trans_cr-
n_trans_sp)/n_trans_sp+sum(usable_cross_work(1:3))-
prop_power(i)/n_trans_sp;
                        load_fraction = (ss_power(j)-
usable_HP_pd(i,j)*n_ss_pd)/n_gen/usable_HP_ss_work(1);
                        sfc_fraction = Lfraction(load_fraction);
                        work_sfc =
sseng_sfc(sep_ss_id)/gen_sfc_FL(sfc_ss_ind)*partial_sfc(sfc_ss_ind,sfc_frac
tion-4)*LHV_correction;
                        % Total working fuel consumption
                        work_fc(i,j) =
propeng_sfc(eng_id_cr)*usable_cross_work(1)+sum(usable_cross_work(2:3))*pro
peng_sfc(eng_id_sp)+load_fraction*usable_HP_ss_work(1)*work_sfc; % [gr/hr]
                    elseif prop_power(i)/n_trans_sp+ss_power(j)/n_ss_pd <
sum(usable_cross_work)+sum(usable_HP_ss_work) && prop_power(i)/n_trans_sp <
sum(usable_cross_work)
                        % Cruise, both sprint and 1st ss engines at full
                        % load, 2nd ss engine at partial load
                        usable_HP_pd(i,j) =
usable_cross_work(1)*(n_trans_cr-
n_trans_sp)/n_trans_sp+sum(usable_cross_work)-prop_power(i)/n_trans_sp;
                        load_fraction = (ss_power(j)-
usable_HP_pd(i,j)*n_ss_pd-
usable_HP_ss_work(1)*n_gen)/n_gen/usable_HP_ss_work(2);
                        sfc_fraction = Lfraction(load_fraction);
                        work_sfc =
sseng_sfc(sep_ss_id)/gen_sfc_FL(sfc_ss_ind)*partial_sfc(sfc_ss_ind,sfc_frac
tion-4)*LHV_correction;
                        % Total working fuel consumption
                        work_fc(i,j) =
propeng_sfc(eng_id_cr)*usable_cross_work(1)+sum(usable_cross_work(2:3))*pro
peng_sfc(eng_id_sp)+usable_HP_ss_work(1)*sseng_sfc(sep_ss_id)+load_fraction
*usable_HP_ss_work(2)*work_sfc; % [gr/hr]
                    end
                else % 2 cruise engines and 1 sprint engine
                    if prop_power(i)/n_trans_cr+ss_power(j)/n_ss_pd <
usable_cross_work(1)
                        % 1st cruise engine at partial load
                        usable_HP_pd(i,j) = usable_cross_work(1)-
prop_power(i)/n_trans_cr;
                        sfc_ind = SFCind(eng_id_cr,system);
                        load_fraction =
(prop_power(i)/n_trans_cr+ss_power(j)/n_ss_pd)/usable_cross_work(1);
                        sfc_fraction = Lfraction(load_fraction);
                        load_fraction2 = n_trans_cr/(1-(1-
n_trans_cr)*loss_mult(sfc_fraction-4))*load_fraction;
                        work_sfc =
propeng_sfc(eng_id_cr)/gen_sfc_FL(sfc_ind)*partial_sfc(sfc_ind,sfc_fraction
-4)*LHV_correction;
                        % Working fuel consumption
                        work_fc(i,j) =
load_fraction2*usable_cross_work(1)*work_sfc; % [gr/hr]
                    elseif prop_power(i)/n_trans_cr+ss_power(j)/n_ss_pd <
usable_cross_work(1)+usable_HP_ss_work(1) && prop_power(i)/n_trans_cr <
usable_cross_work(1)
```

```
                        % 1st cruise engine at full load and 1st ss engine
at
                        % partial load
                        usable_HP_pd(i,j) = usable_cross_work(1)-
prop_power(i)/n_trans_cr;
                        load_fraction = (ss_power(j)-
usable_HP_pd(i,j)*n_ss_pd)/n_gen/usable_HP_ss_work(1);
                        sfc_fraction = Lfraction(load_fraction);
                        work_sfc =
sseng_sfc(sep_ss_id)/gen_sfc_FL(sfc_ss_ind)*partial_sfc(sfc_ss_ind,sfc_frac
tion-4)*LHV_correction;
                        % Total working fuel consumption
                        work_fc(i,j) =
usable_cross_work(1)*propeng_sfc(eng_id_cr)+load_fraction*usable_HP_ss_work
(1)*work_sfc; % [gr/hr]
                    elseif prop_power(i)/n_trans_cr+ss_power(j)/n_ss_pd <
usable_cross_work(1)+sum(usable_HP_ss_work) && prop_power(i)/n_trans_cr <
usable_cross_work(1)
                        % 1st cruise and 1st ss engine at full load, 2nd ss
engine at partial load
                        usable_HP_pd(i,j) = usable_cross_work(1)-
prop_power(i)/n_trans_cr;
                        load_fraction = (ss_power(j)-
usable_HP_pd(i,j)*n_ss_pd-
usable_HP_ss_work(1)*n_gen)/n_gen/usable_HP_ss_work(2);
                        sfc_fraction = Lfraction(load_fraction);
                        work_sfc =
sseng_sfc(sep_ss_id)/gen_sfc_FL(sfc_ss_ind)*partial_sfc(sfc_ss_ind,sfc_frac
tion-4)*LHV_correction;
                        % Total working fuel consumption
                        work_fc(i,j) =
usable_cross_work(1)*propeng_sfc(eng_id_cr)+usable_HP_ss_work(1)*sseng_sfc(
sep_ss_id)+load_fraction*usable_HP_ss_work(2)*work_sfc; % [gr/hr]
                    elseif prop_power(i)/n_trans_cr+ss_power(j)/n_ss_pd <
sum(usable_cross_work(1:2))
                        % 1st cruise engine at full load, 2nd cruise
                        % engine at partial load
                        usable_HP_pd(i,j) = sum(usable_cross_work(1:2))-
prop_power(i)/n_trans_cr;
                        sfc_ind = SFCind(eng_id_cr,system);
                        load_fraction = ((prop_power(i)-
usable_cross_work(1)*n_trans_cr)/n_trans_cr+ss_power(j)/n_ss_pd)/usable_cro
ss_work(2);
                        sfc_fraction = Lfraction(load_fraction);
                        load_fraction2 = n_trans_cr/(1-(1-
n_trans_cr)*loss_mult(sfc_fraction-4))*load_fraction;
                        work_sfc =
propeng_sfc(eng_id_cr)/gen_sfc_FL(sfc_ind)*partial_sfc(sfc_ind,sfc_fraction
-4)*LHV_correction;
                        % Working fuel consumption
                        work_fc(i,j) =
usable_cross_work(1)*propeng_sfc(eng_id_cr)+load_fraction2*usable_cross_wor
k(2)*work_sfc; % [gr/hr]
                    elseif prop_power(i)/n_trans_cr+ss_power(j)/n_ss_pd <
sum(usable_cross_work(1:2))+usable_HP_ss_work(1) &&
prop_power(i)/n_trans_cr < sum(usable_cross_work(1:2))
                        % Both cruise engines at full load, 1st ss engine
                        % at partial load
```

```
                        usable_HP_pd(i,j) = sum(usable_cross_work(1:2))-
prop_power(i)/n_trans_cr;
                        load_fraction = (ss_power(j)-
usable_HP_pd(i,j)*n_ss_pd)/n_gen/usable_HP_ss_work(1);
                        sfc_fraction = Lfraction(load_fraction);
                        work_sfc =
sseng_sfc(sep_ss_id)/gen_sfc_FL(sfc_ss_ind)*partial_sfc(sfc_ss_ind,sfc_frac
tion-4)*LHV_correction;
                        % Total working fuel consumption
                        work_fc(i,j) =
sum(usable_cross_work(1:2))*propeng_sfc(eng_id_cr)+load_fraction*usable_HP_
ss_work(1)*work_sfc; % [gr/hr]
                    elseif prop_power(i)/n_trans_cr+ss_power(j)/n_ss_pd <
sum(usable_cross_work(1:2))+sum(usable_HP_ss_work) &&
prop_power(i)/n_trans_cr < sum(usable_cross_work(1:2))
                        % Both cruise and 1st ss engines at full load, 2nd
                        % ss engine at partial load
                        usable_HP_pd(i,j) = sum(usable_cross_work(1:2))-
prop_power(i)/n_trans_cr;
                        load_fraction = (ss_power(j)-
usable_HP_pd(i,j)*n_ss_pd-
usable_HP_ss_work(1)*n_gen)/n_gen/usable_HP_ss_work(2);
                        sfc_fraction = Lfraction(load_fraction);
                        work_sfc =
sseng_sfc(sep_ss_id)/gen_sfc_FL(sfc_ss_ind)*partial_sfc(sfc_ss_ind,sfc_frac
tion-4)*LHV_correction;
                        % Total working fuel consumption
                        work_fc(i,j) =
sum(usable_cross_work(1:2))*propeng_sfc(eng_id_cr)+usable_HP_ss_work(1)*sse
ng_sfc(sep_ss_id)+load_fraction*usable_HP_ss_work(2)*work_sfc; % [gr/hr]
                    elseif prop_power(i)/n_trans_sp+ss_power(j)/n_ss_pd <
usable_cross_work(3)
                        % Sprint engine at partial load
                        usable_HP_pd(i,j) = usable_cross_work(3)-
prop_power(i)/n_trans_sp;
                        sfc_ind = SFCind(eng_id_sp,system);
                        load_fraction =
(prop_power(i)/n_trans_sp+ss_power(j)/n_ss_pd)/usable_cross_work(3);
                        sfc_fraction = Lfraction(load_fraction);
                        load_fraction2 = n_trans_sp/(1-(1-
n_trans_sp)*loss_mult(sfc_fraction-4))*load_fraction;
                        work_sfc =
propeng_sfc(eng_id_sp)/gen_sfc_FL(sfc_ind)*partial_sfc(sfc_ind,sfc_fraction
-4)*LHV_correction;
                        % Working fuel consumption
                        work_fc(i,j) =
load_fraction2*usable_cross_work(3)*work_sfc; % [gr/hr]
                    elseif prop_power(i)/n_trans_sp+ss_power(j)/n_ss_pd <
usable_cross_work(3)+usable_HP_ss_work(1) && prop_power(i)/n_trans_sp <
usable_cross_work(3)
                        % Sprint engine at full load, 1st ss engine at
                        % partial load
                        usable_HP_pd(i,j) = usable_cross_work(3)-
prop_power(i)/n_trans_sp;
                        load_fraction = (ss_power(j)-
usable_HP_pd(i,j)*n_ss_pd)/n_gen/usable_HP_ss_work(1);
                        sfc_fraction = Lfraction(load_fraction);
                        work_sfc =
```

```
sseng_sfc(sep_ss_id)/gen_sfc_FL(sfc_ss_ind)*partial_sfc(sfc_ss_ind,sfc_frac
tion-4)*LHV_correction;
                        % Total working fuel consumption
                        work_fc(i,j) =
usable_cross_work(3)*propeng_sfc(eng_id_sp)+load_fraction*usable_HP_ss_work
(1)*work_sfc; % [gr/hr]
                    elseif prop_power(i)/n_trans_sp+ss_power(j)/n_ss_pd <
usable_cross_work(3)+sum(usable_HP_ss_work) && prop_power(i)/n_trans_sp <
usable_cross_work(3)
                        % Sprint and 1st ss engines at full load, 2nd ss
                        % engine at partial load
                        usable_HP_pd(i,j) = usable_cross_work(3)-
prop_power(i)/n_trans_sp;
                        load_fraction = (ss_power(j)-
usable_HP_pd(i,j)*n_ss_pd-
usable_HP_ss_work(1)*n_gen)/n_gen/usable_HP_ss_work(2);
                        sfc_fraction = Lfraction(load_fraction);
                        work_sfc =
sseng_sfc(sep_ss_id)/gen_sfc_FL(sfc_ss_ind)*partial_sfc(sfc_ss_ind,sfc_frac
tion-4)*LHV_correction;
                        % Total working fuel consumption
                        work_fc(i,j) =
usable_cross_work(3)*propeng_sfc(eng_id_sp)+usable_HP_ss_work(1)*sseng_sfc(
sep_ss_id)+load_fraction*usable_HP_ss_work(2)*work_sfc; % [gr/hr]
                    elseif prop_power(i)/n_trans_sp+ss_power(j)/n_ss_pd <
sum(usable_cross_work(2:3))
                        % 1st cruise engine at full load, sprint engine at
                        % partial load
                        usable_HP_pd(i,j) =
usable_cross_work(2)*(n_trans_cr-
n_trans_sp)/n_trans_sp+sum(usable_cross_work(2:3))-
prop_power(i)/n_trans_sp;
                        sfc_ind = SFCind(eng_id_sp,system);
                        load_fraction = ((prop_power(i)-
usable_cross_work(2)*n_trans_cr)/n_trans_sp+ss_power(j)/n_ss_pd)/usable_cro
ss_work(3);
                        sfc_fraction = Lfraction(load_fraction);
                        load_fraction2 = n_trans_sp/(1-(1-
n_trans_sp)*loss_mult(sfc_fraction-4))*load_fraction;
                        work_sfc =
propeng_sfc(eng_id_sp)/gen_sfc_FL(sfc_ind)*partial_sfc(sfc_ind,sfc_fraction
-4)*LHV_correction;
                        % Total working fuel consumption
                        work_fc(i,j) =
propeng_sfc(eng_id_cr)*usable_cross_work(2)+load_fraction2*usable_cross_wor
k(3)*work_sfc; % [gr/hr]
                    elseif prop_power(i)/n_trans_sp+ss_power(j)/n_ss_pd <
sum(usable_cross_work(2:3))+usable_HP_ss_work(1) &&
prop_power(i)/n_trans_sp < sum(usable_cross_work(2:3))
                        % 1st cruise and sprint engines at full load, 1st
                        % ss engine at partial load
                        usable_HP_pd(i,j) =
usable_cross_work(2)*(n_trans_cr-
n_trans_sp)/n_trans_sp+sum(usable_cross_work(2:3))-
prop_power(i)/n_trans_sp;
                        load_fraction = (ss_power(j)-
usable_HP_pd(i,j)*n_ss_pd)/n_gen/usable_HP_ss_work(1);
                        sfc_fraction = Lfraction(load_fraction);
```

```
                        work_sfc =
sseng_sfc(sep_ss_id)/gen_sfc_FL(sfc_ss_ind)*partial_sfc(sfc_ss_ind,sfc_frac
tion-4)*LHV_correction;
                        % Total working fuel consumption
                        work_fc(i,j) =
propeng_sfc(eng_id_cr)*usable_cross_work(2)+usable_cross_work(3)*propeng_sf
c(eng_id_sp)+load_fraction*usable_HP_ss_work(1)*work_sfc; % [gr/hr]
                    elseif prop_power(i)/n_trans_sp+ss_power(j)/n_ss_pd <
sum(usable_cross_work(2:3))+sum(usable_HP_ss_work) &&
prop_power(i)/n_trans_sp < sum(usable_cross_work(2:3))
                        % 1st cruise, sprint and 1st ss engines at full
                        % load, 2nd ss engine at partial load
                        usable_HP_pd(i,j) =
usable_cross_work(2)*(n_trans_cr-
n_trans_sp)/n_trans_sp+sum(usable_cross_work(2:3))-
prop_power(i)/n_trans_sp;
                        load_fraction = (ss_power(j)-
usable_HP_pd(i,j)*n_ss_pd-
usable_HP_ss_work(1)*n_gen)/n_gen/usable_HP_ss_work(2);
                        sfc_fraction = Lfraction(load_fraction);
                        work_sfc =
sseng_sfc(sep_ss_id)/gen_sfc_FL(sfc_ss_ind)*partial_sfc(sfc_ss_ind,sfc_frac
tion-4)*LHV_correction;
                        % Total working fuel consumption
                        work_fc(i,j) =
propeng_sfc(eng_id_cr)*usable_cross_work(2)+usable_cross_work(3)*propeng_sf
c(eng_id_sp)+usable_HP_ss_work(1)*sseng_sfc(sep_ss_id)+load_fraction*usable
_HP_ss_work(2)*work_sfc; % [gr/hr]
                    elseif prop_power(i)/n_trans_sp+ss_power(j)/n_ss_pd <
sum(usable_cross_work(1:3))
                        % Both cruise engines at full load, sprint engine
                        % at partial load
                        usable_HP_pd(i,j) =
sum(usable_cross_work(1:2))*(n_trans_cr-
n_trans_sp)/n_trans_sp+sum(usable_cross_work)-prop_power(i)/n_trans_sp;
                        sfc_ind = SFCind(eng_id_sp,system);
                        load_fraction = ((prop_power(i)-
sum(usable_cross_work(1:2))*n_trans_cr)/n_trans_sp+ss_power(j)/n_ss_pd)/usa
ble_cross_work(3);
                        sfc_fraction = Lfraction(load_fraction);
                        load_fraction2 = n_trans_sp/(1-(1-
n_trans_sp)*loss_mult(sfc_fraction-4))*load_fraction;
                        work_sfc =
propeng_sfc(eng_id_sp)/gen_sfc_FL(sfc_ind)*partial_sfc(sfc_ind,sfc_fraction
-4)*LHV_correction;
                        % Total working fuel consumption
                        work_fc(i,j) =
propeng_sfc(eng_id_cr)*sum(usable_cross_work(1:2))+load_fraction2*usable_cr
oss_work(3)*work_sfc; % [gr/hr]
                    elseif prop_power(i)/n_trans_sp+ss_power(j)/n_ss_pd <
sum(usable_cross_work)+usable_HP_ss_work(1) && prop_power(i)/n_trans_sp <
sum(usable_cross_work)
                        % Both cruise and sprint engines at full load, 1st
                        % ss engine at partial load
                        usable_HP_pd(i,j) =
sum(usable_cross_work(1:2))*(n_trans_cr-
n_trans_sp)/n_trans_sp+sum(usable_cross_work)-prop_power(i)/n_trans_sp;
                        load_fraction = (ss_power(j)-
```

```
usable_HP_pd(i,j)*n_ss_pd)/n_gen/usable_HP_ss_work(1);
                        sfc_fraction = Lfraction(load_fraction);
                        work_sfc =
sseng_sfc(sep_ss_id)/gen_sfc_FL(sfc_ss_ind)*partial_sfc(sfc_ss_ind,sfc_frac
tion-4)*LHV_correction;
                        % Total working fuel consumption
                        work_fc(i,j) =
propeng_sfc(eng_id_cr)*sum(usable_cross_work(1:2))+usable_cross_work(3)*pro
peng_sfc(eng_id_sp)+load_fraction*usable_HP_ss_work(1)*work_sfc; % [gr/hr]
                    elseif prop_power(i)/n_trans_sp+ss_power(j)/n_ss_pd <
sum(usable_cross_work)+sum(usable_HP_ss_work) && prop_power(i)/n_trans_sp <
sum(usable_cross_work)
                        % Both cruise, sprint and 1st ss engines at full
                        % load, 2nd ss engine at partial load
                        usable_HP_pd(i,j) =
sum(usable_cross_work(1:2))*(n_trans_cr-
n_trans_sp)/n_trans_sp+sum(usable_cross_work)-prop_power(i)/n_trans_sp;
                        load_fraction = (ss_power(j)-
usable_HP_pd(i,j)*n_ss_pd-
usable_HP_ss_work(1)*n_gen)/n_gen/usable_HP_ss_work(2);
                        sfc_fraction = Lfraction(load_fraction);
                        work_sfc =
sseng_sfc(sep_ss_id)/gen_sfc_FL(sfc_ss_ind)*partial_sfc(sfc_ss_ind,sfc_frac
tion-4)*LHV_correction;
                        % Total working fuel consumption
                        work_fc(i,j) =
propeng_sfc(eng_id_cr)*sum(usable_cross_work(1:2))+usable_cross_work(3)*pro
peng_sfc(eng_id_sp)+usable_HP_ss_work(1)*sseng_sfc(sep_ss_id)+load_fraction
*usable_HP_ss_work(2)*work_sfc; % [gr/hr]
                    end
                end
            end
        end
    end
end


