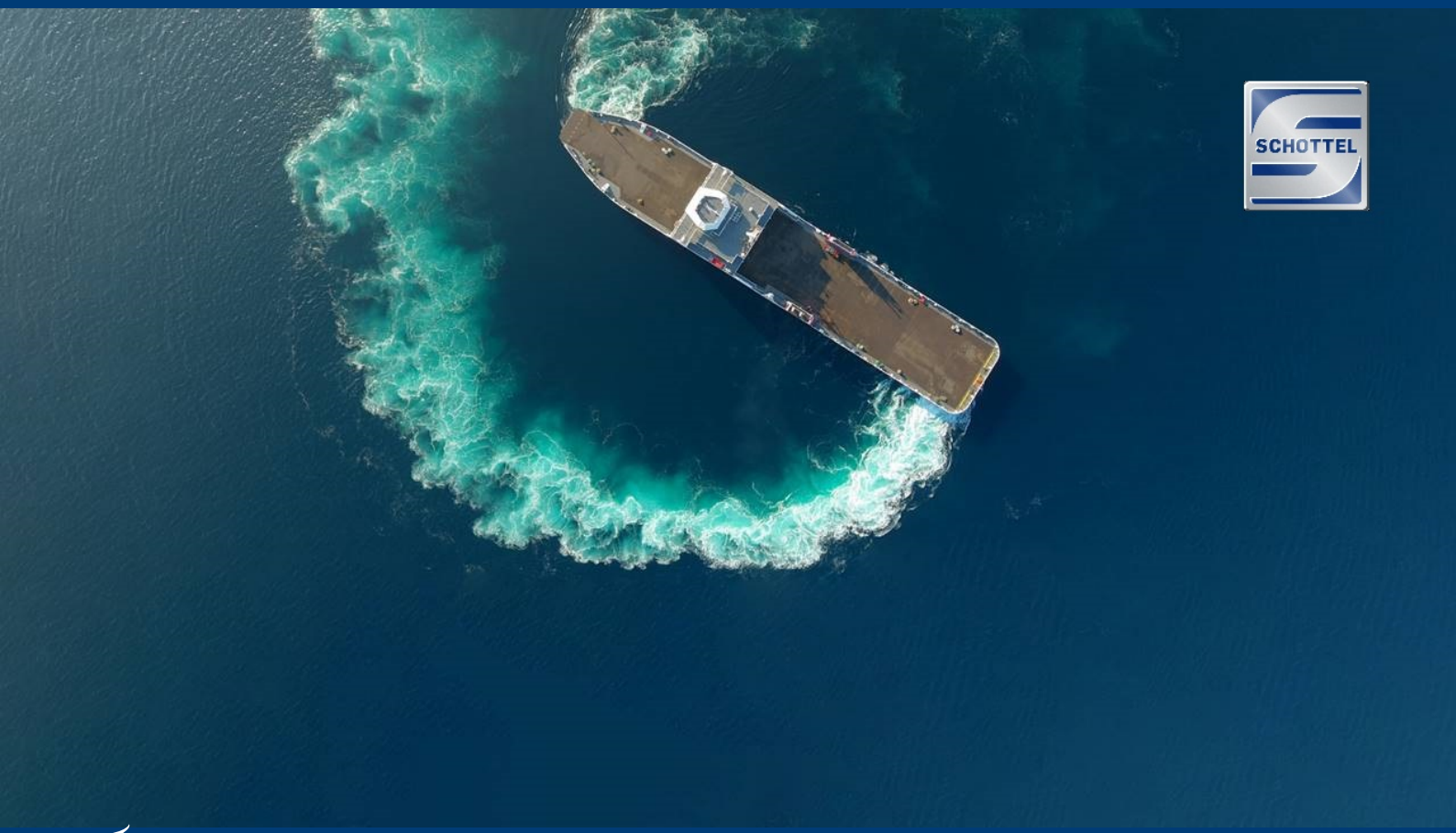
Development of a Combinator Curve Generator

S. A. Sharoubim

MT54035 Master Thesis
SDPO.20.042.m



TU Delft

Development of a Combinator Curve Generator

by

S. A. Sharoubim

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Wednesday June 23, 2021 at 13:30 PM.

Conducted at SCHOTTEL GmbH, Dörth.

SDPO.20.042.m

Student number:	4132475	
Thesis committee:	Ir. K. Visser,	TU Delft, chair
	Dr. ir. P. de Vos,	TU Delft, supervisor
	Dr. P. Mertes,	SCHOTTEL GmbH, supervisor
	Ir. T. Tillack,	SCHOTTEL GmbH, daily supervisor
	Dr. ir. H.J. de Koning Gans,	TU Delft

This thesis is confidential and cannot be made public until June 23, 2023.

TU Delft

SCHOTTEL

Abstract

Often, vessels have multiple operation modes that are specialised for a certain task. If a vessel is employed with a controllable pitch propeller(CPP), the blade pitch can be adjusted, adding a degree of freedom to the system. This advantage creates the possibility to increase the diversity of operation modes of the vessel and allows for flexibility, precision and specialisation for a certain task. Additionally, CPPs can be controlled such, that operational limits of the driving machinery are not exceeded.

The control of CPPs is done with the use of pre-determined settings for the propeller pitch and shaft speed per position of the lever, the collection of these is referred to as a combinator curve. As vessels have become increasingly diverse with respect to their functional abilities, and complex with respect to their propulsion configurations, design of combinator curves becomes increasingly labour intensive. Earlier developed software applications, that aim to support the matching process or combinator settings, lack clear insight of important performance indicators and their impact on the combinator curve design.

In this thesis, a Combinator Curve Generator(CCG) is developed to support the design of combinator curves for vessels that employ CPPs, in order to decrease the labour intensity of the combinator design process. Further, optimisation approaches to set up combinator curves for certain operation modes have been developed in terms of important performance criteria. The approaches are implemented in the CCG such that a combinator curve can be designed, optimised and evaluated.

Resulting from literature research, important performance indicators for which combinator curves should be optimised are identified and quantified. Furthermore, propulsion configuration related constraints are identified and quantified.

For each of the identified performance indicators and their quantification, an optimisation approach has been developed and implemented. In order to implement the procedures, a calculation structure is developed together with algorithms for the approaches. The resulting optimised combinator curves are evaluated on the basis of expected behaviour.

In order to develop the CCG, a design goal is first formulated. Then, on the basis of the programming approach, structures for each of the data models and the work flow model are developed. Finally, a user interface is developed, with the possibility to insert required data and to design, optimise, and evaluate a combinator curve.

The four identified important performance indicators are propeller efficiency, cavitation inception, engine efficiency and fuel consumption per unit time. Various propulsion configurations can be taken into account, these include any type of main engine for which an operational envelope and SFOC map is available, any number of same size and type connected main engines per shaft and constant power take off or power take in.

The optimisation approach of a combinator curve in terms of propeller efficiency is based on the open water efficiency of the propeller open water characteristics (OWCs). The resulting combinator curve shows expected behaviour and can be used for optimisation and evaluation in terms of propeller efficiency.

For the optimised combinator curve in order to minimize the risk of cavitation inception, a new approach was developed which is different from a, in principle proposed quantification method. For this newly developed approach, cavitation inception diagrams are generated for a range of pitch settings, which contain information about the region where cavitation inception is expected to occur. The combinator curve is optimised in terms of the advance coefficient, in order for the operating point to be located such, that the risk of cavitation inception is limited for both pressure and suction side cavitation. The same software from which these cavitation inception diagrams are generated, is used for validation of the resulting optimised combinator curve in terms of cavitation inception.

Optimisation of a combinator curves in terms of engine efficiency and fuel consumption per unit time are both dependent on the minimal specific fuel consumption. For this purpose, an SFOC(Specific Fuel Oil Consumption) map of the main engine is required. If the SFOC map is known, its data can be inserted and used

for the optimisation method. For the optimisation approach in terms of engine efficiency, the minimal specific fuel consumption determines the optimal combination of pitch and shaft speed. For the optimisation approach in terms of fuel consumption per unit time, the minimal product of the specific fuel consumption and the required brake power result in an optimal combination of pitch and shaft speed for each lever position. Both resulting combinator curves show expected behaviour, and can be used in order to optimise and evaluate a combinator curve in terms of these engine related performance indicators.

Besides the set up of optimised combinator curves, there are options to set up a combinator curve manually or for standard operation modes. The first standard operating mode is one where the combinator curve is set up by calculating the combinator settings such, that the pitch setting is constant unless the operating limits of the propeller or engine are exceeded. The second operating mode is the constant speed mode, for which the combinator settings are calculated such that, at each lever position the shaft speed setting is equal to the maximum shaft speed of the engine.

Finally, an additional trip simulation tool is developed and added to the CCG, in order to determine and evaluate the total fuel consumption of a trip for different cruise speeds and a certain time duration, whilst taking into account the operational profile of the vessel, the combinator settings and hotel load.

Important recommendations for further development include extension of the database of inception diagrams for propellers with different blade area ratios, and the broadening of the propulsion configuration scope, such that different main engines and power supply systems can be considered. Finally, it is recommended to research the possibility to calibrate the effective angle of attack method on the basis of the optimisation approach proposed in this thesis.

Acknowledgements

This graduation project was provided by SCHOTTEL GmbH in Dörth, Germany. I would like to thank the company for this opportunity. Thank you, Jasper Grevink, for introducing me to SCHOTTEL, and thank you for the support during my stay in Germany. I would like to thank all colleagues at the hydrodynamic department for their support and conversations during the time I worked at the office. I will not forget the kindness with which I was treated.

I would like to thank Paul Mertes and Thorsten Tillack for the formulation of the project and your recommendations on how to approach the assignment at different stages of the process. In particular, I would like to thank Thorsten Tillack, my daily supervisor. It was motivating and challenging to be supervised by you. I also enjoyed and appreciated our sparring sessions with Hagen Lippok about programming issues or different approaches that could be taken during the project. You both can explain difficult and basic things very well and I learned a lot from your ability to break down complex problems into smaller parts.

I would like to thank Peter de Vos for guiding me through the formal stages of the graduation process and for your support and understanding in the final stages of writing my thesis.

I want to thank my parents, my brothers and my husband for helping me move, for supporting me and believing in me. Special thanks to my husband, Martijn Kist, for your patience, you are my best friend.

Finally, I could mention the Covid-19 crisis and how it impacted the past year, but no matter the circumstance, the following quote applies.

"Never give in. Never give in. Never, never, never, never–in nothing great or small, large or petty–never give in, except to convictions of honour and good sense. Never yield to force. Never yield to the apparently overwhelming might of the enemy."

- Winston Churchill, London

S. A. Sharoubim
June 2021

Contents

Abbreviations

ANN	Artificial Neural Network
CCG	Combinator Curve Generator
CFD	Computational Fluid Dynamics
CIP	Cavitation Inception Prevention
CODLAG	Combined Diesel Electric And Gas
CODOG	Combined Diesel Or Gas
CPP	Controllable Pitch Propeller
CSR	Continuous Service Rating
EAR	Expanded Area Ratio
EEDI	Energy Efficient Design Index
EM	Engine Margin
FPP	Fixed Pitch Propeller
GA	Genetic Algorithm
GUI	Graphical User Interface
IMO	International Maritime Organisation
LNG	Liquid Natural Gas
MCR	Maximum Continuous Rating
OOP	Object Oriented Programming
OWC	Open Water Curves
PCS	Propulsion Control System
PSO	Particle Swarm Optimisation
PTI	Power Take In
PTO	Power Take Off
PWM	Pulse Width Modulator
RTBO	Reduced Time Between Overhaul
SFOC	Specific Fuel Consumption
SM	Sea Margin

Nomenclature

Symbols

A	Surface	[m^2]
B	Width	[m]
c	Camber	[m]
c_1	Correction factor	[-]
D	Diameter	[m]
f	Thickness	[m]
fc	Fuel consumption	
g	Gravitational acceleration	[m/s^2]
h^L	Lower heating value	[J/kg]
i	Ratio	[-]
J	Advance ratio	[-]
k	constant [-]	
K_Q	Torque coefficient	[-]
K_T	Thrust coefficient	[-]
Lpp	Length between perpendiculars	[m]
M	Torque	[Nm]
m	Mass	[m]
n	Rotiational speed	[1/s]
P	Power, Pitch	[W], [m]
p	Pressure	[Pa]
per	Pollutant emission ratio	[g/kg]
Q	Torque, Heat	[Nm], [J]
R	Resistance	[N]
rpm	Rotations per minute	
sfc	Specific fuel consumption	[g/kWh]
spe	Specific pollutant emission	[g/kWh]
t	Thrust deduction factor	[-]
T	Thrust	[N]
T	Draft	[m]
v	Speed	[m/s]
w	Wake fraction	[-]
W	Work	[J]
X	Fuel rack position	[mm]
z	Propeller draft	[m]
Z	Number of blades	[-]

α	Angle	[deg]
β	Inflow angle	[deg]
η	efficieny	[-]
ρ	Density	[kg/m^3]
σ	Cavitation number	[-]
θ	Propeller pitch	[deg]

Subscripts

a	atmospheric, advance
B	brake power
d	design
e	engine
E	effective
eff	effective
f	fuel
GB	gearbox
n	index, number
O	open water
opt	optimum
p	propeller
pd	pitch diameter ratio
PTI	power take in
PTO	power take off
R	rotative, radius
s	ship
S	shaft
sw	seawater
T	thrust
TRM	transmission
v	vapour

1 Introduction

1.1. Motivation

Vessels are often not designed for one specific purpose, but they have multiple operation modes that are specialised for a certain task. Furthermore, ship owners may have various requests depending on the vessel. This thesis deals with vessels that employ a controllable pitch propeller(CPP), which is a propeller of which the blade pitch can be adjusted and results in a different angle of attack. Therefore, a different generated thrust i.e. ship speed, at the same rotational speed of the shaft. This variation in pitch adds a degree of freedom to the system [6]. Vessels that are equipped with a fixed pitch propeller(FPP) often have one main operation mode for which the vessel is designed. If a vessel is employed with a CPP, the diversity of operation modes can be increased depending on the intended use of the vessel. Not only does this allow for flexibility, it also allows for precision and specialisation for a certain task. Utilisation of CPPs can significantly increase vessel manoeuvrability [6]. In addition CPPs can be controlled such, that operational limits of the driving machinery are not exceeded [25].

CPP control is done with the use of pre-determined settings for the propeller pitch and shaft speed per position of the lever. Each lever position corresponds to a certain ship speed demand. The combination of propeller pitch and shaft speed are determined such, that the thrust demand at this ship speed can be generated. The collection of these settings are referred to as a combinator curve. For each designated operation mode, these are design and off-design conditions of a vessel [49], a combinator curve is designed which then serves as input for the control of the CPP.

For cruise ships and yachts for instance, a high level of comfort is important and thus noise, as a result of propeller cavitation, should be minimized as much as possible. At the same time there might be the requirement to be able to sail at high speed for a short amount of time. For a vessel that is designed in order to sail a certain route to transport cargo or individuals from A to B, a certain trip time can be crucial with respect to economic viability. In this case it is important to be able to design a combinator curve that is optimised such that this time requirement can be satisfied. Because of the ability to simultaneously adjust the pitch and shaft speed it is also possible to sail in an efficient mode with respect to the propulsion system or on the other hand a fuel saving mode in order to save costs. In order to design a combinator curve for certain criteria, insight and quantification of system performance is crucial such that a certain combinator curve design can be evaluated and possibly compared to other designs.

In recent years, the complexity of propulsion and power generation configurations has increased due to vessels that require flexibility in terms of their operation modes. An example of vessels that require such flexibility are harbour tugs. They require a large amount of power for thrust during towing operations, but do not need this amount of power during transit. While the harbour tug operates at a low part load of its full power, the engine runs at a very in-efficient operating point in terms of fuel consumption and emissions. A solution for this can be to install a hybrid propulsion system such that the variety of power demand can be facilitated.

Additionally, there is a transition to alternative fuels, of which the motivation to do so is mainly an ethical one. This transition causes a change of conventional vessel design aspects, especially with regards to the

propulsion configuration and its system layout. Depending on the development of both research and commercial availability it is yet to be seen which alternative fuels will eventually be used the most. It is expected that by the year 2050 there will be a mix of various fuel options, and oil will stay the main fuel option with liquid natural gas(LNG) as the second most used fuel [11]. This means that diesel, dual fuel and gas engines are here to stay for at least a few more decades, albeit in combination with an alternative power supply.

Because vessels have become increasingly diverse with respect to their functional abilities. And complex with respect to their propulsion configurations. The design of a combinator curve for each operation mode for each new vessel, becomes increasingly labour intensive.

Furthermore, the urgency to reduce environmental impact on a global scale has increased. The environmental impact that is caused by the shipping industry includes pollution of air, water, oil and noise [37]. The International Maritime Organization (IMO) has several committees that each focus on different aspects of the marine industry. One of them is the Marine Environmental Protection Committee who develop strategies to prevent and control pollution from ships, including reduction of greenhouse gasses. Regulations for air pollution are addressed in MARPOL Annex VI [22]. IMO adopted a strategy to reduce the total yearly amount of shipping emissions with 50% by 2050 compared to 2008 [21]. As a result, various regulations have come into effect of which the most recent came into effect in January 2020. These recent regulations limit the amount of sulphur oxides (SO_x) in fuel oil to 0.5% (mass by mass) on board ships that are operating outside of a designated emission control area. Such regulations have an impact on the design and operation of a propulsion system as a whole. Ship owners are therefore motivated to seek for those systems that include technical solutions to improve propulsion and power plant performance.

With a view on innovation for future developments, propulsion systems utilizing CPPs have been used in studies to design control strategies for automation. These studies have shown that there is significant potential to reduce environmental impacts by improving cavitation inception, fuel consumption, engine loading and manoeuvring behaviour [12, 49].

Finally, regulations for fuel consumption and its emissions are often based on the maximum or nominal power of the engine. In doing this the actual fuel consumption and emissions are not taken into account because the operational profile of a ship is not taken into account [25]. If the intended operational profile of a vessel is known together with an expected amount of auxiliary power, the fuel consumption and emissions can be estimated on the basis of the combinator curve. This can also give insight into the expected fuel costs. Both from an economics point of view and from the perspective of environmental impact it is required to take into account the operational profile.

1.2. Background

This graduation project was provided by the hydrodynamic department of SCHOTTEL GmbH, located in Dörth, Germany. SCHOTTEL GmbH is specialised in the development, design, production and marketing of azimuthing propulsion and manoeuvring systems with power requirements up to 30 MW. The company was founded in 1921 in Spay, Germany and has almost 60 years of experience with the development and manufacturing of propulsion systems. Currently, there are many sales and service locations all over the world [16].

At the hydrodynamic department, efficient propulsion and manoeuvring systems are being developed and designed for a wide range of vessels. The solutions contain azimuthing rudder propellers for main or auxiliary operation, conventional shaft lines, pump jet propulsion and tunnel thrusters used for manoeuvring. Depending on the application and operations, either a CPP or a FPP can be chosen for each of these systems.

1.3. Literature Review

The problem that the design of combinator curves becomes increasingly labour intensive calls for a software application with which the user can design a combinator curve in a flexible user environment. In essence the set up of a combinator curve is a matching problem, i.e. the matching of a ship, a propeller and the driving machinery such that certain criteria are met. In order to solve the matching problem there have been various attempts to develop a tool which facilitates this. Here, a short overview of literature is given on the ship-propeller-engine matching tools that have been developed.

Ship-Engine-Propeller Matching
Already in 1981 Broome and Lambert [5] developed an interactive computer program to facilitate ship-engine-propeller matching for propulsion configurations with a diesel engine, possibly geared drive, and a FPP. Publications of other tools have been mostly developed in the past decade and have often been designed for a specific propulsion configuration or for an engine-propeller selection application.

Ogar et al. [31] developed a tool in order to facilitate the matching of a CODOG propulsion configuration employing a CPP. This software was then used by Bob-Manuel and Okim [3] to investigate the matching problem for a F90 Frigate with the objective to minimize fuel consumption and cavitation. The matching point is only found in the design point and one off-design point and is presented in the propeller open water diagram. No pitch and shaft speed setting for other ship speeds have been presented and no representation of the propeller load in an engine diagram or power absorption diagram for the complete propulsion plant have been shown.

Engine-Propeller Selection
Next, Marques et al. [30] developed a tool with an approach to perform engine-propeller selection for an LNG carrier under rough weather. The propulsion configuration included a dual-fuel diesel engine and a FPP. Continuing with engine-propeller matching tools, Habibi [19] provides a basic tool to produce a matching point for a main diesel engine and a FPP. Finally Lin et al. [27] provides an engine-propeller selection tool, also for diesel engines and FPPs. Despite these last two publications being more recent (2019 and 2015 resp.), the matching point was found without taking into account fuel consumption, emissions or cavitation objectives. In that sense these tools were a reproduction of what was already done before.

Research on Matching Problems
Then, research has been performed where objectives are optimised together in order to investigate the effect of the objective(s) on the matching problem. Coraddu et al. [7] developed a computational tool to investigate cavitation inception prevention and fuel consumption on the optimum matching point. This research was done for a CODLAG propulsion configuration and CPP. In this paper the pitch settings and shaft speed settings were shown for the range of ship speeds, but no representation of the propeller load in an engine diagram.

Important current research is mainly directed towards fuel consumption and emissions reduction. Altosole et al. [1] developed a tool to predict the service performance in terms of ton-mile fuel consumption and exhaust emissions. The prediction included the main and auxiliary diesel engines with CPP. However, the prediction of exhaust emissions could not be validated due to a lack of emissions measurement data of ships in service. Furthermore, the propeller pitch and shaft speed settings have been presented for various ship speeds in an engine diagram, but not in a clear manner in terms of the lever positions.

Next, Ren et al. [33] investigated the influence of the energy efficient design index (EEDI) on ship-engine-propeller matching and used a Genetic Algorithm (GA) to find the optimum solution. This research was mainly based on a low-speed diesel engine and direct driven with FPP.

Optimisation Algorithms to Solve Matching Problem
Also the use of various optimisation algorithms have been investigated. Ren and Diao [34] investigated the effectiveness of GA with a Particle Swarm Optimisation (PSO) operator on ship-engine-propeller matching, in this case for a trawler. The propulsion configuration of this fishing vessel was not specified. Further, to investigate off-design weather and ship-state conditions Koenhardono et al. [26] developed a tool that uses an artificial neural network (ANN) for a trimaran patrol ship with the objective to minimize fuel consumption. The propulsion configuration of this patrol ship includes two main diesel engines with sequential turbochargers and gearboxes and uses a CPP. The output of this application consists of several parameters including the pitch and shaft speed settings for several ship speeds. Finally, Liu and Fan [28] compared the use of GA to PSO to investigate efficiency and cost optimisation on ship-propeller engine matching. No specification was given about the ship, propeller or engine.

Discussion
Based on this literature overview it can be concluded that there is no lack of applications to match the ship-engine-propeller in the design point. In doing so, any research that is aimed to estimate the fuel consumption and emissions are based on the design point and not on the operational profile of the vessel.

There have been multiple studies to match the ship-engine-propeller while optimising for one or more objectives. In the applications that included a propulsion system with a CPP however, the resulting combi-

nator curve was not presented in a clear manner such that it can be used as control input. Next to this, there is no quantification and clear insight of important performance indicators and their impact on the combinator curve design. Because of this the process is sensitive to a level of subjectivity from the engineer and it is difficult to compare one combinator curve design to another.

The publications which are of real value and benefit to current requirements are those with research towards cavitation, fuel and emissions reduction due to the impact on system performance and increasing regulations to help reduce environmental impact. Important recommendations for further research are given by Ren et al. [33] and include investigation for other emission indexes and a variety of propulsion configurations.

1.4. Objectives and Research Questions

In this thesis a Combinator Curve Generator (CCG) is developed to support the design of combinator curves for vessels that employ controllable pitch propellers. Such an application requires the following aspects:

- It should be possible to optimise a combinator curve for a certain operation mode in terms of important performance criteria.
- The calculated or optimised combinator curve should be set up and presented in a form such that it can be used as control input for the CPP in terms of the blade pitch and shaft speed setting per lever position, as per the industry standard to apply Single Lever Command.
- System performance indicators should be quantified such that the combinator curve design can be evaluated and compared to other combinator curve designs.
- It should be possible to consider a variety of propulsion configurations.
- It should be possible to take into account the operational profile of a vessel.

In order to develop the CCG, approaches to set up combinator curves for certain operation modes are developed in terms of important performance criteria. Additionally, a software application is developed in which the approaches are implemented such that a combinator curve can be designed, optimised and evaluated. The objectives and research questions are formulated as follows:

1. Develop and implement approaches to set up combinator curves for different operation modes of a vessel and its propulsion configuration in terms of important performance criteria.
   (a) What important performance indicators can be identified for which a combinator curve can be optimised and how can these be quantified?
   (b) Which propulsion configuration related constraints can be identified and how can they be taken into account in the development of the CCG?
   (c) What approach should be taken to optimise a combinator curve for a certain performance indicator?
2. Develop a software application for the design, optimisation and evaluation of combinator curves.

1.5. Approach

In order to realize the objectives for the development of the CCG the following approaches have been taken.

- **Literature Research**
  In order to answer the research questions for the first objective, and to start the development of the CCG, a literature research is performed. In this research several performance indicators for which a combinator curve can be optimised, are identified. Comparing several methods, an appropriate quantification method is chosen for each performance indicator. Furthermore, propulsion configurations are discussed and choices are made about constraints and what configurations are taken into account, for the development of the CCG. Finally, the approach to develop the CCG and optimisation approaches are discussed.

- **Development and Implementation of Approaches**
  For each of the identified performance indicators an optimisation approach has been developed. In order to implement these, a calculation structure is developed together with algorithms for the approaches. The resulting combinator curves are evaluated on the basis of expected behaviour.

- **Design of Combinator Curve Generator**
  In order to develop the CCG a design goal is first formulated. Then, on the basis of the programming approach, structures for each of the data models and the work flow model are developed. Finally, a user interface is developed, with the possibility to insert required data and to design, optimise, and evaluate a combinator curve. Either manually or for a certain operation mode.

1.6. Thesis Outline

This thesis contains 8 chapters and a short description is given for each of them:

In Chapter 2 theory on which the calculation procedures for the set up of a combinator curve are based are discussed. Firstly, resistance and propulsion theory is laid out. Then, the matching process is explained for operation in design and off-design conditions. Finally, combinator curve control and several operation modes are discussed.

In chapter 3 the literature research is performed in order to answer the proposed research questions. Firstly, performance indicators are identified and quantified, after which propulsion configuration related constraints are discussed. Then, the CCG design approach and optimisation methods are discussed. Finally, the chapter is concluded by answering the research questions.

In Chapter 4 the development of the CCG is explained. First the design goal and programming approach are formulated and explained. Then, the data structure is defined and data models are developed accordingly. Furthermore, the work flow structure and resulting user interface are presented and discussed. Finally, the structure and user interface for the simulator are described.

In Chapter 5 the development and implementation of the approaches are explained. First, the structure of a calculation core is discussed. Followed by an introduction of the benchmark with which the optimisation approaches are tested and evaluated. Furthermore, for each standard calculation option, the approach and resulting combinator curve is discussed and evaluated.

In Chapter 6 the optimisation approaches for the optimisation of combinator curves in terms of the propeller related performance indicators are explained. Furthermore, the resulting combinator curve is discussed and evaluated. Finally, the chapter is concluded.

In Chapter 7 the optimisation approaches for the optimisation of combinator curves in terms of the engine related performance indicators are explained. Furthermore, the resulting combinator curve is discussed and evaluated. Finally, the chapter is concluded.

In Chapter 8 the thesis is concluded in which the results are summarised and recommendations are given for future research and further development of the CCG.

2

Combinator Curve Theory

The basis for determining the operating point for each lever position that make up the combinator curve is explained with theory on resistance and propulsion and then the matching of the propeller load to the driving engine. In this chapter, theory on which the calculation procedures for the set up of a combinator curve are based, will be discussed.

2.1. Resistance and Propulsion

For the design of a combinator curve for a certain operation mode the vessel speed is not only determined for the design point, but for the range from standstill to maximum speed. Operation of the ship is done via the lever which can be put in a number of positions that each correspond to a vessel speed demand. In order to achieve a certain vessel speed, the ship resistance at that speed must be overcome. The required thrust demand to overcome the ship resistance can be determined when details about the speed demand and the corresponding resistance parameters are known. In this section the theory on resistance and propulsion are discussed. In Figure 2.1 a simple drawing is shown to help visualise the principles that will be discussed in this section.

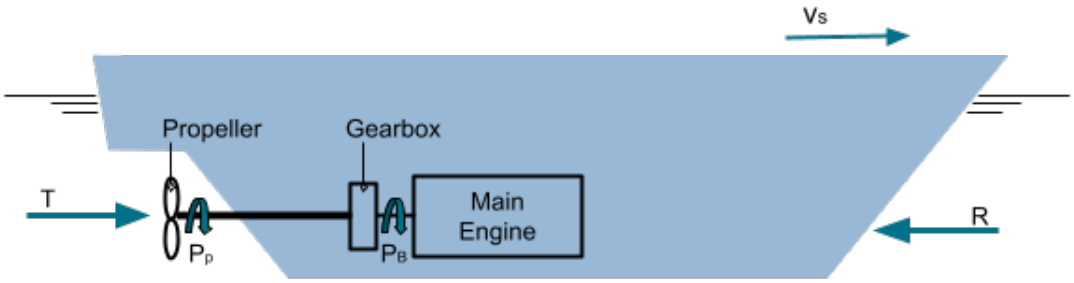


Figure 2.1: Sketch of ship and propulsion system.

The first step in the process of calculating the operating parameters for the propulsion system is to determine the ship resistance $R[N]$. Ship resistance is the force that is required to translate a ship through water [25]. Resistance data is obtained from a model of the ship that is towed in a towing tank at various ship speeds $v_s[m/s]$. The full scale resistance data is then predicted by scaling this data as a function of speed, fouling, hull form, appendages, sea state and water depth. The effective power $P_E[W]$ required to overcome this resistance can then be defined as shown in Equation 2.1.

$$P_E = R \cdot v_s \tag{2.1}$$

The thrust demand $T[N]$ to achieve the ship speed demand is normally higher then the ship resistance at that speed. This is because the propeller draws the water along the hull which creates an under pressure at the aft of the ship, generating an additional force in the same direction of the resistance, and thus adding to the total resistance. This added resistance is accounted for by using a thrust deduction factor $t[-]$. The

total required thrust is divided among the propellers $k_p[-]$, and the thrust per propeller is now calculated as shown in Equation 2.2.

$$T = \frac{R}{k_p \cdot (1-t)} \tag{2.2}$$

Also a boundary layer is formed along the hull that influences the water stream at the location of the propeller disc. The wake field at the propeller is therefore not uniform. Next to this the water flows at a certain angle due to the hull form at the aft of the ship. The velocity at the location of the propeller is called advance velocity $v_a[m/s]$ and is different from the ship speed because of these phenomena. This difference in speed of the water at the propeller is expressed as a ratio and is defined by the wake factor $w[-]$, see Equation 2.3.

$$w = \frac{v_s - v_a}{v_s} \tag{2.3}$$

The required thrust power $P_T[W]$ is now larger than the effective power and is defined as:

$$P_T = T \cdot v_a \tag{2.4}$$

And the ratio between the effective power and the thrust power is expressed as the hull efficiency $\eta_H[-]$:

$$\eta_H = \frac{P_E}{k_p \cdot P_T} = \frac{R \cdot v_s}{k_p \cdot T \cdot v_a} = \frac{1-t}{1-w} \tag{2.5}$$

This is the first efficiency that we find in the calculation steps from the ship resistance to engine break power in order to sail the ship in a certain operating condition. These steps are part of a chain of powers and efficiencies which can be presented as a propulsion chain that can be found in the book from Klein Woud and Stapersma [25].

The next step concerns the propeller. In this thesis the propeller is a CPP, which means that various combinations of pitch and rotational speed can deliver the same thrust demand. And to operate the propeller at a given pitch and rotational speed $n_p[rpm]$ the necessary torque $Q[Nm]$ must be generated. As with the ship resistance data, full scale performance characteristics of the propeller are predicted on the basis of model scale tests. A model is produced from the propeller design and is then tested in a towing tank. Then the thrust and torque are measured for a range of translation velocities, propeller rotational speeds and several pitch settings. These measurements are expressed as non-dimensional coefficients; advance ratio $J[-]$, thrust coefficient $K_T[-]$ and torque coefficient $K_Q[-]$, see Equations 2.6-2.8.

$$J = \frac{v_a}{n_p \cdot D_p} \tag{2.6}$$

$$K_T = \frac{T}{\rho \cdot n_p^2 \cdot D_p^4} \tag{2.7}$$

$$K_Q = \frac{Q}{\rho \cdot n_p^2 \cdot D_p^5} \tag{2.8}$$

With propeller diameter $D_p[m]$ and sea water density $\rho[kg/m^3]$. Using these non-dimensional coefficients, the full scale advance speed, thrust and torque can be determined for the operating point by inserting the full scale propeller speed and propeller diameter parameters in the relations.

The next efficiency, the open water efficiency $\eta_O[-]$, is defined as the ratio of thrust power and the power delivered to the propeller. This definition can be rewritten using the non-dimensional coefficients which results in the following expression, see Equation 2.9.

$$\eta_O = \frac{K_T}{K_Q} \cdot \frac{J}{2\pi} \tag{2.9}$$

In contrast to the behind condition in full scale operation, the flow in front of the propeller is uniform in the towing tank. The actual torque $M_p[Nm]$ delivered to the propeller differs slightly from the open water torque and their ratio is defined as the relative rotative efficiency $\eta_R[-]$, see Equation 2.10.

$$\eta_R = \frac{Q}{M_p} \tag{2.10}$$

Now, the required propeller power $P_p[W]$ can be written as a combination of the expression for the torque coefficient from Equation 2.8 and Equation 2.10, resulting in Equation 2.11.

$$P_p = 2\pi \cdot M_p \cdot n_p = 2\pi \cdot K_Q \cdot \rho \cdot n_p^3 \cdot D_p^5 \cdot \frac{1}{\eta_R} \tag{2.11}$$

The required propeller power is delivered by one or more engines through a shaft and depending on the propulsion configuration also through a gearbox. The shaft losses are expressed in terms of shaft efficiency $\eta_S[-]$, which is the ratio from shaft power to propeller power. Then, gearbox losses are in terms of brake power to shaft power while also taking into account the number of engines, resulting in a gearbox efficiency $\eta_{GB}[-]$. The total transmission efficiency $\eta_{TRM}[-]$ is defined by the ratio of delivered propeller power to brake power and it can be written as the product of the two former mentioned efficiencies. Thus defined as in Equation 2.12.

$$\eta_{TRM} = \eta_S \cdot \eta_{GB} \tag{2.12}$$

Then finally the effective brake power $P_B[W]$ can be determined by Equation 2.13.

$$P_B = \frac{P_p}{\eta_{TRM}} \tag{2.13}$$

These were the basic principles to calculate through the propulsion train from ship resistance to engine break power.

2.2. The Matching Process

The next step in the determination of a combinator curve is the matching process. Klein Woud and Stapersma [25] define this as *"... the process in which the operating envelope of the driving machinery and the load as experienced by the driving machinery are tuned to each other."* In this section this matching process will be explained.

There are two criteria that should be met when matching the propeller to the engine [24]. The first is to make sure that the engine can develop full, or nearly full power at the design condition. The second is that the propulsion plant should be able to function without exceeding the operating limits of the propeller and the engine. Other criteria can be any performance indicator such as fuel consumption or cavitation behaviour.

When all data concerning the ship, the propeller and the driving machinery is known, the operating point of the propeller for the speed at each of the lever positions can be determined for several operating modes for the vessel in question. However, before the design of combinator curves that serve as control input, already some basic matching principles will have been performed. This is because the design of a propulsion system is an iterative process and throughout the design process the hull form, the propeller and the engine are optimised for the design operation of the vessel. These basic matching principles will be explained first.

2.2.1. Operation in Design Point

Basic matching of the ship, the propeller and the engine is first performed for the design condition of the vessel. When a new project for a propeller design is initiated, there are no propeller characteristics available to determine the operating points of this new propeller in the design point. This means that the propeller load curve in the design point must be estimated. At this point the design condition of the ship and its resistance data is known for a range of speeds. For the purpose of explaining the basic matching principles, it is assumed that the engine envelope is known. And with regards to the propeller, it is assumed that the propeller diameter and speed are chosen.

Starting with this information, the estimation of a propeller load curve for a certain propulsion system can be done by using propeller characteristics from a similar propeller unit. Or, by using propeller data from research such as the Wageningen C-D series [9]. In Figure 2.2 an example is shown of results from open water tests for a CPP from the Wageningen C-D series. Here open water efficiency η_O, thrust coefficient K_T and torque coefficient K_Q are plotted against the advance coefficient J for a certain CPP. Note, that each set of η_O, K_T and K_Q curves belong to a certain pitch of the propeller. The thrust that must be delivered by the propeller is dependent on the resistance for the design speed, the required thrust and the amount of propellers (Eq. 2.2). Now the required thrust coefficient $K_{T,ship}$ can be determined with Equation 2.7. This equation can be rewritten so that it becomes a function of advance ratio J, see Equation 2.14 [25].

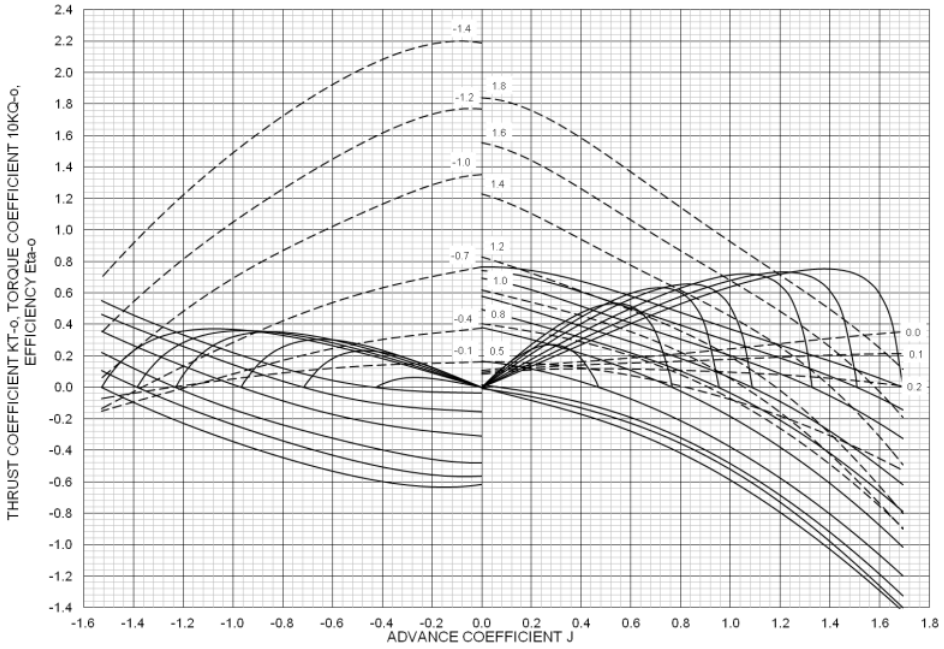


Figure 2.2: Open water propeller characteristics for a CPP from the Wageningen C-D series . Taken from Dang et al. [9].

$$K_{T,ship} = \frac{1}{\rho \cdot D_p^2} \cdot \frac{R}{v_s^2 \cdot k_p \cdot (1-t) \cdot (1-w)^2} \cdot J^2 \tag{2.14}$$

The resulting ship curve is then plotted in the open water diagram as shown in Figure 2.3. In the figure it shows that the ship curve intersects the open water curves(OWCs). The intersection with the thrust coefficient curve determines the operating point of the propeller. Next to the thrust coefficient K_T also the torque coefficient K_Q and open water efficiency η_O can be determined. Knowing the propeller speed, the advance ratio J can be determined from Equation 2.6. Had the ship curve been drawn in Figure 2.2, the ship curve would intersect several thrust coefficient curves at this advance ratio. From these, one can be chosen as the design pitch. If the propeller design is done for high efficiency for example, the propeller pitch is chosen for the operating point that results in the highest efficiency.

The next step is to determine the propeller load using Equation 2.11. For basic matching it is common to use the propeller law when estimating the propeller load curve. The propeller law describes that the delivered power to the propeller is proportional to the cubic propeller or shaft speed. This law is applied under the assumptions that wake and thrust deduction are constant and the resistance curve follows a cubic relation, see Equation 2.15.

$$R = c \cdot v_s^2 \tag{2.15}$$

Because of the assumptions for the propeller law, the ship curve $K_{T,ship}$ does not change. And as long as the pitch remains constant, the operating point of the propeller does not change. Thus, the propeller load becomes a function of a constant multiplied with the cubic propeller speed, see Equation 2.16.

$$P_{p,load} = 2\pi \cdot K_Q \cdot \rho \cdot D_p^5 \cdot \frac{1}{\eta_R} \cdot n_p^3 = c \cdot n_p^3 \tag{2.16}$$

The last step is to calculate the engine brake power. In this case an example is given with a gearbox, as shown in Figure 2.1. In case no gearbox is present Equation 2.13 can be directly used to find the load of the engine. In case a gearbox is used, the ratio is defined as shown in Equation 2.17.

$$i_{GB} = \frac{n_e}{n_p} \tag{2.17}$$

Where n_e is the engine speed. Now the engine break power is calculated while also taking the gearbox ratio into account by combining Equation 2.13, 2.16 and 2.17, see Equation 2.18.

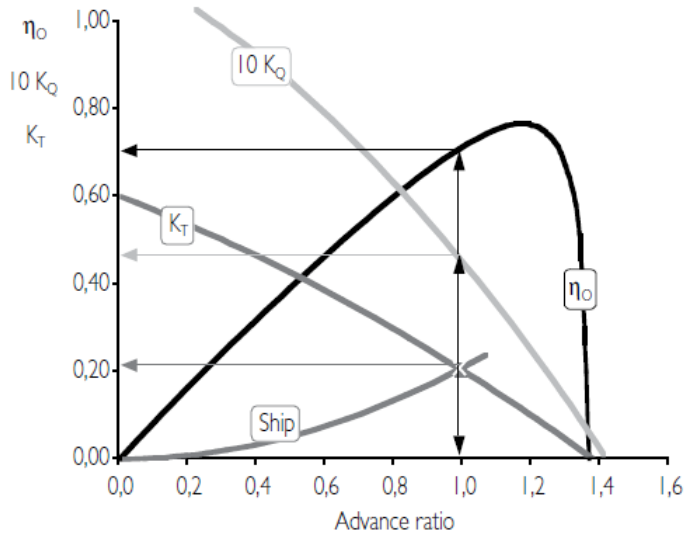


Figure 2.3: Open water propeller curves and ship curve. Taken from Klein Woud and Stapersma [24].

$$P_B = \frac{c}{\eta_{TRM}} \cdot n_p^3 = \frac{c}{\eta_{TRM}} \cdot \left(\frac{n_e}{i_{GB}}\right)^3 \tag{2.18}$$

The propeller load curve can now be shown together with the engine operating envelope to evaluate whether the matching is successful. In Figure 2.4 three propeller loads are shown together with the engine envelope of a turbocharged diesel engine [24].

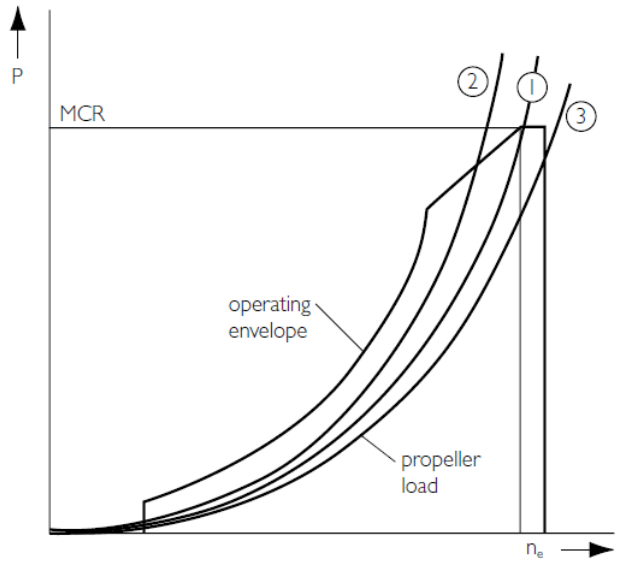


Figure 2.4: Propeller load curves shown in engine operating envelope. Taken from Klein Woud and Stapersma [24].

The propeller load for the design condition is indicated with number 1. In this case the first matching criteria is met because at maximum available power of the engine, the load curve intersects the maximum power limit. Klein Woud and Stapersma [24] mention that this figure clearly shows the benefits of a CPP. If the pitch is too high (Nr. 2), the load curve moves upward and leaves little room between the load and the torque limit of the engine which is limited due to the turbocharger. And when the pitch is too low (Nr. 3), the curve moves downward and intersects with the maximum engine speed line. Now the maximum engine power that can be used by the propeller is limited by the speed limit of the engine.

When it comes to basic matching of propeller and engine in the design point there are some other things to be considered as well, see Figure 2.5 where the propeller load is shown again.

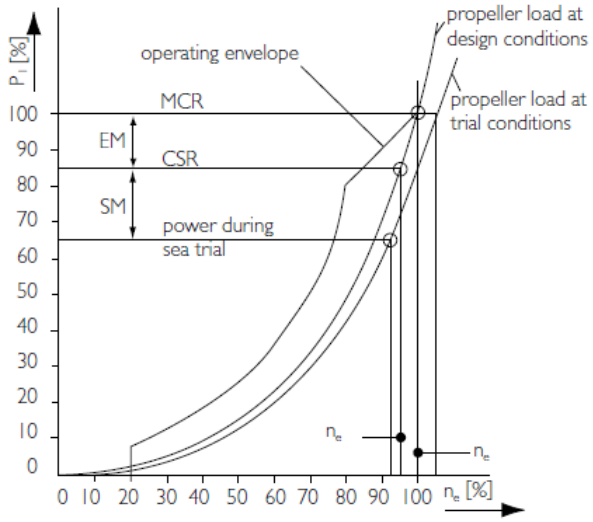


Figure 2.5: Propeller load curves and margins shown in engine operating envelope. Taken from Klein Woud and Stapersma [24].

Besides the propeller load at design conditions, now also the propeller load in trial conditions is shown. The difference between the two curves has to do with the condition of the vessel. When the vessel has its sea trials the hull is clean and usually unloaded, whereas the design condition is calculated based on a resistance that takes into account moderate fouling and design loading [24, 25].

In the figure three operating points have been encircled of which the lowest one is the power during sea trials. The one above that is the design condition which is also referred to as the continuous service rating(CSR). Sea margin(SM) refers to the ratio between the power in service and sea trial condition. The top operating point is called the maximum continuous rating(MCR) at maximum power of the engine. Then, Engine Margin(EM) refers to the ratio between the maximum power and the power in the design condition. This power margin is important because it protects the engine from being overloaded in case the vessel comes into a higher sea state or has increased fouling. It also allows for sailing at a higher vessel speed than the design speed in case of delay during a trip. And it is important with regards to maintenance intervals of the engine [24, 25].

2.2.2. Operation in Off-Design Conditions

The design condition is that of a vessel in its mean service condition. This means full design displacement, two years of fouling and a sea state of 2 or 3 in deep water [25]. All operating conditions that defer from this are denoted off-design conditions. Vrijdag et al. [50] has shown that in case of changes in the ship's state and environmental conditions, one load curve cannot ensure that the criteria for the engine and the propeller are met. Thus, combinator curves for various, likely off-design conditions besides the design condition should be set up. For the purpose of this thesis, we will only get in detail on those off-design conditions that impact the *calculation* for the set up of a combinator curve. It is assumed that for off-design conditions that impact the hull resistance such as a change in draught or a higher sea state, this is taken into account in the resistance data of the vessel. So, when adapting the resistance data, a different combinator curve is set up and can be evaluated individually. The remaining important off-design conditions are the following:

- *Change of pitch*
  Assuming that the vessel speed remains constant in the design point and the wake and thrust deduction as well, the ship curve $K_{T,ship}$ remains constant also. If now the pitch of the propeller changes, the set of OWCs change. The ship curve now intersects with the thrust coefficient curve that belongs to the new pitch setting. Thus, as the pitch changes the operating point of the propeller changes. Hence, the torque coefficient K_Q and the open water efficiency η_O also change.

- *Number of connected engines per shaft*
  In propulsion configurations with more than one engine connected to one propeller shaft via a gearbox, it is possible to either drive the propeller with one engine or to drive the propeller with two or more engines. In Figure 2.6 an example is shown of a configuration with two engines connected to the shaft

via a gearbox.

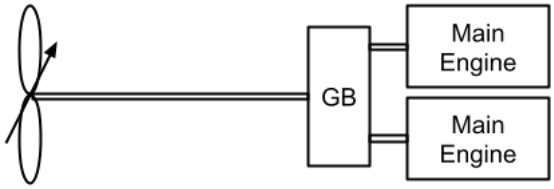


Figure 2.6: Propulsion configuration with two engines connected to one propeller shaft.

The number of engines per shaft is indicated with k_e as shown in Equation 2.19.

$$P_B = \left(\frac{P_p}{\eta_{TRM}} + P_{PTO}\right) \cdot \frac{1}{k_e} \tag{2.19}$$

Parameter PTO will be discussed in the following off-design points. For the configuration in Fig. 2.6 the number of engines $k_e = 2$. If one engine were to be disconnected the number of engines would change to $k_e = 1$. This means that at the same vessel speed, the single engine now has to develop twice the power. In case of a diesel engine with a turbocharger it is likely that the propeller load line will now move outside of the engine envelope. If this is indeed the case, a CPP can be a solution by reducing the pitch and increasing the speed to get the propeller load line back in the operating region of the engine.

Furthermore, if the number of engines reduces, it is likely that the gearbox efficiency decreases due to moving parts that are not driven by all engines.

- *Power Take Off(PTO)*
  The propulsion and auxiliary power plant are designed based on the operational profile of the vessel. Depending on the power demand and the overall efficiency a larger main engine is installed that drives both the propeller and a shaft generator(SG) through a gearbox, see Figure 2.7.

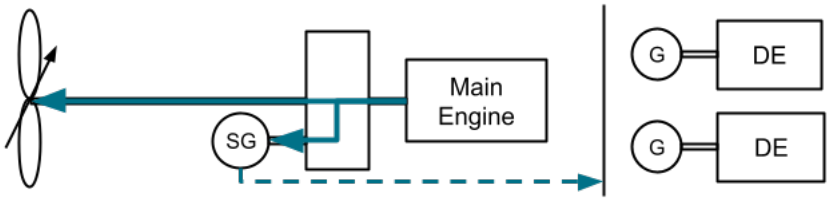


Figure 2.7: Hybrid propulsion configuration in PTO mode.

During transit at the design speed the electricity for the vessel can be supplied to the main network by the main engine through the shaft generator, and the auxiliary gen sets are in stand by. The main engine is often larger than the auxiliary diesel engines and therefore more efficient. Also cost and maintenance are reduced because the main engine uses cheaper fuel and the auxiliary engines have less operating hours [25]. This is not always the most efficient case however. Because the main engine runs at a higher rpm, the engine uses more fuel. Therefore, during transit operators might prefer to operate the main engine in the design condition in order to save fuel while the little hotel load that is required is generated by the auxiliary engines. And PTO is mostly used during manoeuvring mode in port when extra PTO is required together with power supply from the auxiliary gen sets. Next to the power demand for manoeuvring, PTO together with the auxiliary engines are also employed for winch operation and when there are peaks in the hotel load.

If PTO is installed, the propeller load does not change, but as can be seen in Equation 2.19 the PTO is added to the required brake power of the engine. The resulting load curve now shifts upward in the operating envelope and should still lie within the limits. If the vessel comes in any other off-design condition such as a higher sea state, it might be the case that the load curve will get outside of the envelope and measures should be taken to make sure the vessel can still operate without overloading the engine. Either the PTO should be decreased or the pitch should be reduced in case of a propulsion configuration with a CPP.

- *Power Take In(PTI)*
  If an electric motor(EM) is connected through the same gearbox as the main engine, the propeller can now be driven by the electrical motor, the main engine or both. This propulsion configuration is called hybrid drive and makes it possible to have a variety of ways to drive the propulsion system. For example, it could be an option to sail the vessel mainly on the EM for the first lever positions if this is beneficial economically or environmentally. The electric motor could also work in generating mode and in that case it works as the PTO. Another configuration is the situation where both the electric motor and the main engine drive the propeller, see Figure 2.8.

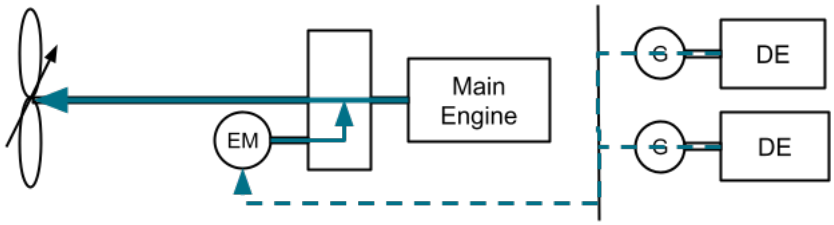


Figure 2.8: Hybrid propulsion configuration in PTI mode.

  This operation mode is often referred to as boost mode because extra power can be delivered if the main engine falls short. The electric motor is driven by the auxiliary engines. This can be beneficial in a heavier sea state or to be able to reach a higher maximum speed than what can be achieved with only the main engine.

- *Number of driven shafts*
  If a vessel has multiple shafts and one or more shafts are trailing or blocked, the required thrust per driven propeller increases at the same vessel speed [25]. This thrust increase is not only due to the reduction of propellers to deliver the thrust, but there is also increased resistance because the propellers are now dragged through the water. As mentioned in the introduction of this section this added resistance will be assumed to be accounted for in the input data before calculation of a combinator curve. The thrust that now has to be delivered by each driven propeller is calculated with Equation 2.2 where k_p defines the number of driven propellers.

2.2.3. Combinator Curve Control

A combinator curve gives the combination of blade pitch and speed setting for a given range of lever positions for the control of CPP's. Because of the ability to vary both propeller pitch and shaft speed, a combinator curve can be optimally designed for a certain operation mode. An example of what a combinator curve might look like is shown in Figure 2.9.

In this figure the pitch and shaft speed settings are shown for forward and astern settings. This simple representation shows the benefits of a CPP because not only can the settings be combined such that the limits of machinery and equipment are not exceeded. The CPP can be operated to sail astern by adjusting the pitch without having to reverse the engine as required for a configuration employing FPP. This ability increases the manoeuvrability in a port or other special operation for a vessel employed with a CPP because less time is required to adjust the pitch and the rpm than to reverse the engine.

Instead of operation in the design pitch for every lever position like with a FPP, the CPP can be operated in combinator mode where the design pitch is held constant as long as the engine and propeller limits are not exceeded. This can result in the presented combinator curve where in Area 1 the pitch is held constant and the engine speed is increased after which both the pitch and engine speed are increased. In Area 2 the engine speed is kept at a constant idle shaft speed and the pitch is increased with ship speed. And lastly, Area 3 shows the astern settings where pitch is decreased at a constant engine speed. In the last astern lever position the pitch cannot be further decreased and the engine speed is increased.

Astern settings are often determined in a similar manner as was shown in the previous combinator curve example. Because astern operation is not a "normal" operation mode, the ship resistance data is not available for astern conditions. Hence, astern settings can not be optimised or calculated in the same way that is done for ahead operation. If the ship resistance data for astern conditions is available however, astern combinator settings can be optimized or calculated as well. For the ahead settings all required data is at hand and therefore the focus in this thesis.

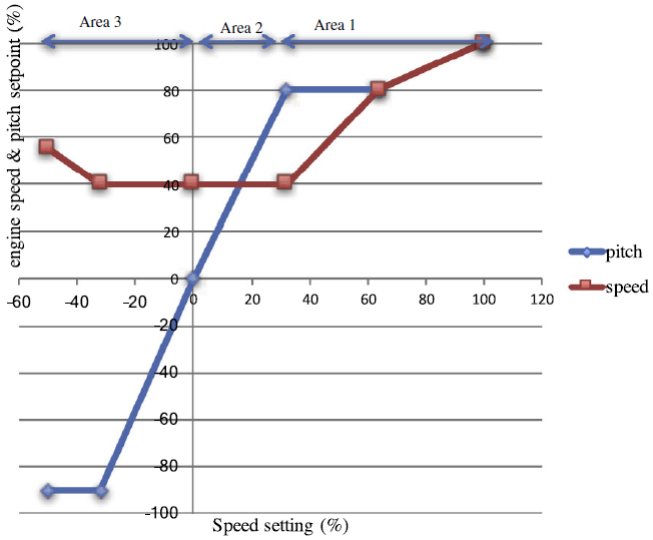


Figure 2.9: Example combinator curve taken from Geertsma [12].

Another important operation mode that proves the beneficial use of a CPP is the constant speed mode. When a vessel operates in the constant speed mode, the shaft speed is kept constant in the maximum speed of the engine while the pitch is varied in order to achieve the required thrust demand. This operation mode is beneficial for vessels with a PTO that is not frequency controlled or if the installed engine is weak.

The operation modes discussed so far are rather straight forward and are often used as a standard procedure. Therefore it is decided to include the option to calculate these modes in the CCG. Finally, combinator curves will be optimised such that the combinator settings are optimal for important performance indicators. The identification of these performance indicators are discussed in the next chapter.

3

Literature Research

In the former chapter, combinator curve theory was explained on the basis of resistance and propulsion and matching theory. To start the development of the CCG, where combinator curves are optimised for certain performance criteria, the research questions of the first objective need to be answered. To this end, a literature research is carried out in this chapter. First, important performance indicators for which a combinator curve can be optimised are identified and quantified. Then, propulsion configuration related constraints are identified. Furthermore, the approach to design an application and optimisation approaches are discussed. Finally, the chapter is concluded by answering the research questions.

3.1. Performance Indicators

An important and exploited advantage of CPPs is the improvement in manoeuvrability of a vessel when it comes to sailing in astern direction and for manoeuvres in a port. Also, with many combinations of shaft speed and propeller pitch, ship speed can be maintained. Vrijdag [49] argues that the CPP has not yet been employed to its full potential in regards to the advantages that come with variable pitch. Another advantage would be to sail at the most fuel efficient combination or to combine the settings such that engine wear is minimal. Next, when it comes to radiated noise there is the option to combine the parameter settings such that minimal or no cavitation is achieved. Thus, there are multiple performance indicators that can be identified for which a combinator curve can be optimised. In this section literature research is performed in order to answer the first research question. This was:

What important performance indicators can be identified for which a combinator curve can be optimised and how can these be quantified?

In order to answer this question, a selection of several performance indicators is chosen from what is found in literature. To start, an overview of the literature discussed in the introduction is shown in the table on Page 19. In the table, the application, optimisation method, objective, approach and the propulsion configuration for each research is shown. Next to such matching tools, there are various operational control strategies for propulsion systems which are elaborated on in great detail by Geertsma et al. [13]. The main control strategies for a propulsion configuration that could be of use to the development of approaches to set up combinator curves are the following:

- Combinator curve control
  This control strategy is described as the determination of a fixed combinator curve which sets the relationship between the speed setting from the lever to the propeller pitch and engine speed. This is the approach as described in Subsection 2.2.3.

- Combinator curve control with pitch reduction
  This control strategy is an extension of the before mentioned strategy. The main objective here is to prevent overloading of the engine by reducing the pitch when the engine supersedes a certain overloading criterion. This overloading criterion is similar to the engine envelope [50]. This strategy pre-

vents overloading of the engine but in operational conditions the impact on propulsion performance, acceleration behaviour [18, 46] and cavitation inception were undesirable [49].

- Effective angle of attack control
  In this control strategy the objective is to reduce cavitation by governing the propeller pitch such, that the risk of cavitation inception is minimal. This is done by using the effective angle of attack which represents the inflow angle of the propeller blade. This control strategy has been tested in sea trials and has proven to reduce cavitation and improves acceleration performance without overloading the engine in operational conditions [49].

Following from the literature, the following selection of important performance indicators appear most frequent and will be dealt with in more detail in terms of their quantification in the next paragraphs:

- Propeller efficiency
- Cavitation
- Fuel consumption
- Emissions

3.1.1. Propeller Efficiency

To start, propeller efficiency is always taken as one of the objectives. In the matching process however, it is always in combination with other objectives, except for the research of Ren and Diao [34] where the main objective is to maximize the propeller efficiency. The propeller efficiency is the ratio of the thrust power delivered by the propeller and the open water power delivered to the propeller as explained in Section 2.1. In order to optimise combinator settings in terms of propeller efficiency, the OWCs for certain pitch settings are required.

3.1.2. Cavitation

Cavitation is an important objective because when cavitation occurs it can cause reduction of delivered propeller thrust. It also causes a lot of noise and structural vibrations. By adapting the pitch of the CPP cavitation can be reduced. Cavitation occurs in different forms and on different places on the propeller blade, the hub and the rudder. The phenomenon of cavitation and cavitation inception are explained in Section 6.2. In this section, several ways in which cavitation inception can be modelled are explained.

A control strategy that is aimed to minimize the risk of cavitation inception in operational conditions was mentioned earlier; the effective angle of attack control proposed by Vrijdag [49]. The effective angle of attack is the angle at which the water flows into the propeller blade at the leading edge of the propeller, as it is expected that inception takes place close to this region. In Figure 3.1 the blade inflow angles are shown for a cambered section of a propeller blade.

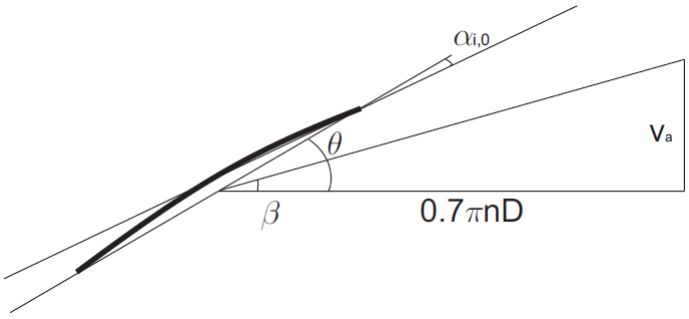


Figure 3.1: Velocity triangle of a cambered section at 0.7R. Taken from Vrijdag et al. [50].

Following from the figure, the effective angle of attack is defined by Equation 3.1.

$$\alpha_{eff} = \theta - \beta - \alpha_{i,0} \tag{3.1}$$

With this method an attempt is made to define a single effective angle of attack that can give the least risk of cavitation inception.

Application	Optimisation Method	Objective	Approach	Propulsion Configuration	Year published	Citation
Investigation of influence EEDI on ship-engine-propeller matching	Genetic Algorithm	EEDI	SFOC; emission factors	low-speed diesel engine, direct driven	2019	[33]
Engine-propeller matching tool	Numerical, using Kt ship	-	-	-	2019	[19]
Optimal matching for F90 frigate	Numerical, using Kt ship	- fuel consumption - cavitation	SFOC; EAR	CODOG	2018	[3, 31]
Approach to perform engine-propeller selection for an LNG carrier under rough weather	Numerical, using Kt ship	- fuel cost	SFOC	dual fuel diesel engine, FPP	2018	[30]
Service performance prediction tool	Numerical, using Kt ship	- fuel efficiency - low emissions	SFOC; emission factors; EAR	main and auxiliary diesel engines, CPP	2015	[1]
Engine-propeller selection tool	Numerical, using Kt ship	-	-	-	2015	[27]
Investigation of effectiveness of GA with PSO operator on ship-engine-propeller matching	Genetic Algorithm based on Particle Swarm Optmisation	- propeller efficiency	open water curves	-	2014	[34]
Investigation of off-design weather and ship state conditions on engine-propeller matching	Artificial Neural Network in real time	- fuel consumption	SFOC	2 main diesel engines, sequential turbochargers, gearboxes, CPP	2014	[26]
Investigation of different objectives and constraints on optimum matching point	Numerical method for matching algorithm: all possible combinations for pitch/rpm for Kt-ship stored in matrix followed by minimisation procedure with weighed function	- fuel consumption - cavitation	SFOC; Panel method for propeller analysis	diesel engine, CPP	2012	[7]
Comparison of GA and PSO for investigation of efficiency and cost optimisation on ship-propeller engine matching	Particle Swarm Optimisation	- cost - cavitation - propeller efficiency	SFOC; EAR; open water curves	-	2010	[28]
Interactive computer program to facilitate diesel engine-propeller matching	Numerical, using Kt ship	-	-	diesel engine, FPP	1981	[5]

To compensate for the reduced pitch that leads to minimal or no cavitation, the engine speed must be increased. Active pitch reduction as described here is also a measure that is taken to prevent engine overloading.

Furthermore, Bob-Manuel and Okim [3] proposed to avoid cavitation by using the Keller formula minimum expanded area ratio (EAR), see Equation 3.2.

$$\left[\frac{A_E}{A_O}\right]_{min} = \frac{(1.3+0.3Z)T}{(p_o - p_v)D^2} + K \tag{3.2}$$

Where Z the number of blades, T, the thrust and K a constant which is 0.0 for fast twin screw ships, 0.1 for twin screw ships and 0.2 for single screw ships [2].

Finally, for the matching algorithm of Coraddu et al. [7] a panel method is used to model the flow field around the propeller blades. At every ship speed and thrust requirement the model is calculated for every pair of P/D ratio and propeller rpm. For each pair, the information about cavitation and efficiency is stored into a matrix. Using a weighed minimized cost function for the cavitation and absolute fuel consumption, the optimal combination of pitch and propeller speed is identified.

Discussion

Coraddu et al. [7] argues that the use of an approach based on the non-dimensional propeller open water coefficients and the propeller speed is inadequate to take into account the effects of cavitation on propeller performance. For this reason a panel method was used that calculates the flow around the propeller blades. This is an important aspect that could be taken into account in further development of the CCG. Such a panel method requires much computational effort. Therefore, a simpler method is required that is at least better than a method solely based on the open water coefficients and propeller speed.

For the method using minimum EAR only the geometry in terms of the blade area ratio for different pitch settings is required and therefore this method is expected to be insufficient to predict cavitation behaviour.

In this respect, the effective angle of attack is the better choice, because for this method a prediction of the inflow angle of the propeller blade using its geometry data and the propeller speed is taken into account. Using a certain correction factor, the risk of propeller cavitation is expected to be minimal for a large range of pitch settings. The effective angle of attack must however be calibrated with full scale cavitation measurements. Unfortunately full scale cavitation measurements were not an option during the development of the CCG. There exists however in-house software that predicts cavitation inception behaviour for a number of shaft speed and propeller pitch settings such that the optimised settings can be evaluated.

3.1.3. Fuel Consumption

The performance of the engine is expressed in terms of overall or effective efficiency, and is defined as the work output W_e to the heat input Q_f, see Equation 3.3 [25].

$$\eta_e = \frac{W_e}{Q_f} = \frac{W_e}{m_f \cdot h^L} \tag{3.3}$$

With m_f[kg] the fuel injected per cycle and h^L[J/kg] the nominal lower heating value of the used fuel. Therefore fuel consumption is directly related to the efficiency, operational cost and emissions of the engine. A measure for fuel consumption is the specific fuel consumption sfc. This parameter relates the fuel consumption of the engine to brake power as shown in Equation 3.4 [25].

$$sfc = \frac{\dot{m}_f}{P_B} \tag{3.4}$$

The fuel consumption can also be further expressed in terms of fuel consumption per mile and is presented as a function of the ships speed as shown in Equation 3.5 [25].

$$fcm = \frac{\dot{m}_f}{v_s} = \frac{sfc \cdot P_B}{v_s} \tag{3.5}$$

Specific fuel consumption is different for different engine types and is not constant over the power range, but steeply increases with low power. This is due to idle fuel flow in the lower power range [25]. In Figure 3.2 examples of specific fuel consumption in terms of the load are shown for different engine types [25].