%% Total fuel consumption for the given operating profile

fc_total = sum(sum(profile_data(2:end-1,2:end-2).*work_fc/1000))/100; %
Total fuel consumption [kg/hr]

f = fc_total+penalty; % Objective function (minimize)
```

Function *IPSf.m*:

```
function f = IPSf(binarystr)
%% Function evaluates integrated power systems (IPS)

global profile_data dspeed dss SHP_3 gen_sfc_FL partial_sfc LHV_correction
M_s...
    ipseng_type ipseng_sfc ipseng_power

%% PGM selection
ips_id_pr = 1+Convert( binarystr, 1, 4 ); % Primary PGM selection ID
ips_id_sec = 1+Convert( binarystr, 5, 4 ); % Secondary PGM selection ID
ips_num_pr = 2+Convert( binarystr, 9, 1 ); % Number of primary PGMs (2 or
3)

ips_sec_dec = 3-ips_num_pr; % Secondary PGMs exist if primary PGMs number
is 2
ips_num_sec = 2*(3-ips_num_pr); % Number of secondary PGMs is 2 when 2
primary PGMs are present

if ips_num_pr==2 && ipseng_power(ips_id_sec)>ipseng_power(ips_id_pr)
    ips_id_trans = ips_id_pr;
    ips_id_pr = ips_id_sec;
    ips_id_sec = ips_id_trans;
end

%% IPS efficiencies

% Efficiency correction for electric drive generators and motors from SNAME
elec_cor_matrix = [26 95.4 ; 37 97 ; 42 97.5 ; 48 98 ; 55.5 98.5 ; 65 99 ;
78 99.5 ; 85 99.7 ; 100 100 ];
pp_elec = spline(elec_cor_matrix(:,1),elec_cor_matrix(:,2));
elec_cor_mult = ppval(pp_elec,25:100);

% % Use to plot datapoints and check piecewise polynomial fits the
datapoints
% figure;
%
plot(elec_cor_matrix(:,1),elec_cor_matrix(:,2),'o',25:100,elec_cor_mult),xl
abel('Load fraction [%]'),ylabel('Correction factor'),title('Efficiency
correction for electric drive generators and motors at partial load
operation')


% Generator, motor drive and propulsion motor efficiencies
n_gen = 0.96;
n_mdrive = 0.96;
n_pmotor = 0.96;
n_ips_prop = n_gen*n_mdrive*n_pmotor;

% Ship service power generation efficiency
n_ips_pcm = 0.965; % PCM efficiency
n_ips_ss = n_gen*n_ips_pcm;

%% Usable power calculation
```

```
usable_HP_ips_work = zeros(ips_num_pr+ips_num_sec,1);
if ips_sec_dec == 1
    for i = 1:ips_num_sec;
    usable_HP_ips_work(i) = ipseng_power(ips_id_sec);
    end
    for i = ips_num_sec+1:ips_num_sec+ips_num_pr
        usable_HP_ips_work(i) = ipseng_power(ips_id_pr);
    end
elseif ips_sec_dec == 0
    for i = 1:ips_num_pr
        usable_HP_ips_work(i) = ipseng_power(ips_id_pr);
    end
end

clear i


%% IPS fuel consumption calculations
% load_fraction2: Engine load increases due to partial load motor drive and
propulsion motor operation
% Check that IPS provides the required HP
i_speed = length(profile_data(2:end-1,1));
i_sspower = length(profile_data(1,2:end-2));
if dss/n_ips_ss+SHP_3(dspeed+1)/n_ips_prop > sum(usable_HP_ips_work)*(1-
M_s)
    work_fc =
ipseng_sfc(ips_id_pr)*sum(usable_HP_ips_work)*ones(i_speed,i_sspower);
    penalty = 0.5*(dss/n_ips_ss+SHP_3(dspeed+1)/n_ips_prop-
sum(usable_HP_ips_work)*(1-M_s));
else
    penalty = 0;
    system = 'ips';
    % Pre-allocate
    prop_power = zeros(i_speed,1);
    ss_power = profile_data(1,2:end-2);
    work_fc = zeros(i_speed,i_sspower); % lin=prop, col=ss
    motor_load_fraction = zeros(i_speed,1);
    motor_fraction = zeros(i_speed,1);
    if ips_num_pr == 2 % 2 primary and 2 secondary engines present
        for i = 1:i_speed
            prop_power(i) = SHP_3(i+1);
            motor_load_fraction(i) = prop_power(i)/SHP_3(dspeed+1); % motor
drive and propulsion motor load fraction
            motor_fraction(i) = Mfraction(motor_load_fraction(i));
            for j = 1:i_sspower
                if ss_power(j)/n_ips_ss+prop_power(i)/n_ips_prop <
sum(usable_HP_ips_work(1:2))
                    % Both secondary engines at partial load
                    load_fraction =
(ss_power(j)/n_ips_ss+prop_power(i)/n_ips_prop)/sum(usable_HP_ips_work(1:2)
);
                    sfc_fraction = Lfraction(load_fraction);
                    sfc_ind = SFCind(ips_id_sec,system);
                    load_fraction2 =
100*load_fraction/elec_cor_mult(motor_fraction(i)-24);
                    work_sfc_ips =
ipseng_sfc(ips_id_sec)/gen_sfc_FL(sfc_ind)*partial_sfc(sfc_ind,sfc_fraction
-4)*LHV_correction;
```

```
                    work_fc(i,j) =
load_fraction2*sum(usable_HP_ips_work(1:2))*work_sfc_ips; % [gr/hr]
                elseif ss_power(j)/n_ips_ss+prop_power(i)/n_ips_prop <
usable_HP_ips_work(1)+usable_HP_ips_work(3)
                    % 1st secondary engine at full load, 1st primary engine
                    % at partial load
                    load_fraction =
(ss_power(j)/n_ips_ss+prop_power(i)/n_ips_prop-
usable_HP_ips_work(1))/usable_HP_ips_work(3);
                    sfc_fraction = Lfraction(load_fraction);
                    sfc_ind = SFCind(ips_id_pr,system);
                    load_fraction2 =
100*load_fraction/elec_cor_mult(motor_fraction(i)-24);
                    work_sfc =
ipseng_sfc(ips_id_pr)/gen_sfc_FL(sfc_ind)*partial_sfc(sfc_ind,sfc_fraction-
4)*LHV_correction;
                    % Total working fuel consumption
                    work_fc(i,j) =
ipseng_sfc(ips_id_sec)*usable_HP_ips_work(1)+load_fraction2*usable_HP_ips_w
ork(3)*work_sfc; % [gr/hr]
                elseif ss_power(j)/n_ips_ss+prop_power(i)/n_ips_prop <
sum(usable_HP_ips_work(1:3)) && sum(usable_HP_ips_work(1:3)) <
sum(usable_HP_ips_work(3:4))
                    % Both secondary engines at full load, 1st primary
                    % engine at partial load
                    load_fraction =
(ss_power(j)/n_ips_ss+prop_power(i)/n_ips_prop-
sum(usable_HP_ips_work(1:2)))/usable_HP_ips_work(3);
                    sfc_fraction = Lfraction(load_fraction);
                    sfc_ind = SFCind(ips_id_pr,system);
                    load_fraction2 =
100*load_fraction/elec_cor_mult(motor_fraction(i)-24);
                    work_sfc =
ipseng_sfc(ips_id_pr)/gen_sfc_FL(sfc_ind)*partial_sfc(sfc_ind,sfc_fraction-
4)*LHV_correction;
                    % Total working fuel consumption
                    work_fc(i,j) =
ipseng_sfc(ips_id_sec)*sum(usable_HP_ips_work(1:2))+load_fraction2*usable_H
P_ips_work(3)*work_sfc; % [gr/hr]
                elseif ss_power(j)/n_ips_ss+prop_power(i)/n_ips_prop <
sum(usable_HP_ips_work(3:4))
                    % 1st primary engine at full load, 2nd primary engine
                    % at partial load
                    load_fraction =
(ss_power(j)/n_ips_ss+prop_power(i)/n_ips_prop-
usable_HP_ips_work(3))/usable_HP_ips_work(4);
                    sfc_fraction = Lfraction(load_fraction);
                    sfc_ind = SFCind(ips_id_pr,system);
                    load_fraction2 =
100*load_fraction/elec_cor_mult(motor_fraction(i)-24);
                    work_sfc =
ipseng_sfc(ips_id_pr)/gen_sfc_FL(sfc_ind)*partial_sfc(sfc_ind,sfc_fraction-
4)*LHV_correction;
                    % Total working fuel consumption
                    work_fc(i,j) =
ipseng_sfc(ips_id_pr)*usable_HP_ips_work(3)+load_fraction2*usable_HP_ips_wo
rk(4)*work_sfc; % [gr/hr]
                elseif ss_power(j)/n_ips_ss+prop_power(i)/n_ips_prop <
```

```
sum(usable_HP_ips_work(1:3)) && sum(usable_HP_ips_work(1:3)) >=
sum(usable_HP_ips_work(3:4))
                    % Both secondary engine at full load, 1st primary
                    % engine at partial load
                    load_fraction =
(ss_power(j)/n_ips_ss+prop_power(i)/n_ips_prop-
sum(usable_HP_ips_work(1:2)))/usable_HP_ips_work(3);
                    sfc_fraction = Lfraction(load_fraction);
                    sfc_ind = SFCind(ips_id_pr,system);
                    load_fraction2 =
100*load_fraction/elec_cor_mult(motor_fraction(i)-24);
                    work_sfc =
ipseng_sfc(ips_id_pr)/gen_sfc_FL(sfc_ind)*partial_sfc(sfc_ind,sfc_fraction-
4)*LHV_correction;
                    % Total working fuel consumption
                    work_fc(i,j) =
ipseng_sfc(ips_id_sec)*sum(usable_HP_ips_work(1:2))+load_fraction2*usable_H
P_ips_work(3)*work_sfc; % [gr/hr]
                elseif ss_power(j)/n_ips_ss+prop_power(i)/n_ips_prop <
sum(usable_HP_ips_work(2:4))
                    % 1st primary and 1st secondary engines at full load,
                    % 2nd primary engine at partial load
                    load_fraction =
(ss_power(j)/n_ips_ss+prop_power(i)/n_ips_prop-
sum(usable_HP_ips_work(2:3)))/usable_HP_ips_work(4);
                    sfc_fraction = Lfraction(load_fraction);
                    sfc_ind = SFCind(ips_id_pr,system);
                    load_fraction2 =
100*load_fraction/elec_cor_mult(motor_fraction(i)-24);
                    work_sfc =
ipseng_sfc(ips_id_pr)/gen_sfc_FL(sfc_ind)*partial_sfc(sfc_ind,sfc_fraction-
4)*LHV_correction;
                    % Total working fuel consumption
                    work_fc(i,j) =
ipseng_sfc(ips_id_sec)*usable_HP_ips_work(2)+ipseng_sfc(ips_id_pr)*usable_H
P_ips_work(3)+load_fraction2*usable_HP_ips_work(4)*work_sfc; % [gr/hr]
                else
                    % 1st primary and both secondary engines at full load,
                    % 2nd primary engine at partial load
                    load_fraction =
(ss_power(j)/n_ips_ss+prop_power(i)/n_ips_prop-
sum(usable_HP_ips_work(1:3)))/usable_HP_ips_work(4);
                    sfc_fraction = Lfraction(load_fraction);
                    sfc_ind = SFCind(ips_id_pr,system);
                    load_fraction2 =
100*load_fraction/elec_cor_mult(motor_fraction(i)-24);
                    work_sfc =
ipseng_sfc(ips_id_pr)/gen_sfc_FL(sfc_ind)*partial_sfc(sfc_ind,sfc_fraction-
4)*LHV_correction;
                    % Total working fuel consumption
                    work_fc(i,j) =
ipseng_sfc(ips_id_sec)*sum(usable_HP_ips_work(1:2))+ipseng_sfc(ips_id_pr)*u
sable_HP_ips_work(3)+load_fraction2*usable_HP_ips_work(4)*work_sfc; %
[gr/hr]
                end
            end
        end
    elseif ips_num_pr == 3
```

```
        for i = 1:i_speed
            prop_power(i) = SHP_3(i+1);
            motor_load_fraction(i) = prop_power(i)/SHP_3(dspeed+1); % motor
drive and propulsion motor load fraction
            motor_fraction(i) = Mfraction(motor_load_fraction(i));
            for j = 1:i_sspower
                if ss_power(j)/n_ips_ss+prop_power(i)/n_ips_prop <
usable_HP_ips_work(1)
                    % 2 engines at partial load
                    load_fraction =
(ss_power(j)/n_ips_ss+prop_power(i)/n_ips_prop)/sum(usable_HP_ips_work(1:2)
);
                    sfc_fraction = Lfraction(load_fraction);
                    sfc_ind = SFCind(ips_id_pr,system);
                    load_fraction2 =
100*load_fraction/elec_cor_mult(motor_fraction(i)-24);
                    work_sfc_ips =
ipseng_sfc(ips_id_pr)/gen_sfc_FL(sfc_ind)*partial_sfc(sfc_ind,sfc_fraction-
4)*LHV_correction;
                    work_fc(i,j) =
load_fraction2*sum(usable_HP_ips_work(1:2))*work_sfc_ips; % [gr/hr]
                elseif ss_power(j)/n_ips_ss+prop_power(i)/n_ips_prop <
sum(usable_HP_ips_work(1:2))
                    % 1st engine at full load, 2nd at partial load
                    load_fraction =
(ss_power(j)/n_ips_ss+prop_power(i)/n_ips_prop-
usable_HP_ips_work(1))/(usable_HP_ips_work(2));
                    sfc_fraction = Lfraction(load_fraction);
                    sfc_ind = SFCind(ips_id_pr,system);
                    load_fraction2 =
100*load_fraction/elec_cor_mult(motor_fraction(i)-24);
                    work_sfc_ips =
ipseng_sfc(ips_id_pr)/gen_sfc_FL(sfc_ind)*partial_sfc(sfc_ind,sfc_fraction-
4)*LHV_correction;
                    % Total working fuel consumption
                    work_fc(i,j) =
ipseng_sfc(ips_id_pr)*usable_HP_ips_work(1)+load_fraction2*usable_HP_ips_wo
rk(2)*work_sfc_ips; % [gr/hr]
                else
                    % 2 engines at full load, 3rd engine at partial load
                    load_fraction =
(ss_power(j)/n_ips_ss+prop_power(i)/n_ips_prop-
sum(usable_HP_ips_work(1:2)))/(usable_HP_ips_work(3));
                    sfc_fraction = Lfraction(load_fraction);
                    sfc_ind = SFCind(ips_id_pr,system);
                    load_fraction2 =
100*load_fraction/elec_cor_mult(motor_fraction(i)-24);
                    work_sfc_ips =
ipseng_sfc(ips_id_pr)/gen_sfc_FL(sfc_ind)*partial_sfc(sfc_ind,sfc_fraction-
4)*LHV_correction;
                    % Total working fuel consumption
                    work_fc(i,j) =
ipseng_sfc(ips_id_pr)*sum(usable_HP_ips_work(1:2))+load_fraction2*usable_HP
_ips_work(3)*work_sfc_ips; % [gr/hr]
                end
            end
        end
    end
```

```
end


%% Total fuel consumption for the given operating profile

fc_total = sum(sum(profile_data(2:end-1,2:end-2).*work_fc/1000))/100; %
Total fuel consumption [kg/hr]
f = fc_total+penalty; % Objective function (minimize)
```

Function *MechSepevalf.m*:

```
function [work_fc_prop,work_sfc_prop,work_fc_avgss,work_sfc_avgss] =
MechSepSSevalf(x)
%% Function evaluates mechanical driven propulsion systems with separate
ship service systems

global profile_data dspeed dss SHP_1 gen_sfc_FL partial_sfc LHV_correction
loss_mult M_s...
    propeng_type sseng_type propeng_sfc sseng_sfc propeng_power sseng_power
propeng_name sseng_name propeng_manuf sseng_manuf


%% Propulsion engine type and number selection
eng_id_cr = 1+Convert( x, 1, 6 ); % Cruise engine selection ID
eng_id_sp = 1+Convert( x, 7, 6 ); % Sprint engine selection ID
eng_num_cr = 1+Convert( x, 13, 1 ); % Number of cruise engines

% Determine number of sprint engines
if eng_num_cr == 1
    eng_num_sp = 2;
else
    eng_num_sp = 1+Convert( x, 14, 1 );
end

comb_id = Convert( x, 15, 1 ); % OR/AND (0/1)


%% Mechanical driven propulsion system identification
% CODAG systems represent cross-connected configurations with 3 prime
movers
if strcmp(propeng_type(eng_id_cr),'DIESEL')==1 &&
strcmp(propeng_type(eng_id_sp),'DIESEL')==1 && eng_num_cr == eng_num_sp
    sys_type = 'CODAD';
elseif strcmp(propeng_type(eng_id_cr),'DIESEL')==1 &&
strcmp(propeng_type(eng_id_sp),'DIESEL')==0 && eng_num_cr == eng_num_sp &&
comb_id == 0
    sys_type = 'CODOG';
elseif strcmp(propeng_type(eng_id_cr),'DIESEL')==1 &&
strcmp(propeng_type(eng_id_sp),'DIESEL')==0 && eng_num_cr == eng_num_sp &&
comb_id == 1
    sys_type = 'CODAG';
elseif strcmp(propeng_type(eng_id_cr),'DIESEL')==0 &&
strcmp(propeng_type(eng_id_sp),'DIESEL')==0 && eng_num_cr == eng_num_sp
    sys_type = 'COGAG';
else
    sys_type = 'Cross-connected';
end


%% Propulsive power availability
% All configurations
usable_HP_cr = eng_num_cr*propeng_power(eng_id_cr); % [kW]
usable_HP_sp = (1-
strcmp(sys_type,'CODOG'))*usable_HP_cr+eng_num_sp*propeng_power(eng_id_sp);
```

```
% [kW], Only sprint engines if CODOG

% Cross-connected configurations
usable_cross_work = zeros(eng_num_cr+eng_num_sp,1);
if strcmp(sys_type,'Cross-connected') == 1
    for i = 1:eng_num_cr;
    usable_cross_work(i) = propeng_power(eng_id_cr);
    end
    for i = eng_num_cr+1:eng_num_cr+eng_num_sp
        usable_cross_work(i) = propeng_power(eng_id_sp);
    end
end
clear i


%% Mechanical transmission efficiencies
% Full-load transmission efficiencies
% GT transmission losses bigger due to 2 stage reduction required
if strcmp(sys_type,'Cross-connected')
    trans_mult = .99;
else
    trans_mult = 1;
end

if strcmp(propeng_type(eng_id_cr),'GT') == 1
    n_trans_cr = 0.98*trans_mult;
else
    n_trans_cr = 0.99*trans_mult;
end

if strcmp(propeng_type(eng_id_sp),'GT') == 1
    n_trans_sp = 0.98*trans_mult;
else
    n_trans_sp = 0.99*trans_mult;
end

%% Ship service engine type and number selection
ss_id = 1+Convert( x, 16, 6 ); % Ship service engine selection ID
ss_num = 3+Convert( x, 22, 1 ); % Number of ship service engines (3 or 4
allowed)


%% Ship service power and efficiency calculations
% Ship service power generation efficiencies

n_gen = 0.96;
system = 'ss';
sfc_ss_ind = SFCind(ss_id,system);

c_imb = .9; % Load factor for imbalances of electrical loads on individual
ship service systems

% Ship service power availability for different number of engines on-line
% N-1 engines
usable_HP_ss_work = zeros(ss_num-1,1); % 1st col=number working, 2nd
col=usable HP
```

```
for i_gen = 1:ss_num-1
    usable_HP_ss_work(i_gen) = i_gen*sseng_power(ss_id);
end

%% Ship service fuel consumption calculations
% On-line engines equally share load
ss_power = profile_data(1,2:end-2);
i_sspower = length(profile_data(1,2:end-2));
% Check that separate ship service plant provides the required maximum ss
% power
if dss > usable_HP_ss_work(end)*n_gen*c_imb % N-1 ss engines meeting max
demand
    disp('Warning: Selected ship-service system does not meet maximum ship-
service power requirement')
else
    % Pre-allocate
    work_fc_ss = zeros(1,i_sspower);
    work_sfc_ss = zeros(1,i_sspower);
    ss_num_work = zeros(1,i_sspower);
    plot_fraction_ss1 = zeros(1,i_sspower);
    plot_fraction_ss2 = zeros(1,i_sspower);
    plot_fraction_ss3 = zeros(1,i_sspower);
    if ss_num == 3 % 2 generators on line should meet the required ss power
profile
        for j = 1:i_sspower
            ss_num_work(j) = 2;
            load_fraction = ss_power(j)/(usable_HP_ss_work(2)*n_gen);
            plot_fraction_ss1(j) = load_fraction;
            plot_fraction_ss2(j) = load_fraction;
            sfc_fraction = Lfraction(load_fraction);
            work_sfc =
sseng_sfc(ss_id)/gen_sfc_FL(sfc_ss_ind)*partial_sfc(sfc_ss_ind,sfc_fraction
-4)*LHV_correction;
            work_fc_ss(j) = load_fraction*usable_HP_ss_work(2)*work_sfc; %
[gr/hr]
            work_sfc_ss(j) = work_sfc; % [gr/KWh]
        end
    elseif ss_num == 4 % 2 or 3 generators on line should meet the required
ss power profile
        for j = 1:i_sspower
            if ss_power(j) < usable_HP_ss_work(2)*n_gen*c_imb % 2 engines
on line meet the required ss power profile
                ss_num_work(j) = 2;
                load_fraction = ss_power(j)/(usable_HP_ss_work(2)*n_gen);
                plot_fraction_ss1(j) = load_fraction;
                plot_fraction_ss2(j) = load_fraction;
                sfc_fraction = Lfraction(load_fraction);
                work_sfc =
sseng_sfc(ss_id)/gen_sfc_FL(sfc_ss_ind)*partial_sfc(sfc_ss_ind,sfc_fraction
-4)*LHV_correction;
                work_fc_ss(j) =
load_fraction*usable_HP_ss_work(2)*work_sfc; % [gr/hr]
                work_sfc_ss(j) = work_sfc; % [gr/KWh]
            else % 3 engines on line meet the required ss power profile
                ss_num_work(j) = 3;
                load_fraction = ss_power(j)/(usable_HP_ss_work(3)*n_gen);
                plot_fraction_ss1(j) = load_fraction;
                plot_fraction_ss2(j) = load_fraction;
```

```
                plot_fraction_ss3(j) = load_fraction;
                sfc_fraction = Lfraction(load_fraction);
                work_sfc =
sseng_sfc(ss_id)/gen_sfc_FL(sfc_ss_ind)*partial_sfc(sfc_ss_ind,sfc_fraction
-4)*LHV_correction;
                work_fc_ss(j) =
load_fraction*usable_HP_ss_work(3)*work_sfc; % [gr/hr]
                work_sfc_ss(j) = work_sfc; % [gr/KWh]
            end
        end
    end
end

%% Propulsion system fuel consumption calculations
% load_fraction_2 expresses increased transmission losses at partial load
i_speed = length(profile_data(2:end-1,1));
% Check that propulsion plant provides required HP to achieve maximum
sustained speed
if SHP_1(dspeed+1) > usable_HP_sp*n_trans_sp*(1-M_s)
    disp('Warning: Selected propulsion system does not meet maximum
sustained speed requirement')
else
    % Pre-allocate
    system = 'prop';
    prop_power = zeros(i_speed,1); % Required SHP_1
    work_fc_prop = zeros(i_speed,1);
    work_sfc_prop = zeros(i_speed,1);
    work_fc_avgss = zeros(i_speed,1);
    work_sfc_avgss = zeros(i_speed,1);
    plot_fraction_cr1 = zeros(i_speed,1);
    plot_fraction_cr2 = zeros(i_speed,1);
    plot_fraction_sp1 = zeros(i_speed,1);
    plot_fraction_sp2 = zeros(i_speed,1);
    plot_fraction_cr = zeros(i_speed,1);
    plot_fraction_sp = zeros(i_speed,1);
    for i = 1:i_speed
        prop_power(i) = SHP_1(i+1); % SHP_1 starts with speed=0, prop_power
starts with speed=1knot
        if strcmp(sys_type,'Cross-connected') == 0
            if prop_power(i) < usable_HP_cr*n_trans_cr
                sfc_ind = SFCind(eng_id_cr,system);
                load_fraction = prop_power(i)/(usable_HP_cr*n_trans_cr);
                sfc_fraction = Lfraction(load_fraction);
                load_fraction2 = n_trans_cr/(1-(1-
n_trans_cr)*loss_mult(sfc_fraction-4))*load_fraction;
                plot_fraction_cr(i) = load_fraction2;
                work_sfc =
propeng_sfc(eng_id_cr)/gen_sfc_FL(sfc_ind)*partial_sfc(sfc_ind,sfc_fraction
-4)*LHV_correction;
                work_fc_prop(i) = load_fraction2*usable_HP_cr*work_sfc; %
[gr/hr]
                work_sfc_prop(i) = work_sfc; % [gr/KWh]
                work_fc_avgss(i) = work_fc_prop(i)+work_fc_ss(7);
                work_sfc_avgss(i) =
work_fc_avgss(i)/(load_fraction2*usable_HP_cr+ss_power(7)/n_gen); %
[gr/KWh]
            else
                if strcmp(sys_type,'CODOG') == 1
```

```
                    load_fraction =
prop_power(i)/(usable_HP_sp*n_trans_sp);
                    sfc_fraction = Lfraction(load_fraction);
                    load_fraction2 = n_trans_sp/(1-(1-
n_trans_sp)*loss_mult(sfc_fraction-4))*load_fraction;
                    plot_fraction_sp(i) = load_fraction2;
                    work_sfc =
propeng_sfc(eng_id_sp)/gen_sfc_FL(5)*partial_sfc(5,sfc_fraction-
4)*LHV_correction;
                    work_fc_prop(i) = load_fraction2*usable_HP_sp*work_sfc;
% [gr/hr]
                    work_sfc_prop(i) = work_sfc; % [gr/KWh]
                    work_fc_avgss(i) = work_fc_prop(i)+work_fc_ss(7);
                    work_sfc_avgss(i) =
work_fc_avgss(i)/(load_fraction2*usable_HP_sp+ss_power(7)/n_gen); %
[gr/KWh]
                else % CODAD,CODAG,COGAG
                    % Sprint engines at partial load, cruise engines
                    % at 100% load
                    sfc_ind = SFCind(eng_id_sp,system);
                    load_fraction = (prop_power(i)-
usable_HP_cr*n_trans_cr)/(usable_HP_sp*n_trans_sp);
                    sfc_fraction = Lfraction(load_fraction);
                    load_fraction2 = n_trans_sp/(1-(1-
n_trans_sp)*loss_mult(sfc_fraction-4))*load_fraction;
                    plot_fraction_cr(i) = 1;
                    plot_fraction_sp(i) = load_fraction2;
                    work_sfc =
propeng_sfc(eng_id_sp)/gen_sfc_FL(sfc_ind)*partial_sfc(sfc_ind,sfc_fraction
-4)*LHV_correction;
                    work_fc_prop(i) =
propeng_sfc(eng_id_cr)*usable_HP_cr+load_fraction2*usable_HP_sp*work_sfc; %
[gr/hr]
                    work_sfc_prop(i) =
work_fc_prop(i)/(usable_HP_cr+load_fraction2*usable_HP_sp); % [gr/KWh]
                    work_fc_avgss(i) = work_fc_prop(i)+work_fc_ss(7);
                    work_sfc_avgss(i) =
work_fc_avgss(i)/(usable_HP_cr+load_fraction2*usable_HP_sp+ss_power(7)/n_ge
n); % [gr/KWh]
                end
            end
        else % Cross-connected configurations
            if eng_num_cr == 1 % 1 cruise engine and 2 sprint engines
                if prop_power(i) < usable_cross_work(1)*n_trans_cr
                    % Cruise engine at partial load
                    sfc_ind = SFCind(eng_id_cr,system);
                    load_fraction =
prop_power(i)/(usable_cross_work(1)*n_trans_cr);
                    sfc_fraction = Lfraction(load_fraction);
                    load_fraction2 = n_trans_cr/(1-(1-
n_trans_cr)*loss_mult(sfc_fraction-4))*load_fraction;
                    plot_fraction_cr(i) = load_fraction2;
                    work_sfc =
propeng_sfc(eng_id_cr)/gen_sfc_FL(sfc_ind)*partial_sfc(sfc_ind,sfc_fraction
-4)*LHV_correction;
                    work_fc_prop(i) =
load_fraction2*usable_cross_work(1)*work_sfc; % [gr/hr]
                    work_sfc_prop(i) = work_sfc; % [gr/KWh]
```

```
                    work_fc_avgss(i) = work_fc_prop(i)+work_fc_ss(7);
                    work_sfc_avgss(i) =
work_fc_avgss(i)/(load_fraction2*usable_cross_work(1)+ss_power(7)/n_gen); %
[gr/KWh]
                elseif prop_power(i) < usable_cross_work(2)*n_trans_sp
                    % 1st sprint engine at partial load
                    sfc_ind = SFCind(eng_id_sp,system);
                    load_fraction =
prop_power(i)/(usable_cross_work(2)*n_trans_sp);
                    sfc_fraction = Lfraction(load_fraction);
                    load_fraction2 = n_trans_sp/(1-(1-
n_trans_sp)*loss_mult(sfc_fraction-4))*load_fraction;
                    plot_fraction_sp1(i) = load_fraction2;
                    work_sfc =
propeng_sfc(eng_id_sp)/gen_sfc_FL(sfc_ind)*partial_sfc(sfc_ind,sfc_fraction
-4)*LHV_correction;
                    work_fc_prop(i) =
load_fraction2*usable_cross_work(2)*work_sfc; % [gr/hr]
                    work_sfc_prop(i) = work_sfc; % [gr/KWh]
                    work_fc_avgss(i) = work_fc_prop(i)+work_fc_ss(7);
                    work_sfc_avgss(i) =
work_fc_avgss(i)/(load_fraction2*usable_cross_work(2)+ss_power(7)/n_gen); %
[gr/KWh]
                elseif prop_power(i) <
usable_cross_work(1)*n_trans_cr+usable_cross_work(2)*n_trans_sp
                    % Cruise engine at full load, 1st sprint engine
                    %  at partial load
                    sfc_ind = SFCind(eng_id_sp,system);
                    load_fraction = (prop_power(i)-
usable_cross_work(1)*n_trans_cr)/(usable_cross_work(2)*n_trans_sp);
                    sfc_fraction = Lfraction(load_fraction);
                    load_fraction2 = n_trans_sp/(1-(1-
n_trans_sp)*loss_mult(sfc_fraction-4))*load_fraction;
                    plot_fraction_cr(i) = 1;
                    plot_fraction_sp1(i) = load_fraction2;
                    work_sfc =
propeng_sfc(eng_id_sp)/gen_sfc_FL(sfc_ind)*partial_sfc(sfc_ind,sfc_fraction
-4)*LHV_correction;
                    work_fc_prop(i) =
propeng_sfc(eng_id_cr)*usable_cross_work(1)+load_fraction2*usable_cross_wor
k(2)*work_sfc; % [gr/hr]
                    work_sfc_prop(i) =
work_fc_prop(i)/(usable_cross_work(1)+load_fraction2*usable_cross_work(2));
% [gr/KWh]
                    work_fc_avgss(i) = work_fc_prop(i)+work_fc_ss(7);
                    work_sfc_avgss(i) =
work_fc_avgss(i)/(usable_cross_work(1)+load_fraction2*usable_cross_work(2)+
ss_power(7)/n_gen); % [gr/KWh]
                elseif prop_power(i) <
sum(usable_cross_work(2:3))*n_trans_sp
                    % 1st sprint engine at full load, 2nd sprint engine
                    % at partial load
                    sfc_ind = SFCind(eng_id_sp,system);
                    load_fraction = (prop_power(i)-
usable_cross_work(2)*n_trans_sp)/(usable_cross_work(3)*n_trans_sp);
                    sfc_fraction = Lfraction(load_fraction);
                    load_fraction2 = n_trans_sp/(1-(1-
n_trans_sp)*loss_mult(sfc_fraction-4))*load_fraction;
```

```
                    plot_fraction_sp1(i) = 1;
                    plot_fraction_sp2(i) = load_fraction2;
                    work_sfc =
propeng_sfc(eng_id_sp)/gen_sfc_FL(sfc_ind)*partial_sfc(sfc_ind,sfc_fraction
-4)*LHV_correction;
                    work_fc_prop(i) =
propeng_sfc(eng_id_sp)*usable_cross_work(2)+load_fraction2*usable_cross_wor
k(3)*work_sfc; % [gr/hr]
                    work_sfc_prop(i) =
work_fc_prop(i)/(usable_cross_work(2)+load_fraction2*usable_cross_work(3));
% [gr/KWh]
                    work_fc_avgss(i) = work_fc_prop(i)+work_fc_ss(7);
                    work_sfc_avgss(i) =
work_fc_avgss(i)/(usable_cross_work(2)+load_fraction2*usable_cross_work(3)+
ss_power(7)/n_gen); % [gr/KWh]
                else
                    % Cruise engine and 1st sprint engine at full load, 2nd
                    % sprint engine at partial load
                    sfc_ind = SFCind(eng_id_sp,system);
                    load_fraction = (prop_power(i)-
usable_cross_work(1)*n_trans_cr-
usable_cross_work(2)*n_trans_sp)/(usable_cross_work(3)*n_trans_sp);
                    sfc_fraction = Lfraction(load_fraction);
                    load_fraction2 = n_trans_sp/(1-(1-
n_trans_sp)*loss_mult(sfc_fraction-4))*load_fraction;
                    plot_fraction_cr(i) = 1;
                    plot_fraction_sp1(i) = 1;
                    plot_fraction_sp2(i) = load_fraction2;
                    work_sfc =
propeng_sfc(eng_id_sp)/gen_sfc_FL(sfc_ind)*partial_sfc(sfc_ind,sfc_fraction
-4)*LHV_correction;
                    work_fc_prop(i) =
propeng_sfc(eng_id_cr)*usable_cross_work(1)+propeng_sfc(eng_id_sp)*usable_c
ross_work(2)+load_fraction2*usable_cross_work(3)*work_sfc; % [gr/hr]
                    work_sfc_prop(i) =
work_fc_prop(i)/(sum(usable_cross_work(1:2))+load_fraction2*usable_cross_wo
rk(3)); % [gr/KWh]
                    work_fc_avgss(i) = work_fc_prop(i)+work_fc_ss(7);
                    work_sfc_avgss(i) =
work_fc_avgss(i)/(sum(usable_cross_work(1:2))+load_fraction2*usable_cross_w
ork(3)+ss_power(7)/n_gen); % [gr/KWh]
                end
            else % 2 cruise engines and 1 sprint engine
                if prop_power(i) < usable_cross_work(1)*n_trans_cr
                    % Cruise engine at partial load
                    sfc_ind = SFCind(eng_id_cr,system);
                    load_fraction =
prop_power(i)/(usable_cross_work(1)*n_trans_cr);
                    sfc_fraction = Lfraction(load_fraction);
                    load_fraction2 = n_trans_cr/(1-(1-
n_trans_cr)*loss_mult(sfc_fraction-4))*load_fraction;
                    plot_fraction_cr1(i) = load_fraction2;
                    work_sfc =
propeng_sfc(eng_id_cr)/gen_sfc_FL(sfc_ind)*partial_sfc(sfc_ind,sfc_fraction
-4)*LHV_correction;
                    work_fc_prop(i) =
load_fraction2*usable_cross_work(1)*work_sfc; % [gr/hr]
                    work_sfc_prop(i) = work_sfc; % [gr/KWh]
```

```
                    work_fc_avgss(i) = work_fc_prop(i)+work_fc_ss(7);
                    work_sfc_avgss(i) =
work_fc_avgss(i)/(load_fraction2*usable_cross_work(1)+ss_power(7)/n_gen); %
[gr/KWh]
                elseif prop_power(i) <
sum(usable_cross_work(1:2))*n_trans_cr
                    % 1st cruise engine at full load, 2nd cruise engine at
                    % partial load
                    sfc_ind = SFCind(eng_id_cr,system);
                    load_fraction = (prop_power(i)-
usable_cross_work(1)*n_trans_cr)/(usable_cross_work(2)*n_trans_cr);
                    sfc_fraction = Lfraction(load_fraction);
                    load_fraction2 = n_trans_cr/(1-(1-
n_trans_cr)*loss_mult(sfc_fraction-4))*load_fraction;
                    plot_fraction_cr1(i) = 1;
                    plot_fraction_cr2(i) = load_fraction2;
                    work_sfc =
propeng_sfc(eng_id_cr)/gen_sfc_FL(sfc_ind)*partial_sfc(sfc_ind,sfc_fraction
-4)*LHV_correction;
                    work_fc_prop(i) =
propeng_sfc(eng_id_cr)*usable_cross_work(1)+load_fraction2*usable_cross_wor
k(2)*work_sfc; % [gr/hr]
                    work_sfc_prop(i) =
work_fc_prop(i)/(usable_cross_work(1)+load_fraction2*usable_cross_work(2));
% [gr/KWh]
                    work_fc_avgss(i) = work_fc_prop(i)+work_fc_ss(7);
                    work_sfc_avgss(i) =
work_fc_avgss(i)/(usable_cross_work(1)+load_fraction2*usable_cross_work(2)+
ss_power(7)/n_gen); % [gr/KWh]
                elseif prop_power(i) < usable_cross_work(3)*n_trans_sp
                    % Sprint engine at partial load
                    sfc_ind = SFCind(eng_id_sp,system);
                    load_fraction =
prop_power(i)/(usable_cross_work(3)*n_trans_sp);
                    sfc_fraction = Lfraction(load_fraction);
                    load_fraction2 = n_trans_sp/(1-(1-
n_trans_sp)*loss_mult(sfc_fraction-4))*load_fraction;
                    plot_fraction_sp(i) = load_fraction2;
                    work_sfc =
propeng_sfc(eng_id_sp)/gen_sfc_FL(sfc_ind)*partial_sfc(sfc_ind,sfc_fraction
-4)*LHV_correction;
                    work_fc_prop(i) =
load_fraction2*usable_cross_work(3)*work_sfc; % [gr/hr]
                    work_sfc_prop(i) = work_sfc; % [gr/KWh]
                    work_fc_avgss(i) = work_fc_prop(i)+work_fc_ss(7);
                    work_sfc_avgss(i) =
work_fc_avgss(i)/(load_fraction2*usable_cross_work(3)+ss_power(7)/n_gen); %
[gr/KWh]
                elseif prop_power(i) <
usable_cross_work(1)*n_trans_cr+usable_cross_work(3)*n_trans_sp
                    % 1st cruise engine at partial load, sprint engine
                    % at 100% load
                    sfc_ind = SFCind(eng_id_sp,system);
                    load_fraction = (prop_power(i)-
usable_cross_work(1)*n_trans_cr)/(usable_cross_work(3)*n_trans_sp);
                    sfc_fraction = Lfraction(load_fraction);
                    load_fraction2 = n_trans_sp/(1-(1-
n_trans_sp)*loss_mult(sfc_fraction-4))*load_fraction;
```