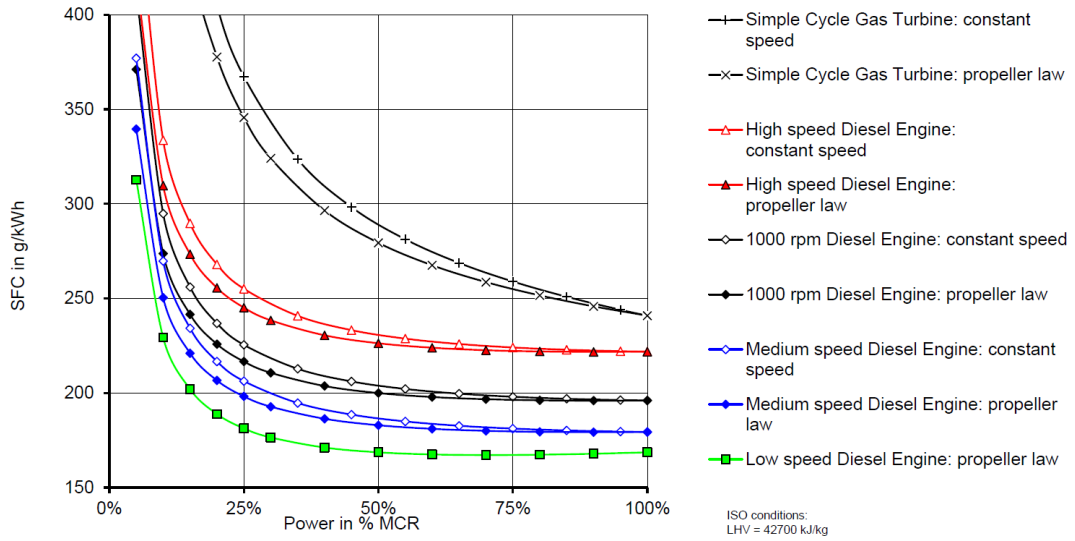


Figure 3.2: Specific fuel consumption of prime movers. Taken from Klein Woud and Stapersma [25].

Figure 3.3 shows an engine envelop and the specific fuel consumption contours. The fuel consumption increases below 70% of the engine top speed which is approximately at 50% of the power. The fuel consumption is optimal between 80% and 100% of the available power.

Altosole et al. [1], Coraddu et al. [7] and Bob-Manuel and Okim [3] developed a tool where at least one of the objectives is to optimise for fuel efficiency by using the engine fuel map containing the specific fuel consumption contours as shown in Figure 3.3. The representation was in the form of fuel consumption in [%] or $[kg/h]$ versus the pitch, see Figures 3.4 and 3.5.

It must be noted that while the power specific fuel consumption is determined by engine efficiency, the ship speed is most important when it comes to determining ton-mile fuel consumption [38]. In an internal study by Tillack [43] it was found that by adapting the shaft speed it is possible to increase the efficiency of the complete system. When the shaft speed at different lever positions is decreased, the pitch must be increased to maintain the ship speed. This increases the propeller efficiency and also the engine efficiency because at higher power rating the specific fuel consumption is lower. However the overall fuel consumption might increase and thus also the amount of exhaust gas. Further, when the individual shaft speeds at the lever positions are increased, the specific fuel consumption is increased, but the total fuel consumption decreases. The result of this study is that in order to sail at the most fuel efficient condition in terms of overall fuel consumption, the speed settings should be selected such that over the widest possible speed range, with equidistant ship speed steps, the propeller should be driven as a FPP. This means that the propeller load should approximate the cubic power curve $P(n) = n^3$. The latter was also concluded in the research of Geertsma [12] for a control strategy to minimise fuel consumption.

Discussion

When it comes to fuel consumption it was noted that while the power specific fuel consumption is determined by engine efficiency, the ship speed is most important when it comes to determining ton-mile specific fuel consumption [38]. Two studies concluded that in order to run the vessel at most fuel efficient mode in terms of total fuel consumption the propeller should be driven according to the cubic propeller load curve. Even though this might be the case for most engines, it is still engine specific and if the fuel map for an engine is known, there is the possibility to design the combinator curve for the objective to minimize ton-mile fuel consumption by means of an algorithm that finds the threshold for when either the specific fuel consumption should be minimized or the engine power for a certain ship speed.

3.1.4. Emissions

Emissions are directly linked to the combustion of fossil fuel and increase with specific fuel consumption. The level of pollutant emission minimisation is also seen as a performance indicator. During combustion

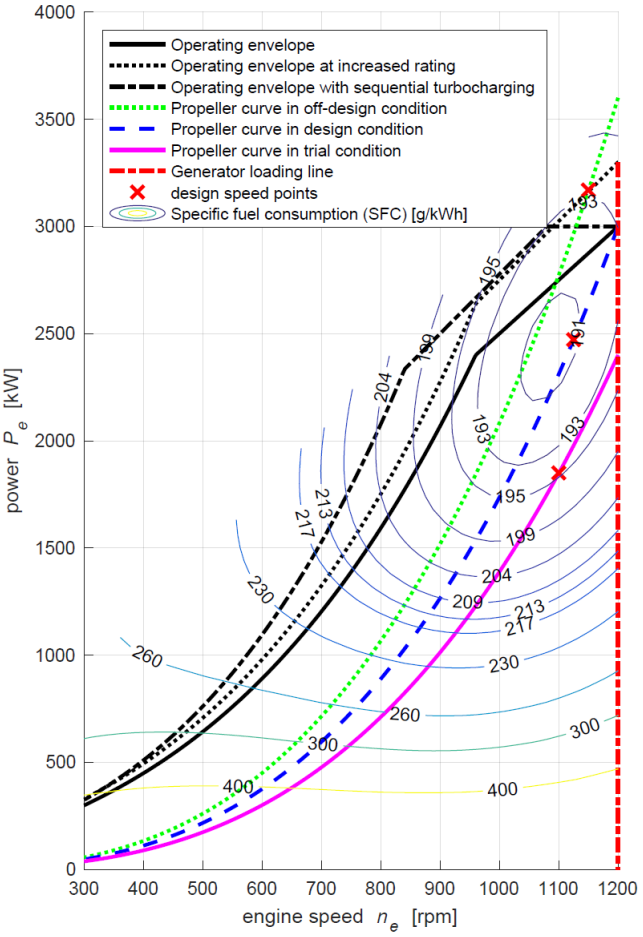


Figure 3.3: Three propeller curves and the generator loading line in three diesel engine operating envelopes with typical SFC contour plot. Taken from Geertsma et al. [13].

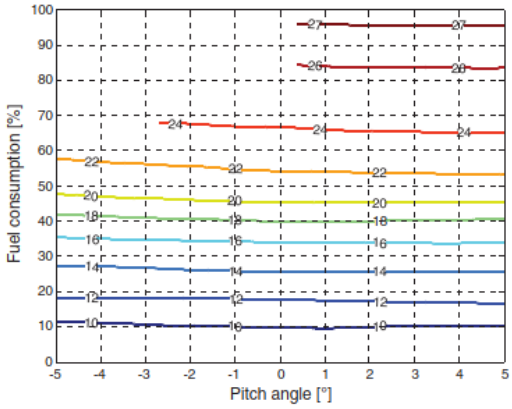


Figure 3.4: Fuel consumption in [%] versus pitch. Taken from Coraddu et al. [7].

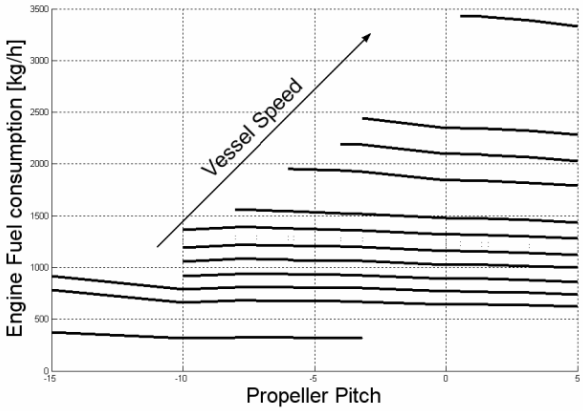


Figure 3.5: Fuel consumption in $[kg/h]$ versus pitch. Taken from Altosole et al. [1].

both non-pollutant and pollutant emissions are formed. There is a difference between fuel related emissions from complete combustion and cylinder process related emissions caused by incomplete combustion. Then there are nitrogen oxides which are categorised as unintended combustion of inert nitrogen in the air [42]. While for instance the amount of carbon dioxide (CO_2) would decrease at higher engine efficiency due to lower specific fuel consumption, nitrogen oxides would increase due to higher combustion temperatures.

This already shows that optimising one or the other might have different impact on the matching point. A parameter to quantify emissions is the specific pollutant emission spe. This parameter relates the mass flow of the pollutant emission to the brake power and is defined as shown in Equation 3.6 [25].

$$spe = \frac{\dot{m}_{pe}}{P_B} \tag{3.6}$$

Another way in which the emissions can be defined is the pollutant emission as a result of combustion. Then the pollutant emission ratio per is the ratio between the pollutant emission and the fuel consumption as shown in Equation 3.7 [25].

$$per = \frac{\dot{m}_{pe}}{\dot{m}_f} \tag{3.7}$$

In order to judge emissions from an environmental point of view the specific pollutant emission spe is a better candidate because it relates the pollutant to an ultimate useful product, that is the brake power of the engine. In Table 3.1 an overview of the emissions for a diesel engine are shown in terms of the parameters spe and per.

	pollutant emission ratio (per) in g/kg	*specific pollutant emission (spe) in g/kWh*
CO_2(85 % C in fuel)	3200	500-700
SO_x per % S in fuel	20	3.2-4.4
NO_x	40-100	6-22
HC(gaseous)	0.5-4	0.1-0.9
CO	2-20	0.3-4.4
Particulates (depending on fuel)	0.5-2	0.1-0.4

Table 3.1: Order of magnitude of diesel engine exhaust emissions. The spe has been determined using an sfc between 160-220 [g/kWh]. Taken from Klein Woud and Stapersma [25].

Trozzi [44] reported on emission estimate methodologies and if the fuel consumption is known the emissions can be computed with fuel related emission factors for the navigation phases cruise, hotelling and manoeuvring. In case fuel consumptions are not known a methodology is proposed to compute the emissions based on installed power.

Next to fuel consumption Altosole et al. [1] also developed the matching tool to estimate the pollutant gasses. Due to a lack of data about exhaust emissions from ships the output of the model for this objective has not been validated. In order to calculate the emissions the engine delivered power has been taken as the main parameter. It was assumed that engines in the same speed range(low, medium, high) have similar average specific emissions. Using specific emission factors the emissions have been calculated on the bases of the part load and the specific fuel consumption.

A mandatory index that is aimed at reducing CO_2 emissions is the EEDI. This index indicates the potential CO_2 emission with respect to the ships capacity to transport useful weight. The regulations are becoming more strict with time and ships built after 2013 have to meet the requirements. This index indicates the potential CO_2 emission with respect to the ships capacity to transport useful weight [33]. The calculation for this index is given in Equation 3.8. The power that is included in the calculation is shown in Figure 3.6.

$$EEDI = \frac{(\prod_{j=1}^{M} f_i)(\sum_{i=1}^{nME} P_{ME(i)} \cdot C_{FME(i)} \cdot SFC_{ME(i)}) + (P_{AE} \cdot C_{FAE} \cdot SFC_{AE}))}{f_w \cdot f_l \cdot f_i \cdot Capacity \cdot V_{ref} \cdot f_w}$$
$$+ \frac{((\prod_{j=1}^{M} f_i \cdot \sum_{i=1}^{nPTI} P_{PTI(i)} - \sum_{i=1}^{neff} f_{eff(i)} \cdot P_{AEeff(i)}) \cdot C_{FAE} \cdot SFC_{AE}) - (\sum_{i=1}^{neff} f_{eff(i)} \cdot P_{eff(i)} \cdot C_{FME} \cdot SFC_{ME})}{f_w \cdot f_l \cdot f_i \cdot Capacity \cdot V_{ref} \cdot f_w} \tag{3.8}$$

In this calculation P is the power of the specific engine at 75% MCR, C is the CO_2 emission factor, SFC the specific fuel consumption and f a non-dimensional factor to account for some specific conditions, V the ship speed at design condition and capacity the dead weight tonnage. After investigation of the EEDI on

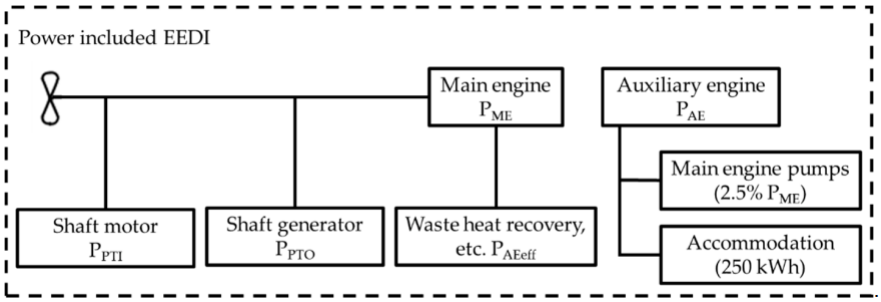


Figure 3.6: Power included in the calculation of EEDI. Taken from Ren et al. [33].

ship-propeller-engine matching it was found that the EEDI can be improved by decreasing the ship speed, optimising the hull and enlarging the propeller. The research was mainly based on low speed direct driven diesel engines without any gearbox ratio. It is recommended to research the influence of various propulsion configurations on the EEDI and to also investigate the influence of other emission regulations.

Discussion

Currently, only one recent research study was found in which a matching problem was optimised for a harmful emission, in this case the EEDI for CO_2 [33]. One of the recommendations of this study is to also research the influence of other emission regulations on the matching problem. Each regulation for harmful emissions is in fact one performance indicator for which a combinator curve could be optimised. Because there is limited research for optimisation of the matching problem for harmful emissions it is chosen to leave harmful emissions out of the scope of this thesis. Up till now harmful emissions that can be estimated based on the matching problem have mostly been estimated by using fuel related emission factors. The focus of this thesis is mainly on developing and implementing approaches to optimise combinator curves for important performance indicators, therefore the estimation of emissions is left out of scope.

3.2. Propulsion Configuration Related Constraints

The combinator curve is primarily constrained by the power, torque and idle/maximum speed limits of the propulsion plant and the minimum and maximum propeller pitch. Propulsion configurations for modern ships can be categorised in mechanical, electric or hybrid propulsion. The power is generated or supplied with the use of combustion engines, fuel cells, energy storage or a hybrid combination of these. An elaborate review of current and future power and propulsion configurations together with their control strategies can be found in the paper from Geertsma et al. [13]. In this section we aim to answer the second question:

What propulsion configuration related constraints can be identified?

The primary sources for the following paragraphs are from Klein Woud and Stapersma [25] and Geertsma [12], any other literature is mentioned specifically.

3.2.1. Mechanically driven propulsion

The source of power for ships has changed throughout recent history. Mechanical propulsion started in the 19th century with the development of the steam engine. An example of a typical mechanical propulsion configuration is shown in Figure 3.7. A power source, which is also referred to as a prime mover (1), drives the propulsor (3). The type of propulsor considered in this thesis is a controllable pitch propeller and can be direct or gearbox (2) driven. Next there is a power plant to supply power to the systems on board of the ship. This plant consists of a separate electrical AC network (6) to generate and distribute electric power to the drives (4) and auxiliary loads (5) and its power is supplied by generators (7). Today the primary sources of power are combustion engines, then for some applications gas turbines, steam turbines and nuclear reactors are employed [8].

Diesel engines can further be categorised in low, medium and high speed engines. Low speed diesel engines can be direct or geared driven, medium and high speed diesel engines require a gearbox between the prime mover and the propulsor. The gearbox ratio was defined in Subsection 2.2.1, see Equation 2.17.

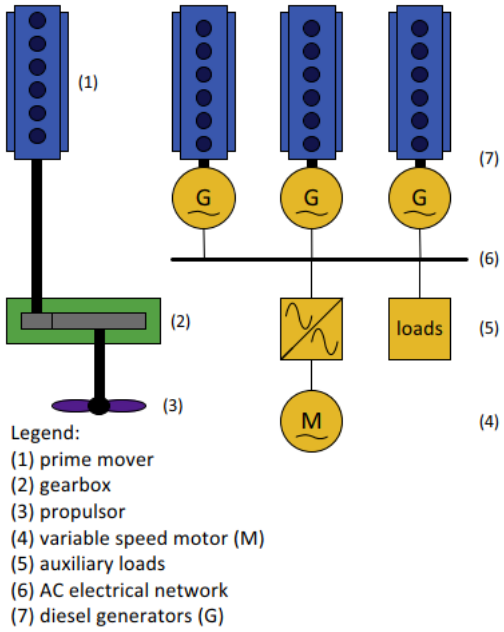


Figure 3.7: Typical mechanical propulsion system. Taken from Geertsma et al. [13]

The gearbox ratio $i = 1$ for direct drive and $i > 1$ for geared drive. Constraints related to a change of propeller pitch, number of connected engines per shaft, number of driven shafts, power take off and power take in have been explained in Subsection 2.2.2.

Finally, an important constraint is the engine overloading criteria. Together with the minimum and maximum shaft speed, the engine overloading criteria is defined by the operating region of the engine in terms of engine revolutions and fuel rack position [46]. In order to prevent engine overloading Vrijdag et al. [50] used the maximum engine speed dependent fuel rack setting which is called the Reduced Time Between Overhaul (RTBO). The engine margin is defined as the fuel rack X in mm that can be added to the current fuel rack position at that shaft speed until the RTBO-line is crossed, see Equation 3.9.

$$margin = X_{RTBO} - X \tag{3.9}$$

These RTBO curves are engine specific and they can be found in project guides from the manufacturer. In case this information is not readily available using an engine margin relative to the power limits of the engine envelope is generally a good indicator for the capacity of the engine to respond to changes in waves and wind. Figure 3.8 shows the RTBO lines together with the maximum and minimum speed lines of the engine and some combinator curves.

This figure shows that combinator 1, which is the combinator curve suitable for calm water and 6 months out of dock, is not suitable for the condition for sea state 6 and 6 months out of dock, while combinator curve 2 is satisfactory for this off-design condition. As mentioned earlier it is thus important to also design the combinator curves for off-design conditions. Especially in higher sea states vessels can suffer from dynamic overloading. Figure 3.9 shows measurements on a *Karel Doorman* class frigate sailing in head waves in Sea State 6 to show how the engine is overloaded in this condition.

For a specific moment in time the fixed combinator curve cannot ensure that engine overloading is prevented and dynamic control of the propeller pitch becomes important [12]. As the fixed combinator curve determines the stationary working points of the machinery, engine overloading can be partly overcome by having at least multiple combinator curves for different off-design conditions [50].

3.2.2. Electrically driven propulsion

Electrical propulsion configurations are becoming more attractive as operating profiles become more diverse. Examples of vessels that employ electrical propulsion systems are cruise ships, ferries, DP-vessels, cable laying vessels, research ships, ice breakers, vessels for offshore applications and naval vessels [35]. The main reason that electrical propulsion systems have become popular for cruise ships especially is because the ho-

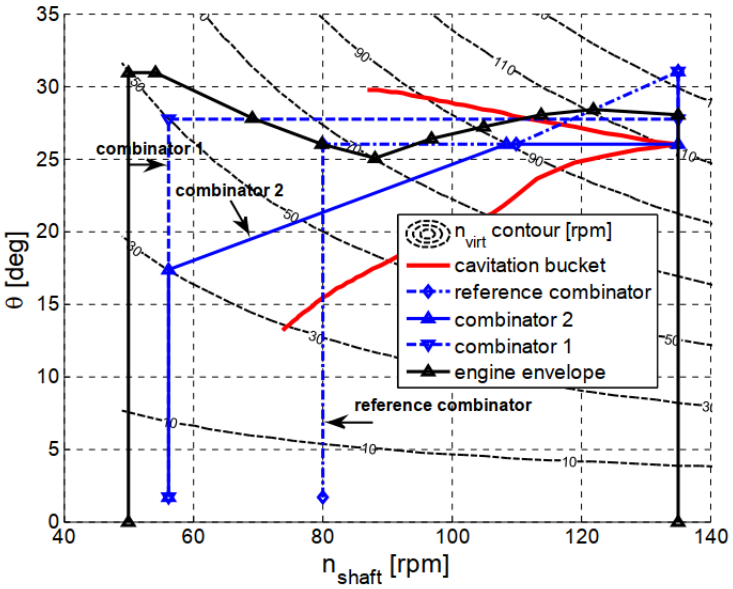


Figure 3.8: The engine envelope (RTBO-line) in the shaft speed-pitch plane. This figure holds for seastate 6, 6 months out of dock. Taken from Vrijdag et al. [50].

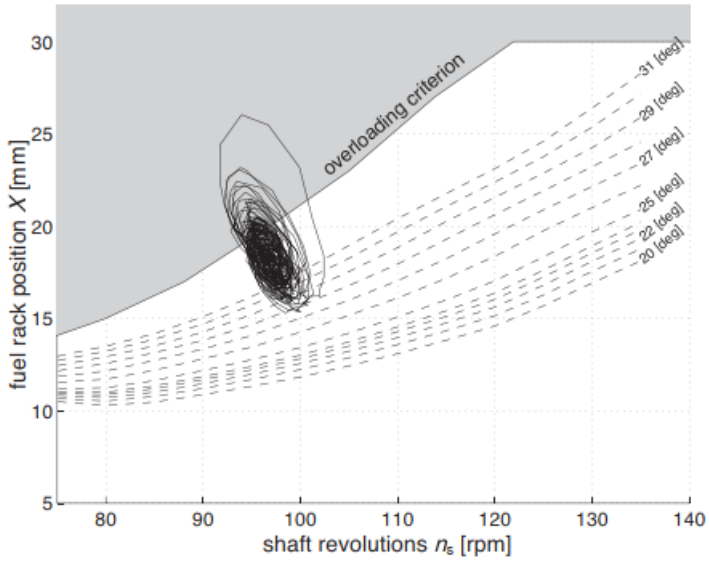


Figure 3.9: Engine speed and fuel rack measurements of diesel direct propulsion in Sea State 6 with head waves ©IFAC 2001. Taken from Van Spronsen and Tousain [46].

tel load is a significant part of the required power supply, and when that is the case electrical propulsion becomes fuel efficient [48]. Another reason why electrical propulsion is attractive is because the space for the engine room is no longer restricted by the main shaft line. The propulsion plant layout can be done in such a way that is optimal for the operating profiles of the vessel. In Figure 3.10 an example of a typical electric configuration is shown.

Several diesel generators (1) feed a fixed frequency high voltage electrical bus (2) which feeds the motor drives (5,6) and the hotel load (6), often through a transformer (3). The electric drives consist of a power electronic converter (4) and is used to control the shaft line speed [13]. In electric configurations a power management system is used to match the running engines to the combined propulsion load and hotel load. Having several generator sets running parallel increases the availability because redundancy is increased.

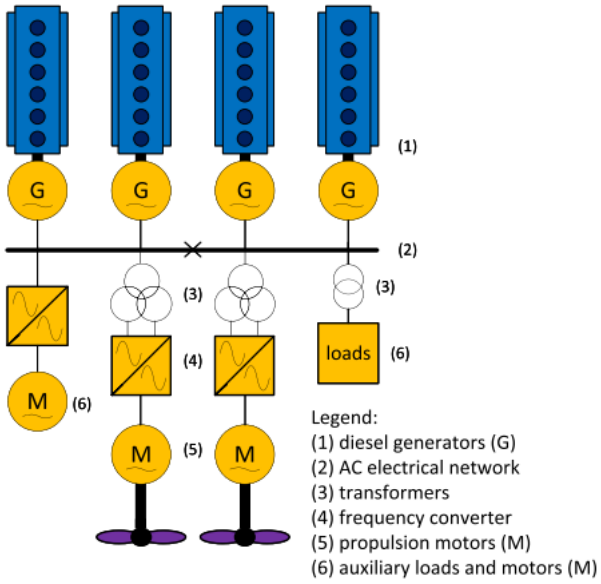


Figure 3.10: Typical electrical propulsion system. Taken from Geertsma et al. [13].

Also, because the diesel generators run at rated speed the NO_x emissions are likely to be lower. On the other hand for propulsion availability for DP vessels the engines must run at low part load which decreases fuel and emission efficiency.

Control strategies for the electric propulsion configuration in the research of Geertsma [12] is aimed at voltage and frequency control and propulsion control. The latter is done by controlling the motor torque and flux by controlling the signals of the pulse with modulator(PWM). Electric motors can provide full torque at any speed and when designing a combinator curve the power limit and the efficiency of the motor must be taken into account. Typical losses from diesel engine mechanical output to electric motor to shaft power are shown in Figure 3.11. And in Figure 3.12 an example is shown of a diagram showing an efficiency map over the power range and shaft speed together with a propeller load curve.

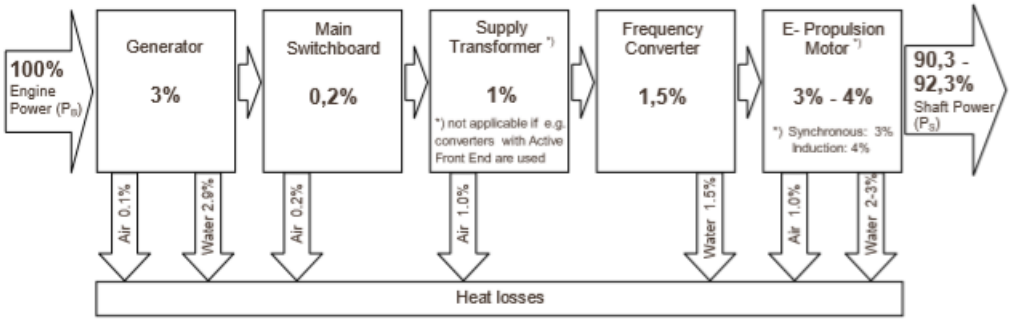


Figure 3.11: Typical losses in a diesel-electric propulsion plant. Taken from Turbo [45].

3.2.3. Hybrid driven propulsion

Hybrid propulsion configurations are a combination of electrical and mechanical propulsion. In case the auxiliary load is only a fraction of the required propulsive power, hybrid configurations become beneficial instead of electrical propulsion configurations. This is due to losses in electrical conversion that lead to increased fuel consumption. Vessels that employ such systems are naval vessels, towing vessels and offshore vessels. A typical hybrid configuration is shown in Figure 3.13.

The propulsor is driven by a mechanical engine (1) or an electrical motor (2), the engines are coupled to the same shaft by a gearbox (3). Generators (G) feed the electrical network (4) [13]. For the lower speed range the electrical motor drives the propulsor and for the higher speed range the mechanical engine drives

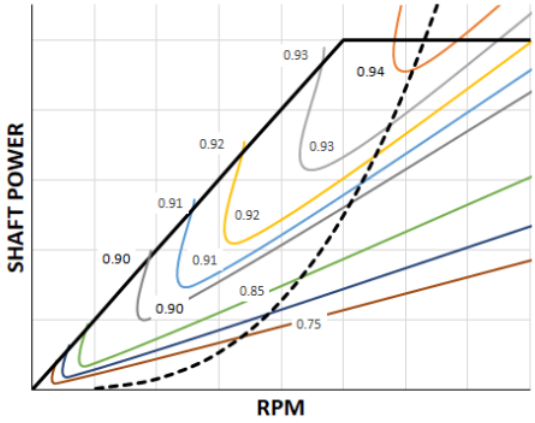


Figure 3.12: Example of an electric motor power vs RPM curve and the efficiency map. Taken from MacPherson [29].

the propulsor. When the mechanical engine is running, the electrical motor can be used as a generator for the electrical loads. Hybrid propulsion configurations and their propulsion configuration related constraints have been partly discussed in Subsection 2.2.2

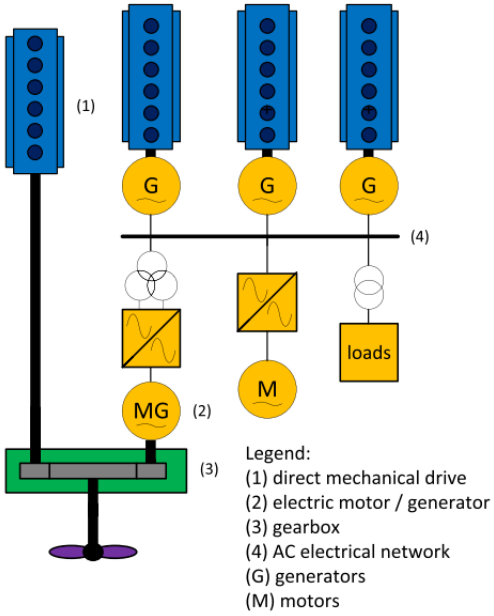


Figure 3.13: Typical hybrid propulsion system. Taken from Geertsma et al. [13].

Discussion

The optimisation of combinator curves is constrained by the configuration and the limits of the sub systems. At this stage of the CCG development the main goal is to be able to design and optimise a combinator curve for a certain performance indicator while constrained by the propulsion configuration. However, the variety of propulsion configurations is vast. Therefore it is chosen to lay the foundation for a CCG in which various propulsion configurations can be taken into account. For this thesis the scope of propulsion configurations that are taken into account are those introduced in Subsection 2.2.2.

There exist more complex configurations such as a father-son configuration where the connected engines are different in size. Or a hybrid configuration with varying load division. Due to the complexity that this brings it is chosen to leave such configurations out of scope. This means that the number of engines connected to a shaft should be identical in size and type, and the PTO or PTI load is considered constant for all lever positions.

Furthermore, fuel consumption is dependent on the engines that drive the propeller and the engines of the auxiliary power plant. The driving engine can be a combustion engine, a gas turbine or an electric motor. In case of the first two engine types there is not much difference in terms of implementation because the form in which data about the operating envelope and SFOC is inserted can be similar.

The electric engine however can be supplied with power from various sources such as combustion engines, batteries, fuel cells or a combination of these. The scope of this thesis is mainly focused on approaches to optimise combinator curves for certain performance indicators and therefore limited to configurations for which the data format of the operating envelope and SFOC map is similar. In case of PTI an exception is made namely, the source from which power is supplied to the PTI is not taken into account. Instead a default efficiency is assumed.

3.3. Generator Design Approach

As soon as the performance indicators and constraints are defined, these can be translated into algorithms and used to optimise and calculate combinator curves. In order to be able to clearly visualise and gain insight about the combinator curve design the CCG must be developed. The approach that is proposed to develop this generator is discussed in this chapter. The data model design will be proposed from the perspective of ship-propulsion modelling and from the perspective of software design in terms of development and implementation. Finally a discussion is done on the approach to optimise for one or more performance indicator such that the last question is answered:

What approach should be taken to optimise a combinator curve for a certain performance indicator?

3.3.1. Ship Propulsion Model

Vrijdag et al. [51] proposes a systematic approach towards modelling, verification, calibration and validation of a ship propulsion simulation model. At this stage only modelling will be discussed. Vrijdag et al. [51] describes that the goal of a ship propulsion simulation model is *"... to represent the reality accurate enough to make it useful for the development of a propulsion controller. The model should also give accurate enough output to enable judgement of diesel engine loading, propeller behaviour and straight line manoeuvring characteristics."* Modelling is further explained by using an example and starting with a conceptual model, see Figure 3.14.

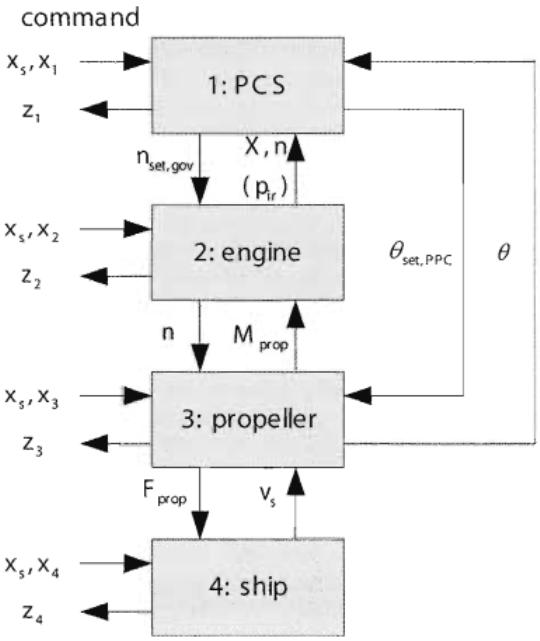


Figure 3.14: Schematics of a conceptual model. Taken from Vrijdag et al. [51].

Propulsion Control System (PCS)
Using this example and applying it to this thesis, the PCS-block represents a combinator curve calculator that determines the stationary behaviour of the power plant. The result of this calculation is a combinator curve that gives a pitch and propeller speed setting for each lever position for a certain performance indicator. All relations of the ship-engine-propeller for the matching process, the performance indicator and the constraints that have been discussed in the former sections will be brought together here.

Engine-Model
The engine in this case, is modelled by means of a look-up table containing the power, torque and speed characteristics of the engine. If, at a later stage of the CCG an electric motor is used, also the efficiency, either in the design point or in the form of an efficiency map will have to be part of the engine data. In this conceptual example the gearbox is included by its gearbox ratio and a constant transmission efficiency.

When looking at the fuel consumption it was discussed that general fuel maps for low, medium and high speed engines could be used to estimate the fuel consumption and emissions in case such information is not known. Diesel engine models exist with which one can generate look-up tables for the fuel map. Diesel engines can be modelled on different levels which have to do with the complexity and the dynamics that are taken into account [14, 40].

The areas of analytical models are Computational Fluid Dynamics(CFD), phenomenological multi zone models, crank angle models, mean value models and transfer function models [36]. The first three models that are mentioned here are used to provide detailed information of the internal processes of the cylinder. These can be used for research and development of diesel engines. When the diesel engine is evaluated as a sub-system of a propulsion train the processes within the diesel engine are often not of interest, but more so the overall engine parameters [36, 40]. For this purpose the latter models can be used where the engine is simulated on a cycle time scale. Geertsma et al. [15] reviewed various diesel engine models. The zero- and first-order are the most basic two that can be used in a larger system in order to evaluate system performance. For such models it is possible to use mathematical equations that can be derived from measurement points or it can be based on look-up tables from experimental data from which can be interpolated and/or extrapolated [49].

An example of a such a model is the "MOSSEL" Model [41]. With this model the fuel consumption and emissions are estimated on the bases of power and engine rotational speed for diesel engines and gas turbines. The reason for the name of this model is due to the fact that the elliptical shapes of specific fuel consumption remind of the form of mussels. Because both engine performance indicators, efficiency and fuel consumption, are dependent on the specific fuel consumption, an SFOC map of the engine is required. Often this information is not readily available. Therefore it is aimed to facilitate the ability to optimise and evaluate the combinator curve for fuel consumption on the bases of a general SFOC map for diesel engines, as these remain the majority of main engines. This general SFOC map for diesel engines will be extracted from this zero order engine model, the "MOSSEL" model.

Propeller Sub-Model
The propeller model in the example of the paper includes a propeller pitch controller, a pitch actuating system, the open water diagram of the propeller and a wake field model. For the case of a CCG however, the propeller model consists of a look-up table with the open water diagram of the propeller and its geometry data. Requirements for a propeller model is that it has to at least deliver the thrust, torque and pitch. Next to this, for the CCG also information on efficiency and cavitation should be delivered by the propeller model.

Ship Sub-Model
The ship model contains a look-up table with the resistance curve and according to the paper it has to deliver at least the ship speed.

So far, the example using the conceptual model. This approach will be used and extended further in order to design an appropriate ship-propulsion data model for the CCG.

3.3.2. Software Design
In order to create a versatile application the CCG will be developed with Microsoft Visual Studio and C# programming language. An important aspect of developing software is to first get a clear description and understanding of what the software should be able to do by defining the design goal. Then, a main structure and its substructures can be developed and visualized in the form of hierarchy or flow diagrams. An example of

this could look like the conceptual model in Figure 3.14. These diagrams can then be put in more detail by writing pseudo code before the actual programming begins.

The approach to programming will be based on Object Oriented Programming(OOP) [20]. So called "Objects" can contain data and methods. For instance in the case of the ship-model it contains the object ship which has the resistance curve data for a certain vessel and by means of an interpolation method resistance and ship speed of this ship can be delivered by the model. By separating the sub-models, their data and methods in this way the software can easily be updated or adjusted without affecting other objects.

Finally, when the calculation core is programmed a Graphical User Interface(GUI) has to be developed. For this, OOP will be used as well.

3.3.3. Optimisation Methods

Several optimisation methods can be used to optimise the combinator curve for one or more performance indicator. The most straight forward method where well known mathematical relations are used and calculated numerically is used in most of the publications shown in the table on Page 19. Next, there have been several publications where multi-objective optimisation methods are used.

Sobey et al. [39] pleas for more GA approaches for maritime applications but also argues that there is no free lunch when it comes to developing an algorithm to solve a problem. Either an algorithm is focused to solve a detailed problem which performs less with other problems, or an algorithm is focused to solve for a range of problems that might never be able to reach the performance prediction of the more specialized solvers. The success in optimisation is to use the correct algorithm for the given problem. Multi-objective optimisation methods can be especially beneficial for holistic ship design for instance. Ship design is complex because of the many possible constraints and optimisation criteria [32]. For holistic ship design GA approaches can be used to explore the design space and to gain insight in how these requirements, constraints, design solutions and performance characteristics relate [10]. The amount of resulting alternative design solutions in the design space can become very large. Also, these algorithms have to be trained to find the optimal solution which requires a lot of computational effort. The question arises whether such intensive algorithms are the right choice if simpler models also achieve satisfactory results for the goal of the solver, that is according to the goal of a ship-propulsion-model mentioned in the former section. The latter will have to be validated for each of the performance indicator during development of the CCG.

The proposed optimisation approach for this thesis is to first design and optimise the combinator curve for each performance indicator individually, such that the optimum value at each lever position is known for each performance indicator. This can be used to indicate the relative impact on the matching problem. After this, a separate method can be used to give a final weighed optimisation that can be chosen by the user.

3.4. Conclusion

In this chapter we aimed to answer the research questions belonging to the first objective posed in the introduction. The following conclusions are made regarding each of these:

- *What important performance indicators can be identified for which a combinator curve can be optimised and how can these be quantified?*

  The following important performance indicators for which a combinator curve can be optimised have been identified together with a way to quantify them.

  - **Propeller efficiency**
    Propeller efficiency is always considered to some extent when solving a matching problem. It is defined and quantified by taking the ratio of the thrust power that is delivered by the propeller and the open water power that is delivered to the propeller. The propeller efficiency is also referred to as the open water efficiency.

  - **Cavitation**
    The occurrence of cavitation can have detrimental impact on performance and equipment. This damage can present itself in the form of possible thrust loss, erosion, vibrations and noise. It is therefore important to be able to indicate whether it is likely that cavitation will occur for a certain combination of propeller pitch and shaft speed at a certain lever position.

Several approaches have been evaluated with which a combinator curve can be optimised in order to limit the risk of cavitation inception. From these approaches it was concluded that the most promising approach to find a combinator setting for which the risk of cavitation inception is minimal, is the *effective angle of attack* method developed by Vrijdag [49].

  - **Engine efficiency and fuel consumption**
    An important performance indicator that is directly related to engine performance is fuel consumption. A characteristic of a combustion engine that is used to determine fuel consumption is the specific fuel consumption. A choice is made to separate two performance indicators based on the fuel consumption, namely engine efficiency and fuel consumption per unit of time. This distinction is made in order to be able to evaluate either engine efficiency or fuel consumption related to ship speed and the operational profile of the vessel.

- *What propulsion configuration related constraints can be identified?*
  The propulsion configuration of a vessel is the specific arrangement of propellers, engines, gearboxes and other sub systems that belong to the propulsion system. These can be categorised based on the engine that drives the propeller which can be mechanically, electrically or hybrid; a combination of both. The power supply is often generated with diesel generators. This power could also be supplied with batteries, fuel cells or an alternative power supply. When the operating envelope and SFOC map of any driving engine is known, this data can be used for the combinator curve design. The possibility to take this data into account is limited to diesel engines, dual fuel engines and gas engines. If the operating envelope of an electric driving engine is known, still combinator curves can be designed, but not optimised and evaluated in terms of engine related performance indicators.

  The propulsion configuration related constraints that will be taken into account are the following:

  - **Number of propellers**
  - **Number of driving engines per propeller**
    Identical engines.
  - **Power take off**
    Constant load for each lever position.
  - **Power take in**
    Using a default efficiency and constant load for each lever position.

- *What approach should be taken to optimise a combinator curve for a certain performance indicator?*
  The question of what optimisation method should be used depends on the purpose of the CCG and the complexity of the optimisation problem. In the studies mentioned in the literature review both algorithms containing mathematical relations and interpolation methods as well as complex multi-objective optimisation methods were used. Because it is chosen to optimise a combinator curve for one performance indicator at a time, it is sufficient to use an algorithm containing mathematical relations and interpolation methods. Thus, in this thesis a numerical approach is taken to optimise the combinator curves for a certain performance indicator.

4

Combinator Curve Generator Design

In the former chapter all described theory and calculations are the foundation with which system data will be processed to set up any combinator curve for a given operation mode. The next step in the development of the CCG is to define structures for the data and work flow. These structures serve as the basis for the programming approach of the application. According to the data structure, information about any ship and its propulsion system can be used and stored. The work flow structure gives a basic understanding of the expected interaction between the user and the application. In this chapter, first the design goal and programming approach is explained. Then, the data structure and data modelling are explained. Furthermore, the work flow structure and resulting user interface are presented and discussed. Finally, the structure and user interface for the simulator are described.

4.1. Design Goal

The approach to set up the data and work flow structures for the CCG is to define its design goal and system boundaries. These can then be translated to flow diagrams that describe the data and work flow. According to Veeke et al. [47] the goal of a system is *"to fulfil certain functions in its environment, functions that the environment needs for its processes"*. The design goal of the CCG is in essence described by the objectives of this thesis. From the perspective of the application we can rewrite the objectives of the thesis to define a design goal of the CCG. Thus, the goal of the application is:

"...to design a combinator curve for a certain operation mode and to perform a multi-performance trade-off evaluation for a given vessel and its propulsion configuration."

To define the system and subsystem boundaries first the most basic structure of the system is described before going deeper into the system step by step. The most basic structure of the application is shown in Figure 4.1.

Figure 4.1: Simple visualisation of system function.

Certain information is entered into the system that serves as input for the main function of the system. Through the function the input is then transformed into output. For the CCG the input consists of all required data about the ship, its propulsion configuration and any information that is required to calculate and evaluate a combinator curve on performance criteria. The main function of the CCG consists of a calculation core that processes the input data resulting in output data in the form of pitch and speed settings per lever position and quantification of the performance indicators. In order to be able to use the application, that is to insert data, to design a combinator curve and to perform an evaluation based on the calculation result. A user interface is required where this interaction can take place.

4.2. Object Oriented Programming

The programming approach for this application is based on OOP [20]. In this approach a software program is organised by using objects that contain data in the form of methods and attributes. These objects can interact with each other depending on their function and the relation between them. An object can contain and refer to anything, for example a "ship" with an attribute "name" and a method *getName()*. This format will be the same for any vessel and thus for any new ship an instance of this object can be created that contains any attribute and method belonging to the object. A typical way to represent an object such as this example is shown in Figure 4.2.

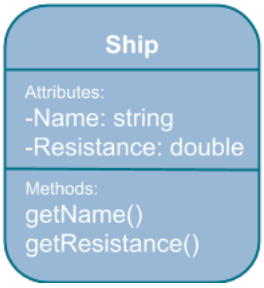


Figure 4.2: Example representation of a class object.

This approach is especially beneficial for complex programs that contain a lot of interdependencies. Instead of organising the program based on functions, each function now becomes part of a certain object to which it belongs. If in any form a new ship object is made and a name is attributed to it, this name can now be used by calling the method *getName()* via this new object. In case an attribute or method must be updated, this can easily be done in the object itself so that everywhere, where this method is called, the updated version is applied. In this chapter arbitrary names for objects and methods will be used to help explain the approach and are examples of what names could be used to specify a certain function or level of the object structure.

4.3. Data Modelling

As mentioned in Section 4.1 the input data of the system includes all data about the ship, its propulsion configuration and any information that is required to calculate and evaluate a combinator curve. And the output data consists of the combinator curve and performance indicators. Without going into detail on what exact information is required and for what purpose, the data structure is shown in Figure 4.3.

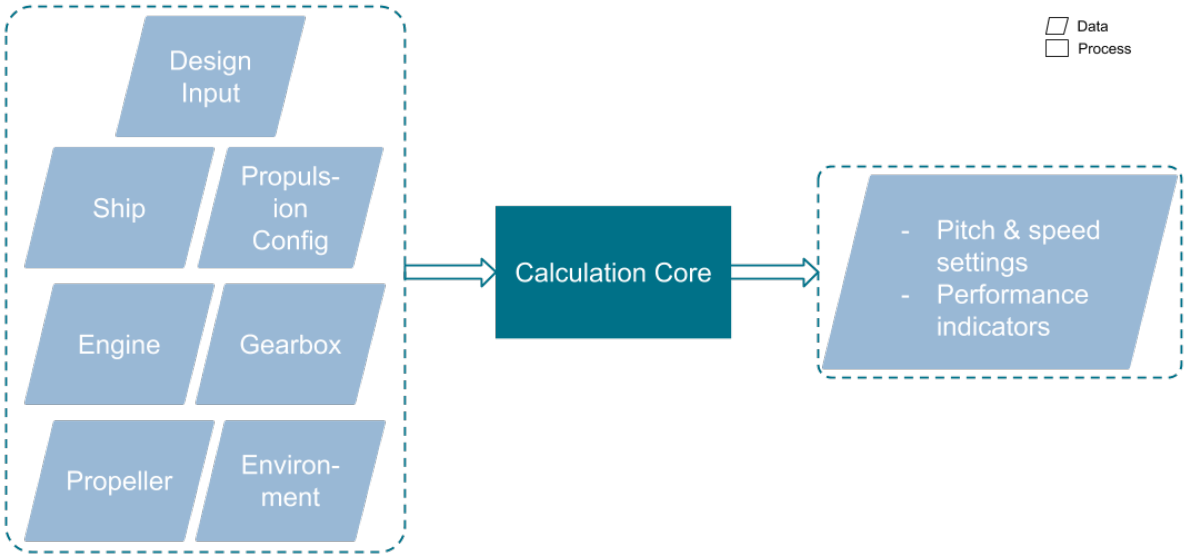


Figure 4.3: Overview of Data Flow Structure.

For now the calculation core will be treated as a black box and we will take a look at each of the data

models. Using OOP, the data is not only stored in a certain object, it is also possible to request information that belongs to that specific sub-system by calling the available methods in that object. For each data model in the next sections the data format is shown. As long as data is inserted in the format belonging to that model, the data will be stored correctly.

4.3.1. Design Input Model

The first data model that is discussed concerns the input data which is necessary to start calculating a combinator curve. Before calculation the user can choose from two options to calculate a combinator curve. The choices are between a set of calculations of at least one certain operation mode or a manual calculation as shown in Figure 4.4.

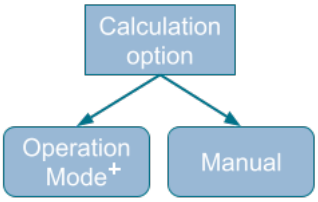


Figure 4.4: Option to calculate for certain performance criteria or for manual design.

In case the user chooses to calculate a combinator curve for a certain criterium, the required input consists of a vessel speed demand as a percentage of the maximum vessel speed for each lever position, see Table 4.1. However, when the user chooses to manually design the combinator curve it is also required to insert the shaft speed settings as a percentage of the maximum shaft speed for each lever position, see Table 4.2. This means that the data model of the design input is programmed as an object with three attributes; lever position LP, vessel speed v_s and shaft speed RPM.

Table 4.1: Representation of Design Input Data when calculating for certain performance criteria.

LP	v_s [%]
0	0
1	10
2	20
...	...

Table 4.2: Representation of Design Input Data when calculating for manually inserted settings.

LP	v_s [%]	RPM [%]
0	0	50
1	10	60
2	20	70
...	...	...

If these attributes would be set directly in this *DesignInput* object, there would exist only one instance of this set of attributes when an instance of this object is created. By using a so called constructor method in a separate object that contains these attributes, *designInputItem* for example. An instance of this separate object can be initiated in the *DesignInput* object for as many that are needed until there exists one instance for each entry in the data set. Within the *DesignInput* object a method is created to fill a list in which the complete data set is stored.

The data set is now stored, but not yet accessible. The calculation of a combinator curve setting starts with knowing the vessel speed at a certain lever position, and depending on the calculation option also the shaft rpm setting. Therefore this *DesignInput* object requires a method to determine the vessel- and shaft speed percentage at a certain lever position. From the constructed list mentioned earlier, the requested data entry can now be accessed by calling this method. If this method is called, the lever position LP must be specified in order to access the correct list entry, for example *getDesignInput(LP)*.

4.3.2. Ship Model

The next data model concerns information about the ship and its resistance data. This resistance data consists of the hull resistance R_T, the wake w and thrust t deduction factors and the rotational efficiency η_{TRM} at each vessel speed v_s for which this data is available. The format of this data set is shown in Table 4.3.

Just like in the procedure from the design input model, these resistance data attributes are not put in the object *Ship* directly, but in a separate object, which we refer to by the name *ResistanceItem*. Apart from the attributes, also methods are put in this *ResistanceItem* object. These methods are mathematical relations that

Table 4.3: Representation of ship resistance data.

v_s [kn]	R_T [kN]	w [-]	t [-]	η_R [-]
0	0	0.14	0.12	0.99
3	12	0.14	0.12	0.99
6	50	0.14	0.12	0.99
...	...	...	...	...

consist of a combination of one or more of these variables and can be accessed through this object. These relations are the effective power, the thrust demand and advance speed, see Equation 4.1.

$$P_E = R_T \cdot v_s, \quad T = \frac{R_T}{(1-t)}, \quad v_a = v_s(1-w) \tag{4.1}$$

Now, to optimise or calculate the combinator settings for a certain vessel speed demand, the resistance data at this vessel speed must be known. Because of this, it is necessary to create a method to interpolate through the data. For this purpose another object *Resistance* is created that contains the method to interpolate through a list that contains the resistance data for a requested vessel speed. Thus, the variables become a function of the vessel speed and a method *getResistanceItem(v_s)* can be used to access the interpolated values. It is in this object where also the list of resistance items is constructed and stored. Then finally the object *Ship* itself may have an attribute *"name"* and requires a method that initializes and creates an instance of a *Resistance* object.

4.3.3. Propeller Model

The data model for the propeller consists of geometry data and OWCs. The top level object gets the name *Propeller* and the geometry data propeller diameter D_p and blade area ratio are directly attributed to this object together with the maximum pitch diameter P/D and the propeller draft. In Table 4.4 an example is shown of the format for the propeller OWCs data.

Table 4.4: Representation of propeller open water data for a controllable pitch propeller.

J [-] / P/D [-]	0.0000	0.1000	0.2000	...
K_T [-]				
2.0000	0.7840	0.7895	0.7560	...
1.7000	0.6985	0.6740	0.6435	...
1.6000	0.6600	0.6425	0.6130	...
...	...	...	...	...
10 * K_Q [-]				
2.0000	2.0450	2.0700	2.0075	...
1.7000	1.5640	1.5240	1.4690	...
1.6000	1.4270	1.3869	1.3350	...
...	...	...	...	...

The OWCs consist of the non-dimensional open water coefficients for a range of pitch-diameter ratios. The objects that make up the model for this data are as follows:

- *propellerOWCSet*
  The complete data set of the propeller OWCs is constructed and stored in the object *propellerOWCSet*. The constructed data set is a list of pitch-diameter ratios and the propeller OWCs that belong to it. Also interpolation methods are added in this object in order to request the OWCs from the propeller model. Either the propeller speed is known in which case the interpolation method is a function of the advance ratio J and the thrust coefficient K_T, for example *getOWC(J,K_T)*. Then, based on the advance ratio and the thrust coefficient the data is interpolated to find the pitch diameter and the complete OWC containing the J, K_T and K_Q characteristics.

  Or, the pitch-diameter P/D is known, in which case the interpolation method *getOWC(PD)* is used. In this case the pitch diameter is known and the OWCs are interpolated that belong to this pitch-diameter.

Both interpolation methods have the purpose of finding the combination of a certain pitch-diameter and the OWC that belongs to it.

- *propellerOWC*
  Then, on the level of a certain pitch-diameter ratio, the entire OWC belonging to this pitch-diameter ratio is constructed and stored as a list in the object *propellerOWC*.

- *propellerOWCItem*
  The lowest object in the hierarchy structure of the propeller model is the object *propellerOWCItem* and it is attributed with J, K_T and K_Q. Next to these attributes also a method is added in this object from which the open water efficiency can be determined (Eq. 2.9).

4.3.4. Main Engine Model

Next, the main engine data model consists of several attributes, the engine operating envelope and the SFOC map of the engine. The *Engine* is attributed with the idle speed $n_{e,idle}$, the lower heating value h^L of the fuel and minimum and maximum specific fuel consumption. Furthermore, the *Engine* is the object in which the objects *Envelope* and *FuelMap* are initialised when a new instance of the *Engine* object is created.

Envelope

First, the operating envelope data contains the shaft speed n_e and brake power P_B. An example of the data format is shown in Table 4.5.

Table 4.5: Representation of operating envelope data for the main engine.

n_e [rpm]	P_B [kW]
300	850
350	1290
400	1790
...	...

A separate object *envelopeItem* is set up that contains the attributes shaft speed and brake power. This object also contains a method with which the engine torque M_B can be calculated, the engine torque is defined by Equation 4.2.

$$M_B = \frac{P_B}{2 \cdot \pi \cdot n_e} \tag{4.2}$$

In order to interpolate through the data set, a list is constructed from the envelope items and is stored in the object *Envelope*. In this object interpolation methods are added to either find the brake power by entering the shaft speed using *getPower*(n_e). Or if the brake power is known the engine speed can be interpolated using *getSpeed*(P_B).

Fuel Map

The engine model also contains objects to store and use the SFOC map of the engine. The data of the fuel map model is in the form of normalized values of the engine shaft speed n_e, brake power P_B and the specific fuel consumption sfc, see Table 4.6.

Table 4.6: Representation of specific fuel map data for the main engine.

P_B [-] / n_e [-]	1.000	0.900	0.800	...
sfc [-]				
1.000	0.380	0.380	0.380	...
0.900	0.400	0.380	0.400	...
0.800	0.420	0.420	0.430	...
...	...	...	...	...

The reason the fuel map format is in normalized values is because it can then be scaled and used for any engine. Often a SFOC map for an engine is presented as contour plots within the operating envelope of the engine as shown in Figure 4.5.

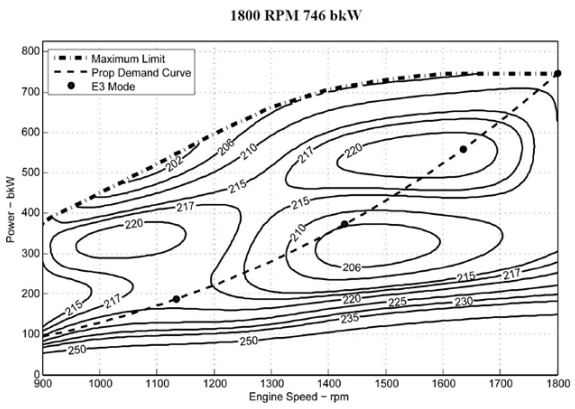


Figure 4.5: Operating envelope and SFOC map.

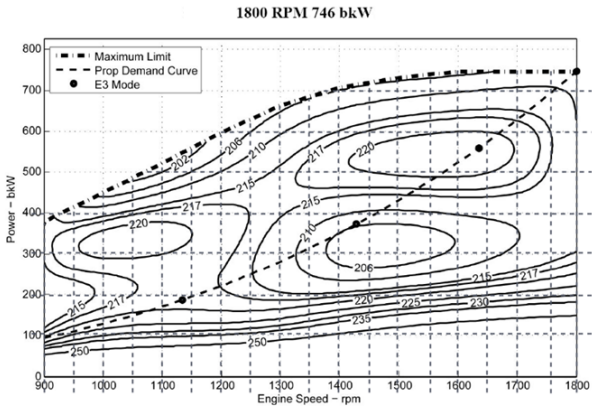


Figure 4.6: Operating envelope and SFOC map overlaid with a grid.

The data can be extracted by laying over a grid as shown in Figure 4.6. Every intersection of the grid is a combination of shaft speed, brake power and specific fuel consumption. In order to be able to scale and use this SFOC map for any engine the data set is normalized. It is chosen to do this in the following way:

- *Shaft speed*
  The shaft speed is normalized to the range $[0,1]$. With 0 representing the minimum shaft speed and 1 the maximum shaft speed, see Equation 4.3.

$$n_{e,norm} = \frac{n_e - n_{e,min}}{n_{e,max} - n_{e,min}} \tag{4.3}$$

- *Brake power*
  Brake power is normalized to the range $[0,1]$ at each shaft speed step. Here 0 represents zero brake power and 1 the break power limit at a certain shaft speed. This results in the following normalisation, see Equation 4.4.

$$P_{B,norm} = \frac{P_B}{P_{B,max}} \tag{4.4}$$

- *Specific fuel consumption*
  The specific fuel consumption is normalized to the range $[0,1]$ where 0 represents the minimal specific fuel consumption and 1 the maximum specific fuel consumption, see Equation 4.5.

$$sfc_{norm} = \frac{sfc - sfc_{min}}{sfc_{max} - sfc_{min}} \tag{4.5}$$

  It might be the case that the minimum and maximum specific fuel consumption are not known. The specific fuel consumption is different for various engine types. Therefore the minimum and maximum specific fuel consumption of the engine are estimated by means of interpolation on the bases of the shaft speed of the engine. For this purpose data from Figure 3.2 is used.

The objects that make up the model for this data are as follows:

- *FuelMap*
  The complete normalized data set is constructed and stored in the object *FuelMap*. The data set is a list of shaft speeds that each belong to a list of brake power data points and a list of specific fuel consumption data points. This object contains interpolation methods such that the specific fuel consumption can be determined based on the shaft speed and the brake power, for example via a method called *getSFC(P_B, n_e)*.

The *FuelMap* object also contains an interpolation method to interpolate through the data set extracted from Figure 3.2. From this interpolation method the minimum and maximum specific fuel consumption values are determined based on the nominal shaft speed of the engine.

- *fuelMapColumn*
  In the object *fuelMapColumn* two lists are constructed belonging to a certain shaft speed step. One list for the normalized values of brake power data points and one for the normalised values of specific fuel consumption data points.

- *fuelMapItem*
  The lowest object in the hierarchy structure of the fuel map model is the *fuelMapItem* which is attributed with $normpb$ and $normsfc$ for the normalized value of the brake power and the specific fuel consumption respectively.

- *fuelMapConverter*
  In order to go from normalized values to full scale and vice versa the object *fuelMapConverter* contains Equations 4.3-4.5.

4.3.5. Propulsion Configuration Model

The propulsion configuration model is an object that consists solely of a constructor method to which data is attributed that describes the propulsion configuration in terms of four parameters. These are:

- Number of driven shafts k_p [-].
- Number of connected engines k_e [-].
- Power Take Off PTO [kW]
- Power Take In PTI [kW]

4.3.6. Gearbox Model

The gearbox model is a data model that contains a constructor method and the following attributes:

- Gearbox ratio i_{GB} [-].
- Transmission efficiency η_{TRM} [-].

4.3.7. Environment Model

There are several parameters about the environment that are stored into an object consisting of a constructor method and several attributes. These include:

- Sea water density ρ_{sw} [kg/m^3]
- Ambient pressure p_a [Pa]
- Vapour pressure p_v [Pa]

4.3.8. Design Result Model

Then finally, the results of the combinator curve calculation are stored in a separate data model as well.

- *combinatorProject*
  The top level object of this structure is called *combinatorProject* in which a complete set of *combinatorResults* is stored for every optimised or calculated combinator curve option.

- *combinatorResult*
  In the object *combinatorResult* the a list is constructed that contains the results per lever position. Furthermore a calculation indication is attributed to this object about the calculation option that was used.

- *combinatorResultItem*
  On the lowest level, the object *combinatorResultItem* contains the combinator curve result for a certain lever position in the form of the combinator curve settings and system performance indicators. Without going into further detail here, the *combinatorResultItem* contains the following data:

  - Lever position LP [-]

- Vessel speed v_s [kn]
- Pitch-diameter ratio P/D [-]
- Propeller speed n_p [rpm]
- Brake Power P_B [kW]
- Propeller open water efficiency η_o [-]
- Advance Ratio J [-] and $\triangle J$ [j]
- Engine Efficiency η_e [-]
- Fuel Consumption fc [ton/hr]

4.4. User Interface Design

Now that the data structure and data format is known a user interface can be designed with which a user can interact to achieve the goal of the CCG. In this section first the structure of the user interface will be discussed. Based on this structure aspects can be identified that are required in order to be able to design a combinator curve and to evaluate the result on the basis of performance indicators. Lastly, the final user interface is shown and discussed.

4.4.1. Work Flow Structure

In order to gain more insight into what features the user interface should possess, a work flow structure is designed. The interaction of the user with a user interface is indicated with the green coloured diagram blocks, see Figure 4.7.

At start, the application is opened and a new project can be created. The user must then be able to insert all design and system input data that is required to start an optimisation or calculation. When starting the process, this data can then be stored into the data models and an optimisation or calculation of a combinator curve can be done. When the process is finished, the resulting data is stored and presented in tables and used to create graphs. The visualisation of the data must be done in such a way that the user is supported while designing a combinator curve. The optimised or calculated combinator curve can then be evaluated based on several performance indicators and the user's knowledge and expertise. If the performance is not satisfactory, the user should be able to adapt the inserted design input data and if necessary, the system data, so that a new optimisation or calculation can be performed. When the combinator curve has the desired performance for the intended operation mode, the user can save and print this data and has completed a successful combinator curve design.

4.4.2. Final User Interface

At opening the application the user will see the user interface as shown in Figure 4.8. To start a project, the user can enter in all data that is required to design the combinator curve in the top left part with the title "Main Input". Next to a project name data about the propulsion configuration, the engine, the gearbox, the propeller and the ship resistance should be inserted.

Main Input

- *Main Engine Data*
  For the engine there exists a data base from which an engine can be selected. When the button next to "Select Engine and Fuel Map" is pressed a pop-up window will appear where an engine can be selected from the data base, see Figure 4.9.

  Besides this, a SFOC map can be chosen from a data base. If the SFOC data is known it can be uploaded from a *.text* file. Furthermore, the lower heating value of the fuel, together with the minimum and maximum specific fuel consumption of the engine can be inserted if these values are known. Otherwise the minimum and maximum specific fuel consumption values are interpolated on the basis of the nominal power and shaft speed of the selected engine.

- *Propeller Data*
  For the propeller OWC characteristics a *.text* file containing this data can be uploaded. Upon pressing the button next to "Select Propeller" a File Explorer window will open from which the correct *.text* file can be searched and selected.

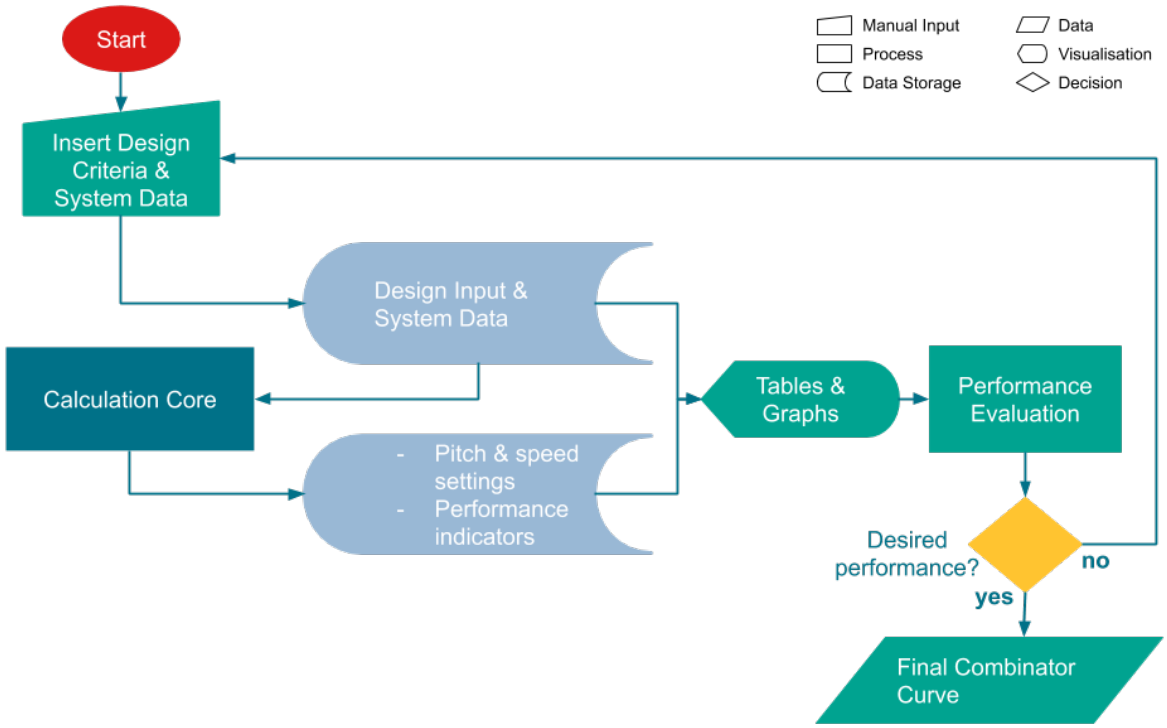


Figure 4.7: Overview of Work Flow Structure.

- *Constraints*
  In the block titled "Constraints" the user is able to indicate whether the brake power limit of the engine should be taken into account in the combinator curve design and if so, what engine margin is desired.

  Furthermore, there is the possibility to insert environmental data. This is done in the pop-up window that appears upon clicking the button "Exert Input", see Figure 4.10. The values that can be seen in the figure are default values for the environmental data. If the user does not adjust these values, the default values are used. The default values for the environmental data are chosen for sea water at a temperature of 15 °C [23].

- *Lever Control Input*
  To design a combinator curve, lever control input must be inserted, see Figure 4.11. The lever control input consists of a certain vessel speed as a percentage of the maximum vessel speed for each lever position. Finally, an optimisation or calculation option should be chosen from the drop down menu. When the manual calculation option is chosen from the drop down menu, a third column appears in the lever control input block. In this column the shaft speed is to be inserted as a percentage of the maximum shaft speed. These inserted speed settings will then be used to calculate the pitch in order to overcome the thrust demand.

Combinator Curve Results

When all data is inserted the combinator curve is optimised or calculated after pressing the button "calculate". As soon as the process is finished, the resulting data will be presented in tables within the block titled "Combinator Curve Results".

The resulting data is also visualised in the form of several graphs; the propeller load within the operating envelope, the combinator settings , the power absorption diagram and the resistance data. In Figure 4.12 the operating envelope is shown together with the propeller load and a colour map to visualise the SFOC map. The darker spot around 450 rpm indicates the lowest specific fuel consumption. The combinator settings show the combinator settings and the vessel speed per lever position. The pitch diameter ratio, shaft speed and vessel speed are shown as a percentage of their maximum value. Next, the power absorption diagram in Figure 4.14 shows the propeller load in the operating envelope for all driven propeller shafts and the number of driving engines per shaft line. And lastly Figure 4.15 shows the resistance data in terms of the resistance,

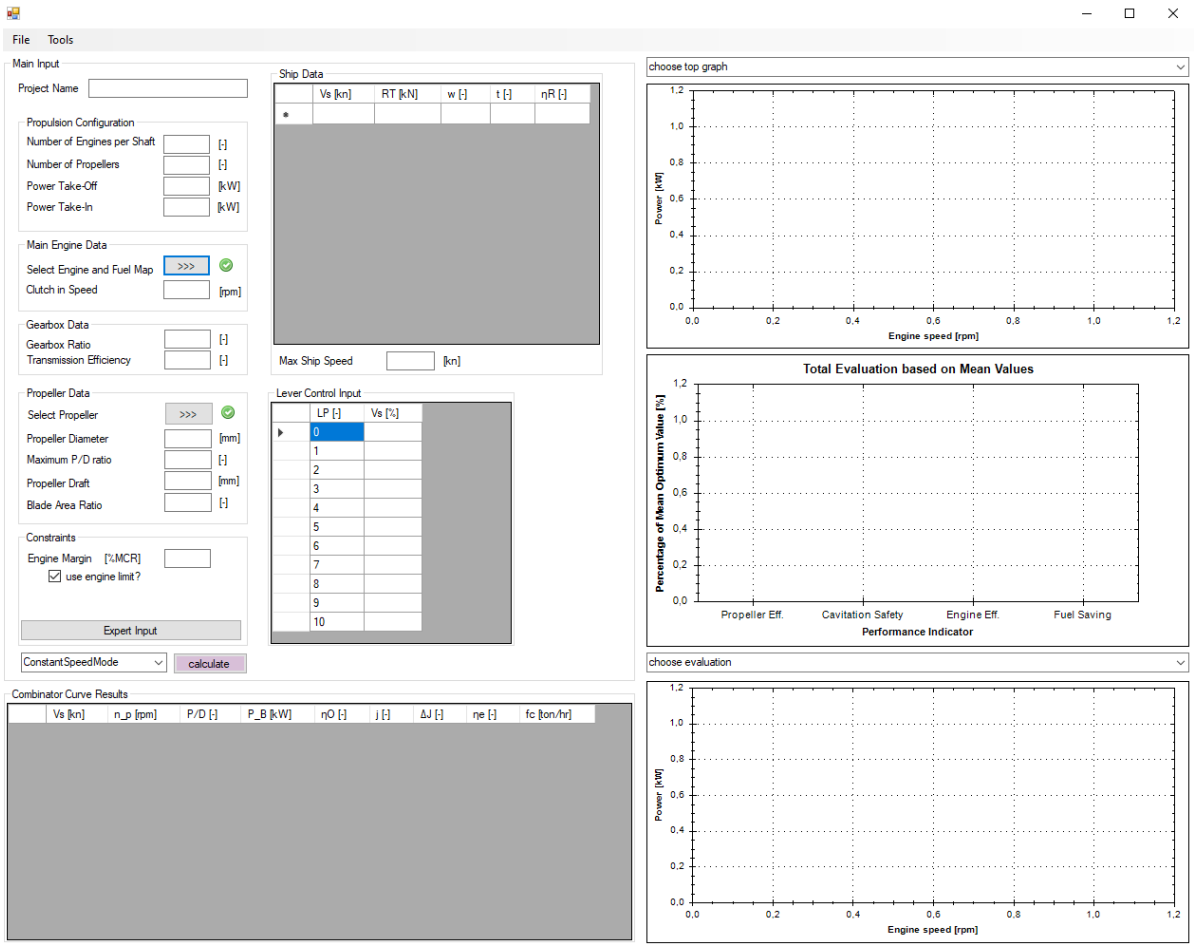


Figure 4.8: User Interface of Combinator Curve Generator.