```
                    plot_fraction_cr1(i) = 1;
                    plot_fraction_sp(i) = load_fraction2;
                    work_sfc =
propeng_sfc(eng_id_sp)/gen_sfc_FL(sfc_ind)*partial_sfc(sfc_ind,sfc_fraction
-4)*LHV_correction;
                    work_fc_prop(i) =
propeng_sfc(eng_id_cr)*usable_cross_work(1)+load_fraction2*usable_cross_wor
k(3)*work_sfc; % [gr/hr]
                    work_sfc_prop(i) =
work_fc_prop(i)/(usable_cross_work(1)+load_fraction2*usable_cross_work(3));
% [gr/KWh]
                    work_fc_avgss(i) = work_fc_prop(i)+work_fc_ss(7);
                    work_sfc_avgss(i) =
work_fc_avgss(i)/(usable_cross_work(1)+load_fraction2*usable_cross_work(3)+
ss_power(7)/n_gen); % [gr/KWh]
                else
                    % Both ruise engines at full load,
                    % sprint engine at partial load
                    sfc_ind = SFCind(eng_id_sp,system);
                    load_fraction = (prop_power(i)-
sum(usable_cross_work(1:2))*n_trans_cr)/(usable_cross_work(3)*n_trans_sp);
                    sfc_fraction = Lfraction(load_fraction);
                    load_fraction2 = n_trans_sp/(1-(1-
n_trans_sp)*loss_mult(sfc_fraction-4))*load_fraction; % Engine load
increases due to increased transmission losses at partial load
                    plot_fraction_cr1(i) = 1;
                    plot_fraction_cr2(i) = 1;
                    plot_fraction_sp(i) = load_fraction2;
                    work_sfc =
propeng_sfc(eng_id_sp)/gen_sfc_FL(sfc_ind)*partial_sfc(sfc_ind,sfc_fraction
-4)*LHV_correction;
                    work_fc_prop(i) =
propeng_sfc(eng_id_cr)*sum(usable_cross_work(1:2))+load_fraction2*usable_cr
oss_work(3)*work_sfc; % [gr/hr]
                    work_sfc_prop(i) =
work_fc_prop(i)/(sum(usable_cross_work(1:2))+load_fraction2*usable_cross_wo
rk(3)); % [gr/KWh]
                    work_fc_avgss(i) = work_fc_prop(i)+work_fc_ss(7);
                    work_sfc_avgss(i) =
work_fc_avgss(i)/(sum(usable_cross_work(1:2))+load_fraction2*usable_cross_w
ork(3)+ss_power(7)/n_gen); % [gr/KWh]
                end
            end
        end
    end
end


%% Total fuel consumption for the given operating profile
fc_total_prop = sum(profile_data(2:end-1,end).*work_fc_prop/1000)/100; %
Propulsion fuel consumption [kg/hr]
fc_total_ss = sum(profile_data(end,2:end-2).*work_fc_ss/1000)/100; % Ship
service fuel consumption [kg/hr]
fc_total = fc_total_prop+fc_total_ss; % Total fuel consumption [kg/hr]

%% Display results
```

```
figure
plot(profile_data(2:end-1,1),work_fc_prop/1000,'r-'),xlabel('Speed
[knots]'),ylabel('Fuel Consumption [kg/h]'),title('Propulsion Fuel
Consumption Profile')
figure
plot(profile_data(1,2:end-2),work_fc_ss/1000,'g-'),xlabel('Ship-Service
Power [kW]'),ylabel('Fuel Consumption [kg/h]'),title('Ship-Service Fuel
Consumption Profile')

name_cr = char(propeng_name(eng_id_cr));
type_cr = char(propeng_type(eng_id_cr));
manuf_cr = char(propeng_manuf(eng_id_cr));
name_sp = char(propeng_name(eng_id_sp));
type_sp = char(propeng_type(eng_id_sp));
manuf_sp = char(propeng_manuf(eng_id_sp));
name_ss = char(sseng_name(ss_id));
type_ss = char(sseng_type(ss_id));
manuf_ss = char(sseng_manuf(ss_id));

disp('----------------------------------------------------------------------
--')
disp('MECHANICAL-DRIVEN PROPULSION WITH SEPARATE SHIP-SERVICE POWER')
fprintf('\t\t     OPTIMAL CONFIGURATION\n\n');
fprintf('Propulsion system: %s\n\n', sys_type);
fprintf('Cruise engines:       %d %s %s (%s) / %dkW\n', eng_num_cr,
name_cr, type_cr, manuf_cr,propeng_power(eng_id_cr));
fprintf('Sprint engines:       %d %s %s (%s) / %dkW\n', eng_num_sp,
name_sp, type_sp, manuf_sp,propeng_power(eng_id_sp));
fprintf('Ship service engines: %d %s %s (%s) / %dkW\n\n', ss_num, name_ss,
type_ss, manuf_ss,sseng_power(ss_id));
fprintf('\n')
fprintf('Average fuel consumption for given combined profile: %5.0f
kg/hr\n', fc_total);
disp('----------------------------------------------------------------------
--')

% Plot propulsion engines load fractions
figure
if strcmp(sys_type,'Cross-connected') == 0
    plot(profile_data(2:end-1,1),plot_fraction_cr,'g',profile_data(2:end-
1,1),plot_fraction_sp,'r'),xlabel('Speed [knots]'),ylabel('Engine Load
Fraction'),legend(strcat(name_cr,' (Cruise)'),strcat(name_sp,'
(Sprint)')),title('Propulsion Engines Load Fractions')
else
    if eng_num_cr == 1
        plot(profile_data(2:end-
1,1),plot_fraction_cr,'b',profile_data(2:end-
1,1),plot_fraction_sp1,'g',profile_data(2:end-
1,1),plot_fraction_sp2,'r'),xlabel('Speed [knots]'),ylabel('Engine Load
Fraction'),legend((name_cr),strcat('1st ',name_sp),strcat('2nd
',name_sp)),title('Propulsion Engines Load Fractions')
    else
        plot(profile_data(2:end-
1,1),plot_fraction_cr1,'b',profile_data(2:end-
1,1),plot_fraction_cr2,'g',profile_data(2:end-
1,1),plot_fraction_sp,'r'),xlabel('Speed [knots]'),ylabel('Engine Load
Fraction'),legend(strcat('1st ',name_cr),strcat('2nd
',name_cr),(name_sp)),title('Propulsion Engines Load Fractions')
```

```
    end
end

% Plot ship-service engines load fractions
figure
plot(profile_data(1,2:end-2),plot_fraction_ss1,'b'),xlabel('Ship Service
Power [kW]'),ylabel('Engine Load Fraction'),legend('SS engines 1 and
2'),title('Ship-Service Engines Load Fractions')
hold on
if ss_num == 4
    plot(profile_data(1,2:end-2),plot_fraction_ss3,'r'),legend('SS engines
1 and 2','SS engine 3')
end
hold off

% Plot propulsion sfc
figure
plot(profile_data(2:end-1,1),work_sfc_prop,'r')
xlabel('Speed [knots]')
ylabel('Specific Fuel Consumption [gr/KWh]')
```

Function *MechPDSSevalf.m*:

```
function [work_fc,work_sfc] = MechPDSSevalf(x)
%% Function evaluates mechanical-driven propulsion systems with propulsion
derived ship-service power
% Separate ship service systems can still be present

global profile_data dspeed dss SHP_2 gen_sfc_FL partial_sfc LHV_correction
loss_mult M_s...
    propeng_type sseng_type propeng_sfc sseng_sfc propeng_power sseng_power
propeng_name sseng_name propeng_manuf sseng_manuf


%% Propulsion engines type and number selection
eng_id_cr = 1+Convert( x, 1, 6 ); % Cruise engine selection ID
eng_id_sp = 1+Convert( x, 7, 6 ); % Sprint engine selection ID

if propeng_power(eng_id_sp) < propeng_power(eng_id_cr)
    temp = eng_id_cr;
    eng_id_cr = eng_id_sp;
    eng_id_sp = temp;
end

eng_num_cr = 1+Convert( x, 13, 1 ); % Number of cruise engines

% Determine number of sprint engines
if eng_num_cr == 1
    eng_num_sp = 2;
else
    eng_num_sp = 1+Convert( x, 14, 1 );
end

comb_id = Convert( x, 15, 1 ); % OR/AND (0/1)

%% Mechanical driven propulsion system identification
% CODAG systems represent cross-connected configurations with 3 prime
movers
if strcmp(propeng_type(eng_id_cr),'DIESEL')==1 &&
strcmp(propeng_type(eng_id_sp),'DIESEL')==1 && eng_num_cr == eng_num_sp
    sys_type = 'CODAD';
elseif strcmp(propeng_type(eng_id_cr),'DIESEL')==1 &&
strcmp(propeng_type(eng_id_sp),'DIESEL')==0 && eng_num_cr == eng_num_sp &&
comb_id == 0
    sys_type = 'CODOG';
elseif strcmp(propeng_type(eng_id_cr),'DIESEL')==1 &&
strcmp(propeng_type(eng_id_sp),'DIESEL')==0 && eng_num_cr == eng_num_sp &&
comb_id == 1
    sys_type = 'CODAG';
elseif strcmp(propeng_type(eng_id_cr),'DIESEL')==0 &&
strcmp(propeng_type(eng_id_sp),'DIESEL')==0 && eng_num_cr == eng_num_sp
    sys_type = 'COGAG';
else
    sys_type = 'Cross-connected';
end
```

```
%% Propulsive power availability
% All configurations
usable_HP_cr = eng_num_cr*propeng_power(eng_id_cr); % [kW]
usable_HP_sp = (1-
strcmp(sys_type,'CODOG'))*usable_HP_cr+eng_num_sp*propeng_power(eng_id_sp);
% [kW], Only sprint engines if CODOG

% Cross-connected configurations
usable_cross_work = zeros(eng_num_cr+eng_num_sp,1);
if strcmp(sys_type,'Cross-connected') == 1
    for i_cross = 1:eng_num_cr;
    usable_cross_work(i_cross) = propeng_power(eng_id_cr);
    end
    for i_cross = eng_num_cr+1:eng_num_cr+eng_num_sp
        usable_cross_work(i_cross) = propeng_power(eng_id_sp);
    end
end


%% Mechanical transmission efficiencies
% Full-load transmission efficiencies
% GT transmission losses bigger due to 2 stage reduction required
if strcmp(sys_type,'Cross-connected')
    trans_mult = .99;
else
    trans_mult = 1;
end

if strcmp(propeng_type(eng_id_cr),'GT') == 1
    n_trans_cr = 0.98*trans_mult;
else
    n_trans_cr = 0.99*trans_mult;
end

if strcmp(propeng_type(eng_id_sp),'GT') == 1
    n_trans_sp = 0.98*trans_mult;
else
    n_trans_sp = 0.99*trans_mult;
end


%% Separate ship service engine type and number selection if present
sep_ss_dec = Convert( x, 16, 1 ); % Separate ship service system: 0/1 =
No/Yes
sep_ss_id = 1+Convert( x, 17, 6 ); % Separate ship service engine selection
ID
sep_ss_num = 2+Convert( x, 23, 1 ); % Number of separate ship service
engines (2 or 3 allowed)

%% Ship service power and efficiency calculations
% Ship service power generation efficiencies
n_gen = 0.96;
system = 'ss';
sfc_ss_ind = SFCind(sep_ss_id,system);

% Ship service power availability for different number of engines on-line
```

```
% N-1 engines
usable_HP_ss_work = zeros(sep_ss_num-1,1);
usable_HP_ss_sep = 0;
if sep_ss_dec == 1
    for i_gen = 1:sep_ss_num-1
        usable_HP_ss_work(i_gen) = sseng_power(sep_ss_id);
    end
    usable_HP_ss_sep = sum(usable_HP_ss_work);
end


% Propulsion derived ship service efficiency
n_ss_pd = 0.915; % VSCF cycloconverter efficiency, ASSET value


%% Propulsion and propulsion derived ship service systems fuel consumption
calculations
% Load_fraction_2: Engine working load fraction (taking into account
transmission losses at partial load)
i_speed = length(profile_data(2:end-1,1));
i_sspower = length(profile_data(1,2:end-2));
% Check that propulsion and ship-service configuration provides the
required HP
if usable_HP_sp*(1-
M_s)*min([n_trans_cr,n_trans_sp])+usable_HP_ss_sep*n_ss_pd <
dss+SHP_2(dspeed+1) || usable_HP_sp*(1-M_s)*min([n_trans_cr,n_trans_sp]) <
SHP_2(dspeed+1)
    disp('Warning: Selected propulsion and ship-service configuration does
not meet power requirements')
else
    system = 'prop';
    % Pre-allocate
    prop_power = zeros(i_speed,1);
    ss_power = profile_data(1,2:end-2);
    work_fc = zeros(i_speed,i_sspower); % lin=prop, col=ss
    work_sfc = zeros(i_speed,i_sspower);
    usable_HP_pd = zeros(i_speed,i_sspower); % Usable PD HP for each speed/
ss power demand combination
    plot_fraction_cr1 = zeros(i_speed,i_sspower);
    plot_fraction_cr2 = zeros(i_speed,i_sspower);
    plot_fraction_sp1 = zeros(i_speed,i_sspower);
    plot_fraction_sp2 = zeros(i_speed,i_sspower);
    plot_fraction_cr = zeros(i_speed,i_sspower);
    plot_fraction_sp = zeros(i_speed,i_sspower);
    plot_fraction_ss1 = zeros(i_speed,i_sspower);
    plot_fraction_ss2 = zeros(i_speed,i_sspower);
    for i = 1:i_speed
        prop_power(i) = SHP_2(i+1); % SHP_2 starts with speed=0, prop_power
starts with speed=1knot
        for j = 1:i_sspower
            if strcmp(sys_type,'Cross-connected') == 0
                if prop_power(i)/n_trans_cr+ss_power(j)/n_ss_pd <
usable_HP_cr && prop_power(i)/n_trans_cr < usable_HP_cr
                    % Cruise engines at partial load
                    usable_HP_pd(i,j) = usable_HP_cr-
prop_power(i)/n_trans_cr;
                    sfc_ind = SFCind(eng_id_cr,system);
```

```
                    load_fraction =
(prop_power(i)/n_trans_cr+ss_power(j)/n_ss_pd)/usable_HP_cr;
                    sfc_fraction = Lfraction(load_fraction);
                    load_fraction2 = n_trans_cr/(1-(1-
n_trans_cr)*loss_mult(sfc_fraction-4))*load_fraction;
                    plot_fraction_cr(i,j) = load_fraction2;
                    work_sfc_cur =
propeng_sfc(eng_id_cr)/gen_sfc_FL(sfc_ind)*partial_sfc(sfc_ind,sfc_fraction
-4)*LHV_correction;
                    % Total working fuel consumption
                    work_fc(i,j) =
load_fraction2*usable_HP_cr*work_sfc_cur; % [gr/hr]
                    work_sfc(i,j) = work_sfc_cur; % [gr/KWh]
                elseif prop_power(i)/n_trans_cr+ss_power(j)/n_ss_pd <
usable_HP_cr+usable_HP_ss_work(1) && prop_power(i)/n_trans_cr <
usable_HP_cr
                    % Cruise engines at full load, ss engine working at
partial load
                    usable_HP_pd(i,j) = usable_HP_cr-
prop_power(i)/n_trans_cr;
                    load_fraction = (ss_power(j)-
usable_HP_pd(i,j)*n_ss_pd)/n_gen/usable_HP_ss_work(1);
                    plot_fraction_cr(i,j) = 1;
                    plot_fraction_ss1(i,j) = load_fraction;
                    sfc_fraction = Lfraction(load_fraction);
                    work_sfc_cur =
sseng_sfc(sep_ss_id)/gen_sfc_FL(sfc_ss_ind)*partial_sfc(sfc_ss_ind,sfc_frac
tion-4)*LHV_correction;
                    % Total working fuel consumption
                    work_fc(i,j) =
usable_HP_cr*propeng_sfc(eng_id_cr)+load_fraction*usable_HP_ss_work(1)*work
_sfc_cur; % [gr/hr]
                    work_sfc(i,j) =
work_fc(i,j)/(usable_HP_cr+load_fraction*usable_HP_ss_work(1)); % [gr/KWh]
                elseif prop_power(i)/n_trans_cr+ss_power(j)/n_ss_pd <
usable_HP_cr+sum(usable_HP_ss_work) && prop_power(i)/n_trans_cr <
usable_HP_cr
                    % Cruise and 1st ss engines at full load, 2nd ss engine
at partial load
                    usable_HP_pd(i,j) = usable_HP_cr-
prop_power(i)/n_trans_cr;
                    load_fraction = (ss_power(j)-usable_HP_pd(i,j)*n_ss_pd-
usable_HP_ss_work(1)*n_gen)/n_gen/usable_HP_ss_work(2);
                    plot_fraction_cr(i,j) = 1;
                    plot_fraction_ss1(i,j) = 1;
                    plot_fraction_ss2(i,j) = load_fraction;
                    sfc_fraction = Lfraction(load_fraction);
                    work_sfc_cur =
sseng_sfc(sep_ss_id)/gen_sfc_FL(sfc_ss_ind)*partial_sfc(sfc_ss_ind,sfc_frac
tion-4)*LHV_correction;
                    % Total working fuel consumption
                    work_fc(i,j) =
usable_HP_cr*propeng_sfc(eng_id_cr)+usable_HP_ss_work(1)*sseng_sfc(sep_ss_i
d)+load_fraction*usable_HP_ss_work(2)*work_sfc_cur; % [gr/hr]
                    work_sfc(i,j) =
work_fc(i,j)/(usable_HP_cr+usable_HP_ss_work(1)+load_fraction*usable_HP_ss_
work(2)); % [gr/KWh]
                elseif prop_power(i)/n_trans_sp+ss_power(j)/n_ss_pd <
```

```
usable_HP_sp && prop_power(i)/n_trans_sp < usable_HP_sp
                    if strcmp(sys_type,'CODOG')
                        % Sprint engines at partial load
                        usable_HP_pd(i,j) = usable_HP_sp-
prop_power(i)/n_trans_sp;
                        load_fraction =
(prop_power(i)/n_trans_sp+ss_power(j)/n_ss_pd)/usable_HP_sp;
                        sfc_fraction = Lfraction(load_fraction);
                        load_fraction2 = n_trans_sp/(1-(1-
n_trans_sp)*loss_mult(sfc_fraction-4))*load_fraction;
                        plot_fraction_sp(i,j) = load_fraction2;
                        work_sfc_cur =
propeng_sfc(eng_id_sp)/gen_sfc_FL(5)*partial_sfc(5,sfc_fraction-
4)*LHV_correction;
                        % Total working fuel consumption
                        work_fc(i,j) =
load_fraction2*usable_HP_sp*work_sfc_cur; % [gr/hr]
                        work_sfc(i,j) = work_sfc_cur; % [gr/KWh]
                    else
                        % Cruise engines at full load, sprint engines at
partial load
                        usable_HP_pd(i,j) = usable_HP_sp-
prop_power(i)/n_trans_sp;
                        sfc_ind = SFCind(eng_id_sp,system);
                        load_fraction = ((prop_power(i)-
usable_HP_cr*n_trans_cr)/n_trans_sp+ss_power(j)/n_ss_pd)/(usable_HP_sp-
usable_HP_cr);
                        sfc_fraction = Lfraction(load_fraction);
                        load_fraction2 = n_trans_sp/(1-(1-
n_trans_sp)*loss_mult(sfc_fraction-4))*load_fraction;
                        plot_fraction_cr(i,j) = 1;
                        plot_fraction_sp(i,j) = load_fraction2;
                        work_sfc_cur =
propeng_sfc(eng_id_sp)/gen_sfc_FL(sfc_ind)*partial_sfc(sfc_ind,sfc_fraction
-4)*LHV_correction;
                        % Total working fuel consumption
                        work_fc(i,j) =
propeng_sfc(eng_id_cr)*usable_HP_cr+load_fraction2*usable_HP_sp*work_sfc_cu
r; % [gr/hr]
                        work_sfc(i,j) =
work_fc(i,j)/(usable_HP_cr+load_fraction2*(usable_HP_sp-usable_HP_cr)); %
[gr/KWh]
                    end
                elseif prop_power(i)/n_trans_sp+ss_power(j)/n_ss_pd <
usable_HP_sp+usable_HP_ss_work(1) && prop_power(i)/n_trans_sp <
usable_HP_sp
                    if strcmp(sys_type,'CODOG')
                        % Sprint engines at full load, ss engine at partial
load
                        usable_HP_pd(i,j) = usable_HP_sp-
prop_power(i)/n_trans_sp;
                        load_fraction = (ss_power(j)-
usable_HP_pd(i,j)*n_ss_pd)/n_gen/usable_HP_ss_work(1);
                        plot_fraction_sp(i,j) = 1;
                        plot_fraction_ss1(i,j) = load_fraction;
                        sfc_fraction = Lfraction(load_fraction);
                        work_sfc_cur =
sseng_sfc(sep_ss_id)/gen_sfc_FL(sfc_ss_ind)*partial_sfc(sfc_ss_ind,sfc_frac
```

```
tion-4)*LHV_correction;
                        % Total working fuel consumption
                        work_fc(i,j) =
usable_HP_sp*propeng_sfc(eng_id_sp)+load_fraction*usable_HP_ss_work(1)*work
_sfc_cur; % [gr/hr]
                        work_sfc(i,j) =
work_fc(i,j)/(usable_HP_sp+load_fraction*usable_HP_ss_work(1)); % [gr/KWh]
                    else
                        % Cruise and sprint engines at full load, ss engine
at partial load
                        usable_HP_pd(i,j) = usable_HP_cr*(n_trans_cr-
n_trans_sp)/n_trans_sp+usable_HP_sp-prop_power(i)/n_trans_sp;
                        load_fraction = (ss_power(j)-
usable_HP_pd(i,j)*n_ss_pd)/n_gen/usable_HP_ss_work(1);
                        plot_fraction_cr(i,j) = 1;
                        plot_fraction_sp(i,j) = 1;
                        plot_fraction_ss1(i,j) = load_fraction;
                        sfc_fraction = Lfraction(load_fraction);
                        work_sfc_cur =
sseng_sfc(sep_ss_id)/gen_sfc_FL(sfc_ss_ind)*partial_sfc(sfc_ss_ind,sfc_frac
tion-4)*LHV_correction;
                        % Total working fuel consumption
                        work_fc(i,j) =
usable_HP_sp*propeng_sfc(eng_id_sp)+(usable_HP_sp-
usable_HP_cr)*propeng_sfc(eng_id_cr)+load_fraction*usable_HP_ss_work(1)*wor
k_sfc_cur; % [gr/hr]
                        work_sfc(i,j) =
work_fc(i,j)/(usable_HP_sp+load_fraction*usable_HP_ss_work(1)); % [gr/KWh]
                    end
                elseif prop_power(i)/n_trans_sp+ss_power(j)/n_ss_pd <
usable_HP_sp+sum(usable_HP_ss_work) && prop_power(i)/n_trans_sp <
usable_HP_sp
                    if strcmp(sys_type,'CODOG')
                        % Sprint and 1st ss engines at full load, 2nd ss
engine at partial load
                        usable_HP_pd(i,j) = usable_HP_sp-
prop_power(i)/n_trans_sp;
                        load_fraction = (ss_power(j)-
usable_HP_pd(i,j)*n_ss_pd-
usable_HP_ss_work(1)*n_gen)/n_gen/usable_HP_ss_work(2);
                        plot_fraction_sp(i,j) = 1;
                        plot_fraction_ss1(i,j) = 1;
                        plot_fraction_ss2(i,j) = load_fraction;
                        sfc_fraction = Lfraction(load_fraction);
                        work_sfc_cur =
sseng_sfc(sep_ss_id)/gen_sfc_FL(sfc_ss_ind)*partial_sfc(sfc_ss_ind,sfc_frac
tion-4)*LHV_correction;
                        % Total working fuel consumption
                        work_fc(i,j) =
usable_HP_sp*propeng_sfc(eng_id_sp)+usable_HP_ss_work(1)*sseng_sfc(sep_ss_i
d)+load_fraction*usable_HP_ss_work(2)*work_sfc_cur; % [gr/hr]
                        work_sfc(i,j) =
work_fc(i,j)/(usable_HP_sp+usable_HP_ss_work(1)+load_fraction*usable_HP_ss_
work(2)); % [gr/KWh]
                    else
                        % Cruise, sprint and 1st ss engines at full load,
2nd ss engine at partial load
                        usable_HP_pd(i,j) = usable_HP_cr*(n_trans_cr-
```

```
n_trans_sp)/n_trans_sp+usable_HP_sp-prop_power(i)/n_trans_sp;
                        load_fraction = (ss_power(j)-
usable_HP_pd(i,j)*n_ss_pd-
usable_HP_ss_work(1)*n_gen)/n_gen/usable_HP_ss_work(2);
                        plot_fraction_cr(i,j) = 1;
                        plot_fraction_sp(i,j) = 1;
                        plot_fraction_ss1(i,j) = 1;
                        plot_fraction_ss2(i,j) = load_fraction;
                        sfc_fraction = Lfraction(load_fraction);
                        work_sfc_cur =
sseng_sfc(sep_ss_id)/gen_sfc_FL(sfc_ss_ind)*partial_sfc(sfc_ss_ind,sfc_frac
tion-4)*LHV_correction;
                        % Total working fuel consumption
                        work_fc(i,j) =
usable_HP_sp*propeng_sfc(eng_id_sp)+(usable_HP_sp-
usable_HP_cr)*propeng_sfc(eng_id_cr)+usable_HP_ss_work(1)*sseng_sfc(sep_ss_
id)+load_fraction*usable_HP_ss_work(2)*work_sfc_cur; % [gr/hr]
                        work_sfc(i,j) =
work_fc(i,j)/(usable_HP_sp+usable_HP_ss_work(1)+load_fraction*usable_HP_ss_
work(2)); % [gr/KWh]
                    end
                end
            else % Cross-connected configurations
                if eng_num_cr == 1 % 1 cruise engine and 2 sprint engines
                    if prop_power(i)/n_trans_cr+ss_power(j)/n_ss_pd <
usable_cross_work(1)
                        % Cruise engine at partial load
                        usable_HP_pd(i,j) = usable_cross_work(1)-
prop_power(i)/n_trans_cr;
                        sfc_ind = SFCind(eng_id_cr,system);
                        load_fraction =
(prop_power(i)/n_trans_cr+ss_power(j)/n_ss_pd)/usable_cross_work(1);
                        sfc_fraction = Lfraction(load_fraction);
                        load_fraction2 = n_trans_cr/(1-(1-
n_trans_cr)*loss_mult(sfc_fraction-4))*load_fraction;
                        plot_fraction_cr(i,j) = load_fraction2;
                        work_sfc_cur =
propeng_sfc(eng_id_cr)/gen_sfc_FL(sfc_ind)*partial_sfc(sfc_ind,sfc_fraction
-4)*LHV_correction;
                        % Working fuel consumption
                        work_fc(i,j) =
load_fraction2*usable_cross_work(1)*work_sfc_cur; % [gr/hr]
                        work_sfc(i,j) = work_sfc_cur; % [gr/KWh]
                    elseif prop_power(i)/n_trans_cr+ss_power(j)/n_ss_pd <
usable_cross_work(1)+usable_HP_ss_work(1) && prop_power(i)/n_trans_cr <
usable_cross_work(1)
                        % Cruise engine at full load and 1st ss engine at
                        % partial load
                        usable_HP_pd(i,j) = usable_cross_work(1)-
prop_power(i)/n_trans_cr;
                        load_fraction = (ss_power(j)-
usable_HP_pd(i,j)*n_ss_pd)/n_gen/usable_HP_ss_work(1);
                        plot_fraction_cr(i,j) = 1;
                        plot_fraction_ss1(i,j) = load_fraction;
                        sfc_fraction = Lfraction(load_fraction);
                        work_sfc_cur =
sseng_sfc(sep_ss_id)/gen_sfc_FL(sfc_ss_ind)*partial_sfc(sfc_ss_ind,sfc_frac
tion-4)*LHV_correction;
```

```
                        % Total working fuel consumption
                        work_fc(i,j) =
usable_cross_work(1)*propeng_sfc(eng_id_cr)+load_fraction*usable_HP_ss_work
(1)*work_sfc_cur; % [gr/hr]
                        work_sfc(i,j) =
work_fc(i,j)/(usable_cross_work(1)+load_fraction*usable_HP_ss_work(1)); %
[gr/KWh]
                    elseif prop_power(i)/n_trans_cr+ss_power(j)/n_ss_pd <
usable_cross_work(1)+sum(usable_HP_ss_work) && prop_power(i)/n_trans_cr <
usable_cross_work(1)
                        % Cruise and 1st ss engines at full load, 2nd ss
engine at partial load
                        usable_HP_pd(i,j) = usable_cross_work(1)-
prop_power(i)/n_trans_cr;
                        load_fraction = (ss_power(j)-
usable_HP_pd(i,j)*n_ss_pd-
usable_HP_ss_work(1)*n_gen)/n_gen/usable_HP_ss_work(2);
                        plot_fraction_cr(i,j) = 1;
                        plot_fraction_ss1(i,j) = 1;
                        plot_fraction_ss2(i,j) = load_fraction;
                        sfc_fraction = Lfraction(load_fraction);
                        work_sfc_cur =
sseng_sfc(sep_ss_id)/gen_sfc_FL(sfc_ss_ind)*partial_sfc(sfc_ss_ind,sfc_frac
tion-4)*LHV_correction;
                        % Total working fuel consumption
                        work_fc(i,j) =
usable_cross_work(1)*propeng_sfc(eng_id_cr)+usable_HP_ss_work(1)*sseng_sfc(
sep_ss_id)+load_fraction*usable_HP_ss_work(2)*work_sfc_cur; % [gr/hr]
                        work_sfc(i,j) =
work_fc(i,j)/(usable_cross_work(1)+usable_HP_ss_work(1)+load_fraction*usabl
e_HP_ss_work(2)); % [gr/KWh]
                    elseif prop_power(i)/n_trans_sp+ss_power(j)/n_ss_pd <
usable_cross_work(2)
                        % 1st sprint engine at partial load
                        usable_HP_pd(i,j) = usable_cross_work(2)-
prop_power(i)/n_trans_sp;
                        sfc_ind = SFCind(eng_id_sp,system);
                        load_fraction =
(prop_power(i)/n_trans_sp+ss_power(j)/n_ss_pd)/usable_cross_work(2);
                        sfc_fraction = Lfraction(load_fraction);
                        load_fraction2 = n_trans_sp/(1-(1-
n_trans_sp)*loss_mult(sfc_fraction-4))*load_fraction;
                        plot_fraction_sp1(i,j) = load_fraction2;
                        work_sfc_cur =
propeng_sfc(eng_id_sp)/gen_sfc_FL(sfc_ind)*partial_sfc(sfc_ind,sfc_fraction
-4)*LHV_correction;
                        % Working fuel consumption
                        work_fc(i,j) =
load_fraction2*usable_cross_work(2)*work_sfc_cur; % [gr/hr]
                        work_sfc(i,j) = work_sfc_cur; % [gr/KWh]
                    elseif prop_power(i)/n_trans_sp+ss_power(j)/n_ss_pd <
usable_cross_work(2)+usable_HP_ss_work(1) && prop_power(i)/n_trans_sp <
usable_cross_work(2)
                        % 1st sprint engine at full load, 1st ss engine at
partial load
                        usable_HP_pd(i,j) = usable_cross_work(2)-
prop_power(i)/n_trans_sp;
                        load_fraction = (ss_power(j)-
```

```
usable_HP_pd(i,j)*n_ss_pd)/n_gen/usable_HP_ss_work(1);
                        plot_fraction_sp1(i,j) = 1;
                        plot_fraction_ss1(i,j) = load_fraction;
                        sfc_fraction = Lfraction(load_fraction);
                        work_sfc_cur =
sseng_sfc(sep_ss_id)/gen_sfc_FL(sfc_ss_ind)*partial_sfc(sfc_ss_ind,sfc_frac
tion-4)*LHV_correction;
                        % Total working fuel consumption
                        work_fc(i,j) =
usable_cross_work(2)*propeng_sfc(eng_id_sp)+load_fraction*usable_HP_ss_work
(1)*work_sfc_cur; % [gr/hr]
                        work_sfc(i,j) =
work_fc(i,j)/(usable_cross_work(2)+load_fraction*usable_HP_ss_work(1)); %
[gr/KWh]
                    elseif prop_power(i)/n_trans_sp+ss_power(j)/n_ss_pd <
usable_cross_work(2)+sum(usable_HP_ss_work) && prop_power(i)/n_trans_sp <
usable_cross_work(2)
                        % 1st sprint and 1st ss engines at full load, 2nd
ss engine at partial load
                        usable_HP_pd(i,j) = usable_cross_work(2)-
prop_power(i)/n_trans_sp;
                        load_fraction = (ss_power(j)-
usable_HP_pd(i,j)*n_ss_pd-
usable_HP_ss_work(1)*n_gen)/n_gen/usable_HP_ss_work(2);
                        plot_fraction_sp1(i,j) = 1;
                        plot_fraction_ss1(i,j) = 1;
                        plot_fraction_ss2(i,j) = load_fraction;
                        sfc_fraction = Lfraction(load_fraction);
                        work_sfc_cur =
sseng_sfc(sep_ss_id)/gen_sfc_FL(sfc_ss_ind)*partial_sfc(sfc_ss_ind,sfc_frac
tion-4)*LHV_correction;
                        % Total working fuel consumption
                        work_fc(i,j) =
usable_cross_work(2)*propeng_sfc(eng_id_sp)+usable_HP_ss_work(1)*sseng_sfc(
sep_ss_id)+load_fraction*usable_HP_ss_work(2)*work_sfc_cur; % [gr/hr]
                        work_sfc(i,j) =
work_fc(i,j)/(usable_cross_work(2)+usable_HP_ss_work(1)+load_fraction*usabl
e_HP_ss_work(2)); % [gr/KWh]
                    elseif prop_power(i)/n_trans_sp+ss_power(j)/n_ss_pd <
sum(usable_cross_work(1:2))
                        % Cruise engine at full load, 1st sprint engine at
partial load
                        usable_HP_pd(i,j) =
usable_cross_work(1)*(n_trans_cr-
n_trans_sp)/n_trans_sp+sum(usable_cross_work(1:2))-
prop_power(i)/n_trans_sp;
                        sfc_ind = SFCind(eng_id_sp,system);
                        load_fraction = ((prop_power(i)-
usable_cross_work(1)*n_trans_cr)/n_trans_sp+ss_power(j)/n_ss_pd)/usable_cro
ss_work(2);
                        sfc_fraction = Lfraction(load_fraction);
                        load_fraction2 = n_trans_sp/(1-(1-
n_trans_sp)*loss_mult(sfc_fraction-4))*load_fraction;
                        plot_fraction_cr(i,j) = 1;
                        plot_fraction_sp1(i,j) = load_fraction2;
                        work_sfc_cur =
propeng_sfc(eng_id_sp)/gen_sfc_FL(sfc_ind)*partial_sfc(sfc_ind,sfc_fraction
-4)*LHV_correction;
```

```
                        % Total working fuel consumption
                        work_fc(i,j) =
propeng_sfc(eng_id_cr)*usable_cross_work(1)+load_fraction2*usable_cross_wor
k(2)*work_sfc_cur; % [gr/hr]
                        work_sfc(i,j) =
work_fc(i,j)/(usable_cross_work(1)+load_fraction2*usable_cross_work(2)); %
[gr/KWh]
                    elseif prop_power(i)/n_trans_sp+ss_power(j)/n_ss_pd <
sum(usable_cross_work(1:2))+usable_HP_ss_work(1) &&
prop_power(i)/n_trans_sp < sum(usable_cross_work(1:2))
                        % Cruise and 1st sprint engines at full load, 1st
ss engine at partial load
                        usable_HP_pd(i,j) =
usable_cross_work(1)*(n_trans_cr-
n_trans_sp)/n_trans_sp+sum(usable_cross_work(1:2))-
prop_power(i)/n_trans_sp;
                        load_fraction = (ss_power(j)-
usable_HP_pd(i,j)*n_ss_pd)/n_gen/usable_HP_ss_work(1);
                        plot_fraction_cr(i,j) = 1;
                        plot_fraction_sp1(i,j) = 1;
                        plot_fraction_ss1(i,j) = load_fraction;
                        sfc_fraction = Lfraction(load_fraction);
                        work_sfc_cur =
sseng_sfc(sep_ss_id)/gen_sfc_FL(sfc_ss_ind)*partial_sfc(sfc_ss_ind,sfc_frac
tion-4)*LHV_correction;
                        % Total working fuel consumption
                        work_fc(i,j) =
propeng_sfc(eng_id_cr)*usable_cross_work(1)+usable_cross_work(2)*propeng_sf
c(eng_id_sp)+load_fraction*usable_HP_ss_work(1)*work_sfc_cur; % [gr/hr]
                        work_sfc(i,j) =
work_fc(i,j)/(usable_cross_work(1)+usable_cross_work(2)+load_fraction*usabl
e_HP_ss_work(1)); % [gr/KWh]
                    elseif prop_power(i)/n_trans_sp+ss_power(j)/n_ss_pd <
sum(usable_cross_work(1:2))+sum(usable_HP_ss_work) &&
prop_power(i)/n_trans_sp < sum(usable_cross_work(1:2))
                        % Cruise, 1st sprint and 1st ss engines at full
load, 2nd ss engine at partial load
                        usable_HP_pd(i,j) =
usable_cross_work(1)*(n_trans_cr-
n_trans_sp)/n_trans_sp+sum(usable_cross_work(1:2))-
prop_power(i)/n_trans_sp;
                        load_fraction = (ss_power(j)-
usable_HP_pd(i,j)*n_ss_pd-
usable_HP_ss_work(1)*n_gen)/n_gen/usable_HP_ss_work(2);
                        plot_fraction_cr(i,j) = 1;
                        plot_fraction_sp1(i,j) = 1;
                        plot_fraction_ss1(i,j) = 1;
                        plot_fraction_ss2(i,j) = load_fraction;
                        sfc_fraction = Lfraction(load_fraction);
                        work_sfc_cur =
sseng_sfc(sep_ss_id)/gen_sfc_FL(sfc_ss_ind)*partial_sfc(sfc_ss_ind,sfc_frac
tion-4)*LHV_correction;
                        % Total working fuel consumption
                        work_fc(i,j) =
propeng_sfc(eng_id_cr)*usable_cross_work(1)+usable_cross_work(2)*propeng_sf
c(eng_id_sp)+usable_HP_ss_work(1)*sseng_sfc(sep_ss_id)+load_fraction*usable
_HP_ss_work(2)*work_sfc_cur; % [gr/hr]
                        work_sfc(i,j) =
```