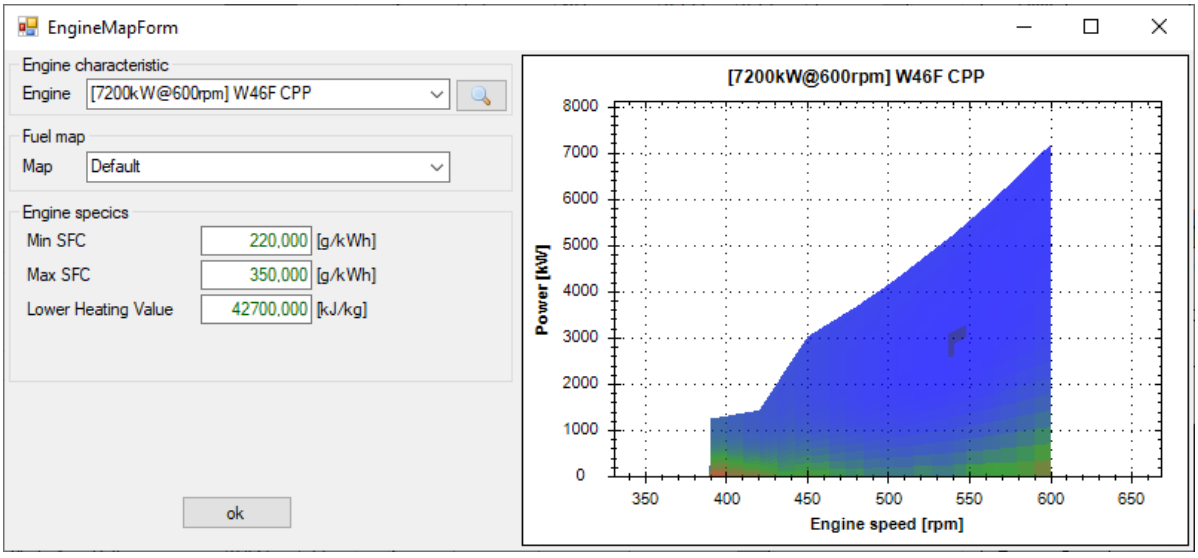


Figure 4.9: Image showing the engine selection user interface.

the wake factor, the thrust deduction factor and the rotative efficiency.

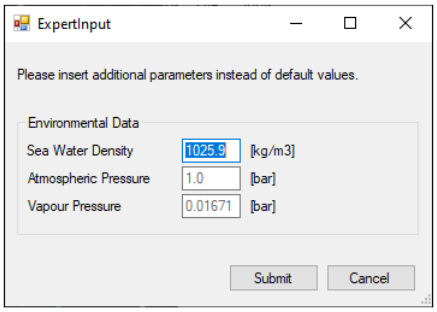


Figure 4.10: Expert Input User Interface

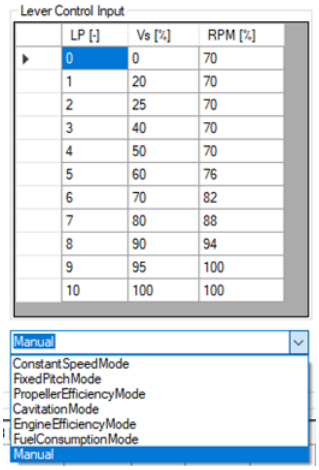


Figure 4.11: Image showing the drop down menu for calculation options and the lever control input columns of which the third column is used to insert the speed settings for each lever position and vessel speed.

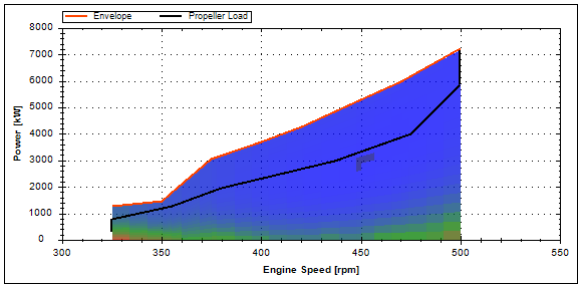


Figure 4.12: Operating Envelope

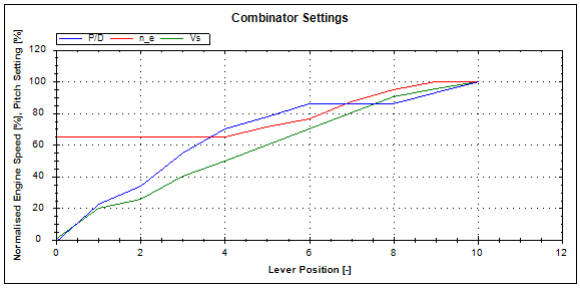


Figure 4.13: Combinator Settings

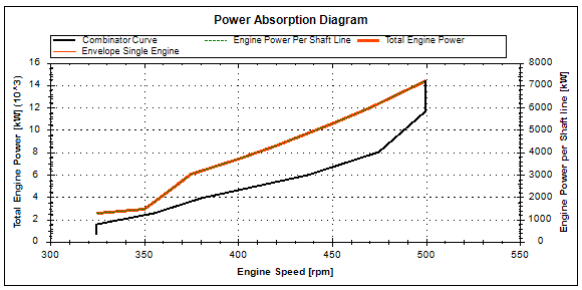


Figure 4.14: Power Absorption Diagram

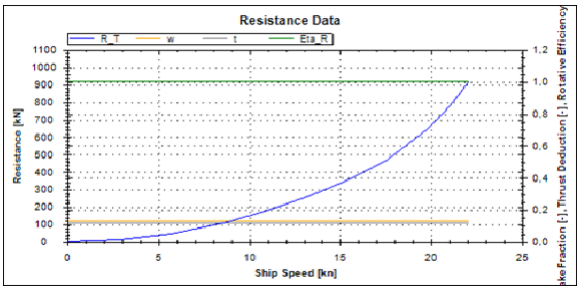


Figure 4.15: Resistance Data

Evaluation

Finally, the performance indicators are visualised in two forms such that the combinator curve can be evalu-

ated on the bases of performance in terms of propeller efficiency, cavitation safety, engine efficiency and fuel consumption.

In Figure 4.16 the total evaluation is shown per performance indicator and based on the mean value as a percentage of the mean optimum value over all lever positions. By showing the performance in one total overview the user can quickly observe and evaluate the mean performance of the combinator curve design. In this case it can be seen that the combinator curve design performs very well in terms of propeller efficiency and fuel consumption, but less in terms of cavitation and engine efficiency.

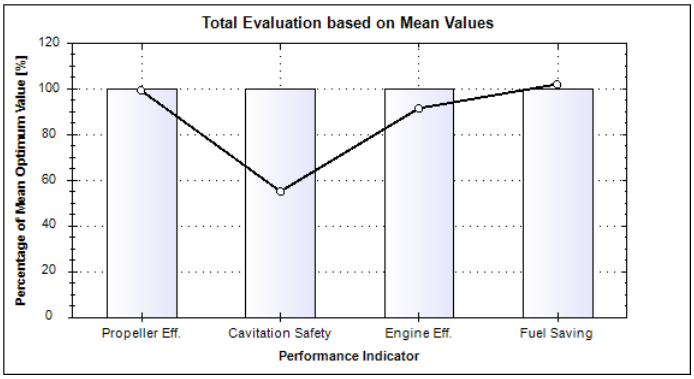


Figure 4.16: Total Evaluation Based on Mean Values

In order to have more detailed insight the combinator curve design performance indicators and the optimum values of the performance indicators per lever position are shown together. For each performance indicator a separate graph is shown. First, in Figure 4.17 it can be observed that the performance graph of the combinator curve in terms of propeller efficiency closely follows the graph of optimum values. In Figure 4.18 the performance graphs for cavitation safety are shown and it can be observed that the optimum values are only achieved in the lowest and highest lever positions. Engine efficiency is shown in Figure 4.19 and up till lever position 7 the values deviate from the optimum values. Lastly, the performance of the combinator curve in terms of fuel consumption closely follows the optimum values, see Figure 4.20.

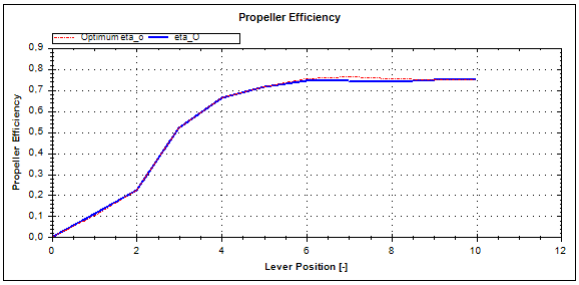


Figure 4.17: Propeller Efficiency

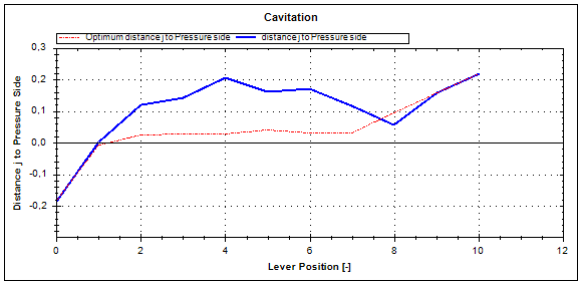


Figure 4.18: Cavitation Safety

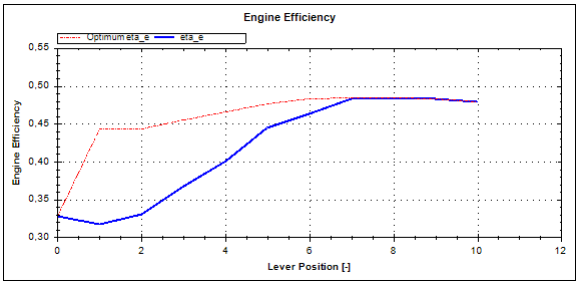


Figure 4.19: Engine Efficiency

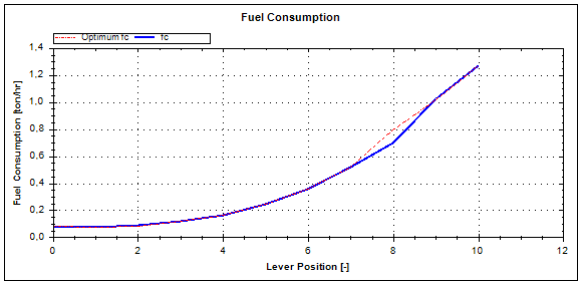


Figure 4.20: Fuel Consumption

4.5. Simulator

The simulator described in this section is complementary to the evaluation for fuel consumption performance. This is done because it is important to take into account the operational profile of a vessel including hotel load. Without going into detail about how the simulator is developed, the object structure and user interface are discussed and presented.

4.5.1. Simulator Model

- *Simulation*
  The object *Simulation* contains all methods required to run the complete simulation and to calculate the results for every time step. Within the object a list is constructed and stored that contains the results for one simulation run at every time step. Furthermore, the object contains a method to construct a list to collect all simulation runs together with an indication of the lever position that was used for cruising. The *Simulation* object is attributed with the following parameters:
  - In port Standby Time [min]
  - Manoeuvre Time [min]
  - Cruise Distance [nm]
  - Hotel Load in port [kW]
  - Hotel Load during manoeuvring [kW]
  - Hotel Load during cruising [kW]
  - Vessel Displacement [m^3]
  - Surge Added Mass [
  - Pitch Ramp Up time [s]
  - Max Trip Duration [min]
- *simulationResult*
  The *simulationResult* contains a constructor method with an attribute Lever Position at cruising and a list attribute for one simulation run.
- *simulationResultItem*
  For each time step in one simulation run the results are stored using the object *simulationResultItem* which contains a constructor method and is attributed with the following items:
  - Total Time [s]
  - Part Time [s]
  - Part Distance [m]
  - Vessel Speed [m/s]
  - In port Standby Hotel Load [W]
  - Propulsion Load [W]
  - Total Fuel Consumption (Hotel Load + Propulsion) [kg]
  - Propulsion Fuel Consumption [kg]
  - Fuel Consumption Step [-]
  - Advance Ratio J [-]
  - Propeller Speed [rps]
  - Pitch [-]
  - Brake Power limit [W]
  - Thrust [N]
  - Thrust Demand [N]

4.5.2. Simulator User Interface

When the user has optimised or calculated a combinator curve design the results can be used to further evaluate the design by using the simulator. Under "Tools" in the menu bar the user can select the item "simulation", see Figure 4.21.

Upon clicking, a window will pop up containing the user interface of the simulator, see Figure 4.22. In the "Trip Data" all required data about the trip can be inserted for the vessel in port, during manoeuvring and during cruising. Under "Simulation Input" the user must insert additional information about the vessel and time requirements. Upon clicking the button "Start Simulation" the simulation process will start. As soon as

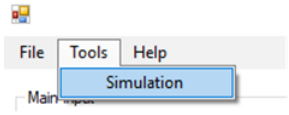


Figure 4.21: Find the Simulator under "Tools".

the process has finished the results will be shown in several graphs. The graph currently presented in Figure 4.22 shows the accumulated total fuel consumption in tons versus the time step in minutes. On the right hand side of the graph the total fuel consumption is indicated at the time stamp where the user draws the cursor over the graph. At the current position of the cursor, the total fuel consumption is indicated for all graphs ($LP = 6 - LP = 10$) at time stamp 138.23 min. Also, the end trip results are shown to indicate the total fuel consumption at the end of the trip.

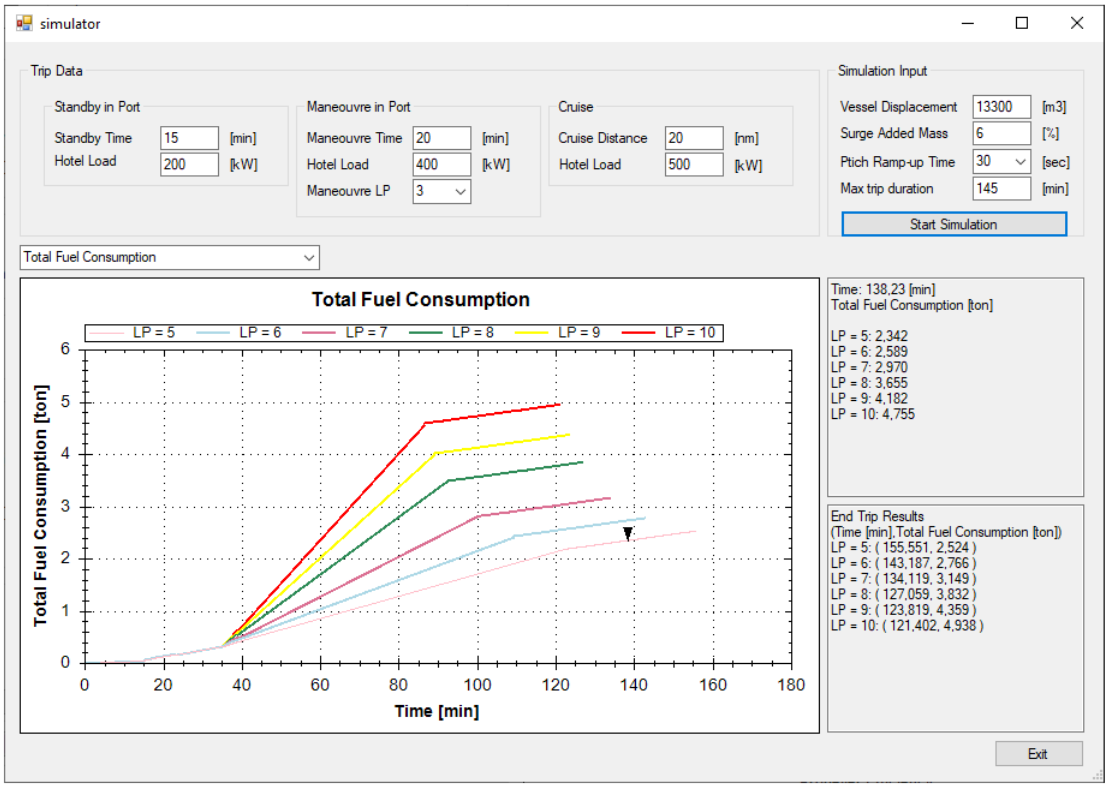


Figure 4.22: Simulator User Interface

5

Calculation Core

In the former chapter, the CCG is explained in terms of its data and work flow structure, and the calculation core was treated as a black box. In this chapter the calculation core will be explained. First, the structure of the calculation core is discussed. Then, the benchmark used to evaluate the resulting combinator curves is introduced. In this chapter the calculation approaches for the operation modes "default", "constant speed" and "manual" are explained and the results shown and discussed. The optimisation approaches and results for the propeller and engine performance indicators are discussed in Chapter 6 and 7 respectively.

5.1. Structure

The programming approach OOP discussed in Section 4.2 is also used for the calculation core. The structure of the hierarchy of the objects that are required to fulfil the function of the calculation core of the CCG is shown in Figure 5.1.

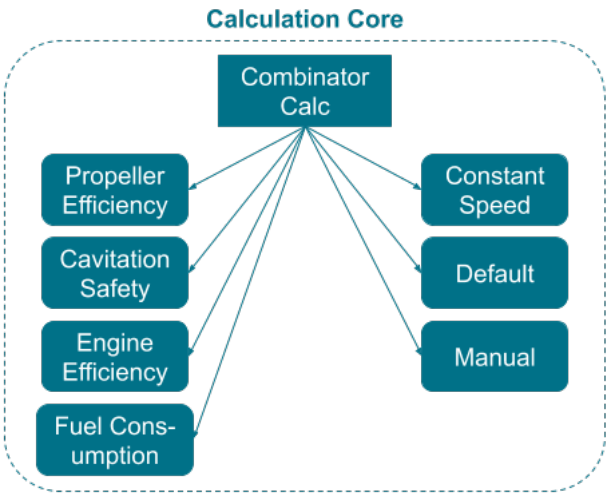


Figure 5.1: Calculation Core Structure.

When the data is stored in the models and the calculation of a combinator curve is initiated, all data is collected in the *CombinatorCalc* object. The object *CombinatorCalc* also contains a method that determines for which of the options a combinator curve is calculated. Then, for each of the calculation options an object exists which contain calculations for a propeller pitch and speed setting and performance indicators of a certain lever position and its corresponding vessel speed demand. Based on the selected calculation option an instance is created of the corresponding calculation object for every lever position in the data set. Every calculation option object contains a method to create an instance of the *combinatorResultItem* discussed in Section 4.3.8. In this way the resulting data is stored in an object for each lever position of a certain calculation option.

5.1.1. Calculation Structure

Before discussing the calculation approach for each of the calculation options, the calculation structure of each of these objects will be discussed because they are similar in each of them. The calculation structure consists of the following parts:

- Declaring Objects, Parameters and Variables

- Calculation or Optimisation Algorithm
  The calculation or optimisation algorithms consist of calculations and procedures that are required to determine a pitch and speed setting for a certain lever position and corresponding vessel speed. The calculations that relate to a certain system such as the ship or the propeller however, are done in the object to which its data belongs. For example, the thrust required to sail at a certain vessel speed is calculated in the object *ResistanceItem*, as was explained in Subsection 4.3.2. If any stored data is required, it is accessed by using the available method via the corresponding object.

- Engine Limit and Engine Margin
  A successful matching of the propeller and the engine is one where the propeller load does not lie too close to the engines torque and power limit to avoid overloading and increasing maintenance costs. For this purpose an engine limit check of the resulting combinator settings can be done. The following procedure is performed to check whether the limit is exceeded, either with or without engine margin, and if necessary to re-calculate the combinator settings:

  1. Determine the thrust demand to overcome the vessel speed at the current lever position via the *ResistanceItem* object.
  2. Determine the engine limit by means of interpolation via the *Envelope* object with the current engine shaft speed setting.
  3. Calculate the advance coefficient using the propeller speed and Equation 2.6.
  4. Calculate the thrust coefficient using Equation 2.7.
  5. Determine the OWC by means of interpolation as a function of $owc(j, kt)$.
  6. Determine the torque coefficient by means of interpolation of the advance ratio via the *PropellerOWC* object.
  7. Calculate the required brake power for the propeller using Equations 2.11 and 2.13.
  8. Check if the required brake power for the propeller exceeds the engine limit.
  9. If the engine limit is exceeded, increase propeller speed and repeat steps 2-8.

  As was shown in Subsection 4.4.2, the user can indicate an engine margin as a percentage of the engine limit and choose whether to use this procedure or not.

- Performance Indicators
  In order to evaluate the combinator settings, several performance indicators are calculated at the end of the algorithm. These indicators will become clear as the calculation and optimisation approaches are further explained, and include the following:

  – *Propeller open water efficiency* η_o *[-]*
    The propeller open water efficiency is determined via the *OWCItem* object.

  – *Advance Ratio* J *[-] and* $\triangle J$ *[j]*
    The advance ratio is known when the combinator settings are calculated and $\triangle J$ [j] is determined, which will be explained in Section 6.2.

  – *Engine Efficiency* η_e *[-]*
    The engine efficiency is defined by Equation 7.2.

  – *Fuel Consumption* fc
    The fuel consumption per unit time is defined by Equation 7.3.

The source code for each of the calculation cores can be found in Appendix B.

5.2. Benchmark

The benchmark with which the CCG is tested and evaluated concerns a vessel similar to the ro-ro ferry shown in Figure 5.2. This particular roro ferry is optimised for propeller efficiency and trip time. The vessel employs two CPPs, each driven by a diesel engine running at 500 rpm at a maximum power of 7200 kW. The cruising speed is 19.8 knots and if needed a maximum vessel speed of 22 knots can be reached. The required data for the CCG consists of the input data about the ship, the propeller and the propulsion configuration, including those shown in Figure 5.3.

Vessel particulars:	
Design Point	Propeller Efficiency
Lpp	143.0 m
B	25.4 m
T	5.9 m
Displacement	13300 m^3
2xCPP	
2xDiesel Engine	7200kW@500rpm
Design Speed	19.8 knots
Max. Speed	22 knots

Figure 5.2: UN Karademiz [17].

Propeller Data

Diameter	[mm]	5000
Max P/D ratio	[-]	1,436
Propeller Draft	[mm]	4000
AeAo	[-]	0,427

Gearbox Data

Gearbox ratio	[-]	4,1841
Transmission efficiency	[-]	0,95

Propulsion Configuration Data

Nr. of propellers	[-]	2
Nr. of engines per shaft	[-]	1
Power Take Off	[kW]	0
Power Take In	[kW]	0

Resistance Data

vs [kn]	R_T [kN]	w [-]	t [-]	Eta_R [-]
0,0	0,0	0,132	0,120	1,000
2,9	12,1	0,132	0,120	1,000
5,7	48,5	0,132	0,120	1,000
8,8	114,7	0,132	0,120	1,000
11,0	179,4	0,132	0,120	1,000
13,2	258,1	0,132	0,120	1,000
15,4	351,5	0,132	0,120	1,000
17,6	472,1	0,132	0,120	1,000
19,8	650,9	0,132	0,120	1,000
21,0	779,3	0,132	0,120	1,000
22,0	915,1	0,132	0,120	1,000

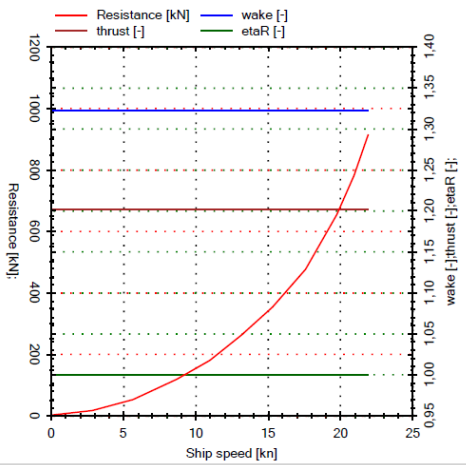


Figure 5.3: Ship and Propulsion System Data

5.3. Constant Speed Mode

As mentioned in Subsection 2.2.3, the constant speed mode is added to the CCG as a standard calculation option. For this operation mode the shaft speed is held at a constant maximum shaft speed and the pitch setting is calculated accordingly.

Procedure

When all data is inserted and the calculation option *Constant Speed* is selected, the pitch and speed setting will be calculated for each lever position and vessel speed demand. In the calculation object for the constant speed mode, the propeller speed remains constant in the maximum speed $n_p = n_{p,max}$ for every lever position. This means that the pitch is varied at each lever position in order to supply the thrust demand to sail at the vessel speed demand. The steps that are taken for this calculation are the following:

1. Calculate the advance coefficient using the known propeller speed and Equation 2.6.
2. Determine the thrust demand to overcome the vessel speed at the current lever position via the *ResistanceItem* object.
3. Calculate the thrust coefficient using Equation 2.7.
4. Determine the pitch setting by means of interpolation of the OWC as a function of $owc(j, kt)$.

5.3.1. Result

The resulting combinator settings for the calculation of the constant speed operation mode are shown in Figure 5.4.

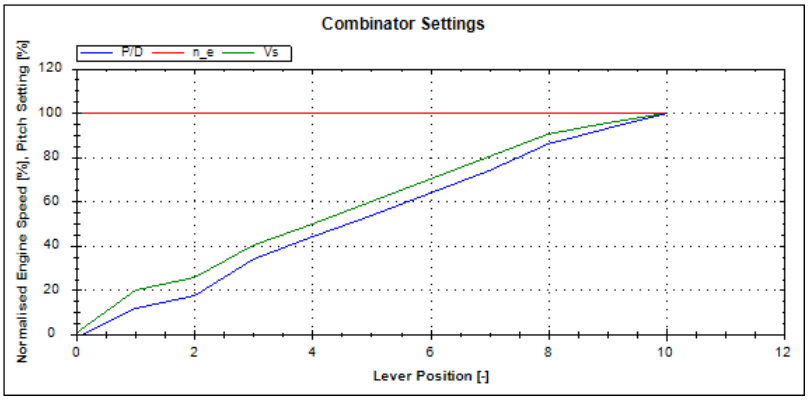


Figure 5.4: Combinator Settings - Fuel Saving

The settings for the pitch-diameter ratio and the rotational speed are shown per lever position together with the vessel speed demand in percentages. For each lever position the rotational speed is at 100%, and as the vessel speed increases at each lever position, so does the pitch setting. The resulting propeller load within the operating envelope of the engine is shown in Figure 5.5.

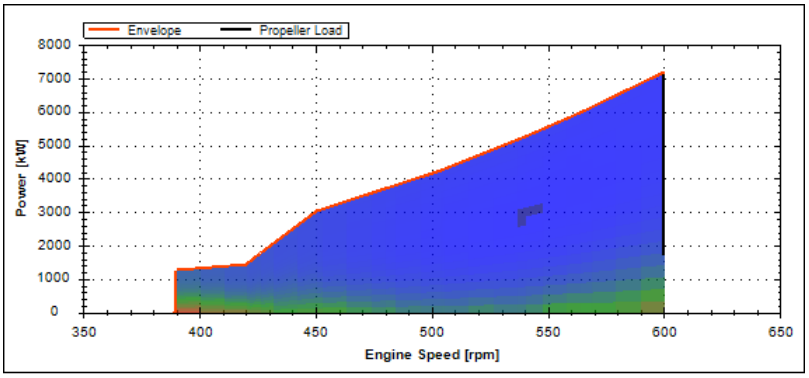


Figure 5.5: Engine Envelope - Constant Speed

In the engine envelope, the black line represents the propeller load. Because the shaft speed is constant in the maximum shaft speed, all working points of the engine are at the limit of 500 rpm. With each lever position the vessel speed and thrust demand increase, and so does the delivered power by the engine.

5.4. Default Mode

As explained in Section 2.2.3, the CPP is often operated in a combinator mode. Here, instead of operation as a FPP in a constant pitch, the combinator curve also depends on propeller pitch limits and the speed, torque and power limits of the engine. This means that the pitch is held constant in the design pitch $P_{pd} = P_{pd,design}$ as long as the resulting operating point do not exceed any machinery limits. This calculation core is given the name *Default Mode* because it is common to set up a combinator curve in this way.

Procedure

The approach for this calculation is to start with a certain propeller speed setting and to decrease the speed until an equilibrium is established between the developed thrust for the combination of design pitch and shaft speed setting and the thrust demand to achieve the current vessel speed demand. For this approach, the maximum propeller speed is chosen as the initial propeller speed, so $n_p = n_{p,max}$. The following steps are taken to find the combination of design pitch and speed to achieve the thrust demand:

1. Determine the thrust demand to overcome the vessel speed at the current lever position via the *ResistanceItem* object.
2. Calculate the advance coefficient using the current propeller speed and Equation 2.6.
3. Determine the thrust coefficient by means of interpolation in the OWC belonging to the design pitch.
4. Decrease propeller speed if $thrust > thrustdemand$.
5. Repeat step 2-4 until threshold for thrust demand is achieved.

Constraints

Often, the idle speed of the engine is too high to develop the thrust demand in the lower lever positions in the design pitch. This means that in the lower lever positions the speed remains constant in the idle speed and the propeller pitch has to be reduced. And in the highest lever positions it might be the case that the maximum speed of the engine is too low in order to achieve the maximum vessel speed in the design pitch. In this case the maximum propeller speed is limited by the maximum engine speed and the pitch is increased. The steps that are taken are the following:

6. Set propeller speed to minimum propeller speed if $n_p < n_{p,min}$.
7. Set propeller speed to maximum propeller speed if $n_p > n_{p,max}$.
8. Calculate the advance coefficient using the propeller speed and Equation 2.6.
9. Calculate the thrust coefficient using Equation 2.7.
10. Determine the pitch setting by means of interpolation of the corresponding OWC as a function of $owc(j, kt)$.

5.4.1. Result

The resulting combinator settings for the default operation mode are shown in Figure 5.6.

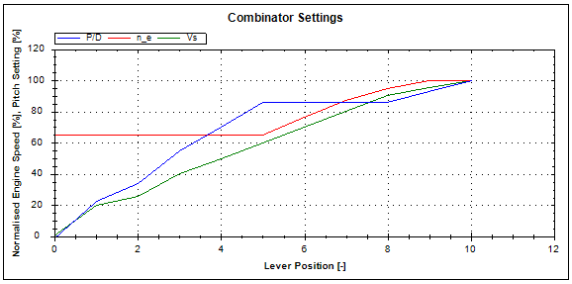


Figure 5.6: Combinator Settings - Default no Engine Margin

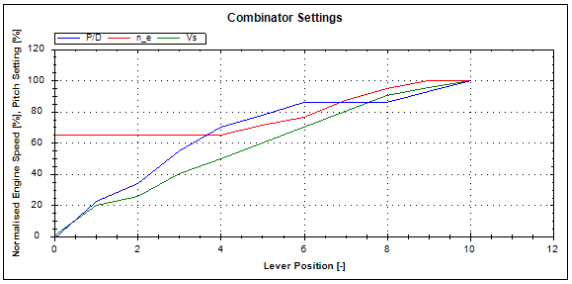


Figure 5.7: Combinator Settings - Default with 85% Engine Margin

For lever positions 0-5 the shaft speed is kept in the idle speed while the pitch increases according to the increased vessel speed demand with each lever position. In Figure 5.8 the propeller load is shown within the operating envelope of the engine.

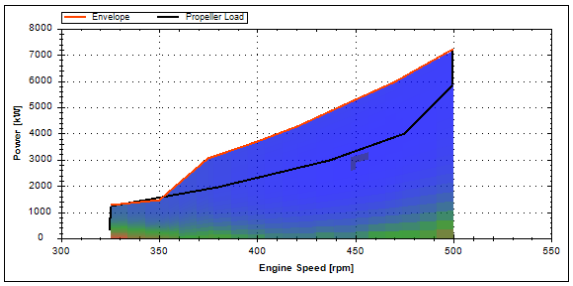


Figure 5.8: Combinator Settings - Default no Engine Margin

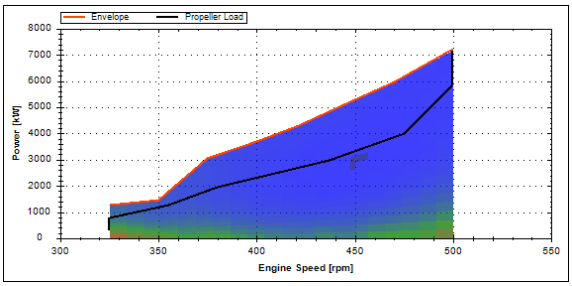


Figure 5.9: Combinator Settings - Default with 85% Engine Margin

As soon as the combination of pitch and shaft speed do not result in a working point that exceeds the power limit of the engine, the pitch is held constant and the shaft speed is increased up to lever position 8. Then, for the final lever positions 9-10, both the pitch and speed are increased up till their maximum working point.

Even though, in Figure 5.8 the engine power is not exceeded, it is likely that the engine will be overloaded when the propeller load is so close or at the power limit of the envelope. To prevent the engine from overloading, it would be better to calculate the combinator settings whilst taking into account an engine margin. To show the result of using an engine margin, the combinator settings that are calculated with an engine margin of 85%MCR are shown in Figure 5.7. Now, the shaft speed is held constant in the idle speed for lever positions 1-4 and for lever position 5 both pitch and shaft speed are increased. The resulting propeller load is shown in Figure 5.9.

5.5. Manual Design Mode

Finally, the manual design mode is a calculation option that allows the user to manually design the combinator curve. This can be done by inserting the desired shaft speed settings in the third column shown in Figure 4.11. Using these settings for the shaft speed, the pitch setting in order to deliverer the required thrust demand is calculated accordingly for each lever position.

For several test cases the originally designed combinator curve settings have been used to test the CCG, and to evaluate the resulting performance indicators. These test cases can be found in Appendix C.

6

Propeller Performance Indicators

In this chapter the optimisation approaches are described for the performance indicators related to the propeller, which are propeller efficiency and cavitation inception. As explained in the former chapter, a separate calculation core is developed, in which the calculation or optimisation algorithm is included for a certain operation mode or performance indicator. For each performance indicator described in this chapter, first the optimisation approach and procedure are described after which the resulting optimised combinator settings are shown and discussed. For the data used to test and evaluate the optimised combinator settings in this chapter, refer to the benchmark introduced in Chapter 5. Finally, the chapter is concluded.

6.1. Propeller Efficiency

The first optimisation approach discussed in this chapter concerns the performance indicator propeller efficiency. The combinator settings are optimised such that the combination of pitch and speed is the optimum working point for the propeller in terms of efficiency, which is defined by the open water efficiency defined by Equation 2.9.

The propeller performance characteristics data, shown in Figure 2.2, shows that open water efficiency increases with increasing advance ratio up to a certain maximum and the advance ratio increases with decreasing rotational speed, Equation 2.6. For the purpose of finding the optimum working point in terms of open water efficiency, the OWC data of the propeller is used as explained in Subsection 4.3.3.

The approach taken to find the optimum combinator settings in terms of open water efficiency, is to vary the rotational speed while simultaneously determining the correct pitch in order to achieve the thrust demand at a certain lever position and respective vessel speed demand. This is done by means of interpolation through the performance characteristics data. For each variation in speed and pitch the efficiency is determined from the respective interpolated OWC data, and finally the combinator setting resulting in the highest open water efficiency is chosen as the optimum setting. In the next paragraph the optimisation procedure is described in more detail.

Procedure

There are a large number of possible pitch and rotational speed setting combinations that all result in a working point that deliver the desired thrust demand at the respective lever position and vessel speed demand. To be able to find which combination has the highest efficiency without having to calculate each possibility, an approach is taken in order to limit the number of calculations.

First, the required pitch to achieve the thrust demand for the current lever position and vessel speed, is determined for a certain range of rotational speeds $[n_{p,min}, n_{p,max}]$ and a certain step size. Then, depending on several constraints discussed later, the resulting interpolated OWC belonging to the latest determined pitch setting is stored in a list. From the resulting list containing the pitch related OWC's, the list entry is chosen that results in the highest efficiency.

The first round of calculations are based on the rotational speed range which is dependent on the rotational speed limits of the engine-gearbox configuration in percentage steps. To narrow down the speed interval for which combinations of pitch and rotational speed settings are calculated, the interval is changed. This is done by taking one speed step before, and one speed step after the rotational speed setting, belonging

to the lastly determined optimal combinator setting. Thus, the rotational speed interval is updated and the procedure is performed for a number of iterations.

By narrowing down the rotational speed interval several times, the accuracy of the resulting optimal combinator setting increases. However, the desired accuracy is limited due to the limited precision with which the propeller pitch and engine shaft speed can be controlled. In other words, after a certain amount of iterations, the difference between two sets of combinator settings will not be felt in full scale application. Besides, the accuracy of the resulting combinator setting depends partly on the data itself. Thus, if the data is more reliable, the accuracy of the resulting combinator setting increases. Finally, in this approach the interval steps are percentage steps and thus the same for each iteration. Because of this, the absolute accuracy will be different for each engine because the speed range varies per engine.

To conclude, the procedure within each iteration and for each speed step is the following:

1. Determine the thrust demand to overcome the vessel speed at the current lever position via the *ResistanceItem* object.
2. Calculate the advance coefficient using the current propeller speed and Equation 2.6.
3. Calculate the thrust coefficient using Equation 2.7.
4. Determine the pitch and OWC by means of interpolation as a function of $owc(j, kt)$.
5. Determine the open water efficiency via the *OWCItem* object.

Constraints

The next steps in the procedure include constraints to determine whether an OWC is included to the list or not. The first constraint is to check if the calculated pitch is smaller or equal to the maximum pitch setting. If so, this OWC is added to the list, otherwise, if the pitch is larger then the maximum pitch, the OWC for this pitch setting is left out of the list.

The next check concerns the propeller efficiency and can be explained in the OWC diagram, see Figure 6.1.

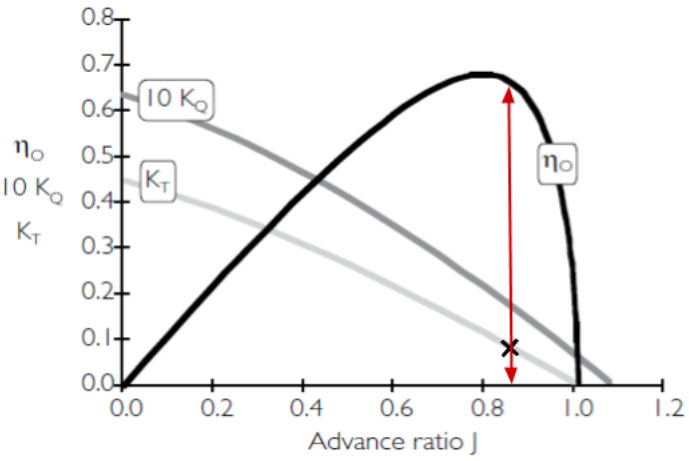


Figure 6.1: Open water propeller curves and ship curve with indication of a working point after the peak of the efficiency curve. Adapted from Klein Woud and Stapersma [24].

In this diagram, the OWC is shown for a certain pitch. The cross indicates a possible working point of the propeller for a certain speed. As can be seen from the diagram, it might occur that a working point is determined such that it lies on the right side of the peak in the efficiency curve. As explained in Section 2.1 the wake field at the propeller disc is not uniform and the water flows at a certain angle due to the hull form at the aft of the ship. The wake factor used to determine the advance ratio (Eq. 2.6) is in fact the mean wake factor. However, depending on the location in one turning cycle of the propeller, the advance ratio will be smaller or larger due to the variation in the wake field. If part of the propeller is at a point in the wake field where the advance ratio is increased, the efficiency decreases and there will be loss of thrust. To avoid this, it is important to choose a working point that lies before the efficiency peak.

Only those OWC's should be added to the list, that have a working point of the propeller before the open water efficiency peak. To ensure this, a constraint is included using the advance ratio without the wake factor, see Equation 6.1.

$$J_{w=0} = \frac{v_s}{n_p \cdot D_p} \tag{6.1}$$

Thus, as long as $\eta_O \leq \eta_{O,w=0}$, the working point of the propeller lies before the open water efficiency peak.

6.1.1. Result

It is expected that the propeller should be operated with high pitch and low rotational speed until the design pitch is reached in order to sail in the most efficient working point of the propeller. Because of this combination of high pitch and low rotational speed, the absorbed power from the engine is lower then for a propeller operated inefficiently. However, due to the low rotational speed the working point of the engine might lie too close to the power limit, thus resulting in a risk of overloading the engine.

The resulting combinator settings for the optimisation in terms of the performance indicator propeller efficiency are shown in Figure 6.3.

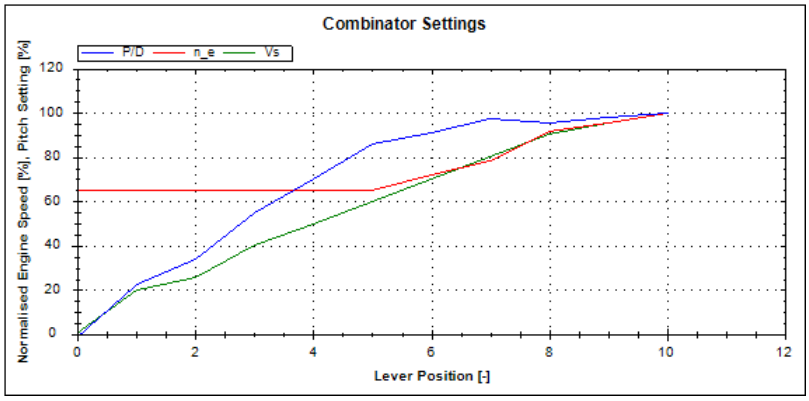


Figure 6.2: Resulting Combinator settings for Propeller Efficiency optimisation.

In this figure, the vessel speed and combinator settings are shown in terms of the pitch-diameter ratio and the engine shaft speed. In the first lever positions the engine speed remains constant in the idle speed and the pitch is increased until the fifth lever position. Then, both the speed and pitch are increased until the pitch is kept around the design pitch and the engine speed is further increased. This result is also according to the expectation of how a propeller should be operated efficiently.

Then, in Figure 6.3 the resulting propeller load is shown in the operating envelope of the engine.

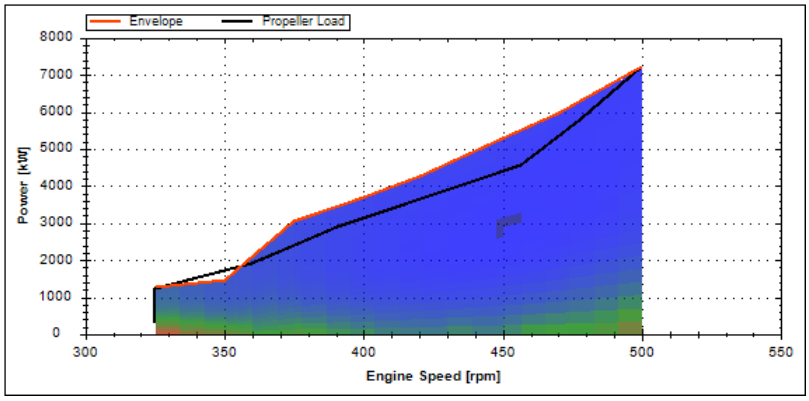


Figure 6.3: Resulting Combinator Curve for Propeller Efficiency optimisation.

Here it becomes clear that it is not possible to operate the propeller at the idle speed up to lever position 5 without overloading the engine. An engine margin can be used to prevent this, as explained in Section 5.1.1. Also for higher lever positions it must be determined whether the operating point of the engine is desirable. This result coincides with the expectation that low rotational shaft speed results in a working point of the engine that lies close to the engine power limit.

6.2. Cavitation Inception

The phenomenon of cavitation is the phase change from liquid to gas due to decreasing pressure which falls below the saturated vapour pressure. Brennen [4] explains that this manifestation has to do with the tensile strength of a liquid. When the pressure decreases such that the liquid ruptures at a roughly constant temperature, it is called cavitation [4]. As vaporisation takes place, small cavities containing vapour and gasses are formed that violently collapse as they encounter local pressure increase. This can be explained in a phase diagram of a fluid, see Figure 6.4.

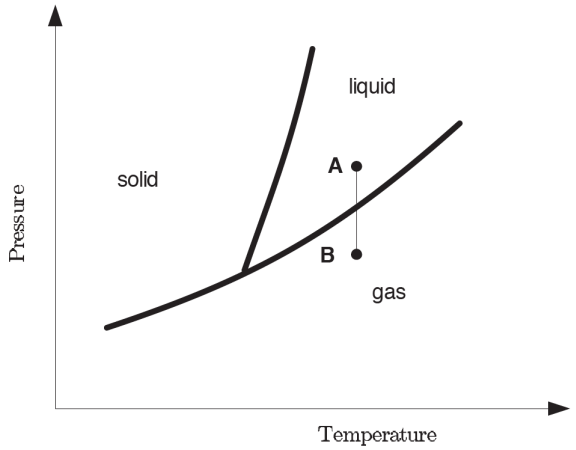


Figure 6.4: Phase Diagram. Taken from Vrijdag [49].

At constant temperature and a pressure drop from A to B it is expected that cavitation starts occurring. Vrijdag [49] explains that there are various other factors that have an effect on whether or not cavitation actually occurs. Such factors include small vapour bubbles and small fixed particles that impact the pressure at which cavitation starts. The moment at which cavitation first appears is referred to as cavitation inception. Cavitation inception is complex because it is also dependent on the boundary layer over the propeller, the place on the propeller blade and the type of cavitation [6].

In this thesis the cavitation that occurs on the propeller blades is of interest in order to calculate the pitch and speed setting such that the risk of cavitation inception is minimal. This minimal risk of cavitation inception is referred to as a measure of cavitation safety. In the field of propeller hydrodynamics cavitation behaviour is presented in a diagram that shows when and what form of inception is expected. This diagram is dependent on the operation point, the wake field and the geometry of the propeller and is different for each pitch of the blade. In Figure 6.5 an example of what such an inception diagram looks like is shown.

It shows the dimensionless cavitation number σ_n versus the thrust coefficient K_t. The basis of the equation for the cavitation number is defined by the ratio of the static and dynamic pressure head of the propeller, which is explained in more detail by Carlton [6]. A commonly used definition for the cavitation number is shown in Equation 6.2.

$$\sigma_n = \frac{p_a - p_v + \rho_{sw} g z}{\frac{1}{2}\rho_{sw} n_p^2 D_p^2} \tag{6.2}$$

Here, p_a is the atmospheric pressure, p_v the vapour pressure of seawater, ρ_{sw} the density of the seawater, g the gravitational acceleration, z the water height above the centre line of the propeller shaft, n_p the shaft speed and D_p the propeller diameter. In Figure 6.5 the operational curve is also plotted and in this case it lies in the middle of the lines that represent the region at which cavitation inception is expected.

The propeller can experience suction side cavitation which is also called sheet cavitation. Then the propeller can experience tip vortex cavitation and pressure face cavitation which is also called propeller-hull cavitation [6]. In Subsection 3.1.2 several methods to predict cavitation inception behaviour have been compared. The strategy that was considered to be most promising is the *'effective angle of attack'* method from Vrijdag [49].

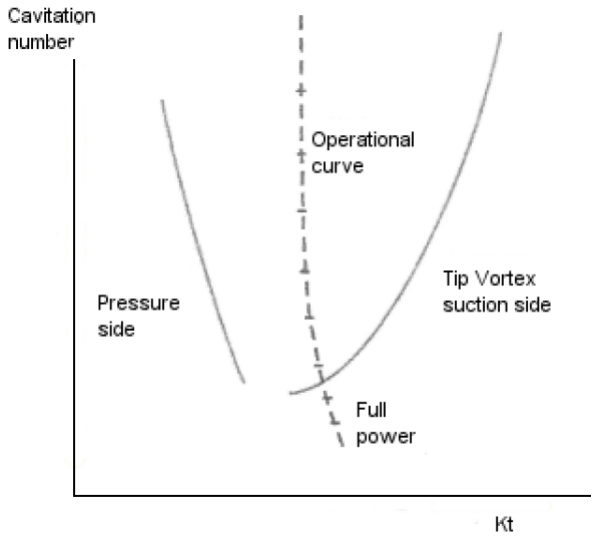


Figure 6.5: Cavitation diagram showing two inception lines. Taken from Vrijdag et al. [50].

6.3. Effective Angle of Attack Method

The strategy proposed by Vrijdag [49] is aimed at minimizing the risk of cavitation inception. This strategy is based on a method to predict cavitation inception for the development of a propulsion control system to increase the cavitation free time in operational conditions.

Vrijdag [49] first performed a review and comparison of existing predictive methods and concluded that there are no convincing arguments that cavitation inception can be predicted with the desired high level of accuracy for a variety of operations without intensive validation. Besides, the operating conditions of the vessel are not constant and the exact conditions are not known beforehand. Nevertheless Vrijdag [49] has proposed a robust method to determine and control the angle of attack that is expected to give the least risk of cavitation inception for the whole propeller.

The form of cavitation that is expected to appear first is tip vortex cavitation. For this reason the term 'effective' refers to that angle of attack at the leading edge of the propeller as this is the region where cavitation is assumed to first appear. The definition of the effective angle of attack includes the propeller geometry, the inflow angles and propeller induced velocities. The blade inflow angles are the pitch θ, the flow angle β and a correction for the shock free entry angle α_i, see Equation 6.3.

$$\alpha_{eff} = \arctan\left(\frac{P_{0.7R}}{0.7 \cdot \pi \cdot D_p}\right) - \arctan\left(\frac{c_1 \cdot v_a}{0.7 \cdot \pi \cdot n_p \cdot D_p}\right) - \alpha_i \tag{6.3}$$

As mentioned earlier, the cavitation inception diagram shown in Figure 6.5 is different for each pitch. Instead of requiring a diagram for each pitch variation, Vrijdag [49] proposes to present the inception diagram containing the cavitation number versus the effective angle of attack. The effective angle of attack must now be chosen such that if the cavitation diagrams them selves would be presented as the cavitation number versus the effective angle of attack instead of coefficient k_t (or k_q or J), the middle of these diagrams and the inception lines would overlap such that the effective angle is at the intersection of these overlapping diagrams. The overlap of these diagrams can be adjusted using the calibration coefficient c_1 such that there is maximum margin against cavitation for each operation point. Vrijdag [49] used several existing full scale cavitation inception diagrams and chose a calibration coefficient that satisfies the overlapping of the middle of the diagrams. Figure 6.6 shows the four inception diagrams that result from four different calibration coefficients. Looking at these diagrams, the ones that come closest to overlapping of the middle of the plots are those with a calibration coefficient of $c_1 = 0.7$ or $c_1 = 0.8$. In the case of this propeller a choice was made for $c_1 = 0.7$ and would result in an effective angle of attack of 4.5 to 5 degrees.

Procedure

For every lever position the pitch and speed setting are calculated using a separate calculation core that employs the definition for the effective angle of attack method. The objective is to operate the propeller in or

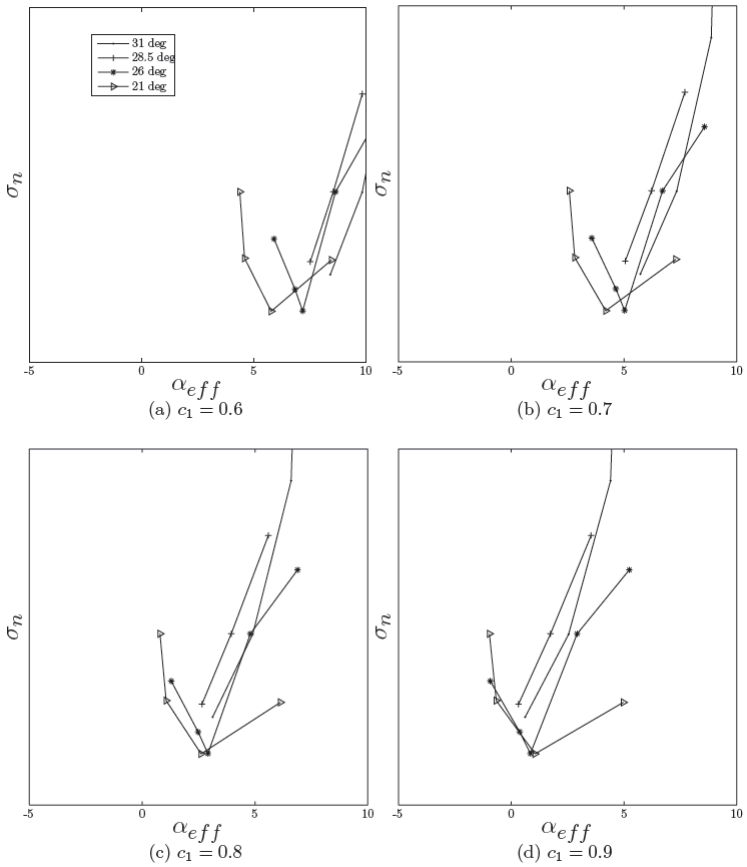


Figure 6.6: Inception diagrams based on α_{eff} for different values of c_1. Taken from Vrijdag [49]

as close as possible to the operating point where the angle of attack is expected to give maximum cavitation safety. Therefore, first this value for the effective angle of attack is determined based on the following equation:

$$\alpha_{eff,opt} = \arctan\left(\frac{P_{eff,opt}}{0.7 \cdot \pi \cdot D_p}\right) - \arctan\left(\frac{c_1 \cdot v_{a,opt}}{0.7 \cdot \pi \cdot n_{p,opt} \cdot D_p}\right) - \alpha_i \tag{6.4}$$

To determine this optimum effective angle of attack, the operating point for which this value is calculated must be determined together with the calibration coefficient. The following choices have been made:

- *Effective angle in design point*
  The optimum effective angle of attack is calculated based on the design point of the propeller:

  $$P_{eff,opt} = P_{d,0.7R} \tag{6.5}$$

  If the propeller is designed for efficiency rather than comfort or low noise, the angle of attack in the design point will be too high for low cavitation behaviour. This means that this method requires reliable (full scale) calibration data to ensure that the calculated operation point lies in the safe region to prevent cavitation inception.

- *Calibration coefficient*
  As was explained earlier the calibration coefficient or correction factor, is used to find a satisfactory overlap of the cavitation inception diagrams. To determine this factor, reliable and preferably full scale measurements of cavitation inception data is required for several pitch settings. Full scale cavitation data is however not readily available. Vrijdag [49] also mentions that unless there exists a method to estimate the cavitation inception with high accuracy before sea trials, the control input must be tested

and calibrated on board to be sure of cavitation safety. For now a calibration coefficient of $c_1 = 0.7$ is used.

The shock free entry angle is dependent on camber and induced velocities near the leading edge, but is here determined based on the camber as is done in Vrijdag et al. [50]. This part is then defined by the following definition with camber c and thickness f_{max}.

$$\alpha_{i,0} = \frac{c f_{max}}{\frac{c^2}{4} - f_{max}^2} \tag{6.6}$$

The procedure for the calculation of an optimum combination of pitch and speed for safety against cavitation is similar to the procedure for the propeller efficiency calculation core. Instead of the efficiency, the effective angle of attack is calculated for a certain propeller speed range and step size. Note, that there are now two values for the effective angle; the optimum value calculated before the iterations and the value at a certain propeller speed and pitch combination. The objective is to find the combination of propeller speed and pitch such that it is as close as possible to the optimum value for the effective angle of attack for each lever position. In order to quantify this, the absolute difference is determined by the optimum effective angle of attack minus the nth calculated effective angle of attack:

$$|\alpha_{eff,opt} - \alpha_{eff,n}| \tag{6.7}$$

Additional to storing one list for the OWC's as done for the propeller efficiency calculation procedure, there is a second list containing the absolute values of the effective angle of attack *differences*. From this list the value with the minimum difference is selected and used for further iteration. After a number of iterations, the minimum difference of the final list is then selected as the iteration for which the combination of pitch and speed has maximum cavitation safety for the respective lever position.

6.3.1. Result

The pitch and speed settings for the case study have been optimised in terms of the effective angle of attack and are visualised for each lever position, see Figure 6.7. The resulting cavitation diagram is shown in Figure 6.8. In this cavitation diagram two lines are shown. The red line represents the resulting angle of attack at each lever position. At this stage there are no reliable (full) scale cavitation inception diagrams available and therefore a fictive inception diagram (green line) is used. The middle of this inception diagram is at the optimal effective angle of attack. It can be seen that the calculated settings for the operation points of the first lever positions lie on the left and outside of the cavitation inception diagram. At lower vessel speeds the operation point of the propeller is limited by the idle speed of the engine. The pitch is increased while the shaft speed remains constant. As soon as the engine speed is increased for higher thrust demand at a faster ship speed, the operation points near the optimal value for the effective angle of attack. For the higher lever positions the effective angle of attack remains near the optimal value but increases as the vessel speed increases. In the cavitation diagram this can be observed as the red line lies more to the right side of the inception diagram. At the last lever position the operation point of the propeller is limited by the maximum pitch ratio and maximum engine speed such that the difference between the optimal value and the resulting effective angle of attack increases.

In Figure 6.9 the propeller load for cavitation safety is shown in the operating envelope of the engine.

Note, that it is the same result as was observed for propeller efficiency in Figure 6.3. Because the optimum effective angle of attack was chosen in the design point of the propeller, the resulting pitch is relatively high for each lever position.

6.3.2. Validation

In order to investigate the resulting operating points for cavitation behaviour the propeller performance must somehow be evaluated. Currently it is not standard procedure to perform cavitation tests during sea trials. Thus, apart from the data from the research of Vrijdag [49], full scale cavitation behaviour data for various CPPs for several operating points is not available. This means that another approach must be taken to be able to evaluate the propeller performance. Another way to obtain cavitation inception measurements is from model tests for several pitch and speed settings. This is however costly and time consuming if it must be done for each new propeller design. At this stage of the CCG yet another way must be found to evaluate cavitation behaviour. The propeller design can be used to indicate the expected cavitation behaviour of the

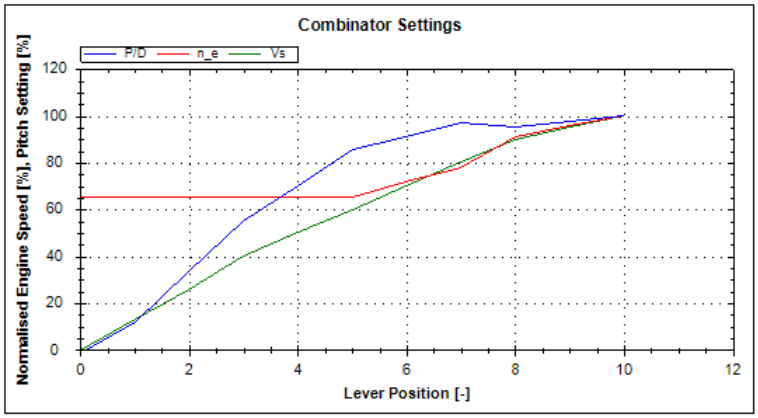


Figure 6.7: Combinator Settings - Cavitation Safety

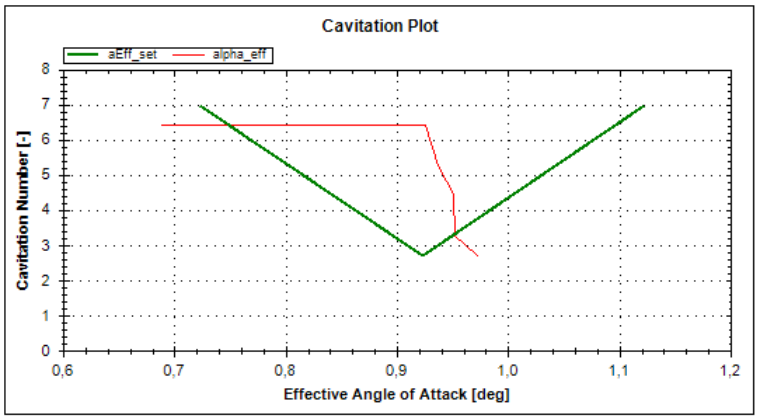


Figure 6.8: Cavitation Diagram - Cavitation Safety

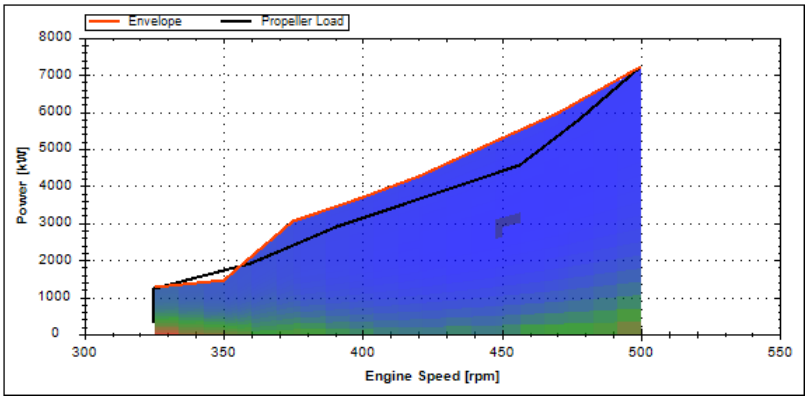


Figure 6.9: Engine Envelope - Cavitation Safety

propeller for the calculated combinator settings. For this purpose software is used that calculates the cavitation inception behaviour for one turning cycle of a propeller in a certain operation point. The resulting output of this software for various operating points can be found in Appendix A. For one lever position ($LP = 7$), the results are shown here in Figure 6.10 to discuss in more detail what can be concluded from it.

The left result shows the cavitation inception behaviour for the operation point that was originally designed for this propeller and on the right the results are shown for the newly optimised operation point based on the *effective angle of attack* method, both for lever position 7. The result shows one blade of the propeller at several instances of a turning cycle. On the blade section a purple colour is used to indicate suction side cavitation on the blade. Also, the cavitation inception diagram that belongs to the respective pitch setting of

Figure 6.10: Cavitation inception results $LP = 7$, left: result for original operation point, right: result of calculated operation point using the *effective angle of attack* method - Cavitation Safety

the blade is shown in terms of the cavitation number versus the advance ratio (σ_n - J). Note that here the red line indicates the suction side inception and the blue line indicates the pressure side inception. In the cavitation inception diagram the operation point is indicated with a cross symbol.

For validation of the implementation of the *effective angle of attack* method we will look at the result on the right of Figure 6.10. There are several aspects to consider. Firstly, as the optimum value for the effective angle of attack was chosen in the design point of the, for efficiency designed propeller, cavitation inception was expected. In the figure this can indeed be observed from the purple colour at the leading edge of the blade in the upper half of the turning cycle. Secondly, the calibration coefficient was set to $c_1 = 0.7$, which is the same value that was chosen for the propeller in the research of Vrijdag [49]. The purpose of the calibration factor was to achieve a satisfactory overlap of the cavitation inception diagrams. The optimum value for the effective angle of attack is then located at the intersection of these overlapping diagrams. The results show that, for the range of lever position $5-8$ the operation point lies close to or in the middle of the cavitation inception diagram, see Appendix A. In this case the calibration coefficient seems to already be close to a satisfactory overlap.

6.3.3. Discussion

It can be concluded that the *effective angle of attack* method gives expected results regarding the cavitation behaviour when it is determined based on the design pitch of the propeller. And even though the calibration coefficient was assumed beforehand, the operation points lie closer to the middle of the inception diagrams as the pitch increases. At this stage however, the *effective angle of attack* method does not ensure that calculated combinator settings result in safety against cavitation inception for propellers that are designed for propeller efficiency. As mentioned earlier, the effective angle of attack was calculated based on the angle of attack in the design point of the propeller. It was also mentioned that (full scale) calibration data is necessary in order to calibrate the effective angle such that cavitation safety can be ensured.

This calibration can be done by adjusting the calibration coefficient. The calibration coefficient is used in the first place to find a satisfactory overlap of the cavitation inception diagrams. Satisfactory in this context means that the inception diagrams overlap such that there is minimal risk against cavitation inception in case of pressure and suction side cavitation. Vrijdag [49] concludes that the definition for the angle of attack only partly satisfies the desired properties to predict the cavitation inception due to the lack of quality of the used full scale data and the definition for the angle of attack itself. He mentions that local flow disturbances, propeller geometry details and inception locations are not taken into account and therefore cause deviation of the predicted effective angle. This means that without reliable calibration data this definition cannot ensure reliable pitch and speed settings with safety against cavitation inception. The challenge is to calibrate the definition for the effective angle of attack such that this method is reliable and can still be used without taking away of its robust character.

The result of cavitation behaviour for the originally determined operation point is shown on the left in Figure 6.10. When comparing this result with the result from the operating point using the *effective angle of attack* method the operating point deviates from the middle of the inception diagram and lies much closer to the pressure side inception line (blue). As a result the vortex cavitation indicated by the purple colour on the

blade is significantly less. Research on the basis of such data experiments can give insight how these results can be used to adapt either the definition for the effective angle of attack or the calibration coefficient. Such research is out of the scope of this thesis, but could improve the *effective angle of attack* method such that it becomes a more reliable method for a broader range of propeller designs, and perhaps even before full scale calibration.

At this stage it is chosen to use the insight from the results to develop an alternative approach to optimise combinator settings such that there is minimal suction side cavitation and sufficient margin against pressure side cavitation. This will be presented in the next sections.

6.4. Cavitation Inception Prevention Method

The approach described in this section is named the *cavitation inception prevention*(CIP) method and is an alternative approach to the *effective angle of attack* method. In this method it is aimed to optimise combinator settings such that there is minimal suction side cavitation and sufficient margin against pressure side cavitation.

For this approach the software mentioned before is used with which inception diagrams can be calculated for a number of operation points. The resulting cavitation inception diagrams each belong to a certain pitch setting. For the CIP method data of these inception diagrams for different pitch settings are collected and used. From the resulting inception diagrams data points are collected in terms of the pressure and suction side cavitation inception line as shown in Figure 6.11.

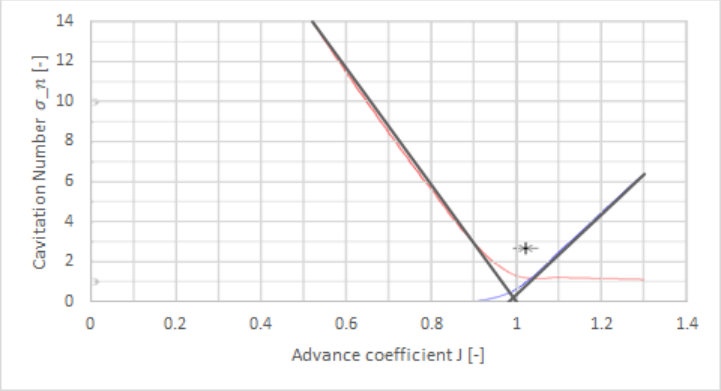


Figure 6.11: Cavitation inception diagram with pressure line (blue), suction line (red) and overlapping linear lines (black).

In this figure the inception diagram is shown in terms of the suction side (red line) and the pressure side (blue line). Overlapping the inception diagram are two linear black lines. The black linear pressure and suction inception lines are each defined by two data points (x_i, y_i) in terms of the cavitation number and the advance ratio. Note that the base of the red and blue line are not taken into account with this approach. This means that back bubble cavitation inception in this region is not accounted for. However, it is expected that tip vortex cavitation is first to appear. This results in a new data set containing cavitation inception data for a range of pitch settings of the benchmark propeller, see Table 6.1.

Table 6.1: Representation of cavitation inception data.

P/D [-]	J [-]	σ_n [-]
1.229	*pressure line* (x_i, y_i)	
	0.98	0
	1.3	6.4
	suction line (x_i, y_i)	
	0.52	14
	1	0
...	...	...

By means of interpolation a cavitation inception diagram can be determined for a certain P/D ratio.

Procedure

In order to achieve sufficient margin against pressure side cavitation and minimal suction side cavitation, the combinator settings are optimised such that the working point of the propeller lies as close to the pressure side inception line as possible. The procedure for this optimisation method follows the same steps as for the method to determine the optimal combination of pitch and speed setting in terms of propeller efficiency. For each iteration step the required thrust is determined at a certain rotational speed together with the advance ratio, the torque coefficient, the OWC and respective pitch- diameter setting.

For the CIP method two aspects must be determined. Firstly, it must be checked if the resulting operating point for the current iteration step and respective rotational speed lies inside the inception diagram belonging to the pitch- diameter ratio. Secondly, the distance from the operational point to the pressure side inception line is determined in terms of the advance ratio.

In order to ensure sufficient margin against pressure side cavitation it is chosen to determine the distance of the operation point to the interpolated pressure side inception line in terms of the advance ratio without the wake fraction and using a constant parameter $c > 1$ to include a margin for the pressure side cavitation, see Equation 6.8.

$$J_{w=0} = c \cdot \frac{v_s}{n_p \cdot D_p} \tag{6.8}$$

If the operation point lies within the interpolated inception diagram, the *distance* to the pressure side inception line as well as the OWC belonging to the current pitch- diameter setting are stored within a list. From the list of distances the value with the minimum distance to te pressure side cavitation inception line is selected and used for further iteration. After a number of iterations, the optimum combination of pitch and rotational speed is determined by the minimum distance to the pressure side inception cavitation line, selected from the final list.

6.4.1. Result

For the CIP method the pitch and speed settings have been optimised and visualised for each lever position such that there is minimal suction side cavitation and sufficient margin against pressure side cavitation, see Figure 6.12.

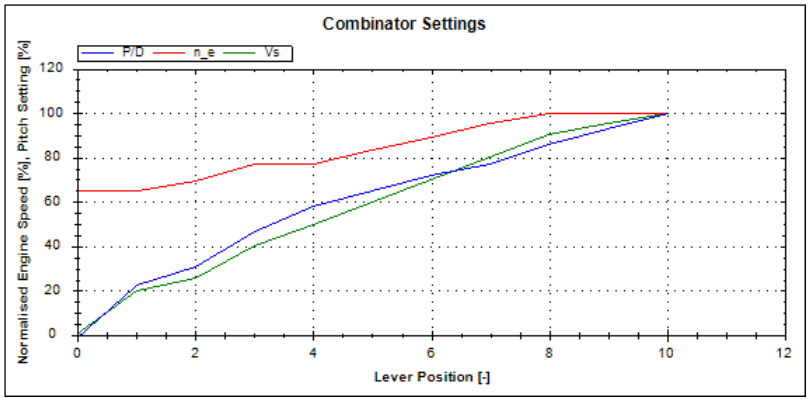


Figure 6.12: Combinator Settings - Cavitation Safety

Compared to the former results it can be observed that the rational speed is increased earlier and the pitch settings are decreased significantly. It is likely that risk of cavitation inception decreases as the pitch is decreased because the load on the tip of the propeller blade also decreases.

Further, instead of a fictive cavitation inception diagram that represents the combined overlapping inception diagrams for a number of pitch settings, there now exists an interpolated cavitation inception diagram for each resulting pitch setting. For each lever position the interpolated cavitation inception diagram is shown together with the corresponding operation point in Figure 6.13.

Note, that the operation points are shown in terms of the cavitation number and the advance ratio *with* the wake factor. For lever positions 2-7 the operation points lie within the interpolated inception diagram. The operation points at lever position 1, 8, 9 and 10 the operation point do not lie within the interpolated

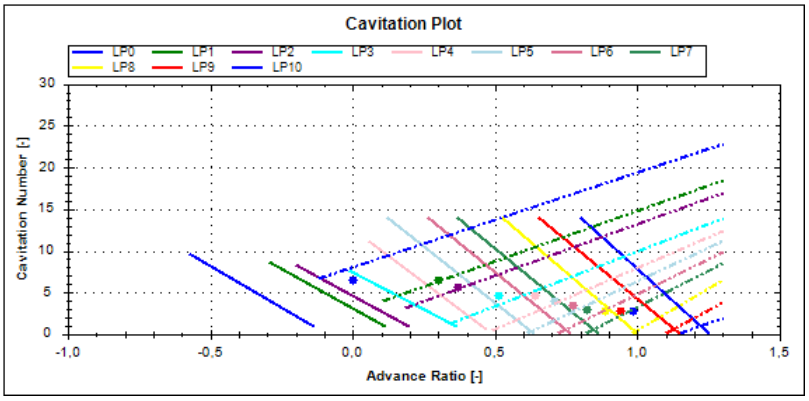


Figure 6.13: Cavitation Diagram - Cavitation Safety

inception diagram due to engine limitations. This results in pressure side cavitation for lever position 1, and suction side cavitation for lever position 8, 9 and 10. These deviations can be explained with the load curve within the operating envelope of the engine, see Figure 6.14.

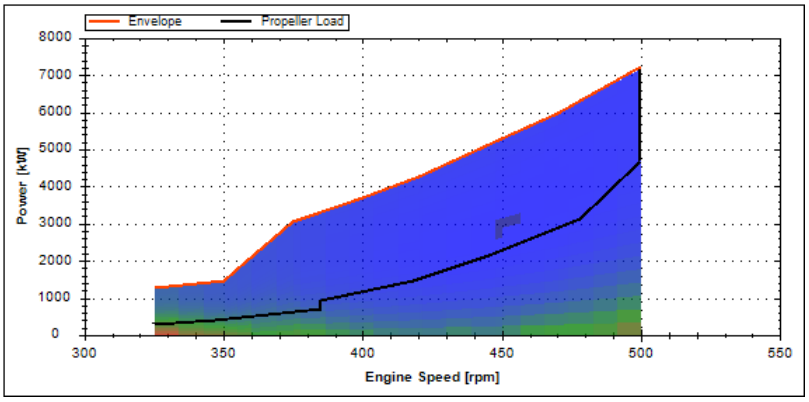


Figure 6.14: Engine Envelope - Cavitation Safety

In the case of the first lever position the operating point is limited by the idle speed of the engine and the pitch has to decrease. For lever position 8, 9 and 10 the operating point is limited by the maximum engine speed. As the speed can not be increased, the pitch must increase in order to sail at higher vessel speeds.

6.4.2. Validation

To investigate the resulting operating points for cavitation behaviour, the propeller performance is evaluated with the software that was introduced earlier. For lever position 4-10 the cavitation inception diagrams have been calculated, these can be found in Appendix A. For lever position 7 the results are shown in Figure 6.15.

The left result shows the cavitation inception behaviour for the operation point that was calculated based on the *effective angle of attack* method. On the right the results are shown for the CIP method where the combinator settings have been optimised based on the distance from the pressure side cavitation inception line. It can be observed that the operating point lies close too or on the pressure side inception line for lever positions 4-7. Because the operating points here represent the working points without the wake factor, the operating point of the ship, it appears that the margin against pressure side cavitation is not sufficient. However, if the operating point with the wake factor is taken into account, the advance ratio decreases slightly and will be positioned left with respect to the pressure inception line. Together with uncertainties due to the diagram generation used in the software, it is expected that the margin is sufficient. For the lever positions 8-10 the operation points deviate and move more to the suction inception line. The result in terms of cavitation behaviour of the new combinator settings can be seen from the purple colour on the blade section. The cavitation behaviour is significantly less with the combinator settings that are optimised using the CIP method.

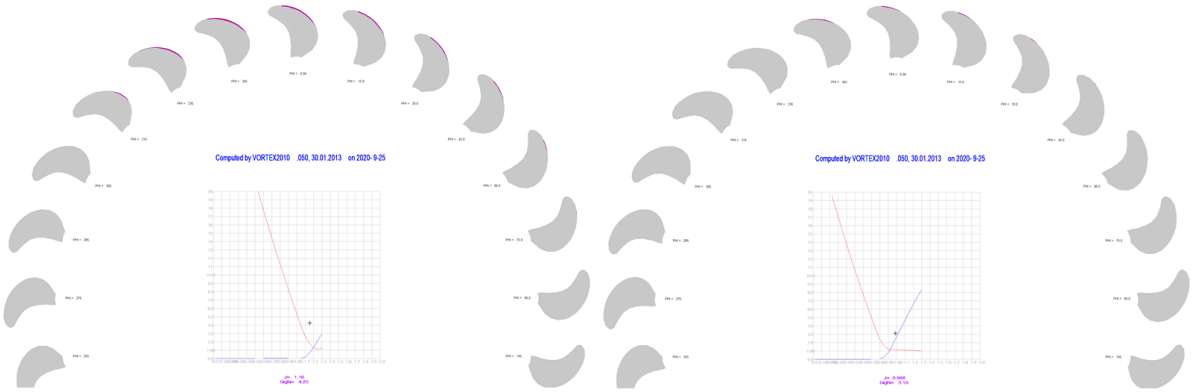


Figure 6.15: Cavitation inception results $LP = 7$, left: result for operation point using *Effective Angle of Attack*, right: result of calculated operation point using the *CIP* method - Cavitation Safety

6.5. Conclusion

In this chapter optimisation approaches have been described for the performance indicators propeller efficiency and cavitation inception. For each optimisation approach, the resulting optimised combinator settings are shown and discussed.

Propeller Efficiency

The developed approach for the optimisation of a combinator curve for propeller efficiency is based on the open water efficiency of the propeller OWCs. The resulting combinator curve shows expected behaviour and delivers the highest possible propeller efficiency for all lever positions. For lever positions at low vessel speeds, the resulting combinator curve shows increasing pitch at a constant idle speed until the limit of the operating envelope is reached. At the lever positions corresponding to higher vessel speeds, the pitch is further increased together with slightly increased shaft speed. Due to the combinator settings of high pitch and low rotational shaft speed, the working point of the engine lies close to the power limit and thus might risk overloading of the engine. This can be prevented by choosing a satisfactory engine limit.

As the resulting combinator curve shows expected behaviour it is concluded that this approach can be used for optimisation and evaluation in terms of propeller efficiency.

Cavitation Safety

The development of the approach to limit the risk of cavitation inception started with the implementation of the *effective angle of attack* method proposed by Vrijdag [49]. In order to validate the resulting combinator curve, the combinator settings have been evaluated with a software program that calculates the cavitation inception behaviour for one turning cycle of the propeller. From these results it was observed that the implemented approach did not result in combinator settings such that the risk of cavitation inception is limited. On the contrary, the results showed that there was increased vortex cavitation compared to the originally designed combinator settings for the respective propeller.

The reason for this result is because the *effective angle of attack* was determined based on the design pitch of the propeller. As the respective propeller is designed for propeller efficiency, the resulting angle of attack is too high to prevent cavitation inception. To ensure minimum risk of cavitation inception, a calibration coefficient exists with which the effective angle is to be calibrated. The value for this coefficient was set to an arbitrary value at first. Interestingly however, the requirements for the calibration coefficient seemed to already be met when observing the resulting operating points of the propeller.

Currently there is no way to validate or calibrate the *effective angle of attack* on the basis of full scale cavitation measurements. Besides, it is not clear how this method can be used for propellers that are designed for propeller efficiency by tuning the calibration coefficient in a different way, than based on full scale measurements. Therefore, it was concluded that the robust *effective angle of attack* method can not be used as an approach to optimise and evaluate a combinator curve to ensure minimum risk of cavitation inception.

Because of this an attempt is made to develop an alternative approach, the CIP method. The results from the evaluated combinator settings based on the *effective angle of attack* method gave insight into the cavitation behaviour. In comparison to the operating points resulting from the *effective angle of attack* method, the operating points of the originally designed combinator curve lie closer to the region where it is expected that pressure side cavitation occurs.

Based on this observation a second approach has been developed where the combinator settings are optimised in terms of the advance ratio such that there is minimum suction side cavitation and sufficient margin against pressure side cavitation. The resulting combinator settings have been evaluated with the software program that calculates the expected inception behaviour. This time, it was observed that the cavitation behaviour was similar and slightly better than the cavitation behaviour of the originally designed combinator curve. Thus, the CIP method can be used to optimise and evaluate combinator curves in order to limit cavitation inception.

7

Engine Performance Indicators

In the former chapter optimisation approaches for performance indicators that have to do with the propeller have been addressed. In this chapter engine performance is addressed in terms of engine efficiency and fuel consumption per unit of time. First, it is explained how a general SFOC map for diesel engines is obtained. Then, the approach and procedure to optimise a combinator curve in terms of engine efficiency and fuel consumption are explained. Subsequent to the optimisation in terms of fuel consumption per unit of time, the approach and procedure of the simulation is explained. Finally, the chapter is concluded.

7.1. General SFOC Contour Map

Both performance indicators are dependent on the specific fuel consumption of the engine. This data is often presented in the form of a contour plot referred to as an SFOC map. This SFOC map is specific for each engine and if available, its data can be used in the CCG in the format described in Subsection 4.3.4. It might be the case that data about the specific fuel consumption is available for the main diesel engine, but depending on the manufacturer this information will not always be readily available. Therefore the CCG is supplied with a general SFOC map that can be scaled and used for the main engines if they are diesel engines. At this stage of the CCG the general SFOC map is only available for a diesel engine, but it is possible to expand the library with data from general fuel maps that belong to gas or diesel generator engine types.

The SFOC map used for the CCG was extracted from the zero order MOSSEl model introduced in Subsection 3.3.1. This model makes use of an algorithm for the diesel engine where one can adapt several parameters such as nominal power and nominal shaft speed. The engine model behaves like a 'rubber' engine that is mapped onto a standard diesel engine. Because of this, the SFOC map that can be extracted after running the model, will have a similar shape, no matter what parameters are changed. Thus, the default settings of the engine model are used and resulting SFOC data is further used and adapted such that it is in the correct format as described in Subsection 4.3.4. The operating envelop together with the SFOC contours are shown in Figure 7.1.

The black line in the figure represents the model engine operating envelope, and the blue and red line represent the propeller load for a propeller law and generator law respectively. The data that is used from this model are the contour lines that represent the SFOC in relation to the operational envelope, similar to the approach in case the SFOC map of an engine is available, as explained in Subsection 4.3.4. This data can now be used for the optimisation algorithms in order to optimise a combinator curve in terms of engine performance indicators.

7.2. Engine Efficiency

In this section the optimisation approach for the objective to determine the combinator settings blade pitch and shaft speed such that the working point of the engine is most efficient, is explained. Engine efficiency relates the work output W_e to the heat input Q_f [25], see Equation 3.3. The amount of fuel injected is commonly expressed as the specific fuel consumption *sfc*, which is defined by the fuel consumption $\dot{m}_f$ of the engine related to the brake power P_B, see Equation 7.1.

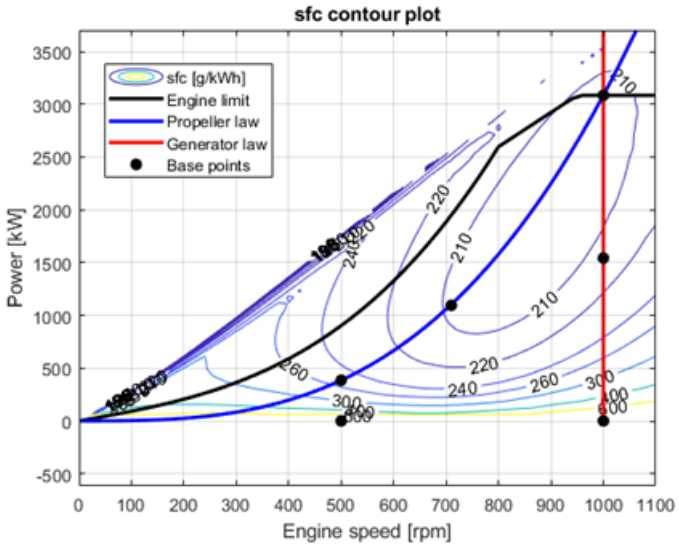


Figure 7.1: SFC contour plot from MOSSEL model.

$$sfc = \frac{\dot{m}_f}{P_B} = \frac{m_f}{W_e} \tag{7.1}$$

It is common to express engine efficiency such that the units are not SI units, but *sfc* in [g/kWh], P_B in [kW] and h^L in [kJ/kg]. Using Equation 3.3 and 7.1 the engine efficiency can be rewritten as a function of the specific fuel consumption and the lower heating value, see Equation 7.2.

$$\eta_e = \frac{3600000}{sfc \cdot h^L} \tag{7.2}$$

This means that engine efficiency increases with decreasing specific fuel consumption. To find the optimum combinator settings in terms of engine efficiency is similar to the approach taken for the optimisation for propeller efficiency. In the next paragraph, this procedure is explained in more detail.

Procedure

The procedure for this optimisation approach is similar to that of the optimisation approach for the propeller efficiency. In order to limit the number of calculations, the calculation steps are performed for a number of iterations while narrowing down the rotational speed interval at each iteration. For each iteration, the rotational speed is varied and the respective pitch is determined to achieve the thrust demand at the lever position and vessel speed. For each combination of pitch an rotational speed, the required engine power is determined as well. Using interpolation functions, the specific fuel consumption for the combination of engine power and shaft speed is determined and stored in a list. Finally, from this list, the combinator settings that result in the lowest value of specific fuel consumption is chosen as the optimum setting.

To conclude, the procedure that is done within each iteration and for each speed step is the following:

1. Determine the thrust demand to overcome the vessel speed at the current lever position via the *ResistanceItem* object.
2. Calculate the advance coefficient using the current propeller speed and Equation 2.6.
3. Calculate the thrust coefficient using Equation 2.7.
4. Determine the pitch and OWC by means of interpolation as a function of $owc(j, kt)$.
5. Determine the torque coefficient from the OWC.
6. Determine the required engine power using Equation 2.13.
7. Determine the specific fuel consumption by means of interpolation as a function of $sfc(n_e, P_B)$

Constraints

Not all resulting specific fuel consumption values for the calculated combinations of pitch and speed are added to the list. Besides this, the pitch and speed settings are dependent on the presence of a PTI or PTO.

Before it is checked whether to add an sfc item to the list, the PTI or PTO is considered.

In case of a PTI, the following is considered:

- If the required power is higher than 90% of the power limit, the maximum power used from the main diesel engine is set to 90%. The rest of the power requirement is taken up by the PTI. With regards to engine efficiency, only the fuel consumption from the main driving engine is taken into account.

In case of a PTO, the following is considered:

- The required propulsion power is added to the PTO, from which the required power is assumed constant. At the same shaft speed and the combined power, the specific fuel consumption is determined by means of an interpolation function.

Finally, the following constraints determine whether a resulting specific fuel consumption item is added to the list:

- pitch- diameter setting ≤ maximal pitch- diameter setting
- required brake power < brake power limit

7.2.1. Result

The resulting combinator settings for the blade pitch and rotational speed for a combinator curve that is optimised in terms of engine efficiency, are shown for each lever position in Figure 7.2.

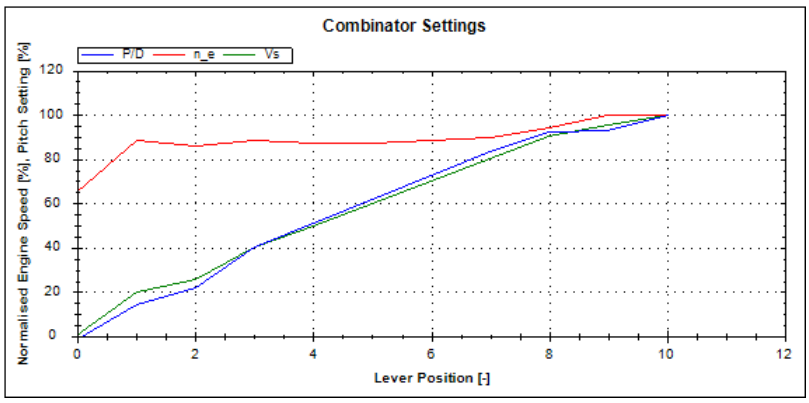


Figure 7.2: Combinator Settings - Engine Efficiency

These combinator settings belong to an engine with the general SFOC map. From lever position 1 through 7 the shaft speed seems almost constant and relatively high. The pitch setting is increased from lever position 0-10 to deliver the thrust demand in order to reach the ship speed demand. The corresponding propeller load curve within the operating envelope of the engine is shown in Figure 7.3.

In the figure the colour scheme represents the SFOC map where the colours blue-green-red represent low, medium and high values for specific fuel consumption respectively. The combinator settings correspond with a propeller load curve in the region where the specific fuel consumption is lowest. At lever position 0 however, the ship speed demand is equal to 0 knots and the pitch and speed both have to be decreased to their minimum setting. As soon as the ship speed demand is increased the shaft speed is increased and the pitch kept relatively low, resulting in a low demand from the engine, but still in the blue region. As the pitch is further increased the propeller load curve remains in the region with the lowest specific fuel consumption values.

As was earlier explained, it is possible to add a SFOC map from a different engine than the general SFOC map. This different SFOC map can be chosen and used for any engine instead of the general SFOC map from the MOSSEL model. The results of the propeller load within the same operating envelope is shown in Figure 7.4.

The SFOC map used here belongs to the engine envelope shown in Figure 4.5, and it can be seen that the coloured SFOC map corresponds to the contours with the blue regions containing the lowest specific fuel consumption values. The resulting propeller load is now mostly in the blue regions where it is expected that the specific fuel consumption is lowest and thus engine efficiency is highest.

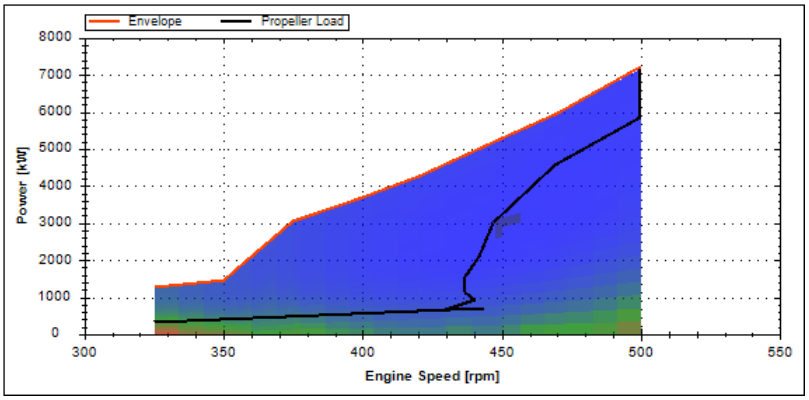


Figure 7.3: Resulting propeller load using general SFOC map - Engine Efficieny. The dark colour blue around 450 rpm is shown to indicate the minimum value of specific fuel consumption of the general SFOC map.

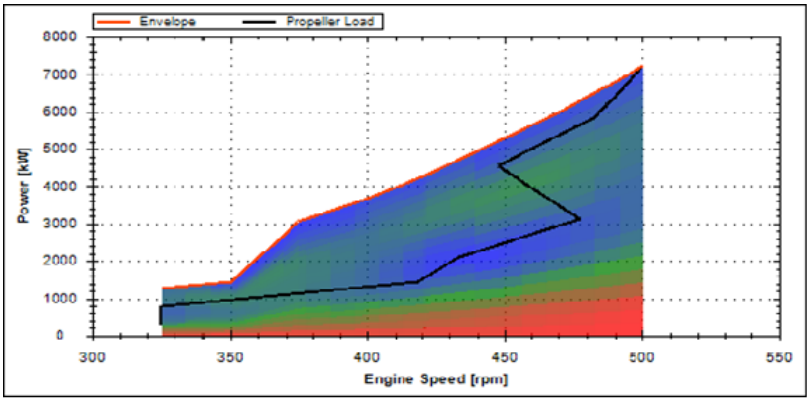


Figure 7.4: Resulting propeller load using engine specific SFOC map - Engine Efficieny.

7.3. Fuel Consumption

In this section the optimisation approach with the objective to minimize fuel consumption per unit of time is explained. The fuel consumption per unit of time can be determined by multiplying the specific fuel consumption by the required brake power of the engine, see Equation 7.3.

$$\dot{m}_f = sfc \cdot P_B \tag{7.3}$$

To find the optimum combinator settings in terms of fuel consumption per unit of time is similar to the approach taken for the optimisation for engine efficiency, with a few differences in the procedure and constraints.

Procedure

The procedure for the optimisation of a combinator curve in terms of fuel consumption per unit of time is equal to that of the procedure explained in the section concerning engine efficiency, with one extra step:

8. Determine the fuel consumption by multiplying the interpolated specific fuel consumption by the required engine power.

Constraints

Before the constraints are considered in order to determine whether or not the current resulting fuel consumption item is stored for the selection procedure, the availability of a PTI or PTO is considered.

In case of a PTI, the following is considered:

- If the required power is higher than 90% of the power limit, the maximum power used from the main diesel engine is set to 90%. The rest of the power requirement is taken up by the PTI. For the estimation of the total fuel consumption of both the main driving engine(s) and the generator engine(s)

that drive the PTI, the fuel consumption of the generator(s) are estimated. For the estimation of the fuel consumption by the diesel generator, an efficiency of 95% and the engine running at 80% power is assumed. With interpolation through the data from Figure 3.2 the specific fuel consumption is determined [25].

In case of a PTO, the following is considered:

- The required propulsion power is added to the PTO, from which the required power is assumed constant. At the same shaft speed and the combined power, the specific fuel consumption is determined by means of an interpolation function. Finally, the fuel consumption per unit of time is determined using the combined required brake power from the engine.

Finally, the following constraints determine whether a resulting fuel consumption item is added to the list:

- pitch- diameter setting ≤ maximal pitch- diameter setting
- required brake power < brake power limit

7.3.1. Result

The pitch and rotational speed settings have been optimised in terms of fuel consumption and shown in Figure 7.5.

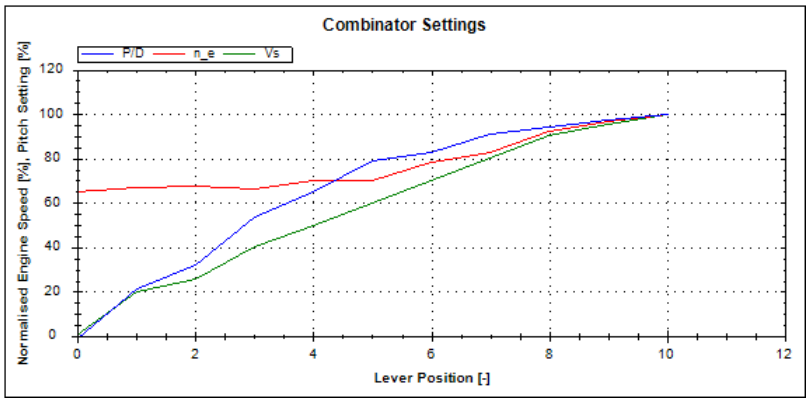


Figure 7.5: Combinator Settings - Fuel Saving

The combinator settings are different from the combinator settings optimised for engine efficiency. Instead of a relatively high, and more or less constant shaft speed, it increases slowly up till lever position 5 and faster from lever position 5 up till lever position 10. The pitch settings do not increase with the same gradient as the vessel speed, but increases steadily up till the last lever position.

In Figure 7.6, the operating envelope is shown together with the propeller load and a colour map that represents the fuel consumption per unit time. The colours blue, white and red indicate a relatively low, medium and high amount of fuel consumption respectively.

When the combinator curve is optimised for fuel consumption per unit time, the propeller load lies more close to the power limit of the envelope compared to the combinator settings for engine efficiency for this engine and the general SFOC map. In Subsection 3.1.3 it was concluded that in order to sail the vessel at most fuel efficient mode, the propeller should be operated according to the cubic propeller load curve. One of the calculation options to set up a combinator curve was called the default mode. In this mode, the propeller is not fully operated with a constant pitch, but for as long as the limits of the machinery are not exceeded. The fuel consumption results for the default mode are compared to the optimal results to see whether this expectation of optimal fuel consumption for a combinator curve mostly operating with constant pitch also holds in the CCG. These results can be evaluated with Figure 7.7.

Here, the resulting fuel consumption in ton per hour versus the lever positions is shown for two combinator curves. The red line shows the results for the combinator settings optimised in terms of fuel consumption per unit of time and the blue line shows the results for the combinator curve calculated with the default mode. The lines completely overlap, except for the result at lever position 8 which is most likely due to the constraints. This indicates that indeed the default mode can also be considered when designing an optimised combinator curve in terms of fuel consumption per unit time.

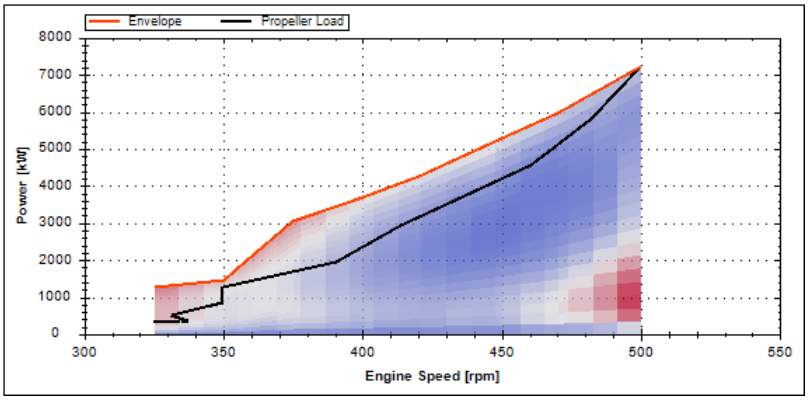


Figure 7.6: Engine Envelope - Fuel Saving

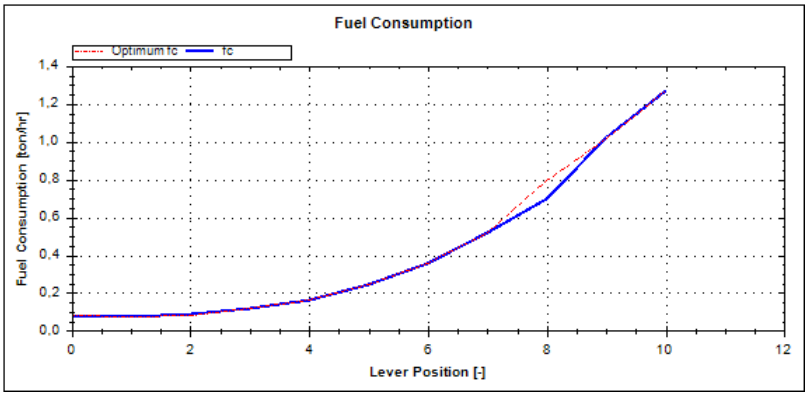


Figure 7.7: Fuel Consumption

7.4. Trip Simulator

With the optimisation method in terms of fuel consumption per unit time, a combinator curve can be designed for optimal fuel consumption. Besides this, any other combinator curve design can be evaluated using these optimal results. With the combinator curve result data, the fuel consumption in tonnes per mile at each lever position and respective ship speed demand is calculated. Besides fuel consumption related to ship speed, it is important to take into account the operational profile of a ship. For a certain trip from port A to port B, the ship sails at different speeds in port, in canals or at open sea for a certain time or distance. Furthermore, it is important to take into account not only the fuel consumption due to propulsion, but also the fuel consumption due to the hotel load at different stages of the trip. For this reason, the CCG has been expanded with a basic simulator where the operational profile of a vessel can be taken into account in order to evaluate the total fuel consumption for a certain trip from port A to port B. In the next paragraphs the procedure and results for this trip simulation is explained.

Procedure

For a trip simulation in the CCG, the combinator settings from the combinator design that was last calculated are used to determine the combinator settings at the lever positions as well as the fuel consumption at a certain vessel speed demand. Because the simulator is a tool on its own, it can be used for any combinator curve, and by default the lastly designed combinator design is used. The input and result parameters for the calculation procedure of the simulator have been introduced in Section 4.5. These include input parameters for different parts of a trip and parameters about the ship and the propulsion which are required for the acceleration and deceleration behaviour of the vessel. Furthermore, result parameters include the accumulated time, vessel speed, accumulated distance, propulsion and hotel load, propulsion settings and the resulting accumulated fuel consumption.

The calculation for the evaluation of the total fuel consumption of a single trip is divided in three parts namely, the ship in standby in a port, the ship manoeuvring in port and the ship cruising from port A to port B.

In order to evaluate total fuel consumption and trip duration for different cruising speeds, the trip simulation is performed for several lever positions, namely lever position 5 up to the maximum lever position. The sequence of one trip simulation is made up of the following steps:

1. Standby in port A
2. Acceleration from standstill to manoeuvre speed
3. Manoeuvre in port A
4. Acceleration from manoeuvre to cruising speed
5. Cruise
6. Deceleration from cruise speed to manoeuvre speed
7. Manoeuvre in port B
8. Deceleration from manoeuvre to standstill

- **Acceleration and Deceleration**
  The acceleration and deceleration of a vessel are dependent on the inertia which is taken into account by a percentage of the displacement of the vessel, the surge added mass. Furthermore, the method for the acceleration and deceleration is modelled by increasing or decreasing the pitch and rotational speed settings from the current combinator setting to the combinator setting belonging the next lever position within a certain amount of time. This time is called the pitch ramp-up time.

- **Standby in Port**
  The first part of a trip starts in port. The input parameters for this procedure are the time duration and the hotel load. In standby mode, the vessel speed is zero and the combinator settings from lever position zero of the lastly designed combinator curve determine the pitch en rotational speed setting of the propulsion system. Besides the combinator settings for a certain lever position, the fuel consumption data is known. Using this information, the fuel consumption due to propulsion can be determined for the duration that the ship is in standby mode. Furthermore, the fuel consumption as a result of the hotel load is estimated, using a constant value for the specific fuel consumption and a generator efficiency of 0.95%.

- **Manoeuvre in Port**
  As the vessel leaves its place and moves out of the port, it sails at a certain ship speed and it takes a certain amount of time for the vessel to leave the port. This part of the trip is simplified and split in two parts. The first part is the acceleration from stand still up till a certain constant vessel speed with which the vessel sails while still in port. The second part regards the ship sailing at a constant vessel speed until it has left the port. The input parameters are the expected duration from standstill until the vessel leaves the port, the lever position in which the vessel will sail while in port and finally, the expected hotel load during this trip part. For the part where the ship is at constant speed, the fuel consumption is determined with information about the duration minus the acceleration or deceleration time, the combinator settings and other information about the lever position and the hotel load.

- **Cruise**
  As soon as the ship leaves port, it starts to accelerate up till a certain lever position, after which the ship sails at this constant lever position for as long that it takes to cover the distance from port A to port B. The input parameters for this part of the trip are the distance from port A to port B and the hotel load during this part of the trip. This part is split into three namely, acceleration from manoeuvring speed to cruise speed, cruising at constant speed and finally deceleration from cruising speed to manoeuvring speed. The distance for which the vessel sails at a constant speed depends on the acceleration and deceleration distance.

7.4.1. Result

The trip simulation has been performed for the optimised combinator curve in terms of fuel consumption per unit time. The input data for the simulator is shown in Figure 4.22. In Figure 7.8 the trip results are shown in terms of accumulated fuel consumption up till the total fuel consumption at the end of the trip.

In this graph the accumulated fuel consumption is shown versus the time in minutes. The accumulated fuel consumption is shown for different for cruising speeds namely, for vessel speeds at respective lever positions 5 through 10. When sailing at a lower cruise speeds, the trip duration becomes longer but the accumulated fuel consumption decreases significantly. Depending on the maximum trip time, it is economically and environmentally beneficial to sail at a lower ship speed.

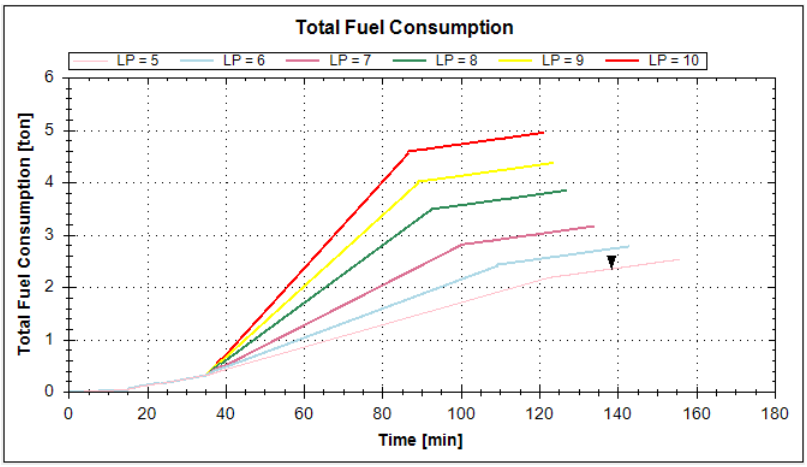


Figure 7.8: Simulator User Interface

The resulting simulation data is presented in other graphs as well. These graphs include the fuel consumption in [ton/hr], the propulsion and hotel load in [kW], the vessel speed [kn] and the accumulated total distance [km], each versus time [min]. The latter two are shown in Figure 7.9 and 7.10 respectively.

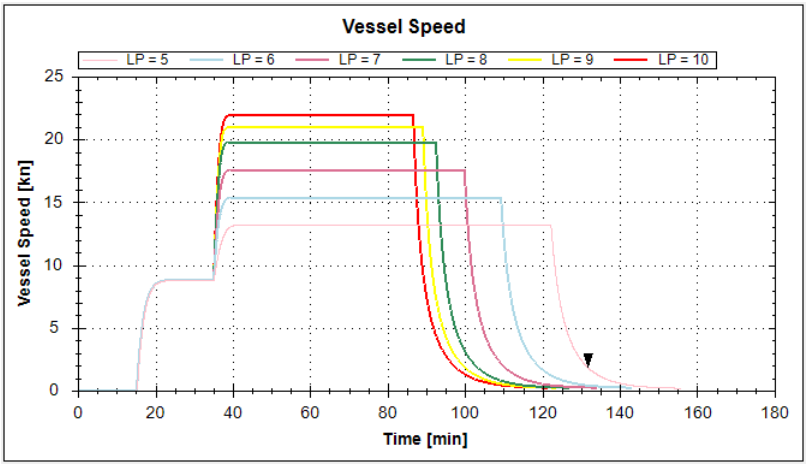


Figure 7.9: Simulator User Interface

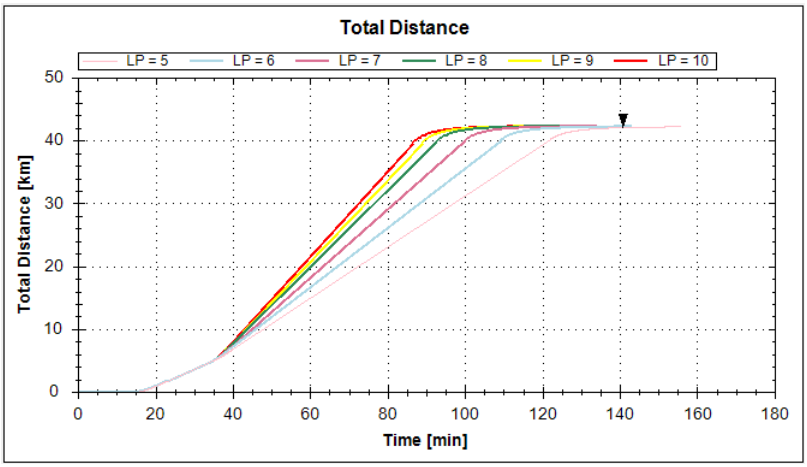


Figure 7.10: Simulator User Interface

7.5. Conclusion

In this chapter optimisation approaches in terms of engine efficiency and fuel consumption per unit of time are explained. Finally, the results of both optimisation approaches have been discussed.

Engine Efficiency

In order to optimise the combinator curve in terms of engine efficiency, an approach is developed where the SFOC map of an engine is used. The propeller pitch and shaft speed setting for a certain vessel speed demand have been determined by looking for the minimal specific fuel consumption value of a number of combinator settings that each result in an operating point to deliver the thrust demand. The expected behaviour for the resulting combinator curve was that the propeller load curve in the operating envelope of the engine would follow the points of the SFOC map with the lowest specific fuel consumption.

SFOC maps of two different engines have been compared and in both cases the optimised combinator settings resulted in a propeller load curve that followed the points with minimal specific fuel consumption. When it comes to the resulting load curve in the operating envelope it was observed that this is not a combinator curve that can be used as it is. As such it is concluded that this approach to optimise a combinator curve in terms of engine efficiency with the use of an SFOC map is successful, but does not result in a satisfactory combinator curve. However, the design of a combinator curve is always an iterative process and thus, if the operating envelope and SFOC map of the driving engine is available, the developed approach can be used for optimisation and evaluation in terms of engine efficiency.

Fuel Consumption

The approach to optimise the combinator curve in terms of the performance indicator fuel consumption per unit time is also based on the SFOC map of the engine. However, in this approach not the minimal specific fuel consumption determines the optimal operating point, but the minimal value of the multiplication of specific fuel consumption and required brake power. It was expected that the propeller should be driven as a fixed pitch propeller(FPP) in order to sail in the most fuel efficient condition in terms of total fuel consumption.

When comparing the combinator settings of the *default mode* to the resulting optimised combinator curve in terms of fuel consumption per unit time, the settings are not entirely similar but lie very close. Especially the graphs of the combinator settings versus the lever positions from zero to maximum ship speed show similar trends. Further, the graph of the fuel consumption for the *default* combinator settings are compared to the optimised values, both versus the lever positions. It was observed that at each lever position the difference is negligible. Thus, both the optimised combinator curve and the combinator curve calculated with the default mode can be used to optimise and evaluate a combinator curve in terms of fuel consumption per unit time.

Simulator

In order to have a complete evaluation for fuel consumption, a simulator is developed with which the fuel consumption can be estimated based on the designed combinator curve and the operational profile of the vessel. If an SFOC map is used that belongs to the engine of the propulsion system of the vessel, the estimation for the total fuel consumption for a certain trip can be estimated with this simulator. Finally, the total fuel consumption can be evaluated for trips at various cruise speeds.

8

Conclusions & Recommendations

In this thesis optimisation approaches to set up combinator curves for operation modes in terms of important performance indicators of a vessel and its propulsion configuration are developed and implemented. Besides this, a Combinator Curve Generator (CCG) software application is developed for the design, optimisation and evaluation of combinator curves for vessels that employ controllable pitch propellers (CPP). The objectives and research questions proposed in the introduction, in order to realize the CCG, were formulated as follows:

1. Develop and implement approaches to set up combinator curves for different operation modes of a vessel and its propulsion configuration in terms of important performance criteria.
   (a) What important performance indicators can be identified for which a combinator curve can be optimised and how can these be quantified?
   (b) Which propulsion configuration related constraints can be identified and how can they be taken into account in the development of the CCG?
   (c) What approach should be taken to optimise a combinator curve for a certain performance indicator?
2. Develop a software application for the design, optimisation and evaluation of combinator curves.

8.1. Literature Research

Firstly, a literature research has been performed in order to answer the three research questions of the first objective that was introduced in the introduction.

(a) Four important performance indicators have been identified and quantified for which a combinator curve can be optimised namely: propeller efficiency, cavitation inception, engine efficiency and fuel consumption per unit time.

(b) Concerning the propulsion configuration related constraints; the change of pitch, number of connected engines per shaft, power take off(PTO) and power take in(PTI) are taken into account. The following was considered within the scope for the CCG: connected engines per shaft of similar type and size, constant PTO or PTI and main focus on diesel, dual fuel and gas engines for fuel consumption related performance indicators that require data of the operating envelope and specific fuel oil consumption (SFOC) maps. The propulsion configuration is further constrained by the limits of the propeller, gearbox and driving engine.

(c) A numerical approach is taken to optimise the combinator curves for a certain performance indicator, resulting in algorithms containing mathematical relations and interpolation methods.

8.2. Developed and Implemented Approaches

The optimisation approaches for the set up of a combinator curve for each of the performance indicators are the following, hence fulfilling the first objective:

- **Propeller Efficiency**
  The optimisation approach of a combinator curve in terms of propeller efficiency is based on the open water efficiency of the propeller open water characteristics (OWCs). The resulting combinator curve shows expected behaviour and can be used for optimisation and evaluation in terms of propeller efficiency.

- **Cavitation Inception**
  In order to limit the risk of cavitation inception, first the effective angle of attack proposed by Vrijdag [49], was implemented. The resulting combinator curve showed expected results according to the method. However, as this method was intended for propulsion systems in operational conditions, including the ability to measure cavitation behaviour, the possibility to calibrate and validate the resulting combinator curve is currently limited. For this reason a new approach has been developed, called the Cavitation Inception Prevention(CIP) method, using generated cavitation inception diagrams for a range of pitch settings, which contain information about the region where cavitation inception is expected to occur. The combinator curve can now be optimised in terms of the advance coefficient in order for the operating point to be located such, that there is sufficient margin against pressure side cavitation and minimal suction side cavitation. The same software from which these cavitation inception diagrams are generated, is currently used for validation of the resulting optimised combinator curve in terms of cavitation inception. With the use of pitch related inception diagrams for several operating points, combinator curves can be optimised and evaluated in order to limit cavitation inception in terms of pressure and suction side cavitation.

- **Engine Efficiency**
  Optimisation of a combinator curve in terms of engine efficiency is dependent on the minimal specific fuel consumption. For this purpose a SFOC map is required. It is possible to use a general SFOC map in case the main engine is a diesel engine. This general SFOC map is available in the CCG. If the SFOC map of the particular main engine is known, its data can be inserted and used for the optimisation method. By means of iteration and interpolation, the optimal combination of pitch and shaft speed are determined, resulting in a combinator curve optimised in terms of engine efficiency. If the operating envelope and SFOC map of the driving engine are available, the resulting combinator curve can be used for optimisation and evaluation of a combinator curve design.

- **Fuel consumption per unit time**
  As for the optimisation of a combinator curve in terms of engine efficiency, the optimisation approach in terms of fuel consumption per unit time is dependent on the SFOC map. For this approach, not the minimal specific fuel consumption, but the minimal product of the specific fuel consumption and the required brake power result in an optimal combination of pitch and shaft speed for each lever position. The optimised combinator curve in terms of fuel consumption per unit time can be used to optimise and evaluate a combinator curve.

Complementary to the evaluation of fuel consumption a **trip simulator** is developed. With this tool, the total fuel consumption can be calculated for a trip based on the operational profile, including hotel load, for the designed combinator curve. The simulation is conducted for several different cruise speeds such that the optimal vessel speed can be identified for a certain trip distance and duration in order to minimize the total fuel consumption.

Besides the approaches to optimise a combinator curve for a certain performance indicator, approaches for several operation modes are developed and implemented:

- **Manual Mode**
  A combinator curve can be designed manually by choosing the desired rotational speed at each lever position. The pitch is then calculated accordingly in order to achieve the vessel speed demand at the respective lever position. For several test cases the original combinator settings have been inserted into the input data and a combinator curve was calculated via the manual mode. The resulting combinator curve was compared to combinator settings of the originally designed combinator curve and showed similar combinator settings. In this way, the CCG is verified in terms of expected behaviour.

- **Default Mode**
  A standard operation mode for which a combinator curve can be set up is the default mode. In this

approach the combinator curve is set up by calculating the combinator settings such that the pitch setting is constant unless the operating limits of the propeller or engine are exceeded.

- **Constant Speed Mode**
  The last standard operation mode for which a combinator can be set up is the constant speed mode. For this mode the combinator settings are calculated such that at each lever position the shaft speed setting is equal to the maximum shaft speed of the engine.

8.3. Developed Combinator Curve Generator

The CCG is developed such that the requirements for such a software application proposed in the introduction are met, thus fulfilling the the second objective. This means that:

- It is possible to optimise a combinator curve for a certain operation mode in terms of important performance criteria.
- The calculated or optimised combinator curve is set up and presented in a form such that it can be used as control input for the CPP in terms of the blade pitch and shaft speed setting per lever position, as per the industry standard to apply Single Lever Command.
- System performance indicators are quantified such that the combinator curve design can be evaluated and compared to other combinator curve designs.
- It is possible to consider a variety of propulsion configurations.
- It is possible to take into account the operational profile of a vessel.

The CCG is developed using Object Orientated Programming and contains a Graphical User Interface where all required data can be inserted and the optimisation, design and evaluation of a combinator curve can be performed, either manually or for a certain operation mode.

8.4. Recommendations

During the development of the CCG the scope of work was focused mostly on ensuring that optimisation approaches for the identified performance indicators are developed and implemented such that a combinator curve can be optimised and evaluated. The following points include recommendations for further development of the CCG and future research.

- **Cavitation inception**
  - The cavitation inception diagrams that are in the data base of the CCG and used for the optimisation approach to interpolate cavitation inception diagrams for various pitch settings, belong to the propeller from the benchmark. To ensure reliable results, they have to be updated for each new propeller for a new project. This increases the labour intensity. For further development, this data base of cavitation inception diagrams could perhaps be extended for a number of different propeller types and blade area ratios. Perhaps not only an extension of the database, but also the possibility to generate inception diagrams by iteration through a number of CPPs with different blade area ratios. It is recommended to investigate the possibility to do this with a larger set of propellers and different blade area ratios and to perform a data based validation with the resulting combinator curve settings.
  - Besides extending and validating the possibility to interpolate through the database, it is recommended to perform full scale or model scale experiments in order to test whether the combinator settings indeed result in minimal suction side cavitation and sufficient margin against pressure side cavitation.
  - Part of validating the resulting combinator settings, it should also be determined whether other forms of cavitation, such as bubble cavitation are accounted for. If not, constraints should be added to the optimisation algorithm.

- **Trip simulation tool**
  - Currently, the simulation tool is specifically for the evaluation of total fuel consumption during a trip at various cruise speeds. The tool could be extended by also giving an indication of the fuel costs on a yearly basis, taking into account the amount of trips the vessel is expected to take.
  - Furthermore, the simulation tool could be extended for specific operation modes such as bollard pull or to investigate the manoeuvrability of a vessel.
  - Finally, it is recommended to investigate the possibility to elevate the simulation tool to a higher degree of optimisation in order to estimate and optimise the route, crew size, system efficiency and fuel consumption. This is especially interesting for companies that are investing in automation and alternative power supply systems such as batteries and fuel cells.

- **Propulsion configuration**

  - Compared to other tools, the variety of propulsion configurations that can be considered in the CCG for a certain operation mode, is sufficient for a large portion of the vessels that are currently built and a big step forward. However, the scope of propulsion configurations is still limited and as mentioned before, propulsion configurations are becoming more and more complex. In order to be able to capture the whole scope of propulsion configurations, the different options for main engines and power supply systems should be considered as separate parts within the CCG, such that each of these systems can be designated with their own characteristics such as efficiencies, operating envelope and SFOC maps.
  - Not only the scope of propulsion configurations, but also the ability to evaluate and determine the optimal load sharing could be an interesting feature. And if system dynamics are taken into account, then also the control systems can be modelled which increases the accuracy and reliability of the resulting combinator settings.

- **Emissions**
  Because each regulation or performance indicator for harmful emissions is in fact one performance indicator for which a combinator curve could be optimised and due to limited research on the impact of various emissions on the matching problem, it was chosen to leave optimisation in terms of harmful emissions out of scope. When further research considering various propulsion configurations and investigation of the influence on the matching problem of other emission regulations is done, as recommended by Ren et al. [33], combinator curves can also be optimised for emission related performance indicators.

- **Effective angle of attack method**
  For the optimisation approach in order to limit the risk of cavitation inception, at first the effective angle of attack method was implemented. This method was intended for propulsion control in operational conditions, and the calibration coefficient could be calibrated using full scale measurements. During this thesis project it was however not possible to do full scale measurements, and also in commercial shipping it is not standard to perform cavitation measurements during sea trials. If perhaps in the future this is done, then this method is a robust and reliable one, which can be used for the optimisation of combinator curves in the CCG. At this stage however, it is recommended to investigate the possibility to calibrate the coefficient, or the overlapping approach of the inception diagrams to determine the optimal effective angle of attack, on the basis of the optimisation approach proposed in this thesis.

- **Resulting combinator settings after sea trials**
  Finally, the control input as it was designed by the engineer is often not the final result after sea trials. Before using the resulting combinator curve, the data is adapted such that it can be used for the software and hardware of the system control system aboard the vessel. Besides this, the combinator settings are calibrated during sea trials according to vessel behaviour and the captains desired operation. It is recommended to analyse the combinator settings after sea trials with the CCG in order to evaluate what is left from the combinator curve design in terms of performance.

A

Results Cavitation Behaviour

In the figures A.1 through A.7 the cavitation behaviour is shown resulting from cavitation inception calculations for lever positions 4-10 for the benchmark introduced in Chapter 5. For figures A.1-A.6 the top left result depicts the cavitation behaviour of the originally calculated combinator settings for the respective lever position. Then, the top right result shows the cavitation behaviour of the optimised combinator settings according to the *effective angle of attack* method. And finally, the bottom result shows the cavitation behaviour of the optimised combinator settings according to the CIP method. For Figure A.7 the combinator setting was similar for each calculation with maximum propeller pitch and maximum rotational speed.

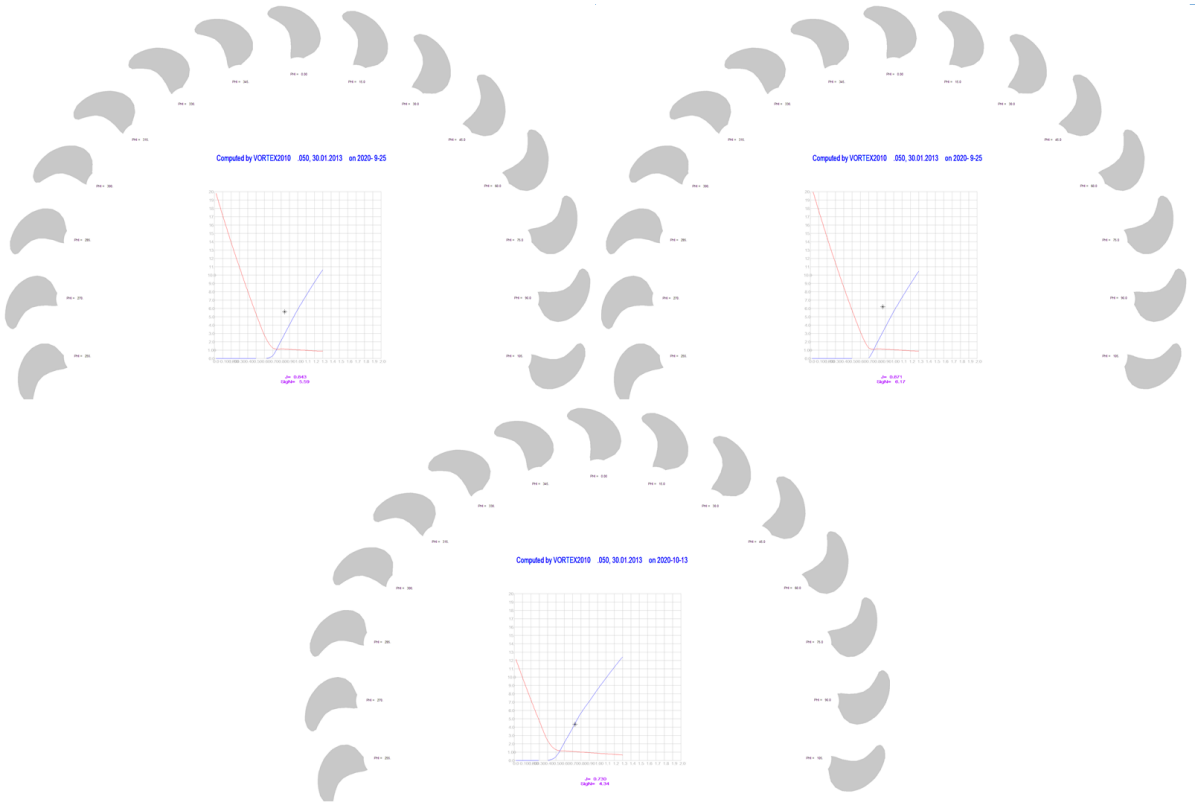


Figure A.1: Cavitation inception results $LP = 4$, top left: result for original operation point, top right: result of optimised operation point using the *effective angle of attack* method, bottom: result of optimised operation point using the CIP method.

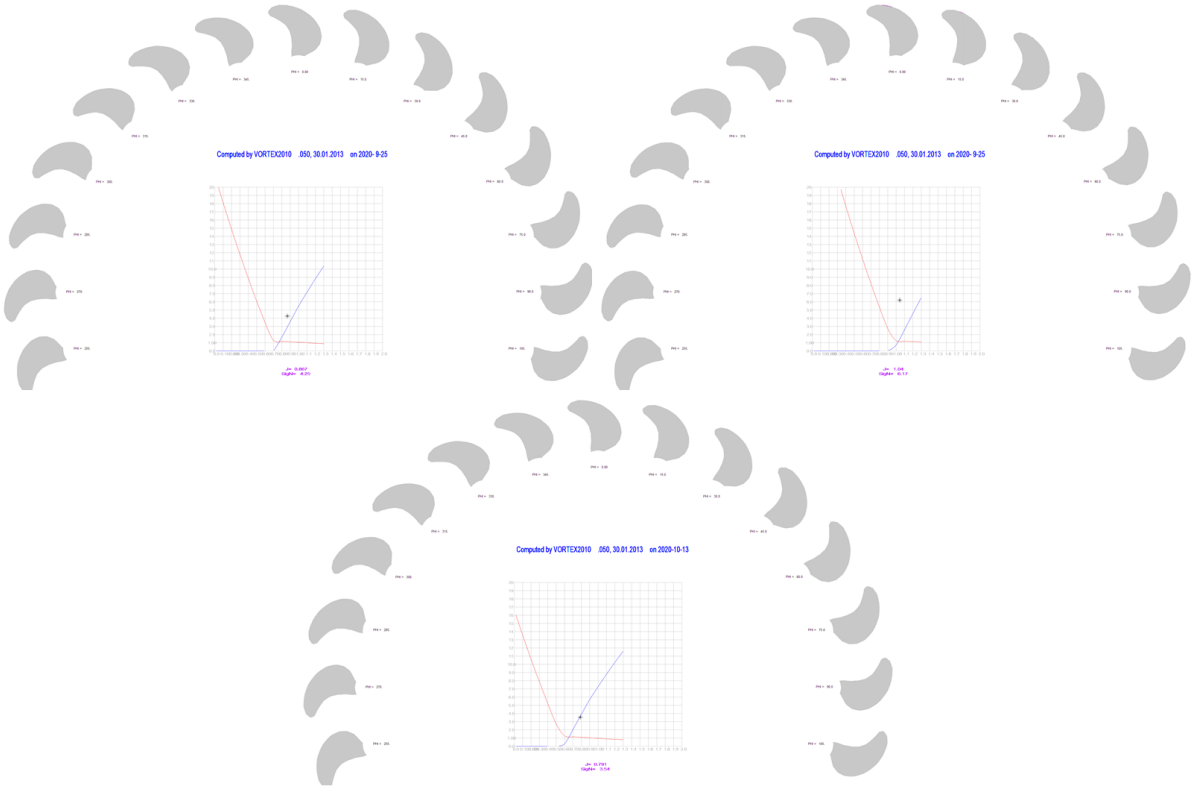


Figure A.2: Cavitation inception results $LP = 5$, top left: result for original operation point, top right: result of optimised operation point using the *effective angle of attack* method, bottom: result of optimised operation point using the CIP method.

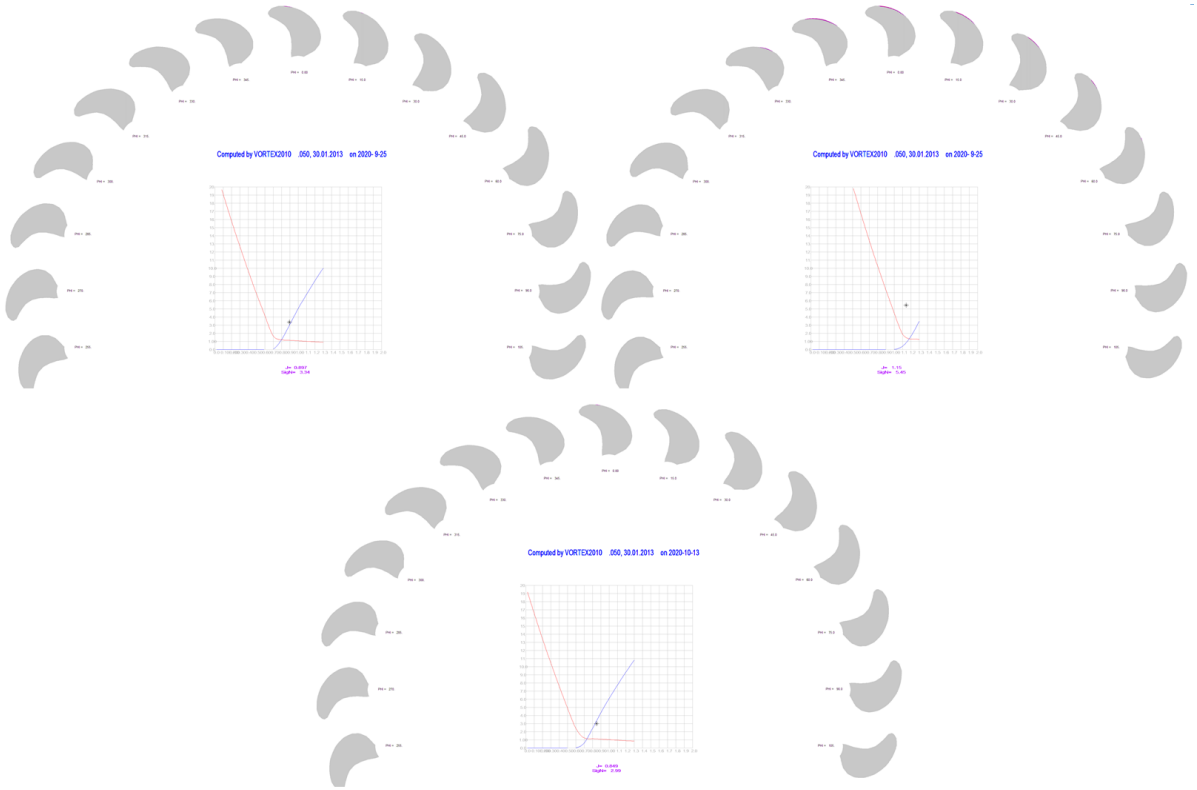


Figure A.3: Cavitation inception results $LP = 6$, top left: result for original operation point, top right: result of optimised operation point using the *effective angle of attack* method,bottom: result of optimised operation point using the CIP method.

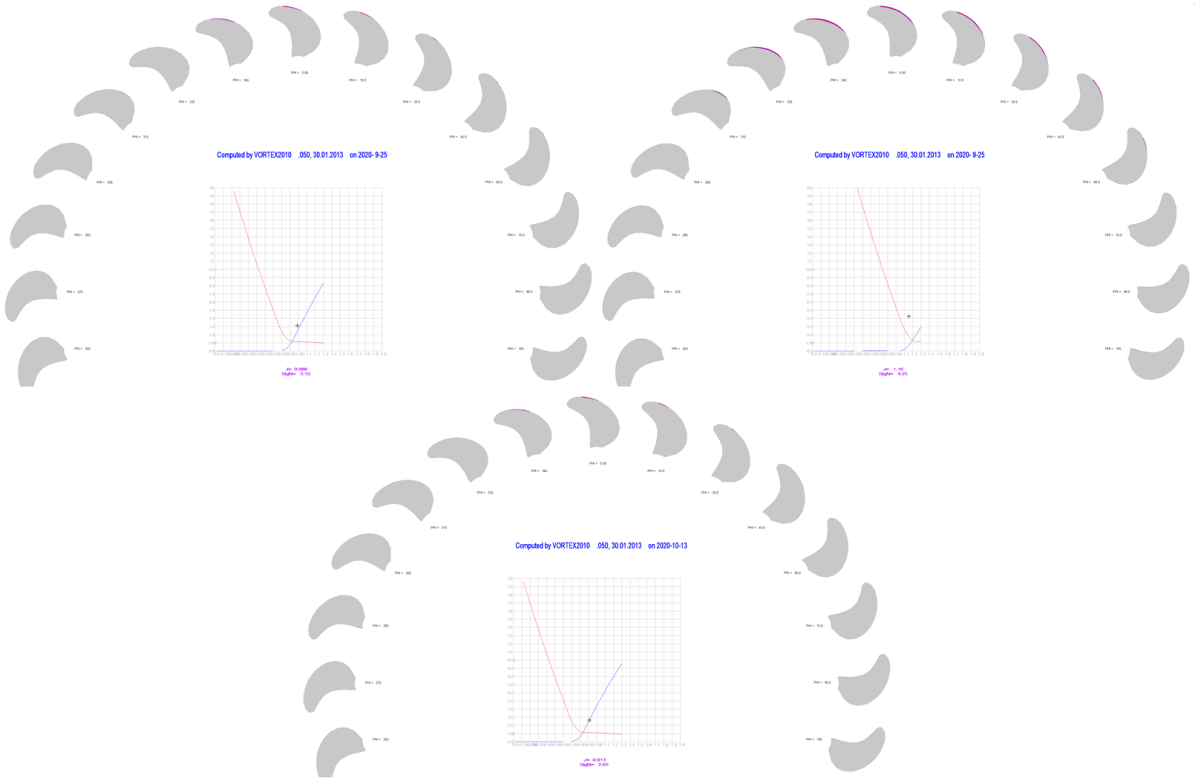


Figure A.4: Cavitation inception results $LP = 7$, top left: result for original operation point, top right: result of optimised operation point using the *effective angle of attack* method, bottom: result of optimised operation point using the CIP method.

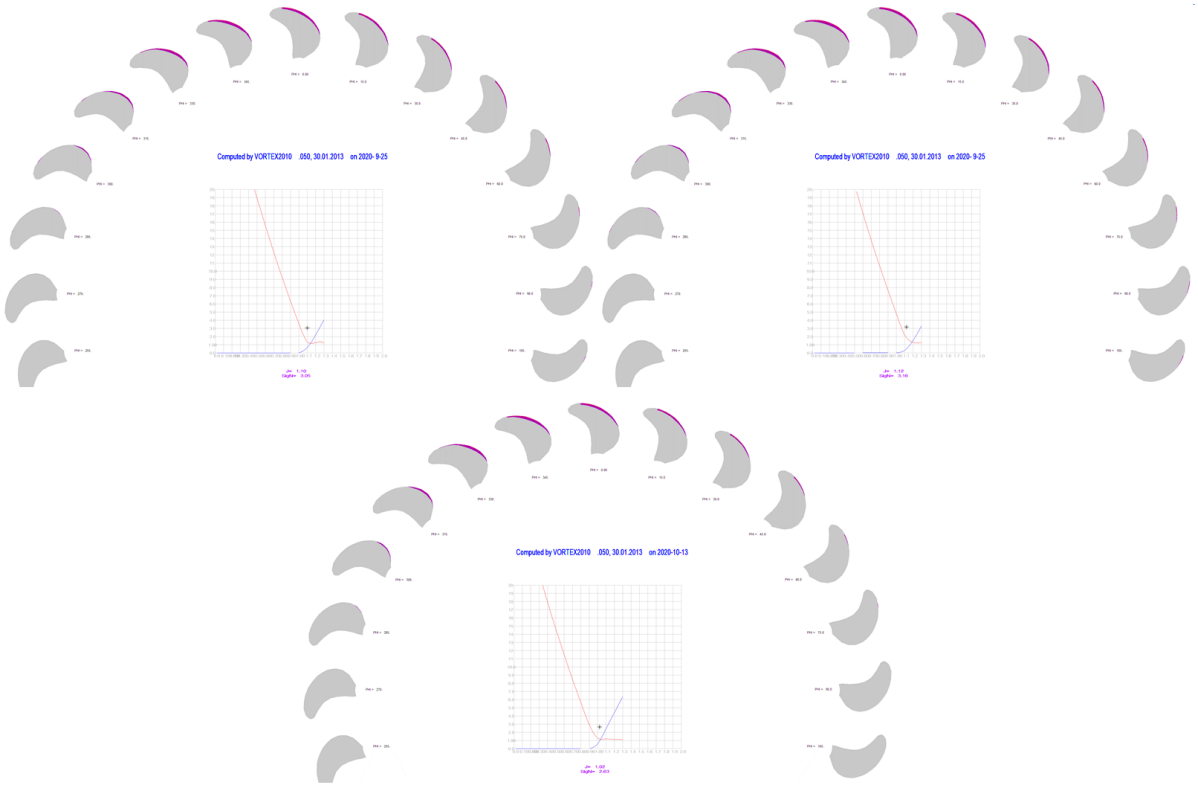


Figure A.5: Cavitation inception results $LP = 8$, top left: result for original operation point, top right: result of optimised operation point using the *effective angle of attack* method, bottom: result of optimised operation point using the CIP method.