```
work_fc(i,j)/(usable_cross_work(1)+usable_cross_work(2)+usable_HP_ss_work(1
)+load_fraction*usable_HP_ss_work(2)); % [gr/KWh]
                    elseif prop_power(i)/n_trans_sp+ss_power(j)/n_ss_pd <
sum(usable_cross_work(2:3))
                        % 1st sprint engine at full load, 2nd at partial
                        % load
                        usable_HP_pd(i,j) = sum(usable_cross_work(2:3))-
prop_power(i)/n_trans_sp;
                        sfc_ind = SFCind(eng_id_sp,system);
                        load_fraction = ((prop_power(i)-
usable_cross_work(2)*n_trans_sp)/n_trans_sp+ss_power(j)/n_ss_pd)/usable_cro
ss_work(3);
                        sfc_fraction = Lfraction(load_fraction);
                        load_fraction2 = n_trans_sp/(1-(1-
n_trans_sp)*loss_mult(sfc_fraction-4))*load_fraction;
                        plot_fraction_sp1(i,j) = 1;
                        plot_fraction_sp2(i,j) = load_fraction2;
                        work_sfc_cur =
propeng_sfc(eng_id_sp)/gen_sfc_FL(sfc_ind)*partial_sfc(sfc_ind,sfc_fraction
-4)*LHV_correction;
                        % Total working fuel consumption
                        work_fc(i,j) =
propeng_sfc(eng_id_sp)*usable_cross_work(2)+load_fraction2*usable_cross_wor
k(3)*work_sfc_cur; % [gr/hr]
                        work_sfc(i,j) =
work_fc(i,j)/(usable_cross_work(2)+load_fraction2*usable_cross_work(3)); %
[gr/KWh]
                    elseif prop_power(i)/n_trans_sp+ss_power(j)/n_ss_pd <
sum(usable_cross_work(2:3))+usable_HP_ss_work(1) &&
prop_power(i)/n_trans_sp < sum(usable_cross_work(2:3))
                        % Both sprint engines at full load, 1st ss engine
at partial load
                        usable_HP_pd(i,j) = sum(usable_cross_work(2:3))-
prop_power(i)/n_trans_sp;
                        load_fraction = (ss_power(j)-
usable_HP_pd(i,j)*n_ss_pd)/n_gen/usable_HP_ss_work(1);
                        plot_fraction_sp1(i,j) = 1;
                        plot_fraction_sp2(i,j) = 1;
                        plot_fraction_ss1(i,j) = load_fraction;
                        sfc_fraction = Lfraction(load_fraction);
                        work_sfc_cur =
sseng_sfc(sep_ss_id)/gen_sfc_FL(sfc_ss_ind)*partial_sfc(sfc_ss_ind,sfc_frac
tion-4)*LHV_correction;
                        % Total working fuel consumption
                        work_fc(i,j) =
sum(usable_cross_work(2:3))*propeng_sfc(eng_id_sp)+load_fraction*usable_HP_
ss_work(1)*work_sfc_cur; % [gr/hr]
                        work_sfc(i,j) =
work_fc(i,j)/(sum(usable_cross_work(2:3))+load_fraction*usable_HP_ss_work(1
)); % [gr/KWh]
                    elseif prop_power(i)/n_trans_sp+ss_power(j)/n_ss_pd <
sum(usable_cross_work(2:3))+sum(usable_HP_ss_work) &&
prop_power(i)/n_trans_sp < sum(usable_cross_work(2:3))
                        % Both sprint and 1st ss engines at full load, 2nd
                        % ss engine at partial load
                        usable_HP_pd(i,j) = sum(usable_cross_work(2:3))-
prop_power(i)/n_trans_sp;
                        load_fraction = (ss_power(j)-
```

```
usable_HP_pd(i,j)*n_ss_pd-
usable_HP_ss_work(1)*n_gen)/n_gen/usable_HP_ss_work(2);
                        plot_fraction_sp1(i,j) = 1;
                        plot_fraction_sp2(i,j) = 1;
                        plot_fraction_ss1(i,j) = 1;
                        plot_fraction_ss2(i,j) = load_fraction;
                        sfc_fraction = Lfraction(load_fraction);
                        work_sfc_cur =
sseng_sfc(sep_ss_id)/gen_sfc_FL(sfc_ss_ind)*partial_sfc(sfc_ss_ind,sfc_frac
tion-4)*LHV_correction;
                        % Total working fuel consumption
                        work_fc(i,j) =
sum(usable_cross_work(2:3))*propeng_sfc(eng_id_sp)+usable_HP_ss_work(1)*sse
ng_sfc(sep_ss_id)+load_fraction*usable_HP_ss_work(2)*work_sfc_cur; %
[gr/hr]
                        work_sfc(i,j) =
work_fc(i,j)/(sum(usable_cross_work(2:3))+usable_HP_ss_work(1)+load_fractio
n*usable_HP_ss_work(2)); % [gr/KWh]
                    elseif prop_power(i)/n_trans_sp+ss_power(j)/n_ss_pd <
sum(usable_cross_work)
                        % Cruise and 1st sprint engine at full load, 2nd
                        % sprint engine at partial load
                        usable_HP_pd(i,j) =
usable_cross_work(1)*(n_trans_cr-
n_trans_sp)/n_trans_sp+sum(usable_cross_work)-prop_power(i)/n_trans_sp;
                        sfc_ind = SFCind(eng_id_sp,system);
                        load_fraction = ((prop_power(i)-
usable_cross_work(1)*n_trans_cr-
usable_cross_work(2)*n_trans_sp)/n_trans_sp+ss_power(j)/n_ss_pd)/usable_cro
ss_work(3);
                        sfc_fraction = Lfraction(load_fraction);
                        load_fraction2 = n_trans_sp/(1-(1-
n_trans_sp)*loss_mult(sfc_fraction-4))*load_fraction;
                        plot_fraction_cr(i,j) = 1;
                        plot_fraction_sp1(i,j) = 1;
                        plot_fraction_sp2(i,j) = load_fraction2;
                        work_sfc_cur =
propeng_sfc(eng_id_sp)/gen_sfc_FL(sfc_ind)*partial_sfc(sfc_ind,sfc_fraction
-4)*LHV_correction;
                        % Total working fuel consumption
                        work_fc(i,j) =
propeng_sfc(eng_id_cr)*usable_cross_work(1)+propeng_sfc(eng_id_sp)*usable_c
ross_work(2)+load_fraction2*usable_cross_work(3)*work_sfc_cur; % [gr/hr]
                        work_sfc(i,j) =
work_fc(i,j)/(sum(usable_cross_work(1:2))+load_fraction2*usable_cross_work(
3)); % [gr/KWh]
                    elseif prop_power(i)/n_trans_sp+ss_power(j)/n_ss_pd <
sum(usable_cross_work)+usable_HP_ss_work(1) && prop_power(i)/n_trans_sp <
sum(usable_cross_work)
                        % Cruise and both sprint engines at full load, 1st
ss engine at partial load
                        usable_HP_pd(i,j) =
usable_cross_work(1)*(n_trans_cr-
n_trans_sp)/n_trans_sp+sum(usable_cross_work(1:3))-
prop_power(i)/n_trans_sp;
                        load_fraction = (ss_power(j)-
usable_HP_pd(i,j)*n_ss_pd)/n_gen/usable_HP_ss_work(1);
                        plot_fraction_cr(i,j) = 1;
```

```
                        plot_fraction_sp1(i,j) = 1;
                        plot_fraction_sp2(i,j) = 1;
                        plot_fraction_ss1(i,j) = load_fraction;
                        sfc_fraction = Lfraction(load_fraction);
                        work_sfc_cur =
sseng_sfc(sep_ss_id)/gen_sfc_FL(sfc_ss_ind)*partial_sfc(sfc_ss_ind,sfc_frac
tion-4)*LHV_correction;
                        % Total working fuel consumption
                        work_fc(i,j) =
propeng_sfc(eng_id_cr)*usable_cross_work(1)+sum(usable_cross_work(2:3))*pro
peng_sfc(eng_id_sp)+load_fraction*usable_HP_ss_work(1)*work_sfc_cur; %
[gr/hr]
                        work_sfc(i,j) =
work_fc(i,j)/(sum(usable_cross_work(1:3))+load_fraction*usable_HP_ss_work(1
)); % [gr/KWh]
                    elseif prop_power(i)/n_trans_sp+ss_power(j)/n_ss_pd <
sum(usable_cross_work)+sum(usable_HP_ss_work) && prop_power(i)/n_trans_sp <
sum(usable_cross_work)
                        % Cruise, both sprint and 1st ss engines at full
                        % load, 2nd ss engine at partial load
                        usable_HP_pd(i,j) =
usable_cross_work(1)*(n_trans_cr-
n_trans_sp)/n_trans_sp+sum(usable_cross_work)-prop_power(i)/n_trans_sp;
                        load_fraction = (ss_power(j)-
usable_HP_pd(i,j)*n_ss_pd-
usable_HP_ss_work(1)*n_gen)/n_gen/usable_HP_ss_work(2);
                        plot_fraction_cr(i,j) = 1;
                        plot_fraction_sp1(i,j) = 1;
                        plot_fraction_sp2(i,j) = 1;
                        plot_fraction_ss1(i,j) = 1;
                        plot_fraction_ss2(i,j) = load_fraction;
                        sfc_fraction = Lfraction(load_fraction);
                        work_sfc_cur =
sseng_sfc(sep_ss_id)/gen_sfc_FL(sfc_ss_ind)*partial_sfc(sfc_ss_ind,sfc_frac
tion-4)*LHV_correction;
                        % Total working fuel consumption
                        work_fc(i,j) =
propeng_sfc(eng_id_cr)*usable_cross_work(1)+sum(usable_cross_work(2:3))*pro
peng_sfc(eng_id_sp)+usable_HP_ss_work(1)*sseng_sfc(sep_ss_id)+load_fraction
*usable_HP_ss_work(2)*work_sfc_cur; % [gr/hr]
                        work_sfc(i,j) =
work_fc(i,j)/(sum(usable_cross_work(1:3))+usable_HP_ss_work(1)+load_fractio
n*usable_HP_ss_work(2)); % [gr/KWh]
                    end
                else % 2 cruise engines and 1 sprint engine
                    if prop_power(i)/n_trans_cr+ss_power(j)/n_ss_pd <
usable_cross_work(1)
                        % 1st cruise engine at partial load
                        usable_HP_pd(i,j) = usable_cross_work(1)-
prop_power(i)/n_trans_cr;
                        sfc_ind = SFCind(eng_id_cr,system);
                        load_fraction =
(prop_power(i)/n_trans_cr+ss_power(j)/n_ss_pd)/usable_cross_work(1);
                        sfc_fraction = Lfraction(load_fraction);
                        load_fraction2 = n_trans_cr/(1-(1-
n_trans_cr)*loss_mult(sfc_fraction-4))*load_fraction;
                        plot_fraction_cr1(i,j) = load_fraction2;
                        work_sfc_cur =
```

```
propeng_sfc(eng_id_cr)/gen_sfc_FL(sfc_ind)*partial_sfc(sfc_ind,sfc_fraction
-4)*LHV_correction;
                        % Working fuel consumption
                        work_fc(i,j) =
load_fraction2*usable_cross_work(1)*work_sfc_cur; % [gr/hr]
                        work_sfc(i,j) = work_sfc_cur;
                    elseif prop_power(i)/n_trans_cr+ss_power(j)/n_ss_pd <
usable_cross_work(1)+usable_HP_ss_work(1) && prop_power(i)/n_trans_cr <
usable_cross_work(1)
                        % 1st cruise engine at full load and 1st ss engine
at
                        % partial load
                        usable_HP_pd(i,j) = usable_cross_work(1)-
prop_power(i)/n_trans_cr;
                        load_fraction = (ss_power(j)-
usable_HP_pd(i,j)*n_ss_pd)/n_gen/usable_HP_ss_work(1);
                        plot_fraction_cr1(i,j) = 1;
                        plot_fraction_ss1(i,j) = load_fraction;
                        sfc_fraction = Lfraction(load_fraction);
                        work_sfc_cur =
sseng_sfc(sep_ss_id)/gen_sfc_FL(sfc_ss_ind)*partial_sfc(sfc_ss_ind,sfc_frac
tion-4)*LHV_correction;
                        % Total working fuel consumption
                        work_fc(i,j) =
usable_cross_work(1)*propeng_sfc(eng_id_cr)+load_fraction*usable_HP_ss_work
(1)*work_sfc_cur; % [gr/hr]
                        work_sfc(i,j) =
work_fc(i,j)/(usable_cross_work(1)+load_fraction*usable_HP_ss_work(1)); %
[gr/KWh]
                    elseif prop_power(i)/n_trans_cr+ss_power(j)/n_ss_pd <
usable_cross_work(1)+sum(usable_HP_ss_work) && prop_power(i)/n_trans_cr <
usable_cross_work(1)
                        % 1st cruise and 1st ss engine at full load, 2nd ss
engine at partial load
                        usable_HP_pd(i,j) = usable_cross_work(1)-
prop_power(i)/n_trans_cr;
                        load_fraction = (ss_power(j)-
usable_HP_pd(i,j)*n_ss_pd-
usable_HP_ss_work(1)*n_gen)/n_gen/usable_HP_ss_work(2);
                        plot_fraction_cr1(i,j) = 1;
                        plot_fraction_ss1(i,j) = 1;
                        plot_fraction_ss2(i,j) = load_fraction;
                        sfc_fraction = Lfraction(load_fraction);
                        work_sfc_cur =
sseng_sfc(sep_ss_id)/gen_sfc_FL(sfc_ss_ind)*partial_sfc(sfc_ss_ind,sfc_frac
tion-4)*LHV_correction;
                        % Total working fuel consumption
                        work_fc(i,j) =
usable_cross_work(1)*propeng_sfc(eng_id_cr)+usable_HP_ss_work(1)*sseng_sfc(
sep_ss_id)+load_fraction*usable_HP_ss_work(2)*work_sfc_cur; % [gr/hr]
                        work_sfc(i,j) =
work_fc(i,j)/(usable_cross_work(1)+usable_HP_ss_work(1)+load_fraction*usabl
e_HP_ss_work(2)); % [gr/KWh]
                    elseif prop_power(i)/n_trans_cr+ss_power(j)/n_ss_pd <
sum(usable_cross_work(1:2))
                        % 1st cruise engine at full load, 2nd cruise
                        % engine at partial load
                        usable_HP_pd(i,j) = sum(usable_cross_work(1:2))-
```

```
prop_power(i)/n_trans_cr;
                        sfc_ind = SFCind(eng_id_cr,system);
                        load_fraction = ((prop_power(i)-
usable_cross_work(1)*n_trans_cr)/n_trans_cr+ss_power(j)/n_ss_pd)/usable_cro
ss_work(2);
                        sfc_fraction = Lfraction(load_fraction);
                        load_fraction2 = n_trans_cr/(1-(1-
n_trans_cr)*loss_mult(sfc_fraction-4))*load_fraction;
                        plot_fraction_cr1(i,j) = 1;
                        plot_fraction_cr2(i,j) = load_fraction2;
                        work_sfc_cur =
propeng_sfc(eng_id_cr)/gen_sfc_FL(sfc_ind)*partial_sfc(sfc_ind,sfc_fraction
-4)*LHV_correction;
                        % Working fuel consumption
                        work_fc(i,j) =
usable_cross_work(1)*propeng_sfc(eng_id_cr)+load_fraction2*usable_cross_wor
k(2)*work_sfc_cur; % [gr/hr]
                        work_sfc(i,j) =
work_fc(i,j)/(usable_cross_work(1)+load_fraction2*usable_cross_work(2)); %
[gr/KWh]
                    elseif prop_power(i)/n_trans_cr+ss_power(j)/n_ss_pd <
sum(usable_cross_work(1:2))+usable_HP_ss_work(1) &&
prop_power(i)/n_trans_cr < sum(usable_cross_work(1:2))
                        % Both cruise engines at full load, 1st ss engine
                        % at partial load
                        usable_HP_pd(i,j) = sum(usable_cross_work(1:2))-
prop_power(i)/n_trans_cr;
                        load_fraction = (ss_power(j)-
usable_HP_pd(i,j)*n_ss_pd)/n_gen/usable_HP_ss_work(1);
                        plot_fraction_cr1(i,j) = 1;
                        plot_fraction_cr2(i,j) = 1;
                        plot_fraction_ss1(i,j) = load_fraction;
                        sfc_fraction = Lfraction(load_fraction);
                        work_sfc_cur =
sseng_sfc(sep_ss_id)/gen_sfc_FL(sfc_ss_ind)*partial_sfc(sfc_ss_ind,sfc_frac
tion-4)*LHV_correction;
                        % Total working fuel consumption
                        work_fc(i,j) =
sum(usable_cross_work(1:2))*propeng_sfc(eng_id_cr)+load_fraction*usable_HP_
ss_work(1)*work_sfc_cur; % [gr/hr]
                        work_sfc(i,j) =
work_fc(i,j)/(sum(usable_cross_work(1:2))+load_fraction*usable_HP_ss_work(1
)); % [gr/KWh]
                    elseif prop_power(i)/n_trans_cr+ss_power(j)/n_ss_pd <
sum(usable_cross_work(1:2))+sum(usable_HP_ss_work) &&
prop_power(i)/n_trans_cr < sum(usable_cross_work(1:2))
                        % Both cruise and 1st ss engines at full load, 2nd
                        % ss engine at partial load
                        usable_HP_pd(i,j) = sum(usable_cross_work(1:2))-
prop_power(i)/n_trans_cr;
                        load_fraction = (ss_power(j)-
usable_HP_pd(i,j)*n_ss_pd-
usable_HP_ss_work(1)*n_gen)/n_gen/usable_HP_ss_work(2);
                        plot_fraction_cr1(i,j) = 1;
                        plot_fraction_cr2(i,j) = 1;
                        plot_fraction_ss1(i,j) = 1;
                        plot_fraction_ss2(i,j) = load_fraction;
                        sfc_fraction = Lfraction(load_fraction);
```