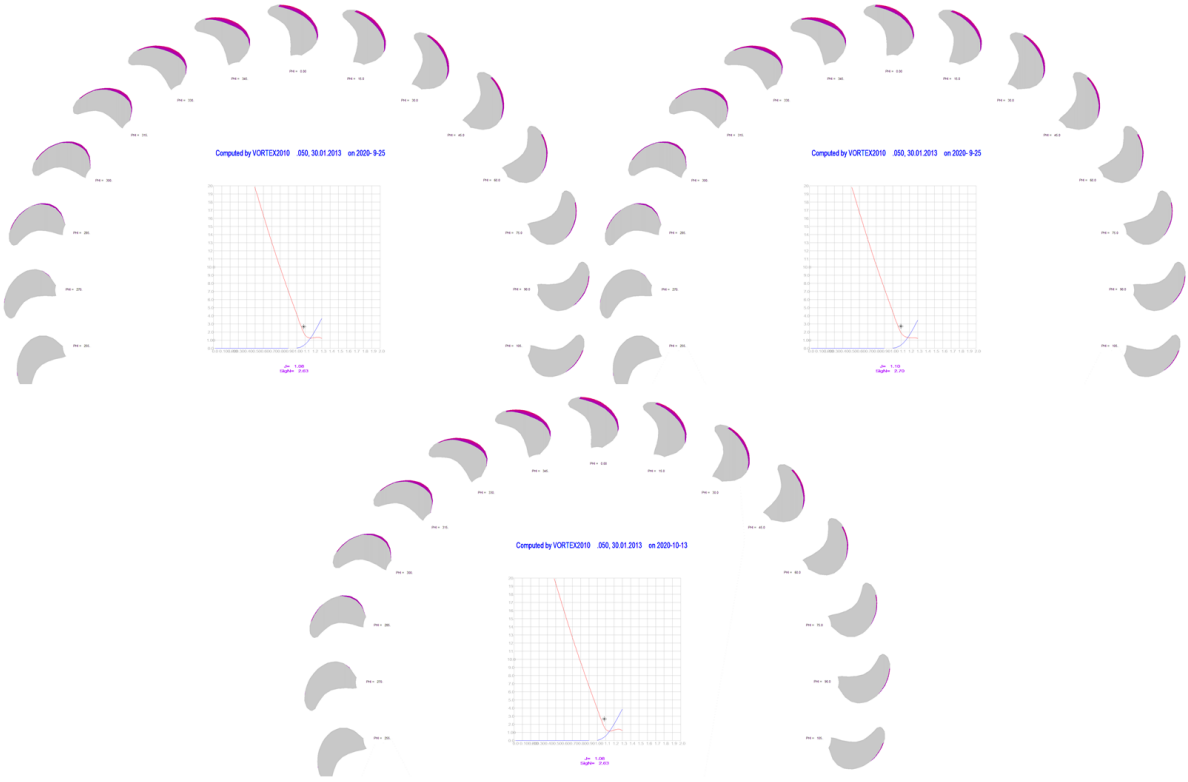


Figure A.6: Cavitation inception results $LP = 9$, top left: result for original operation point, top right: result of optimised operation point using the *effective angle of attack* method, bottom: result of optimised operation point using the CIP method.

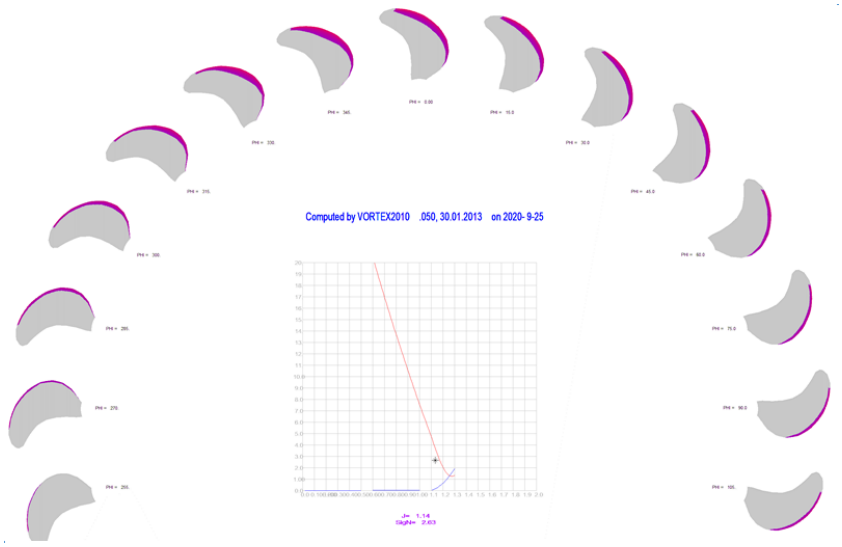


Figure A.7: Cavitation inception results $LP = 10$

B

Source Code

The source code for each of the calculation or optimisation options in the calculation core are shown on the next pages:

Default Core

...tor\CombinatorCalcModel\calculationCores\defaultCore.cs 1

```
1  using MathNet.Numerics.Interpolation;
2  using System;
3  using System.Collections.Generic;
4  using System.ComponentModel.Design;
5  using System.Data;
6  using System.Drawing;
7  using System.Globalization;
8  using System.Linq;
9  using System.Net.Sockets;
10 using System.Reflection.Emit;
11 using System.Text;
12 using System.Threading.Tasks;
13 using static CombinatorCalcModel.defaultCore;
14 using static CombinatorCalcModel.cavitationInceptionCore;
15
16 namespace CombinatorCalcModel
17 {
18     public class defaultCore
19     {
20         public enum eFactorType { PDfactor, PowerFactor, maxInternalPD }
21         public static Dictionary<eFactorType, double> getFactors
             (CombinatorConstraints in_constraints,
22             PropellerModel in_propeller, ShipModel in_ship, EngineModel
               in_engine, GearboxModel in_gearbox)
23         {
24             PropellerOWC owc;
25             PropellerOWCItem owci;
26             ShipResistanceItem in_sri;
27             Dictionary<eFactorType, double> res = new Dictionary<eFactorType,
                 double>();
28
29             // max MCR
30             double maxMCR = in_engine.Envelope.EngineEnvelopeItems.Max(x =>
                 x.Pb) - in_engine.PTO + in_engine.PTI;
31
32             // max ne, np
33             double maxNe = in_engine.Envelope.EngineEnvelopeItems.Max(x =>
                 x.ne);
34             double maxNp = maxNe / 60.0 / in_gearbox.igb;
35
36             // temporary calculation @maxPropellerSpeed
37             double n = maxNp;
38             double maxVs = in_ship.MaxVesselSpeed;
39             double vs = maxVs;
40             in_sri = in_ship.Resistance.Interpolate(vs);
41             double w = in_sri.w;
42             double thrustdemand = in_sri.RequiredThrust /
                 in_constraints.PropulsionConfigConstraints.Kp;
43             double j = vs * (1 - w) * 1.852 / 3.6 / (maxNp *
                 in_propeller.PropellerDiameter / 1000);
44             double kt = calcKT(in_constraints, in_propeller, maxNp,
                 thrustdemand);
45             owc = in_propeller.OpenWaterCurveSet.Interpolate(j, kt);
46             owci = owc.Interpolate(j);
47             double kq = owci.Kq;
```

```
                    cavin, sfc, fc, pb, etae, j, distanceToPressureSide);
325
326         }
327
328         private static double calcPb(CombinatorConstraints in_constraints,
              PropellerModel in_propeller, ShipResistanceItem sri, GearboxModel
              in_gearbox, double n, double kq)
329         {
330             return kq / 10 * in_constraints.EnvironmentalConstraints.RhoSw *
                  Math.Pow(n, 3) * Math.Pow(in_propeller.PropellerDiameter / 1000,
                   5) * 2.0 * Math.PI / in_gearbox.etaTRM / sri.etar / 1000;
331         }
332
333         private static double calcT(CombinatorConstraints in_constraints,
              PropellerModel in_propeller, double n, double kt)
334         {
335             return kt * in_constraints.EnvironmentalConstraints.RhoSw *
                  Math.Pow(n, 2) * Math.Pow(in_propeller.PropellerDiameter / 1000,
                   4);
336         }
337
338         private static double calcKT(CombinatorConstraints in_constraints,
              PropellerModel in_propeller, double n, double thrust)
339         {
340             return thrust / in_constraints.EnvironmentalConstraints.RhoSw /
                  Math.Pow(n, 2) / Math.Pow(in_propeller.PropellerDiameter / 1000,
                   4);
341         }
342     }
343 }
344
345
```

Constant Speed Core

...mbinatorCalcModel\calculationCores\constantSpeedCore.cs 1

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6  using static CombinatorCalcModel.defaultCore;
7  using static CombinatorCalcModel.cavitationInceptionCore;
8
9  namespace CombinatorCalcModel
10 {
11     public class constantSpeedCore
12     {
13         public static CombinatorResultItem Calculate(CombinatorInputItem
             in_cii, CombinatorConstraints in_constraints, ShipResistanceItem
             in_sri,
14             PropellerModel in_propeller, ShipModel in_ship, GearboxModel
               in_gearbox, EngineModel in_engine)
15         {
16             PropellerOWC owc;
17             PropellerOWCItem owci;
18             EngineFuelMapItem efmi;
19             FuelMapConverter fmc = new FuelMapConverter(in_engine);
20
21             double EAR = in_propeller.BladeAreaRatio;
22
23             // correction factors
24             Dictionary<eFactorType, double> facts = defaultCore.getFactors
               (in_constraints, in_propeller, in_ship, in_engine, in_gearbox);
25             double pdfact = facts[eFactorType.PDfactor];
26             double pfact = facts[eFactorType.PowerFactor];
27
28             // max ne, np
29             double maxNe = in_engine.Envelope.EngineEnvelopeItems.Max(x =>
               x.ne);
30             double maxNp = maxNe / 60.0 / in_gearbox.igb;
31
32             // determine parmeters for constant speed@ maxrpm
33             double n = maxNp;
34             double j = in_sri.InflowSpeed / (n *
               in_propeller.PropellerDiameter / 1000d);
35             double thrust = in_sri.RequiredThrust /
               in_constraints.PropulsionConfigConstraints.Kp;
36             double kt = calcKT(in_constraints, in_propeller, n, thrust);
37             owc = in_propeller.OpenWaterCurveSet.Interpolate(j, kt);
38             double pd = owc.PD;
39             owci = owc.Interpolate(j);
40             double kq = owci.Kq;
41
42             // correction for engine load
43             kq /= pfact;
44             double pb = calcPb(in_constraints, in_propeller, in_sri,
               in_gearbox, n, kq);
45
46             // correction for pitch ratio
47             pd /= pdfact;
48
```

```
49              // calculations for performance indicators
50              // propeller efficiency
51              owci = owc.Interpolate(j);
52              double etao = owci.etao * pfact;
53              // cavitation
54              double cavin = PropellerOWCItem.Cavin(in_constraints,
                  in_propeller, n);
55              double j_nowake = in_sri.VS / (n *
                  in_propeller.PropellerDiameter / 1000d);
56              double distanceToPressureSide = cavitationInceptionCore.GetDefault
                  ().GetDistanceInJPressureSide(EAR, pd, cavin, j_nowake);
57              //double aEff = Math.Atan(pd / (0.7 * Math.PI)) - Math.Atan
                  (in_propeller.cFactor * in_sri.InflowSpeed / (0.7 * Math.PI * n
                  * in_propeller.PropellerDiameter / 1000)) -
                  in_propeller.EntryAngle;
58              // engine efficiency
59              double pbLimit = in_engine.Envelope.InterpolateNe(n * 60.0 *
                  in_gearbox.igb).Pb;
60
61              if (in_engine.PTI != 0)
62              {
63                  efmi = in_engine.FuelMap.Interpolate(fmc.NormalizePB(0.9 *
                      pbLimit, n * 60.0 * in_gearbox.igb), fmc.NormalizeNE(n *
                      60.0 * in_gearbox.igb));
64              }
65              else
66              {
67                  efmi = in_engine.FuelMap.Interpolate(fmc.NormalizePB(pb +
                      in_engine.PTO, n * 60.0 * in_gearbox.igb), fmc.NormalizeNE(n
                       * 60.0 * in_gearbox.igb));
68              }
69
70
71              double sfc = fmc.SFCToFullScale(efmi.normsfc);
72              double fc = 0;
73
74              if (in_engine.PTI != 0)
75              {
76                  fc = sfc * (0.9 * pbLimit) + in_engine.GenSFC *
                      in_engine.PTI / in_engine.PTIloss;
77              }
78              else
79              {
80                  fc = sfc * (pb + in_engine.PTO);
81              }
82
83              double etae = 3600000 / (sfc * in_engine.LowerHeatingValue);
84
85
86              n *= 60;
87              return new CombinatorResultItem(in_cii.LP, in_sri.vs, pd, n, etao,
                    cavin, sfc, fc, pb, etae, j, distanceToPressureSide);
88
89          }
90
91          private static double calcPb(CombinatorConstraints in_constraints,
```

Propeller Efficiency Core

...orCalcModel\calculationCores\propellerEfficiencyCore.cs 1

```
 1  using System;
 2  using MathNet.Numerics.Interpolation;
 3  using System.Collections.Generic;
 4  using System.Data;
 5  using System.Drawing;
 6  using System.Linq;
 7  using System.Text;
 8  using System.Threading.Tasks;
 9  using System.Globalization;
10  using System.Data.OleDb;
11  using MathNet.Numerics.Integration;
12  using THPToolSet;
13  using System.Diagnostics;
14  using System.IO;
15  using MathNet.Numerics.Optimization;
16  using static CombinatorCalcModel.defaultCore;
17
18  namespace CombinatorCalcModel
19  {
20      public class propellerEfficiencyCore
21      {
22          public static CombinatorResultItem Calculate(CombinatorInputItem
              in_cii, CombinatorConstraints in_constraints, ShipResistanceItem
              in_sri,
23              PropellerModel in_propeller, ShipModel in_ship, GearboxModel
                in_gearbox, EngineModel in_engine)
24          {
25              PropellerOWC owc;
26              PropellerOWC towc;
27              PropellerOWCItem owcii;
28              EngineFuelMapItem efmi;
29              FuelMapConverter fmc = new FuelMapConverter(in_engine);
30
31              // correction factors
32              Dictionary<eFactorType, double> facts = defaultCore.getFactors
                  (in_constraints, in_propeller, in_ship, in_engine, in_gearbox);
33              double maxInternalPD = facts[eFactorType.maxInternalPD];
34              double pdfact = facts[eFactorType.PDfactor];
35              double pfact = facts[eFactorType.PowerFactor];
36
37              // min/max ne, np
38              double maxNe = in_engine.Envelope.EngineEnvelopeItems.Max(x =>
                  x.ne);
39              double clutchInNe = in_engine.ClutchIn;
40              double minNp = clutchInNe / 60.0 / in_gearbox.igb;
41              double maxNp = maxNe / 60.0 / in_gearbox.igb;
42
43              double vs = 0, n = 0, j = 0, j_check = 0, kt = 0, kq = 0, thrust =
                   0, pd = 0, pb = 0, pbLimit = 0, etao = 0, etae, cavin = 0, sfc
                  = 0, fc = 0;
44              double Dp = in_propeller.PropellerDiameter / 1000;
45              double EAR = in_propeller.BladeAreaRatio;
46
47              int steps = 8;
48              List<PropellerOWCItem> owcItems = new List<PropellerOWCItem>();
49              double nmin = clutchInNe/maxNe, nmax = 1;
```

```
50              double maxEta0, checketao;
51              int ind, minind, maxind;
52
53              for (int i = 0; i < 3; i++)
54              {
55                  for (int ii = 0; ii <= steps; ii++)
56                  {
57                      n = THPToolSet.MathHelpers.InterpolateLinear(0, nmin,
                        steps, nmax, ii) * maxNp;
58                      thrust = in_sri.RequiredThrust /
                        in_constraints.PropulsionConfigConstraints.Kp;
59                      j = in_sri.InflowSpeed / (n * Dp);
60                      kt = calcKT(in_constraints, in_propeller, n, thrust);
61                      owc = in_propeller.OpenWaterCurveSet.Interpolate(j, kt);
62
63                      vs = in_cii.SpeedPercentage / 100d *
                        in_ship.MaxVesselSpeed;
64                      j_check = vs / (n * Dp);

65
66                      etao = owc.Interpolate(j).etao;
67                      checketao = owc.Interpolate(j_check).etao;
68
69                      if (in_cii.LP < 6)
70                      {
71                          if ((owc.PD <= maxInternalPD && etao < 1) /*|| j <=
                        0.3*/ || ii == steps)
72                              owcItems.Add(owc.Interpolate(j));
73                      }
74                      if (in_cii.LP >= 6)
75                      {
76                          if ((owc.PD <= maxInternalPD && etao <= checketao &&
                        etao < 1) /*|| j <= 0.3*/ || ii == steps)
77                              owcItems.Add(owc.Interpolate(j));
78                      }
79
80
81              }
82                  maxEta0 = owcItems.Max(y => y.etao);
83                  ind = owcItems.FindIndex(y => y.etao == maxEta0);
84                  minind = Math.Max(0, ind - 1);
85                  maxind = Math.Min(ind + 1, owcItems.Count() - 1);
86
87                  if (minind != maxind)
88                  {
89                      //search for minrpm
90                      owcii = owcItems[minind];
91                      nmin = in_sri.InflowSpeed / (owcii.J * Dp) / maxNp;
92
93                      owcii = owcItems[maxind];
94                      nmax = in_sri.InflowSpeed / (owcii.J * Dp) / maxNp;
95
96                  }
97                  else break;
98
99              }
```

```
100
101             maxEta0 = owcItems.Max(y => y.etao);
102             owcItems = owcItems.OrderByDescending(x => x.etao).ToList();
103
104             foreach (PropellerOWCItem owci in owcItems)
105             {
106                 towc = in_propeller.OpenWaterCurveSet.Interpolate(owci.J,
                      owci.Kt);
107                 if ((towc.PD < maxInternalPD) || owcItems.Count() == 1)
108                 {
109                     pd = towc.PD;
110                     if (owci.J > 0)
111                         n = in_sri.InflowSpeed / (owci.J * Dp);
112                     else
113                         n = minNp;
114
115                     j = owci.J;
116                     kt = owci.Kt;
117                     kq = owci.Kq;
118                     break;
119                 }
120             }
121
122             // correction for engine load
123             kq /= pfact;
124             pb = calcPb(in_constraints, in_propeller, in_sri, in_gearbox, n,
                  kq);
125
126             //check for engine overload
127             pbLimit = in_engine.Envelope.InterpolateNe(n * 60.0 *
                  in_gearbox.igb).Pb;
128
129             if (in_constraints.UseEngineLimit && pb >
                  in_constraints.DesignPointConstraints.EngineMargin / 100d *
                  pbLimit && in_cii.LP < 10 && in_engine.PTI == 0)
130             {
131                 CombinatorResultItem engineOverloadResult =
                      defaultCore.CalculateEngineOverload(in_cii, in_constraints,
                      in_propeller, in_gearbox, in_engine, in_ship);
132
133                 n = engineOverloadResult.np;
134                 j = in_sri.InflowSpeed / (n * Dp);
135                 kt = calcKT(in_constraints, in_propeller, n, thrust);
136                 owc = in_propeller.OpenWaterCurveSet.Interpolate(j, kt);
137                 owcii = owc.Interpolate(j);
138                 kq = owcii.Kq / pfact;
139                 pb = calcPb(in_constraints, in_propeller, in_sri, in_gearbox,
                      n, kq);
140             }
141             else if (in_cii.LP == 10)
142             {
143                 n = maxNp;
144                 j = in_sri.InflowSpeed / (n * Dp);
145                 kt = calcKT(in_constraints, in_propeller, n, thrust);
146                 owc = in_propeller.OpenWaterCurveSet.Interpolate(j, kt);
147                 owcii = owc.Interpolate(j);
```

```
148                 kq = owcii.Kq / pfact;
149                 pb = calcPb(in_constraints, in_propeller, in_sri, in_gearbox,
                      n, kq);
150             }
151             else { }
152
153             owc = in_propeller.OpenWaterCurveSet.Interpolate(j, kt);

154             owcii = owc.Interpolate(j);
155
156             // correction for pitch ratio
157             pd = owc.PD;
158             pd /= pdfact;
159
160             // calculations for performance indicators
161             // propeller efficiency
162             etao = owcii.etao * pfact;
163                 // cavitation
164                 cavin = PropellerOWCItem.Cavin(in_constraints, in_propeller,
                      n);
165             double j_nowake = in_sri.VS / (n *
                  in_propeller.PropellerDiameter / 1000d);
166             double distanceToPressureSide = cavitationInceptionCore.GetDefault
                  ().GetDistanceInJPressureSide(EAR, pd, cavin, j_nowake);
167             //double aEff = Math.Atan(pd / (0.7 * Math.PI)) - Math.Atan
                  (in_propeller.cFactor * in_sri.InflowSpeed / (0.7 * Math.PI * n
                  * Dp)) - in_propeller.EntryAngle;
168             // engine efficiency & fuel consumption
169             if (in_engine.PTI != 0 && pb > 0.9 * pbLimit)
170             {
171                 efmi = in_engine.FuelMap.Interpolate(fmc.NormalizePB(0.9 *
                      pbLimit, n * 60.0 * in_gearbox.igb), fmc.NormalizeNE(n *
                      60.0 * in_gearbox.igb));
172             }
173             else
174             {
175                 efmi = in_engine.FuelMap.Interpolate(fmc.NormalizePB(pb +
                      in_engine.PTO, n * 60.0 * in_gearbox.igb), fmc.NormalizeNE(n
                       * 60.0 * in_gearbox.igb));
176             }
177
178             if (in_constraints.UseEngineLimit)
179             {
180                 sfc = fmc.SFCToFullScale(efmi.normsfc);
181
182                 if (in_engine.PTI != 0 && pb > 0.9 * pbLimit)
183                 {
184                     fc = sfc * (0.9 * pbLimit) + in_engine.GenSFC * (pb - 0.9
                       * pbLimit) / in_engine.PTIloss;
185                 }
186                 else
187                 {
188                     fc = sfc * (pb + in_engine.PTO);
189                 }
190
191                 etae = 3600000 / (sfc * in_engine.LowerHeatingValue);
```

```
192             }
193             else
194             {
195                 sfc = double.NaN;
196                 fc = double.NaN;
197                 etae = double.NaN;
198             }
199 
200 
201             n *= 60;
202 
203             return new CombinatorResultItem(in_cii.LP, in_sri.vs, pd, n, etao,
                    cavin, sfc, fc, pb, etae, j, distanceToPressureSide);
204         }
205 
206         private static double calcPb(CombinatorConstraints in_constraints,
              PropellerModel in_propeller, ShipResistanceItem sri, GearboxModel
              in_gearbox, double n, double kq)
207         {
208             return kq / 10 * in_constraints.EnvironmentalConstraints.RhoSw *
                  Math.Pow(n, 3) * Math.Pow(in_propeller.PropellerDiameter / 1000,
                  5) * 2.0 * Math.PI / in_gearbox.etaTRM / sri.etar / 1000;
209         }
210         private static double calcKT(CombinatorConstraints in_constraints,
              PropellerModel in_propeller, double n, double thrust)
211         {
212             return thrust / in_constraints.EnvironmentalConstraints.RhoSw /
                  Math.Pow(n, 2) / Math.Pow(in_propeller.PropellerDiameter / 1000,
                  4);
213         }
214     }
215 }
216 
```

Cavitation Inception Core

...\CombinatorCalcModel\calculationCores\cavitationCore.cs 1

```
 1  using System;
 2  using System.Collections.Generic;
 3  using System.Linq;
 4  using System.Text;
 5  using System.Threading.Tasks;
 6  using static CombinatorCalcModel.defaultCore;
 7  using static CombinatorCalcModel.cavitationInceptionCore;
 8
 9  namespace CombinatorCalcModel
10  {
11      public class cavitationCore
12      {
13          public static CombinatorResultItem Calculate(CombinatorInputItem
              in_cii, CombinatorConstraints in_constraints, ShipResistanceItem
              in_sri,
14              PropellerModel in_propeller, ShipModel in_ship, GearboxModel
                in_gearbox, EngineModel in_engine)
15          {
16              PropellerOWC owc;
17              PropellerOWCItem owci;
18              EngineFuelMapItem efmi;
19              FuelMapConverter fmc = new FuelMapConverter(in_engine);
20
21              double j = 0, j_nowake = 0, kt, n = 1, kq = 0, thrust = 0, pb = 0,
                   pbLimit = 0, etao = 0, etae = 0, cavin = 0, sfc = 0, fc = 0,
                  pd, minDiff = 0;
22              double EAR = 0.5; // TODO in propeller Model
23              double Dp = in_propeller.PropellerDiameter / 1000;
24              double distanceToPressureSide = 0;
25
26              // correction factors
27              Dictionary<eFactorType, double> facts = defaultCore.getFactors
                  (in_constraints, in_propeller, in_ship, in_engine, in_gearbox);
28              double maxInternalPD = facts[eFactorType.maxInternalPD];
29              double pdfact = facts[eFactorType.PDfactor];
30              double pfact = facts[eFactorType.PowerFactor];
31
32              // min/max ne, np
33              double maxNe = in_engine.Envelope.EngineEnvelopeItems.Max(x =>
                  x.ne);
34              double clutchInNe = in_engine.ClutchIn;
35              double minNp = clutchInNe / 60.0 / in_gearbox.igb;
36              double maxNp = maxNe / 60.0 / in_gearbox.igb;
37
38              int steps = 8;
39              //List<double> aEffItems = new List<double>();
40              List<double> diffItems = new List<double>();
41              List<PropellerOWCItem> owcItems = new List<PropellerOWCItem>();
42              double nmin = clutchInNe / maxNe, nmax = 1;
43              int ind, minind, maxind;
44
45              for (int i = 0; i < 3; i++)
46              {
47                  for (int ii = 0; ii <= steps; ii++)
48                  {
49                      n = THPToolSet.MathHelpers.InterpolateLinear(0, nmin,
```

```
                    steps, nmax, ii) * maxNp;
50                  thrust = in_sri.RequiredThrust /
                    in_constraints.PropulsionConfigConstraints.Kp;
51                  j = in_sri.InflowSpeed / (n * Dp);
52
53                  if (in_cii.LP < 2)
54                  {
55                      j_nowake = in_sri.VS / (n * Dp) * 1.0;
56                  }
57                  else if(in_cii.LP >= 2)
58                  {
59                      j_nowake = in_sri.VS / (n * Dp) * 1.03;
60                  }
61                  kt = calcKT(in_constraints, in_propeller, n, thrust);
62                  owc = in_propeller.OpenWaterCurveSet.Interpolate(j, kt);
63                  pd = owc.PD;
64
65                  cavin = PropellerOWCItem.Cavin(in_constraints,
                    in_propeller, n);
66
67                  distanceToPressureSide =
                    cavitationInceptionCore.GetDefault
                    ().GetDistanceInJPressureSide(EAR, pd / pdfact, cavin,
                    j_nowake);
68
69                  if ((pd <= maxInternalPD && distanceToPressureSide <= 0)
                    || ii == steps)
70                  {
71                      diffItems.Add(Math.Abs(distanceToPressureSide));
72                      owcItems.Add(owc.Interpolate(j));
73                  }
74
75              }
76
77              minDiff = diffItems.Min(y => y);
78              ind = diffItems.FindIndex(y => y == minDiff);
79              minind = Math.Max(0, ind - 1);
80              maxind = Math.Min(ind + 1, diffItems.Count() - 1);
81
82              if (minind != maxind && owcItems[minind].J > 0)
83              {
84                  owci = owcItems[minind];
85                  nmin = in_sri.InflowSpeed / (owci.J * Dp) / maxNp;
86
87                  owci = owcItems[maxind];
88                  nmax = in_sri.InflowSpeed / (owci.J * Dp) / maxNp;
89
90              }
91              else break;
92
93          }
94
95          minDiff = diffItems.Min(y => y);
96          ind = diffItems.FindIndex(y => y == minDiff);
97          owci = owcItems[ind];
98
```

```
 99              owc = in_propeller.OpenWaterCurveSet.Interpolate(owci.J, owci.Kt);
100              if ((owc.PD < maxInternalPD) || owcItems.Count() == 1)
101              {
102                  pd = owc.PD;
103                  if (owci.J > 0)
104                  {
105                      n = in_sri.InflowSpeed / (owci.J * Dp);
106                      if (n >= maxNp)
107                          n = maxNp;
108                  }
109                  else
110                      n = minNp;
111
112                  j = owci.J;
113                  kt = owci.Kt;
114                  kq = owci.Kq;
115
116              }
117
118
119              // correction for engine load
120              kq /= pfact;
121              pb = calcPb(in_constraints, in_propeller, in_sri, in_gearbox, n,
                   kq);
122
123              //check for engine overload
124              pbLimit =  in_engine.Envelope.InterpolateNe(n * 60.0 *
                   in_gearbox.igb).Pb;
125
126              if (in_constraints.UseEngineLimit && pb >
                   in_constraints.DesignPointConstraints.EngineMargin / 100d *
                   pbLimit && in_cii.LP < 10 && in_engine.PTI == 0)
127              {
128                  CombinatorResultItem engineOverloadResult =
                       defaultCore.CalculateEngineOverload(in_cii, in_constraints,
                       in_propeller, in_gearbox, in_engine, in_ship);
129
130                  n = engineOverloadResult.np;
131                  j = in_sri.InflowSpeed / (n * Dp);
132                  kt = calcKT(in_constraints, in_propeller, n, thrust);
133                  owc = in_propeller.OpenWaterCurveSet.Interpolate(j, kt);
134                  owci = owc.Interpolate(j);
135                  kq = owci.Kq / pfact;
136                  pb = calcPb(in_constraints, in_propeller, in_sri, in_gearbox,
                       n, kq);
137              }
138              else if (in_cii.LP == 10)
139              {
140                  n = maxNp;
141                  j = in_sri.InflowSpeed / (n * Dp);
142                  kt = calcKT(in_constraints, in_propeller, n, thrust);
143                  owc = in_propeller.OpenWaterCurveSet.Interpolate(j, kt);
144                  owci = owc.Interpolate(j);
145                  kq = owci.Kq / pfact;
146                  pb = calcPb(in_constraints, in_propeller, in_sri, in_gearbox,
                       n, kq);
```

```
147             }
148             else { }
149
150             // correction for pitch ratio
151             pd = owc.PD;
152             pd /= pdfact;
153
154             // calculations for performance indicators
155             // propeller efficiency
156             etao = owci.etao * pfact;
157             // cavitation
158             cavin = PropellerOWCItem.Cavin(in_constraints, in_propeller, n);
159             j_nowake = in_sri.VS / (n * Dp);
160             distanceToPressureSide = cavitationInceptionCore.GetDefault
                  ().GetDistanceInJPressureSide(EAR, pd, cavin, j_nowake);
161             //aEff = Math.Atan(pd / (0.7 * Math.PI)) - Math.Atan
                  (in_propeller.cFactor * in_sri.InflowSpeed / (0.7 * Math.PI * n
                  * Dp)) - in_propeller.EntryAngle;
162             // engine efficiency & fuel consumption
163             if (in_engine.PTI != 0 && pb > 0.9 * pbLimit)
164             {
165                 efmi = in_engine.FuelMap.Interpolate(fmc.NormalizePB(0.9 *
                      pbLimit, n * 60.0 * in_gearbox.igb), fmc.NormalizeNE(n *
                      60.0 * in_gearbox.igb));
166             }
167             else
168             {
169                 efmi = in_engine.FuelMap.Interpolate(fmc.NormalizePB(pb +
                      in_engine.PTO, n * 60.0 * in_gearbox.igb), fmc.NormalizeNE(n
                       * 60.0 * in_gearbox.igb));
170             }
171
172             if (in_constraints.UseEngineLimit)
173             {
174                 sfc = fmc.SFCToFullScale(efmi.normsfc);
175
176                 if (in_engine.PTI != 0 && pb > 0.9 * pbLimit)
177                 {
178                     fc = sfc * (0.9 * pbLimit) + in_engine.GenSFC * (pb - 0.9
                       * pbLimit) / in_engine.PTIloss;
179                 }
180                 else
181                 {
182                     fc = sfc * (pb + in_engine.PTO);
183                 }
184
185                 etae = 3600000 / (sfc * in_engine.LowerHeatingValue);
186             }
187             else
188             {
189                 sfc = double.NaN;
190                 fc = double.NaN;
191                 etae = double.NaN;
192             }
193
194             n *= 60;
```

```
195
196             return new CombinatorResultItem(in_cii.LP, in_sri.vs, pd, n, etao,
                    cavin, sfc, fc, pb, etae, j, distanceToPressureSide);
197
198         }
199
200         private static double calcPb(CombinatorConstraints in_constraints,
              PropellerModel in_propeller, ShipResistanceItem sri, GearboxModel
              in_gearbox, double n, double kq)
201         {
202             return kq / 10 * in_constraints.EnvironmentalConstraints.RhoSw *
                  Math.Pow(n, 3) * Math.Pow(in_propeller.PropellerDiameter / 1000,
                    5) * 2.0 * Math.PI / in_gearbox.etaTRM / sri.etar / 1000;
203         }
204
205         private static double calcKT(CombinatorConstraints in_constraints,
              PropellerModel in_propeller, double n, double thrust)
206         {
207             return thrust / in_constraints.EnvironmentalConstraints.RhoSw /
                  Math.Pow(n, 2) / Math.Pow(in_propeller.PropellerDiameter / 1000,
                    4);
208         }
209     }
210 }
211
212
```

Engine Efficiency Core

...natorCalcModel\calculationCores\engineEfficiencyCore.cs 1

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6  using static CombinatorCalcModel.defaultCore;
7
8  namespace CombinatorCalcModel
9  {
10     public class engineEfficiencyCore
11     {
12         public static CombinatorResultItem Calculate(CombinatorInputItem
             in_cii, CombinatorConstraints in_constraints, ShipResistanceItem
             in_sri,
13             PropellerModel in_propeller, ShipModel in_ship, GearboxModel
               in_gearbox, EngineModel in_engine)
14         {
15             PropellerOWC owc;
16             PropellerOWCItem owci;
17             EngineFuelMapItem efmi;
18             FuelMapConverter fmc = new FuelMapConverter(in_engine);
19
20             double n = 0, j = 0, kt = 0, kq = 0, thrust = 0, pd = 0, pb = 0,
               etao = 0, etae = 0, cavin = 0, sfc = 0, fc = 0, pbLimit = 0;
21             double Dp = in_propeller.PropellerDiameter / 1000;
22             double EAR = in_propeller.BladeAreaRatio;
23
24             // correction factors
25             Dictionary<eFactorType, double> facts = defaultCore.getFactors
               (in_constraints, in_propeller, in_ship, in_engine, in_gearbox);
26             double maxInternalPD = facts[eFactorType.maxInternalPD];
27             double pdfact = facts[eFactorType.PDfactor];
28             double pfact = facts[eFactorType.PowerFactor];
29
30             // min/max ne, np
31             double maxNe = in_engine.Envelope.EngineEnvelopeItems.Max(x =>
               x.ne);
32             double clutchInNe = in_engine.ClutchIn;
33             double minNp = clutchInNe / 60.0 / in_gearbox.igb;
34             double maxNp = maxNe / 60.0 / in_gearbox.igb;
35
36             int steps = 10;
37             List<EngineFuelMapItem> efmItems = new List<EngineFuelMapItem>();
38             List<PropellerOWCItem> owcItems = new List<PropellerOWCItem>();
39
40             double nmin = clutchInNe/maxNe, nmax = 1;
41             double minsfc;
42             int ind, minind, maxind;
43
44             for (int i = 0; i < 3; i++)
45             {
46                 for (int ii = 0; ii <= steps; ii++)
47                 {
48                     n = THPToolSet.MathHelpers.InterpolateLinear(0, nmin,
                       steps, nmax, ii) * maxNp;
49                     thrust = in_sri.RequiredThrust /
```

```
                    in_constraints.PropulsionConfigConstraints.Kp;
50                  j = in_sri.InflowSpeed / (n * Dp);
51                  kt = calcKT(in_constraints, in_propeller, n, thrust);
52                  owc = in_propeller.OpenWaterCurveSet.Interpolate(j, kt);
53                  owci = owc.Interpolate(j);
54                  kq = owci.Kq / pfact;
55                  pb = calcPb(in_constraints, in_propeller, in_sri,
                    in_gearbox, n, kq);
56
57
58                  pbLimit = in_engine.Envelope.InterpolateNe(n * 60.0 *
                    in_gearbox.igb).Pb;
59                  if (in_engine.PTI != 0 && pb > 0.9 * pbLimit)

60                      efmi = in_engine.FuelMap.Interpolate(fmc.NormalizePB
                    (0.9 * pbLimit, n * 60.0 * in_gearbox.igb),
                    fmc.NormalizeNE(n * 60.0 * in_gearbox.igb));
61
62                  else
63                      efmi = in_engine.FuelMap.Interpolate(fmc.NormalizePB
                    (pb + in_engine.PTO, n * 60.0 * in_gearbox.igb),
                    fmc.NormalizeNE(n * 60.0 * in_gearbox.igb));
64
65                  if ((owc.PD <= maxInternalPD && pb < pbLimit && in_cii.LP
                    < 10 && efmi.normsfc > 0) || ii == steps)
66                  {
67                      owcItems.Add(owc.Interpolate(j));
68                      efmItems.Add(efmi);
69                  }
70              }
71              minsfc = efmItems.Min(y => y.normsfc);
72              ind = efmItems.FindIndex(y => y.normsfc == minsfc);
73
74              minind = Math.Max(0, ind - 1);
75              maxind = Math.Min(ind + 1, efmItems.Count() - 1);
76
77              if (minind != maxind && owcItems[minind].J > 0)
78              {
79                  owci = owcItems[minind];
80                  nmin = in_sri.InflowSpeed / (owci.J * Dp) / maxNp;
81
82                  owci = owcItems[maxind];
83                  nmax = in_sri.InflowSpeed / (owci.J * Dp) / maxNp;
84
85              }
86              else break;
87
88          }
89
90          minsfc = efmItems.Min(y => y.normsfc);
91          ind = efmItems.FindIndex(y => y.normsfc == minsfc);
92          owci = owcItems[ind];
93
94          owc = in_propeller.OpenWaterCurveSet.Interpolate(owci.J, owci.Kt);
95          if ((owc.PD < maxInternalPD) || owcItems.Count() == 1)
96          {
```

```
 97                  pd = owc.PD;
 98                  if (owci.J > 0)
 99                  {
100                      n = in_sri.InflowSpeed / (owci.J * Dp);
101                  }
102                  else
103                      n = minNp;
104
105                  j = owci.J;
106                  kt = owci.Kt;
107                  kq = owci.Kq;
108              }
109
110              // correction for engine load
111              kq /= pfact;
112              pb = calcPb(in_constraints, in_propeller, in_sri, in_gearbox, n,
                   kq);
113
114              //check for engine overload
115              pbLimit = in_engine.Envelope.InterpolateNe(n * 60.0 *
                   in_gearbox.igb).Pb;
116
117              // correction for pitch ratio
118              pd /= pdfact;
119
120              // calculations for performance indicators
121                  // propeller efficiency
122                  etao = owci.etao * pfact;
123                  // cavitation
124                  cavin = PropellerOWCItem.Cavin(in_constraints, in_propeller,
                       n);
125              double j_nowake = in_sri.VS / (n * Dp);
126              double distanceToPressureSide = cavitationInceptionCore.GetDefault
                   ().GetDistanceInJPressureSide(EAR, pd, cavin, j_nowake);
127              //double aEff = Math.Atan(pd / (0.7 * Math.PI)) - Math.Atan
                   (in_propeller.cFactor * in_sri.InflowSpeed / (0.7 * Math.PI * n
                   * Dp)) - in_propeller.EntryAngle;
128              // engine efficiency
129              efmi = in_engine.FuelMap.Interpolate(fmc.NormalizePB(pb +
                   in_engine.PTO, n * 60.0 * in_gearbox.igb), fmc.NormalizeNE(n *
                   60.0 * in_gearbox.igb));
130                  sfc = fmc.SFCToFullScale(efmi.normsfc);
131                  etae = 3600000 / (sfc * in_engine.LowerHeatingValue);
132              // fuel consumption
133              fc = sfc * (pb + in_engine.PTO) ; // check how PTO is taken into
                   account
134
135
136
137              n *= 60;
138
139              return new CombinatorResultItem(in_cii.LP, in_sri.vs, pd, n, etao,
                    cavin, sfc, fc, pb, etae, j, distanceToPressureSide);
140          }
141
142          private static double calcPb(CombinatorConstraints in_constraints,
```

```
            PropellerModel in_propeller, ShipResistanceItem sri, GearboxModel
            in_gearbox, double n, double kq)
143         {
144             return kq / 10 * in_constraints.EnvironmentalConstraints.RhoSw *
                  Math.Pow(n, 3) * Math.Pow(in_propeller.PropellerDiameter / 1000,
                   5) * 2.0 * Math.PI / in_gearbox.etaTRM / sri.etar / 1000;
145         }
146         private static double calcKT(CombinatorConstraints in_constraints,
            PropellerModel in_propeller, double n, double thrust)
147         {
148             return thrust / in_constraints.EnvironmentalConstraints.RhoSw /
                  Math.Pow(n, 2) / Math.Pow(in_propeller.PropellerDiameter / 1000,
                   4);
149         }
150     }
151 }
152
```

Fuel Consumption Core

...inatorCalcModel\calculationCores\fuelConsumptionCore.cs 1

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6  using static CombinatorCalcModel.defaultCore;
7
8  namespace CombinatorCalcModel
9  {
10     public class fuelConsumptionCore
11     {
12         public static CombinatorResultItem Calculate(CombinatorInputItem
             in_cii, CombinatorConstraints in_constraints, ShipResistanceItem
             in_sri,
13             PropellerModel in_propeller, ShipModel in_ship, GearboxModel
               in_gearbox, EngineModel in_engine)
14         {
15             PropellerOWC owc;
16             PropellerOWCItem owci;
17             EngineFuelMapItem efmi;
18             FuelMapConverter fmc = new FuelMapConverter(in_engine);
19
20             double n=0, pd = 0, j = 0, kt = 0 , thrust = 0 , kq = 0, pb,
               pbLimit, etao, etae, cavin, sfc, fc;
21             double Dp = in_propeller.PropellerDiameter / 1000;
22             double EAR = in_propeller.BladeAreaRatio;
23
24             // correction factors
25             Dictionary<eFactorType, double> facts = defaultCore.getFactors
               (in_constraints, in_propeller, in_ship, in_engine, in_gearbox);
26             double maxInternalPD = facts[eFactorType.maxInternalPD];
27             double pdfact = facts[eFactorType.PDfactor];
28             double pfact = facts[eFactorType.PowerFactor];
29
30             // min/max ne, np
31             double maxNe = in_engine.Envelope.EngineEnvelopeItems.Max(x =>
               x.ne);
32             double clutchInNe = in_engine.ClutchIn;
33             double minNp = clutchInNe / 60.0 / in_gearbox.igb;
34             double maxNp = maxNe / 60.0 / in_gearbox.igb;
35
36             int steps = 8;
37             List<PropellerOWCItem> owcItems = new List<PropellerOWCItem>();
38             List<EngineFuelMapItem> efmItems = new List<EngineFuelMapItem>();
39             List<double> fcItems = new List<double>();
40             double nmin = clutchInNe/maxNe, nmax = 1;
41             double minfc;
42             int ind, minind, maxind;
43
44             for (int i = 0; i < 3 ; i++)
45             {
46                 for (int ii = 0; ii <= steps; ii++)
47                 {
48                     n = THPToolSet.MathHelpers.InterpolateLinear(0, nmin,
                       steps, nmax, ii) * maxNp;
49                     thrust = in_sri.RequiredThrust /
```

```
                    in_constraints.PropulsionConfigConstraints.Kp;
50                  j = in_sri.InflowSpeed / (n * Dp);
51                  kt = calcKT(in_constraints, in_propeller, n, thrust);
52                  owc = in_propeller.OpenWaterCurveSet.Interpolate(j, kt);
53                  owci = owc.Interpolate(j);
54                  kq = owci.Kq / pfact;
55                  pb = calcPb(in_constraints, in_propeller, in_sri,
                    in_gearbox, n, kq);
56                  etao = owc.Interpolate(j).etao;
57
58                  pbLimit = in_engine.Envelope.InterpolateNe(n * 60.0 *
                    in_gearbox.igb).Pb;
59
60                  if (in_engine.PTI != 0 && pb > 0.9 * pbLimit)
61                  {
62                      efmi = in_engine.FuelMap.Interpolate(fmc.NormalizePB
                    (0.9 * pbLimit, n * 60.0 * in_gearbox.igb),
                    fmc.NormalizeNE(n * 60.0 * in_gearbox.igb));
63                      sfc = fmc.SFCToFullScale(efmi.normsfc);
64                      fc = sfc * (0.9 * pbLimit) + in_engine.GenSFC * (pb -
                    0.9 * pbLimit) / in_engine.PTIloss;
65
66                      if ((owc.PD <= maxInternalPD && in_cii.LP < 10 && etao
                     < 1) || ii == steps)
67                      {
68                          owcItems.Add(owc.Interpolate(j));
69                          fcItems.Add(fc);
70                      }
71                  }
72                  else
73                  {
74                      efmi = in_engine.FuelMap.Interpolate(fmc.NormalizePB
                    (pb + in_engine.PTO, n * 60.0 * in_gearbox.igb),
                    fmc.NormalizeNE(n * 60.0 * in_gearbox.igb));
75                      sfc = fmc.SFCToFullScale(efmi.normsfc);
76                      fc = sfc * (pb + in_engine.PTO);
77
78                      if ((owc.PD <= maxInternalPD && pb < pbLimit &&
                    in_cii.LP < 10 && etao < 1) || ii == steps)
79                      {
80                          owcItems.Add(owc.Interpolate(j));
81                          fcItems.Add(fc);
82                      }
83                  }
84              }
85
86              minfc = fcItems.Min(y => y);
87              ind = fcItems.FindIndex(y => y == minfc);
88
89              minind = Math.Max(0, ind - 1);
90              maxind = Math.Min(ind + 1, fcItems.Count() - 1);
91
92              if (minind != maxind && owcItems[minind].J > 0)
93              {
94                  owci = owcItems[minind];
95                  nmin = in_sri.InflowSpeed / (owci.J * Dp) / maxNp;
```

```
 96
 97                      owci = owcItems[maxind];
 98                      nmax = in_sri.InflowSpeed / (owci.J * Dp) / maxNp;
 99                  }
100                  else break;
101              }
102
103              minfc = fcItems.Min(y => y);
104              ind = fcItems.FindIndex(y => y == minfc);
105
106              owci = owcItems[ind];
107
108              owc = in_propeller.OpenWaterCurveSet.Interpolate(owci.J, owci.Kt);
109              if ((owc.PD < maxInternalPD) || owcItems.Count() == 1)
110              {
111                  pd = owc.PD;
112                  if (owci.J > 0)
113                  {
114                      n = in_sri.InflowSpeed / (owci.J * Dp);
115                  }
116                  else
117                      n = minNp;
118
119                  j = owci.J;
120                  kt = owci.Kt;
121                  kq = owci.Kq;
122              }
123
124              // correction for engine load
125              kq /= pfact;
126              pb = calcPb(in_constraints, in_propeller, in_sri, in_gearbox, n,
                    kq);
127
128              // correction for pitch ratio
129              pd /= pdfact;
130
131              // calculations for performance indicators
132              // propeller efficiency
133              etao = owci.etao * pfact;
134              // cavitation
135              cavin = PropellerOWCItem.Cavin(in_constraints, in_propeller, n);
136              double j_nowake = in_sri.VS / (n * Dp);
137              double distanceToPressureSide = cavitationInceptionCore.GetDefault
                    ().GetDistanceInJPressureSide(EAR, pd, cavin, j_nowake);
138              //double aEff = Math.Atan(pd / (0.7 * Math.PI)) - Math.Atan
                    (in_propeller.cFactor * in_sri.InflowSpeed / (0.7 * Math.PI * n
                    * Dp)) - in_propeller.EntryAngle;
139              // engine efficiency & fuel consumption
140              efmi = in_engine.FuelMap.Interpolate(fmc.NormalizePB(pb +
                    in_engine.PTO, n * 60.0 * in_gearbox.igb), fmc.NormalizeNE(n *
                    60.0 * in_gearbox.igb));
141              sfc = fmc.SFCToFullScale(efmi.normsfc);
142              etae = 3600000 / (sfc * in_engine.LowerHeatingValue);
143              // fuel consumption
144              //fc = sfc * (pb + in_engine.PTO);
```

```
145
146
147             n *= 60;
148
149             return new CombinatorResultItem(in_cii.LP, in_sri.vs, pd, n, etao,
                    cavin, sfc, minfc, pb, etae, j, distanceToPressureSide);
150         }
151         private static double calcPb(CombinatorConstraints in_constraints,
              PropellerModel in_propeller, ShipResistanceItem sri, GearboxModel
              in_gearbox, double n, double kq)
152         {
153             return kq / 10 * in_constraints.EnvironmentalConstraints.RhoSw *
                  Math.Pow(n, 3) * Math.Pow(in_propeller.PropellerDiameter / 1000,
                   5) * 2.0 * Math.PI / in_gearbox.etaTRM / sri.etar / 1000;
154         }
155
156         private static double calcKT(CombinatorConstraints in_constraints,
              PropellerModel in_propeller, double n, double thrust)
157         {
158             return thrust / in_constraints.EnvironmentalConstraints.RhoSw /
                  Math.Pow(n, 2) / Math.Pow(in_propeller.PropellerDiameter / 1000,
                   4);
159         }
160     }
161 }
162
```

Simulator Core

...s\CombinatorGenerator\CombinatorCalcModel\Simulation.cs 1

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Drawing;
5  using System.Text;
6  using System.Threading.Tasks;
7  using static CombinatorCalcModel.defaultCore;
8  using System.IO;
9  using THPExtensions;
10 using static CombinatorCalcModel.Simulation;
11
12 namespace CombinatorCalcModel
13 {
14     public class Simulation
15     {
16         public static SimulationResultItem[] CalculateConstantSpeed
             (SimulationResultItem[] simri, CombinatorResultItem[]
             combinatorResults, ShipModel ship, PropellerModel propeller,
             EngineModel engine, GearboxModel gearbox,
17             CombinatorConstraints constraints, int constantLP, double
               constantTime, double accelerationTime, double decelerationTime,
               double constantDistance, double accelerationDistance, double
               decelerationDistance, double HotelLoad)
18         {
19             // correction factors
20             Dictionary<eFactorType, double> facts = defaultCore.getFactors
               (constraints, propeller, ship, engine, gearbox);
21             double pdfact = facts[eFactorType.PDfactor];
22             double pfact = facts[eFactorType.PowerFactor];
23
24             PropellerOWC owc;
25             ShipResistanceItem sri;
26             double Dp = propeller.PropellerDiameter / 1000d;
27             double SFC = 220;
28
29             List<SimulationResultItem> res = new List<SimulationResultItem>();
30
31             double TotalTime = simri.Last().TotalTime;
32             double TotalDistance = simri.Last().TotalDistance;
33             double TotalFuelConsumption = simri.Last().TotalFuelConsumption;
34
35             double PropulsionLoad = combinatorResults[constantLP].pb;
36             double VesselSpeed = combinatorResults[constantLP].vs * 1.852 /
                 3.6;
37             double PropSpeed = combinatorResults[constantLP].np / 60d;
38             double Pitch = combinatorResults[constantLP].pd * pdfact;
39             sri = ship.Resistance.Interpolate(VesselSpeed / 1.852 * 3.6);
40             double ThrustDemand = sri.RequiredThrust;
41             owc = propeller.OpenWaterCurveSet.Interpolate(Pitch);
42             double j = sri.InflowSpeed / (PropSpeed * Dp);
43             double kt = owc.Interpolate(j).Kt;
44             double Thrust = kt * constraints.EnvironmentalConstraints.RhoSw *
                 Math.Pow(PropSpeed, 2) * Math.Pow(Dp, 4) *
                 constraints.PropulsionConfigConstraints.Kp;
45             double pbLimit = engine.Envelope.InterpolateNe(PropSpeed * 60.0 *
                 gearbox.igb).Pb;
```

```
46
47              double PartTime = 0;
48              double PartDistance = 0;
49
50              if (constantTime != 0)
51              {
52
53                  PartTime = constantTime - accelerationTime - decelerationTime;
54
55                  TotalTime += PartTime;
56
57                  PartDistance = VesselSpeed * PartTime;
58                  TotalDistance += PartDistance;
59              }
60
61              if (constantDistance != 0)
62              {
63                  PartDistance = constantDistance - accelerationDistance -
                      decelerationDistance;
64                  TotalDistance += PartDistance;
65
66                  PartTime = PartDistance / VesselSpeed;
67                  TotalTime += PartTime;
68              }
69
70              double PropulsionConsumption = PartTime * combinatorResults
                  [constantLP].fc / 3600d;
71              double FuelConsumptionStep = PartTime * (HotelLoad / 0.95 * SFC /
                  3600d) + PropulsionConsumption;
72              TotalFuelConsumption += FuelConsumptionStep;
73
74              res.Add(new SimulationResultItem(TotalTime, PartTime,
                  TotalDistance, PartDistance, VesselSpeed, HotelLoad,
                  PropulsionLoad, TotalFuelConsumption, PropulsionConsumption,
                  FuelConsumptionStep, j, PropSpeed, Pitch, pbLimit, Thrust,
                  ThrustDemand));
75
76              return res.ToArray();
77          }
78
79          public static SimulationResultItem[] CalculateAcceleration
              (SimulationResultItem[] simri, CombinatorResultItem[]
              combinatorResults, ShipModel ship, PropellerModel propeller,
              EngineModel engine, GearboxModel gearbox,
80              CombinatorConstraints constraints, int CurrentLP, int RequiredLP,
                  double HotelLoad, double VesselDisplacement, double RampUpTime)
81          {
82              // correction factors
83              Dictionary<eFactorType, double> facts = defaultCore.getFactors
                  (constraints, propeller, ship, engine, gearbox);
84              double pdfact = facts[eFactorType.PDfactor];
85              double pfact = facts[eFactorType.PowerFactor];
86
87              PropellerOWC owc;
88              ShipResistanceItem sri;
89              EngineFuelMapItem efmi;
```

```
90              FuelMapConverter fmc = new FuelMapConverter(engine);
91              double Dp = propeller.PropellerDiameter / 1000d;
92
93              List<SimulationResultItem> res = new List<SimulationResultItem>();
94              double Operator = 1;
95              if (CurrentLP <= RequiredLP) { }
96              else { Operator = -1; }
97
98              double timeStep = 1;
99
100             double SFC = 220, sfc = 0; ; // TODO: interpolate
101             double Displacement = VesselDisplacement * 1.05;
102
103             double CurrentVesselSpeed = combinatorResults[CurrentLP].vs *
                  1.852 / 3.6;
104
105             double PropulsionLoad = combinatorResults[CurrentLP].pb;
106             double PropSpeed = combinatorResults[CurrentLP].np / 60d;
107             double Pitch = combinatorResults[CurrentLP].pd * pdfact;
108
109             sri = ship.Resistance.Interpolate(CurrentVesselSpeed / 1.852 *
                  3.6);
110             double ThrustDemand = sri.RequiredThrust;
111             owc = propeller.OpenWaterCurveSet.Interpolate(Pitch);
112             double j = sri.InflowSpeed / (PropSpeed * Dp);
113             double kt = owc.Interpolate(j).Kt;
114             double Thrust = kt * constraints.EnvironmentalConstraints.RhoSw *
                  Math.Pow(PropSpeed, 2) * Math.Pow(Dp, 4) *
                  constraints.PropulsionConfigConstraints.Kp;
115             double kq = 0;
116
117             double pbLimit = engine.Envelope.InterpolateNe(PropSpeed * 60.0 *
                  gearbox.igb).Pb;
118             double PropulsionConsumption = timeStep * combinatorResults
                  [CurrentLP].fc / 3600d;
119             double FuelConsumptionStep = timeStep * (HotelLoad / 0.95 * SFC /
                  3600d) + PropulsionConsumption;
120
121             double RequiredVesselSpeed = combinatorResults[RequiredLP].vs *
                  1.852 / 3.6;
122             if (RequiredLP > 0)
123             {
124
125             }
126             else if (RequiredLP == 0)
127             {
128                 RequiredVesselSpeed = 0.1;
129             }
130
131
132             double p1 = combinatorResults[CurrentLP].pd * pdfact;
133             double p2 = combinatorResults[RequiredLP].pd * pdfact;
134             double n1 = combinatorResults[CurrentLP].np / 60d;
135             double n2 = combinatorResults[RequiredLP].np / 60d;
136             double pitchstep = (p2 - p1) / (RampUpTime / timeStep);
137             double propspeedstep = (n2 - n1) / (RampUpTime / timeStep);
```

```
138
139             double Acceleration = 0;
140             double TotalTime = simri.Last().TotalTime;
141             double TotalDistance = simri.Last().TotalDistance;
142             double TotalFuelConsumption = simri.Last().TotalFuelConsumption;
143
144             double PartTime = 0;
145             double PartDistance = 0;
146
147
148             int iterationCounter = 0;
149
150             do
151             {
152                 owc = propeller.OpenWaterCurveSet.Interpolate(Pitch);
153                 j = sri.InflowSpeed / (PropSpeed * Dp);
154                 kq = owc.Interpolate(j).Kq / pfact;
155                 PropulsionLoad = calcPb(constraints, propeller, sri, gearbox,
                      PropSpeed, kq);
156
157                 if (Operator == 1)
158                 {
159
160                     if (PropSpeed >= n2)
161                     {
162                         PropSpeed = n2;
163                     }
164                     else
165                     {
166                         PropSpeed += propspeedstep;
167                     }
168
169                     if (Pitch >= p2)
170                     {
171                         Pitch = p2;
172                     }
173                     else if (PropulsionLoad >= pbLimit)
174                     {
175                         do
176                         {
177                             Pitch -= pitchstep;
178                             owc = propeller.OpenWaterCurveSet.Interpolate
                        (Pitch);
179                             j = sri.InflowSpeed / (PropSpeed * Dp);
180                             kq = owc.Interpolate(j).Kq / pfact;
181                             PropulsionLoad = calcPb(constraints, propeller,
                        sri, gearbox, PropSpeed, kq);
182
183                         } while (PropulsionLoad > pbLimit);
184
185                     }
186                     else
187                     {
188                         Pitch += pitchstep;
189                     }
190
```

```
191                 }
192
193                 if (Operator == -1)
194                 {
195
196                     if (PropSpeed <= n2)
197                     {
198                         PropSpeed = n2;
199                     }
200                     else
201                     {
202                         PropSpeed += propspeedstep;
203                     }
204
205                     if (Pitch <= p2)
206                     {
207                         Pitch = p2;
208                     }
209                     else if (PropulsionLoad <= 0)
210                     {
211                         do
212                         {
213                             Pitch += pitchstep;
214                             owc = propeller.OpenWaterCurveSet.Interpolate
                    (Pitch);
215                             j = sri.InflowSpeed / (PropSpeed * Dp);
216                             kq = owc.Interpolate(j).Kq / pfact;
217                             PropulsionLoad = calcPb(constraints, propeller,
                    sri, gearbox, PropSpeed, kq);
218
219                         } while (PropulsionLoad < 0);
220                     }
221                     else
222                     {
223                         Pitch += pitchstep;
224                     }
225
226
227                 }
228
229                 TotalTime += timeStep;
230                 PartTime += timeStep;
231
232
233                 sri = ship.Resistance.Interpolate(CurrentVesselSpeed / 1.852 *
                    3.6);
234                 ThrustDemand = sri.RequiredThrust;
235                 owc = propeller.OpenWaterCurveSet.Interpolate(Pitch);
236                 j = sri.InflowSpeed / (PropSpeed * Dp);
237                 kt = owc.Interpolate(j).Kt;
238                 Thrust = kt * constraints.EnvironmentalConstraints.RhoSw *
                    Math.Pow(PropSpeed, 2) * Math.Pow(Dp, 4) *
                    constraints.PropulsionConfigConstraints.Kp;
239                 Acceleration = Math.Abs(ThrustDemand - Thrust) /
                    (constraints.EnvironmentalConstraints.RhoSw * Displacement);
240                 CurrentVesselSpeed += Operator * Acceleration * timeStep;
```

```
241
242                 PartDistance += CurrentVesselSpeed * timeStep + 0.5 *
                      Acceleration * Math.Pow(timeStep, 2);
243                 TotalDistance += CurrentVesselSpeed * timeStep + 0.5 *
                      Acceleration * Math.Pow(timeStep, 2);
244
245
246                 kq = owc.Interpolate(j).Kq / pfact;
247                 PropulsionLoad = calcPb(constraints, propeller, sri, gearbox,
                      PropSpeed, kq);
248
249                 pbLimit = engine.Envelope.InterpolateNe(PropSpeed * 60.0 *
                      gearbox.igb).Pb;
250
251                 efmi = engine.FuelMap.Interpolate(fmc.NormalizePB
                      (PropulsionLoad, PropSpeed * 60.0 * gearbox.igb),
                      fmc.NormalizeNE(PropSpeed * 60.0 * gearbox.igb));
252                 sfc = fmc.SFCToFullScale(efmi.normsfc);
253                 PropulsionConsumption = timeStep * sfc * PropulsionLoad /
                      3600d;
254                 FuelConsumptionStep = timeStep * (HotelLoad / 0.95 * SFC /
                      3600d) + PropulsionConsumption;
255                 TotalFuelConsumption += FuelConsumptionStep;
256
257                 res.Add(new SimulationResultItem(TotalTime, PartTime,
                      TotalDistance, PartDistance, CurrentVesselSpeed, HotelLoad,
                      PropulsionLoad, TotalFuelConsumption, PropulsionConsumption,
                       FuelConsumptionStep, j, PropSpeed, Pitch, pbLimit, Thrust,
                      ThrustDemand));
258
259                 ++iterationCounter;
260
261
262             } while (Math.Abs(1 - Math.Pow((CurrentVesselSpeed /
                  RequiredVesselSpeed), Operator)) >= 0.005 && ++iterationCounter
                  < 10000);
263
264
265
266             return res.ToArray();
267
268         }
269
270
271         CombinatorResultItem[] combinatorResults;
272         ShipModel ship;
273         PropellerModel propeller;
274         EngineModel engine;
275         GearboxModel gearbox;
276         CombinatorConstraints constraints;
277
278         private double PT = 0;
279         private double MT = 0;
280         private double MaxT = 0;
281         private double CD = 0;
282
```

```
283         private double PL = 0;
284         private double ML = 0;
285         private double CL = 0;
286         private double VD = 0;
287         private int MLP = 0;
288         private double RUT = 0;
289         private double SAM = 0;
290
291         public double in_portStandbyTime { get { return PT * 3600; } set { PT = value / 3600; } }
292         public double in_portManeouvreTime { get { return MT * 3600; } set { MT = value / 3600; } }
293         public double MaxTripTime { get { return MaxT * 3600; } set { MaxT = value / 3600; } }
294         public double CruiseDistance { get { return CD * 1.852; } set { CD = value / 1.852; } }
295
296         public double in_portStandbyHotelload { get { return PL * 1000; } set { PL = value / 1000; } }
297         public double in_portManeouvreHotelLoad { get { return ML * 1000; } set { ML = value / 1000; } }
298         public double CruiseHotelLoad { get { return CL * 1000; } set { CL = value / 1000; } }
299
300         public double VesselDisplacement { get { return VD; } set { VD = value; } }
301         public double SurgeAddedMass { get { return SAM; } set { SAM = value; } }
302
303         public int ManeouvreLP { get { return MLP; } set { MLP = value; } }
304
305         public double RampUpTime { get { return RUT; } set { RUT = value; } }
306
307         public Simulation(CombinatorResultItem[] in_combinator, CombinatorProject in_project)
308         {
309             combinatorResults = in_combinator;
310             ship = in_project.Ship;
311             propeller = in_project.Propeller;
312             engine = in_project.Engine;
313             gearbox = in_project.Gearbox;
314             constraints = in_project.constraints;
315         }
316
317         public SimulationResult[] DoCalculation()
318         {
319             List<SimulationResult> resres = new List<SimulationResult>(); // final result for ALL LPs
320
321             SimulationResultItem[] accelerationResult;
322             SimulationResultItem[] decelerationResult;
323             SimulationResultItem[] maneouvreResultPortA;
324             SimulationResultItem[] maneouvreResultPortB;
325             SimulationResultItem[] cruiseResult;
326
327             // correction factors
```

```
328             Dictionary<eFactorType, double> facts = defaultCore.getFactors
                  (constraints, propeller, ship, engine, gearbox);
329             double pdfact = facts[eFactorType.PDfactor];
330
331             PropellerOWC owc;
332             ShipResistanceItem sritem;
333             double Dp = propeller.PropellerDiameter / 1000d;
334
335             double SFC = 220; // TODO: interpolate
336             double Displacement = VesselDisplacement * (1 + SurgeAddedMass /
                  100d);
337
338             List<int> cruiseLP = new List<int>() { 5, 6, 7, 8, 9, 10 }; //};
339
340             for (int i = 0; i < cruiseLP.Count; i++)
341             {
342                 List<SimulationResultItem> res = new
                      List<SimulationResultItem>(); //result per LP
343
344                 double VesselSpeed = 0;
345                 double TotalTime = 0;
346                 double PartTime = 0;
347
348                 double TotalDistance = 0;
349                 double PartDistance = 0;
350
351                 double TotalFuelConsumption = 0;
352                 double PropulsionConsumption = 0;
353                 double FuelConsumptionStep = 0;
354
355                 double accelerationDistance = 0, ManoeuvreAccelerationTime =
                      0;
356                 double decelerationDistance = 0, ManoeuvreDecelerationTime =
                      0;
357
358                 // start from portstandby
359                 double PropulsionLoad = combinatorResults[0].pb;
360                 double PropSpeed = combinatorResults[0].np / 60d;
361                 double Pitch = combinatorResults[0].pd * pdfact;
362                 sritem = ship.Resistance.Interpolate(VesselSpeed / 1.852 *
                      3.6);
363                 double ThrustDemand = sritem.RequiredThrust;
364                 owc = propeller.OpenWaterCurveSet.Interpolate(Pitch);
365                 double j = sritem.InflowSpeed / (PropSpeed * Dp);
366                 double kt = owc.Interpolate(j).Kt;
367                 double Thrust = kt *
                      constraints.EnvironmentalConstraints.RhoSw * Math.Pow
                      (PropSpeed, 2) * Math.Pow(Dp, 4) *
                      constraints.PropulsionConfigConstraints.Kp;
368                 double pbLimit = engine.Envelope.InterpolateNe(PropSpeed *
                      60.0 * gearbox.igb).Pb;
369
370
371                 res.Add(new SimulationResultItem(TotalTime, PartTime,
                      TotalDistance, PartDistance, VesselSpeed,
                      in_portStandbyHotelload, PropulsionLoad,
```

```
                    TotalFuelConsumption, PropulsionConsumption,
                    FuelConsumptionStep, j, PropSpeed, Pitch, pbLimit, Thrust,
                    ThrustDemand));
372
373                 // in port standby
374                 PartTime = in_portStandbyTime * 60d;
375                 TotalTime += PartTime;
376
377                 FuelConsumptionStep = PartTime * in_portStandbyHotelload /
                      0.95 * SFC / 3600d;
378                 TotalFuelConsumption += FuelConsumptionStep;
379
380                 res.Add(new SimulationResultItem(TotalTime, PartTime,
                      TotalDistance, PartDistance, VesselSpeed,
                      in_portStandbyHotelload, PropulsionLoad,
                      TotalFuelConsumption, PropulsionConsumption,
                      FuelConsumptionStep, j, PropSpeed, Pitch, pbLimit, Thrust,
                      ThrustDemand));
381
382
383
384                 // acceleration from standstil to maneouvrespeed
385                 accelerationResult = Simulation.CalculateAcceleration
                      (res.ToArray(), combinatorResults, ship, propeller, engine,
                      gearbox, constraints, 0, ManeouvreLP,
                      in_portManeouvreHotelLoad, Displacement, RampUpTime);
386                 res.AddRange(accelerationResult);
387
388                 // in port maneouvring
389                 ManoeuvreAccelerationTime = accelerationResult.Last
                      ().PartTime;
390                 maneouvreResultPortA = Simulation.CalculateConstantSpeed
                      (res.ToArray(), combinatorResults, ship, propeller, engine,
                      gearbox, constraints, ManeouvreLP, in_portManeouvreTime *
                      60d, ManoeuvreAccelerationTime, 0, 0, 0, 0,
                      in_portManeouvreHotelLoad);
391
392                 res.AddRange(maneouvreResultPortA);
393
394                 // acceleration from maneouvre to cruise
395                 accelerationResult = Simulation.CalculateAcceleration
                      (res.ToArray(), combinatorResults, ship, propeller, engine,
                      gearbox, constraints, ManeouvreLP, cruiseLP[i],
                      CruiseHotelLoad, Displacement, RampUpTime);
396                 res.AddRange(accelerationResult);
397
398
399                 // constant cruise speed
400                 accelerationDistance = accelerationResult.Last().PartDistance;
401
402                 decelerationResult = Simulation.CalculateAcceleration
                      (res.ToArray(), combinatorResults, ship, propeller, engine,
                      gearbox, constraints, cruiseLP[i], ManeouvreLP,
                      CruiseHotelLoad, Displacement, RampUpTime);
403                 decelerationDistance = decelerationResult.Last().PartDistance;
404
```

```
405                 cruiseResult = Simulation.CalculateConstantSpeed(res.ToArray
                      (), combinatorResults, ship, propeller, engine, gearbox,
                      constraints, cruiseLP[i], 0, 0, 0, CruiseDistance * 1.852 *
                      1000, accelerationDistance, decelerationDistance,
                      CruiseHotelLoad);
406
407                 res.AddRange(cruiseResult);
408
409
410                 // in port maneouvre time minus decelleration time
411                 decelerationResult = Simulation.CalculateAcceleration
                      (res.ToArray(), combinatorResults, ship, propeller, engine,
                      gearbox, constraints, ManeouvreLP, 0,
                      in_portManeouvreHotelLoad, Displacement, RampUpTime);
412                 ManoeuvreDecelerationTime = decelerationResult.Last
                      ().PartTime;
413
414                 if (ManoeuvreDecelerationTime >= in_portManeouvreTime * 60d)
415                 {
416                     decelerationResult = Simulation.CalculateAcceleration
                      (res.ToArray(), combinatorResults, ship, propeller,
                      engine, gearbox, constraints, cruiseLP[i], 0,
                      in_portManeouvreHotelLoad, Displacement, RampUpTime);
417                     res.AddRange(decelerationResult);
418                 }
419                 else
420                 {
421                     // decceleration from cruise to maneouvre
422                     decelerationResult = Simulation.CalculateAcceleration
                      (res.ToArray(), combinatorResults, ship, propeller,
                      engine, gearbox, constraints, cruiseLP[i], ManeouvreLP,
                      CruiseHotelLoad, Displacement, RampUpTime);
423                     res.AddRange(decelerationResult);
424
425                     maneouvreResultPortB = Simulation.CalculateConstantSpeed
                      (res.ToArray(), combinatorResults, ship, propeller,
                      engine, gearbox, constraints, ManeouvreLP,
                      in_portManeouvreTime * 60d, 0, ManoeuvreDecelerationTime,
                      0, 0, 0, in_portManeouvreHotelLoad); // TODO decelleration
                       is way to long, therefore hardecode for maneouvre time
426                     res.AddRange(maneouvreResultPortB);
427
428                     //maneouvre to standstill decelleration
429                     decelerationResult = Simulation.CalculateAcceleration
                      (res.ToArray(), combinatorResults, ship, propeller,
                      engine, gearbox, constraints, ManeouvreLP, 0,
                      in_portManeouvreHotelLoad, Displacement, RampUpTime);
430                     res.AddRange(decelerationResult);
431
432                 }
433
434
435
436                 resres.Add(new SimulationResult(cruiseLP[i].ToString(),
                      res.ToArray()));
437             }
```

```
438
439             return resres.ToArray();
440
441         }
442
443
444         private static double calcPb(CombinatorConstraints in_constraints,
              PropellerModel in_propeller, ShipResistanceItem sri, GearboxModel
              in_gearbox, double n, double kq)
445         {
446             return kq / 10 * in_constraints.EnvironmentalConstraints.RhoSw *
                  Math.Pow(n, 3) * Math.Pow(in_propeller.PropellerDiameter /
                  1000d, 5) * 2.0 * Math.PI / in_gearbox.etaTRM / sri.etar / 1000;
447         }
448     }
449 }
450
```

C

Test Cases

In this appendix, prints from the CCG with data and results for several test cases are included. These can be found on the next pages:

RoRo-Vessel

Combinator project

Ship Data

Type of ship		RoRo-Vessel
Maximum Vessel Speed	[kn]	22

Propeller Data

Diameter	[mm]	5000
Max P/D ratio	[-]	1,436
Propeller Draft	[mm]	4000
AeAo	[-]	0,427

Gearbox Data

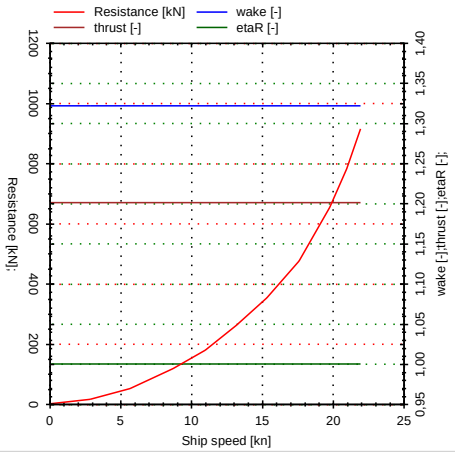
Gearbox ratio	[-]	4,1841
Transmission efficiency	[-]	0,95

Propulsion Configuration Data

Nr. of propellers	[-]	2
Nr. of engines per shaft	[-]	1
Power Take Off	[kW]	0
Power Take In	[kW]	0

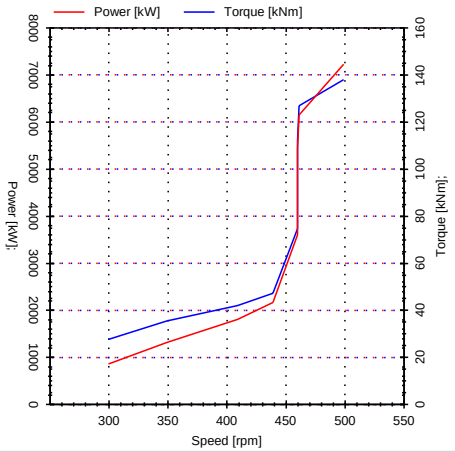
Resistance Data

vs [kn]	R_T [kN]	w [-]	t [-]	Eta_R [-]
0,0	0,0	0,132	0,120	1,000
2,9	12,1	0,132	0,120	1,000
5,7	48,5	0,132	0,120	1,000
8,8	114,7	0,132	0,120	1,000
11,0	179,4	0,132	0,120	1,000
13,2	258,1	0,132	0,120	1,000
15,4	351,5	0,132	0,120	1,000
17,6	472,1	0,132	0,120	1,000
19,8	650,9	0,132	0,120	1,000
21,0	779,3	0,132	0,120	1,000
22,0	915,1	0,132	0,120	1,000



Engine Envelope

Speed [rpm]	Power [kW]	Torque [kNm]
300	855	27
350	1292	35
410	1795	42
440	2156	47
450	2868	61
460	3590	75
461	5395	112
462	6117	126
500	7200	138



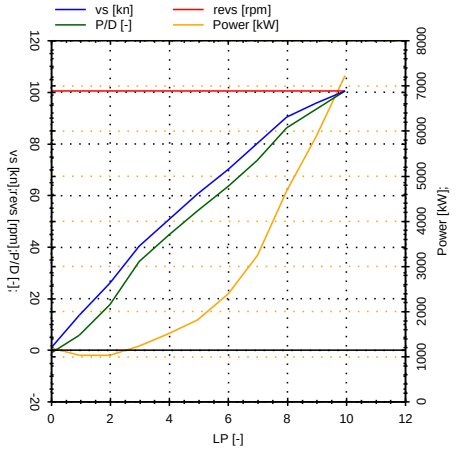
Combinator Result

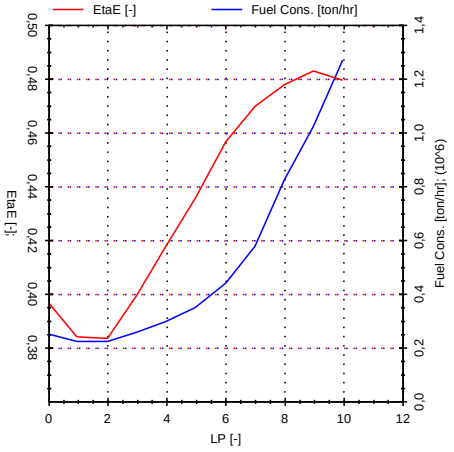
Combinator Settings

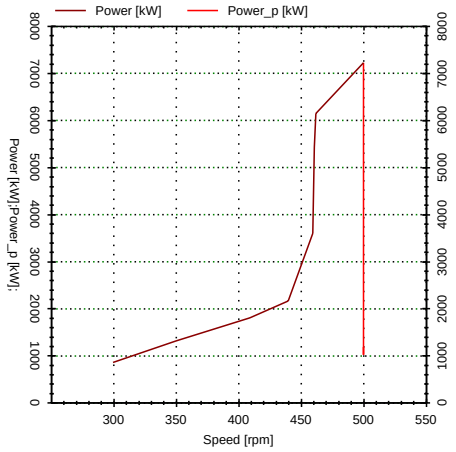
LP [-]	vs [kn]	revs [rpm]	P/D [-]	P_p [kW]
0	0,0	500	-0,022	1179
1	2,9	500	0,076	1019
2	5,7	500	0,253	1013
3	8,8	500	0,485	1215
4	11,0	500	0,633	1476
5	13,2	500	0,768	1801
6	15,4	500	0,911	2359
7	17,6	500	1,057	3210
8	19,8	500	1,229	4661
9	21,0	500	1,336	5849
10	22,0	500	1,436	7200

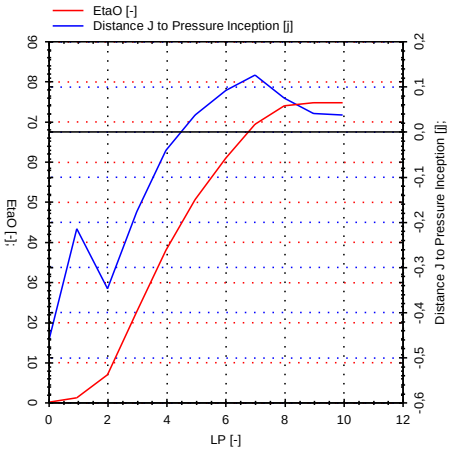
Performance

EtaO [-]	DeltaJ [j]	EtaE [-]	Fuel Co. [ton/hr]
0,000	0,47	0,397	0,25
0,011	0,22	0,384	0,22
0,069	0,35	0,383	0,22
0,224	0,18	0,400	0,26
0,377	0,04	0,417	0,30
0,504	-0,04	0,435	0,35
0,607	-0,09	0,456	0,44
0,691	-0,12	0,470	0,58
0,740	-0,07	0,478	0,82
0,747	-0,04	0,483	1,02
0,747	-0,04	0,479	1,27









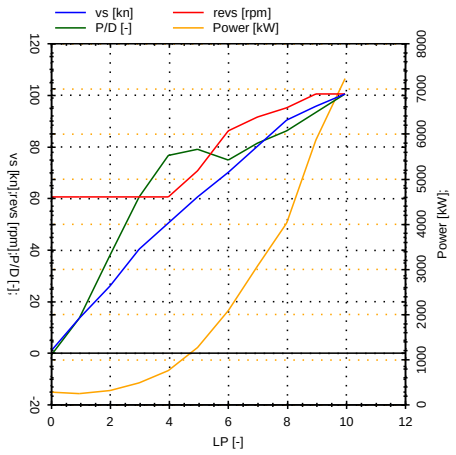
Combinator Result

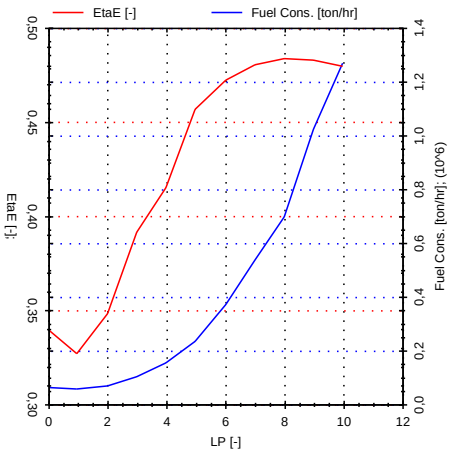
Combinator Settings

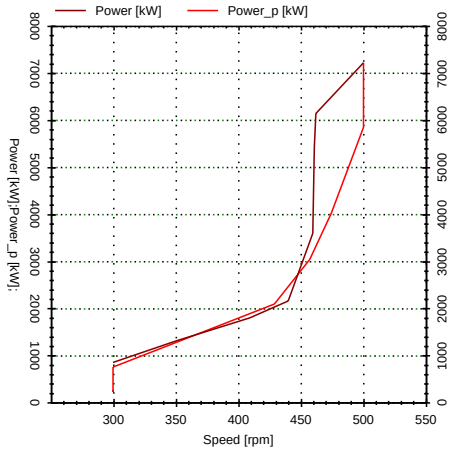
LP [-]	vs [kn]	revs [rpm]	P/D [-]	P_p [kW]
0	0,0	300	-0,022	255
1	2,9	300	0,192	217
2	5,7	300	0,537	281
3	8,8	300	0,863	464
4	11,0	300	1,098	744
5	13,2	351	1,130	1264
6	15,4	430	1,071	2072
7	17,6	457	1,167	3040
8	19,8	475	1,229	4001
9	21,0	500	1,336	5849
10	22,0	500	1,436	7198

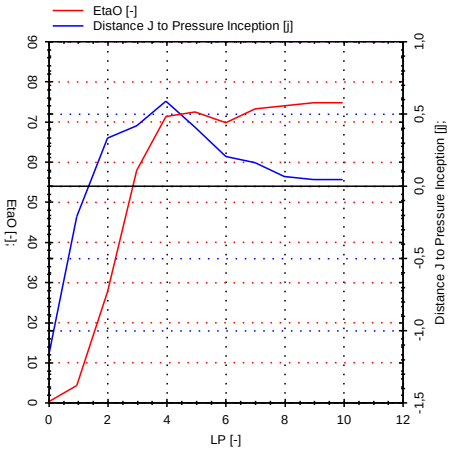
Performance

EtaO [-]	DeltaJ [j]	EtaE [-]	Fuel Co. [ton/hr]
0,000	1,22	0,340	0,06
0,042	0,21	0,327	0,06
0,276	-0,33	0,348	0,07
0,576	-0,41	0,391	0,10
0,709	-0,58	0,415	0,15
0,721	-0,40	0,456	0,23
0,698	-0,20	0,472	0,37
0,731	-0,16	0,480	0,53
0,740	-0,06	0,484	0,70
0,747	-0,04	0,483	1,02
0,747	-0,04	0,479	1,27









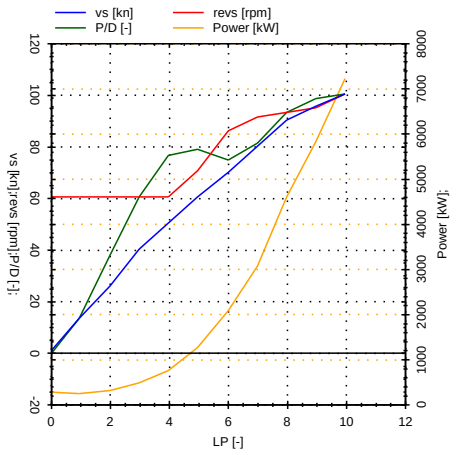
Combinator Result

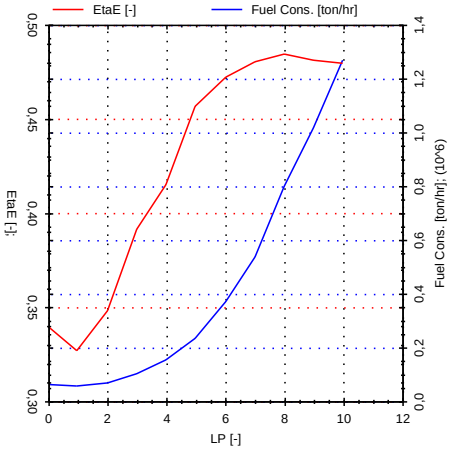
Combinator Settings

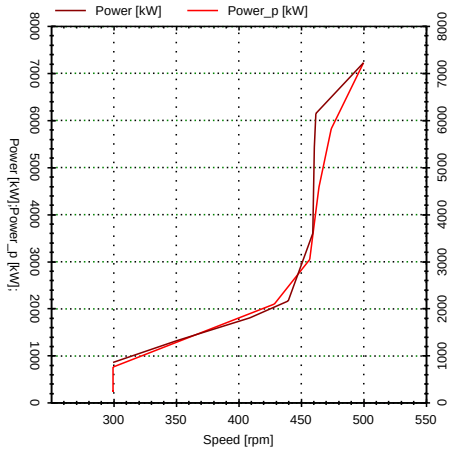
LP [-]	vs [kn]	revs [rpm]	P/D [-]	P_p [kW]
0	0,0	300	-0,019	255
1	2,9	300	0,192	217
2	5,7	300	0,542	281
3	8,8	300	0,861	464
4	11,0	300	1,099	744
5	13,2	351	1,130	1264
6	15,4	430	1,071	2073
7	17,6	457	1,167	3041
8	19,8	465	1,337	4572
9	21,0	475	1,417	5800
10	22,0	500	1,436	7200

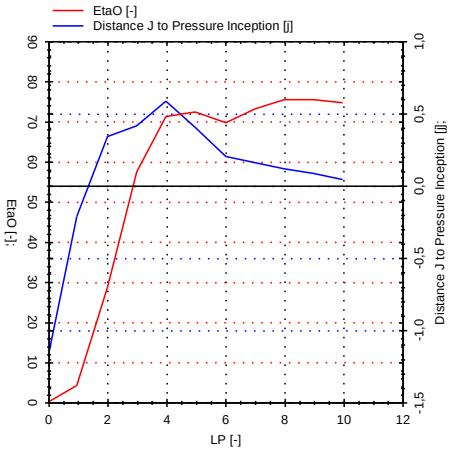
Performance

EtaO [-]	DeltaJ [j]	EtaE [-]	Fuel Co. [ton/hr]
0,000	1,20	0,340	0,06
0,042	0,22	0,327	0,06
0,287	-0,34	0,348	0,07
0,574	-0,41	0,391	0,10
0,709	-0,58	0,415	0,15
0,721	-0,40	0,456	0,23
0,698	-0,20	0,472	0,37
0,731	-0,16	0,480	0,53
0,752	-0,11	0,484	0,80
0,754	-0,08	0,481	1,02
0,747	-0,04	0,479	1,27









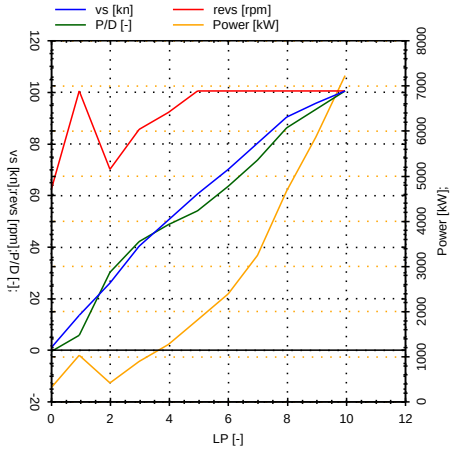
Combinator Result

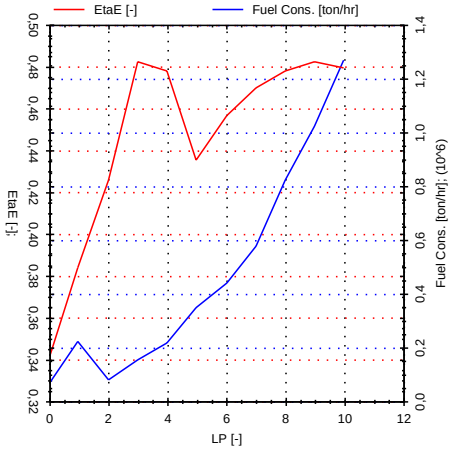
Combinator Settings

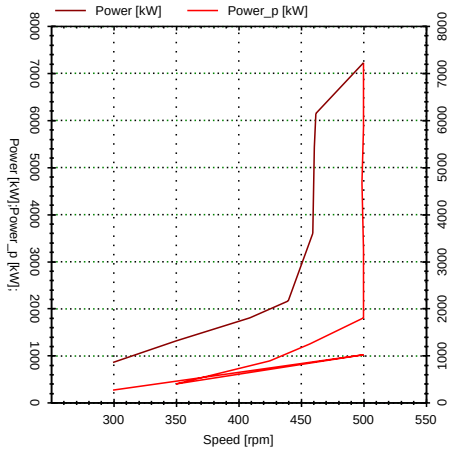
LP [-]	vs [kn]	revs [rpm]	P/D [-]	P_p [kW]
0	0,0	300	-0,019	255
1	2,9	500	0,079	1019
2	5,7	350	0,431	391
3	8,8	425	0,599	860
4	11,0	458	0,698	1236
5	13,2	500	0,768	1801
6	15,4	500	0,909	2359
7	17,6	500	1,057	3210
8	19,8	500	1,231	4660
9	21,0	500	1,336	5849
10	22,0	500	1,436	7200

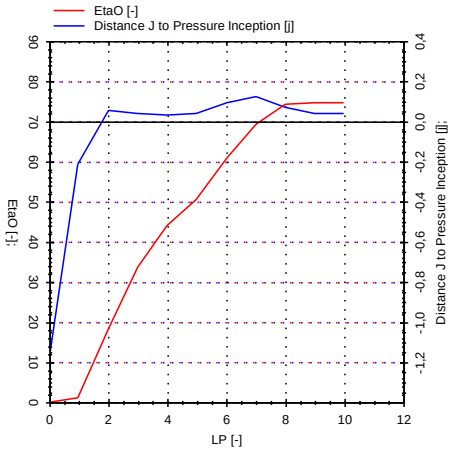
Performance

EtaO [-]	DeltaJ [j]	EtaE [-]	Fuel Co. [ton/hr]
0,000	1,20	0,340	0,06
0,011	0,21	0,384	0,22
0,182	-0,05	0,426	0,08
0,335	-0,04	0,482	0,15
0,439	-0,03	0,478	0,22
0,504	-0,04	0,435	0,35
0,607	-0,09	0,456	0,44
0,691	-0,12	0,470	0,58
0,740	-0,07	0,478	0,82
0,747	-0,04	0,483	1,02
0,747	-0,04	0,479	1,27









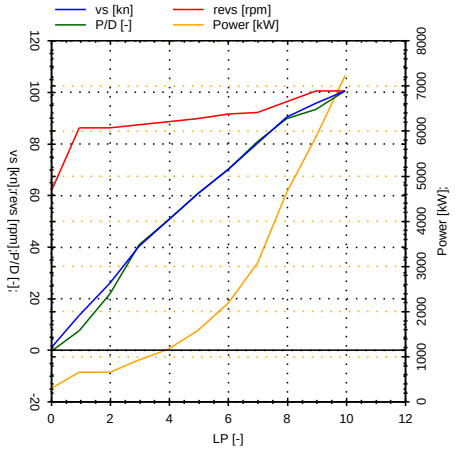
Combinator Result

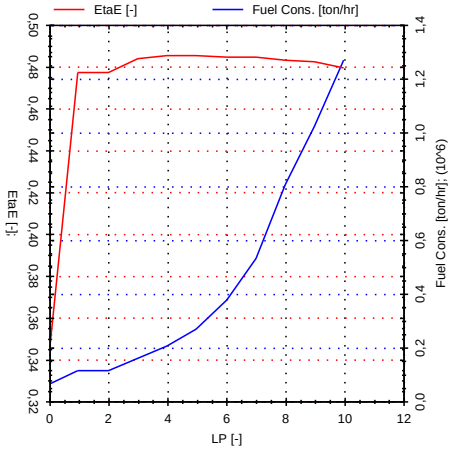
Combinator Settings

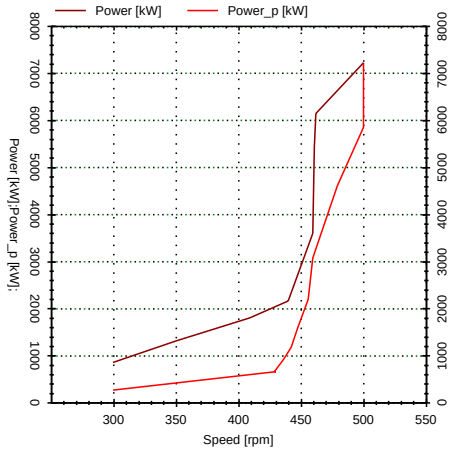
LP [-]	vs [kn]	revs [rpm]	P/D [-]	P_p [kW]
0	0,0	300	-0,019	255
1	2,9	430	0,101	644
2	5,7	430	0,311	651
3	8,8	437	0,581	911
4	11,0	443	0,722	1159
5	13,2	447	0,867	1552
6	15,4	456	1,005	2171
7	17,6	460	1,158	3052
8	19,8	480	1,288	4606
9	21,0	500	1,336	5849
10	22,0	500	1,436	7200

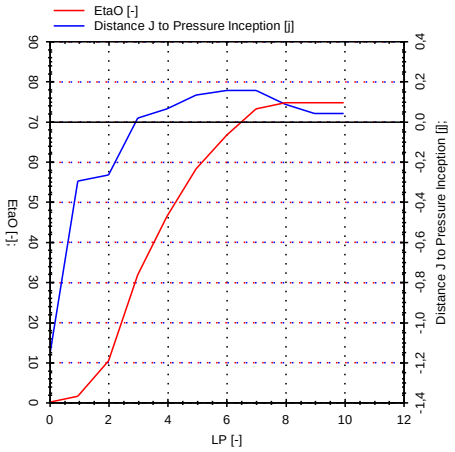
Performance

EtaO [-]	DeltaJ [j]	EtaE [-]	Fuel Co. [ton/hr]
0,000	1,20	0,340	0,06
0,015	0,30	0,477	0,11
0,104	0,27	0,477	0,11
0,315	-0,01	0,483	0,16
0,463	-0,06	0,485	0,20
0,580	-0,13	0,485	0,27
0,664	-0,15	0,484	0,38
0,729	-0,15	0,484	0,53
0,747	-0,08	0,483	0,80
0,747	-0,04	0,483	1,02
0,747	-0,04	0,479	1,27









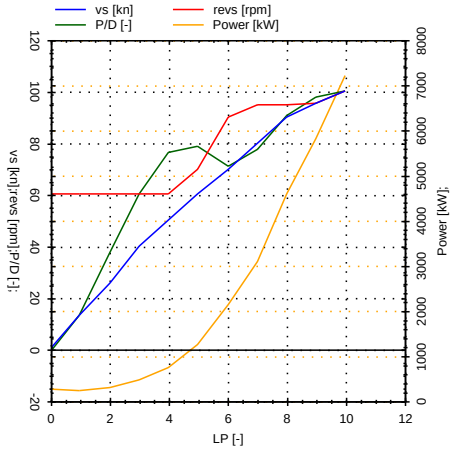
Combinator Result

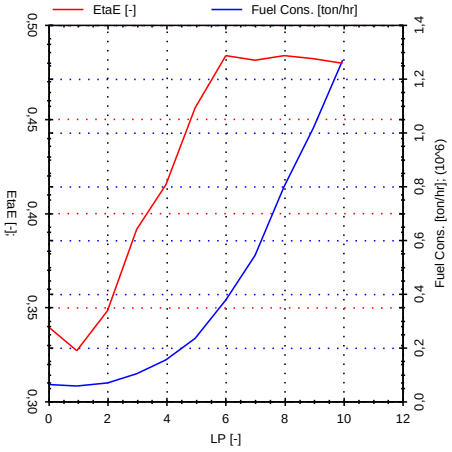
Combinator Settings

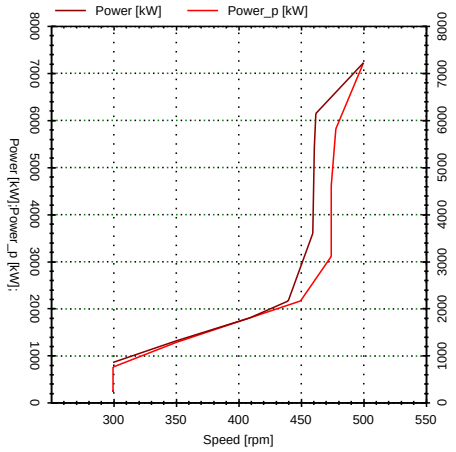
LP [-]	vs [kn]	revs [rpm]	P/D [-]	P_p [kW]
0	0,0	300	-0,019	255
1	2,9	300	0,192	217
2	5,7	300	0,542	281
3	8,8	300	0,861	464
4	11,0	300	1,099	744
5	13,2	350	1,133	1263
6	15,4	450	1,019	2147
7	17,6	475	1,118	3103
8	19,8	475	1,304	4595
9	21,0	478	1,406	5805
10	22,0	500	1,436	7200

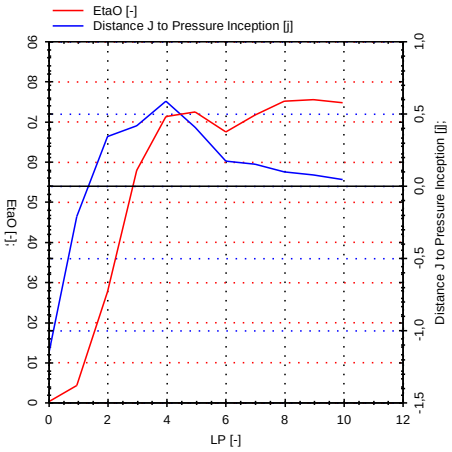
Performance

EtaO [-]	DeltaJ [j]	EtaE [-]	Fuel Co. [ton/hr]
0,000	1,20	0,340	0,06
0,042	0,22	0,327	0,06
0,276	-0,34	0,348	0,07
0,576	-0,41	0,391	0,10
0,709	-0,58	0,415	0,15
0,722	-0,41	0,456	0,23
0,672	-0,17	0,483	0,37
0,716	-0,14	0,481	0,54
0,749	-0,09	0,484	0,80
0,753	-0,07	0,482	1,02
0,747	-0,04	0,479	1,27









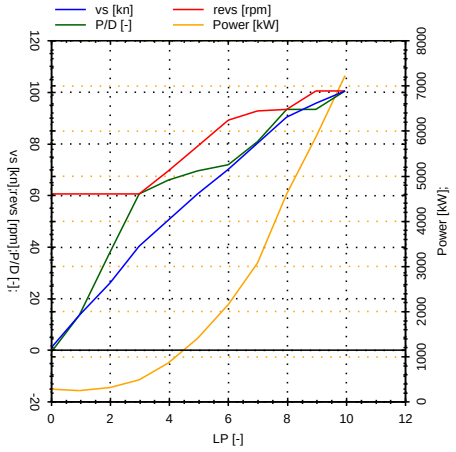
Combinator Result

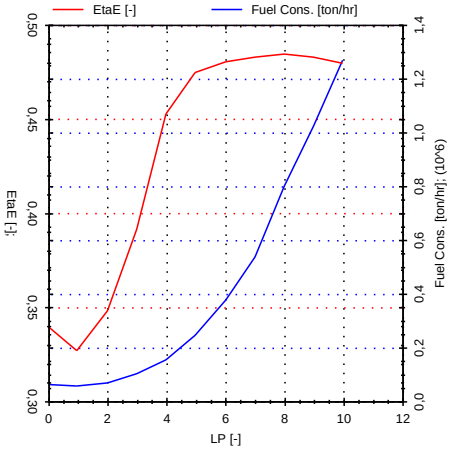
Combinator Settings

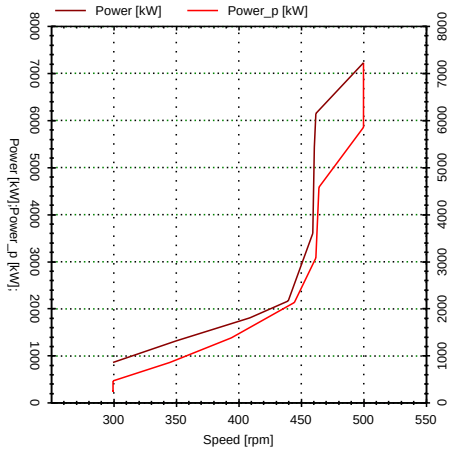
LP [-]	vs [kn]	revs [rpm]	P/D [-]	P_p [kW]
0	0,0	300	-0,022	255
1	2,9	300	0,192	217
2	5,7	300	0,537	281
3	8,8	300	0,863	464
4	11,0	345	0,946	833
5	13,2	395	0,994	1379
6	15,4	445	1,032	2128
7	17,6	462	1,153	3057
8	19,8	465	1,336	4573
9	21,0	500	1,336	5849
10	22,0	500	1,436	7200

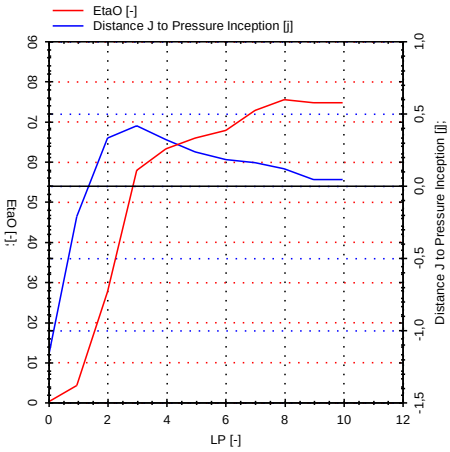
Performance

EtaO [-]	DeltaJ [j]	EtaE [-]	Fuel Co. [ton/hr]
0,000	1,22	0,340	0,06
0,042	0,21	0,327	0,06
0,276	-0,33	0,348	0,07
0,576	-0,41	0,391	0,10
0,629	-0,31	0,452	0,16
0,657	-0,23	0,475	0,24
0,678	-0,18	0,480	0,37
0,727	-0,15	0,483	0,53
0,752	-0,11	0,484	0,80
0,747	-0,04	0,483	1,02
0,747	-0,04	0,479	1,27









Container

Combinator project

Ship Data

Type of ship		JHContainer
Maximum Vessel Speed	[kn]	18,4

Propeller Data

Diameter	[mm]	5800
Max P/D ratio	[-]	1,011
Propeller Draft	[mm]	4500
AeAo	[-]	0,56

Gearbox Data

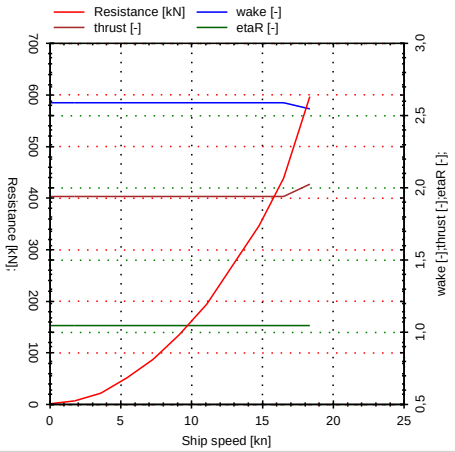
Gearbox ratio	[-]	1
Transmission efficiency	[-]	0,9

Propulsion Configuration Data

Nr. of propellers	[-]	1
Nr. of engines per shaft	[-]	1
Power Take Off	[kW]	0
Power Take In	[kW]	0

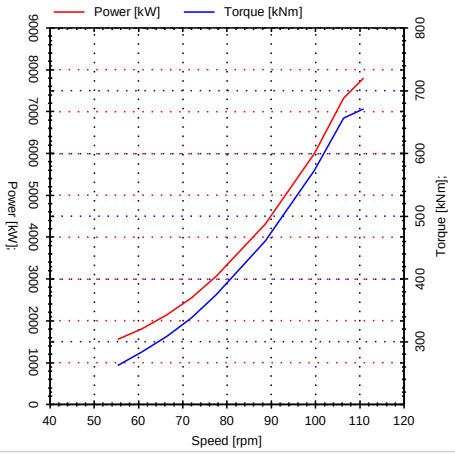
Resistance Data

vs [kn]	R_T [kN]	w [-]	t [-]	Eta_R [-]
0,0	0,0	0,258	0,193	1,045
1,8	5,4	0,258	0,193	1,045
3,7	21,5	0,258	0,193	1,045
5,5	48,4	0,258	0,193	1,045
7,4	86,1	0,258	0,193	1,045
9,2	134,4	0,258	0,193	1,045
11,1	193,4	0,258	0,193	1,045
12,9	263,6	0,258	0,193	1,045
14,8	344,0	0,258	0,193	1,045
16,6	435,7	0,258	0,193	1,045
18,4	594,7	0,254	0,202	1,045



Engine Envelope

Speed [rpm]	Power [kW]	Torque [kNm]
56	1515	261
61	1803	282
67	2137	306
72	2540	336
78	3030	372
89	4273	460
100	5983	572
107	7305	655
111	7770	668



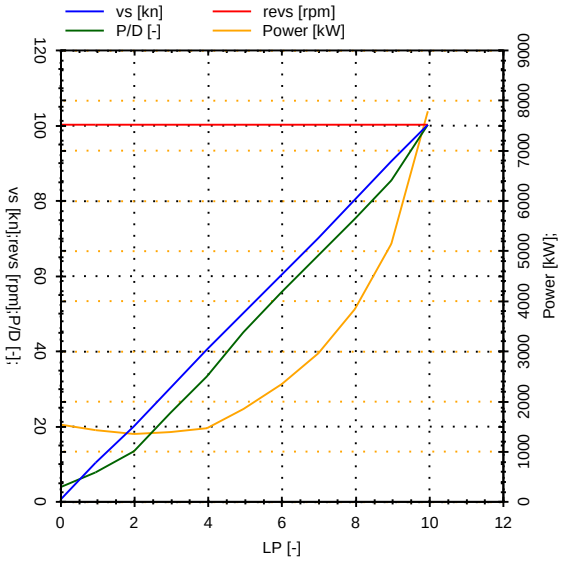
Combinator Result

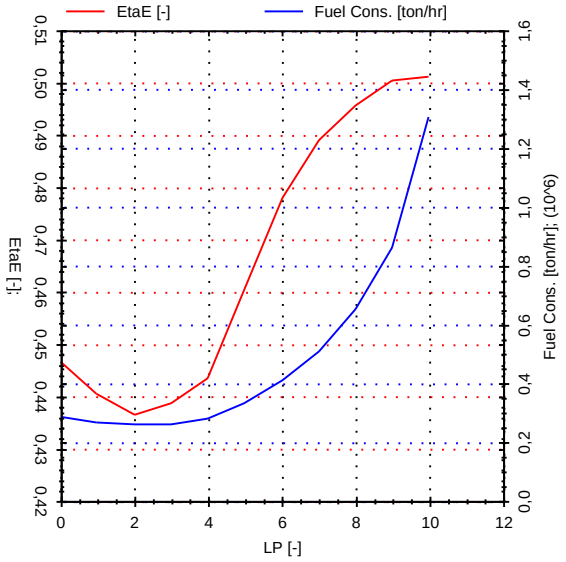
Combinator Settings

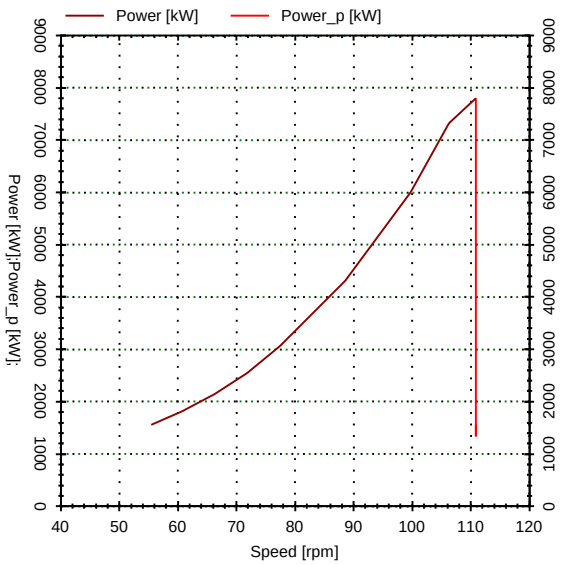
LP [-]	vs [kn]	revs [rpm]	P/D [-]	P_p [kW]
0	0,0	111	0,033	1517
1	1,8	111	0,075	1398
2	3,7	111	0,135	1327
3	5,5	111	0,235	1364
4	7,4	111	0,334	1453
5	9,2	111	0,454	1811
6	11,0	111	0,562	2330
7	12,9	111	0,659	2949
8	14,7	111	0,757	3821
9	16,6	111	0,862	5116
10	18,4	111	1,011	7770

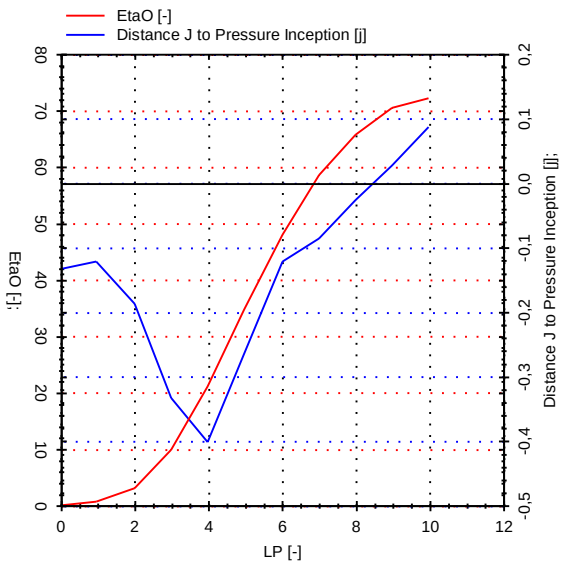
Performance

EtaO [-]	DeltaJ [j]	EtaE [-]	Fuel Cons. [ton/hr]
0,000	0,13	0,447	0,29
0,007	0,12	0,440	0,27
0,030	0,19	0,436	0,26
0,099	0,33	0,439	0,26
0,209	0,40	0,443	0,28
0,346	0,27	0,460	0,33
0,479	0,12	0,478	0,41
0,583	0,09	0,489	0,51
0,656	0,03	0,495	0,65
0,703	-0,03	0,500	0,86
0,720	-0,09	0,501	1,31









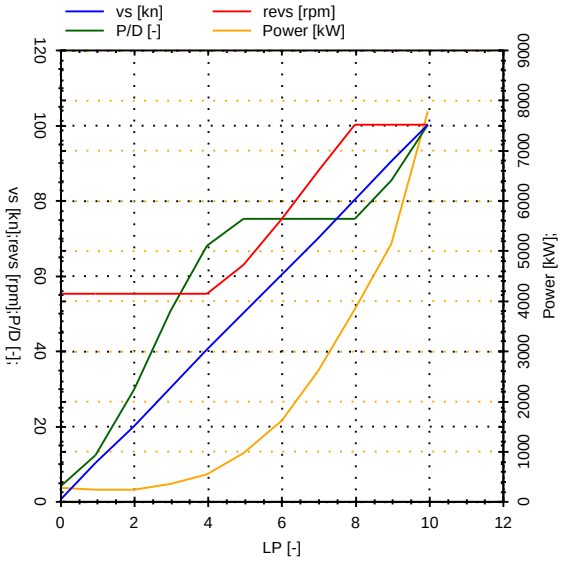
Combinator Result

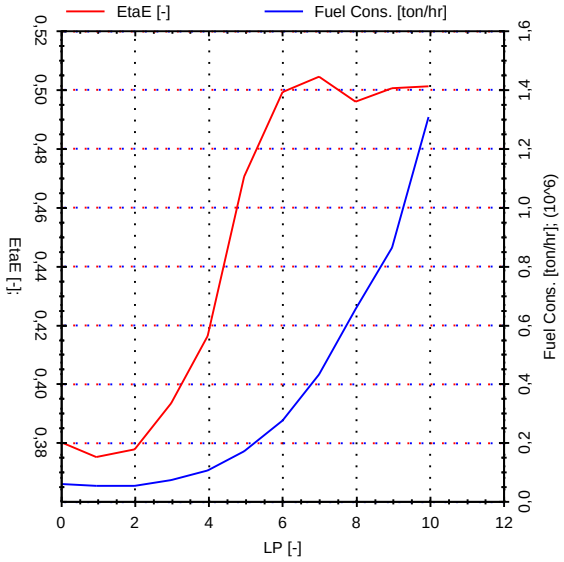
Combinator Settings

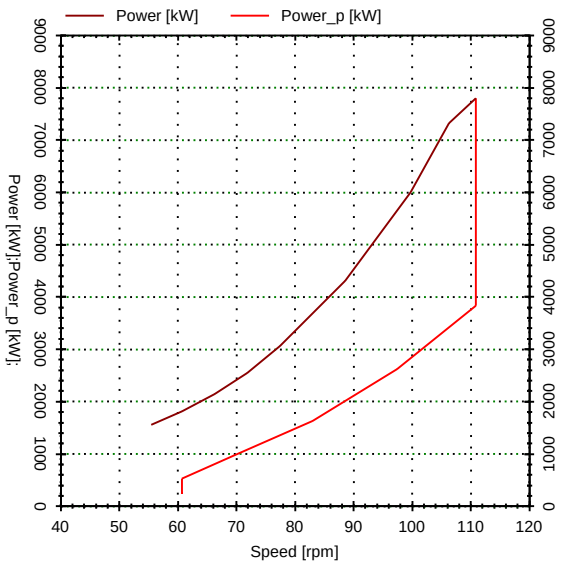
LP [-]	vs [kn]	revs [rpm]	P/D [-]	P_p [kW]
0	0,0	61	0,033	252
1	1,8	61	0,123	221
2	3,7	61	0,298	234
3	5,5	61	0,509	342
4	7,4	61	0,684	521
5	9,2	70	0,755	944
6	11,0	83	0,755	1610
7	12,9	97	0,755	2584
8	14,7	111	0,757	3821
9	16,6	111	0,862	5116
10	18,4	111	1,011	7767

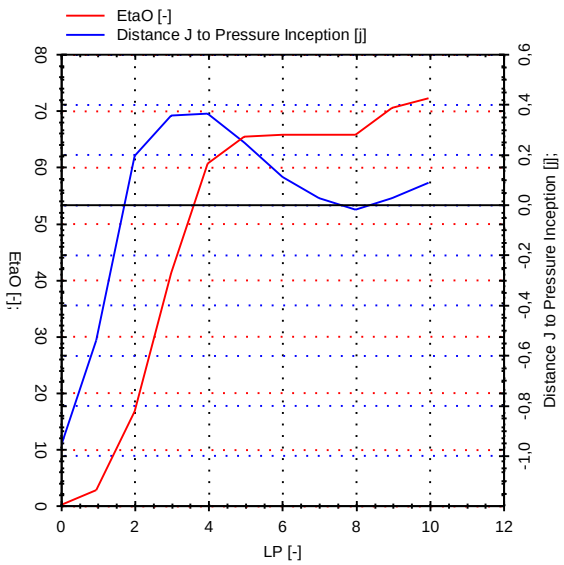
Performance

EtaO [-]	DeltaJ [j]	EtaE [-]	Fuel Cons. [ton/hr]
0,000	0,98	0,380	0,06
0,025	0,54	0,375	0,05
0,165	-0,19	0,377	0,05
0,412	-0,35	0,393	0,07
0,606	-0,36	0,416	0,11
0,654	-0,24	0,470	0,17
0,655	-0,11	0,499	0,27
0,654	-0,03	0,504	0,43
0,656	0,03	0,495	0,65
0,703	-0,03	0,500	0,86
0,720	-0,09	0,501	1,31









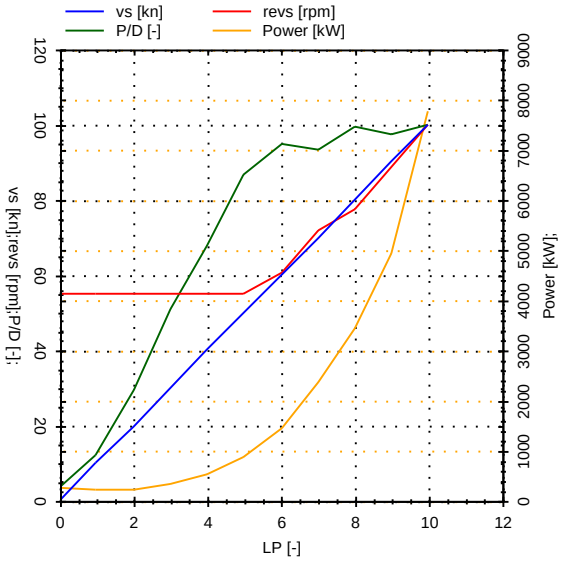
Combinator Result

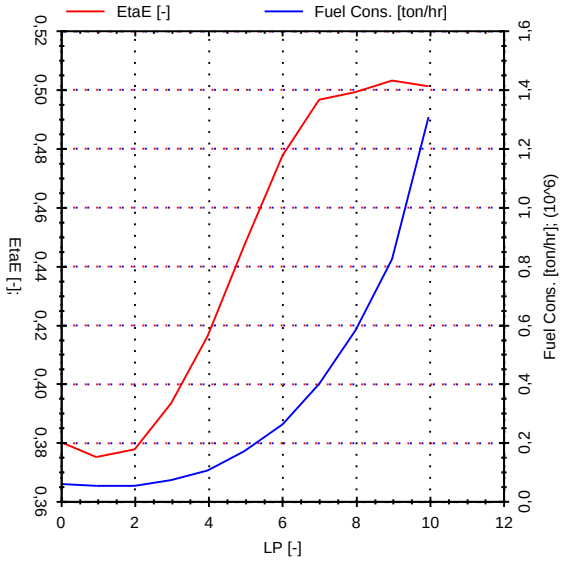
Combinator Settings

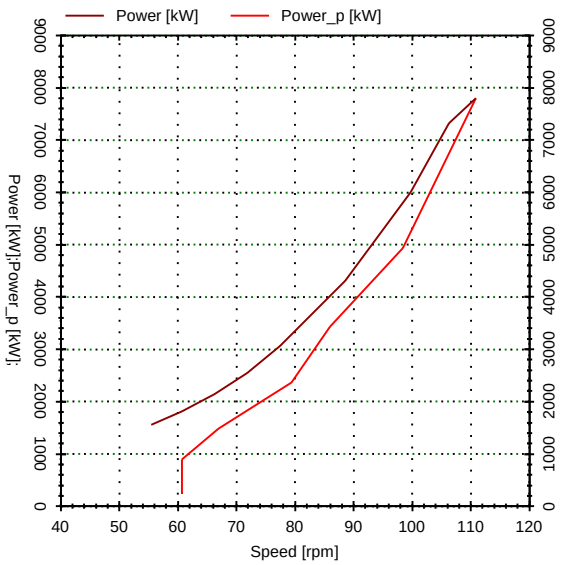
LP [-]	vs [kn]	revs [rpm]	P/D [-]	P_p [kW]
0	0,0	61	0,035	252
1	1,8	61	0,125	221
2	3,7	61	0,296	234
3	5,5	61	0,515	342
4	7,4	61	0,684	521
5	9,2	61	0,874	879
6	11,0	67	0,958	1455
7	12,9	80	0,943	2346
8	14,7	86	1,005	3422
9	16,6	99	0,986	4912
10	18,4	111	1,011	7770

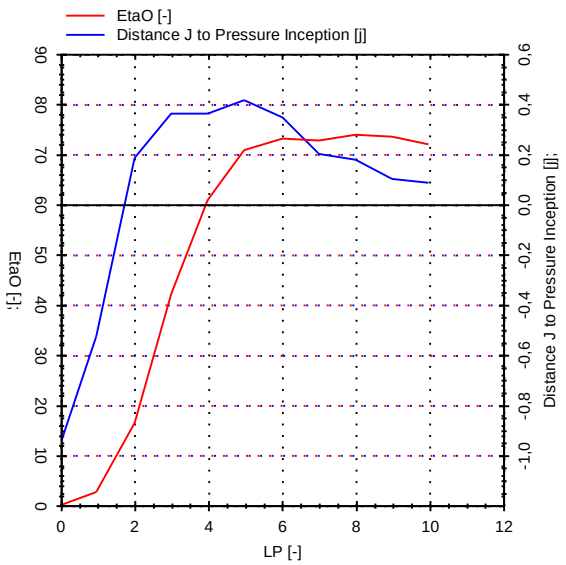
Performance

EtaO [-]	DeltaJ [j]	EtaE [-]	Fuel Cons. [ton/hr]
0,000	0,96	0,380	0,06
0,027	0,53	0,375	0,05
0,162	-0,19	0,377	0,05
0,421	-0,36	0,393	0,07
0,606	-0,36	0,416	0,11
0,706	-0,41	0,446	0,17
0,730	-0,34	0,477	0,26
0,726	-0,20	0,497	0,40
0,737	-0,18	0,499	0,58
0,734	-0,10	0,503	0,82
0,720	-0,09	0,501	1,31









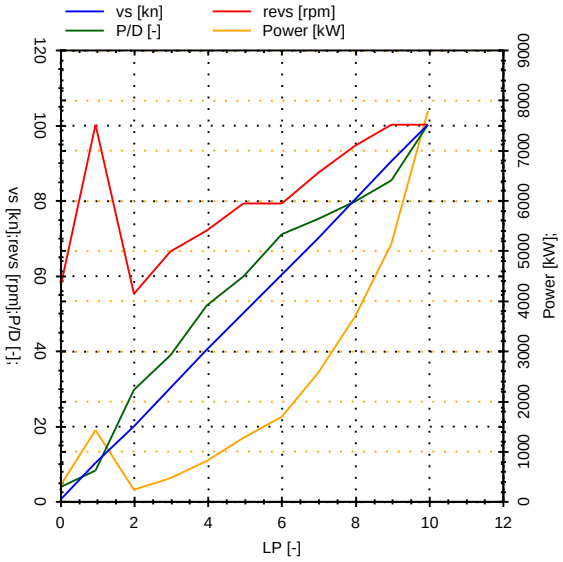
Combinator Result

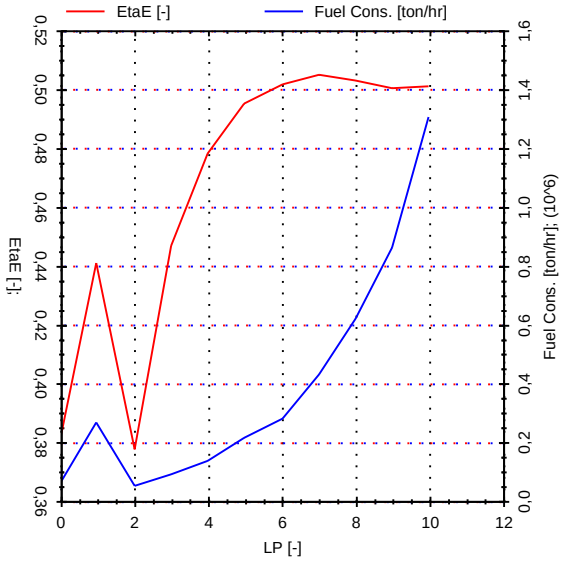
Combinator Settings

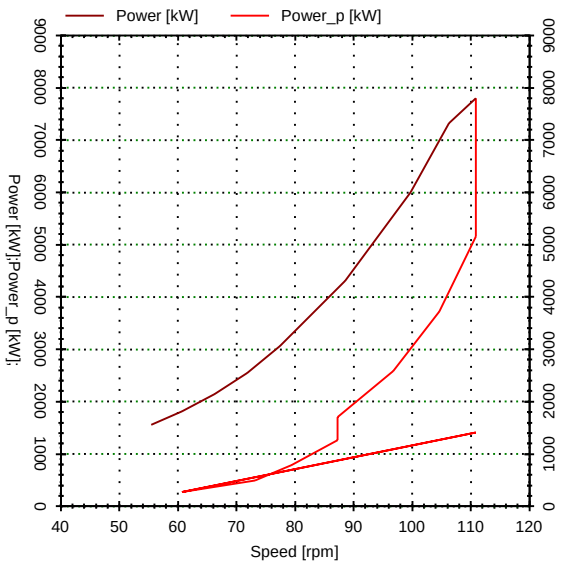
LP [-]	vs [kn]	revs [rpm]	P/D [-]	P_p [kW]
0	0,0	61	0,035	252
1	1,8	111	0,080	1398
2	3,7	61	0,296	234
3	5,5	74	0,392	464
4	7,4	80	0,525	779
5	9,2	88	0,603	1246
6	11,0	88	0,715	1678
7	12,9	97	0,758	2574
8	14,7	105	0,805	3684
9	16,6	111	0,862	5116
10	18,4	111	1,011	7770

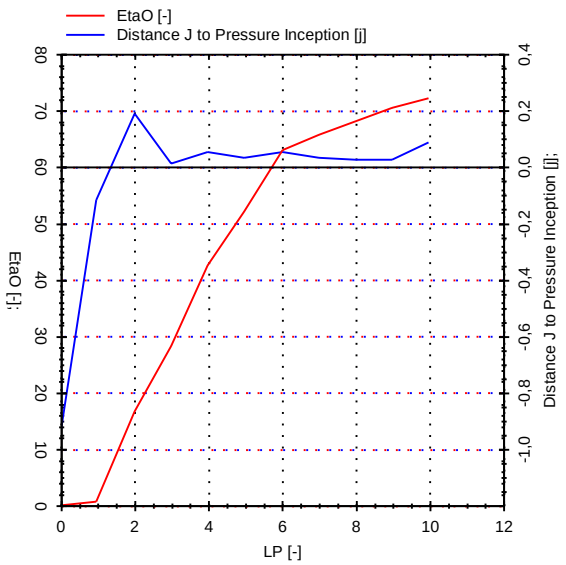
Performance

EtaO [-]	DeltaJ [j]	EtaE [-]	Fuel Cons. [ton/hr]
0,000	0,96	0,380	0,06
0,007	0,12	0,440	0,27
0,165	-0,19	0,377	0,05
0,281	-0,01	0,447	0,09
0,423	-0,05	0,478	0,14
0,519	-0,03	0,495	0,21
0,630	-0,05	0,501	0,28
0,657	-0,03	0,505	0,43
0,681	-0,02	0,503	0,62
0,703	-0,02	0,500	0,86
0,720	-0,09	0,501	1,31









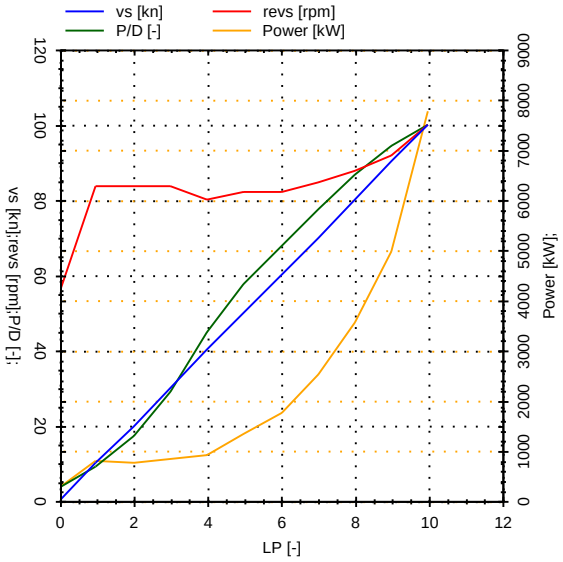
Combinator Result

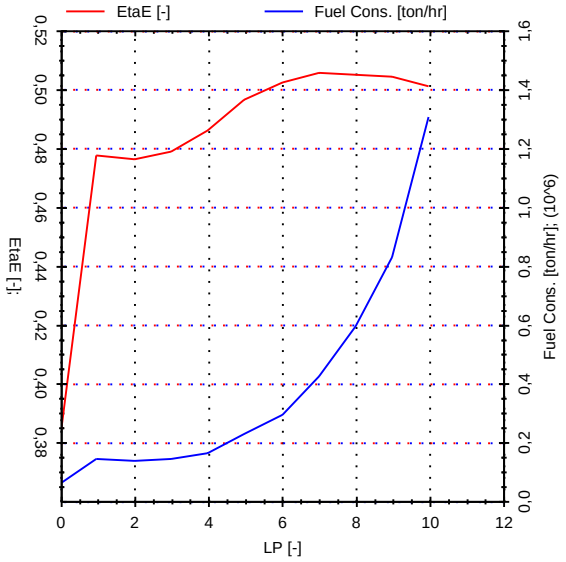
Combinator Settings

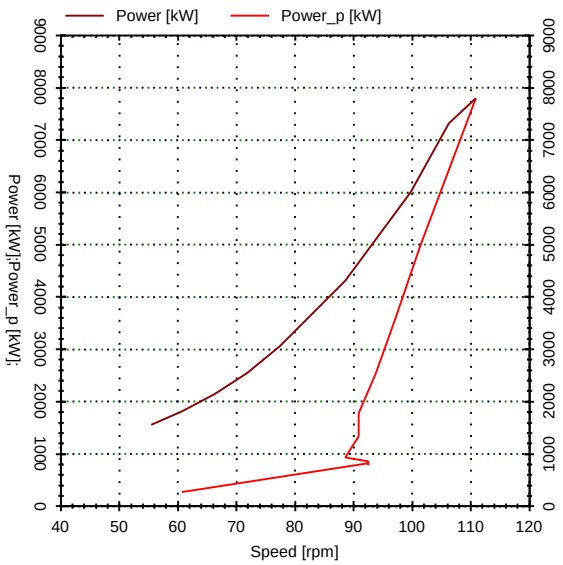
LP [-]	vs [kn]	revs [rpm]	P/D [-]	P_p [kW]
0	0,0	61	0,035	252
1	1,8	93	0,090	805
2	3,7	93	0,172	772
3	5,5	93	0,293	819
4	7,4	89	0,454	923
5	9,2	91	0,581	1323
6	11,0	91	0,688	1746
7	12,9	94	0,784	2522
8	14,7	97	0,875	3556
9	16,6	102	0,952	4952
10	18,4	111	1,011	7770

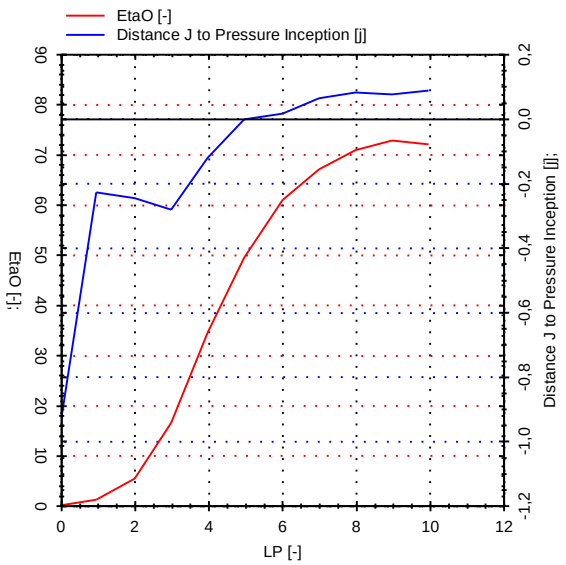
Performance

EtaO [-]	DeltaJ [j]	EtaE [-]	Fuel Cons [ton/hr]
0,000	0,96	0,380	0,06
0,010	0,23	0,478	0,14
0,052	0,25	0,476	0,14
0,162	0,29	0,478	0,14
0,343	0,12	0,486	0,16
0,492	0,00	0,497	0,22
0,609	-0,01	0,502	0,29
0,670	-0,06	0,505	0,42
0,708	-0,08	0,505	0,59
0,728	-0,07	0,504	0,83
0,720	-0,09	0,501	1,31









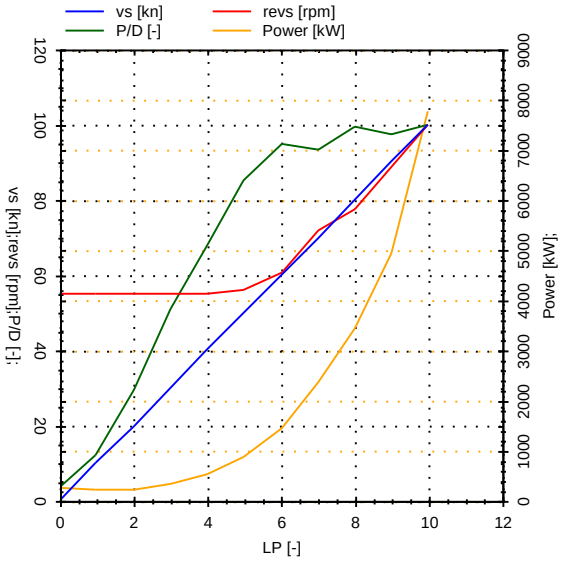
Combinator Result

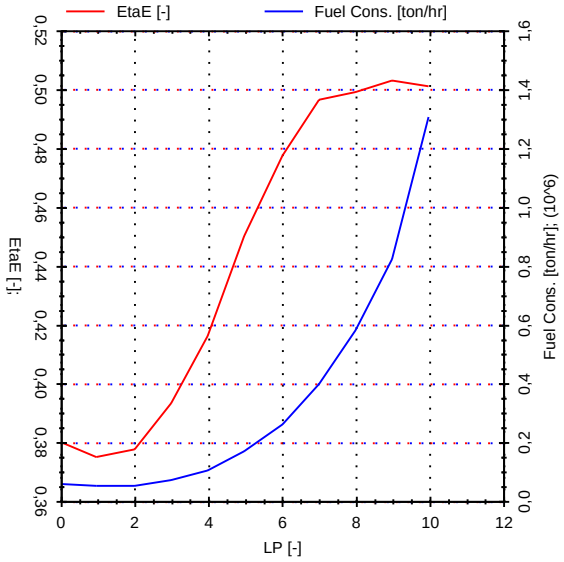
Combinator Settings

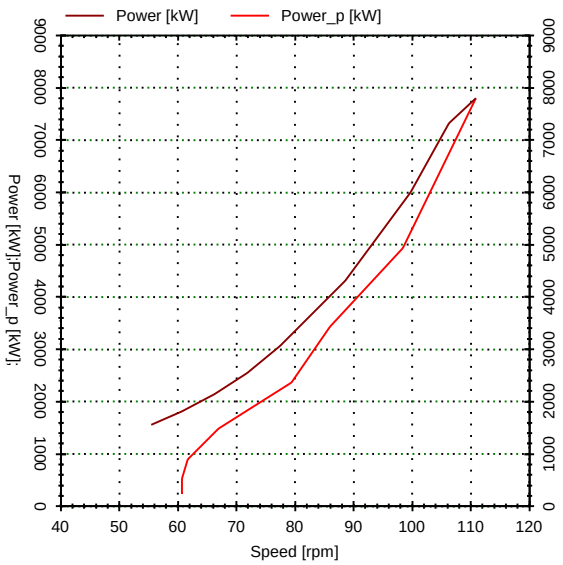
LP [-]	vs [kn]	revs [rpm]	P/D [-]	P_p [kW]
0	0,0	61	0,035	252
1	1,8	61	0,125	221
2	3,7	61	0,296	234
3	5,5	61	0,515	342
4	7,4	61	0,684	521
5	9,2	62	0,858	885
6	11,0	67	0,958	1455
7	12,9	80	0,943	2346
8	14,7	86	1,005	3422
9	16,6	99	0,986	4912
10	18,4	111	1,011	7770