```
                        work_sfc_cur =
sseng_sfc(sep_ss_id)/gen_sfc_FL(sfc_ss_ind)*partial_sfc(sfc_ss_ind,sfc_frac
tion-4)*LHV_correction;
                        % Total working fuel consumption
                        work_fc(i,j) =
sum(usable_cross_work(1:2))*propeng_sfc(eng_id_cr)+usable_HP_ss_work(1)*sse
ng_sfc(sep_ss_id)+load_fraction*usable_HP_ss_work(2)*work_sfc_cur; %
[gr/hr]
                        work_sfc(i,j) =
work_fc(i,j)/(sum(usable_cross_work(1:2))+usable_HP_ss_work(1)+load_fractio
n*usable_HP_ss_work(2)); % [gr/KWh]
                    elseif prop_power(i)/n_trans_sp+ss_power(j)/n_ss_pd <
usable_cross_work(3)
                        % Sprint engine at partial load
                        usable_HP_pd(i,j) = usable_cross_work(3)-
prop_power(i)/n_trans_sp;
                        sfc_ind = SFCind(eng_id_sp,system);
                        load_fraction =
(prop_power(i)/n_trans_sp+ss_power(j)/n_ss_pd)/usable_cross_work(3);
                        sfc_fraction = Lfraction(load_fraction);
                        load_fraction2 = n_trans_sp/(1-(1-
n_trans_sp)*loss_mult(sfc_fraction-4))*load_fraction;
                        plot_fraction_sp(i,j) = load_fraction2;
                        work_sfc_cur =
propeng_sfc(eng_id_sp)/gen_sfc_FL(sfc_ind)*partial_sfc(sfc_ind,sfc_fraction
-4)*LHV_correction;
                        % Working fuel consumption
                        work_fc(i,j) =
load_fraction2*usable_cross_work(3)*work_sfc_cur; % [gr/hr]
                        work_sfc(i,j) = work_sfc_cur;
                    elseif prop_power(i)/n_trans_sp+ss_power(j)/n_ss_pd <
usable_cross_work(3)+usable_HP_ss_work(1) && prop_power(i)/n_trans_sp <
usable_cross_work(3)
                        % Sprint engine at full load, 1st ss engine at
                        % partial load
                        usable_HP_pd(i,j) = usable_cross_work(3)-
prop_power(i)/n_trans_sp;
                        load_fraction = (ss_power(j)-
usable_HP_pd(i,j)*n_ss_pd)/n_gen/usable_HP_ss_work(1);
                        plot_fraction_sp(i,j) = 1;
                        plot_fraction_ss1(i,j) = load_fraction;
                        sfc_fraction = Lfraction(load_fraction);
                        work_sfc_cur =
sseng_sfc(sep_ss_id)/gen_sfc_FL(sfc_ss_ind)*partial_sfc(sfc_ss_ind,sfc_frac
tion-4)*LHV_correction;
                        % Total working fuel consumption
                        work_fc(i,j) =
usable_cross_work(3)*propeng_sfc(eng_id_sp)+load_fraction*usable_HP_ss_work
(1)*work_sfc_cur; % [gr/hr]
                        work_sfc(i,j) =
work_fc(i,j)/(usable_cross_work(3)+load_fraction*usable_HP_ss_work(1)); %
[gr/KWh]
                    elseif prop_power(i)/n_trans_sp+ss_power(j)/n_ss_pd <
usable_cross_work(3)+sum(usable_HP_ss_work) && prop_power(i)/n_trans_sp <
usable_cross_work(3)
                        % Sprint and 1st ss engines at full load, 2nd ss
                        % engine at partial load
                        usable_HP_pd(i,j) = usable_cross_work(3)-
```

```
prop_power(i)/n_trans_sp;
                        load_fraction = (ss_power(j)-
usable_HP_pd(i,j)*n_ss_pd-
usable_HP_ss_work(1)*n_gen)/n_gen/usable_HP_ss_work(2);
                        plot_fraction_sp(i,j) = 1;
                        plot_fraction_ss1(i,j) = 1;
                        plot_fraction_ss2(i,j) = load_fraction;
                        sfc_fraction = Lfraction(load_fraction);
                        work_sfc_cur =
sseng_sfc(sep_ss_id)/gen_sfc_FL(sfc_ss_ind)*partial_sfc(sfc_ss_ind,sfc_frac
tion-4)*LHV_correction;
                        % Total working fuel consumption
                        work_fc(i,j) =
usable_cross_work(3)*propeng_sfc(eng_id_sp)+usable_HP_ss_work(1)*sseng_sfc(
sep_ss_id)+load_fraction*usable_HP_ss_work(2)*work_sfc_cur; % [gr/hr]
                        work_sfc(i,j) =
work_fc(i,j)/(usable_cross_work(3)+usable_HP_ss_work(1)+load_fraction*usabl
e_HP_ss_work(2)); % [gr/KWh]
                    elseif prop_power(i)/n_trans_sp+ss_power(j)/n_ss_pd <
sum(usable_cross_work(2:3))
                        % 1st cruise engine at full load, sprint engine at
                        % partial load
                        usable_HP_pd(i,j) =
usable_cross_work(2)*(n_trans_cr-
n_trans_sp)/n_trans_sp+sum(usable_cross_work(2:3))-
prop_power(i)/n_trans_sp;
                        sfc_ind = SFCind(eng_id_sp,system);
                        load_fraction = ((prop_power(i)-
usable_cross_work(2)*n_trans_cr)/n_trans_sp+ss_power(j)/n_ss_pd)/usable_cro
ss_work(3);
                        sfc_fraction = Lfraction(load_fraction);
                        load_fraction2 = n_trans_sp/(1-(1-
n_trans_sp)*loss_mult(sfc_fraction-4))*load_fraction;
                        plot_fraction_cr(i,j) = 1;
                        plot_fraction_sp1(i,j) = load_fraction2;
                        work_sfc_cur =
propeng_sfc(eng_id_sp)/gen_sfc_FL(sfc_ind)*partial_sfc(sfc_ind,sfc_fraction
-4)*LHV_correction;
                        % Total working fuel consumption
                        work_fc(i,j) =
propeng_sfc(eng_id_cr)*usable_cross_work(2)+load_fraction2*usable_cross_wor
k(3)*work_sfc_cur; % [gr/hr]
                        work_sfc(i,j) =
work_fc(i,j)/(usable_cross_work(2)+load_fraction2*usable_cross_work(3)); %
[gr/KWh]
                    elseif prop_power(i)/n_trans_sp+ss_power(j)/n_ss_pd <
sum(usable_cross_work(2:3))+usable_HP_ss_work(1) &&
prop_power(i)/n_trans_sp < sum(usable_cross_work(2:3))
                        % 1st cruise and sprint engines at full load, 1st
                        % ss engine at partial load
                        usable_HP_pd(i,j) =
usable_cross_work(2)*(n_trans_cr-
n_trans_sp)/n_trans_sp+sum(usable_cross_work(2:3))-
prop_power(i)/n_trans_sp;
                        load_fraction = (ss_power(j)-
usable_HP_pd(i,j)*n_ss_pd)/n_gen/usable_HP_ss_work(1);
                        plot_fraction_cr(i,j) = 1;
                        plot_fraction_sp1(i,j) = 1;
```

```
                        plot_fraction_ss1(i,j) = load_fraction;
                        sfc_fraction = Lfraction(load_fraction);
                        work_sfc_cur =
sseng_sfc(sep_ss_id)/gen_sfc_FL(sfc_ss_ind)*partial_sfc(sfc_ss_ind,sfc_frac
tion-4)*LHV_correction;
                        % Total working fuel consumption
                        work_fc(i,j) =
propeng_sfc(eng_id_cr)*usable_cross_work(2)+usable_cross_work(3)*propeng_sf
c(eng_id_sp)+load_fraction*usable_HP_ss_work(1)*work_sfc_cur; % [gr/hr]
                        work_sfc(i,j) =
work_fc(i,j)/(sum(usable_cross_work(2:3))+load_fraction*usable_HP_ss_work(1
)); % [gr/KWh]
                    elseif prop_power(i)/n_trans_sp+ss_power(j)/n_ss_pd <
sum(usable_cross_work(2:3))+sum(usable_HP_ss_work) &&
prop_power(i)/n_trans_sp < sum(usable_cross_work(2:3))
                        % 1st cruise, sprint and 1st ss engines at full
                        % load, 2nd ss engine at partial load
                        usable_HP_pd(i,j) =
usable_cross_work(2)*(n_trans_cr-
n_trans_sp)/n_trans_sp+sum(usable_cross_work(2:3))-
prop_power(i)/n_trans_sp;
                        load_fraction = (ss_power(j)-
usable_HP_pd(i,j)*n_ss_pd-
usable_HP_ss_work(1)*n_gen)/n_gen/usable_HP_ss_work(2);
                        plot_fraction_cr(i,j) = 1;
                        plot_fraction_sp1(i,j) = 1;
                        plot_fraction_ss1(i,j) = 1;
                        plot_fraction_ss2(i,j) = load_fraction;
                        sfc_fraction = Lfraction(load_fraction);
                        work_sfc_cur =
sseng_sfc(sep_ss_id)/gen_sfc_FL(sfc_ss_ind)*partial_sfc(sfc_ss_ind,sfc_frac
tion-4)*LHV_correction;
                        % Total working fuel consumption
                        work_fc(i,j) =
propeng_sfc(eng_id_cr)*usable_cross_work(2)+usable_cross_work(3)*propeng_sf
c(eng_id_sp)+usable_HP_ss_work(1)*sseng_sfc(sep_ss_id)+load_fraction*usable
_HP_ss_work(2)*work_sfc_cur; % [gr/hr]
                        work_sfc(i,j) =
work_fc(i,j)/(sum(usable_cross_work(2:3))+usable_HP_ss_work(1)+load_fractio
n*usable_HP_ss_work(2)); % [gr/KWh]
                    elseif prop_power(i)/n_trans_sp+ss_power(j)/n_ss_pd <
sum(usable_cross_work(1:3))
                        % Both cruise engines at full load, sprint engine
                        % at partial load
                        usable_HP_pd(i,j) =
sum(usable_cross_work(1:2))*(n_trans_cr-
n_trans_sp)/n_trans_sp+sum(usable_cross_work)-prop_power(i)/n_trans_sp;
                        sfc_ind = SFCind(eng_id_sp,system);
                        load_fraction = ((prop_power(i)-
sum(usable_cross_work(1:2))*n_trans_cr)/n_trans_sp+ss_power(j)/n_ss_pd)/usa
ble_cross_work(3);
                        sfc_fraction = Lfraction(load_fraction);
                        load_fraction2 = n_trans_sp/(1-(1-
n_trans_sp)*loss_mult(sfc_fraction-4))*load_fraction;
                        plot_fraction_cr1(i,j) = 1;
                        plot_fraction_cr2(i,j) = 1;
                        plot_fraction_sp(i,j) = load_fraction2;
                        work_sfc_cur =
```

```
propeng_sfc(eng_id_sp)/gen_sfc_FL(sfc_ind)*partial_sfc(sfc_ind,sfc_fraction
-4)*LHV_correction;
                        % Total working fuel consumption
                        work_fc(i,j) =
propeng_sfc(eng_id_cr)*sum(usable_cross_work(1:2))+load_fraction2*usable_cr
oss_work(3)*work_sfc_cur; % [gr/hr]
                        work_sfc(i,j) =
work_fc(i,j)/(sum(usable_cross_work(1:2))+load_fraction2*usable_cross_work(
3)); % [gr/KWh]
                    elseif prop_power(i)/n_trans_sp+ss_power(j)/n_ss_pd <
sum(usable_cross_work)+usable_HP_ss_work(1) && prop_power(i)/n_trans_sp <
sum(usable_cross_work)
                        % Both cruise and sprint engines at full load, 1st
                        % ss engine at partial load
                        usable_HP_pd(i,j) =
sum(usable_cross_work(1:2))*(n_trans_cr-
n_trans_sp)/n_trans_sp+sum(usable_cross_work)-prop_power(i)/n_trans_sp;
                        load_fraction = (ss_power(j)-
usable_HP_pd(i,j)*n_ss_pd)/n_gen/usable_HP_ss_work(1);
                        plot_fraction_cr1(i,j) = 1;
                        plot_fraction_cr2(i,j) = 1;
                        plot_fraction_sp(i,j) = 1;
                        plot_fraction_ss1(i,j) = load_fraction;
                        sfc_fraction = Lfraction(load_fraction);
                        work_sfc_cur =
sseng_sfc(sep_ss_id)/gen_sfc_FL(sfc_ss_ind)*partial_sfc(sfc_ss_ind,sfc_frac
tion-4)*LHV_correction;
                        % Total working fuel consumption
                        work_fc(i,j) =
propeng_sfc(eng_id_cr)*sum(usable_cross_work(1:2))+usable_cross_work(3)*pro
peng_sfc(eng_id_sp)+load_fraction*usable_HP_ss_work(1)*work_sfc_cur; %
[gr/hr]
                        work_sfc(i,j) =
work_fc(i,j)/(sum(usable_cross_work(1:3))+load_fraction*usable_HP_ss_work(1
)); % [gr/KWh]
                    elseif prop_power(i)/n_trans_sp+ss_power(j)/n_ss_pd <
sum(usable_cross_work)+sum(usable_HP_ss_work) && prop_power(i)/n_trans_sp <
sum(usable_cross_work)
                        % Both cruise, sprint and 1st ss engines at full
                        % load, 2nd ss engine at partial load
                        usable_HP_pd(i,j) =
sum(usable_cross_work(1:2))*(n_trans_cr-
n_trans_sp)/n_trans_sp+sum(usable_cross_work)-prop_power(i)/n_trans_sp;
                        load_fraction = (ss_power(j)-
usable_HP_pd(i,j)*n_ss_pd-
usable_HP_ss_work(1)*n_gen)/n_gen/usable_HP_ss_work(2);
                        plot_fraction_cr1(i,j) = 1;
                        plot_fraction_cr2(i,j) = 1;
                        plot_fraction_sp(i,j) = 1;
                        plot_fraction_ss1(i,j) = 1;
                        plot_fraction_ss2(i,j) = load_fraction;
                        sfc_fraction = Lfraction(load_fraction);
                        work_sfc_cur =
sseng_sfc(sep_ss_id)/gen_sfc_FL(sfc_ss_ind)*partial_sfc(sfc_ss_ind,sfc_frac
tion-4)*LHV_correction;
                        % Total working fuel consumption
                        work_fc(i,j) =
propeng_sfc(eng_id_cr)*sum(usable_cross_work(1:2))+usable_cross_work(3)*pro
```

```
peng_sfc(eng_id_sp)+usable_HP_ss_work(1)*sseng_sfc(sep_ss_id)+load_fraction
*usable_HP_ss_work(2)*work_sfc_cur; % [gr/hr]
                        work_sfc(i,j) =
work_fc(i,j)/(sum(usable_cross_work(1:3))+usable_HP_ss_work(1)+load_fractio
n*usable_HP_ss_work(2)); % [gr/KWh]
                    end
                end
            end
        end
    end

% Load fraction can happen to be slightly above 1 or below 0 due to
differences in propulsion and generator efficiencies
% Load fraction for 2nd ss engine plot fix
for i = 1:i_speed
    for j = 1:i_sspower
        if plot_fraction_ss2(i,j)>1 && plot_fraction_ss2(i,j)<1.1
            plot_fraction_ss2(i,j)=1;
        end
        if plot_fraction_ss2(i,j)<0
            plot_fraction_ss2(i,j)=0;
        end
    end
end


%% Total fuel consumption for the given operating profile
fc_total = sum(sum(profile_data(2:end-1,2:end-2).*work_fc/1000))/100; %
Total fuel consumption [kg/hr]


%% Display results
% Cell to string
name_cr = char(propeng_name(eng_id_cr));
type_cr = char(propeng_type(eng_id_cr));
manuf_cr = char(propeng_manuf(eng_id_cr));
name_sp = char(propeng_name(eng_id_sp));
type_sp = char(propeng_type(eng_id_sp));
manuf_sp = char(propeng_manuf(eng_id_sp));
name_ss = char(sseng_name(sep_ss_id));
type_ss = char(sseng_type(sep_ss_id));
manuf_ss = char(sseng_manuf(sep_ss_id));

% Plot propulsion engines load fractions and fuel consumption
% Display screen results
[X1,Y1] = meshgrid(profile_data(1,2:end-2),profile_data(2:end-1,1));
figure
meshz(X1,Y1,work_fc/1000),xlabel('Ship service power [kW]'),ylabel('Speed
[knots]'),zlabel('Fuel Consumption [kg/h]'),title('PDSS Fuel Consumption
Profile')

if strcmp(sys_type,'Cross-connected') == 0
    figure
    meshz(X1,Y1,plot_fraction_cr),xlabel('Ship service power
[kW]'),ylabel('speed [knots]'),zlabel('Load fraction'),title('Cruise
Engines Load Fraction Profile')
    figure
```

```
    meshz(X1,Y1,plot_fraction_sp),xlabel('Ship service power
[kW]'),ylabel('speed [knots]'),zlabel('Load fraction'),title('Sprint
Engines Load Fraction Profile')
    else
    if eng_num_cr == 1
        figure
        meshz(X1,Y1,plot_fraction_cr),xlabel('Ship service power
[kW]'),ylabel('speed [knots]'),zlabel('Load
fraction'),title(strcat(name_cr,'load fraction profile'))
        figure
        meshz(X1,Y1,plot_fraction_sp1),xlabel('Ship service power
[kW]'),ylabel('speed [knots]'),zlabel('Load fraction'),title(strcat('1st-
',name_sp,'load fraction profile'))
        figure
        meshz(X1,Y1,plot_fraction_sp2),xlabel('Ship service power
[kW]'),ylabel('speed [knots]'),zlabel('Load fraction'),title(strcat('2nd-
',name_sp,'load fraction profile'))
    else
        figure
        meshz(X1,Y1,plot_fraction_cr1),xlabel('Ship service power
[kW]'),ylabel('speed [knots]'),zlabel('Load fraction'),title(strcat('1st-
',name_cr,'load fraction profile'))
        figure
        meshz(X1,Y1,plot_fraction_cr2),xlabel('Ship service power
[kW]'),ylabel('speed [knots]'),zlabel('Load fraction'),title(strcat('2nd-
',name_cr,'load fraction profile'))
        figure
        meshz(X1,Y1,plot_fraction_sp1),xlabel('Ship service power
[kW]'),ylabel('speed [knots]'),zlabel('Load
fraction'),title(strcat(name_sp,'load fraction profile'))
    end
end

disp('-----------------------------------------------------------------------
--')
disp('MECHANICAL-DRIVEN PROPULSION WITH PROPULSION DERIVED SHIP-SERVICE
POWER')
fprintf('\t\t     OPTIMAL CONFIGURATION\n\n');
fprintf('Propulsion system: %s\n\n', sys_type);
fprintf('Cruise engines:       %d %s %s (%s) / %dkW\n', eng_num_cr,
name_cr, type_cr, manuf_cr,propeng_power(eng_id_cr));
fprintf('Sprint engines:       %d %s %s (%s) / %dkW\n', eng_num_sp,
name_sp, type_sp, manuf_sp,propeng_power(eng_id_sp));

if sep_ss_dec == 1
    fprintf('Ship service engines: %d %s %s (%s) / %dkW\n\n', sep_ss_num,
name_ss, type_ss, manuf_ss,sseng_power(sep_ss_id));
    if sep_ss_num == 2
        figure
        meshz(X1,Y1,plot_fraction_ss1),xlabel('Ship service power
[kW]'),ylabel('speed [knots]'),zlabel('Load fraction'),title('Separate SS
Engine Load Fraction Profile')
    else
        figure
        meshz(X1,Y1,plot_fraction_ss1),xlabel('Ship service power
[kW]'),ylabel('speed [knots]'),zlabel('Load fraction'),title('1st Separate
SS Engine Load Fraction Profile')
        figure
```

```
        meshz(X1,Y1,plot_fraction_ss2),xlabel('Ship service power
[kW]'),ylabel('speed [knots]'),zlabel('Load fraction'),title('2nd Separate
SS Engine Load Fraction Profile')
    end
else
    disp('No separate ship-service engines exist')
    fprintf('\n')
end
fprintf('Average fuel consumption for given combined profile: %5.0f
kg/hr\n', fc_total);
disp('-------------------------------------------------------------------
--')

end
```

Function *IPSevalf.m*:

```
function
[plot_fraction_pr1,plot_fraction_pr2,plot_fraction_sec1,plot_fraction_sec2]
= IPSevalf(x)
%% Function evaluates integrated power systems (IPS)

global profile_data dspeed dss SHP gen_sfc_FL partial_sfc LHV_correction
M_s...
    ipseng_type ipseng_sfc ipseng_power ipseng_name ipseng_manuf

%% PGM selection
ips_id_pr = 1+Convert( x, 1, 4 ); % Primary PGM selection ID
ips_id_sec = 1+Convert( x, 5, 4 ); % Secondary PGM selection ID
ips_num_pr = 2+Convert( x, 9, 1 ); % Number of primary PGMs (2 or 3)

ips_sec_dec = 3-ips_num_pr; % Secondary PGMs exist if primary PGMs number
is 2
ips_num_sec = 2*(3-ips_num_pr); % Number of secondary PGMs is 2 when 2
primary PGMs are present

if ips_num_pr==2 && ipseng_power(ips_id_sec)>ipseng_power(ips_id_pr)
    ips_id_trans = ips_id_pr;
    ips_id_pr = ips_id_sec;
    ips_id_sec = ips_id_trans;
end

%% IPS efficiencies
% Efficiency correction for electric drive generators and motors from SNAME
elec_cor_matrix = [26 95.4 ; 37 97 ; 42 97.5 ; 48 98 ; 55.5 98.5 ; 65 99 ;
78 99.5 ; 85 99.7 ; 100 100 ];
pp_elec = spline(elec_cor_matrix(:,1),elec_cor_matrix(:,2));
elec_cor_mult = ppval(pp_elec,25:100);

% % Use to plot datapoints and check piecewise polynomial fits the
datapoints
% figure;
%
plot(elec_cor_matrix(:,1),elec_cor_matrix(:,2),'o',25:100,elec_cor_mult),xl
abel('Load fraction [%]'),ylabel('Correction factor'),title('Efficiency
correction for electric drive generators and motors at partial load
operation')


% Generator, motor drive and propulsion motor efficiencies
n_gen = 0.96;
n_mdrive = 0.96;
n_pmotor = 0.96;
n_ips_prop = n_gen*n_mdrive*n_pmotor;

% Ship service power generation efficiency
n_ips_pcm = 0.965; % PCM efficiency
n_ips_ss = n_gen*n_ips_pcm;

%% Usable power calculation
```

```
usable_HP_ips_work = zeros(ips_num_pr+ips_num_sec,1);
if ips_sec_dec == 1
    for i = 1:ips_num_sec;
    usable_HP_ips_work(i) = ipseng_power(ips_id_sec);
    end
    for i = ips_num_sec+1:ips_num_sec+ips_num_pr
        usable_HP_ips_work(i) = ipseng_power(ips_id_pr);
    end
elseif ips_sec_dec == 0
    for i = 1:ips_num_pr
        usable_HP_ips_work(i) = ipseng_power(ips_id_pr);
    end
end

clear i



%% IPS fuel consumption calculations
% load_fraction2: Engine load increases due to partial load motor drive and
propulsion motor operation
% Check that IPS provides the required HP
i_speed = length(profile_data(2:end-1,1));
i_sspower = length(profile_data(1,2:end-2));
if dss/n_ips_ss+SHP(dspeed+1)/n_ips_prop > sum(usable_HP_ips_work)*(1-M_s)
    disp('Warning: Selected IPS configuration does not meet power
requirements')
else
    system = 'ips';
    % Pre-allocate
    prop_power = zeros(i_speed,1);
    ss_power = profile_data(1,2:end-2);
    work_fc = zeros(i_speed,i_sspower); % lin=prop, col=ss
    motor_load_fraction = zeros(i_speed,1);
    motor_fraction = zeros(i_speed,1);
    plot_fraction_pr1 = zeros(i_speed,i_sspower); % Used to plot load
fractions
    plot_fraction_pr2 = zeros(i_speed,i_sspower);
    plot_fraction_pr3 = zeros(i_speed,i_sspower);
    plot_fraction_sec1 = zeros(i_speed,i_sspower);
    plot_fraction_sec2 = zeros(i_speed,i_sspower);
    if ips_num_pr == 2
        % 2 primary and 2 secondary PGMs present
        for i = 1:i_speed
            prop_power(i) = SHP(i+1);
            motor_load_fraction(i) = prop_power(i)/SHP(dspeed+1); % motor
drive and propulsion motor load fraction
            motor_fraction(i) = Mfraction(motor_load_fraction(i));
            for j = 1:i_sspower
                if ss_power(j)/n_ips_ss+prop_power(i)/n_ips_prop <
sum(usable_HP_ips_work(1:2))
                    % Both secondary engines at partial load
                    load_fraction =
(ss_power(j)/n_ips_ss+prop_power(i)/n_ips_prop)/sum(usable_HP_ips_work(1:2)
);
                    sfc_fraction = Lfraction(load_fraction);
```

```
                    sfc_ind = SFCind(ips_id_sec,system);
                    load_fraction2 =
100*load_fraction/elec_cor_mult(motor_fraction(i)-24);
                    plot_fraction_sec1(i,j) = load_fraction2;
                    plot_fraction_sec2(i,j) = load_fraction2;
                    work_sfc_ips =
ipseng_sfc(ips_id_sec)/gen_sfc_FL(sfc_ind)*partial_sfc(sfc_ind,sfc_fraction
-4)*LHV_correction;
                    work_fc(i,j) =
load_fraction2*sum(usable_HP_ips_work(1:2))*work_sfc_ips; % [gr/hr]
                elseif ss_power(j)/n_ips_ss+prop_power(i)/n_ips_prop <
usable_HP_ips_work(1)+usable_HP_ips_work(3)
                    % 1st secondary engine at full load, 1st primary engine
                    % at partial load
                    load_fraction =
(ss_power(j)/n_ips_ss+prop_power(i)/n_ips_prop-
usable_HP_ips_work(1))/usable_HP_ips_work(3); % Assumes secondary PGM
working at 100%
                    sfc_fraction = Lfraction(load_fraction);
                    sfc_ind = SFCind(ips_id_pr,system);
                    load_fraction2 =
100*load_fraction/elec_cor_mult(motor_fraction(i)-24); % Engine load
increases due to partial load motor drive and propulsion motor operation
                    plot_fraction_sec1(i,j) = 1;
                    plot_fraction_pr1(i,j) = load_fraction2;
                    work_sfc =
ipseng_sfc(ips_id_pr)/gen_sfc_FL(sfc_ind)*partial_sfc(sfc_ind,sfc_fraction-
4)*LHV_correction;
                    % Total working fuel consumption
                    work_fc(i,j) =
ipseng_sfc(ips_id_sec)*usable_HP_ips_work(1)+load_fraction2*usable_HP_ips_w
ork(3)*work_sfc; % [gr/hr]
                elseif ss_power(j)/n_ips_ss+prop_power(i)/n_ips_prop <
sum(usable_HP_ips_work(1:3)) && sum(usable_HP_ips_work(1:3)) <
sum(usable_HP_ips_work(3:4))
                    % Both secondary engines at full load, 1st primary
                    % engine at partial load
                    load_fraction =
(ss_power(j)/n_ips_ss+prop_power(i)/n_ips_prop-
sum(usable_HP_ips_work(1:2)))/usable_HP_ips_work(3); % Secondary PGM
working at 100%
                    sfc_fraction = Lfraction(load_fraction);
                    sfc_ind = SFCind(ips_id_pr,system);
                    load_fraction2 =
100*load_fraction/elec_cor_mult(motor_fraction(i)-24); % Engine load
increases due to partial load motor drive and propulsion motor operation
                    plot_fraction_sec1(i,j) = 1;
                    plot_fraction_sec2(i,j) = 1;
                    plot_fraction_pr1(i,j) = load_fraction2;
                    work_sfc =
ipseng_sfc(ips_id_pr)/gen_sfc_FL(sfc_ind)*partial_sfc(sfc_ind,sfc_fraction-
4)*LHV_correction;
                    % Total working fuel consumption
                    work_fc(i,j) =
ipseng_sfc(ips_id_sec)*sum(usable_HP_ips_work(1:2))+load_fraction2*usable_H
P_ips_work(3)*work_sfc; % [gr/hr]
                elseif ss_power(j)/n_ips_ss+prop_power(i)/n_ips_prop <
sum(usable_HP_ips_work(3:4)) % Both primary and PGMs on line, 2nd at
```

```
partial load
                    % 1st primary engine at full load, 2nd primary engine
                    % at partial load
                    load_fraction =
(ss_power(j)/n_ips_ss+prop_power(i)/n_ips_prop-
usable_HP_ips_work(3))/usable_HP_ips_work(4); % 1 primary PGM working at
100%
                    sfc_fraction = Lfraction(load_fraction);
                    sfc_ind = SFCind(ips_id_pr,system);
                    load_fraction2 =
100*load_fraction/elec_cor_mult(motor_fraction(i)-24); % Engine load
increases due to partial load motor drive and propulsion motor operation
                    plot_fraction_pr1(i,j) = 1;
                    plot_fraction_pr2(i,j) = load_fraction2;
                    work_sfc =
ipseng_sfc(ips_id_pr)/gen_sfc_FL(sfc_ind)*partial_sfc(sfc_ind,sfc_fraction-
4)*LHV_correction;
                    % Total working fuel consumption
                    work_fc(i,j) =
ipseng_sfc(ips_id_pr)*usable_HP_ips_work(3)+load_fraction2*usable_HP_ips_wo
rk(4)*work_sfc; % [gr/hr]
                elseif ss_power(j)/n_ips_ss+prop_power(i)/n_ips_prop <
sum(usable_HP_ips_work(1:3)) && sum(usable_HP_ips_work(1:3)) >=
sum(usable_HP_ips_work(3:4))
                    % Both secondary engine at full load, 1st primary
                    % engine at partial load
                    load_fraction =
(ss_power(j)/n_ips_ss+prop_power(i)/n_ips_prop-
sum(usable_HP_ips_work(1:2)))/usable_HP_ips_work(3); % Secondary PGM
working at 100%
                    sfc_fraction = Lfraction(load_fraction);
                    sfc_ind = SFCind(ips_id_pr,system);
                    load_fraction2 =
100*load_fraction/elec_cor_mult(motor_fraction(i)-24); % Engine load
increases due to partial load motor drive and propulsion motor operation
                    plot_fraction_sec1(i,j) = 1;
                    plot_fraction_sec2(i,j) = 1;
                    plot_fraction_pr1(i,j) = load_fraction2;
                    work_sfc =
ipseng_sfc(ips_id_pr)/gen_sfc_FL(sfc_ind)*partial_sfc(sfc_ind,sfc_fraction-
4)*LHV_correction;
                    % Total working fuel consumption
                    work_fc(i,j) =
ipseng_sfc(ips_id_sec)*sum(usable_HP_ips_work(1:2))+load_fraction2*usable_H
P_ips_work(3)*work_sfc; % [gr/hr]
                elseif ss_power(j)/n_ips_ss+prop_power(i)/n_ips_prop <
sum(usable_HP_ips_work(2:4))
                    % 1st primary and 1st secondary engines at full load,
                    % 2nd primary engine at partial load
                    load_fraction =
(ss_power(j)/n_ips_ss+prop_power(i)/n_ips_prop-
sum(usable_HP_ips_work(2:3)))/usable_HP_ips_work(4); % 1 primary PGM
working at 100%
                    sfc_fraction = Lfraction(load_fraction);
                    sfc_ind = SFCind(ips_id_pr,system);
                    load_fraction2 =
100*load_fraction/elec_cor_mult(motor_fraction(i)-24); % Engine load
increases due to partial load motor drive and propulsion motor operation
```

```
                    plot_fraction_sec1(i,j) = 1;
                    plot_fraction_pr1(i,j) = 1;
                    plot_fraction_pr2(i,j) = load_fraction2;
                    work_sfc =
ipseng_sfc(ips_id_pr)/gen_sfc_FL(sfc_ind)*partial_sfc(sfc_ind,sfc_fraction-
4)*LHV_correction;
                    % Total working fuel consumption
                    work_fc(i,j) =
ipseng_sfc(ips_id_sec)*usable_HP_ips_work(2)+ipseng_sfc(ips_id_pr)*usable_H
P_ips_work(3)+load_fraction2*usable_HP_ips_work(4)*work_sfc; % [gr/hr]
                else
                    % 1st primary and both secondary engines at full load,
                    % 2nd primary engine at partial load
                    load_fraction =
(ss_power(j)/n_ips_ss+prop_power(i)/n_ips_prop-
sum(usable_HP_ips_work(1:3)))/usable_HP_ips_work(4); % 1 primary PGM
working at 100%
                    sfc_fraction = Lfraction(load_fraction);
                    sfc_ind = SFCind(ips_id_pr,system);
                    load_fraction2 =
100*load_fraction/elec_cor_mult(motor_fraction(i)-24); % Engine load
increases due to partial load motor drive and propulsion motor operation
                    plot_fraction_sec1(i,j) = 1;
                    plot_fraction_sec2(i,j) = 1;
                    plot_fraction_pr1(i,j) = 1;
                    plot_fraction_pr2(i,j) = load_fraction2;
                    work_sfc =
ipseng_sfc(ips_id_pr)/gen_sfc_FL(sfc_ind)*partial_sfc(sfc_ind,sfc_fraction-
4)*LHV_correction;
                    % Total working fuel consumption
                    work_fc(i,j) =
ipseng_sfc(ips_id_sec)*sum(usable_HP_ips_work(1:2))+ipseng_sfc(ips_id_pr)*u
sable_HP_ips_work(3)+load_fraction2*usable_HP_ips_work(4)*work_sfc; %
[gr/hr]
                end
            end
        end
    elseif ips_num_pr == 3
        for i = 1:i_speed
            prop_power(i) = SHP(i+1); % SHP starts with speed=0, change
with new profile
            motor_load_fraction(i) = prop_power(i)/SHP(dspeed+1); % motor
drive and propulsion motor load fraction
            motor_fraction(i) = Mfraction(motor_load_fraction(i));
            for j = 1:i_sspower
                if ss_power(j)/n_ips_ss+prop_power(i)/n_ips_prop <
usable_HP_ips_work(1)
                    % 2 engines at partial load
                    load_fraction =
(ss_power(j)/n_ips_ss+prop_power(i)/n_ips_prop)/sum(usable_HP_ips_work(1:2)
);
                    sfc_fraction = Lfraction(load_fraction);
                    sfc_ind = SFCind(ips_id_pr,system);
                    load_fraction2 =
100*load_fraction/elec_cor_mult(motor_fraction(i)-24); % Engine load
increases due to partial load motor drive and propulsion motor operation
                    plot_fraction_pr1(i,j) = load_fraction2;
                    plot_fraction_pr2(i,j) = load_fraction2;
```

```
                    work_sfc_ips =
ipseng_sfc(ips_id_pr)/gen_sfc_FL(sfc_ind)*partial_sfc(sfc_ind,sfc_fraction-
4)*LHV_correction;
                    work_fc(i,j) =
load_fraction2*sum(usable_HP_ips_work(1:2))*work_sfc_ips; % [gr/hr]
                elseif ss_power(j)/n_ips_ss+prop_power(i)/n_ips_prop <
sum(usable_HP_ips_work(1:2)) % 2 primary PGMs on line and 1 can work at
100% load
                    % 1st engine at full load, 2nd at partial load
                    load_fraction =
(ss_power(j)/n_ips_ss+prop_power(i)/n_ips_prop-
usable_HP_ips_work(1))/(usable_HP_ips_work(2));
                    sfc_fraction = Lfraction(load_fraction);
                    sfc_ind = SFCind(ips_id_pr,system);
                    load_fraction2 =
100*load_fraction/elec_cor_mult(motor_fraction(i)-24); % Engine load
increases due to partial load motor drive and propulsion motor operation
                    plot_fraction_pr1(i,j) = 1;
                    plot_fraction_pr2(i,j) = load_fraction2;
                    work_sfc_ips =
ipseng_sfc(ips_id_pr)/gen_sfc_FL(sfc_ind)*partial_sfc(sfc_ind,sfc_fraction-
4)*LHV_correction;
                    % Total working fuel consumption
                    work_fc(i,j) =
ipseng_sfc(ips_id_pr)*usable_HP_ips_work(1)+load_fraction2*usable_HP_ips_wo
rk(2)*work_sfc_ips; % [gr/hr]
                else
                    % 2 engines at full load, 3rd engine at partial load
                    load_fraction =
(ss_power(j)/n_ips_ss+prop_power(i)/n_ips_prop-
sum(usable_HP_ips_work(1:2)))/(usable_HP_ips_work(3));
                    sfc_fraction = Lfraction(load_fraction);
                    sfc_ind = SFCind(ips_id_pr,system);
                    load_fraction2 =
100*load_fraction/elec_cor_mult(motor_fraction(i)-24); % Engine load
increases due to partial load motor drive and propulsion motor operation
                    plot_fraction_pr1(i,j) = 1;
                    plot_fraction_pr2(i,j) = 1;
                    plot_fraction_pr3(i,j) = load_fraction2;
                    work_sfc_ips =
ipseng_sfc(ips_id_pr)/gen_sfc_FL(sfc_ind)*partial_sfc(sfc_ind,sfc_fraction-
4)*LHV_correction;
                    % Total working fuel consumption
                    work_fc(i,j) =
ipseng_sfc(ips_id_pr)*sum(usable_HP_ips_work(1:2))+load_fraction2*usable_HP
_ips_work(3)*work_sfc_ips; % [gr/hr]
                end
            end
        end
    end
end


%% Total fuel consumption for the given operating profile

fc_total = sum(sum(profile_data(2:end-1,2:end-2).*work_fc/1000))/100; %
Total fuel consumption [kg/hr]
```

```
%% Display results

figure
[X1,Y1] = meshgrid(profile_data(1,2:end-2),profile_data(2:end-1,1));
meshz(X1,Y1,plot_fraction_pr1),xlabel('Ship service power
[kW]'),ylabel('speed [knots]'),zlabel('Load fraction'),title('PGM1 Primary
Engine Load Fraction Profile')
figure
meshz(X1,Y1,plot_fraction_pr2),xlabel('Ship service power
[kW]'),ylabel('speed [knots]'),zlabel('Load fraction'),title('PGM2 Primary
Engine Load Fraction Profile')
figure
meshz(X1,Y1,work_fc/1000),xlabel('Ship service power [kW]'),ylabel('Speed
[knots]'),zlabel('Fuel Consumption [kg/h]'),title('IPS Fuel Consumption
Profile')

name_pr = char(ipseng_name(ips_id_pr));
type_pr = char(ipseng_type(ips_id_pr));
manuf_pr = char(ipseng_manuf(ips_id_pr));
name_sec = char(ipseng_name(ips_id_sec));
type_sec = char(ipseng_type(ips_id_sec));
manuf_sec = char(ipseng_manuf(ips_id_sec));

disp('------------------------------------------------------------------------
--')
disp('                  INTEGRATED POWER SYSTEM (IPS)')
fprintf('\t\t     OPTIMAL CONFIGURATION\n\n');
fprintf('Primary engines  :       %d %s %s (%s) / %dkW\n', ips_num_pr,
name_pr, type_pr, manuf_pr,ipseng_power(ips_id_pr));
if ips_sec_dec == 1
    fprintf('Secondary engines:       %d %s %s (%s) / %dkW\n', ips_num_sec,
name_sec, type_sec, manuf_sec,ipseng_power(ips_id_sec));
    figure
    meshz(X1,Y1,plot_fraction_sec1),xlabel('Ship service power
[kW]'),ylabel('speed [knots]'),zlabel('Load fraction'),title('PGM1
Secondary Engine Load Fraction Profile')
    figure
    meshz(X1,Y1,plot_fraction_sec2),xlabel('Ship service power
[kW]'),ylabel('speed [knots]'),zlabel('Load fraction'),title('PGM2
Secondary Engine Load Fraction Profile')
else
    figure
    meshz(X1,Y1,plot_fraction_pr3),xlabel('Ship service power
[kW]'),ylabel('speed [knots]'),zlabel('Load fraction'),title('PGM3 Primary
Engine Load Fraction Profile')
end
fprintf('\n')
fprintf('Average fuel consumption for given combined profile: %5.1f
kg/hr\n', fc_total);
disp('------------------------------------------------------------------------
--')
```