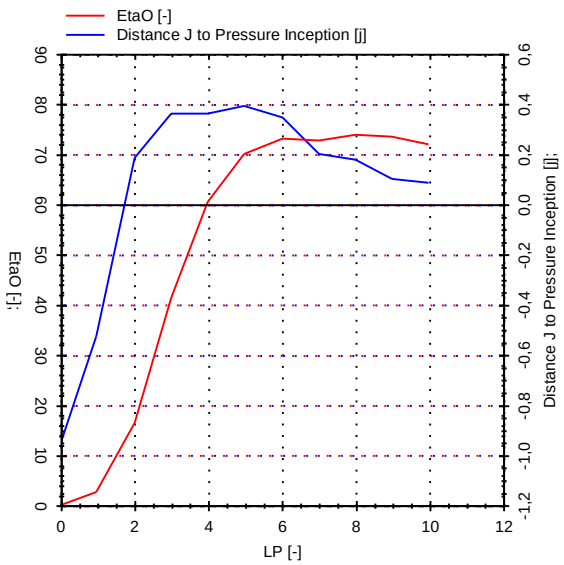
Performance

EtaO [-]	DeltaJ [j]	EtaE [-]	Fuel Cons. [ton/hr]
0,000	0,96	0,380	0,06
0,025	0,53	0,375	0,05
0,165	-0,19	0,377	0,05
0,412	-0,36	0,393	0,07
0,606	-0,36	0,416	0,11
0,701	-0,39	0,450	0,17
0,730	-0,34	0,477	0,26
0,726	-0,20	0,497	0,40
0,737	-0,18	0,499	0,58
0,734	-0,10	0,503	0,82
0,720	-0,09	0,501	1,31









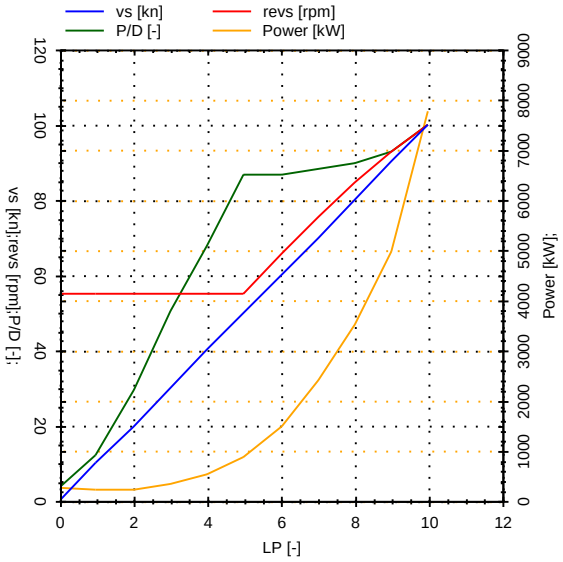
Combinator Result

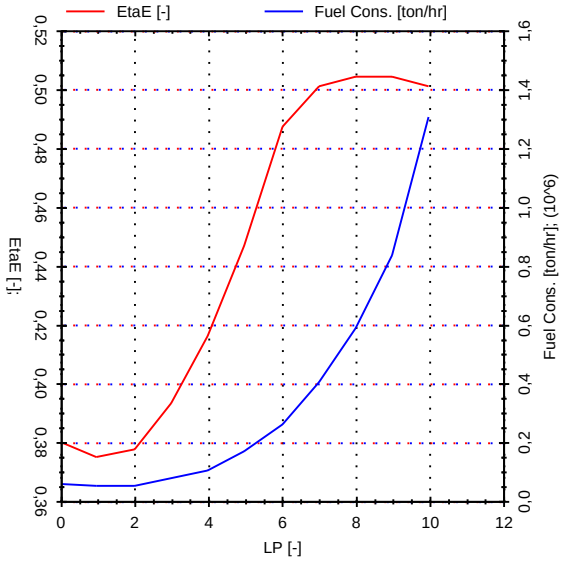
Combinator Settings

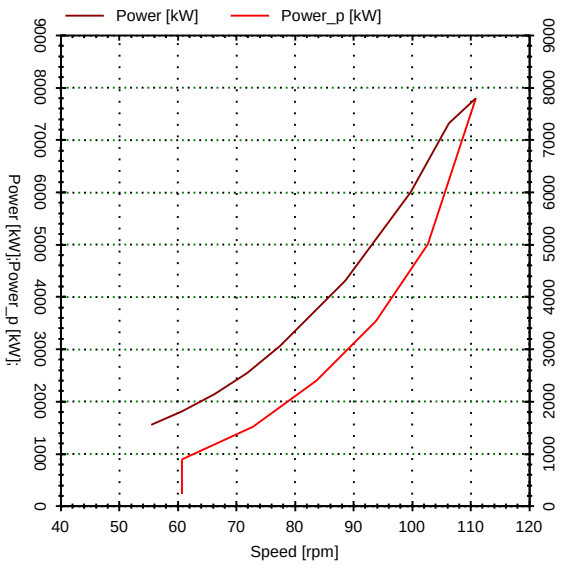
LP [-]	vs [kn]	revs [rpm]	P/D [-]	P_p [kW]
0	0,0	61	0,033	252
1	1,8	61	0,123	221
2	3,7	61	0,298	234
3	5,5	61	0,508	343
4	7,4	61	0,684	521
5	9,2	61	0,874	879
6	11,0	73	0,874	1498
7	12,9	84	0,890	2389
8	14,7	94	0,909	3510
9	16,6	103	0,937	4974
10	18,4	111	1,011	7770

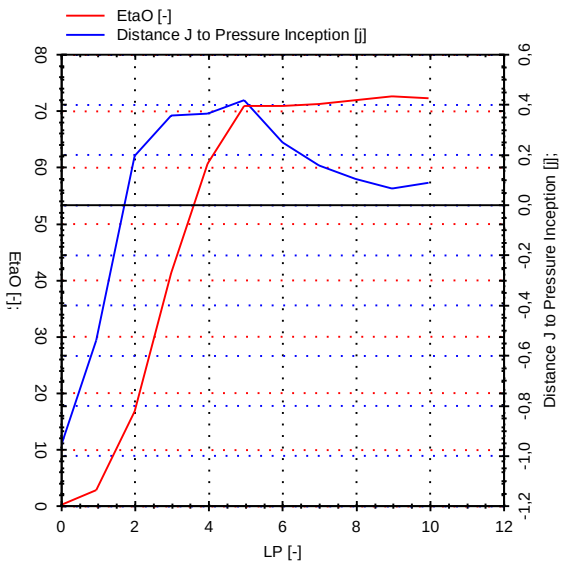
Performance

EtaO [-]	DeltaJ [j]	EtaE [-]	Fuel Cons [ton/hr]
0,000	0,98	0,380	0,06
0,025	0,54	0,375	0,05
0,165	-0,19	0,377	0,05
0,412	-0,35	0,393	0,07
0,606	-0,36	0,416	0,11
0,706	-0,41	0,447	0,17
0,707	-0,25	0,487	0,26
0,711	-0,16	0,501	0,40
0,718	-0,10	0,504	0,59
0,725	-0,06	0,504	0,83
0,720	-0,09	0,501	1,31









OPV

Combinator project

Ship Data

Type of ship		OPV
Maximum Vessel Speed	[kn]	21,3

Propeller Data

Diameter	[mm]	2300
Max P/D ratio	[-]	1,255
Propeller Draft	[mm]	2600
AeAo	[-]	0,733

Gearbox Data

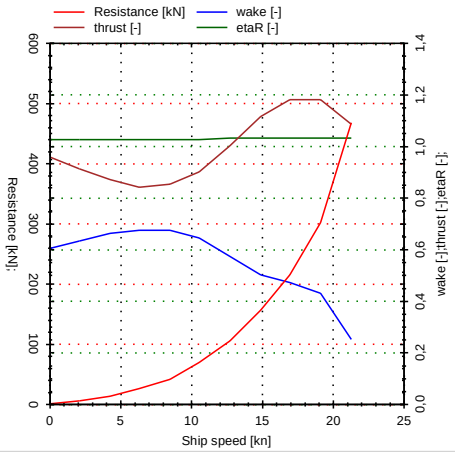
Gearbox ratio	[-]	2,8249
Transmission efficiency	[-]	0,96

Propulsion Configuration Data

Nr. of propellers	[-]	2
Nr. of engines per shaft	[-]	1
Power Take Off	[kW]	0
Power Take In	[kW]	0

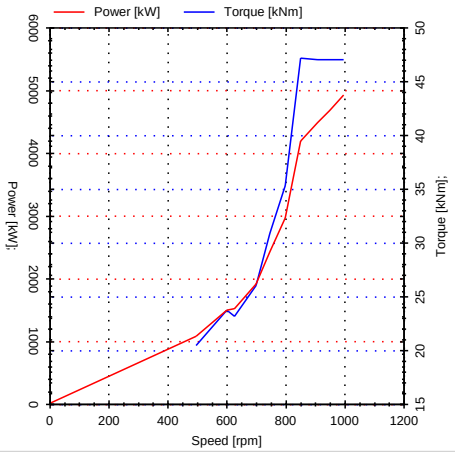
Resistance Data

vs [kn]	R_T [kN]	w [-]	t [-]	Eta_R [-]
0,0	0,0	0,060	0,096	1,025
2,1	5,0	0,063	0,091	1,024
4,3	12,2	0,066	0,087	1,023
6,4	23,6	0,067	0,084	1,023
8,5	41,4	0,067	0,085	1,023
10,6	67,7	0,064	0,090	1,024
12,8	104,8	0,057	0,100	1,026
14,9	154,9	0,050	0,111	1,027
17,0	214,5	0,047	0,118	1,029
19,2	301,1	0,043	0,118	1,030
21,3	466,6	0,025	0,108	1,028



Engine Envelope

Speed [rpm]	Power [kW]	Torque [kNm]
0	0	
500	1064	20
600	1484	24
630	1520	23
700	1900	26
750	2424	31
800	2948	35
850	4192	47
910	4478	47
950	4674	47
1000	4920	47



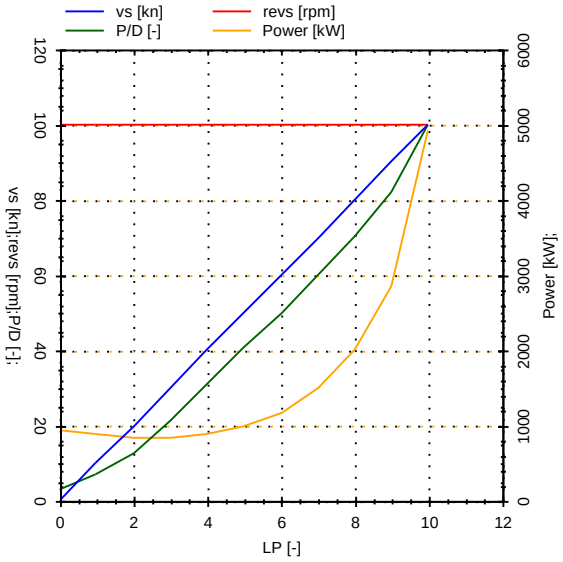
Combinator Result

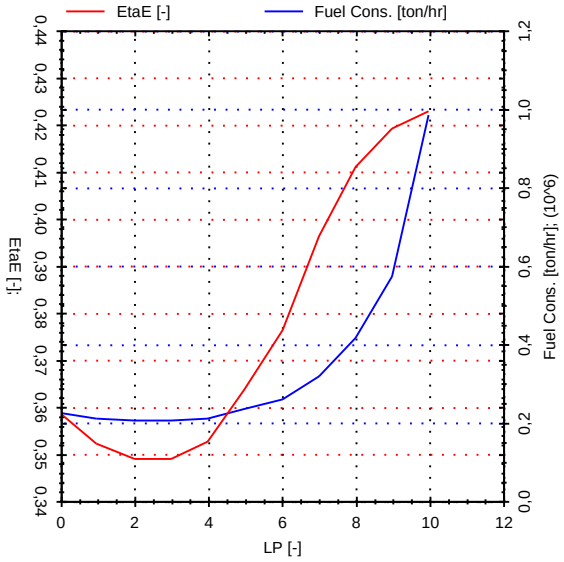
Combinator Settings

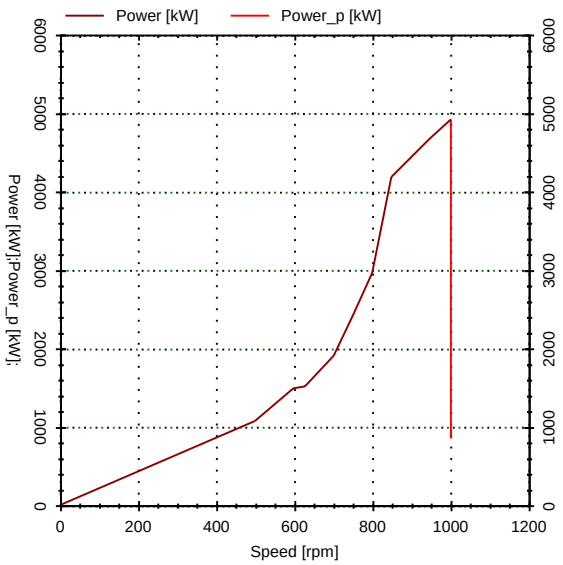
LP [-]	vs [kn]	revs [rpm]	P/D [-]	P_p [kW]
0	0,0	1000	0,038	944
1	2,1	1000	0,089	876
2	4,3	1000	0,161	840
3	6,4	1000	0,267	844
4	8,5	1000	0,386	880
5	10,7	1000	0,512	1000
6	12,8	1000	0,627	1162
7	14,9	1000	0,755	1497
8	17,0	1000	0,884	2014
9	19,2	1000	1,027	2850
10	21,3	1000	1,255	4920

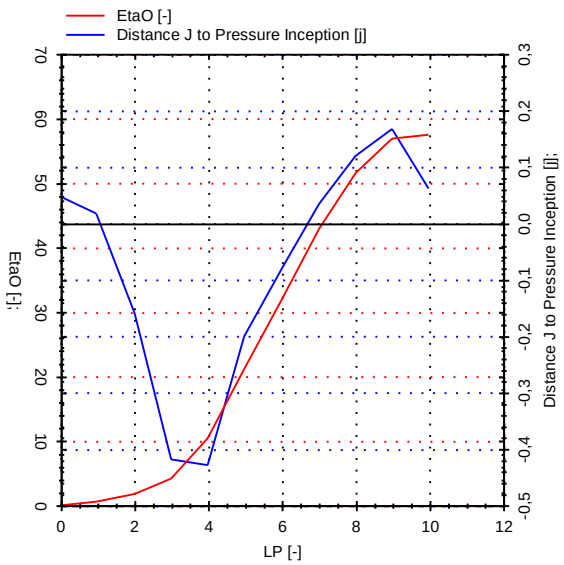
Performance

EtaO [-]	DeltaJ [j]	EtaE [-]	Fuel Cons. [ton/hr]
0,000	-0,05	0,359	0,22
0,006	-0,02	0,352	0,21
0,017	0,16	0,349	0,20
0,042	0,42	0,349	0,20
0,103	0,43	0,353	0,21
0,210	0,20	0,363	0,23
0,320	0,08	0,376	0,26
0,429	-0,03	0,396	0,32
0,514	-0,12	0,411	0,41
0,568	-0,17	0,419	0,57
0,576	-0,06	0,423	0,98









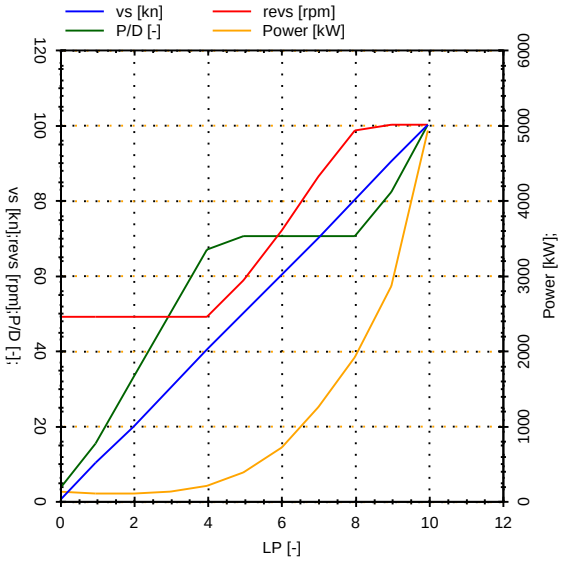
Combinator Result

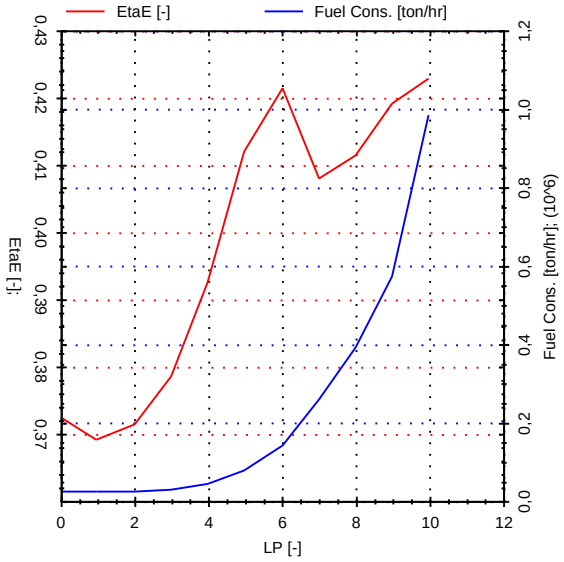
Combinator Settings

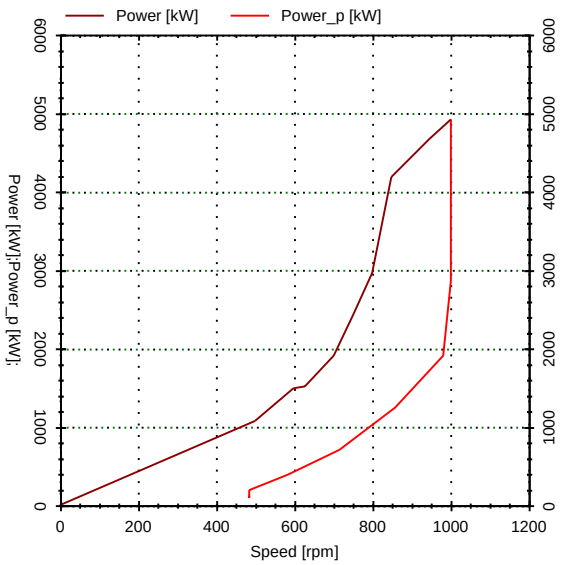
LP [-]	vs [kn]	revs [rpm]	P/D [-]	P_p [kW]
0	0,0	488	0,038	110
1	2,1	488	0,189	97
2	4,3	488	0,413	106
3	6,4	488	0,623	132
4	8,5	488	0,837	196
5	10,7	586	0,884	381
6	12,8	717	0,884	709
7	14,9	859	0,884	1254
8	17,0	983	0,884	1914
9	19,2	1000	1,027	2850
10	21,3	1000	1,255	4920

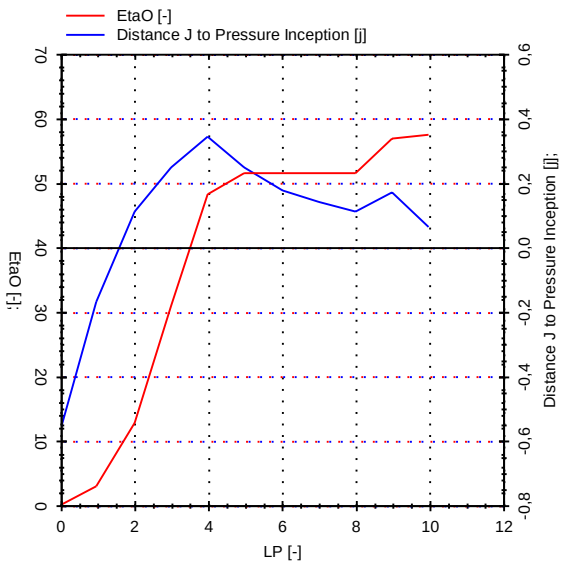
Performance

EtaO [-]	DeltaJ [j]	EtaE [-]	Fuel Cons. [ton/hr]
0,000	0,57	0,373	0,02
0,030	0,17	0,369	0,02
0,129	-0,11	0,371	0,02
0,310	-0,25	0,379	0,03
0,482	-0,34	0,393	0,04
0,515	-0,25	0,412	0,08
0,515	-0,17	0,421	0,14
0,515	-0,14	0,408	0,26
0,514	-0,11	0,411	0,39
0,568	-0,17	0,419	0,57
0,576	-0,06	0,423	0,98









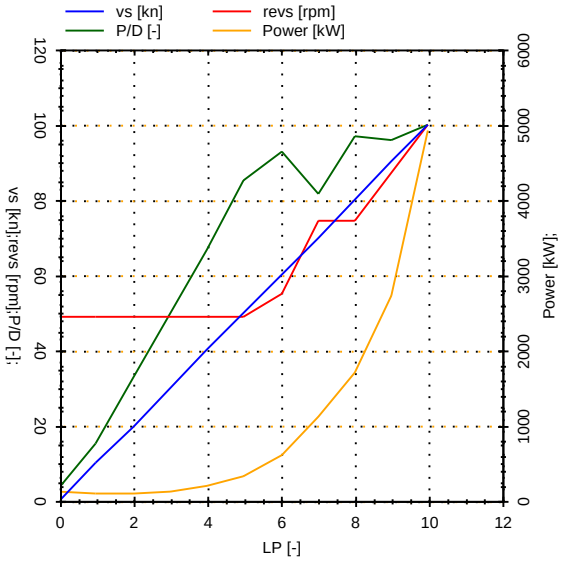
Combinator Result

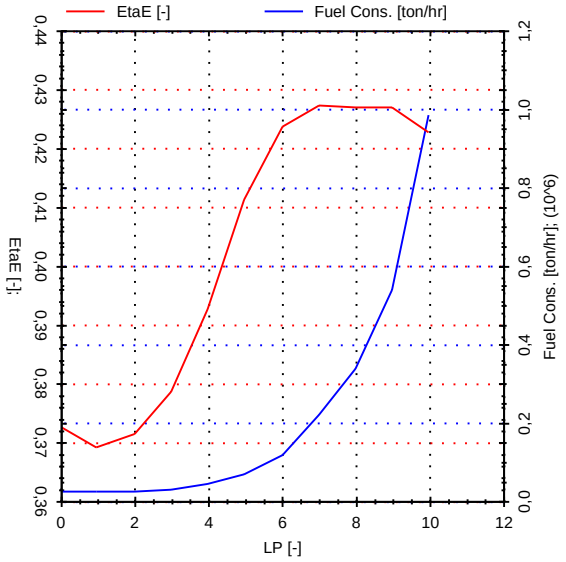
Combinator Settings

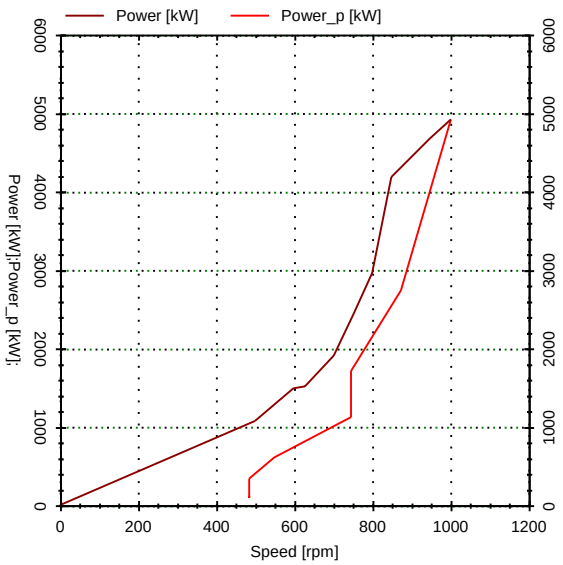
LP [-]	vs [kn]	revs [rpm]	P/D [-]	P_p [kW]
0	0,0	488	0,044	110
1	2,1	488	0,190	97
2	4,3	488	0,412	106
3	6,4	488	0,626	132
4	8,5	488	0,837	196
5	10,7	488	1,071	329
6	12,8	552	1,166	594
7	14,9	744	1,026	1113
8	17,0	744	1,217	1706
9	19,2	872	1,200	2730
10	21,3	1000	1,255	4920

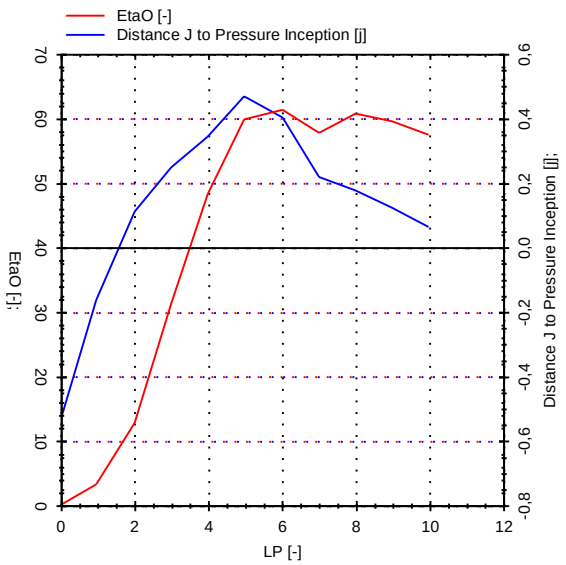
Performance

EtaO [-]	DeltaJ [j]	EtaE [-]	Fuel Cons. [ton/hr]
0,000	0,54	0,373	0,02
0,031	0,17	0,369	0,02
0,127	-0,11	0,371	0,02
0,313	-0,25	0,379	0,03
0,482	-0,34	0,393	0,04
0,598	-0,47	0,411	0,07
0,614	-0,40	0,424	0,12
0,578	-0,22	0,427	0,22
0,606	-0,18	0,427	0,34
0,594	-0,12	0,427	0,54
0,576	-0,06	0,423	0,98









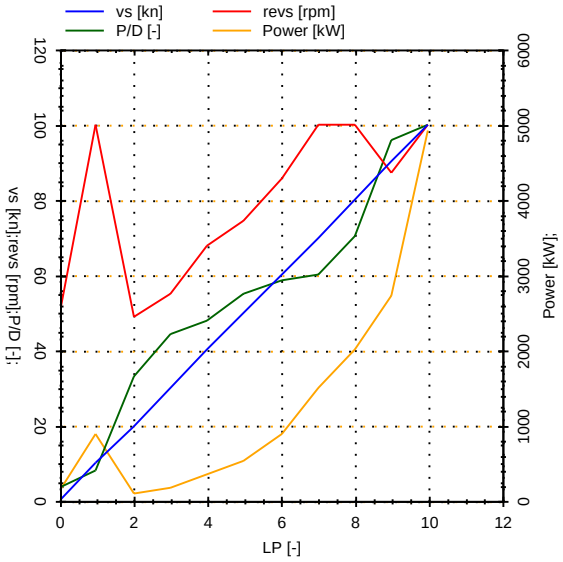
Combinator Result

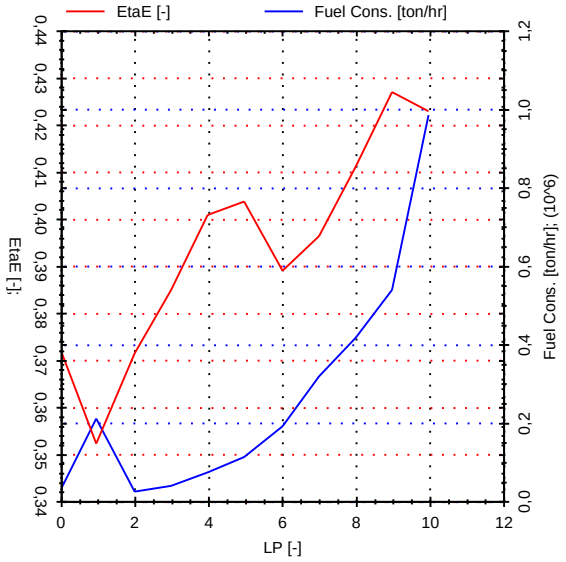
Combinator Settings

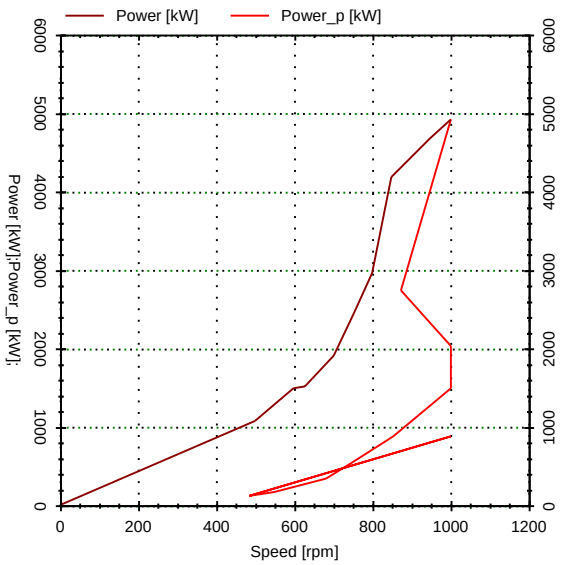
LP [-]	vs [kn]	revs [rpm]	P/D [-]	P_p [kW]
0	0,0	488	0,044	110
1	2,1	1000	0,098	876
2	4,3	488	0,412	106
3	6,4	552	0,558	175
4	8,5	680	0,600	345
5	10,7	744	0,691	527
6	12,8	856	0,736	884
7	14,9	1000	0,755	1497
8	17,0	1000	0,884	2014
9	19,2	872	1,200	2730
10	21,3	1000	1,255	4920

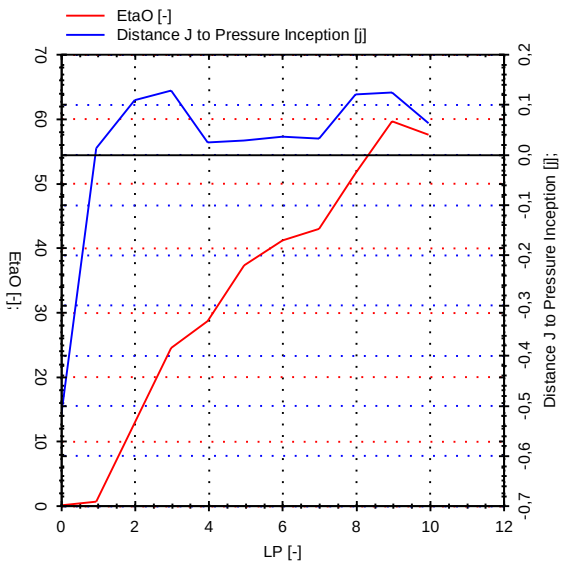
Performance

EtaO [-]	DeltaJ [j]	EtaE [-]	Fuel Cons. [ton/hr]
0,000	0,54	0,373	0,02
0,006	-0,01	0,352	0,21
0,129	-0,11	0,371	0,02
0,244	-0,12	0,385	0,04
0,285	-0,02	0,401	0,07
0,371	-0,03	0,404	0,11
0,411	-0,03	0,389	0,19
0,429	-0,03	0,396	0,32
0,514	-0,12	0,411	0,41
0,594	-0,12	0,427	0,54
0,576	-0,06	0,423	0,98









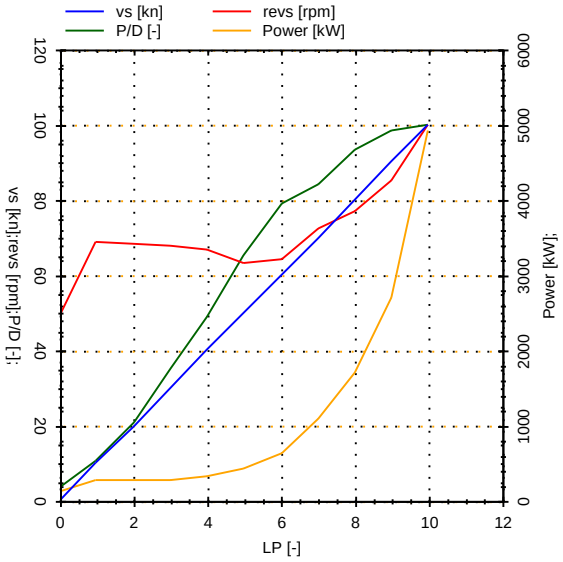
Combinator Result

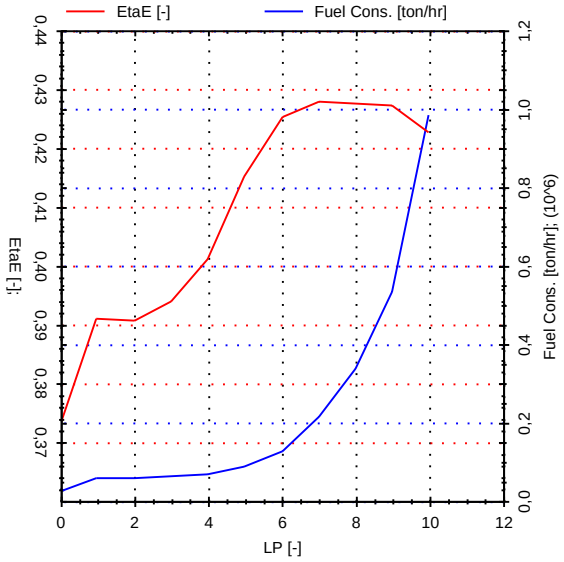
Combinator Settings

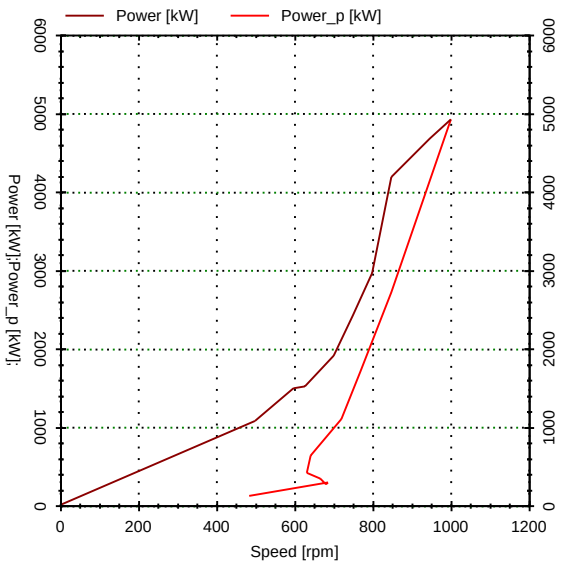
LP [-]	vs [kn]	revs [rpm]	P/D [-]	P_p [kW]
0	0,0	488	0,044	110
1	2,1	687	0,130	275
2	4,3	685	0,260	271
3	6,4	676	0,443	286
4	8,5	666	0,612	329
5	10,7	631	0,820	416
6	12,8	642	0,990	640
7	14,9	724	1,056	1095
8	17,0	769	1,171	1712
9	19,2	850	1,233	2709
10	21,3	1000	1,255	4920

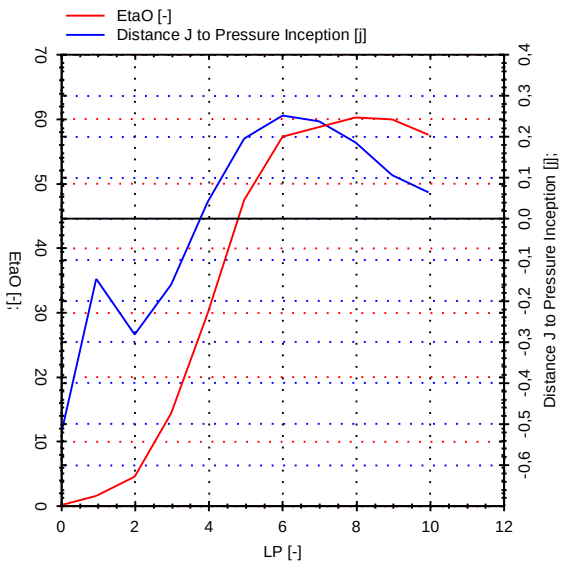
Performance

EtaO [-]	DeltaJ [j]	EtaE [-]	Fuel Cons. [ton/hr]
0,000	0,54	0,373	0,02
0,013	0,15	0,391	0,06
0,045	0,28	0,391	0,06
0,144	0,16	0,394	0,06
0,296	-0,04	0,401	0,07
0,472	-0,19	0,415	0,08
0,570	-0,25	0,425	0,13
0,587	-0,23	0,428	0,22
0,601	-0,19	0,427	0,34
0,598	-0,10	0,427	0,53
0,576	-0,06	0,423	0,98









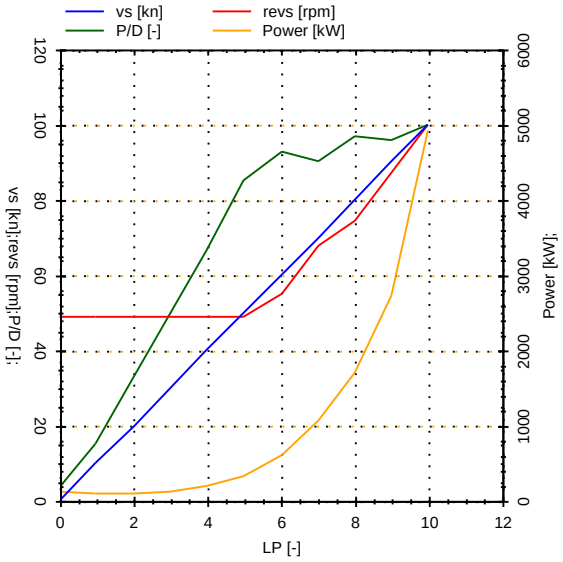
Combinator Result

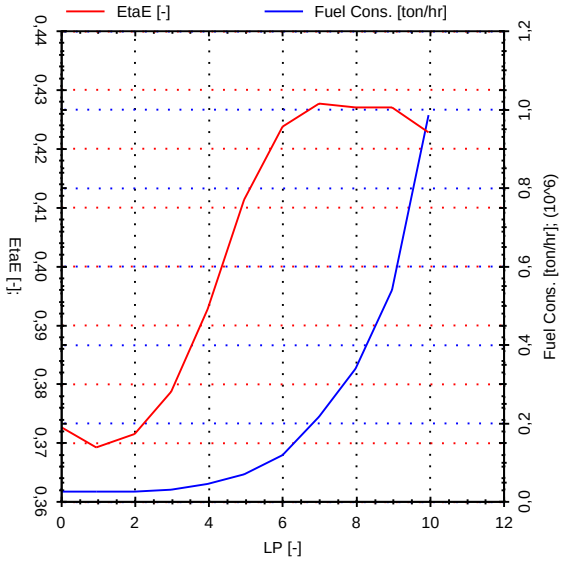
Combinator Settings

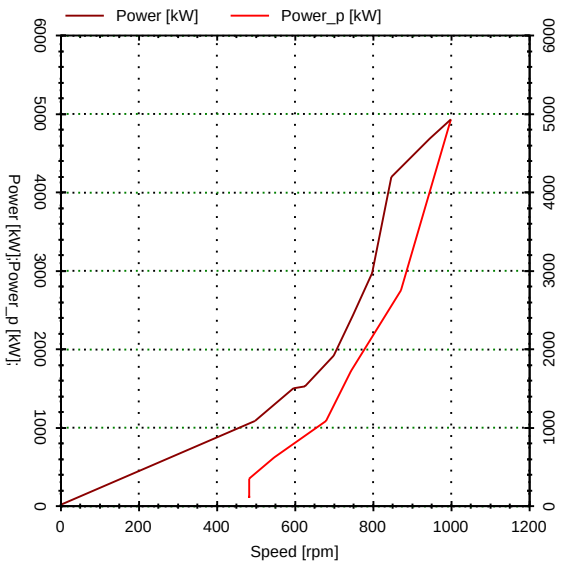
LP [-]	vs [kn]	revs [rpm]	P/D [-]	P_p [kW]
0	0,0	488	0,044	110
1	2,1	488	0,190	97
2	4,3	488	0,412	106
3	6,4	488	0,626	132
4	8,5	488	0,837	196
5	10,7	488	1,071	329
6	12,8	552	1,166	594
7	14,9	680	1,136	1075
8	17,0	744	1,217	1706
9	19,2	872	1,200	2730
10	21,3	1000	1,255	4920

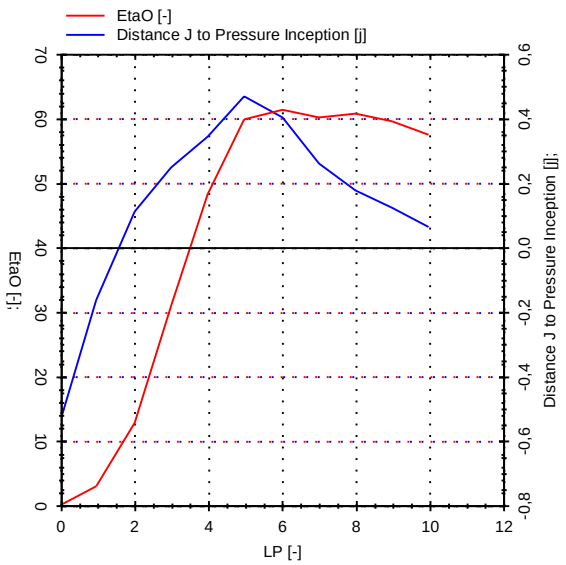
Performance

EtaO [-]	DeltaJ [j]	EtaE [-]	Fuel Cons. [ton/hr]
0,000	0,54	0,373	0,02
0,030	0,17	0,369	0,02
0,129	-0,11	0,371	0,02
0,310	-0,25	0,379	0,03
0,482	-0,34	0,393	0,04
0,598	-0,47	0,411	0,07
0,614	-0,40	0,424	0,12
0,601	-0,26	0,427	0,21
0,606	-0,18	0,427	0,34
0,594	-0,12	0,427	0,54
0,576	-0,06	0,423	0,98









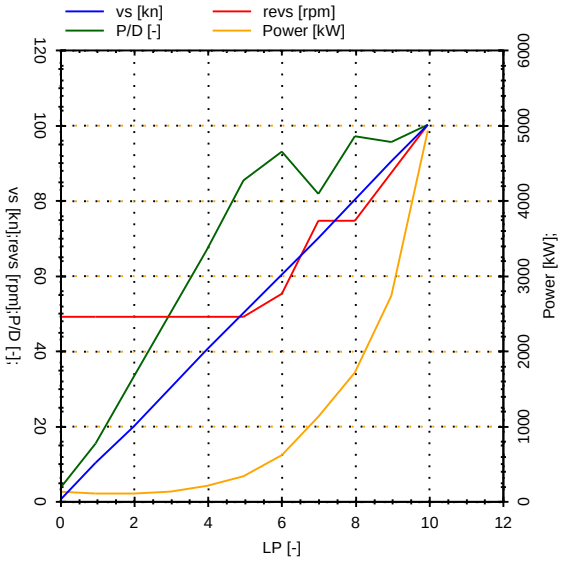
Combinator Result

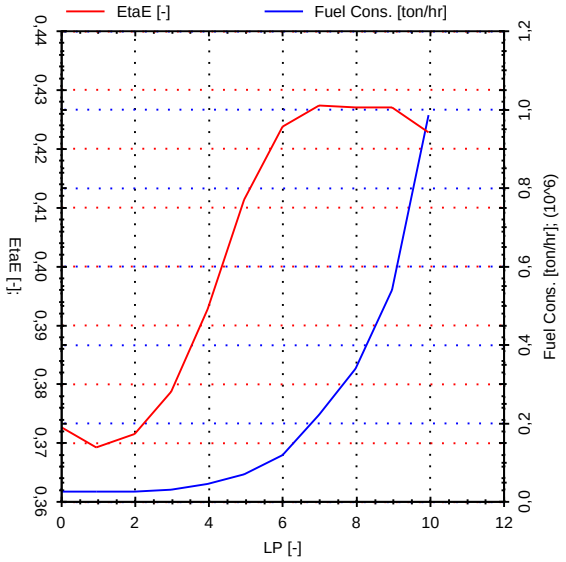
Combinator Settings

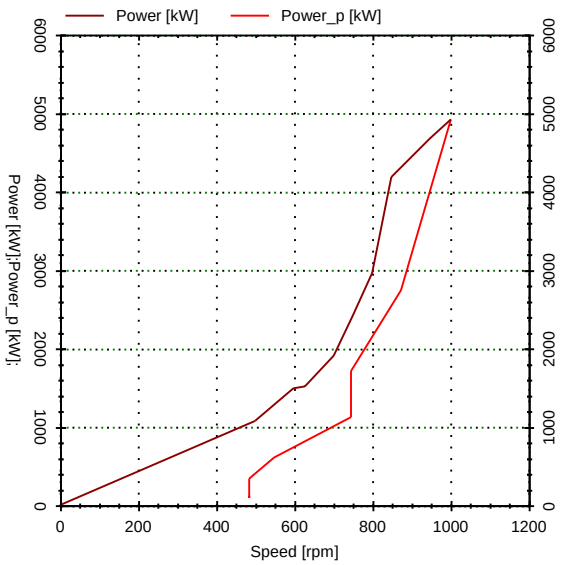
LP [-]	vs [kn]	revs [rpm]	P/D [-]	P_p [kW]
0	0,0	488	0,038	110
1	2,1	488	0,189	97
2	4,3	488	0,413	106
3	6,4	488	0,623	132
4	8,5	488	0,837	196
5	10,7	488	1,070	329
6	12,8	552	1,166	594
7	14,9	744	1,026	1113
8	17,0	744	1,218	1706
9	19,2	872	1,199	2730
10	21,3	1000	1,255	4920

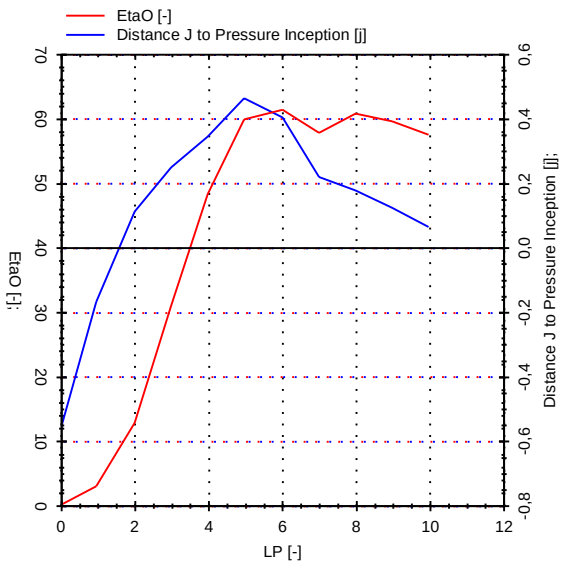
Performance

EtaO [-]	DeltaJ [j]	EtaE [-]	Fuel Cons. [ton/hr]
0,000	0,57	0,373	0,02
0,030	0,17	0,369	0,02
0,129	-0,11	0,371	0,02
0,310	-0,25	0,379	0,03
0,482	-0,34	0,393	0,04
0,598	-0,46	0,411	0,07
0,614	-0,40	0,424	0,12
0,578	-0,22	0,427	0,22
0,606	-0,18	0,427	0,34
0,594	-0,12	0,427	0,54
0,576	-0,06	0,423	0,98









Mega Yacht

Combinator project

Ship Data

Type of ship		MegaYacht
Maximum Vessel Speed	[kn]	20,1

Propeller Data

Diameter	[mm]	2700
Max P/D ratio	[-]	1,377
Propeller Draft	[mm]	2600
AeAo	[-]	0,657

Gearbox Data

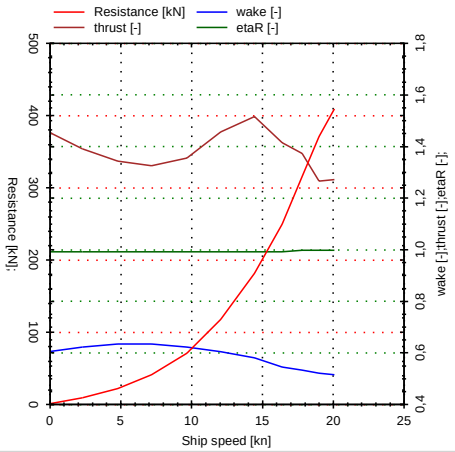
Gearbox ratio	[-]	8,5081
Transmission efficiency	[-]	0,96

Propulsion Configuration Data

Nr. of propellers	[-]	2
Nr. of engines per shaft	[-]	1
Power Take Off	[kW]	0
Power Take In	[kW]	0

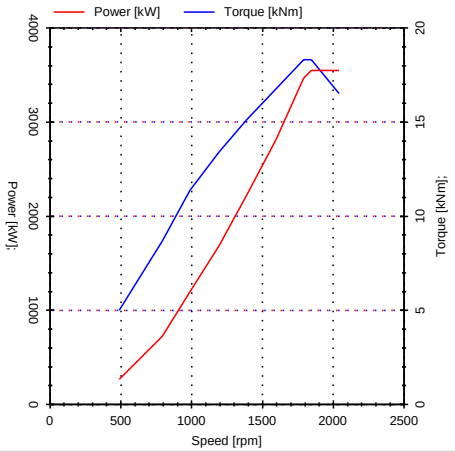
Resistance Data

vs [kn]	R_T [kN]	w [-]	t [-]	Eta_R [-]
0,0	0,0	0,060	0,145	0,990
2,4	8,1	0,062	0,139	0,989
4,8	19,9	0,063	0,134	0,988
7,2	39,3	0,063	0,132	0,987
9,7	70,0	0,062	0,135	0,988
12,1	116,0	0,060	0,145	0,990
14,5	180,5	0,058	0,151	0,991
16,5	247,6	0,054	0,141	0,991
17,9	313,3	0,053	0,137	0,992
19,1	369,6	0,052	0,126	0,992
20,1	407,3	0,051	0,127	0,992



Engine Envelope

Speed [rpm]	Power [kW]	Torque [kNm]
500	260	5
800	720	9
1000	1190	11
1200	1680	13
1400	2210	15
1600	2800	17
1800	3450	18
1850	3540	18
2050	3540	16



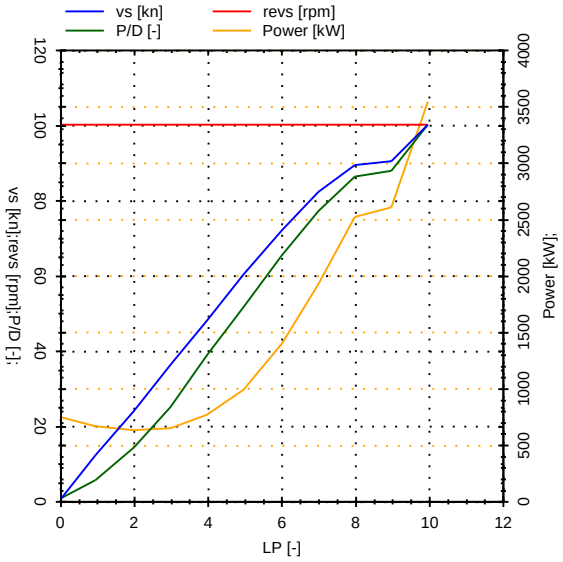
Combinator Result

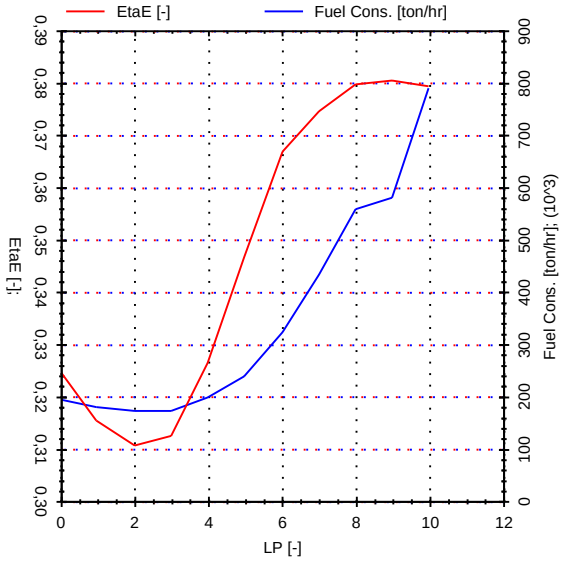
Combinator Settings

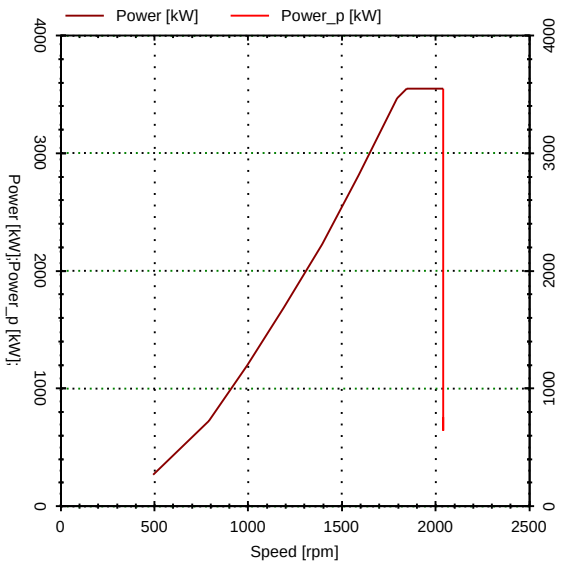
LP [-]	vs [kn]	revs [rpm]	P/D [-]	P_p [kW]
0	0,0	2050	0,004	747
1	2,4	2050	0,075	665
2	4,8	2050	0,193	628
3	7,2	2050	0,346	641
4	9,6	2050	0,532	761
5	12,1	2050	0,711	980
6	14,5	2050	0,899	1398
7	16,5	2050	1,057	1915
8	17,9	2050	1,188	2513
9	18,1	2050	1,206	2607
10	20,1	2050	1,377	3540

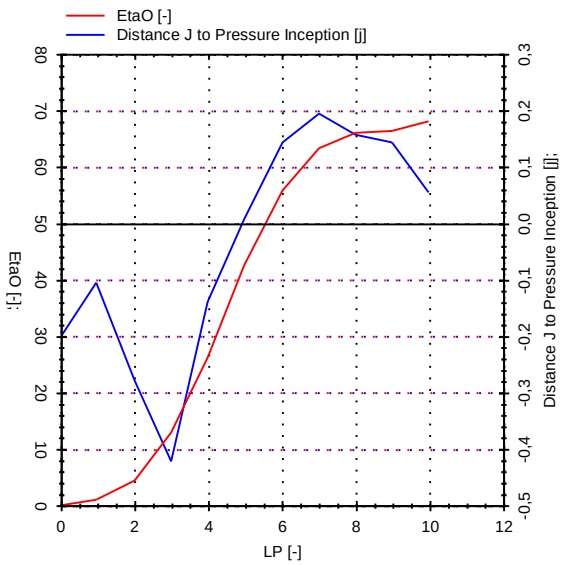
Performance

EtaO [-]	DeltaJ [j]	EtaE [-]	Fuel Cons. [ton/hr]
0,000	0,20	0,325	0,19
0,009	0,11	0,315	0,18
0,045	0,28	0,311	0,17
0,128	0,42	0,312	0,17
0,262	0,14	0,326	0,20
0,423	-0,01	0,347	0,24
0,558	-0,14	0,367	0,32
0,632	-0,19	0,374	0,43
0,660	-0,15	0,380	0,56
0,662	-0,14	0,380	0,58
0,679	-0,05	0,379	0,79









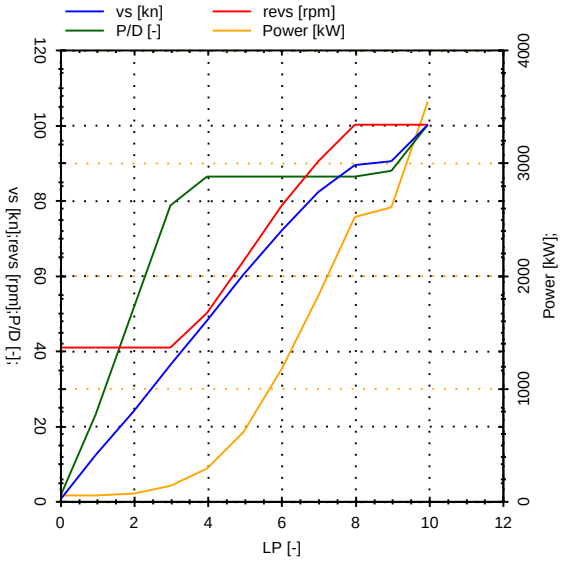
Combinator Result

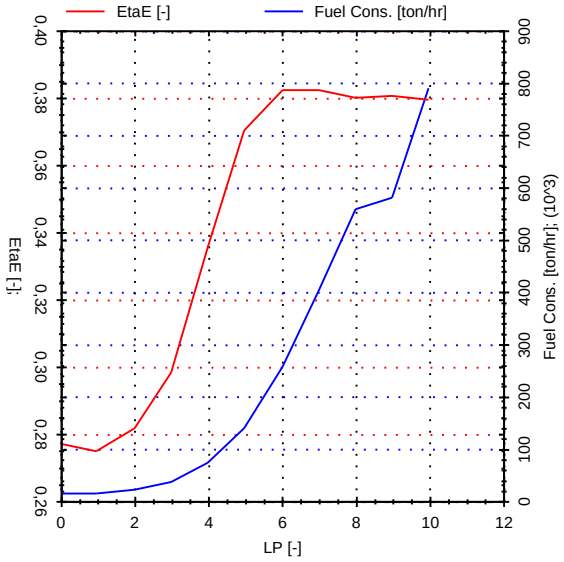
Combinator Settings

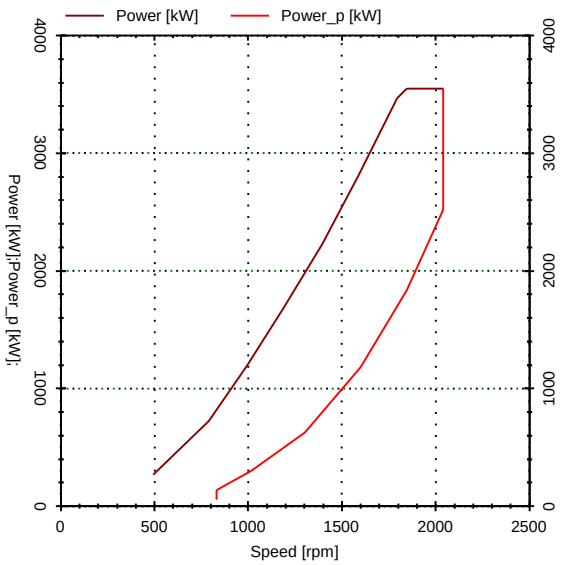
LP [-]	vs [kn]	revs [rpm]	P/D [-]	P_p [kW]
0	0,0	837	0,004	51
1	2,4	837	0,314	44
2	4,8	837	0,706	67
3	7,2	837	1,083	130
4	9,6	1027	1,188	289
5	12,1	1308	1,188	611
6	14,5	1606	1,188	1166
7	16,5	1854	1,188	1819
8	17,9	2050	1,188	2513
9	18,1	2050	1,206	2607
10	20,1	2050	1,377	3540

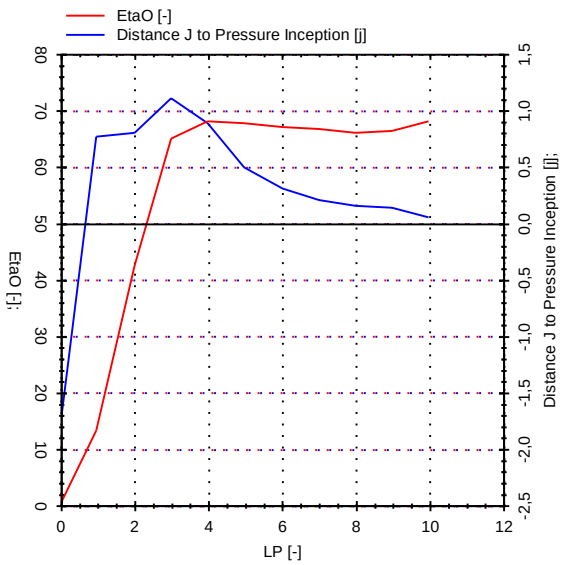
Performance

EtaO [-]	DeltaJ [j]	EtaE [-]	Fuel Cons. [ton/hr]
0,000	1,84	0,277	0,02
0,130	-0,77	0,275	0,01
0,423	-0,79	0,281	0,02
0,649	-1,10	0,298	0,04
0,680	-0,89	0,335	0,07
0,675	-0,50	0,370	0,14
0,669	-0,30	0,382	0,26
0,666	-0,20	0,382	0,40
0,660	-0,15	0,380	0,56
0,662	-0,14	0,380	0,58
0,679	-0,05	0,379	0,79









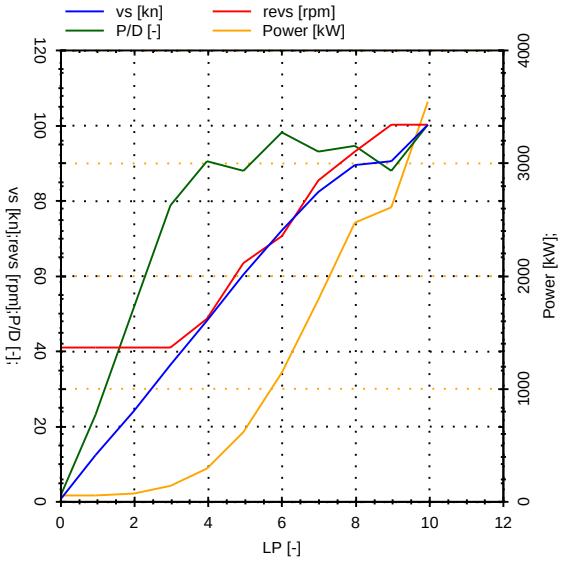
Combinator Result

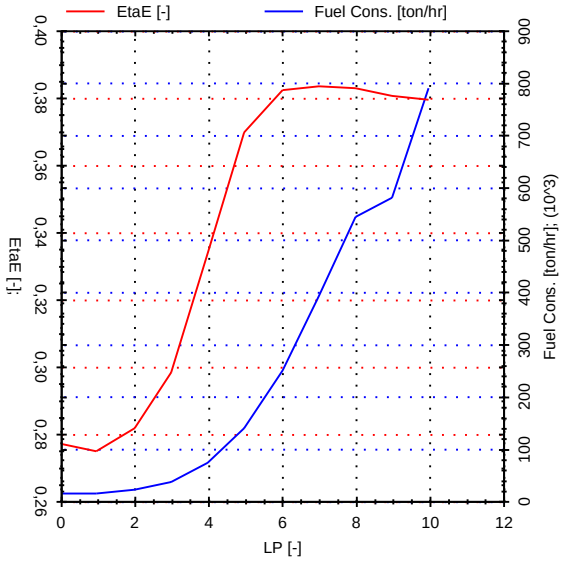
Combinator Settings

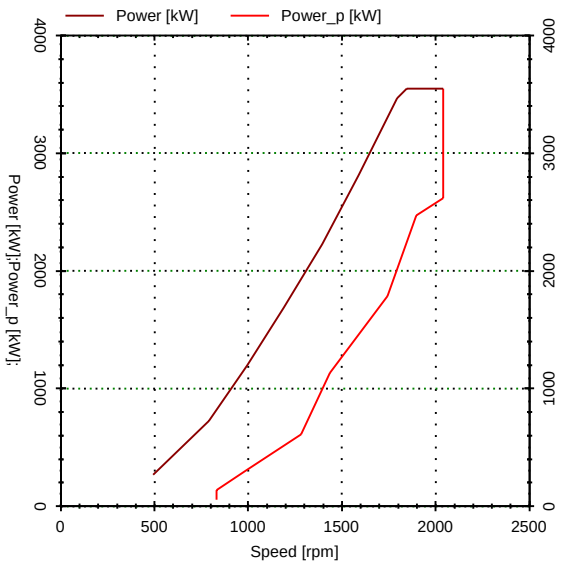
LP [-]	vs [kn]	revs [rpm]	P/D [-]	P_p [kW]
0	0,0	837	0,004	51
1	2,4	837	0,313	44
2	4,8	837	0,707	67
3	7,2	837	1,083	130
4	9,6	989	1,240	284
5	12,1	1292	1,204	608
6	14,5	1444	1,347	1128
7	16,5	1747	1,275	1783
8	17,9	1898	1,302	2457
9	18,1	2050	1,206	2607
10	20,1	2050	1,377	3540

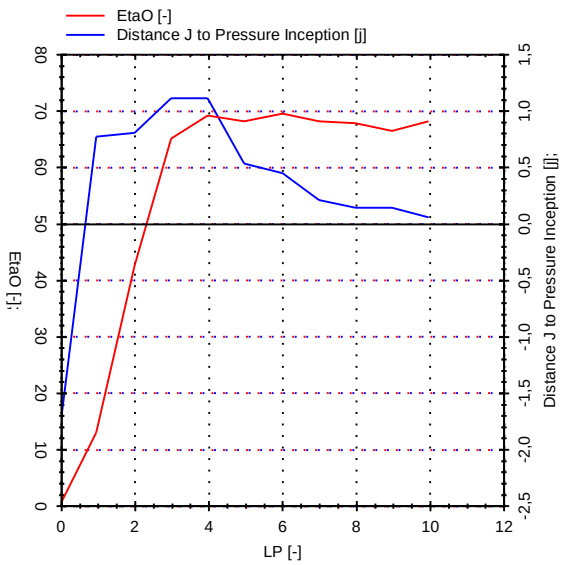
Performance

EtaO [-]	DeltaJ [j]	EtaE [-]	Fuel Cons. [ton/hr]
0,000	1,84	0,277	0,02
0,130	-0,77	0,275	0,01
0,424	-0,79	0,281	0,02
0,649	-1,10	0,298	0,04
0,691	-1,11	0,333	0,07
0,679	-0,52	0,370	0,14
0,692	-0,44	0,382	0,25
0,679	-0,21	0,383	0,39
0,675	-0,14	0,383	0,54
0,662	-0,14	0,380	0,58
0,679	-0,05	0,379	0,79









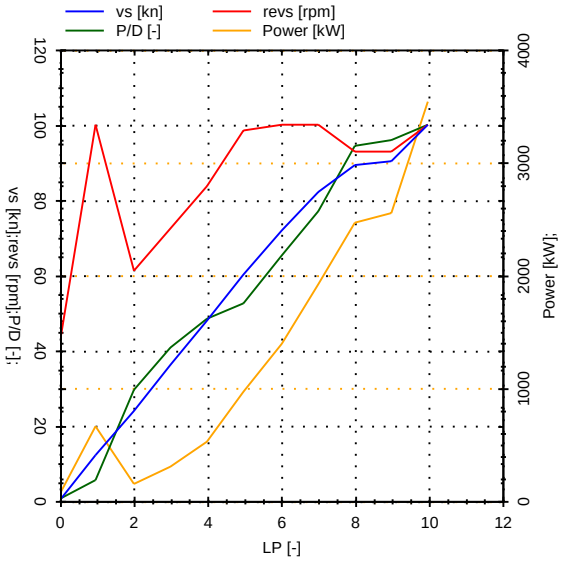
Combinator Result

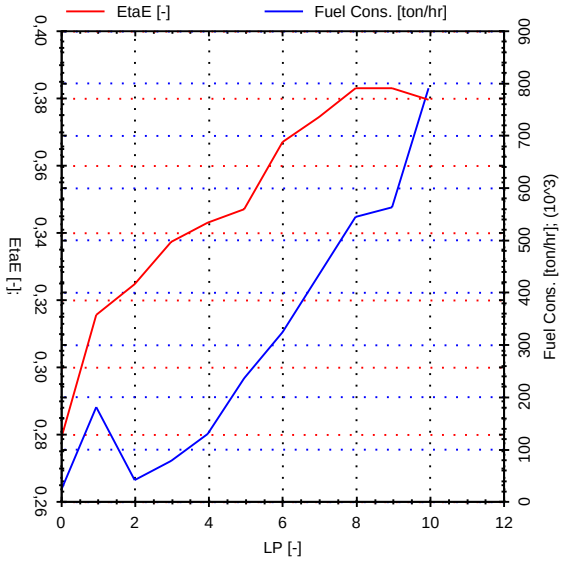
Combinator Settings

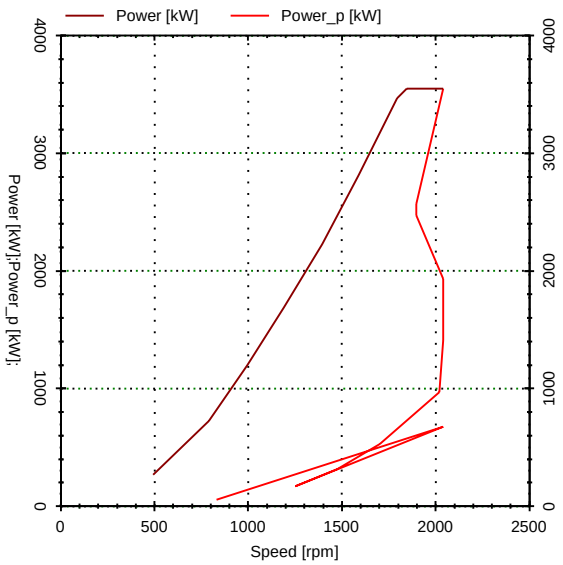
LP [-]	vs [kn]	revs [rpm]	P/D [-]	P_p [kW]
0	0,0	837	0,004	51
1	2,4	2050	0,076	665
2	4,8	1254	0,408	155
3	7,2	1481	0,563	298
4	9,6	1709	0,668	524
5	12,1	2022	0,723	957
6	14,5	2050	0,899	1398
7	16,5	2050	1,057	1915
8	17,9	1898	1,302	2457
9	18,1	1898	1,323	2554
10	20,1	2050	1,377	3540

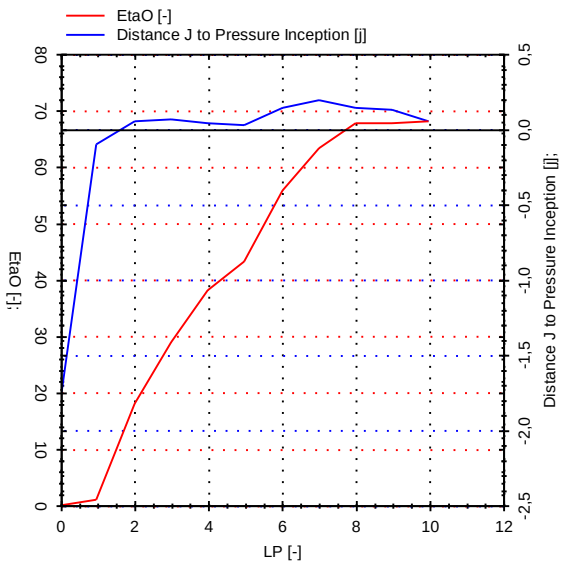
Performance

EtaO [-]	DeltaJ [j]	EtaE [-]	Fuel Co. [ton/hr]
0,000	1,84	0,277	0,02
0,009	0,10	0,315	0,18
0,179	-0,06	0,324	0,04
0,288	-0,06	0,337	0,07
0,380	-0,03	0,342	0,13
0,432	-0,02	0,347	0,23
0,558	-0,14	0,367	0,32
0,632	-0,19	0,374	0,43
0,675	-0,14	0,383	0,54
0,676	-0,13	0,383	0,56
0,679	-0,05	0,379	0,79









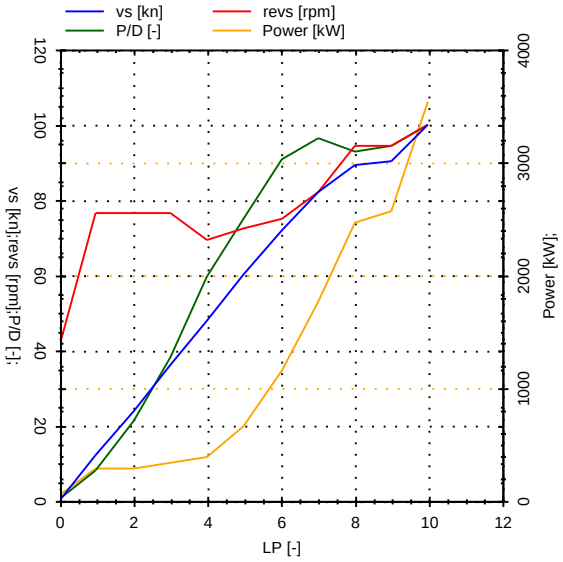
Combinator Result

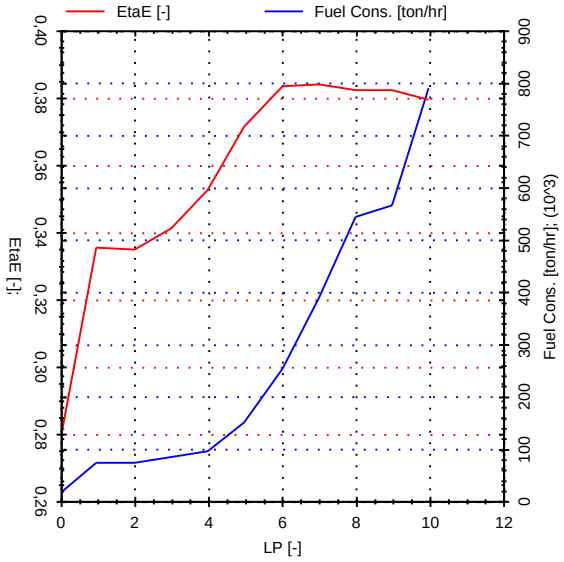
Combinator Settings

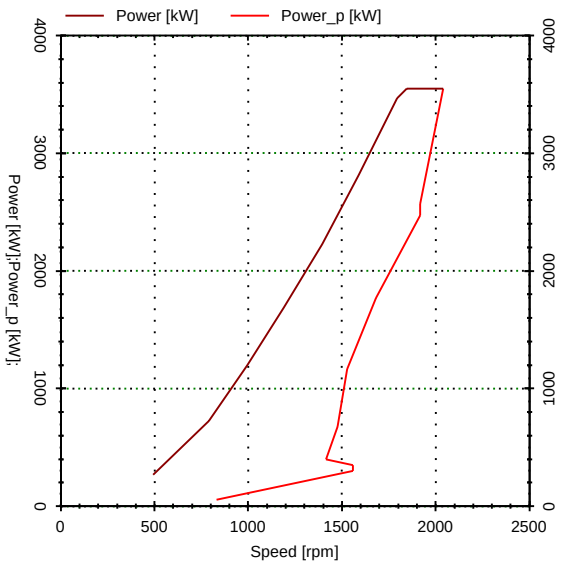
LP [-]	vs [kn]	revs [rpm]	P/D [-]	P_p [kW]
0	0,0	837	0,004	51
1	2,4	1565	0,110	286
2	4,8	1565	0,294	282
3	7,2	1565	0,523	335
4	9,6	1424	0,821	390
5	12,1	1482	1,029	659
6	14,5	1536	1,252	1147
7	16,5	1686	1,330	1767
8	17,9	1929	1,277	2466
9	18,1	1929	1,298	2563
10	20,1	2050	1,377	3540

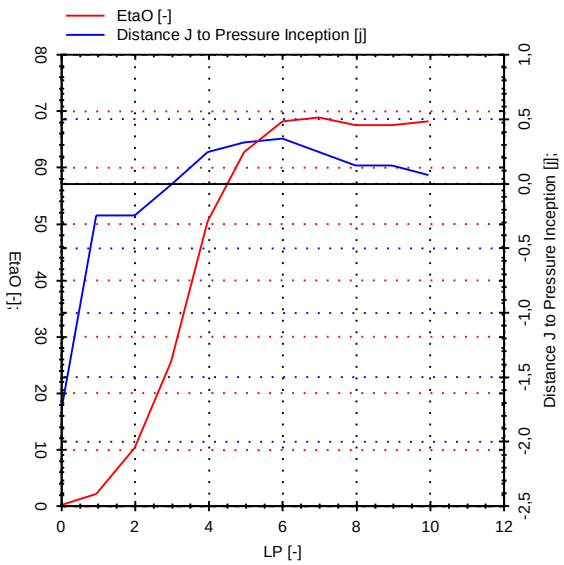
Performance

EtaO [-]	DeltaJ [j]	EtaE [-]	Fuel Cons. [ton/hr]
0,000	1,84	0,277	0,02
0,020	0,25	0,335	0,07
0,100	0,26	0,335	0,07
0,254	0,02	0,341	0,08
0,503	-0,24	0,353	0,09
0,626	-0,32	0,371	0,15
0,680	-0,34	0,383	0,25
0,686	-0,24	0,384	0,39
0,672	-0,13	0,382	0,54
0,674	-0,13	0,382	0,57
0,679	-0,05	0,379	0,79









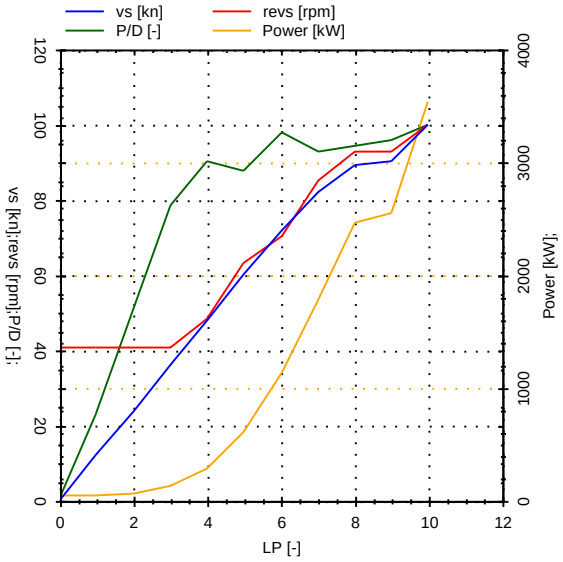
Combinator Result

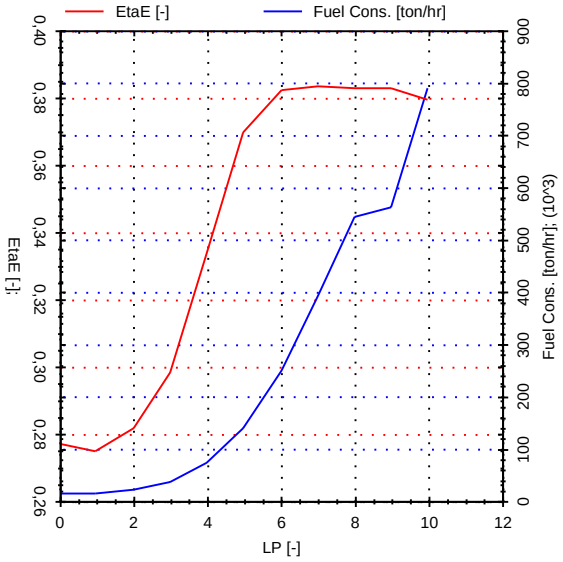
Combinator Settings

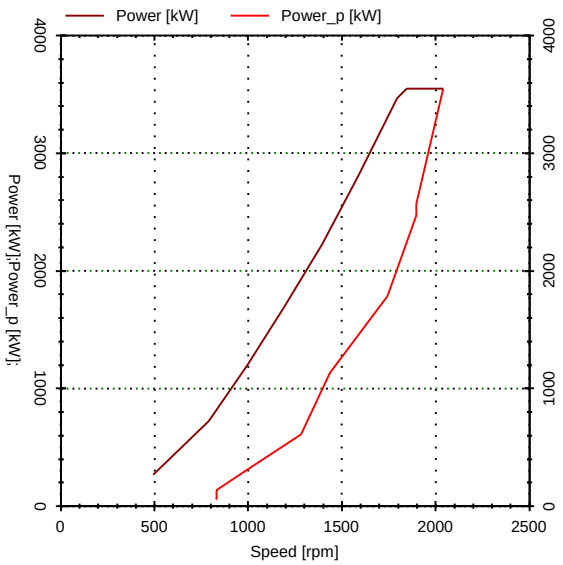
LP [-]	vs [kn]	revs [rpm]	P/D [-]	P_p [kW]
0	0,0	837	0,004	51
1	2,4	837	0,313	44
2	4,8	837	0,707	67
3	7,2	837	1,083	130
4	9,6	989	1,240	284
5	12,1	1292	1,204	608
6	14,5	1444	1,347	1128
7	16,5	1747	1,275	1783
8	17,9	1898	1,302	2457
9	18,1	1898	1,323	2554
10	20,1	2050	1,377	3540

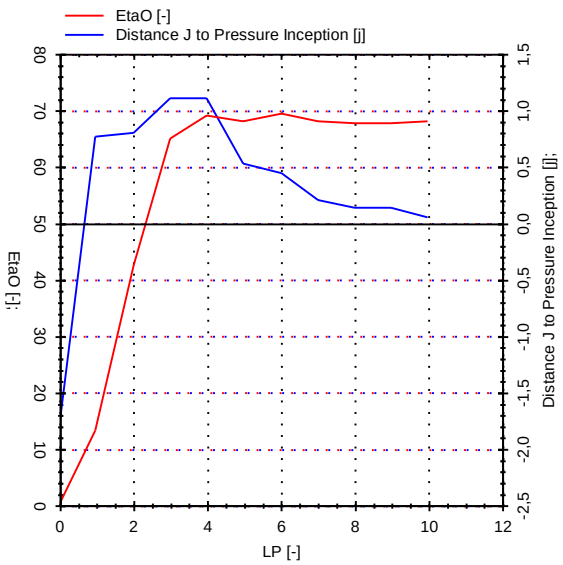
Performance

EtaO [-]	DeltaJ [j]	EtaE [-]	Fuel Cons. [ton/hr]
0,000	1,84	0,277	0,02
0,130	-0,77	0,275	0,01
0,423	-0,79	0,281	0,02
0,649	-1,10	0,298	0,04
0,691	-1,11	0,333	0,07
0,679	-0,52	0,370	0,14
0,692	-0,44	0,382	0,25
0,679	-0,21	0,383	0,39
0,675	-0,14	0,383	0,54
0,676	-0,13	0,383	0,56
0,679	-0,05	0,379	0,79









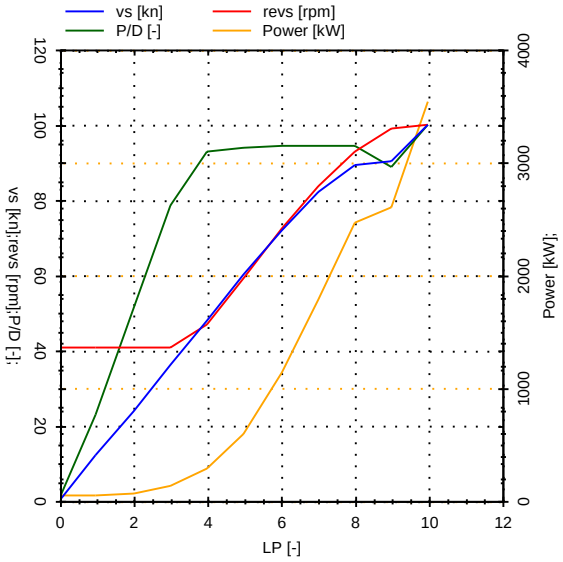
Combinator Result

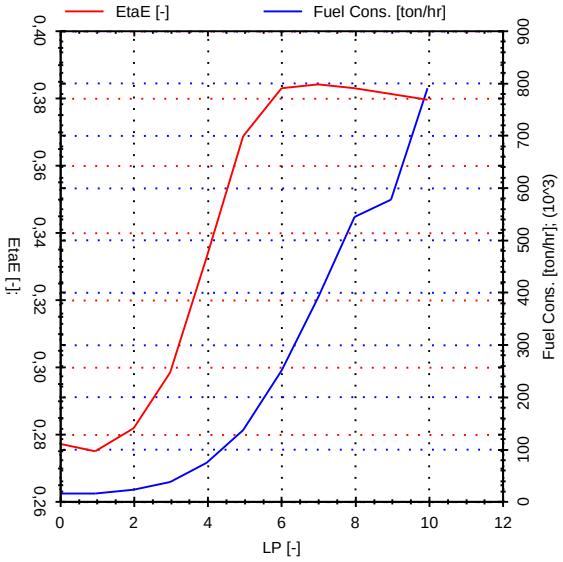
Combinator Settings

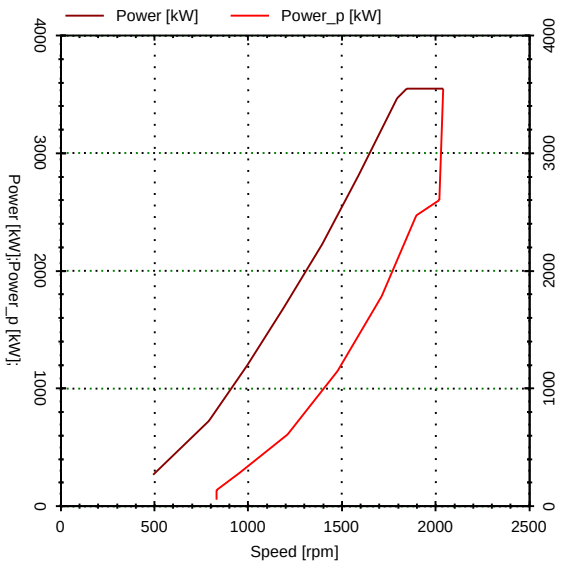
LP [-]	vs [kn]	revs [rpm]	P/D [-]	P_p [kW]
0	0,0	837	0,004	51
1	2,4	837	0,314	44
2	4,8	837	0,706	67
3	7,2	837	1,083	130
4	9,6	963	1,279	281
5	12,1	1214	1,295	594
6	14,5	1486	1,301	1136
7	16,5	1716	1,302	1774
8	17,9	1904	1,297	2459
9	18,1	2030	1,221	2599
10	20,1	2050	1,377	3540

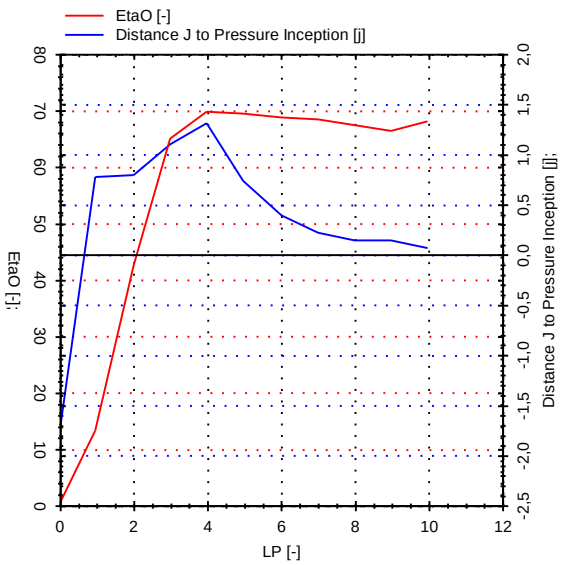
Performance

EtaO [-]	DeltaJ [j]	EtaE [-]	Fuel Cons. [ton/hr]
0,000	1,84	0,277	0,02
0,130	-0,77	0,275	0,01
0,423	-0,79	0,281	0,02
0,649	-1,10	0,298	0,04
0,698	-1,30	0,332	0,07
0,695	-0,73	0,369	0,14
0,687	-0,39	0,383	0,25
0,683	-0,22	0,384	0,39
0,674	-0,14	0,383	0,54
0,664	-0,14	0,381	0,58
0,679	-0,05	0,379	0,79









Trawler

Combinator project

Ship Data

Type of ship		Trawler_ducted
Maximum Vessel Speed	[kn]	12,5

Propeller Data

Diameter	[mm]	2300
Max P/D ratio	[-]	1,067
Propeller Draft	[mm]	2000
AeAo	[-]	0,68

Gearbox Data

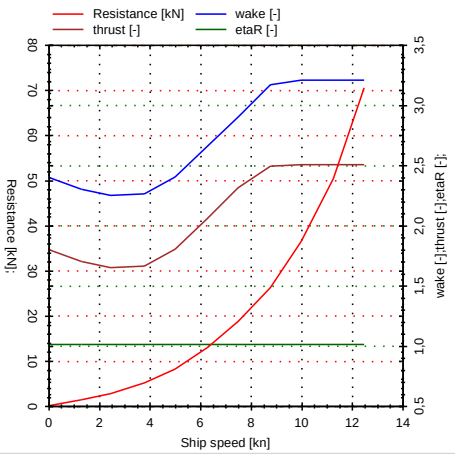
Gearbox ratio	[-]	6,19266
Transmission efficiency	[-]	0,97

Propulsion Configuration Data

Nr. of propellers	[-]	1
Nr. of engines per shaft	[-]	1
Power Take Off	[kW]	0
Power Take In	[kW]	0

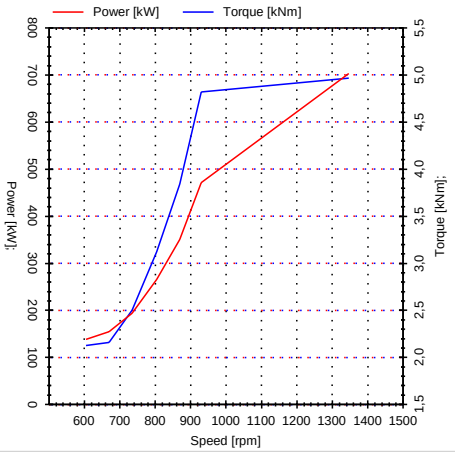
Resistance Data

vs [kn]	R_T [kN]	w [-]	t [-]	Eta_R [-]
0,0	0,0	0,240	0,180	1,010
1,3	1,2	0,230	0,170	1,010
2,5	2,7	0,224	0,164	1,010
3,8	5,0	0,226	0,166	1,010
5,0	8,2	0,240	0,180	1,010
6,3	12,7	0,265	0,205	1,010
7,5	18,6	0,290	0,231	1,010
8,8	26,1	0,317	0,249	1,010
10,0	36,3	0,320	0,250	1,010
11,3	50,4	0,320	0,250	1,010
12,5	70,5	0,320	0,250	1,010



Engine Envelope

Speed [rpm]	Power [kW]	Torque [kNm]
608	135	2
673	151	2
738	193	2
809	263	3
872	350	4
932	470	5
1350	700	5



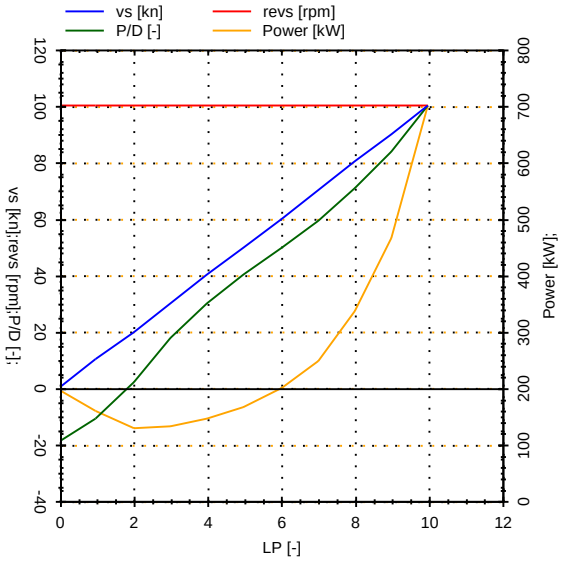
Combinator Result

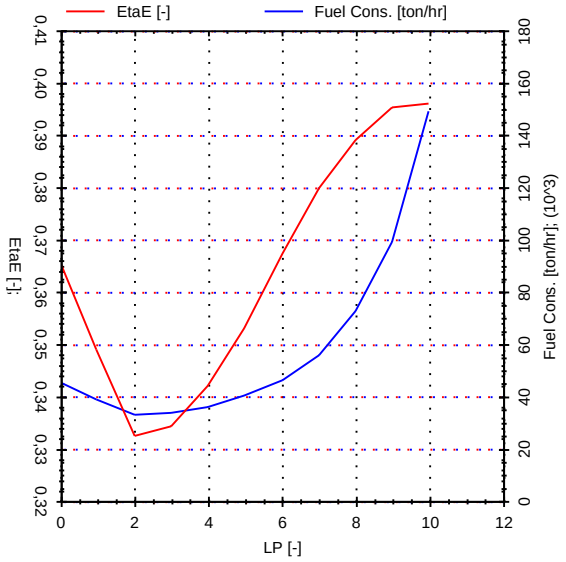
Combinator Settings

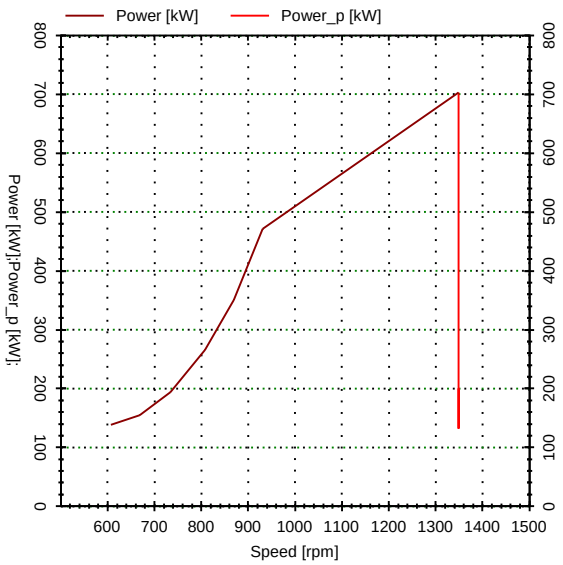
LP [-]	vs [kn]	revs [rpm]	P/D [-]	P_p [kW]
0	0,0	1350	-0,205	198
1	1,3	1350	-0,119	159
2	2,5	1350	0,022	130
3	3,8	1350	0,187	132
4	5,0	1350	0,323	146
5	6,3	1350	0,429	167
6	7,5	1350	0,533	201
7	8,8	1350	0,632	249
8	10,0	1350	0,758	336
9	11,3	1350	0,894	466
10	12,5	1350	1,067	700

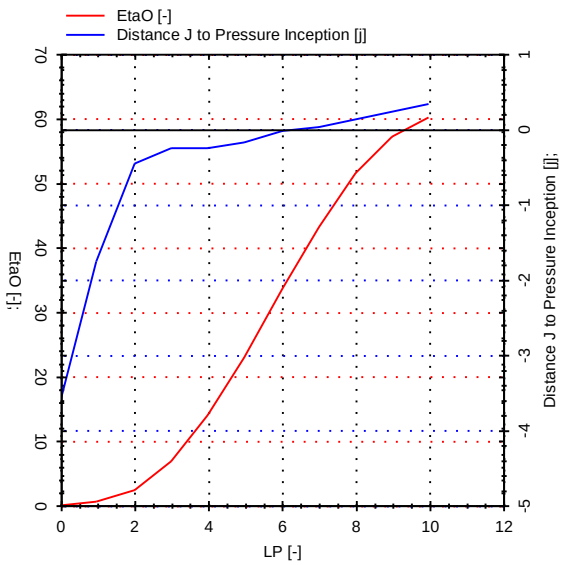
Performance

EtaO [-]	DeltaJ [j]	EtaE [-]	Fuel Cons. [ton/hr]
0,000	3,66	0,366	0,05
0,005	1,76	0,349	0,04
0,024	0,47	0,333	0,03
0,066	0,26	0,334	0,03
0,141	0,27	0,342	0,04
0,229	0,17	0,353	0,04
0,337	0,03	0,367	0,05
0,430	-0,03	0,380	0,06
0,515	-0,13	0,389	0,07
0,571	-0,22	0,395	0,10
0,600	-0,32	0,396	0,15









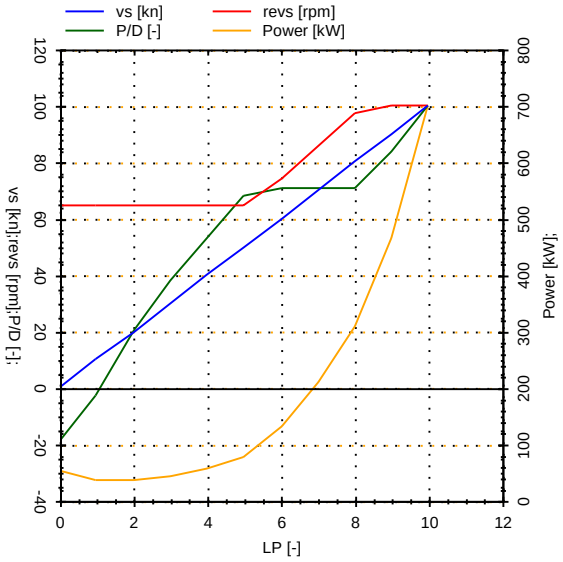
Combinator Result

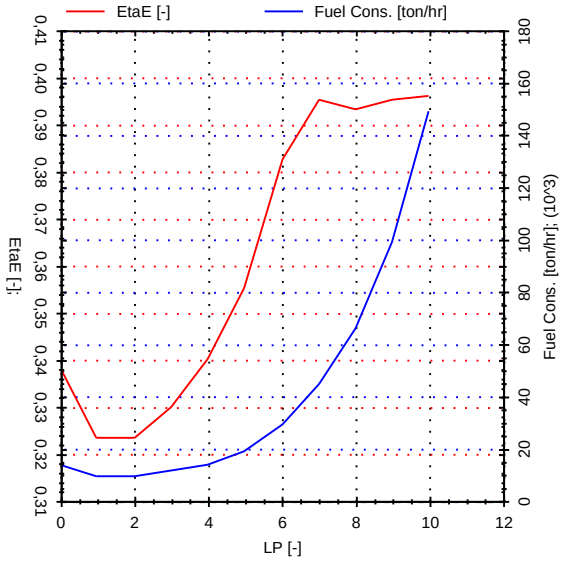
Combinator Settings

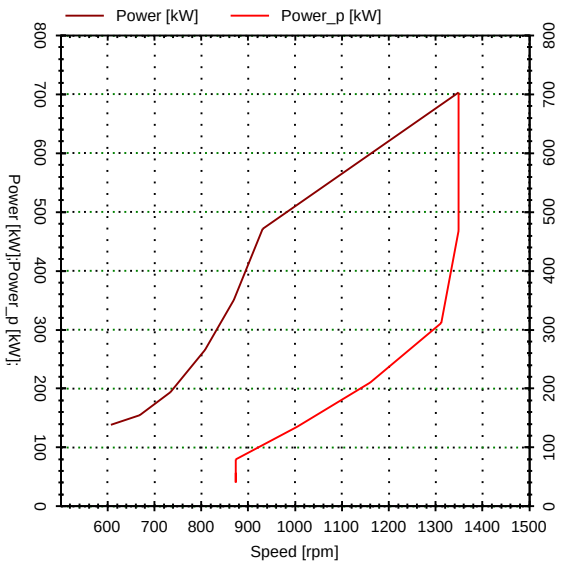
LP [-]	vs [kn]	revs [rpm]	P/D [-]	P_p [kW]
0	0,0	877	-0,205	54
1	1,3	877	-0,031	37
2	2,5	877	0,220	37
3	3,8	877	0,406	44
4	5,0	877	0,566	57
5	6,3	877	0,723	79
6	7,5	1006	0,758	133
7	8,8	1161	0,758	210
8	10,0	1313	0,758	309
9	11,3	1350	0,894	466
10	12,5	1350	1,067	700

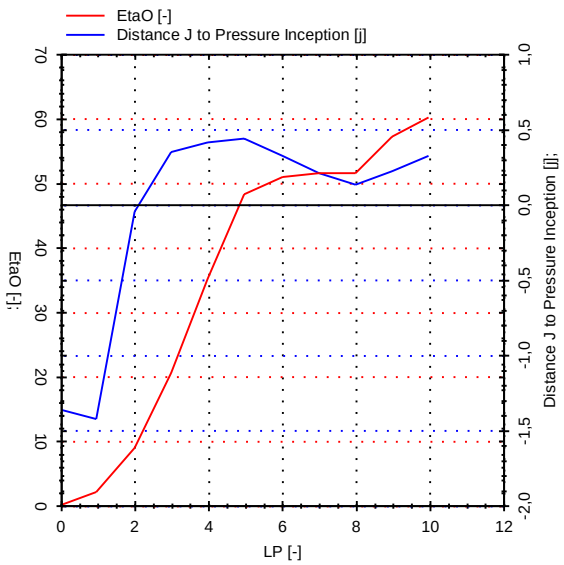
Performance

EtaO [-]	DeltaJ [j]	EtaE [-]	Fuel Cons. [ton/hr]
0,000	1,37	0,338	0,01
0,019	1,42	0,323	0,01
0,088	0,06	0,323	0,01
0,203	-0,34	0,330	0,01
0,352	-0,40	0,340	0,01
0,481	-0,44	0,355	0,02
0,510	-0,32	0,382	0,03
0,514	-0,20	0,395	0,04
0,516	-0,13	0,393	0,07
0,571	-0,22	0,395	0,10
0,600	-0,32	0,396	0,15









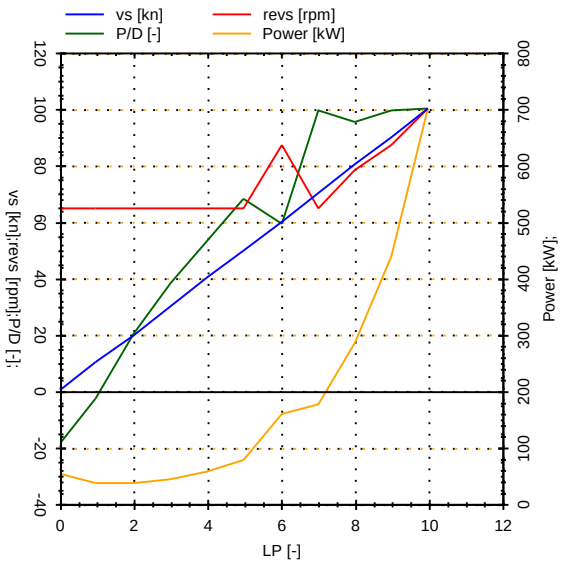
Combinator Result

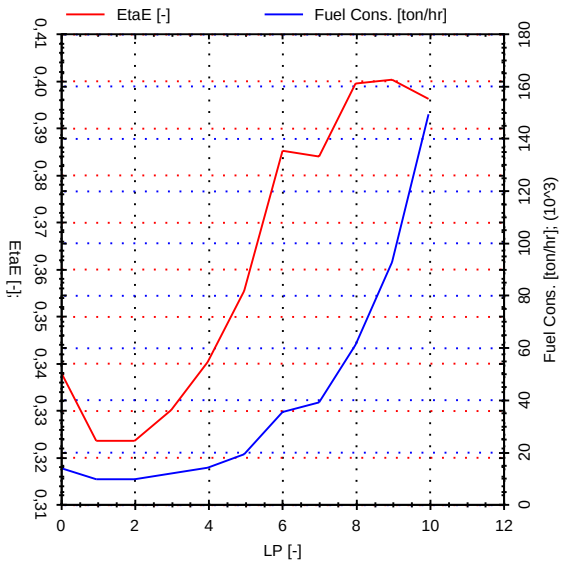
Combinator Settings

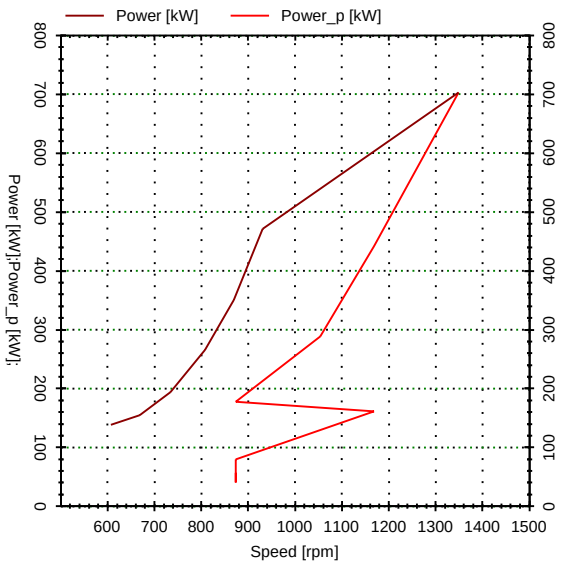
LP [-]	vs [kn]	revs [rpm]	P/D [-]	P_p [kW]
0	0,0	877	-0,206	54
1	1,3	877	-0,027	37
2	2,5	877	0,219	37
3	3,8	877	0,406	44
4	5,0	877	0,566	57
5	6,3	877	0,723	79
6	7,5	1173	0,630	159
7	8,8	877	1,062	176
8	10,0	1054	1,020	287
9	11,3	1173	1,062	439
10	12,5	1350	1,067	700

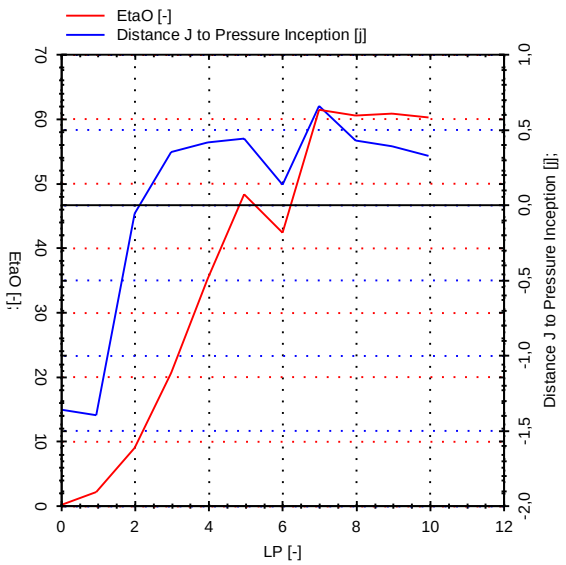
Performance

EtaO [-]	DeltaJ [j]	EtaE [-]	Fuel Cons. [ton/hr]
0,000	1,36	0,338	0,01
0,020	1,41	0,323	0,01
0,088	0,06	0,323	0,01
0,204	-0,34	0,330	0,01
0,351	-0,40	0,340	0,01
0,481	-0,44	0,355	0,02
0,422	-0,12	0,385	0,03
0,612	-0,65	0,384	0,04
0,603	-0,42	0,399	0,06
0,606	-0,38	0,400	0,09
0,600	-0,32	0,396	0,15









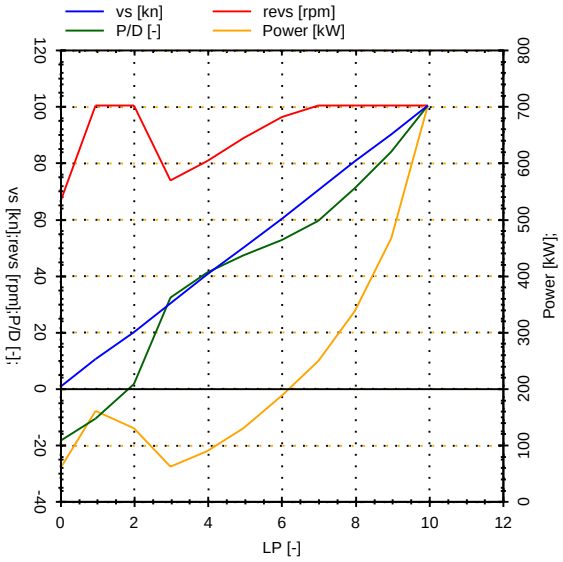
Combinator Result

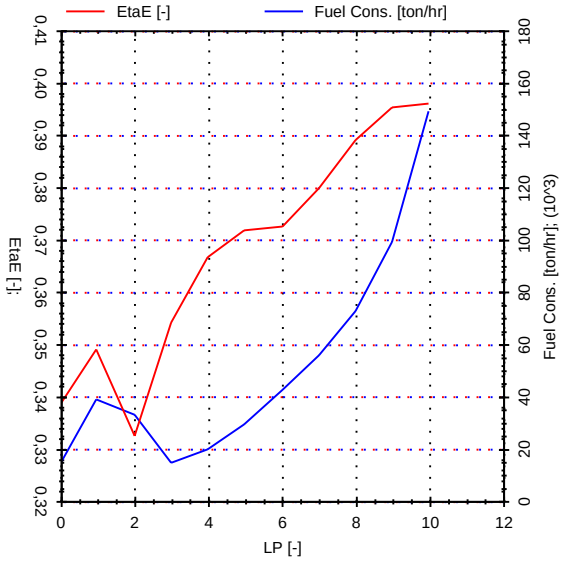
Combinator Settings

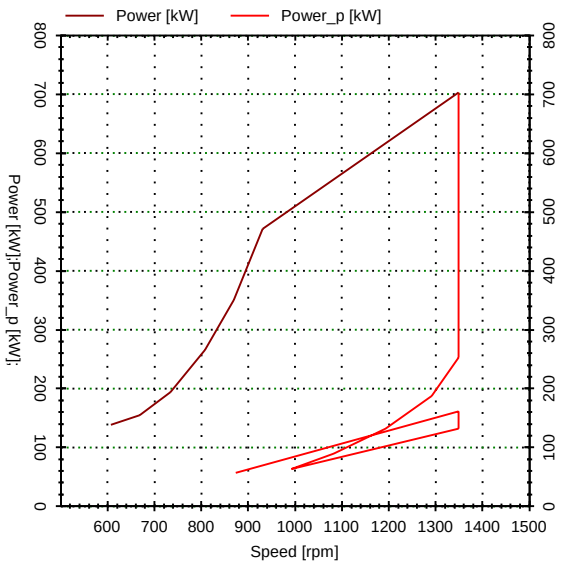
LP [-]	vs [kn]	revs [rpm]	P/D [-]	P_p [kW]
0	0,0	877	-0,206	54
1	1,3	1350	-0,116	159
2	2,5	1350	0,017	130
3	3,8	995	0,343	60
4	5,0	1084	0,439	87
5	6,3	1195	0,502	129
6	7,5	1295	0,560	187
7	8,8	1350	0,632	249
8	10,0	1350	0,758	336
9	11,3	1350	0,894	466
10	12,5	1350	1,067	700

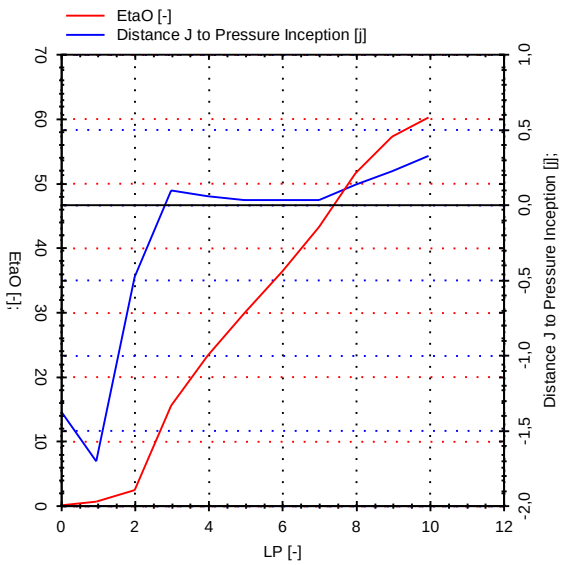
Performance

EtaO [-]	DeltaJ [j]	EtaE [-]	Fuel Cons. [ton/hr]
0,000	1,36	0,338	0,01
0,005	1,72	0,349	0,04
0,024	0,49	0,333	0,03
0,154	-0,08	0,354	0,01
0,232	-0,05	0,366	0,02
0,296	-0,02	0,372	0,03
0,362	-0,02	0,372	0,04
0,430	-0,03	0,380	0,06
0,515	-0,13	0,389	0,07
0,571	-0,22	0,395	0,10
0,600	-0,32	0,396	0,15









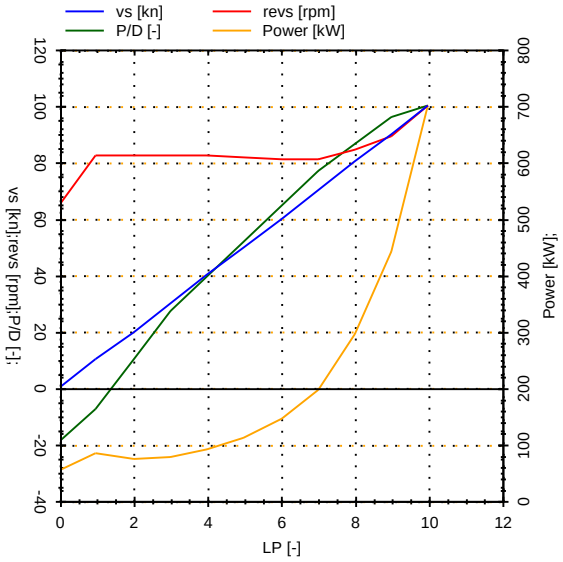
Combinator Result

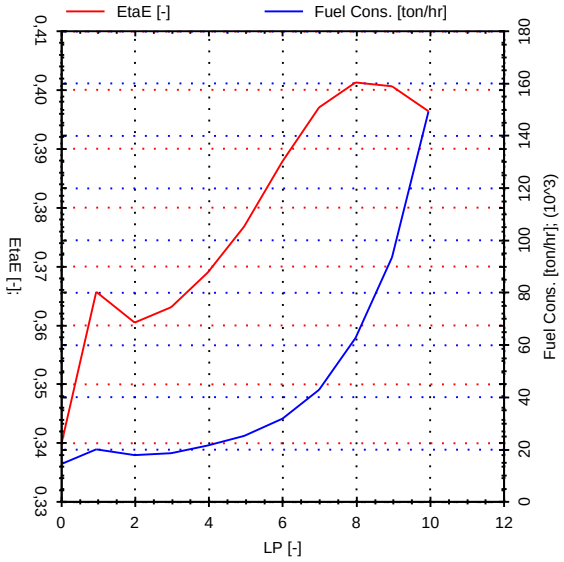
Combinator Settings

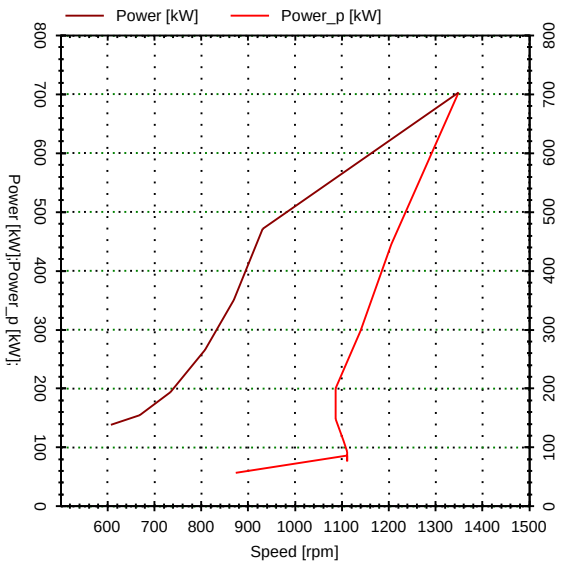
LP [-]	vs [kn]	revs [rpm]	P/D [-]	P_p [kW]
0	0,0	877	-0,206	54
1	1,3	1114	-0,081	85
2	2,5	1114	0,107	73
3	3,8	1114	0,288	79
4	5,0	1114	0,423	92
5	6,3	1104	0,550	111
6	7,5	1091	0,688	145
7	8,8	1089	0,818	198
8	10,0	1142	0,926	297
9	11,3	1208	1,024	443
10	12,5	1350	1,067	700

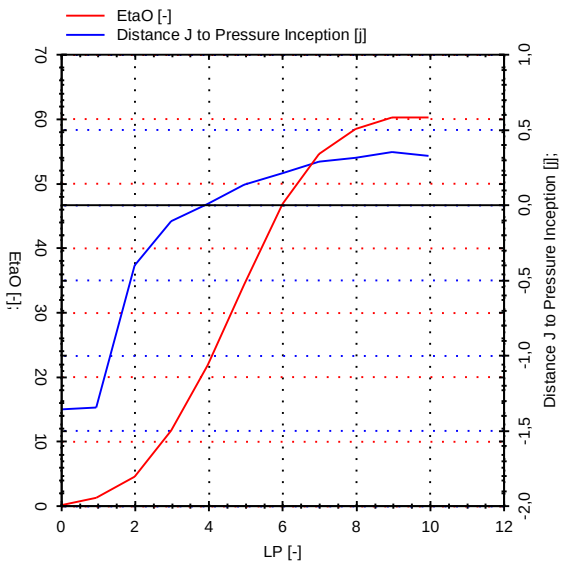
Performance

EtaO [-]	DeltaJ [j]	EtaE [-]	Fuel Cons. [ton/hr]
0,000	1,36	0,338	0,01
0,010	1,36	0,365	0,02
0,045	0,40	0,360	0,02
0,117	0,12	0,363	0,02
0,218	-0,01	0,369	0,02
0,342	-0,13	0,377	0,02
0,466	-0,20	0,388	0,03
0,544	-0,28	0,397	0,04
0,582	-0,31	0,401	0,06
0,601	-0,34	0,400	0,09
0,600	-0,32	0,396	0,15









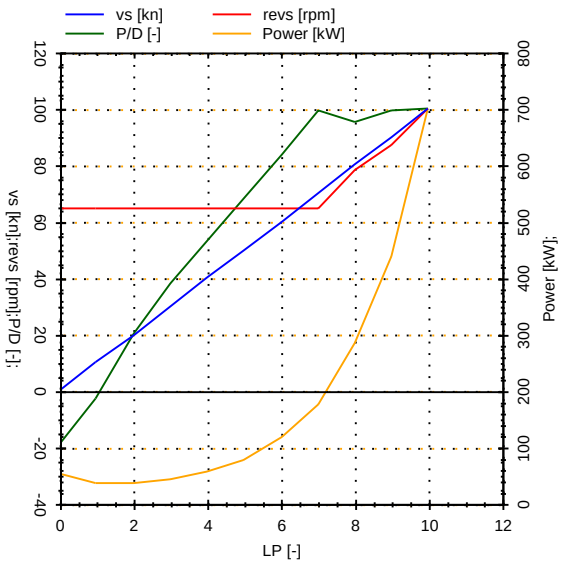
Combinator Result

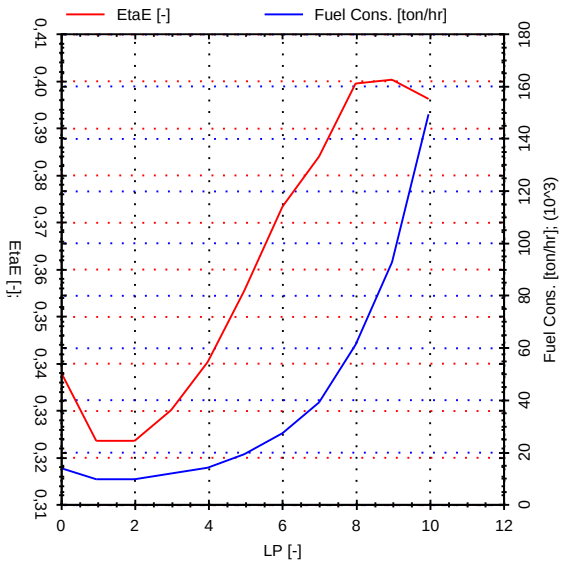
Combinator Settings

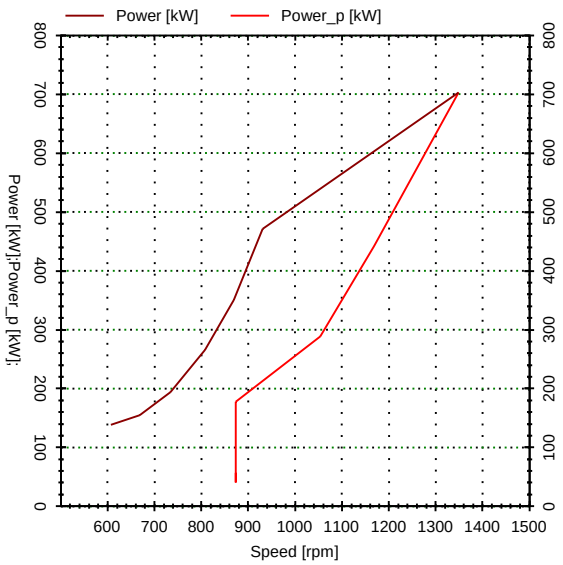
LP [-]	vs [kn]	revs [rpm]	P/D [-]	P_p [kW]
0	0,0	877	-0,206	54
1	1,3	877	-0,027	37
2	2,5	877	0,219	37
3	3,8	877	0,406	44
4	5,0	877	0,566	57
5	6,3	877	0,723	79
6	7,5	877	0,891	118
7	8,8	877	1,062	176
8	10,0	1054	1,020	287
9	11,3	1173	1,062	439
10	12,5	1350	1,067	700

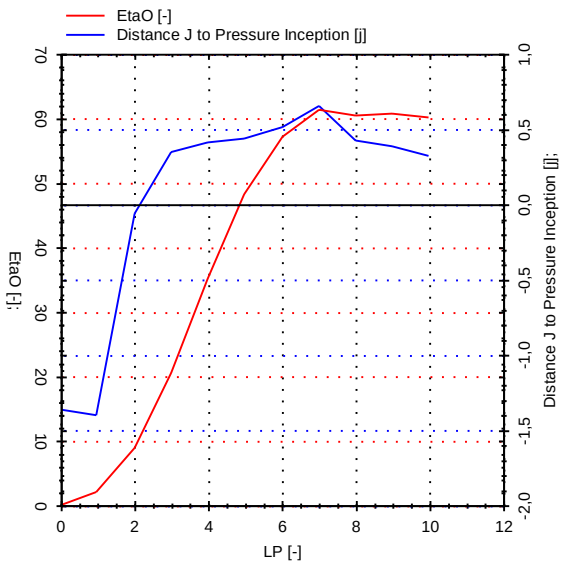
Performance

EtaO [-]	DeltaJ [j]	EtaE [-]	Fuel Co. [ton/hr]
0,000	1,36	0,338	0,01
0,019	1,41	0,323	0,01
0,088	0,06	0,323	0,01
0,203	-0,34	0,330	0,01
0,352	-0,40	0,340	0,01
0,481	-0,44	0,355	0,02
0,572	-0,51	0,373	0,03
0,612	-0,65	0,384	0,04
0,603	-0,42	0,399	0,06
0,606	-0,38	0,400	0,09
0,600	-0,32	0,396	0,15









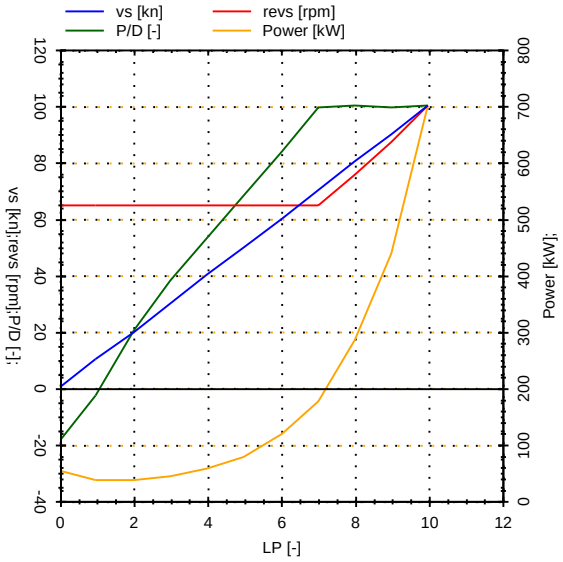
Combinator Result

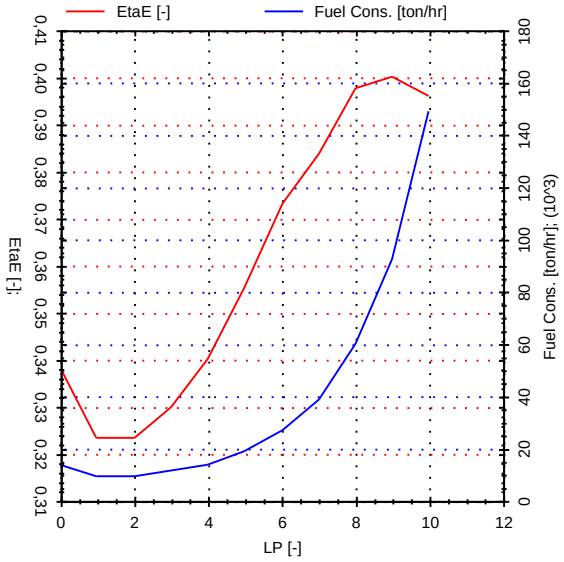
Combinator Settings

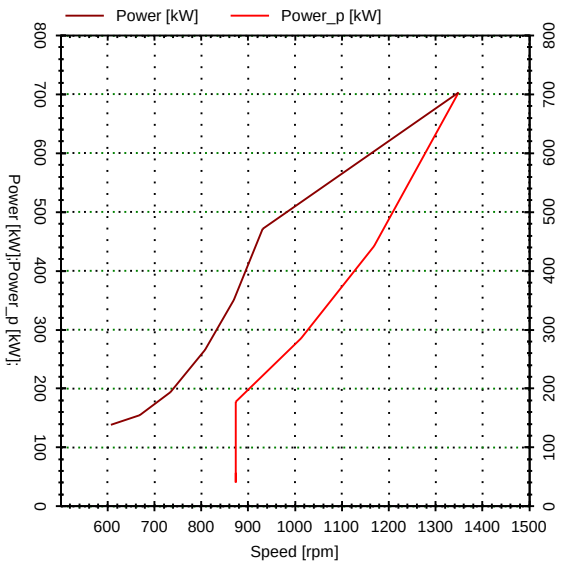
LP [-]	vs [kn]	revs [rpm]	P/D [-]	P_p [kW]
0	0,0	877	-0,205	54
1	1,3	877	-0,031	37
2	2,5	877	0,220	37
3	3,8	877	0,406	44
4	5,0	877	0,566	57
5	6,3	877	0,723	79
6	7,5	877	0,891	118
7	8,8	876	1,063	176
8	10,0	1016	1,067	283
9	11,3	1173	1,062	439
10	12,5	1350	1,067	700

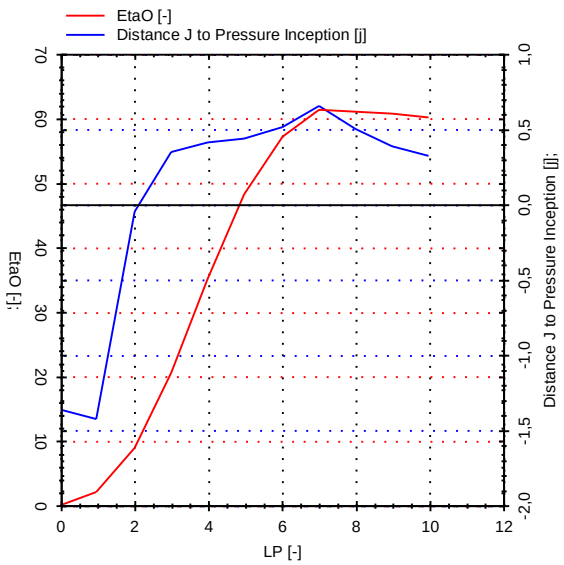
Performance

EtaO [-]	DeltaJ [j]	EtaE [-]	Fuel Cons. [ton/hr]
0,000	1,37	0,338	0,01
0,019	1,42	0,323	0,01
0,088	0,06	0,323	0,01
0,203	-0,34	0,330	0,01
0,352	-0,40	0,340	0,01
0,481	-0,44	0,355	0,02
0,572	-0,51	0,373	0,03
0,612	-0,65	0,384	0,04
0,610	-0,50	0,398	0,06
0,606	-0,38	0,400	0,09
0,600	-0,32	0,396	0,15









Bibliography

[1] M. Altosole, M. Borlenghi, M. Capasso, and M. Figari. Computer-based design tool for a fuel efficient-low emissions marine propulsion plant. *ICMRT Proceedings*, 2007.

[2] L. Birk. *Fundamentals of Ship Hydrodynamics: Fluid Mechanics, Ship Resistance and Propulsion*. John Wiley & Sons, 2019.

[3] K.D.H. Bob-Manuel and B.O. Okim. Optimising technique in matching combined diesel engine or gas turbine (codog) propulsion system to hull and propeller of a frigate. In *14th International Naval Engineering Conference & Exhibition (INEC), Glasgow, UK, 2 – 4 October 2018*, 2018.

[4] C.E. Brennen. *Cavitation and bubble dynamics*. Cambridge University Press, 2014.

[5] D.R. Broome and T.H. Lambert. Interactive computer programme to facilitate diesel engine/propeller matching. *Computers & Graphics*, 6(3):101–108, 1982.

[6] J. Carlton. *Marine propellers and propulsion*. Butterworth-Heinemann, 2018.

[7] A. Coraddu, S. Gaggero, D. Villa, and M. Figari. A new approach in engine-propeller matching. In *14th International Congress of the International Maritime Association of the Mediterranean, IMAM 2011*, pages 631–637, 2012.

[8] R. Curley et al. *The Complete History of Ships and Boats: From Sails and Oars to Nuclear-Powered Vessels*. Britannica Educational Publishing, 2011.

[9] J. Dang, H.J.J. Van den Boom, and J.T. Ligtelijn. The wageningen c-and d-series propellers. In *12th International Conference on Fast Sea Transportation FAST*, 2013.

[10] P. de Vos, D. Stapersma, E. Duchateau, and B. van Oers. Design space exploration for on-board energy distribution systems - a new case study. In *17th Conference on Computer and IT Applications in the Maritime Industries Pavone, 14-16 May 2018*, pages 463–481, 2018.

[11] DNV-GL. Energy transition outlook 2020, 2020. URL https://eto.dnvgl.com/2020/index.html.

[12] R.D. Geertsma. *Autonomous control for adaptive ships with hybrid propulsion and power generation*. Doctoral thesis, 2019. URL https://doi.org/10.4233/uuid:ad81b0ee-76be-4054-a7e8-bd2eeecdb156.

[13] R.D. Geertsma, R.R. Negenborn, K. Visser, and J.J. Hopman. Design and control of hybrid power and propulsion systems for smart ships: A review of developments. *Applied Energy*, 194:30–54, 2017.

[14] R.D. Geertsma, R.R. Negenborn, K. Visser, M.A. Loonstijn, and J.J. Hopman. Pitch control for ships with diesel mechanical and hybrid propulsion: Modelling, validation and performance quantification. *Applied Energy*, 206:1609–1631, 2017.

[15] R.D. Geertsma, K. Visser, and R.R. Negenborn. Adaptive pitch control for ships with diesel mechanical and hybrid propulsion. *Applied Energy*, 228:2490–2509, 2018.

[16] SCHOTTEL GmbH. Homepage. URL https://www.schottel.de/home/.

[17] DFDS Group. Investment in environmental compliance for mediterranean route netwrok. URL https://www.dfds.com/en/about/media/news/investment-in-environmental-compliance-for-mediterranean-route-network.

[18] J.R. Guillemette and P. Bussieres. Proposed optimal controller for the canadian patrol frigate diesel propulsion system. In *The 1997 11 th Ship Control Systems Symposium. Part 1(of 2), Southampton, UK, 04/97*, pages 219–236, 1997.

[19] Habibi. Design of software for calculation and analysis engine propeller matching. *IOP Conference Series: Materials Science and Engineering*, 676:012021, dec 2019.

[20] B. Helmut. *Lehrbuch der Software-Technik: Software-Entwicklung*. 1996.

[21] IMO. Strategy on the reduction of ghg emissions from ships, 2018.

[22] IMO. Prevention of air pollution from ships, 2019. URL http://www.imo.org/en/OurWork/Environment/PollutionPrevention/AirPollution/Pages/Air-Pollution.aspx.

[23] ITTC. *International Towing Tank Conference Recommended Procedures – Fresh Water and Seawater Properties Procedure 7.5-02-01-03, Revision 02*. 2011.

[24] H. Klein Woud and D. Stapersma. Matching propulsion engine with propulsor. *Journal of Marine Engineering & Technology*, 4(2):25–32, 2005.

[25] H. Klein Woud and D. Stapersma. *Design of Propulsion and Electric Power Generating Systems*. IMarEST, 2012.

[26] E.S. Koenhardono, E. B. Djatmiko, A. Soeprijanto, and M.I. Irawan. Neural network-based engine propeller matching (nn-epm) for trimaran patrol ship. In *Applied Mechanics and Materials*, volume 493, pages 388–394. Trans Tech Publ, 2014.

[27] S. Lin, J. Sun, and D. Xie. Design of ship-engine-propeller simultaneous matching and development of a propeller and engine selecting system. 2019.

[28] A. Liu and S. Fan. The research on the matching design of the ship-engine-propeller based on multi-objective particle swarm optimization. In *2010 2nd International Workshop on Intelligent Systems and Applications*, pages 1–4. IEEE, 2010.

[29] D MacPherson. Electric motor selection for underwater vehicles: Considerations of partial load efficiency. 2020.

[30] C.H. Marques, C.R.P. Belchior, and J.D. Caprace. Optimising the engine-propeller matching for a liquefied natural gas carrier under rough weather. *Applied Energy*, 232:187–196, 2018.

[31] O. B. Ogar, S. Nitonye, I. John-Hope, et al. Design analysis and optimal matching of a controllable pitch propeller to the hull and diesel engine of a codog system. *Journal of Power and Energy Engineering*, 6 (03):53, 2018.

[32] A. Papanikolaou. Holistic ship design optimization. *Computer-Aided Design*, 42(11):1028–1044, 2010.

[33] H. Ren, Y. Ding, and C. Sui. Influence of eedi (energy efficiency design index) on ship–engine–propeller matching. *Journal of Marine Science and Engineering*, 7(12):425, 2019.

[34] L. Ren and Y. Diao. Matching optimization of ship engine propeller and net for the trawler based on genetic algorithm. In *2014 10th International Conference on Natural Computation (ICNC)*, pages 617–621. IEEE, 2014.

[35] V.M.M. Sáiz and A.P. López. Future trends in electric propulsion systems for commercial vessels. *Journal of Maritime Research*, 4(2):81–100, 2007.

[36] P.J.M. Schulten. The interaction between diesel engines, ship and propellers during manoeuvring. 2005.

[37] C. Sheppard. *World Seas: An Environmental Evaluation: Volume III: Ecological Issues and Environmental Impacts*. Academic Press, 2018.

[38] W. Shi, H.T. Grimmelius, and D. Stapersma. Analysis of ship propulsion system behaviour and the impact on fuel consumption. *International shipbuilding progress*, 57(1-2):35–64, 2010.

[39] A. Sobey, J. Blanchard, P. Grudniewski, and T. Savasta. There's no free lunch: A study of genetic algorithm use in maritime applications. In *18th Conference on Computer and IT Applications in the Maritime Industries Tullamore, 25-27 March 2019*, pages 375–390, 2019.

[40] F. Soto, G. Marques, E. Torres-Jiménez, B. Vieira, A. Lacerda, O. Armas, and F. Guerrero-Villar. A comparative study of performance and regulated emissions in a medium-duty diesel engine fueled with sugarcane diesel-farnesane and sugarcane biodiesel-ls9. *Energy*, 176:392–409, 2019.

[41] D. Stapersma. Modellering brandstofverbruik in deellast, diesel motoren en gasturbines, 2009.

[42] D Stapersma. Diesel engines volume 4: Emission and heat transfer. *Lecture notes, TU Delft*, 2010.

[43] T. Tillack. Optimierung der fahrkurven, 2009.

[44] C. Trozzi. Emission estimate methodology for maritime navigation. *Techne Consulting, Rome*, 2010.

[45] MAN Diesel & Turbo. Diesel electric propulsion plants: A brief guideline how to design a diesel-electric propulsion plant. 2017.

[46] P.J. Van Spronsen and R.L. Tousain. An optimal control approach to preventing marine diesel engine overloading aboard karel doorman class frigates. *IFAC Proceedings Volumes*, 34(7):493–498, 2001.

[47] H.P.M. Veeke, J.A. Ottjes, and G. Lodewijks. *The Delft systems approach: Analysis and design of industrial systems*. Springer Science & Business Media, 2008.

[48] R. Vie. Commercial experience with electric propulsion on passenger cruise vessels. *TRANSACTIONS-INSTITUTE OF MARINE ENGINEERS-SERIES C-*, 110:1–10, 1998.

[49] A. Vrijdag. *Control of Propeller cavitation in operational conditions*. Doctoral thesis, 2009. URL http://resolver.tudelft.nl/uuid:2a17faba-f033-439a-9970-dfdce4a1fcdf.

[50] A. Vrijdag, D. Stapersma, and T. van Terwisga. Tradeoffs in ship propulsion control: engine overloading and cavitation inception in operational conditions. In *Proceedings of the 9th International Naval Conference and Exhibition, Hamburg, Germany, INEC'08, Report 1584-P, TUDelft, Ship Hydromechanics Laboratory*, 2008.

[51] A. Vrijdag, D. Stapersma, and T. Van Terwisga. Systematic modelling, verification, calibration and validation of a ship propulsion simulation model. *Journal of Marine Engineering & Technology*, 8(3):3–20, 2009.